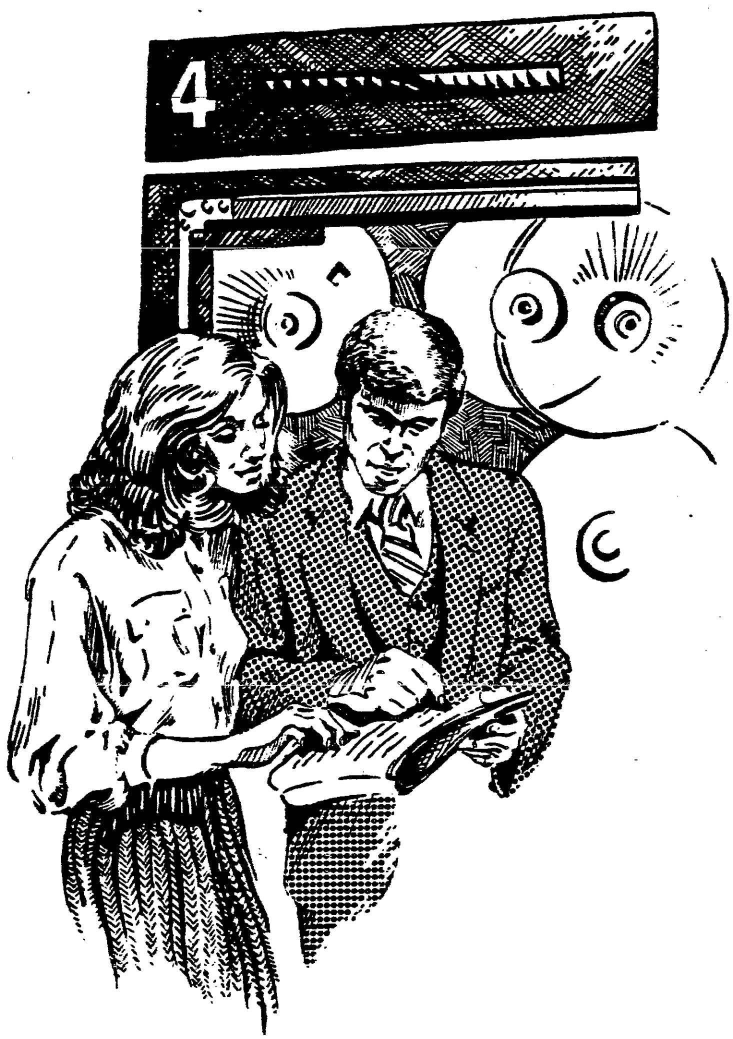
HONEYWELL

LARGE SYSTEMS EDUCATION



MULTICS VIRTUAL MEMORY ANALYSIS AND METERING

STUDENT HANDBOOK

COURSE CODE F80A

MULTICS VIRTUAL MEMORY ANALYSIS AND METERING

STUDENT HANDBOOK

COURSE CODE F80A

ISSUE DATE: June 1978

REVISION: 3

REVISION DATE: March, 1983

COURSE DESCRIPTION

F80A Multics Virtual Memory Analysis and Metering

Duration: Five Days

Intended For: Personnel requiring analysis-level knowledge of Multics virtual memory implementation, metering and tuning. Especially useful for personnel concerned with maximizing system performance..

Synopsis: The Multics supervisor is logically divided into seven distinct subsystems. This course details the functions performed (and the data bases maintained) by five of these subsystems: Volume Management, Name and Address Space Management, Directory Control, Segment Control, and Page Control. Knowledge of the virtual memory implementation gives the student insight into the metering and tuning of the system. Other topics include overviews of the Multics system hardware, the Multics Supervisor, and a comparison with other memory management techniques. Question and answer periods are given daily to reinforce the material presented.

Objectives: Upon completion of this course, the student should be able to:

1. Understand the functions of the Multics supervisor subsystems, especially those subsystems which implement the Multics virtual memory.
2. Make optimal design choices when writing system applications to run in the Multics environment.
3. Evaluate and tune the system's performance by analyzing the system's virtual memory meters.

Prerequisites: Multics Subsystem Programming (F15D), Source Level Debugging & The Process Environment (F21) or equivalent experience.

Major Topics: System Hardware Components
Volume (Disk) Management
Name Space/Address Space Management
Directory, Segment and Page Control
Memory Management Techniques

Course Topic Map

to be

Inserted Here

CONTENTS

CONTENTS (con't)

CONTENTS (con't)

THIS PAGE INTENTIONALLY LEFT BLANK

TOPIC I

Multics Design Philosophy

Page

MAJOR DESIGN GOALS

- MULTIPLEXED INFORMATION AND COMPUTING SERVICE (MULTICS)

- MULTICS WAS ONE OF THE FIRST OPERATING SYSTEMS TO BE THOROUGHLY DESIGNED FROM THE TOP DOWN. THE DEVELOPERS:

  - STARTED WITH A SET OF GOALS

  - CREATED A SYSTEM WHICH WOULD SATISFY THESE GOALS

  - DEVELOPED GENERAL SOLUTIONS INSTEAD OF SPECIFIC SOLUTIONS (MAKING THE PRODUCT EXTENDABLE)

  - PRODUCED A VIABLE AND MARKETABLE PRODUCT

MAJOR DESIGN GOALS

- THESE GOALS WERE CAREFULLY CHOSEN TO CHARACTERIZE A 'UTILITY-GRADE' COMPUTER SYSTEM, AND ARE OUTLINED BELOW:

  - VIRTUAL MEMORY ORGANIZATION
  - SELECTIVE, CONTROLLED SHARING
  - SECURITY
  - OPEN-ENDED, MODULAR SYSTEM
  - DECENTRALIZED ADMINISTRATION
  - FLEXIBLE USER INTERFACES, END-USER ORIENTATION
  - CONTINUOUS OPERATION
  - RELIABLE FILE SYSTEM
  - REMOTE ACCESS
  - EFFICIENT SERVICE TO LARGE AND SMALL USER

VIRTUAL MEMORY ORGANIZATION

- MOTIVATION

  - INFORMATION STORED ON-LINE IN LARGE INFORMATION UTILITIES OFTEN EXCEEDS THE SIZE OF AVAILABLE MAIN MEMORY

  - THIS INFORMATION SHOULD BE DIRECTLY (AND CONTINUOUSLY) ACCESSIBLE BY THE USER COMMUNITY

  - THE SIZE OF MAIN MEMORY SHOULD ONLY AFFECT PROCESSING TIME, NOT PROCESSING CAPABILITY

  - MAIN MEMORY MANAGEMENT SHOULD BE A TASK FOR THE OPERATING SYSTEM, NOT THE PROGRAMMER

- IMPLEMENTATION

  - ALL ON-LINE INFORMATION IS <u>PROCESSOR ADDRESSABLE</u>

  - <u>ALL</u> INFORMATION (PROCEDURE AND DATA) IS COMPARTMENTALIZED INTO UNITS CALLED "SEGMENTS" ALLOWING THE ASSOCIATION OF ATTRIBUTES WITH EACH SEGMENT(1)

---

(1) THROUGHOUT THIS DOCUMENT, GENERIC REFERENCES TO "SEGMENTS" INCLUDE DIRECTORIES AS WELL, SINCE DIRECTORIES ARE SIMPLY A SPECIAL KIND OF SEGMENT

VIRTUAL MEMORY ORGANIZATION

- SEGMENTS ARE MADE PROCESS ADDRESSABLE AS THEY ARE REFERENCED

- ALL SEGMENTS ARE DIVIDED INTO AN INTEGRAL NUMBER OF 1024 WORD PAGES. THESE PAGES ARE BROUGHT INTO MAIN MEMORY IF AND ONLY IF THEY ARE REFERENCED (NEEDED)-AT THE TIME THEY ARE REFERENCED BY ANY PROCESS

- THE MULTICS HARDWARE INTERPRETS ALL ADDRESSES AS OFFSETS WITHIN A SPECIFIED SEGMENT (SEGNO|OFFSET)

- THE HARDWARE MAKES NO DISTINCTION BETWEEN PROCEDURE AND DATA SEGMENTS. BOTH ARE PAGED IN THE SAME MANNER, BOTH ARE ADDRESSED IN THE SAME MANNER

- ALL COMPILERS PRODUCE LOAD MODULES - NO MODIFICATION IS REQUIRED TO EXECUTE PROCEDURE CODE

SELECTIVE, CONTROLLED SHARING

- MOTIVATION

  - USERS SHOULD BE ABLE TO USE COMMON PROCEDURE AND DATA SEGMENTS DIRECTLY (NOT COPIES)

  - USERS SHOULD BE ABLE TO SHARE PRIVATE CODE IN A SELECTIVE MANNER

- IMPLEMENTATION

  - PURE, REENTRANT CODE IS ALWAYS GENERATED BY THE COMPILERS (ALLOWING SHARING OF PROCEDURE CODE IN A MULTI-PROCESS ENVIRONMENT)

  - EVERYTHING THE USER TOUCHES (EXECUTE OR REFERENCE) WILL BE A SEGMENT HAVING ITS OWN ATTRIBUTES

  - THE ACCESS ATTRIBUTES OF EACH SEGMENT ARE ESTABLISHED BY THE OWNER OF THAT SEGMENT

SECURITY

- MOTIVATION

  - IN AN ENVIRONMENT OF SEVERAL COEXISTING PROCESSES, USERS MUST BE PROTECTED FROM ACCIDENTALLY OR INTENTIONALLY INTERFERING WITH EACH OTHER

  - THE SUPERVISOR MUST BE PROTECTED FROM DAMAGE BY USERS

  - CHANGES IN ACCESS TO INFORMATION MUST BE IMMEDIATELY EFFECTIVE

  - DISCLOSURE OF INFORMATION SHOULD BE ALLOWED IN A SELECTIVE AND CONTROLLED MANNER (VERSUS ALL-OR-NONE APPROACH)

  - UNWARRANTED DENIAL OF ACCESS TO INFORMATION MUST BE PROHIBITED

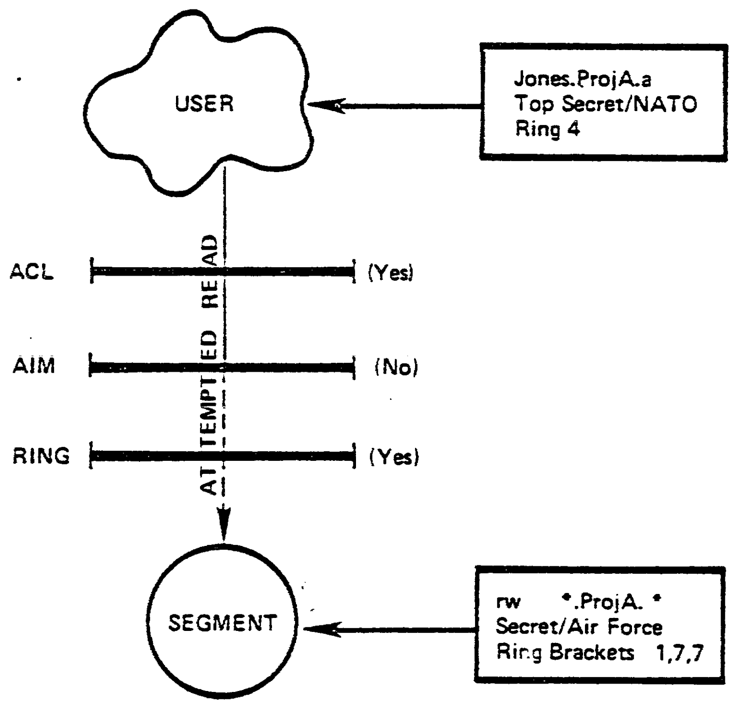
- IMPLEMENTATION

  - PER-SEGMENT ACCESS CONTROL LIST (ACL) - SPECIFYING BY WHOM AND HOW THE SEGMENT MAY BE ACCESSED

  - RING PROTECTION MECHANISM - ISOLATES SEGMENTS AND PROCESSES

  - ACCESS ISOLATION MECHANISM (AIM) - ISOLATES SEGMENTS ACCORDING TO CATEGORIES AND SECURITY LEVELS

  - PASSWORDS AND AUDIT TRAILS


USER
Jones.ProjA.a
Top Secret/NATO
Ring 4
ACL
(Yes)
AIM
(No)
RING
(Yes)
ATTEMPTED READ
SEGMENT
rw *.ProjA.*
Secret/Air Force
Ring Brackets 1,7,7

OPEN-ENDED, MODULAR SYSTEM

- MOTIVATION

  - SOFTWARE SHOULD BE EASY TO MODIFY AND EXTEND

  - THE OPERATING SYSTEM SHOULD BE MODULAR, AND THE MODULES SHOULD BE COMPREHENSIBLE

- IMPLEMENTATION

  - MODULAR DESIGN OF OPERATING SYSTEM AND USER PROGRAMS (COMPILERS, PROGRAMMING ENVIRONMENT ENCOURAGE MODULAR DESIGN)

  - UNIFORM PROGRAMMING CONVENTIONS ARE FOLLOWED THROUGHOUT MOST SYSTEM CODE

  - MORE THAN 92% OF THE OPERATING SYSTEM OBJECT CODE ORIGINATED FROM PL/I SOURCE

  - DYNAMIC LINKING (ELIMINATES RE-COMPILING, RE-EDITING WHEN UN-BOUND MODULES ARE REPLACED)

  - ON-LINE MODIFICATION, TESTING AND INSTALLATION OF SYSTEM MODULES (MULTICS INSTALLATION FACILITY - MIS)

  - PATCH-FREE SYSTEM

  - LIBRARY MANAGEMENT TOOLS AND LIBRARY CONVENTIONS

DECENTRALIZED ADMINISTRATION

- MOTIVATION

  - SYSTEM RESOURCES MUST BE EFFECTIVELY ADMINISTERED

  - RESOURCE ALLOCATION, ACCOUNTING, REGISTRATION, BILLING, ETC, IS TOO MUCH FOR SINGLE INDIVIDUAL

  - DIFFERENT GROUPS OF USERS HAVE DIFFERENT ADMINISTRATIVE NEEDS

- IMPLEMENTATION

  - GROUPING OF USERS, BY FUNCTION OR MANAGEMENT, INTO PROJECTS

  - THREE-LEVEL HIERARCHY OF ADMINISTRATION

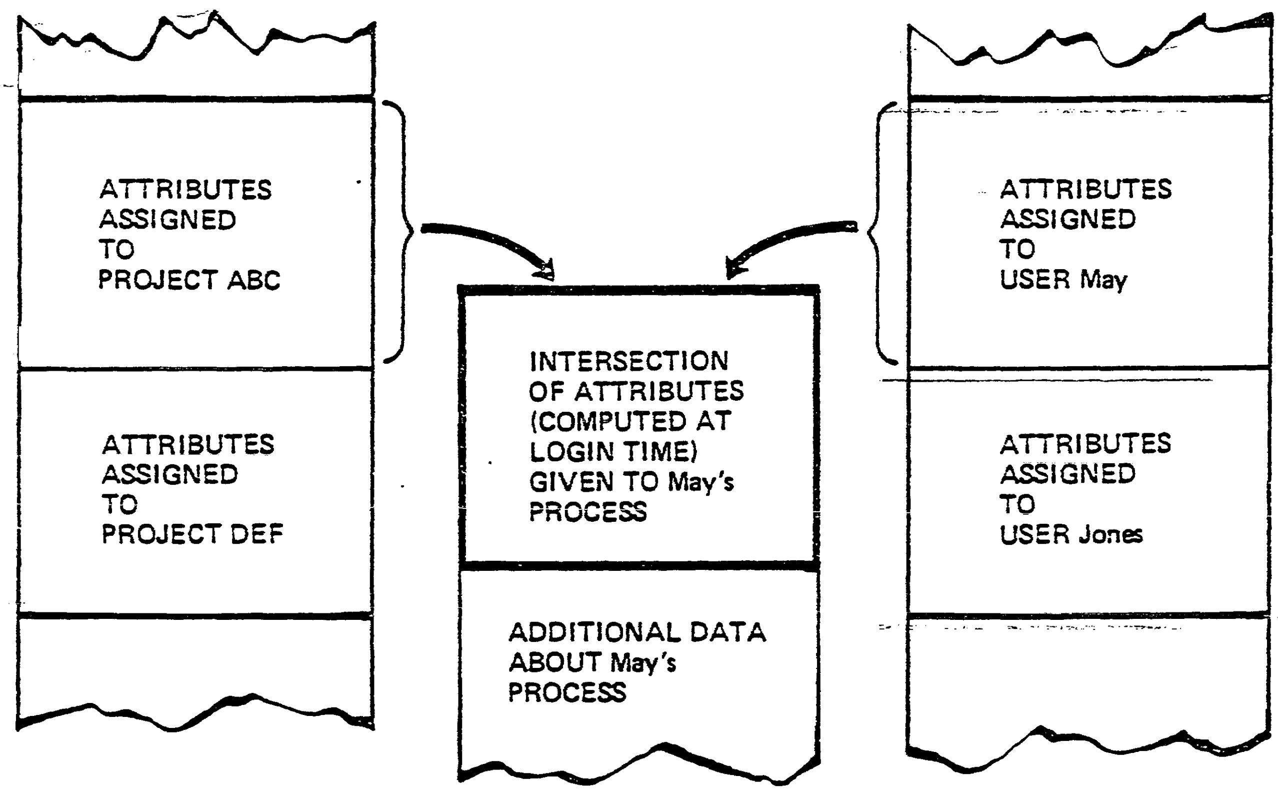
    - SYSTEM ADMINISTRATOR: DISTRIBUTES RESOURCES AND ASSIGNS ATTRIBUTES TO PROJECTS

    - PROJECT ADMINISTRATORS: DISTRIBUTES RESOURCES AND ASSIGNS ATTRIBUTES TO USERS

    - USERS: HAS FULL CONTROL OVER ALLOCATED RESOURCES, MODIFIED BY ASSIGNED ATTRIBUTES

  - THE PROJECT ADMINISTRATOR MAY PASS DOWNWARD ONLY THOSE RESOURCES AND ATTRIBUTES THAT HAVE BEEN GIVEN TO THE PROJECT

PROCESS ATTRIBUTES



SYSTEM ADMINISTRATION TABLE
(>sc1>sat)

CREATED AND MAINTAINED
BY SYSTEM ADMINISTRATOR

PROCESS INITIALIZATION TABLE
([pd]>pit)

ANSWERING SERVICE
TABLE ENTRY
(ANSWER_TABLE,
ABSENTEE_USER_TABLE,
DAEMON_USER_TABLE)

CREATED AT LOGIN
DESTROYED AT LOGOUT

PROJECT DEFINITION TABLE
(>SC1>pdt>ABC.pdt)

CREATED AND MAINTAINED
BY PROJECT ADMINISTRATOR

FLEXIBLE USER INTERFACES

- MOTIVATION

  - THE STANDARD USER ENVIRONMENT SHOULD BE EXTENSIVELY USER-MODIFIABLE

  - THE CAPABILITY SHOULD EXIST TO DEVELOP AND IMPOSE CLOSED SUBSYSTEMS WHICH CAN PROVIDE ANY DESIRED ENVIRONMENT

- IMPLEMENTATION

  - USER HAS ABILITY TO CHANGE OR REPLACE CONTROL PROGRAMS IN THE USER'S RING

  - PROJECT ADMINISTRATOR CAN IMPOSE A CLOSED SUBSYSTEM ENVIRONMENT OR A DIFFERENT process_overseer ON USERS

  - start_up.ec, ABBREV PROCESSOR, general_ready, ready_off, add_search_rules, CONDITION HANDLING, ETC.

  - OTHER TOOLS PROVIDE SIMULATION, ENCAPSULATION CAPABILITY (enter_lss, project_start_up_)

  - STANDARD INTERFACE FOR INTERACTIVE SUBSYSTEMS (ssu_) ENCOURAGES UNIFORM, FAMILIAR BEHAVIOUR OF USER SUBSYSTEMS.

CONTINUOUS OPERATION

- MOTIVATION

  - UTILITY CONCEPT: SYSTEM SHOULD BE AVAILABLE ON DEMAND AT ALL TIMES

- IMPLEMENTATION

  - ON-LINE SOFTWARE INSTALLATION

  - ON-LINE MAINTENANCE: MOVE MORE AND MORE BOS CAPABILITY INTO MULTICS (EG: RE-BOOT FNP FROM MULTICS)

  - ON-LINE FILE BACKUP AND RECOVERY

  - ON-LINE ACCOUNTING AND BILLING

  - DYNAMIC RECONFIGURATION

  - DYNAMIC FAILSOFT DECONFIGURATION OF FAILING HARDWARE

  - UNATTENDED SERVICE

  - AUTOMATIC REBOOT

RELIABLE FILE SYSTEM

- MOTIVATION

  - MUST PROVIDE USERS SOME ASSURANCE THAT THEIR ON-LINE INFORMATION IS SAFE

  - MUST PROVIDE CHECKPOINT CAPABILITY FOR RECOVERY FROM USER ERROR OR SYSTEM DISASTER

- IMPLEMENTATION

  - AUTOMATIC BACKUP/RETRIEVAL FACILITY

  - CONSOLIDATED STORAGE SYSTEM DUMPS

  - PHYSICAL AND LOGICAL SAVE/RESTORE

  - ALL STORAGE SYSTEM RECOVERY PROCEDURES RUN WHILE SYSTEM IS UP

  - DAMAGE RECOVERY RUN AUTOMATICALLY FOLLOWING SYSTEM FAILURE

REMOTE ACCESS

- MOTIVATION

  - UTILITY CONCEPT: FULL ACCESS FROM ANY PHONE IN THE WORLD VIA ANY REMOTE DEVICE

  - WISH TO PROVIDE ONE "COMMAND LANGUAGE", TO SERVE ALL USERS, WHETHER LOCAL OR REMOTE, INTERACTIVE OR BATCH

- IMPLEMENTATION

  - MULTICS COMMUNICATION SYSTEM (MCS)

  - IN PRINCIPLE, ANY REMOTE DEVICE/TERMINAL IS CONNECTABLE

  - SINGLE COMMAND LANGUAGE

  - REMOTE JOB ENTRY (RJE) AND BULK I/O CAPABILITIES

  - DIRECT ATTACHMENTS TO PUBLIC DATA NETWORKS VIA X.25

EFFICIENT SERVICE TO LARGE OR SMALL USERS

- MOTIVATION

  - UTILITY CONCEPT: SYSTEM SHOULD BE AVAILABLE FOR, AND CAPABLE OF, ANY SIZE TASK

  - RUNNING BOTH LARGE AND SMALL TASKS TOGETHER SHOULD NOT IMPACT THE EFFICIENCY OF EITHER

- IMPLEMENTATION

  - DYNAMIC RESOURCE ALLOCATION (DON'T HAVE TO PRE-ALLOCATE OR GUESS-TIMATE RESOURCES REQUIRED)

  - SERVICE ON DEMAND

  - DYNAMIC SYSTEM TUNING TO ACCOMMODATE CHANGING SYSTEM WORKLOADS

STRUCTURE OF THE OPERATING SYSTEM

- SUBROUTINES (550)

  - DESCRIBED IN THE MPM MANUALS "Multics Subroutines" (AG93) AND "Subsystems Writer's Guide" (OFTEN ABBREVIATED AS "SWG") (AK92)

- TOOLS (220)

  - DESCRIBED IN THE MPM MANUAL "Multics Commands and Active Functions" (AG92)

- ADMINISTRATIVE ROUTINES (200)

  - DESCRIBED IN THE MAM MANUALS "System Administrator" (AK50), "Registration & Accounting Administrator" (AS68), "Project Administrator" (AK51)

- OPERATOR COMMANDS (150)

  - DESCRIBED IN THE MANUAL "Operator's Handbook" (AM81)

- WHAT IS THE MULTICS SUPERVISOR?

  - A COLLECTION OF MANY LOGICAL SUBSYSTEMS WHICH IMPLEMENT THE FUNCTIONS OF MULTICS

  - THE PRIMARY PURPOSE OF MULTICS IS TO RUN PROGRAMS, WHICH ACCESS DATA, AND THUS THE MAJOR PURPOSE OF THE MULTICS SUPERVISOR IS TO MAKE THAT DATA ACCESSIBLE

  - THESE SUBSYSTEMS FALL INTO FOUR MAJOR GROUPS:

    - THE FILE SYSTEM

    - SUPPORT SERVICES FOR THE FILE SYSTEM

    - MISCELLANEOUS SUPERVISOR SERVICES

    - SUBSYSTEMS RELATED TO, BUT NOT STRICTLY PART OF THE SUPERVISOR

  - THESE DIVISIONS ARE SOMEWHAT ARTIFICIAL, BECAUSE THE SUBSYSTEMS ARE ALL INTIMATELY RELATED TO EACH OTHER. THE DIVISIONS REPRESENT A PARTICULAR VIEWPOINT OF SYSTEM FUNCTION.

  - A MULTICS SUBSYSTEM IS A SET OF PROGRAMS PERFORMING A SPECIFIC SERVICE FOR THE USER COMMUNITY - AND FOR THE OPERATING SYSTEM ITSELF

  - TOGETHER, ALL THESE SUBSYSTEMS IMPLEMENT THE FUNCTIONS DESCRIBED IN THE MPM SUBROUTINES AND SWG MANUALS, (ESSENTIALLY hcs_ AND THE VIRTUALS MEMORY).

THE MAJOR SUPERVISOR SUBSYSTEMS

- MAJOR MULTICS SUPERVISOR SUBSYSTEMS: FOUR GROUPS OF ABOUT FOUR SUBSYSTEMS EACH

  - THE FILE SYSTEM - THOSE SUBSYSTEMS WHICH ARE CONCERNED WITH STORING DATA, MANAGING DATA, AND MAKING IT AVAILABLE TO USERS. FIVE MAJOR COMPONENTS:

    - NAME SPACE / ADDRESS SPACE CONTROL

    - DIRECTORY CONTROL

    - VOLUME MANAGEMENT

    - SEGMENT CONTROL

    - PAGE CONTROL

  - SERVICES TO SUPPORT THE FILE SYSTEM, WHICH MULTIPLEX ITS FACILITIES BETWEEN DIFFERENT USERS, AND ENSURE ITS RELIABILITY. FOUR MAJOR COMPONENTS:

    - TRAFFIC CONTROL

    - FAULT AND INTERRUPT HANDLING

    - SYSTEM INITIALIZATION

    - THE FILE SYSTEM SALVAGERS

- MISCELLANEOUS SUPERVISOR SERVICES - THESE ARE THINGS DONE IN THE SUPERVISOR FOR REASONS OF ACCESS CONTROL AND SHARING, BUT NOT DIRECTLY RELATED TO THE FILE SYSTEM

- BECAUSE THEY ARE NOT DIRECTLY RELATED, THEY WILL NOT BE COVERED IN ANY DETAIL

  - MULTICS COMMUNICATIONS SYSTEM

  - RESOURCE CONTROL

  - USER DEVICE I/O - ioi_

  - LOW LEVEL SUPERVISOR I/O

  - RECONFIGURATION

  - SYSTEM ERROR HANDLING (syserr / verify_lock)

- RELATED SUBSYSTEMS - THESE ARE NOT ACTUALLY PART OF THE SUPERVISOR, BUT ARE CLOSELY RELATED

  - METERING AND TUNING

  - THE Initializer.SysDaemon

- THE MULTICS SUPERVISOR IS DESIGNED AROUND THE "LAYERED MACHINE" CONCEPT

THE MAJOR SUPERVISOR SUBSYSTEMS

- CONSTRUCT A SIMPLE SET OF OPERATIONS CALLED A "KERNEL" WHICH IMPLEMENTS THE MOST FUNDAMENTAL (PRIMITIVE) OPERATIONS REQUIRED

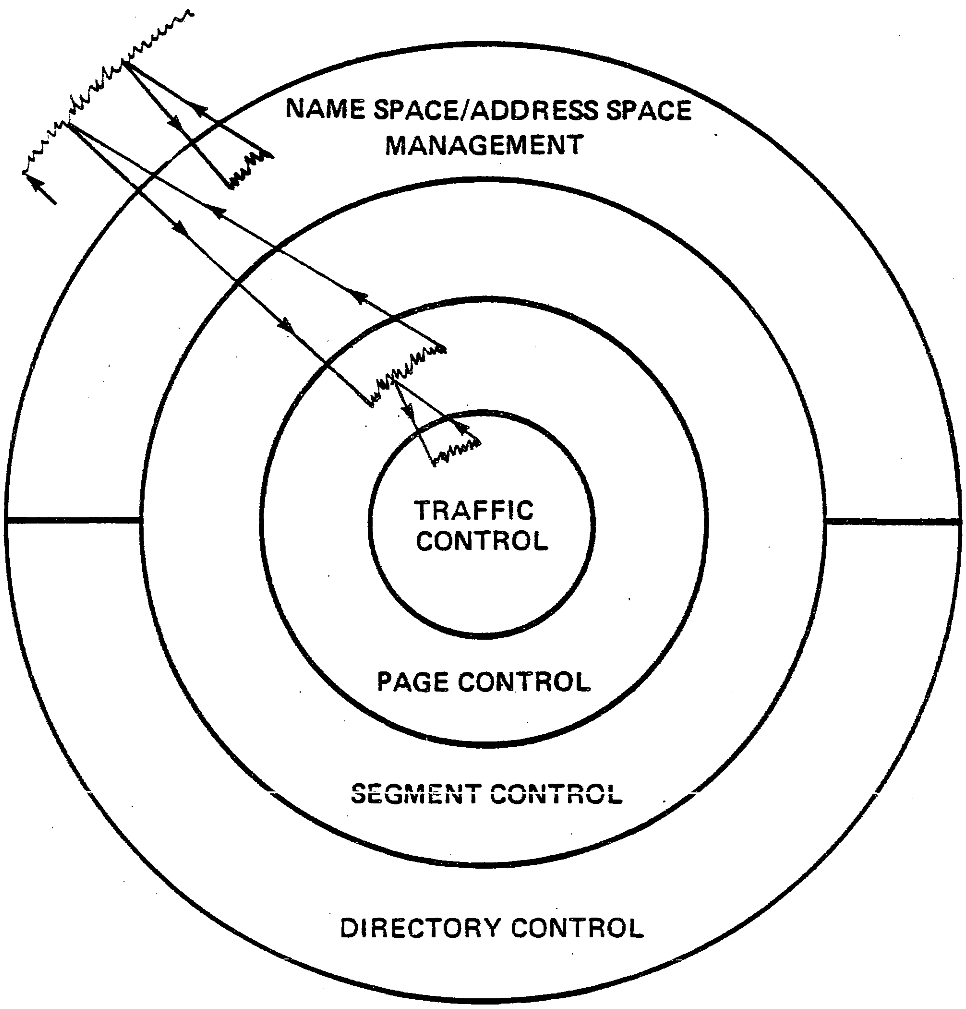
- CONSTRUCT A SLIGHTLY MORE SOPHISTICATED SET OF OPERATIONS WHICH ASSUMES AND RELIES ON THE CORRECT FUNCTIONING OF THE KERNEL - ANOTHER "LAYER"

- CONSTRUCT A MORE SOPHISTICATED LAYER WHICH ASSUMES AND RELIES ON THE CORRECT FUNCTIONING OF THE PREVIOUS MACHINES

- ETC

- THE "LAYERS" OF THE MULTICS SUPERVISOR PARTIALLY MAP INTO THE ABOVE SUBSYSTEMS

- THE FOLLOWING DIAGRAM REPRESENTS THIS MAPPING:



THE MULTICS SUPERVISOR

COMPONENTS ARE ASYNCHRONOUSLY INVOKED

NAME SPACE/ADDRESS SPACE MANAGEMENT

- FUNCTION

  - IMPLEMENT THE PER PROCESS VIRTUAL MEMORY

- BASIC PHILOSOPHY

  - AS A NEWLY LOGGED IN USER ATTEMPTS TO TOUCH VARIOUS SEGMENTS A CONSIDERABLE AMOUNT OF MANAGEMENT INFORMATION MUST BE (TRANSPARENTLY) FOUND AND/OR COMPUTED BEFORE THE USER'S REFERENCE IS ACTUALLY ACCOMPLISHED

  - FOR EVERY SEGMENT REFERENCED BY THE USER, THE SUPERVISOR:

    - ASSIGNS A SEGMENT NUMBER (FOR REASON OF HARDWARE ADDRESSING), AND

    - RECORDS (REMEMBERS) THE MANAGEMENT INFORMATION (FOR REASON OF SOFTWARE EFFICIENCY AND CONTROL)

  - SUCH SEGMENTS ARE SAID TO BE "KNOWN TO THE PROCESS"

  - THE MANAGEMENT INFORMATION IS MAINTAINED ON A PER PROCESS BASIS IN THREE COMPLEMENTING AREAS: DSEG, KST, AND RNT

NAME SPACE/ADDRESS SPACE MANAGEMENT

- MANAGES TWO DISTINCT SETS OF INFORMATION:

  - ADDRESS SPACE - CORRESPONDENCE BETWEEN SEGMENT NUMBERS AND THE SEGMENTS THEMSELVES

  - NAME SPACE - CORRESPONDENCE BETWEEN SEGMENT NUMBERS AND NAMES THE USER REFERS TO THEM BY

- CALLS DIRECTORY CONTROL TO LOCATE SEGMENTS INITIALLY

- NAME SPACE / ADDRESS SPACE MANAGEMENT IS INVOKED BY SUBROUTINE CALLS, AND BY LINKAGE FAULTS (THE "DYNAMIC LINKER")

- PRINCIPAL USER INTERFACES

  - COMMAND LEVEL

    - initiate, terminate, terminate_segno, terminate_ref_name, terminate_single_ref_name, list_ref_name

    - THE COMMAND PROCESSOR ITSELF - WHICH USES THESE SERVICES TO LOCATE COMMANDS

  - SUBROUTINE LEVEL

    - hcs_$initiate, hcs_$initiate_count_, hcs_$terminate_file, hcs_$terminate_seg, hcs_$terminate_name, hcs_$terminate_noname, term_

NAME SPACE/ADDRESS SPACE MANAGEMENT

- MAJOR DATA BASES

  - DESCRIPTOR SEGMENT (DSEG) - ONE PER PROCESS

    - SEGMENT DESCRIPTOR WORD (SDW) - ONE PER KNOWN SEGMENT

    - DEFINES THE USER'S ADDRESS SPACE TO THE HARDWARE

  - KNOWN SEGMENT TABLE (KST) - ONE PER PROCESS

    - KNOWN SEGMENT TABLE ENTRY (KSTE) - ONE PER KNOWN SEGMENT (EXCEPT SUPERVISOR SEGMENTS)

    - DEFINES THE USER'S ADDRESS SPACE TO THE SUPERVISOR AND THE USER

    - EACH KSTE ASSOCIATES A USER'S SEGMENT NUMBER WITH THE SEGMENT CONTROL ATTRIBUTES OF THAT SEGMENT

    - THE SEARCH FOR AN AVAILABLE KSTE DETERMINES A SEGMENT'S NUMBER

  - REFERENCE NAME TABLE (RNT) - ONE PER EACH RING IN EACH PROCESS

    - NOT A SEGMENT - KEPT AS A REGION ALLOCATED IN THE "LINKAGE AREA" FOR EACH RING

    - REFERENCE NAME TABLE ENTRY (RNTE) - ONE PER REFERENCE NAME

    - USED BY THE DYNAMIC LINKER TO IMPLEMENT THE "initiated_segments" SEARCH RULE

NAME SPACE/ADDRESS SPACE MANAGEMENT

- DEFINES THE USER'S NAME SPACE TO THE USER

- NAME SPACE MAY BE DIFFERENT IN DIFFERENT RINGS OF THE SAME PROCESS

DIRECTORY CONTROL

- FUNCTION

  - DIRECTORY CONTROL IS A SET OF HARDCORE MODULES RESPONSIBLE FOR THE MAINTENANCE OF THE MULTICS DIRECTORY STRUCTURE -- IE: THE HIERARCHY

  - ITS TASKS INCLUDE CREATING, MANIPULATING AND INTERPRETING THE CONTENTS OF DIRECTORY SEGMENTS, TO INCLUDE:

    - ACCESS CONTROL LISTS (ACL'S), NAMES, AND VTOCE POINTERS OF ENTRIES DESCRIBED THEREIN

  - ONLY DIRECTORY CONTROL IS ALLOWED TO ALTER THE CONTENTS OF DIRECTORY SEGMENTS

  - DIRECTORY CONTROL IMPLICITLY RELIES UPON THE SERVICES OF OTHER SUBSYSTEMS SUCH AS SEGMENT CONTROL AND PAGE CONTROL, AND ALSO INVOKES THEM DIRECTLY BY SUBROUTINE CALL

    - DIRECTORIES ARE SIMPLY SEGMENTS TO THESE SUBSYSTEMS

  - DIRECTORY CONTROL IS INVOKED ONLY BY SUBROUTINE CALLS

- PRINCIPAL USER INTERFACES

  - COMMAND LEVEL

  - create, create_dir, link, set_acl, delete_acl, status, list, add_name, rename

- SUBROUTINE LEVEL

  - hcs_$append_branch, hcs_$add_acl_entries, hcs_$append_link, hcs_$delete_acl_entries, hcs_$status_, hcs_$chname_file

- MAJOR DATA BASES

  - DIRECTORY SEGMENTS

    - CONTAIN THE ATTRIBUTES AND OTHER INFORMATION ABOUT THEIR SEGMENTS (NEEDED TO FIND SEGMENTS, RETURN STATUS INFORMATION, AND BUILD VTOCE'S AT SEGMENT CREATION)

  - THE DIRLOCKT_SEG

    - SEGMENT WHERE DIRECTORY LOCKING IS MANAGED

VOLUME MANAGEMENT

- FUNCTION

  - VOLUME MANAGEMENT IS RESPONSIBLE FOR THE MANAGEMENT OF PHYSICAL AND LOGICAL VOLUMES

  - ITS TASKS INCLUDE:

    - ACCEPTANCE AND DEMOUNTING OF PHYSICAL VOLUMES

    - MAINTAINING THE ASSOCIATION BETWEEN PHYSICAL VOLUMES, LOGICAL VOLUMES, AND DISK DRIVES

    - ENSURING THE INTEGRITY OF VOLUME CONTENTS

    - MAKING VOLUME CONTENTS ACCESSABLE TO PAGE CONTROL (PAGES) AND SEGMENT CONTROL (VTOC ENTRIES)

  - VOLUME MANAGEMENT IS INVOKED ONLY BY SUBROUTINE CALLS

- MAJOR DATA BASES

  - PHYSICAL VOLUME TABLE (PVT) - ONE PER SYSTEM

    - PHYSICAL VOLUME TABLE ENTRY (PVTE) - ONE PER DISK DRIVE KNOWN TO THE SYSTEM

    - EACH PVTE IDENTIFIES A DRIVE'S DEVICE NUMBER, SUBSYSTEM NAME, DEVICE TYPE, AND INFORMATION ABOUT THE PHYSICAL VOLUME CURRENTLY MOUNTED

    - USED TO MAP REFERENCES TO PAGES OF SEGMENTS INTO AN I/O REQUEST TO THE CORRECT DISK DRIVE

  - LOGICAL VOLUME TABLE (LVT) - ONE PER SYSTEM

    - LOGICAL VOLUME TABLE ENTRY (LVTE) - ONE PER MOUNTED LOGICAL VOLUME

    - EACH LVTE CONTAINS THE LOGICAL VOLUME ID, POINTERS TO MEMBER PVTE'S, AIM CLASS LIMITS, ETC.

    - USED TO DETERMINE A USER'S ACCESS TO A LOGICAL VOLUME (PRIVATE OR PUBLIC) AND TO LOCATE MEMBER PHYSICAL VOLUMES

  - VOLUME HEADER - ONE PER PACK

    - VOLUME LABEL (REGISTRATION AND ACCEPTANCE INFORMATION)

    - VOLUME MAP (OCCUPIED/VACANT INFORMATION FOR VOLUME CONTENTS)

VTOCES

VOLUME MANAGEMENT

- RECORD STOCKS - ONE PER MOUNTED VOLUME

  - ONLINE CACHE OF INFORMATION ABOUT USED / UNUSED RECORDS ON THE VOLUME

  - THIS INFORMATION IS DERIVED FROM THE VOLUME MAP, BUT KEPT ONLINE TO AVOID THE NECESSITY OF REFERRING TO THE VOLUME MAP ON DISK EVERY TIME A RECORD IS ALLOCATED OR FREED

  - WHEN THE CACHE BECOMES COMPLETELY EMPTY OR COMPLETELY FULL, IT MUST BE UPDATED FROM/TO DISK - A PROTOCOL ENSURES THAT THE COPY ON DISK IS ALWAYS CONSISTENT

  - PROVIDED BY VOLUME MANAGEMENT, BUT USED BY PAGE CONTROL

- VTOCE STOCKS - ONE PER VOLUME

  - SIMILAR TO RECORD STOCKS, BUT MAINTAINS INFORMATION ABOUT USED / UNUSED VTOC ENTRIES ON THE VOLUME

  - PROVIDED BY VOLUME MANAGEMENT, BUT USED BY SEGMENT CONTROL

- PHYSICAL VOLUME HOLD TABLE (PVHT) - ONE PER SYSTEM

  - RECORDS THE COMMENCEMENT OF COMPOUND I/O OPERATIONS UPON A PHYSICAL VOLUME

  - THIS INFORMATION PREVENTS A VOLUME FROM BEING DEMOUNTED WHILE SUCH AN OPERATION IS IN PROGRESS

SEGMENT CONTROL

- FUNCTION

  - SEGMENT CONTROL IS RESPONSIBLE FOR THE MANAGEMENT OF LOGICAL MEMORY

  - ITS TASKS INCLUDE:

    - MAINTAINING THE DISK RESIDENT MAPS OF SEGMENTS (IE: THEIR VTOCE'S)

    - SEGMENT CREATION, TRUNCATION AND DELETION

    - SEGMENT ACTIVATION AND DEACTIVATION (ASTE MULTIPLEXING)

  - SEGMENT CONTROL CAN BE INVOKED EITHER BY SUBROUTINE CALLS OR BY SEGMENT FAULTS

- BASIC PHILOSOPHY OF ACTIVATION/DEACTIVATION

  - OF ALL SEGMENTS RESIDENT WITHIN THE SYSTEM'S MOUNTED PHYSICAL VOLUMES, ONLY A SMALL SUBSET WILL REQUIRE ACCESSING AT ANY ONE TIME. SUCH SEGMENTS WILL BE CALLED "ACTIVE SEGMENTS"

  - A PART OF MAIN MEMORY, CALLED THE "ACTIVE SEGMENT TABLE" (AST), WILL BE RESERVED TO HOLD MANAGEMENT INFORMATION FOR THESE ACTIVE SEGMENTS (IDENTITY, PVT INDEX, LOCATION OF PAGES, ETC.)

- AS SEGMENTS FALL INTO DISUSE, THEIR "MANAGEMENT INFORMATION" IN THE AST WILL BE REPLACED WITH INFORMATION OF OTHER SEGMENTS REQUIRING ACTIVATION

- USER INTERFACE

  - COMMAND LEVEL

    - create, delete, truncate, etc.

  - SUBROUTINE LEVEL

    - hcs_$append_branch, hcs_$append_branchx, hcs_$delentry_seg, hcs_$delentry_file, hcs_$truncate_seg, hcs_$truncate_file, hcs_$force_write, etc

- MAJOR DATA BASES

  - SYSTEM SEGMENT TABLE (SST) - ONE PER SYSTEM, SHARED WITH PAGE CONTROL. ONE MAJOR COMPONENT IS "OWNED" BY SEGMENT CONTROL:

    - ACTIVE SEGMENT TABLE (AST) - ONE PER SYSTEM

    - THE AST IS A LIST OF ACTIVE (CURRENTLY BEING USED) SEGMENTS

    - ACTIVE SEGMENT TABLE ENTRY (ASTE) - ONE PER ACTIVE SEGMENT

      - ASTES CONTAIN PHYSICAL VOLUME ID'S (PVID'S) AND VTOC INDEX'S (VTOCX'S) OF SEGMENTS. NEEDED BY SEGMENT CONTROL TO FIND THE SEGMENT ON DISK (HARDWARE)

- AST HASH TABLE

  - ALLOWS EFFICIENT SEARCHING OF ASTE'S

  - LOGICALLY PART OF THE AST, BUT ELSEWHERE FOR HISTORICAL REASONS

- DIRECTORY SEGMENTS

  - CONTAIN LOCATIONS AND ATTRIBUTES OF SEGMENTS. LOCATION INFORMATION FROM DIRECTORY SEGMENTS IS PROVIDED TO SEGMENT CONTROL BY DIRECTORY CONTROL

- VOLUME TABLE OF CONTENTS (VTOC) - ONE PER PHYSICAL VOLUME

  - VOLUME TABLE OF CONTENTS ENTRY (VTOCE) - ONE PER DISK-RESIDENT SEGMENT

  - EACH VTOCE CONTAINS THE SEGMENT'S UNIQUE ID, CURRENT LENGTH, FILE MAP, ETC (NEED TO BUILD ASTE'S AND PT'S)

  - VTOCES ARE READ AND WRITTEN ONLY BY SEGMENT CONTROL

- VTOCE STOCKS - FROM VOLUME MANAGEMENT

  - USED WHEN CREATING AND DELETING VTOCES FOR SEGMENTS

PAGE CONTROL

- FUNCTION

  - PAGE CONTROL IS RESPONSIBLE FOR THE MANAGEMENT OF PHYSICAL MEMORY TO INCLUDE THE MULTIPLEXING OF MAIN MEMORY FRAMES, AND THE MANAGEMENT OF DISK STORAGE

  - ITS TASKS INCLUDE:

    - TRANSFERRING THE PAGES OF SEGMENTS BETWEEN THE MEMORY DEVICES, AND RECORDING THE LOCATION OF "THE" COPY OF THESE PAGES

    - REPORTING THE STATUS AND FILE MAPS OF SEGMENTS TO SEGMENT CONTROL

  - PAGE CONTROL IS LARGELY CODED IN MULTICS ASSEMBLER LANGUAGE (ALM)

  - PAGE CONTROL CAN BE INVOKED EITHER BY SUBROUTINE CALLS OR BY PAGE FAULTS

  - THERE ARE NO EXPLICIT USER INTERFACES TO PAGE CONTROL

PAGE CONTROL

- BASIC PHILOSOPHY

  - OF ALL THE SEGMENTS ACTIVE AT A GIVEN TIME, ONLY A SMALL SUBSET OF THEIR TOTAL PAGES WILL BE REQUIRED FOR ACCESSING

  - PAGES WILL BE READ INTO MAIN MEMORY AS THEY ARE REQUIRED

  - THE READING OF A PAGE INTO MAIN MEMORY WILL (PROBABLY) REQUIRE THE EVICTION OF A PREVIOUSLY REQUIRED PAGE

  - THE CHOICE OF A PAGE FOR EVICTION WILL BE BASED ESSENTIALLY UPON A "LEAST RECENTLY USED" CRITERIA

  - AN EVICTED PAGE NEED BE WRITTEN BACK TO DISK <u>ONLY</u> IF IT WAS MODIFIED DURING ITS RESIDENCY IN MAIN MEMORY

- MAJOR DATA BASES

  - PHYSICAL VOLUME TABLE (PVT) - ONE PER SYSTEM. PROVIDED BY VOLUME MANAGEMENT

    - PHYSICAL VOLUME TABLE ENTRY (PVTE) - ONE PER DISK DRIVE CONFIGURED

    - EACH PVTE CONTAINS:

      - THE DEVICE ID (DISK DRIVE ID) AND THE ID OF THE PHYSICAL VOLUME (DISK PACK) CURRENTLY MOUNTED

  - THE NUMBER OF RECORDS LEFT UNALLOCATED ON THE PHYSICAL VOLUME, POINTER TO THE RECORD STOCK, ETC

- RECORD STOCKS - ONE PER MOUNTED PHYSICAL VOLUME, PROVIDED BY VOLUME MANAGEMENT

  - CONTAINS AN IN-MEMORY CACHE OF THE IN-USE STATUS OF RECORDS ON THE VOLUME, FROM THE VOLUME MAP, USED WHEN ALLOCATING OR FREEING PAGES

  - ACCESSED BY A COMPLEX MECHANISM WHICH USES NORMAL PAGE I/O BUT HAS A PROTOCOL TO ENSURE SYNCHRONIZATION OF DISK CONTENTS AND RECORD STOCK CONTENTS

- SYSTEM SEGMENT TABLE (SST) - ONE PER SYSTEM. SHARED WITH SEGMENT CONTROL. CONTAINS THE FOLLOWING FIVE DATA BASES USED BY PAGE CONTROL:

  - SYSTEM SEGMENT TABLE (SST) HEADER - ONE PER SYSTEM

    - CONTAINS A LARGE NUMBER OF COUNTERS AND POINTERS VITAL TO THE MAINTENANCE AND METERING OF THE STORAGE SYSTEM

    - CONTAINS LOCKWORDS USED TO SYNCHRONIZE PAGE CONTROL AND SEGMENT CONTROL OPERATIONS

  - CORE MAP - THE core_map SEGMENT - ONE PER SYSTEM

    - CORE MAP ENTRY (CME) - ONE PER FRAME (1024 WORDS) OF CONFIGURED MAIN MEMORY

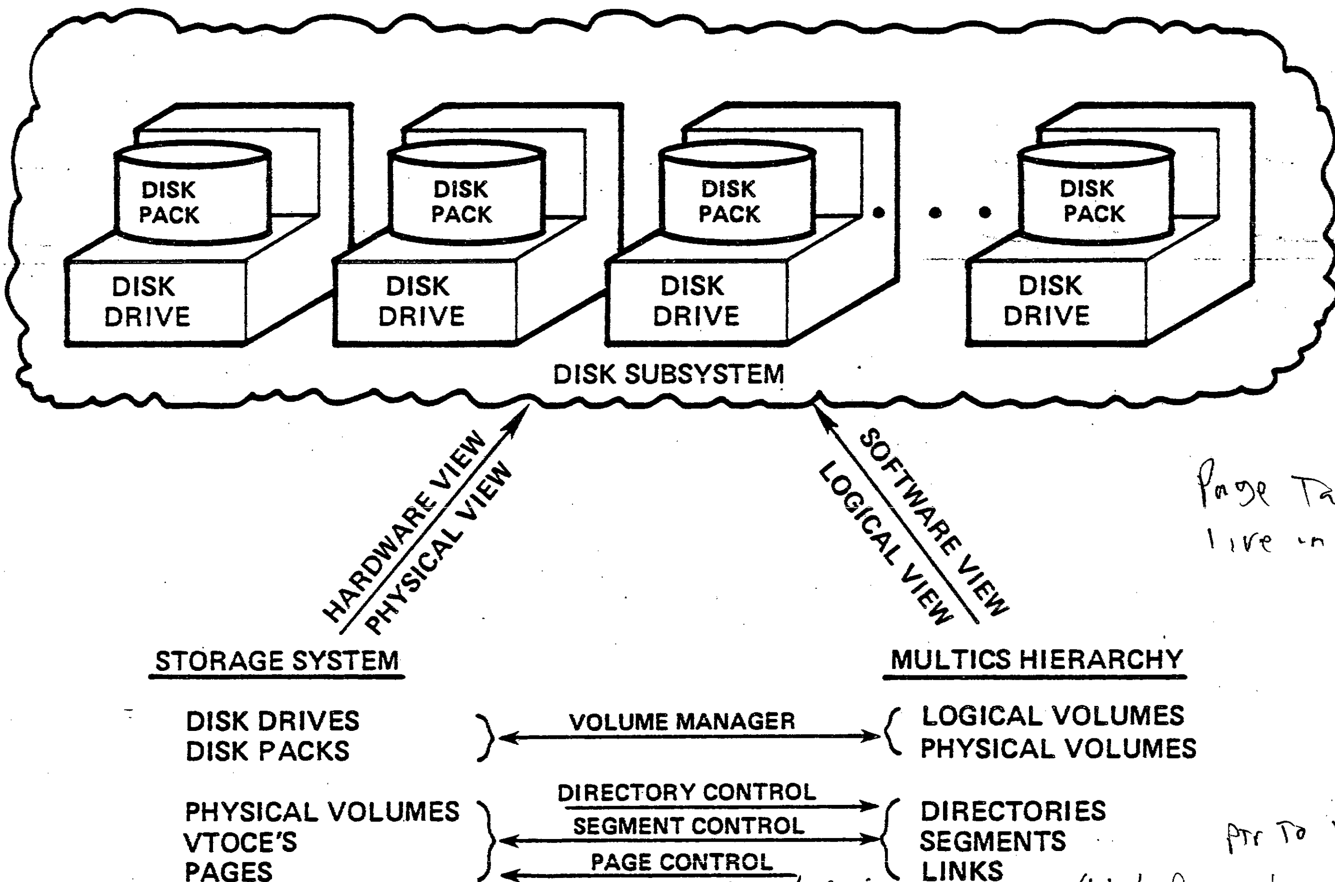
    - EACH CME REPRESENTS A FRAME OF MAIN MEMORY AND IDENTIFIES THE CURRENT OCCUPANT OF THAT FRAME

    -

NOT PART OF THE SST SEGMENT ANY MORE, BUT LOGICALLY PART OF THE SST

- ACTIVE SEGMENT TABLE (AST) - ONE PER SYSTEM
  - ACTIVE SEGMENT TABLE ENTRY (ASTE) - ONE PER ACTIVE SEGMENT
  - LIST OF ACTIVE (CURRENTLY BEING USED) SEGMENTS
- PAGE TABLES (PT) - ONE PER ACTIVE SEGMENT, THE OTHER HALF OF EACH ASTE
  - PAGE TABLE WORD (PAGE PTW) - EITHER 4, 16, 64, OR 256 PER PAGE TABLE
  - EACH PTW DEFINES THE CURRENT LOCATION OF A PAGE OF THE SEGMENT: DISK, MAIN MEMORY ADDRESS, OR NULL

THE MULTICS FILE SYSTEM

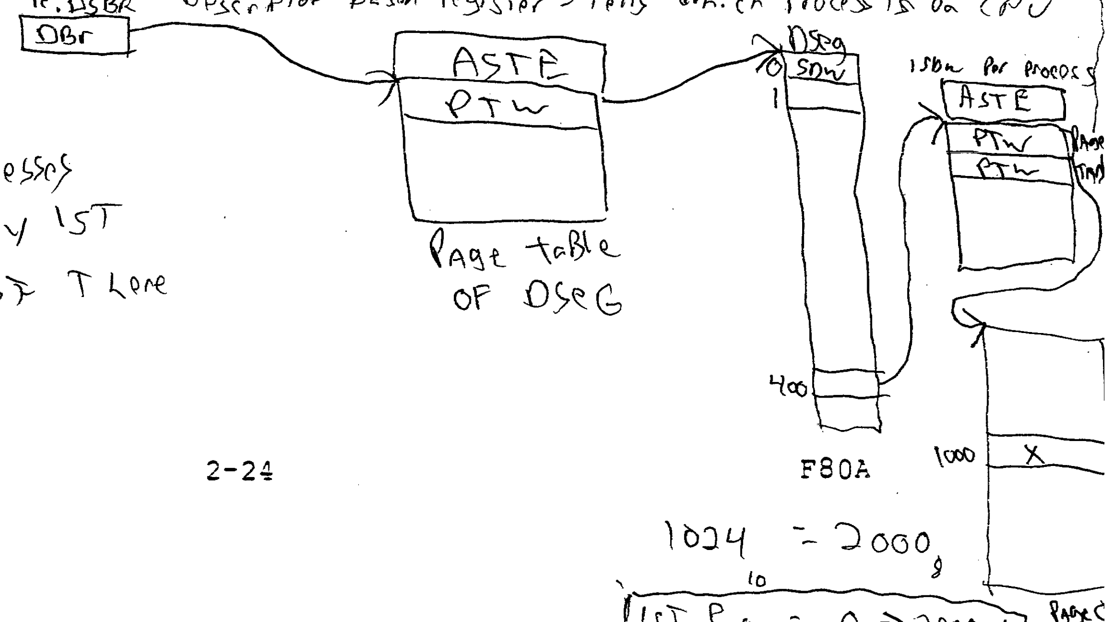


Page Tables live in AST

ptr To Dseg

ie: DSBR Descriptor Based register - Tells which Process is on CPU

Segno offset
400/1000



most Processes use only 1ST Page of T here DSeg.

$1024_{10} = 2000_8$

1ST Page = 0 → 2000-1

- FUNCTION

  - TRAFFIC CONTROL (OR THE "TRAFFIC CONTROLLER") IS RESPONSIBLE FOR MANAGING THE ASSIGNMENT OF PHYSICAL PROCESSORS TO MULTICS PROCESSES AND IMPLEMENTING THE SYSTEM'S WAIT/NOTIFY AND INTERPROCESS COMMUNICATION PRIMITIVES

  - THE FUNCTIONS ASSUMED BY THE TRAFFIC CONTROLLER ARE KNOWN AS MULTIPROGRAMMING, MULTIPROCESSING, SCHEDULING, DISPATCHING, PROCESSOR MANAGEMENT, AND INTERPROCESS COMMUNICATION.

  - ITS MAJOR FUNCTION IS ALLOWING PROCESSES TO AWAIT THE COMPLETION OF FILE SYSTEM OPERATIONS, SUCH AS PAGE I/O

  - TRAFFIC CONTROL CAN BE INVOKED BY SUBROUTINE CALLS AND INTERRUPTS

  - THERE ARE NO IMPORTANT USER SUBROUTINE INTERFACES, BUT THERE ARE PRIVILEGED SUBROUTINE INTERFACES FOR PROCESS CREATION, ADJUSTMENT OF SCHEDULING PARAMETERS, ETC.

- MAJOR DATA BASES

  - TC_DATA SEGMENT - ONE PER SYSTEM. CONTAINS THE FOLLOWING FOUR DATA BASES:

    - TC_DATA HEADER - ONE PER SYSTEM

      - CONTAINS VARIOUS METERS, COUNTERS AND POINTERS USED BY THE TRAFFIC CONTROLLER

- ACTIVE PROCESS TABLE (APT) - ONE PER SYSTEM
  - ACTIVE PROCESS TABLE ENTRY (APTE) - ONE OCCUPIED PER ACTIVE PROCESS (TOTAL NUMBER IS DETERMINED BY CONFIG DECK)
  - EACH APTE CONTAINS VARIOUS ATTRIBUTES OF AN ACTIVE PROCESS INCLUDING THE PROCESS ID, STATE, THE VALUE OF ITS DESCRIPTOR BASE REGISTER (DBR), SCHEDULING PARAMETERS, AND A POINTER TO THE PROCESS'S ITT ENTRIES
  - THE APTE CONTAINS ALL INFORMATION THE SUPERVISOR NEEDS TO KNOW ABOUT A PROCESS WHEN THE PROCESS IS NOT RUNNING
- INTERPROCESS TRANSMISSION TABLE (ITT) - ONE PER SYSTEM
  - ITT ENTRY - ONE OCCUPIED PER OUTSTANDING IPC WAKEUP
  - A QUEUE FOR TEMPORARILY STORING IPC WAKEUP INFORMATION (CHANNEL NAME, RANDOM DATA, PROCESS ID, ETC)
- WORK CLASS TABLE (WCT) - ONE PER SYSTEM
  - WORK CLASS TABLE ENTRY (WCTE) - ONE PER WORKCLASS
  - EACH WCTE CONTAINS ADMINISTRATOR DEFINED PARAMETERS OF THE WORKCLASS, VARIOUS METERS AND POINTERS

FAULT AND INTERRUPT HANDLING

- FUNCTION

  - RESPONSIBLE FOR HANDLING ALL EXCEPTIONS IN A CPU WHETHER INTERNAL TO THE PROCESSOR (REFERRED TO AS FAULTS) OR EXTERNAL (REFERRED TO AS INTERRUPTS)

  - ESTABLISHES THE SUPERVISOR ENVIRONMENT AT FAULT AND INTERRUPT TIME. SAVES THE MACHINE CONDITIONS AND TRANSFERS TO THE APPROPRIATE HANDLER

  - MAJOR COMPONENTS: THE FAULT INTERCEPT MODULE (fim), WIRED-FAULT INTERCEPT MODULE (wired_fim), I/O INTERRUPT HANDLER (io_interrupt), sys_trouble, page_fault

- MAJOR DATA BASES

  - INTERRUPT VECTORS - ONE SET PER SYSTEM (WIRED)

    - INTERRUPT PAIR (2 INSTRUCTIONS) - ONE PAIR PER DEFINED INTERRUPT TYPE

    - LOCATED AT ABSOLUTE ADDRESS 0. A HARDWARE RECOGNIZED DATA BASE

    - DESCRIBE WHERE TO SAVE THE CONTEXT, AND WHERE TO TRANSFER TO TO PROCESS THE INTERRUPT (ALWAYS io_interrupt)

FAULT AND INTERRUPT HANDLING

- FAULT VECTORS - ONE SET PER SYSTEM (WIRED)

  - VECTOR PAIR (2 INSTRUCTIONS) - ONE PAIR PER DEFINED FAULT TYPE

  - LOCATED AT ABSOLUTE ADDRESS 100 (OCTAL) IMMEDIATELY ABOVE THE INTERRUPT VECTORS. A HARDWARE RECOGNIZED DATA BASE

  - DESCRIBE WHERE TO SAVE THE CONTEXT, AND WHERE TO TRANSFER TO TO PROCESS THE FAULT (fim, wired_fim, page_fault)

- PROCESS DATA SEGMENT (PDS) - ONE PER PROCESS (WIRED WHEN ELIGIBLE)

  - CONTAINS PROCESS RELEVANT INFO SUCH AS PROCESS ID, USER ID, HOME/WORKING/PROCESS DIRECTORIES, AIM CLASSIFICATION, INITIAL RING, ETC

  - CONTAINS ALL INFORMATION ABOUT THE PROCESS NEEDED BY THE SUPERVISOR CODE WHEN THE PROCESS IS RUNNING

  - CONTAINS SAVE AREAS FOR CONTEXT INFORMATION ABOUT FAULTS WHICH CAN RESULT IN GIVING UP THE PROCESSOR: PAGE FAULTS, SEGMENT FAULTS, AND ALL FAULTS NOT HANDLED BY THE SUPERVISOR

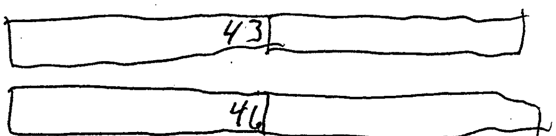
- PROCESSOR DATA SEGMENT (PRDS) - ONE PER CONFIGURED CPU (WIRED)

  - SERVES AS RING-ZERO STACK FOR PAGE CONTROL AND TRAFFIC CONTROL

  - ALSO CONTAINS SAVE AREAS FOR CONTEXT INFORMATION ABOUT FAULTS WHICH USUALLY DO NOT MEAN GIVING UP THE PROCESSOR: CONNECT FAULTS AND INTERRUPTS.

- FIM_TABLE Part of Fim not a real Table.

  - A TABLE IN THE FIM PROGRAM WHICH DESCRIBES THE ACTION TO BE TAKEN FOR VARIOUS TYPES OF FAULTS

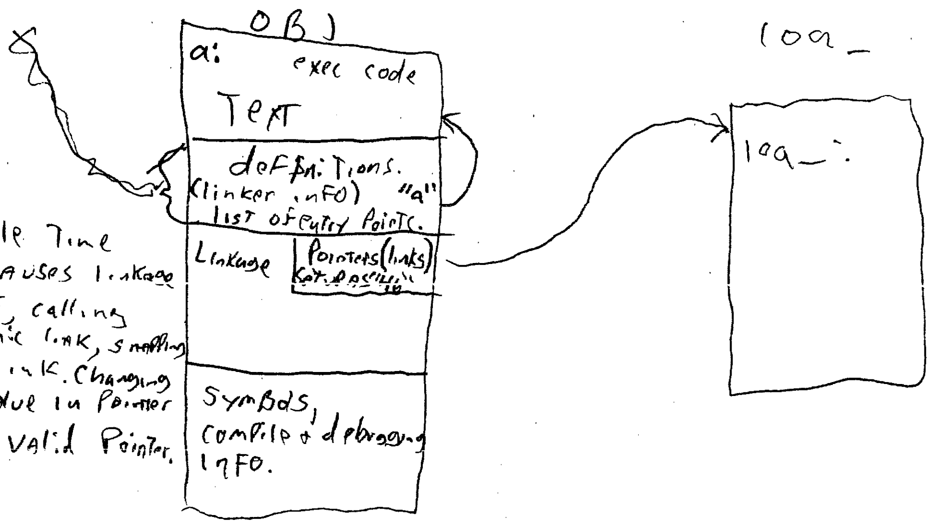


→46 causes linkage fault.

LINKS

~~also has list~~

a.P11



Compile Time 46 causes linkage fault, calling dynamic link, snapping The link. Changing 46 value in Pointer To a valid Pointer.

linkage section shared among Processes, So links snapped are Per Process. So first Time Link is snapped, linkage Section is copied Per Process.
Internal static is also in linkage Section.

SYSTEM INITIALIZATION

Supervisor Link Snapping Is Static, not Dynamic. ie. done ahead of Time. Ring 0 does no dynamic Linking.

- FUNCTION

  - PREPARE THE SYSTEM TO OPERATE, STARTING FROM A COMPLETELY EMPTY MACHINE

    SNAP ie. Putting Together or linking.

  - READS IN SUPERVISOR PROGRAMS FROM SYSTEM TAPE, SNAPS LINKS BETWEEN SUPERVISOR COMPONENTS, VERIFIES AND INITIALIZES HARDWARE CONFIGURATION, SETS UP SYSTEM DATABASES, ACCEPTS STORAGE SYSTEM DISKS AND PREPARES THEM FOR USE BY THE FILE SYSTEM

  - MOST PROGRAMS IN SYSTEM INITIALIZATION ARE DELETED AFTER INITIALIZATION IS COMPLETE. only Text section is kept.

  - SUPERVISOR PROGRAMS ARE LOADED IN THREE "COLLECTIONS", EACH OF WHICH DEPENDS ON THE MECHANISMS SET UP BY THE PREVIOUS ONE

- MAJOR DATA BASES

  - THESE DATA BASES ARE ALL BUILT DURING THE PROCESS OF INITIALIZATION (EXCEPT FOR THE CONFIG DECK) AND KEPT AFTER INITIALIZATION IS FINISHED for debugging

  - SEGMENT LOADING TABLE (>sl1>slt)

    - CONTAINS AN ENTRY DESCRIBING THE ATTRIBUTES OF EACH SEGMENT IN THE SUPERVISOR

SYSTEM INITIALIZATION

for debugging

- NAME TABLE (>sl1>name_table)

  & seg ~~no's~~ numbers

  - CONTAINS A LIST OF NAMES FOR EACH OF THE SEGMENTS IN THE SUPERVISOR
    ie. bound-interceptor

for debugging (Azm)

- DEFINITIONS SEGMENT (>sl1>definitions_)

  - CONTAINS THE DEFINITIONS SECTIONS FOR ALL THE SEGMENTS IN THE SUPERVISOR, WHICH ARE USED IN ORDER TO SNAP LINKS BETWEEN THE SUPERVISOR MODULES

- CONFIG DECK (>sl1>config_deck)

  - CONTAINS A DESCRIPTION OF THE HARDWARE CONFIGURATION AND CERTAIN SOFTWARE PARAMETERS

  - PROVIDED TO SYSTEM INITIALIZATION BY BOS

- SHUTDOWN -- TERMINATES THE ACTIVITIES OF THE SYSTEM IN AN ORDERLY FASHION

  - TWO TYPES OF SHUTDOWN:

    - NORMAL -- REQUESTED BY THE INITIALIZER, RUNS IN THE USUAL SUPERVISOR ENVIRONMENT

    - EMERGENCY -- USED AFTER A CRASH, MUST MAKE THE SUPERVISOR ENVIRONMENT OPERABLE BEFORE PROCEEDING

SYSTEM INITIALIZATION

- BOTH TYPES EXIST PRIMARILY TO SHUT DOWN THE FILE SYSTEM -- THAT IS, TO WRITE ALL DATA IN MEMORY INTO ITS PROPER HOME ON DISK

  - INCLUDES PAGES OF SEGMENTS, VTOCES, VOLUME AND VTOC MAPS

- SHUTDOWN ESSENTIALLY RUNS THE STEPS OF INITIALIZATION BACKWARDS, BUT WITH A LOT OF SHORTCUTS

FILE SYSTEM SALVAGERS

- FUNCTION

  - ENSURE THE CONSISTENCY OF THE FILE SYSTEM DATABASES AND PERFORM PERIODIC PREVENTIVE MAINTENANCE OPERATIONS

  - THERE ARE SEVERAL SALVAGERS, EACH WITH A DIFFERENT FUNCTION

  - BECAUSE OF THE COMPLICATED INTERACTIONS THEY HAVE WITH THE REST OF THE FILE SYSTEM, THE SALVAGERS ARE PERHAPS THE MOST COMPLICATED SINGLE PROGRAMS IN THE SUPERVISOR

  - SOME SALVAGING IS DONE AUTOMATICALLY, WHEN THE SYSTEM DETECTS AN INCONSISTENCY. OTHER SALVAGE OPERATIONS ARE EXPLICITLY REQUESTED, BY PRIVILEGED USERS.

  - EXCEPT FOR SUPERVISOR BUGS, THE ONLY TIME DAMAGE OCCURS THAT REQUIRES SALVAGING TO FIX IS AFTER A CRASH WHERE EMERGENCY SHUTDOWN FAILS

- THE SALVAGERS:

  - DIRECTORY SALVAGER

    - CORRECTS INCONSISTENCIES IN DIRECTORY SEGMENTS BY REBUILDING THEM

    - THIS IS THE ONLY SALVAGER INVOKED AUTOMATICALLY IN USER PROCESSES: ANY ATTEMPT TO LEAVE RING ZERO WITH A DIRECTORY LOCKED FOR WRITING WILL CAUSE IT TO BE SALVAGED

FILE SYSTEM SALVAGERS

- DIRECTORY SALVAGING ALSO RECLAIMS WASTED SPACE IN THE DIRECTORY, AND IS RUN PERIODICALLY TO COMPACT DIRECTORIES

- QUOTA SALVAGER

  - CORRECTS INCONSISTENCIES IN THE QUOTA SYSTEM

- PHYSICAL VOLUME SCAVENGER Fixes Volume Map & VTOCE MAP frees up records marked as being used.

  - RECONSTRUCTS RECORD AND VTOCE STOCK INFORMATION FROM THE VTOCES ON A VOLUME, THEREBY RECLAIMING ANY RECORDS OR VTOCES WHICH MIGHT HAVE BEEN LOST

  - RUNS ENTIRELY ONLINE WHILE THE SYSTEM IS UP FOR USERS (NEW IN MR10.1)

  - THIS TYPE OF DAMAGE IS USUALLY BENIGN, SO RUNNING THE SCAVENGER CAN BE DELAYED.

- PHYSICAL VOLUME SALVAGER

  - RECONSTRUCTS RECORD AND VTOCE STOCK INFORMATION

  - RUNS ONLY DURING INITIALIZATION, AND THEREFORE DELAYS CRASH RECOVERY

  - NOW USED ONLY FOR RARE CASES WHERE THERE IS NOT ENOUGH FREE SPACE LEFT FOR THE SCAVENGER TO RUN. IN THESE RARE CASES, IT IS INVOKED AUTOMATICALLY BY SYSTEM INITIALIZATION.

- SWEEP_PV Fixes reverse connection failure. Fixes dirs where scav fixes seg

FILE SYSTEM SALVAGERS

- DELETES UNUSED VTOC ENTRIES WHICH HAVE NO DIRECTORY ENTRY POINTING TO THEM

- RUNS ENTIRELY IN USER RING, EXCEPT FOR ACTUALLY READING VTOC ENTRIES AND DIRECTORY ENTRIES

- PURELY A HOUSEKEEPING FUNCTION, AND RUN ONLY RARELY.

METERING & TUNING

- WHILE NOT A SUBSYSTEM ITSELF, METERING AND TUNING IS A POLICY AND CAPABILITY COMMON TO ALL OF THE SUPERVISOR'S SUBSYSTEMS

- FUNCTION

  - METERING (CONSISTS OF THREE ACTIVITIES)

    - ACCUMULATING DATA: THIS IS PERFORMED THROUGHOUT THE SUPERVISOR BY CODE WHICH

      - RECORDS THE NUMBER OF TIMES AN EVENT HAPPENS OR A PARTICULAR PIECE OF CODE IS EXECUTED; AND/OR

      - RECORDS THE TIME REQUIRED TO PERFORM A TASK

      - SUCH DATA IS STORED IN AREAS REFERRED TO AS "METERING CELLS"

    - EXTRACTING DATA: THIS IS PERFORMED BY NUMEROUS METERING COMMANDS WHICH (WHEN INVOKED)

      - READ AND STORE THE CURRENT VALUES OF RELEVANT METERING CELLS

    - REPORTING THE DATA: THIS IS PERFORMED BY THE METER COMMANDS WHICH (WHEN INVOKED)

      - COMPARE CURRENT METERING CELL VALUES WITH PREVIOUSLY READ VALUES

      - PERFORM THE APPROPRIATE ARITHMETIC COMPUTATIONS UPON THE DATA IN ORDER TO ARRIVE AT THE DESIRED STATISTIC

      - ARRANGE THE DATA IN A USEFUL FORMAT (A REPORT OR DIAGRAM) AND PRINT IT

- TUNING
  - CHANGING THE SYSTEM'S OPERATING PARAMETERS AND/OR CONFIGURATION BASED UPON THE DATA AND INSIGHTS FROM THE SYSTEM'S METERS

- MAJOR DATA BASES
  - SST HEADER, TC_DATA HEADER, ETC.

INITIALIZER.SYSDAEMON

- FUNCTION

  - THE SYSTEM'S INITIALIZATION, ADMINISTRATIVE AND CONTROL PROCESS (Initializer.SysDaemon.z), RESPONSIBLE FOR:

    - INITIALIZING THE OPERATING SYSTEM AT BOOTLOAD, FOLLOWING SUCCESSFUL INITIALIZATION OF THE SUPERVISOR

    - ANSWERING SERVICE (login and logout)

    - PROCESS CREATION AND DESTRUCTION

    - MESSAGE COORDINATOR (DAEMON COORDINATION)

    - SYSTEM ADMINISTRATION FUNCTIONS

    - SYSTEM ACCOUNTING FUNCTIONS

- MAJOR DATA BASES, ALL KEPT IN >sc1

  - ANSWER_TABLE

  - ABSENTEE_USER_TABLE

  - DAEMON_USER_TABLE

INITIALIZER.SYSDAEMON

- MASTER GROUP TABLE (MGT)
- CHANNEL DEFINITION TABLE (CDT)
- SYSTEM ADMINISTRATION TABLE (SAT)
- PERSON NAME TABLE (PNT)
- PROJECT DEFINITION TABLES (PDT'S)

TOPIC III

The Multics Environment

Page

WHAT IS A PROCESS

- A MULTICS PROCESS IS A WELL DEFINED COLLECTION OF SEGMENTS, EACH WITH DEFINED ACCESS, OVER WHICH A SINGLE EXECUTION POINT IS FREE TO ROAM (I.E., FETCH INSTRUCTIONS AND MAKE DATA REFERENCES)

- THE ADDRESS SPACE OF A PROCESS IS THE ABOVE "COLLECTION OF SEGMENTS". SUCH SEGMENTS ARE SAID TO BE KNOWN TO THE PROCESS

- EVERY LOGGED IN USER HAS A PROCESS

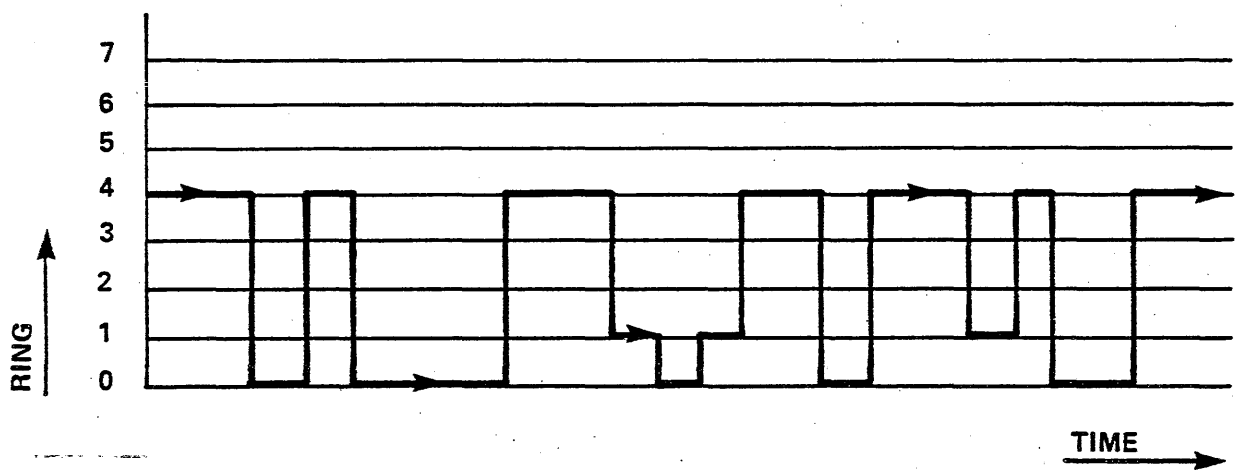
- VERY IMPORTANT CONCEPT: THE MULTICS SUPERVISOR RUNS IN THE USER'S PROCESS (IE: IN THE USER'S ADDRESS SPACE), BUT IN A DIFFERENT RING

- A PROCESS TAKES ON THE IDENTITY OF THE SOFTWARE IT IS EXECUTING WHERE EVER IT GOES

  - WHEN A USER WISHES TO CREATE A SEGMENT, IT IS THE USER'S PROCESS WHICH EXECUTES THE SUPERVISOR CODE hcs_$append, CREATING THE SEGMENT

- A PROCESS CAN BE VIEWED AS A CONTINUAL FLOW OF EXECUTION FLUCTUATING BETWEEN DIFFERENT RINGS: PRIMARILY RING FOUR AND RING ZERO

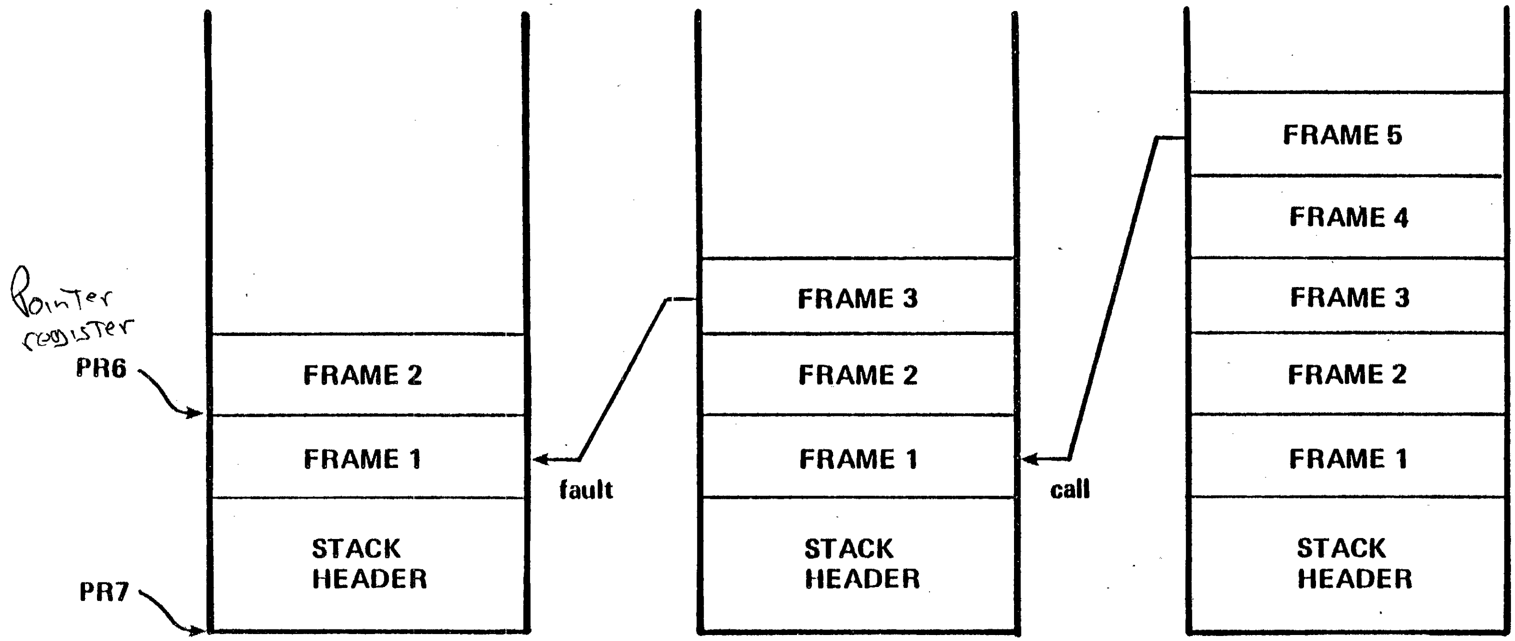
PROCESS FLOW OF EXECUTION



- ALL PROCEDURE CODE (WHETHER SUPERVISOR OR USER CODE) MUST HAVE A STACK FRAME CONTAINING ITS ARGUMENTS AND ENVIRONMENT DATA

- FOR REASONS OF SECURITY, MULTICS REQUIRES ONE STACK PER RING OF EXECUTION. WHEN EXECUTING RING "N" PROCEDURES, THERE WILL EXIST A RING "N" STACK CONTAINING STACK FRAMES FOR THESE PROCEDURES

STACK PER RING



Seg # 230 stack_0 (0,0,0) SEVERAL (10-30) PER SYSTEM, SHARED	# 231 stack_1 (1,1,1) IN USER'S PROCESS DIRECTORY	# 234 stack_4 (4,4,4) IN USER'S PROCESS DIRECTORY

first one ending in zero after #'s arrived for supervisor code which is at bootload

* PR7 IS THE STACK BASE POINTER REGISTER. POINTS TO CURRENT STACK. HARDWARE LOADED WITH DBR.stack *8 + RING. THE VALUE OF DBR.stack CAN CHANGE WITH EACH RELEASE. IT IS 23 IN MR10.1.

* PR6 IS THE STACK FRAME POINTER REGISTER. POINTS TO CURRENT STACK FRAME.

COOPERATING PROCESSES

- ALL ACTIVE PROCESSES (INTERACTIVE, ABSENTEE, AND DAEMONS) APPEAR TO BE AUTONOMOUS AND INDEPENDENT OF ONE ANOTHER

- IN REALITY, ALL PROCESSES ARE CONTINUALLY COOPERATING, COMPETING AND SHARING

  - EXAMPLES OF COOPERATION

    - VOLUNTARY

      - THE SENDING AND ACCEPTING OF MESSAGES AND MAIL

    - PREPLANNED BY SYSTEM PROGRAMMERS

      - EVERY PROCESS, BEFORE RELINQUISHING A PROCESSOR, CHOOSES THE MOST DESERVING REPLACEMENT AND EXECUTES THE CODE WHICH DISPATCHES THE CHOSEN PROCESS

      - EVERY PROCESS, WHEN RUNNING, WILL SERVICE ALL INTERRUPTS FIELDED BY ITS PROCESSOR. THESE INTERRUPTS ARE GENERALLY THE REPLIES TO THE REQUESTS OF <u>OTHER</u> PROCESSES (IE: THE ARRIVAL OF A PAGE REQUESTED SOME TIME EARLIER)

    - PREPLANNED BY APPLICATION PROGRAMMERS

      - THE MULTICS TRANSACTION PROCESSOR IS COMPOSED OF MANY COOPERATING, INTER-DEPENDENT PROCESSES

  - EXAMPLES OF COMPETITION

    - ALL PROCESSES COMPETE FOR PROCESSOR TIME AND MAIN MEMORY RESOURCES

- THIS COMPETITION IS HIGHLY REGULATED IN ORDER FOR ALL PROCESSES TO BE TREATED FAIRLY

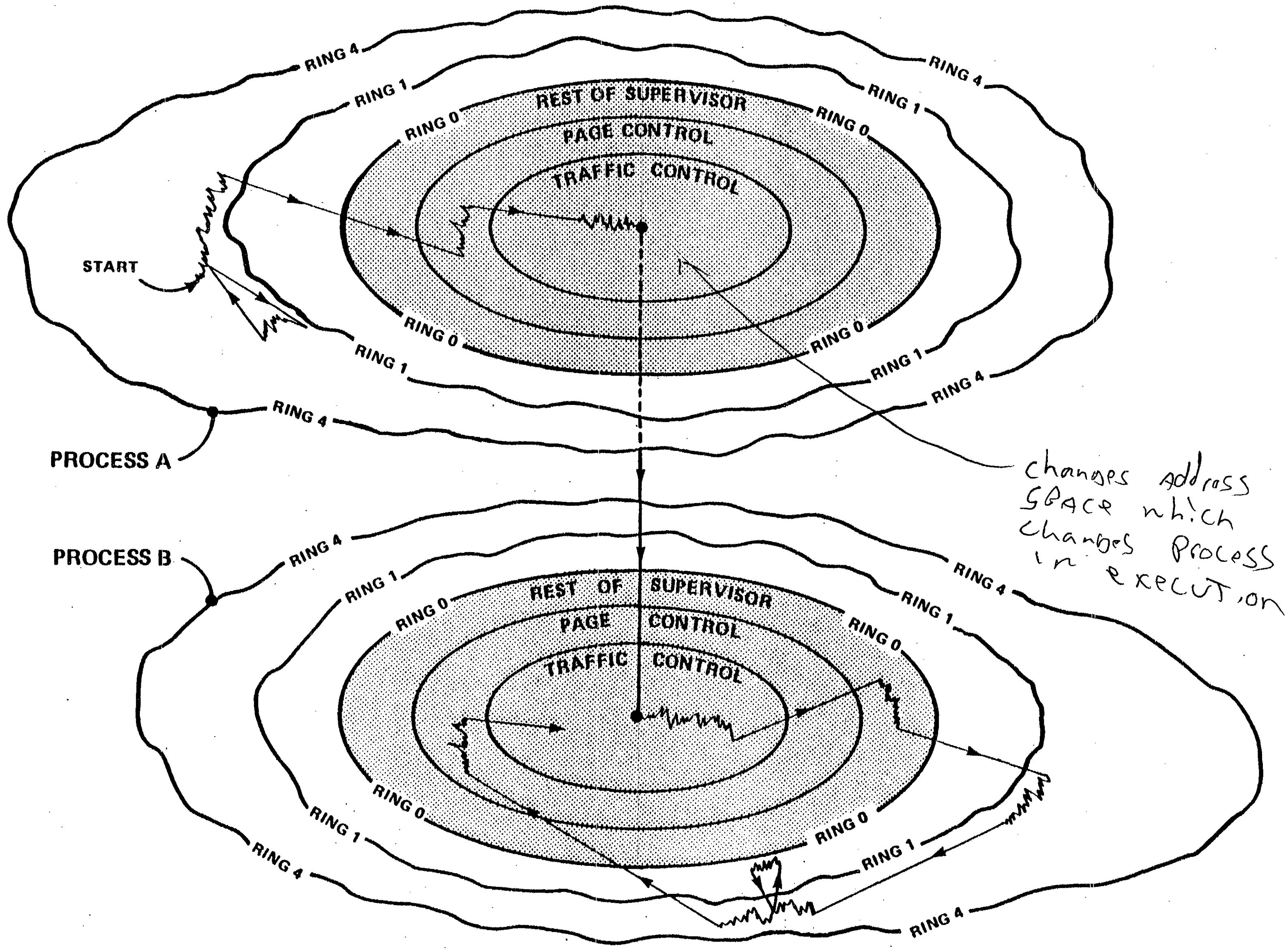
- THE COMPETITION IS ALSO SUBJECT TO VERSATILE ADMINISTRATIVE CONTROLS

- EXAMPLES OF SHARING

  - BY DESIGN, A SIGNIFICANT PART OF THE ADDRESS SPACE OF ALL PROCESSES IS IDENTICAL (THE SUPERVISOR SEGMENTS)

  - BY DEFAULT, REFERENCES TO SEGMENT foo BY TWO DIFFERENT PROCESSES WILL RESULT IN REFERENCES TO THE SAME SEGMENT (LOGICALLY, PHYSICALLY, ACTUALLY AND ABSOLUTELY)

- THERE IS NO SEPARATE ENTITY IN MULTICS LIKE AN EXECUTIVE DOING THINGS ON BEHALF OF THE USER. THE Initializer.SysDaemon IS NOT THE TIME-SHARE EXECUTIVE OF MULTICS


START
RING 4
RING 1
RING 0
REST OF SUPERVISOR
PAGE CONTROL
TRAFFIC CONTROL
PROCESS A
PROCESS B
changes address space which changes process in execution

THE PL/I OPERATORS

- OPERATORS ARE LANGUAGE DEPENDENT PIECES OF CODE WHICH IMPLEMENT HARDWARE OR OPERATING-SYSTEM DEPENDENT FUNCTIONS SUCH AS CALLING AND SIGNALLING

- CURRENTLY THERE ARE OPERATORS FOR PL/I, COBOL, AND BASIC. ALM AND FORTRAN SHARE THE PL/I OPERATORS

- ALL OPERATORS IN MULTICS ARE PURE, SHARED AND RE-ENTRANT ALM CODE

- OPERATORS COULD BE GENERATED BY THE COMPILERS AND PLACED IN LINE WITH OTHER CODE, HOWEVER, THERE ARE DISADVANTAGES:

  - SOME OPERATORS ARE TOO BULKY TO BE INCLUDED WITH EACH USE (SUCH AS COMPLICATED I/O STATEMENTS)

  - SOME OPERATORS MIGHT CHANGE IN THE FUTURE (SUCH AS ENTRY AND RETURN SEQUENCES)

- OPERATORS ARE SIMILAR TO QUICK INTERNAL PROCEDURES IMPLEMENTING WHAT IS OFTEN CALLED "LIBRARY FUNCTIONS" IN OTHER OPERATING SYSTEMS

THE PL/I OPERATORS

- INSTEAD OF PASSING ARGUMENT LIST, ARGUMENTS ARE USUALLY PASSED TO THE OPERATORS IN THE CPU'S REGISTERS

- INSTEAD OF BEING CALLED BY A PROCEDURE CALL, A SINGLE TRANSFER INSTRUCTION IS USED (tsx0 or tsp3)

- THE PL/I OPERATORS IMPLEMENT THE SUPPORT FUNCTIONS FOR THE PL/I ENVIRONMENT

- SINCE A MULTICS PROCESS IS A PL/I ENVIRONMENT, THE PL/I OPERATORS ARE VITAL TO THE MULTICS SUPERVISOR (AND ANY OTHER PROGRAMS WRITTEN IN PL/I)

INTERFACES TO SYSTEM MODULES

- UNLIKE OTHER SUPERVISORS, THE MULTICS SUPERVISOR IS NOT SEQUENTIAL - THAT IS, THE CONCEPT OF "JOB FLOW" DOES NOT REALLY APPLY

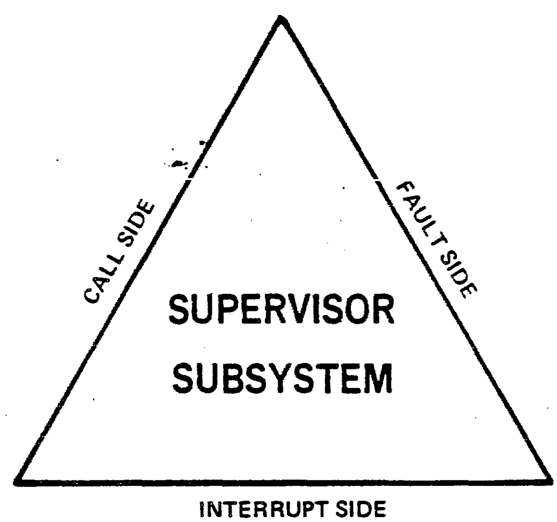
  - INSTEAD, THE SUBSYSTEMS PERFORM ASYNCHRONOUSLY, BEING 'INVOKED' BY THOSE PROCESSES WHO REQUIRE THEIR SERVICES

  - THESE SUBSYSTEMS ARE INVOKED BY THE USER'S PROCESS IN ONE OF THREE WAYS:

    - EXPLICITLY - VIA A SUBROUTINE CALL OR A COMMAND

    - IMPLICITLY - VIA A FAULT
      not repeatable so not Billed, system overhead.

    - IMPLICITLY - VIA AN INTERRUPT



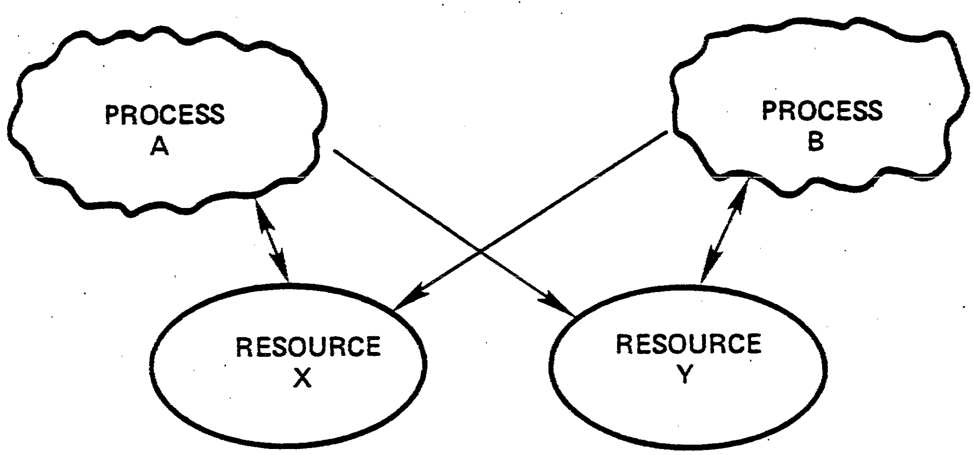
CALL SIDE:	SERVICES PERFORMED AS A RESULT OF EXPLICIT SUBROUTINE CALLS. LOCKS ARE NORMALLY WAIT LOCKS.
FAULT SIDE:	SERVICES PERFORMED AS A RESULT OF FAULTS. LOCKS ARE NORMALLY WAIT LOCKS.
INTERRUPT SIDE:	SERVICE PERFORMED AS A RESULT OF INTERRUPTS. LOCKS ARE LOOP LOCKS.

DEADLOCK PREVENTION

- WHAT IS DEADLOCK?

  - DEADLOCK CAN OCCUR IN ANY MULTI-PROGRAMMING ENVIRONMENT WHEN TWO OR MORE PROCESSES COMPETE RANDOMLY FOR SERIALLY REUSABLE RESOURCES

  - THE CLASSIC EXAMPLE OF DEADLOCK IS THE "DEADLY EMBRACE"



DEADLY EMBRACE

- PROCESS A IS WAITING FOR A RESOURCE OWNED BY PROCESS B.
- PROCESS B IS WAITING FOR A RESOURCE OWNED BY PROCESS A.

DEADLOCK PREVENTION

- DEADLOCK SOLUTIONS

  - DETECTION AND UNLOCKING

    - SOME SYSTEMS EMPLOY SCHEMES WHICH DETECT THE OCCURRENCE OF DEADLOCK AND "UNTANGLE" THE INVOLVED PROCESSES

    - DETECTION SCHEMES ARE USUALLY DIFFICULT TO IMPLEMENT AND EXPENSIVE IN TERMS OF OVERHEAD

    - THE ACT OF UNTANGLING THE INVOLVED PROCESSES USUALLY RESULTS IN AT LEAST ONE OF THEM LOSING RESOURCES, PRIORITY, OR EVEN ITS LIFE

  - PREVENTION

    - MOST SYSTEMS ADOPT SOME FORM OF PREVENTION INSTEAD OF DETECTION

    - PREVENTION SCHEMES NORMALLY TAKE ONE OF TWO FORMS:

      - CHECKING: WHEREBY REQUESTS FOR RESOURCES ARE SCREENED FOR DEADLOCK POTENTIAL PRIOR TO ACCEPTANCE

      - IMPOSED POLICY: WHEREBY REQUESTS FOR MORE THAN ONE RESOURCE <u>MUST</u> BE MADE:

        - TOGETHER AS ONE TOTAL REQUEST BEFORE THE "JOB" OR "JOB-STEP" COMMENCES (ALL OR NOTHING); OR

        - SERIALLY, IN A FIXED, PRE-DEFINED ORDER

DEADLOCK PREVENTION

- MULTICS, IN GENERAL, ADOPTS THE FOLLOWING DEADLOCK PREVENTION SCHEME:

  - USER ASSIGNABLE RESOURCES (SUCH AS TAPE DRIVES, CARD PUNCHES, ETC)

    - WHEN THE USER IS INTERACTIVE NO POLICY IS ENFORCED. THE USER IS INFORMED IF THE RESOURCE IS BUSY AND MAY EITHER TRY AGAIN OR GIVE UP

    - WHEN THE USER IS NOT INTERACTIVE, THE "ALL" OR "NONE" APPROACH SHOULD BE USED. THE AVAILABILITY OF ALL REQUIRED RESOURCES BECOMES THE DETERMINING FACTOR IN SCHEDULING THE USER (SEE THE "RESOURCE CONTROL PACKAGE")

    - SHOULD A NON-INTERACTIVE USER ATTEMPT SERIAL REQUESTS FOR RESOURCES, A DEADLOCK SITUATION COULD POTENTIALLY ARISE AND EXIST UNTIL THE AUTOMATIC LOGOUT DUE TO INACTIVITY OCCURS

  - USER ACCESSIBLE RESOURCES (SUCH AS FILES, DATA BASES, ETC)

    - IN GENERAL, USER SEGMENTS IN THE HIERARCHY POSE NO DEADLOCK PROBLEM SINCE THEY ARE A SIMULTANEOUSLY USABLE RESOURCE (IE: THERE IS NO DEFAULT CONCURRENCY MECHANISM ASSOCIATED WITH USER SEGMENTS)

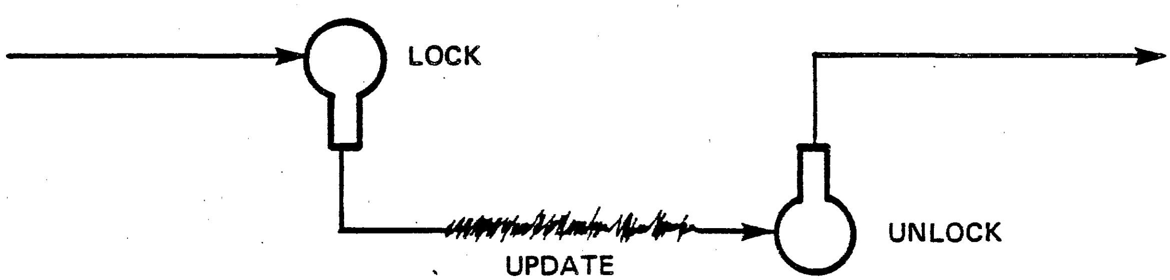
    - SEGMENTS MAY BE PROTECTED FROM POTENTIAL CONCURRENCY PROBLEMS THROUGH USE OF LOCK WORDS AND THE set_lock_ MECHANISM. THIS REQUIRES MUTUAL AGREEMENT AMONG ALL PROCESSES ACCESSING SUCH SEGMENTS

    - SOME SEGMENTS SUCH AS THOSE USED BY THE MULTICS DATA BASE MANAGER (MDBM), USE A "COMMITMENT/ROLLBACK" SCHEME

- SUPERVISOR RESOURCES (SUCH AS HARDCORE DATABASE)
  (traffic lights)

  - LOCKWORDS (OR SIMPLY "LOCKS") ARE USED IN MULTICS TO IMPLEMENT CONCURRENT ACCESS CONTROL IN THE MULTI-PROCESS ENVIRONMENT

LOCKING CONCEPT

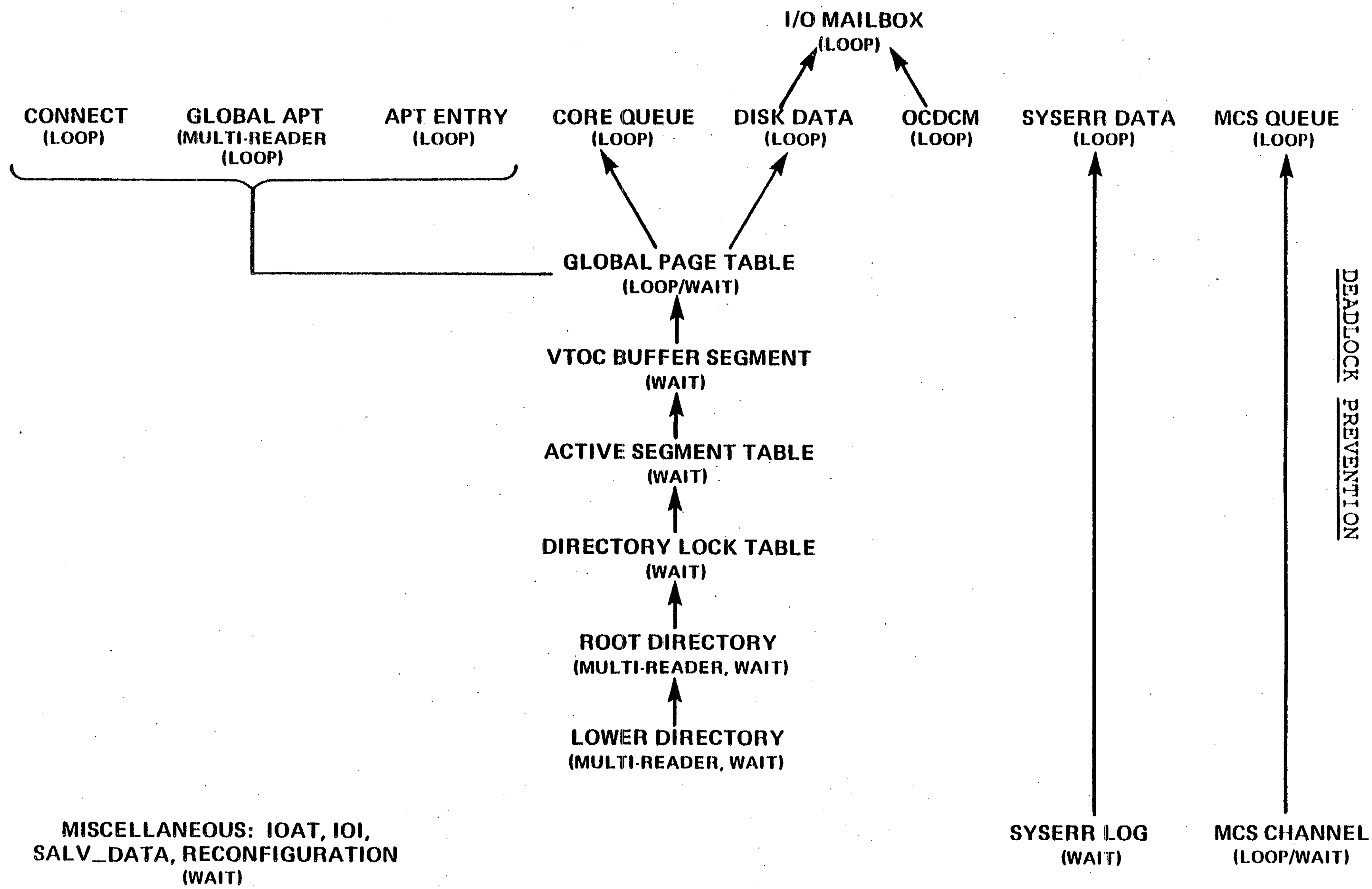


- THE SUPERVISOR LOCKS ARE ARRANGED IN A PARTIAL ORDER AND A CODING CONVENTION PREVENTS WAITING ON A LOCK IF THE PROCESS HAS A HIGHER LOCK LOCKED

- THIS PARTIAL ORDER IS DETERMINED BY AN ANALYSIS OF THE OPERATING SYSTEM'S BEHAVIOR. FOR EXAMPLE: SINCE A PAGE FAULT MAY PROPERLY OCCUR WHILE A PROCESS HAS THE ACTIVE SEGMENT TABLE (AST) LOCKED, AND PAGE FAULT HANDLING REQUIRES THE LOCKING OF THE PAGE TABLE LOCK, THE PAGE TABLE LOCK MUST BE PLACED "HIGHER" IN THE PARTIAL ORDER THAN THE AST LOCK

- TO THE DEGREE THAT THE SYSTEM PROGRAMMERS OBEY THIS PARTIAL ORDER, A DEADLY EMBRACE CANNOT OCCUR WITHIN THE MULTICS SUPERVISOR

MULTICS LOCKING HIERARCHY



TYPES OF LOCKS

- LOCKS WITHIN MULTICS: Protocall

  ie Traffic Light

  - ARE 36 BIT WORDS CONTAINING EITHER ZERO (UNLOCKED) OR A PROCESS_ID (LOCKED)

    Usually near Beginning of Segment

  - CONTROL PROCESSES, NOT PROCESSORS

  - ARE MUTUALLY EXCLUSIVE LOCKS

  - THE HARDWARE SUPPORTS SEVERAL INDIVISIBLE INSTRUCTIONS USED IN IMPLEMENTING THE LOCKING PRIMITIVES. FOR EXAMPLE:

  - STAC (STORE A CONDITIONAL)

    - IF C(Y)=0 THEN C(A) -> C(Y)

    - TYPICAL USE: LOCKING. IF THE LOCKWORD (Y) IS UNLOCKED (=0) THEN LOCK THE LOCK BY STORING THE PROCESS_ID (WHICH IS IN A) INTO THE LOCKWORD

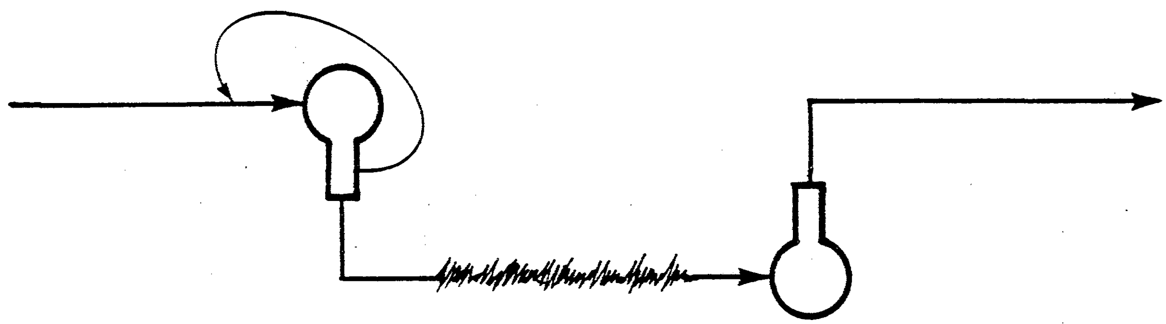
    - SPECIAL HARDWARE PROHIBITS SIMILAR REFERENCES BY OTHER PROCESSORS DURING THE TEST AND DATA TRANSFER WINDOW

  - ALSO STACQ (STORE A CONDITIONAL ON Q), LDAC (LOAD A AND CLEAR), LDQC (LOAD Q AND CLEAR), SZNC (SET Z AND N AND CLEAR)

TYPES OF LOCKS

- WITHIN THE MULTICS SUPERVISOR EXISTS TWO TYPES OF LOCKS: LOOP LOCKS AND WAIT LOCKS

LOOP LOCKS



- SIMPLIFIED PL/I ANALOGY:

```
do while (lockword ^=0);
end;
lockword = process_id;
<update data>
lockword = 0;
```

- LOOP LOCKS ARE USED WHEN IT WOULD NOT BE ACCEPTABLE TO GIVE UP THE PROCESSOR BEFORE LOCKING THE LOCK.

- LOOP LOCKS TYPICALLY PROTECT THE LOWEST LEVEL OF CRITICAL SUPERVISOR DATABASES: TRAFFIC CONTROL, PAGE CONTROL, ETC.

WAIT LOCKS (SIMPLIFIED)

A: PROCESS GIVES UP THE CPU

B: PROCESS DISPATCHED TO A CPU

- SIMPLIFIED PL/I ANALOGY:

```
do while (lockword ^=0);
   <GIVE UP PROCESSOR>
   <WAIT FOR NOTIFICATION>
end;
Lockword = process_id;
<update data>
Lockword = 0
<SEND NOTIFICATION>
```

- MOST SUPERVISOR LOCKS ARE WAIT LOCKS

- IN GENERAL, A PROCESS IS ALLOWED TO GIVE UP ITS PROCESSOR WHEN IT HAS WAIT LOCKS LOCKED

- THE WAIT LOCK MECHANISM WILL BE DESCRIBED IN MORE DETAIL IN TOPIC 9

TOPIC IV

Name Space and Address Space Management

NAME/ADDRESS SPACE OVERVIEW

All online info is Processor Addressable
They are made processor addressable as they are referenced

- FUNCTION

  - IMPLEMENT THE PER PROCESS VIRTUAL MEMORY

    Known Segs

- BASIC PHILOSOPHY

  - AS A NEWLY LOGGED IN USER ATTEMPTS TO TOUCH VARIOUS SEGMENTS A CONSIDERABLE AMOUNT OF MANAGEMENT INFORMATION MUST BE (TRANSPARENTLY) FOUND AND/OR COMPUTED BEFORE THE USER'S REFERENCE IS ACTUALLY ACCOMPLISHED

  - FOR EVERY SEGMENT REFERENCED BY THE USER, THE SUPERVISOR:

    - ASSIGNS A SEGMENT NUMBER (FOR REASON OF HARDWARE ADDRESSING), AND

    - RECORDS (REMEMBERS) THE MANAGEMENT INFORMATION (FOR REASON OF SOFTWARE EFFICIENCY AND CONTROL)

  - SUCH SEGMENTS ARE SAID TO BE "KNOWN TO THE PROCESS"

  - THE MANAGEMENT INFORMATION IS MAINTAINED ON A PER PROCESS BASIS IN THREE COMPLEMENTING AREAS: DSEG, KST, AND RNT

NAME/ADDRESS SPACE OVERVIEW

- MANAGES TWO DISTINCT SETS OF INFORMATION:

  - ADDRESS SPACE - CORRESPONDENCE BETWEEN SEGMENT NUMBERS AND THE SEGMENTS THEMSELVES

  - NAME SPACE - CORRESPONDENCE BETWEEN SEGMENT NUMBERS AND NAMES THE USER REFERS TO THEM BY
    initiated segs

- CALLS DIRECTORY CONTROL TO LOCATE SEGMENTS INITIALLY

- NAME SPACE / ADDRESS SPACE MANAGEMENT IS INVOKED BY SUBROUTINE CALLS, AND BY LINKAGE FAULTS (THE "DYNAMIC LINKER")

- PRINCIPAL USER INTERFACES

  - COMMAND LEVEL

    - initiate, terminate, terminate_segno, terminate_ref_name, terminate_single_ref_name, list_ref_name

    - THE COMMAND PROCESSOR ITSELF - WHICH USES THESE SERVICES TO LOCATE COMMANDS

  - SUBROUTINE LEVEL

    - hcs_$initiate, hcs_$initiate_count_, hcs_$terminate_file, hcs_$terminate_seg, hcs_$terminate_name, hcs_$terminate_noname, term_

- MAJOR DATA BASES

  array of SDW'S

  - DESCRIPTOR SEGMENT (DSEG) - ONE PER PROCESS

    - SEGMENT DESCRIPTOR WORD (SDW) - ONE PER KNOWN SEGMENT

    - DEFINES THE USER'S ADDRESS SPACE TO THE HARDWARE

  - KNOWN SEGMENT TABLE (KST) - ONE PER PROCESS

    - KNOWN SEGMENT TABLE ENTRY (KSTE) - ONE PER KNOWN SEGMENT (EXCEPT SUPERVISOR SEGMENTS)

    - DEFINES THE USER'S ADDRESS SPACE TO THE SUPERVISOR AND THE USER

    - EACH KSTE ASSOCIATES A USER'S SEGMENT NUMBER WITH THE SEGMENT CONTROL ATTRIBUTES OF THAT SEGMENT

    - THE SEARCH FOR AN AVAILABLE KSTE DETERMINES A SEGMENT'S NUMBER

  - REFERENCE NAME TABLE (RNT) - ONE PER EACH RING IN EACH PROCESS

    - NOT A SEGMENT - KEPT AS A REGION ALLOCATED IN THE "LINKAGE AREA" FOR EACH RING

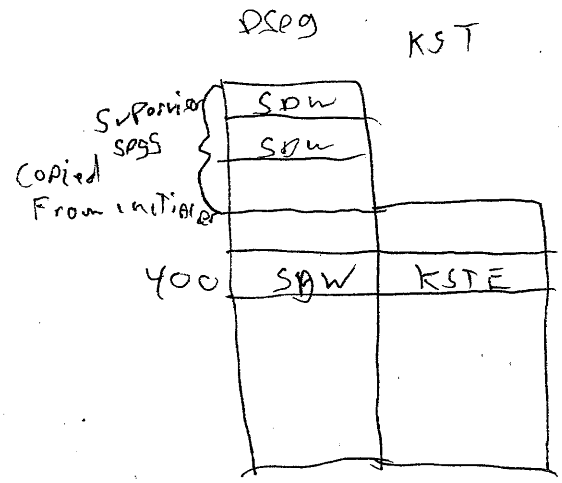
    - REFERENCE NAME TABLE ENTRY (RNTE) - ONE PER REFERENCE NAME

    - USED BY THE DYNAMIC LINKER TO IMPLEMENT THE "initiated_segments" SEARCH RULE

NAME/ADDRESS SPACE OVERVIEW

- DEFINES THE USER'S NAME SPACE TO THE USER

- NAME SPACE MAY BE DIFFERENT IN DIFFERENT RINGS OF THE SAME PROCESS



NAME/ADDRESS SPACE TERMINOLOGY

SEGMENT DESCRIPTOR WORD (SDW):
A TWO WORD PAIR USED BY THE HARDWARE WHEN REFERENCING A SEGMENT.

DESCRIPTOR SEGMENT (DSEG):
THE MOST FUNDAMENTALLY IMPORTANT SEGMENT IN A PROCESS. CONTAINS AN ARRAY OF SDW'S DEFINING THE ADDRESS SPACE OF THE PROCESS

ADDRESS SPACE:
THE SET OF ALL SEGMENTS (PROCEDURE AND DATA) FOR WHICH THE PROCESS HAS A SEGMENT NUMBER AND A CORRESPONDING SDW. THE ADDRESS SPACE EXPANDS AND CONTRACTS DURING A SEGMENT'S LIFE

~~KST~~ S

12 BITS

IF YOU ADJUST KST Also Adjust IOT Size.

KST Size in SAT + PDT default values

hardware limit is 4096

SEGMENT NUMBER:
AN OCTAL NUMBER 0-1777 (0-1023 DECIMAL) ASSIGNED UNIQUELY TO A SEGMENT. USED BY THE HARDWARE AS AN OFFSET INTO THE ARRAY OF SDW'S WHEN REFERENCING A SEGMENT

MAKING KNOWN:
THE ACT OF ASSIGNING A SEGMENT NUMBER TO A SEGMENT, THEREBY ADDING IT TO THE ADDRESS SPACE. SEGMENTS MUST BE MADE KNOWN BEFORE THEY CAN BE REFERENCED

NAME/ADDRESS SPACE TERMINOLOGY

NAME SPACE:

THE SET OF ALL SEGMENTS FOR WHICH THE PROCESS HAS A REFERENCE NAME. THE REFERENCE NAME MAY BE DIFFERENT THAN THE SEGMENTS ACTUAL NAME (ITS ENTRYNAME). SINCE SOME SEGMENTS IN THE ADDRESS SPACE HAVE NO REFERENCE NAME, THE NAME SPACE IS A PROPER SUBSET OF THE ADDRESS SPACE

NAME/ADDRESS SPACE CONCEPTS

- A MULTICS PROCESS IS A WELL DEFINED COLLECTION OF UNIQUE SEGMENTS, EACH WITH DEFINED ACCESS, OVER WHICH A SINGLE EXECUTION POINT IS FREE TO ROAM (I.E., FETCH INSTRUCTIONS AND MAKE DATA REFERENCES)

- THE ADDRESS SPACE OF A PROCESS IS THE ABOVE "COLLECTION OF SEGMENTS". SUCH SEGMENTS ARE SAID TO BE KNOWN TO THE PROCESS

  - ALL SUPERVISOR (RING 0) SEGMENTS ARE PLACED INTO THE ADDRESS SPACE AT PROCESS CREATION TIME. THIS ADDRESS SPACE IS SAID TO BE "CLONED" FROM THE INITIALIZER ADDRESS SPACE

    - THE RING ZERO ADDRESS SPACE IN ANY PROCESS IS THE SAME AS THE INITIALIZER'S RING ZERO ADDRESS SPACE, EXCEPT FOR THE DSEG, KST, PDS, PRDS, AND STACK_0

    - THE RING ZERO ADDRESS SPACE HAS NO RNT, BECAUSE IT IS SET UP DURING SYSTEM INITIALIZATION, AND DOES NOT CHANGE
      RNT used during dynamic linking only

  - OTHER SEGMENTS ARE MADE KNOWN AND UNKNOWN DURING THE LIFE OF THE PROCESS

    - IMPLICITLY BY THE DYNAMIC LINKER (LINKAGE FAULT) OR A SYSTEM COMMAND

      ```
      print my_dir>my_seg

      tester
      ```

    - EXPLICITLY BY COMMANDS OR SUBROUTINES THAT MANAGE THE ADDRESS SPACE

      ```
      initiate my_prog call hcs_$terminate_segno
      ```

NAME/ADDRESS SPACE CONCEPTS

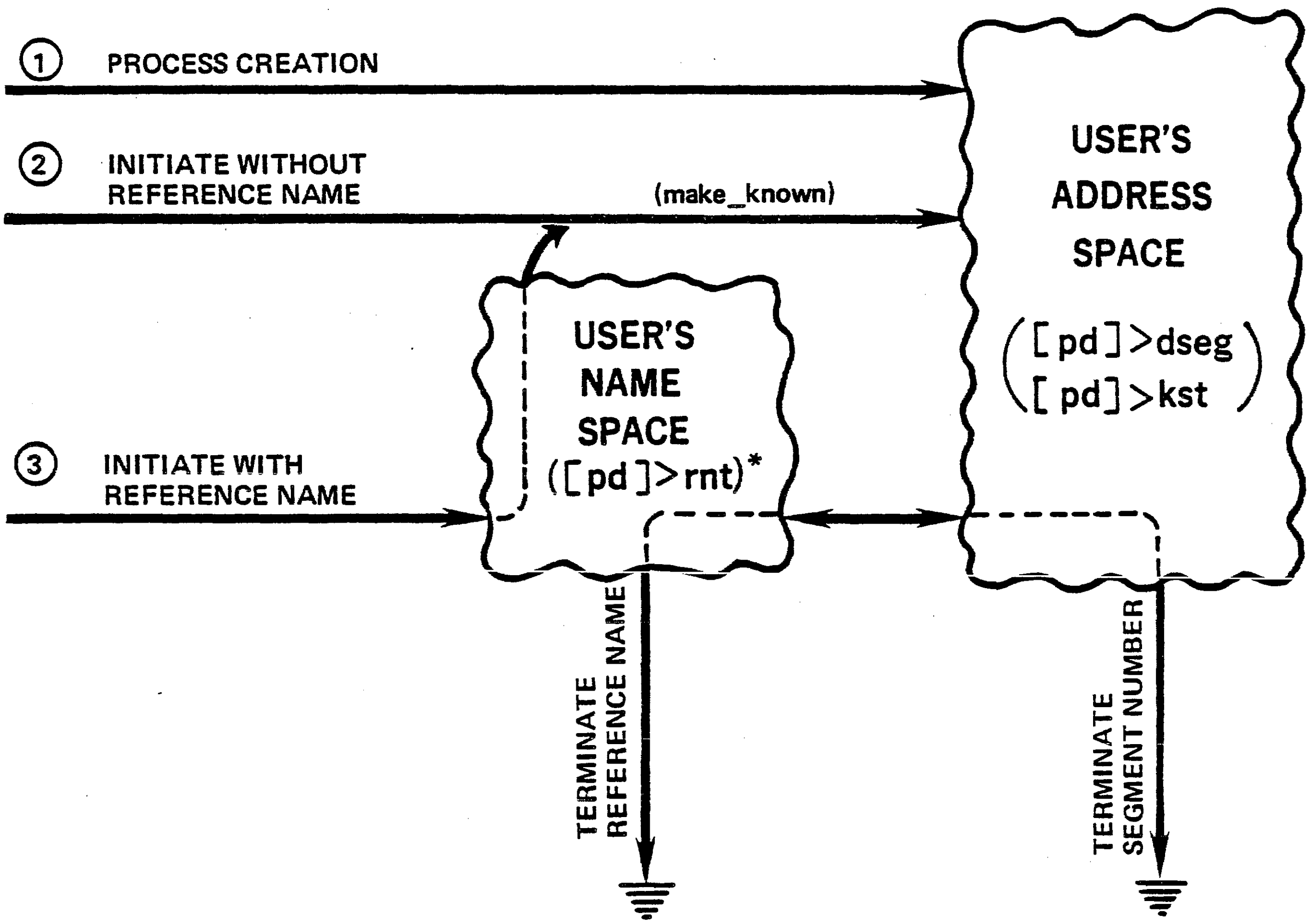
- MAKING A SEGMENT KNOWN IS SIMILAR TO DECLARING A VARIABLE IN A PL/I PROGRAM. IT SIGNIFIES INTENT, BUT NOT USAGE
  reserves slot in KST

- THE PRESENCE OF ONE OR MORE PAGES OF A SEGMENT IN MAIN MEMORY IMPLIES THAT THE SEGMENT IS KNOWN TO (AND IS BEING USED BY) AT LEAST ONE USER

- BEING KNOWN DOES NOT IMPLY PRESENCE IN MAIN MEMORY

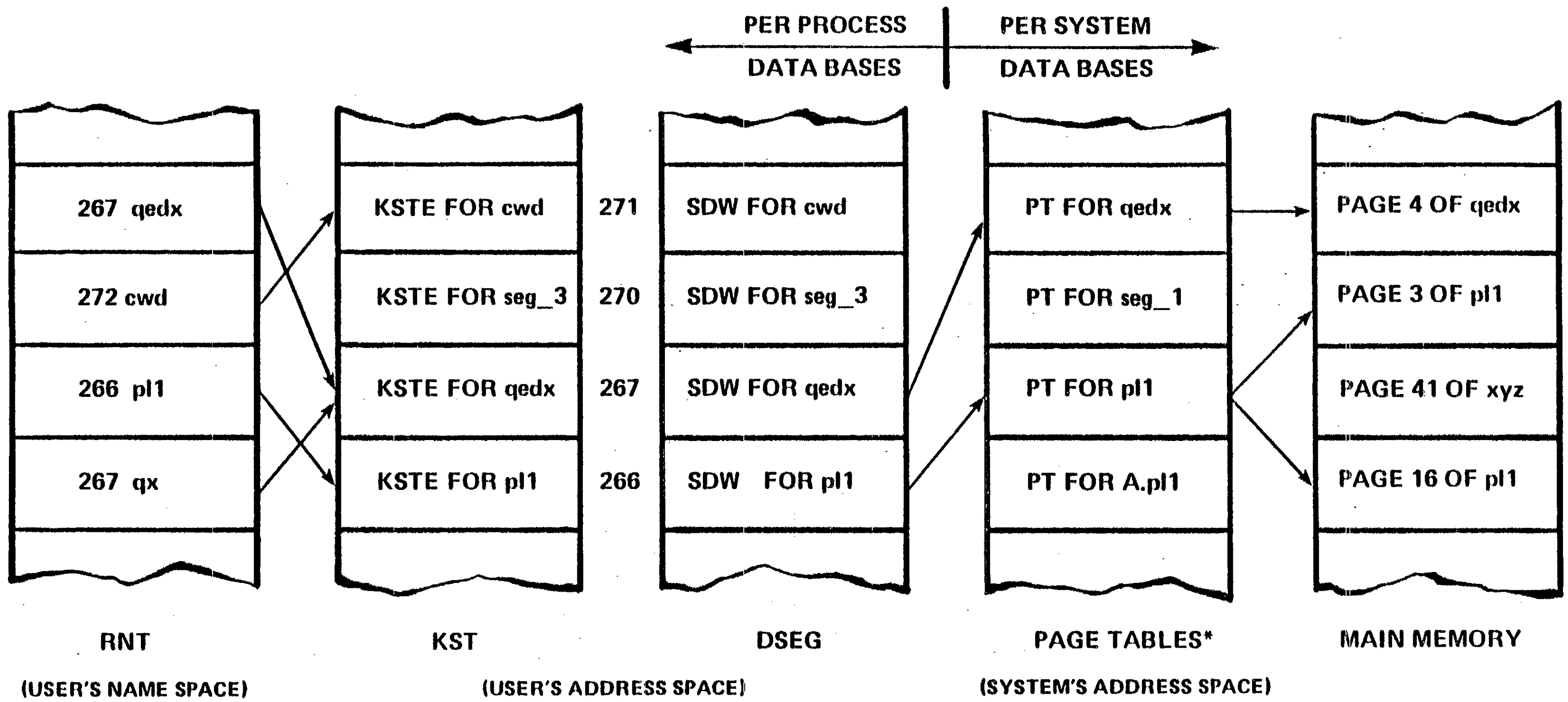
- NOTE THAT THIS SET OF SEGMENTS, THE EXECUTION POINT, AND THE REGISTERS AND INDICATORS OF THE PROCESSOR, UNIQUELY DEFINES THE STATE OF THE PROCESS

NAME SPACE AND ADDRESS SPACE MANIPULATION



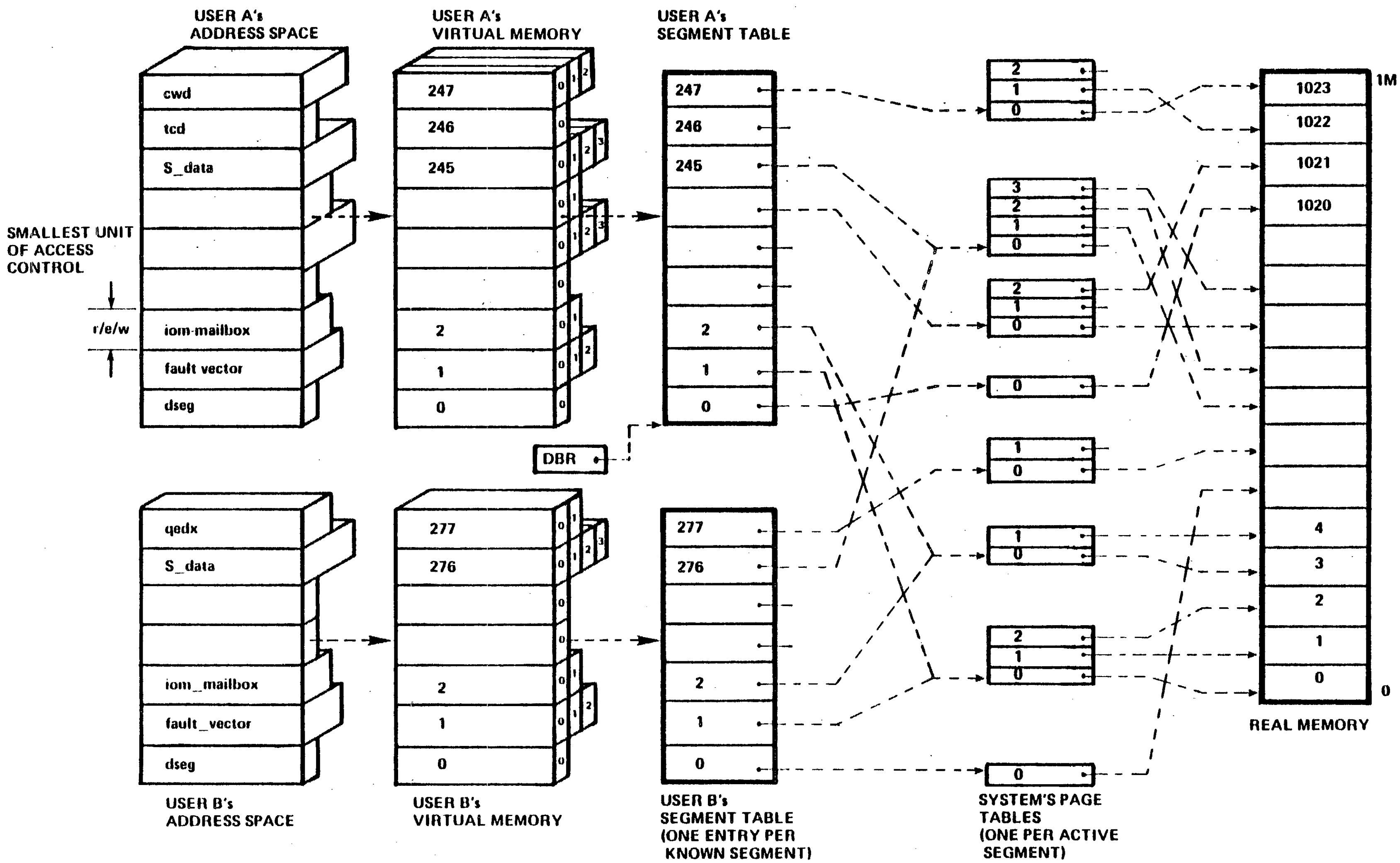
* THE rnt IS AN AREA WITHIN [pd]>[unique].area.linker

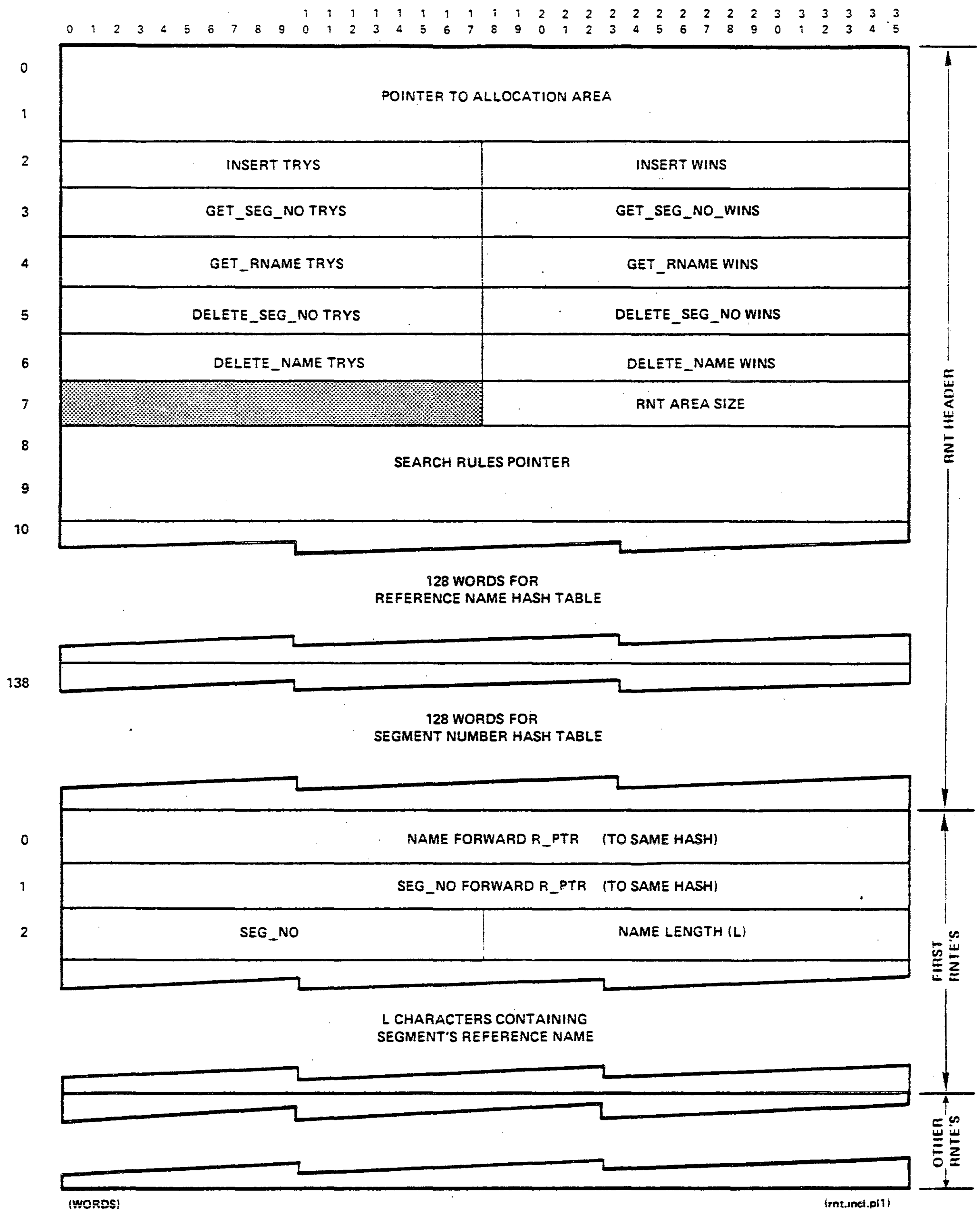
NAME SPACE AND ADDRESS SPACE MANAGEMENT



* THE ASTE'S ASSOCIATED WITH THE PAGE TABLES ARE NOT SHOWN IN THIS DIAGRAM

MULTICS VIRTUAL MEMORY STRUCTURE





REFERENCE NAME TABLE (RNT)

A PAGED DATA BASE ([pd] > rnt) - ONE PER ACTIVE PROCESS

"INITIATED SEGMENTS" IN SEARCH RULES

REFERENCE NAME TABLE ENTRY (RNTE)

ONE PER REFERENCE NAME

NAME/ADDRESS SPACE DATA BASES

KNOWN SEGMENT TABLE (KST)

OCTAL

r_ptr
relative ptr or
offset of ptr
giving address
inside KST

seg only made
unknown when
Times initiated
= 0.

KST HEADER

Word	Bits 0–17	Bits 18–35
0		LOWEST SEGMENT NUMBER
1		HIGHEST SEGMENT NUMBER
2		HIGHEST SEGMENT NUMBER YET USED
3		NUMBER OF PRIVATE LOGICAL VOLUMES
4–5	TIME OF BOOTLOAD	
6	NO. OF KST GARBAGE COLLECTIONS	NO. OF KST ENTRIES RECOVERED
7	FIRST FREE KSTE R_PTR	R1 R2 R3 R4 R5 R6 R7 TEMP 256K

64 WORD ARRAY FOR UID HASH TABLE

FIRST KSTE

0	FORWARD R_PTR		SEGMENT NUMBER	
1	NUMBER	OF	TIMES	THIS
2	SEGMENT	INITIATED	(PER	RING)
3	BRANCH ENTRY POINTER			
4	SEGMENT'S UNIQUE IDENTIFIER (UID)			
5	DATE TIME BRANCH ENTRY MODIFIED (DTBM)			
6	EXTENDED ACCESS			R E W
7	W RING, R RING, E RING, H RING	DIR, WRIT, PRIV, TMS, TUS, AUD, EXPL	INF COUNT OR LV INDEX	

(Calculated from aim & ACL)

OTHER KSTE's
ONE PER INITIATED SEGMENT

256 WORD LIST OF PRIVATE LOGICAL
VOLUME CONNECTIONS

KST TAIL

END OF KST

(WORDS) kst.incl.pl1

KNOWN SEGMENT TABLE (KST)

A PER PROCESS DATA BASE (pds$kst) ONE PER ACTIVE PROCESS
NOT WIRED (ENCACHEABLE)

KNOWN SEGMENT TABLE ENTRY (KSTE)

ONE PER INITIATED SEGMENT IN PROCESS

15 18

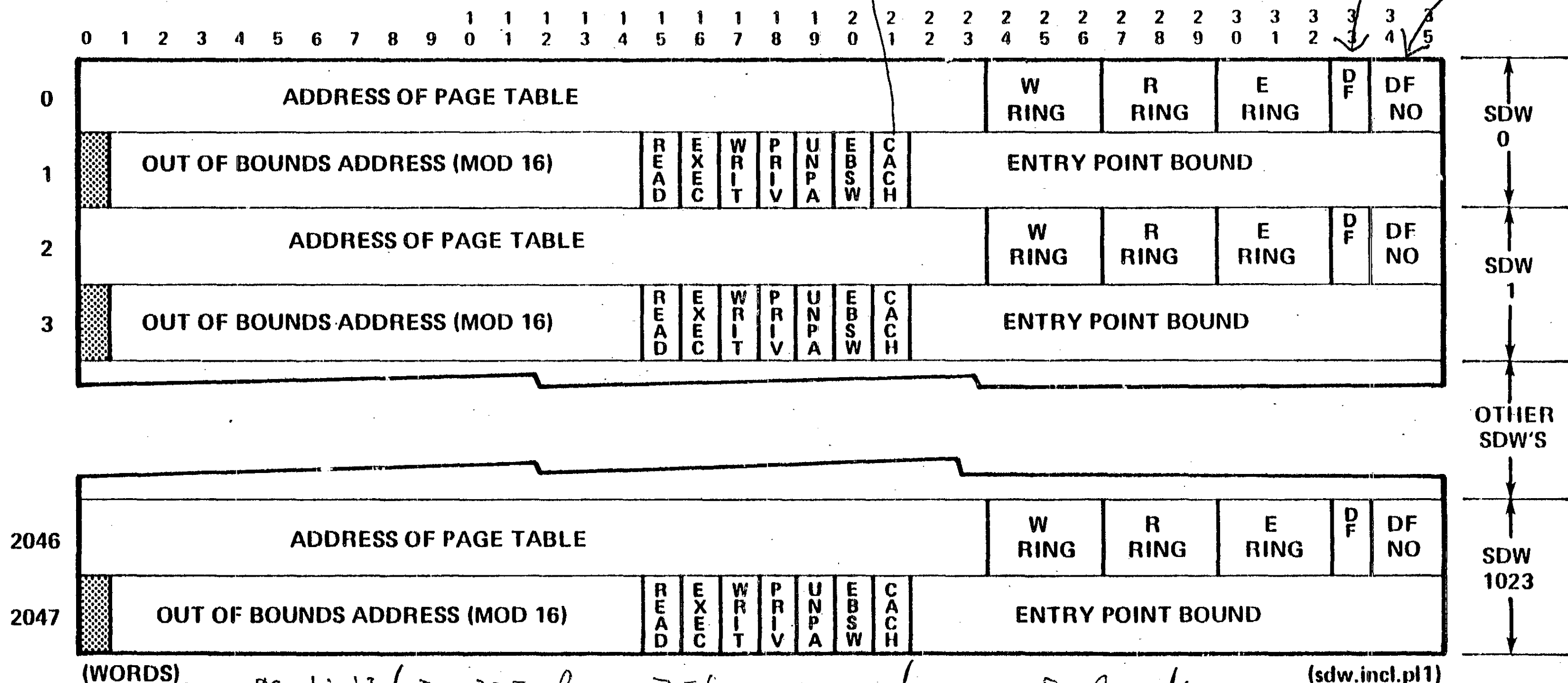
NAME/ADDRESS SPACE DATA BASES

DESCRIPTOR SEGMENT (DSEG)

1 - we have address of page table
0 - causes seg fault
Valid page table bit

0 directed fault
1
2
3 } not used

Used with write notify cable
L68's use this bit
cause no cable.
DPS8's ignore it.



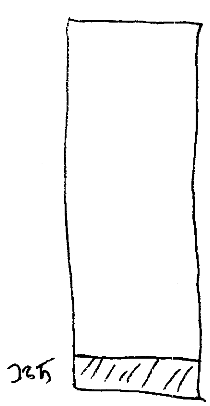
segments limited to 255 pages, 256th marked out of bounds.

DESCRIPTOR SEGMENT (DSEG)

A PAGED ([pd] > dseg) DATA BASE – ONE PER ACTIVE PROCESS

SEGMENT DESCRIPTOR WORD (SDW)

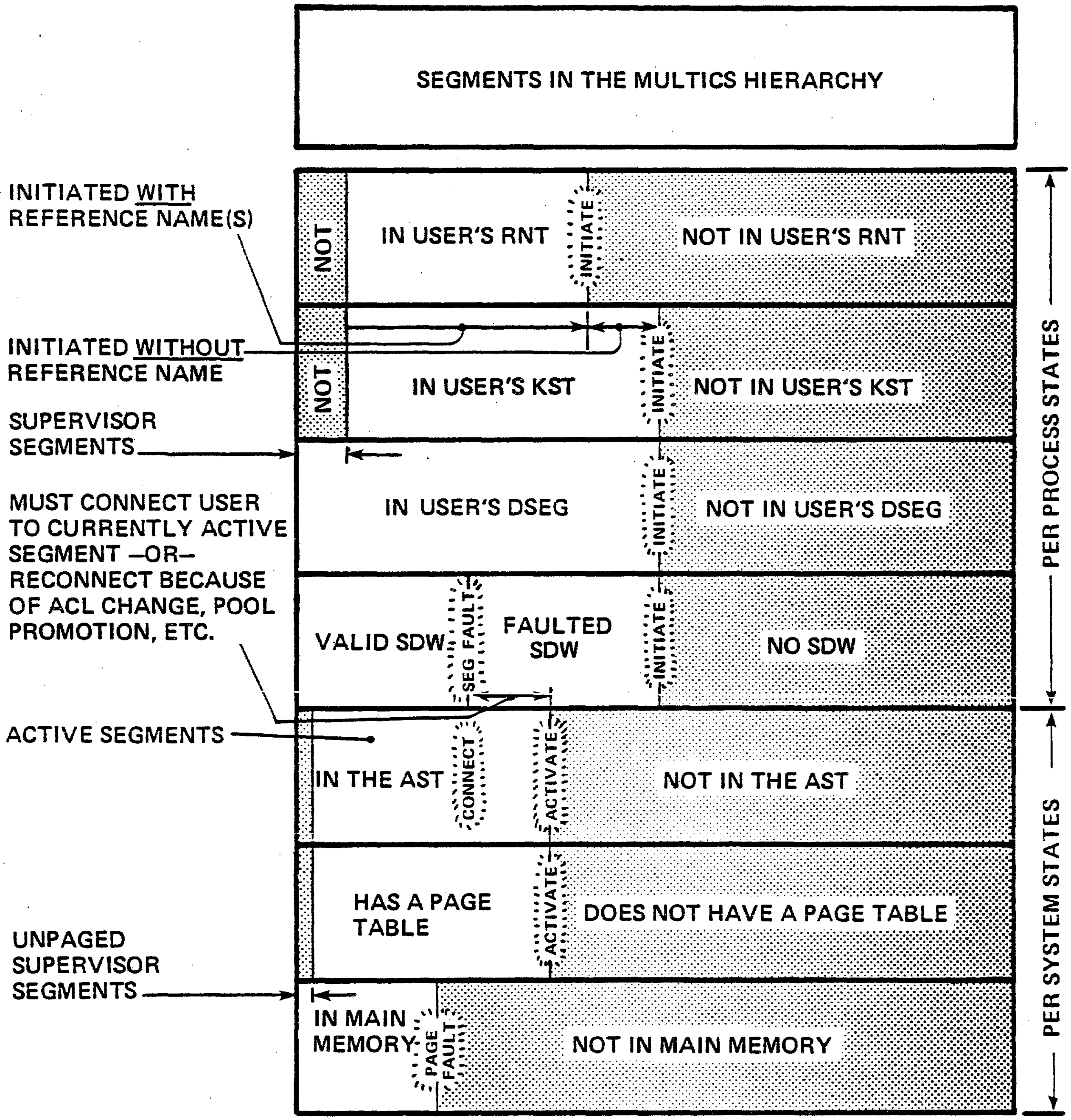
ONE PER INITIATED SEGMENT IN PROCESS (MAX 1024)



Small negative offsets ie: Bad programs, causes wrap around.

set_max length

DESCRIPTOR SEGMENT (DSEG)



STATES OF SEGMENTS AND PAGES

TYPICAL ADDRESS SPACE

- COLLECTION ZERO

  - SEGMENTS WHICH MUST BE PRESENT TO RUN THE VERY FIRST LOADING PROGRAM

  - SEGMENTS WHICH HAVE FIXED ABSOLUTE ADDRESSES TO INTERFACE WITH HARDWARE

  - ALL DESCRIBED IN template_slt_.cds

0 [pd]>dseg (ring 0)
The descriptor segment. The Initializer's dseg comes from the system tape and is built during system initialization; all others are created by process creation.

1 fault_vector (ring 0, perm-wired)
Contains the interrupt vector, fault vector, and the ITS pairs for SCU and TRA instructions. Located at absolute locations 0-577. Used by the CPU hardware.

2 iom_mailbox (ring 0, perm-wired)
Mailboxes (communications areas) for up to four IOMs. Located at locations 1200-3377 absolute. Used by the IOM hardware.

3 >sl1>config_deck (ring 0, deciduous)
The online copy of the config deck. This is built from the config deck provided by BOS during system initialization, but it is not the copy BOS actually uses.

4 dn355_mailbox (ring 0)
Mailboxes (communications areas) for up to eight FNPs. Located at absolute locations 3400-6377. Used by the FNP hardware.

5 bos_toehold (ring 0, perm-wired)
The segment containing the tiny program used to switch between Multics and BOS at crash time. Located at absolute locations 10000-11777.

6 flagbox (ring 0, perm-wired)
A region inside the bos_toehold segment (yes, it really overlaps the toehold) used to access the BOS/Multics communication region.

7 >sl1>slt (ring 0, deciduous)
10 >sl1>name_table (ring 0, deciduous)
The two primary databases of system initialization. The SLT contains one entry for every supervisor segment read from the system boot tape, containing all its attributes. The separate name_table is used to hold the names, because each segment may have several

This marks the end of Collection Zero. All the rest of the segments in the address space are either read from the system tape or found in the online system.

TYPICAL ADDRESS SPACE

- COLLECTION ONE

  - FIRST BATCH OF SEGMENTS READ FROM SYSTEM TAPE

  - ALL THE PROGRAMS AND DATABASES NEEDED TO MAKE PAGING RUN

  - ALL SEGMENTS WHICH MUST BE "perm-wired" -- PERMANENTLY ALLOCATED IN LOW MEMORY, WITHOUT PAGE TABLES

11 lot (ring 0)
The supervisor's linkage offset table. Used to find linkage sections. Built as the segments are read in from tape.

12 as_linkage (ring 0)
13 ws_linkage (ring 0)
The permanent supervisor combined linkage regions. The names mean "Active Supervisor Linkage" and "Wired Supervisor Linkage", respectively. The linkage sections of all permanent supervisor segments are put in one of these as the segments are read from the tape.

14 >sl1>definitions_ (ring 0, deciduous)
The segment containing all the definitions sections of supervisor programs. The definitions sections are placed here as the programs are read from the tape, and used by the hardcore prelinker.

15 sst_seg (ring 0, perm-wired)
The segment containing all ASTEs and page tables. Covered in under Page and Segment Control. This segment is allocated at the very top end of the bootload SCU.

16 core_map (ring 0, perm-wired)
All the core map entries, describing all system memory. Covered under Page Control. This used to be part of the SST, but was moved out to make more room for page tables.

17 abs_seg (ring 0, abs-seg)
An abs-seg used for complex call-side operations in

page control, such as evict_page and reconfiguration.

20 abs_seg1 (ring 0, abs-seg)
An abs-seg used only by page control for checking page frame contents for zeros, and zeroing newly allocated page frames.

21 backup_abs_seg (ring 0, abs-seg)
An abs-seg used to access the segment being dumped in a Volume Dumper process. It is given an SDW which refers to any ordinary segment. This is a very special hardcore segment, because it has trailer entries, and it is special-cased by the trailer manipulation program, setfaults. Normal hardcore segments never receive trailers, since they are never activated or deactivated.

22 fim_abs_seg (ring 0, abs-seg)
An abs-seg used by the FIM to do something

23 isolts_abs_seg (ring 0, abs-seg)
An abs-seg used by ISOLTS, which gives it an SDW describing the low 64K of the SCU being used for ISOLTS testing.

24 volmap_abs_seg (ring 0, abs-seg)
An abs-seg used by page control to access record stocks. It is given the SDW of whichever volume's stock is needed.

25 bound_active_1 (ring 0)
26 bound_disk_util (ring 0)
27 bound_disk_util_wired (ring 0, perm-wired)
30 bound_error_active (ring 0)
31 bound_error_wired (ring 0, perm-wired)
32 bound_interceptors (ring 0, perm-wired)
This is where the FIM lives.

33 bound_io_wired (ring 0, perm-wired)
34 bound_iom_support (ring 0, perm-wired)
35 bound_page_control (ring 0, perm-wired)
36 bound_priv_1 (ring 0, perm-wired)
37 >sl1>bound_sss_wired_ (ring 0, deciduous)
This is where pl1_operators_ and a whole host of other miscellaneous subroutines live. It is wired in a very special way, because not all of its contents need to be wired. In particular, only about half of pl1_operators_ needs to be wired, and there is a special hack in make_sdw.pl1 which finds the definition in the middle of pl1_operators_ which

marks the end of the wired portion, and makes an ASTE for bound_sss_wired_ which has all its pages up to and including that definition wired, and the rest unwired.

40 bound_tc_priv (ring 0, perm-wired)
41 bound_tc_wired (ring 0, perm-wired)
42 bound_unencacheable (ring 0, perm-wired)
43 dir_seg (ring 0, abs-seg)
An abs-seg now used only at process termination time to loop through the dead process's KST in order to flush any trailers it had for active segments.

44 disk_post_queue_seg (ring 0, perm-wired)
The segment where the core address queue lives: see core_queue_man.alm. This is discussed under Page Control.

45 disk_seg (ring 0, perm-wired)
The segment containing the disk DIM's databases: device table, channel table, and I/O queues.

46 dn355_data (ring 0, perm-wired)

The segment containing software communications regions for the FNPs. This is not where FNP buffers are kept, but only the mailboxes that describe the buffers, and some control information.

47 ds_seg (ring 0, abs-seg)
The segment used by setfaults when accessing another process's DSEG in order to remove a trailer. Covered under Segment Control.

50 emergency_shutdown (ring 0, perm-wired)
The procedure segment which starts an ESD. It is a separate segment because BOS has to be able to find it and transfer to it.

51 hardcore_sct_seg (ring 0)
The segment containing the static condition table for ring zero; it's just like the one which is kept in an outer ring stack header. The only static handlers in ring zero are those used to invoke the copy_on_write mechanism.

52 idle_dsegs (ring 0, perm-wired)
53 idle_pdses (ring 0, perm-wired)
Two similar segments: they contain the DSEG and PDS segments for all the idle processes, all in a row. When an idle process is constructed, the SDWs for

its DSEG and PDS are set up to point into the middle of one of these segments.

54 init_processor (ring 0)

55 inzr_stk0 (ring 0)
The stack segment used by the Initializer during initialization and shutdown. During normal operation, the Initializer participates in ordinary ring zero stack sharing.

56 iobm_data (ring 0, perm-wired)
The database of iobm.pl1, the I/O Buffer Manager.

57 ioi_abs_seg (ring 0, abs-seg)
The abs-seg used by ioi_interrupt.pl1 to access a user ioi_ buffer at interrupt time, for storing status information.

60 ioi_data (ring 0, perm-wired)
The database for ioi_ -- describes all user-accessable, or potentially user-accessable devices. See the programs in bound_io_active and bound_io_wired.

61 iom_data (ring 0, perm-wired)
Describes the configuration of the IOMs, and software information about IOM channels. Contains assignment information, software status queue location, and metering cells.

62 oc_data (ring 0, perm-wired)
The database for the ring zero operator's console mechanism. This is used by syserr in ring zero, and by the Initializer to write on the system console (but not message coordinator consoles).

63 [pd]>pds (ring 0, deciduous)
The Process Data Segment. This contains all the miscellaneous information that makes a process unique to the supervisor, and need not be readily accessable to other processes. Most per-process variables are referenced symbolically, such as pds$processid, pds$page_fault_data (machine conditions for last page fault), etc.

64 >sl1>prds (ring 0, deciduous)
Like the PDS, but per-processor. Contains the same sort of miscellaneous information, and is also used as the ring zero stack for certain types of faults (page faults, connects, and timer runouts) and all interrupts. There is a PRDS per processor, named

>sl1>cpu_A.prds, cpu_B.prds, etc. The PRDS segment in ring zero is changed at LDBR time to indicate the actual PRDS of the processor that the process is going to run on; thus, it's sort of an abs-seg. All PRDS's are wired, but have page tables.

65 >sl1>pvt (ring 0, deciduous)
The Physical Volume Table. Described unver Volume Management and Page Control.

66 rdisk_seg (ring 0, abs-seg)
A PTW-type abs-seg (the only one where the PTW ever changes). Used only by the program read_disk, which does I/O to arbitrary pages on any disk, its aste.pvtx and PTW are switched around to indicate the right page, which is faulted on and (if needed) written back out by pc$cleanup.

67 restart_fault (ring 0)
70 return_to_ring_0_ (ring 0)
These two procedures are used to implement the restarting of faults (such as a QUIT signal) from the user ring. When a fault occurs, a frame is pushed on the user ring stack, with its owner set to be return_to_ring_0_. Additionally, the machine conditions for the fault are saved in ring zero (in the PDS) so that when restart_fault is called to restart a possibly modified set of conditions, it can compare and validate.

71 scas (ring 0, abs-seg)
The System Controller Addressing Segment. This segment has a page overlaid on a page of every system controller. No data is ever accessed through this segment; its page table is not even in the SST, but in the SCS. It is used only for certain privileged instructions which require an effective address in an particular SCU in order to read or set control registers in the SCU.

72 scs (ring 0, perm-wired)
The System Configuration Segment. It describes most of the hardware configuration, and contains various control words used by privileged control instructions.

73 signaller (ring 0)
The procedure which implements user ring fault signalling.

74 >sl1>sst_names_ (ring 0, deciduous)
The SST name table. This is a debugging feature; it contains (when in use) the primary name corresponding to every segment in the AST. It can be maintained online, by use of the PARM ASTK config card, but usually is not. It is always filled in by FDUMP after a crash if it was not already in use.

75 stack_0_data (ring 0, perm-wired)
Data segment describing the available segments for use in ring zero stack sharing. These segments, seen later on as segment 230, are recorded here and multiplexed among eligible processes.

76 stock_seg (ring 0, perm-wired)
The segment containing all in-core record and VTOCE stocks for use by page control and segment control, and covered under those topics.

77 >sl1>sys_boot_info (ring 0, deciduous)
A data segment containing information about the I/O devices (tape and disk) used during bootload.

100 >sl1>sys_info (ring 0, deciduous)
Contains assorted global wired information shared by the user ring and supervisor. Some is set during bootload, and some comes off the tape; none is modified after initialization is complete.

101 syserr_data (ring 0, perm-wired)
Data segment for the lowest level of the syserr mechanism. Syserr messages are built and queued here, and sent to the console. They are also copied out by the syserr logger hardcore process, into the syserr_log.

102 syserr_log (ring 0, abs-seg)
This segment overlays the LOG partition on some disk, which is used to reliably store syserr messages until they can be copied into the perm_syserr_log maintained in the Hierarchy.

103 tc_data (ring 0)
The traffic control data segment; contains all traffic control data. Covered under Traffic Control.

104 wired_hardcore_data (ring 0, perm-wired)
Miscellaneous data used by the wired supervisor.

- COLLECTION TWO

I THE UNWIRED PORTION OF THE SUPERVISOR

I READ IN BY COLLECTION ONE, DIRECTLY INTO PAGED SEGMENTS IN THE HARDCORE PARTITIONS

105 >sl1>active_all_rings_data (ring 0, deciduous)
Miscellaneous data shared between the unwired supervisor and the outer rings.

106 active_hardcore_data (ring 0)
Miscellaneous data used by the unwired portion of the supervisor: system-wide locks, size constants for directory control, system search rule info, and metering for directory control and the dynamic linker.

107 >sl1>admin_gate_ (ring 0, deciduous)
110 ast_lock_meter_seg (ring 0)
A segment used to collect AST lock metering, normally off (enabled by the ast_lock_metering tuning parameter).

111	>sl1>audit_gate_	(ring 0, deciduous)
112	bound_355_wired	(ring 0)
113	bound_file_system	(ring 0)
114	bound_hc_backup	(ring 0)
115	bound_hc_reconfig	(ring 0)
116	bound_hc_tuning	(ring 0)
117	bound_imp_dim_	(ring 0)
120	bound_imp_status	(ring 0)
121	bound_io_active	(ring 0)
122	bound_mcs_util	(ring 0)
123	bound_priv_mpx	(ring 0)
124	bound_network0_	(ring 0)
125	bound_priv_procs	(ring 0)
126	bound_process_creation	(ring 0)
127	bound_salvager	(ring 0)
130	bound_scavenger	(ring 0)
131	>sl1>bound_sss_active_	(ring 0, deciduous)
132	bound_system_faults	(ring 0)
133	bound_tty_active	(ring 0)
134	bound_vtoc_man	(ring 0)
135	bound_x25_mpx	(ring 0)

136 dbm_seg (ring 0)
The segment used to hold the dumper bit maps for volumes being volume-dumped. The bitmaps are read from the volume header when a dump begins, and written back when finished.

137 dirlockt_seg (ring 0)
The segment used to keep track of all directory locks. Directory locks are kept in a supervisor segment, rather than in directories themselves.

140 >sl1>dm_hcs_ (ring 0, deciduous)
141 >sl1>dm_journal_seg_ (ring 0, deciduous)
142 >sl1>error_table_ (ring 0, deciduous)
143 fnp_dump_seg (ring 0)
Segment used for data buffering by FNP dump and patch operations (but not bootload).

144 hasp_mpx (ring 0)
145 >sl1>hc_backup_ (ring 0, deciduous)
146 >sl1>hcs_ (ring 0, deciduous)
147 >sl1>hphcs_ (ring 0, deciduous)
150 ibm3270_mpx (ring 0)
151 imp_data (ring 0)
152 imp_dim_buf_ (ring 0)
153 imp_tables (ring 0)
154 imp_wired_buffers (ring 0)
These four segments were used by the ring zero IMP DIM (part of the ARPAnet support), which has since been decomissioned.

155 initializer_abs_seg (ring 0, abs-seg)
An abs-seg used solely in order to copy a process's stack_0 segment into its process directory on process termination.

156 >sl1>initializer_gate_ (ring 0, deciduous)
157 io_page_tables (ring 0)
The segment which contains page tables used for I/O if the IOM is operating in Paged mode. It is initialized by ioi_init and used only for ioi_ I/O.

160 ioat (ring 0)
The I/O attach table. This is a largely obsolete database, the relic of an earlier I/O device attachment scheme. It is now used only for loading and dumping FNPs.

161 >sl1>ioi_ (ring 0, deciduous)
62 [pd]>kst_seg (ring 0, deciduous)

The Known Segment Table. Described in Name & Address Space Management.

163 lvt (ring 0)
The Logical Volume Table. Described in Volume Management.

164 >sl1>mhcs_ (ring 0, deciduous)
165 ncp_tables_ (ring 0)
Another part of the now-decommissioned ring zero ARPAnet support, no longer used.

166 >sl1>net_ring0_admin_ (ring 0, deciduous)
167 >sl1>net_ring0_sys_ (ring 0, deciduous)
170 >sl1>net_ring0_user_ (ring 0, deciduous)
171 >sl1>phcs_ (ring 0, deciduous)
172 polled_vip_mpx (ring 0)
173 pv_salv_seg (ring 0)
This data segment is created in order to run the physical volume salvager (now rarely used). It contains various databases used by the salvager. It is created (by calling grab_aste.pl1) and destroyed for each volume salvage, in each process running a salvage, rather than being a shared segment.

174 salv_abs_seg_00 (ring 0, abs-seg)
175 salv_abs_seg_01 (ring 0, abs-seg)
176 salv_abs_seg_02 (ring 0, abs-seg)
177 salv_abs_seg_03 (ring 0, abs-seg)
200 salv_abs_seg_04 (ring 0, abs-seg)
These five segments are used to overlay the VTOC of a volume being salvaged, and are set up and referenced by vm_vio.pl1. Covered under File System Salvagers.

201 salv_dir_space (ring 0)
202 salv_data (ring 0)
203 salv_temp_dir (ring 0)
These three segments are used by the directory salvager when invoked as the online salvager, in response to a crawlout or bad_dir_ condition. Only one instance of the online salvager may be running at a time, and this is ensured by a lock on salv_data. The online salvager does not interfere with demand directory salvages, however. Covered under File System Salvagers.

204 scavenger_data (ring 0)
The segment containing the tables used when running the online volume scavenger. As many volumes as tables can be fit nere may be scavenged at one time.

The in-use portion of the segment is wired while a scavenge is being done. Covered under File System Salvagers.

205 >sl1>shcs_ (ring 0, deciduous)
206 str_seg (ring 0)
The segment trailer segment. The trailers in this segment record which processes have an SDW for any particular non-supervisor segment, and the backup_abs_seg (see above). Covered under Segment Control.

207 syserr_daemon_dseg (ring 0)
210 syserr_daemon_pds (ring 0)
211 syserr_daemon_stack (ring 0)
The DSEG, PDS, and ring zero stack of the syserr logging daemon. They are filled in when the daemon processs (which runs entirely in ring zero) is created. They are in the global system address space primarily for debugging and recordkeeping purposes.

212 >sl1>system_privilege_ (ring 0, deciduous)
213 >sl1>tandd_ (ring 0, deciduous)
214 template_pds (ring 0)
A template for the PDS, used when creating a process. It is a copy of the pds template which came off the system tape, but which was used to create the Initializer's PDS.

215 tty_area (ring 0)
An unwired database used by ring zero communications system, used primarily for saved metering information on each channel.

216 tty_buf (ring 0)
The important segment in the ring zero communications system. Contains the logical channel table, all communications I/O buffers, and all multiplexer databases.

217 tty_tables (ring 0)
An unwired database used by ring zero communications system to keep the tables used for input and output translation/conversion.

220 vtoc_buffer_seg (ring 0)
The segment containing all buffers for VTOC I/O, and a small amount of control information. Covered under Segment Control.

221 <<unused>>
222 <<unused>>
223 <<unused>>
224 <<unused>>
225 <<unused>>
226 <<unused>>
227 <<unused>>

These segments are unused, since the first segment after the supervisor address space is the ring zero stack, and its segment number must be zero mod 8.

230 >sl1>stack_0.016 (ring 0, deciduous)

The ring zero stack. One is allocated from a pool (>sl1>stack_0.NNN) whenever a process becomes eligible. When a process is not eligible (blocked, usually), it has no ring zero context, and needs no ring zero stack.

231 [pd]>stack_1
232 <<not used>>
232 <<not used>>
234 [pd]>stack_4

The rest of the stacks. There would be others if other rings were being used. A stack is created for each ring as it is needed; the segment number is automatically generated by the CALL6 instruction (from DBR.stack), and when a segment fault occurs on a stack not yet extant (pds$stacks(ring) is null), seg_fault.pl1 calls makestack.pl1 to create one.

TYPICAL ADDRESS SPACE

- NON-SUPERVISOR SEGMENTS

  - REMAINDER OF A PROCESS ADDRESS SPACE.

  - BUILT BY THE NORMAL NAME AND ADDRESS SPACE MANAGEMENT MECHANISMS AS THE PROCESS GETS GOING AND RUNS

  - DIFFERENT IN DIFFERENT PROCESSES; THIS IS ONLY AN EXAMPLE

First Seg made known dynamicly

240 > (the ROOT)

The ROOT directory. The recursive nature of the initiate/segment fault mechanism ensures that this will be the first non-supervisor segment in the address space.

241	>pdd	(directory)
242	>pdd	
243	>sss>bound_process_init_	
244	>udd	(directory)
245	>udd>MED	(directory)
246	>udd>MED>Sibert	(directory)
247	[pd]>!BBBMjzKmkcngb.area.linker	
250	>sss	(directory)
251	>unb	(directory)
252	>sl1	(directory)
253	>tools	(directory)
254	>am	(directory)
255	>sl1>bound_sss_active_	
256	>sl1>operator_pointers_	
257	>sl1>bound_sss_wired_	
260	>sl1>bound_process_env_	
261	>sl1>hcs_	
262	[pd]>pit	
263	>sl1>bound_error_handlers_	
264	>sl1>bound_ipc_	
265	>sss>bound_as_requests_	
266	>sss>bound_info_rtns_	
267	>sc1	(directory)
270	>sc1>whotab	
271	>sclinstallation_parms	
272	>sl1>sys_info	
273	>sss>bound_command_loop_	

TYPICAL ADDRESS SPACE

```
274  >scl>command_usage_counts
275  >scl>command_usage_counts>command_usage_list_
276  >scl>command_usage_counts>command_usage_totals_
277  >sss>bound_exec_com_
300  >sll>error_table_
```

NAME/ADDRESS SPACE METERS

system link meters

- SYSTEM_LINK_METERS - RECORDS CPU TIME AND PAGING INFORMATION USED BY THE DYNAMIC LINKER IN ALL PROCESSES

Linkage Meters:
CPU Metering time 4:58:57

Total time in linker	0:50:53
Average time per link	6.01 msec.
Percentage of real time in linker	17.03
Percentage of CPU time in linker	4.66 %

Time slot (msec)	<25	25-50	50-75	>75
Calls	498469	8357	414	1100
Total time in slot	0:42:03	0:04:16	0:00:24	0:04:09
Percent total time	82.63	8.40	0.81	8.16
Percent total calls	98.06	1.64	0.08	0.22
Average time	5.06	30.68	59.63	226.59
Average page faults	0.18	2.29	6.34	5.65
Segment Search				
Average time	2.58	25.13	53.15	8.70
Average page faults	0.04	0.95	3.53	0.22
Percent time in slot	57.06	82.58	86.46	3.80
Get Linkage				
Average time	0.84	4.08	7.16	219.16
Average page faults	0.06	0.62	0.93	5.33
Percent time in slot	18.54	13.41	11.64	95.71
Definition Search				
Average time	0.24	0.24	0.23	0.20
Average page faults	0.02	0.11	0.24	0.00
Percent time in slot	5.26	0.80	0.37	0.09

NAME/ADDRESS SPACE METERS

link meters

- LINK_METERS - RECORDS CPU TIME AND PAGING INFORMATION USED BY THE DYNAMIC LINKER IN THE PROCESS RUNNING IT

Linkage Meters: To snap link

slot	calls	avg time	avg pf	tot time	% time
<25	1245	4.689	0.3	5.838	76.1
25-50	34	30.201	2.9	1.027	13.4
50-75	6	62.226	6.2	0.373	4.9
>75	2	216.545	6.0	0.433	5.6
Total	1287	5.961	0.4	7.671	

display kst entry

- DISPLAY_KST_ENTRY - DISPLAYS INFORMATION FROM A KST   PLcS_

```
! display_kst_entry >udd>Multics>Sibert

                          kst
segno:     246  at  162|270
usage:     7, 0, 0, 0, 0, 0, 0, 0
entryp:    245|20750        >udd> mult      sibert
uid:       102401170050
dtbm:      446556324757
mode:      7 (0, 0, 0) — rew
ex mode:   70000000000 (7, 7, 7) — sma
infcount:  3
hdr:       4
flags:     dirsw write tms
```

rzd - ring zero dump

rzd 162 270 10

Segno   OFFSET   10 OCTAL words long

TOPIC V

Directory Control

Page

DIRECTORY CONTROL OVERVIEW

- FUNCTION

  - DIRECTORY CONTROL IS A SET OF HARDCORE MODULES RESPONSIBLE FOR THE MAINTENANCE OF THE MULTICS DIRECTORY STRUCTURE -- IE: THE HIERARCHY

  - ITS TASKS INCLUDE CREATING, MANIPULATING AND INTERPRETING THE CONTENTS OF DIRECTORY SEGMENTS, TO INCLUDE:

    - ACCESS CONTROL LISTS (ACL'S), NAMES, AND VTOCE POINTERS OF ENTRIES DESCRIBED THEREIN

  - ONLY DIRECTORY CONTROL IS ALLOWED TO ALTER THE CONTENTS OF DIRECTORY SEGMENTS

  - DIRECTORY CONTROL IMPLICITLY RELIES UPON THE SERVICES OF OTHER SUBSYSTEMS SUCH AS SEGMENT CONTROL AND PAGE CONTROL, AND ALSO INVOKES THEM DIRECTLY BY SUBROUTINE CALL

    - DIRECTORIES ARE SIMPLY SEGMENTS TO THESE SUBSYSTEMS

  - DIRECTORY CONTROL IS INVOKED ONLY BY SUBROUTINE CALLS

- PRINCIPAL USER INTERFACES

  - COMMAND LEVEL

- create, create_dir, link, set_acl, delete_acl, status, list, add_name, rename

- SUBROUTINE LEVEL

  - hcs_$append_branch, hcs_$add_acl_entries, hcs_$append_link, hcs_$delete_acl_entries, hcs_$status_, hcs_$chname_file

- MAJOR DATA BASES

  - DIRECTORY SEGMENTS

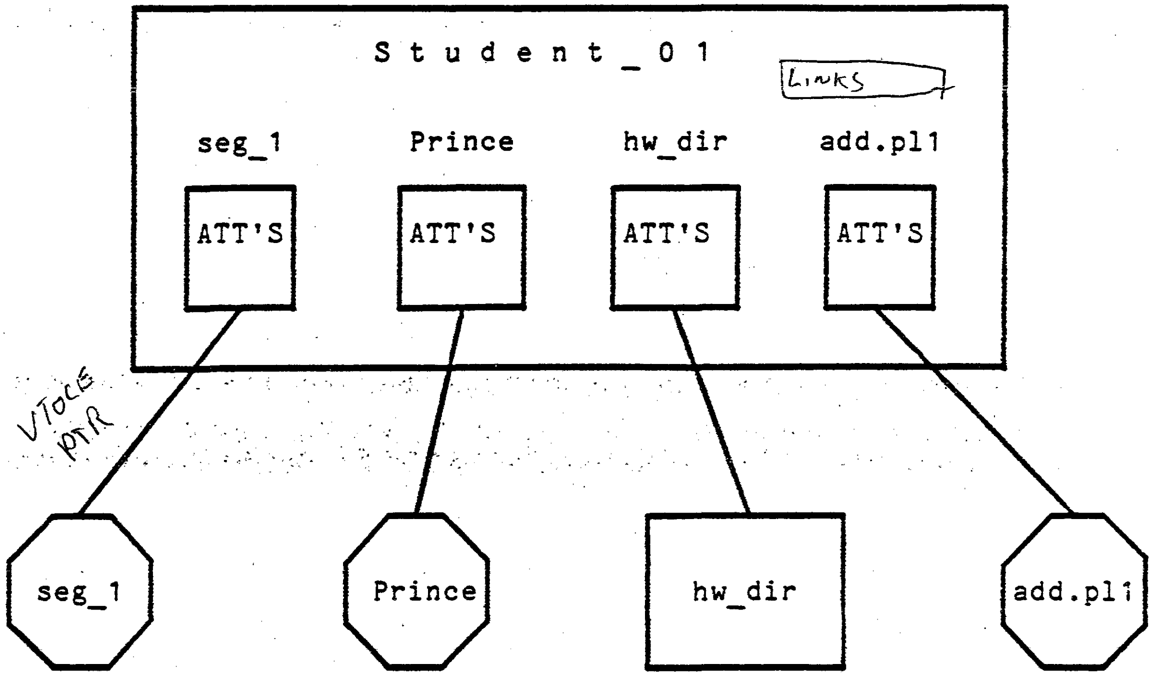
    - CONTAIN THE ATTRIBUTES AND OTHER INFORMATION ABOUT THEIR SEGMENTS (NEEDED TO FIND SEGMENTS, RETURN STATUS INFORMATION, AND BUILD VTOCE'S AT SEGMENT CREATION)

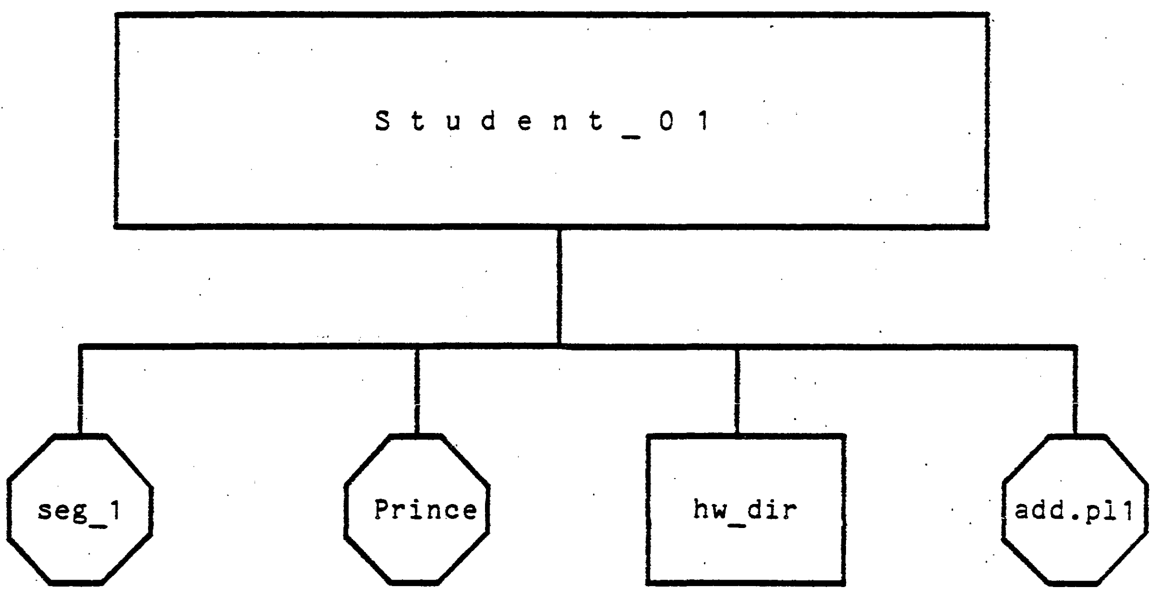
  - THE DIRLOCKT_SEG

    - SEGMENT WHERE DIRECTORY LOCKING IS MANAGED

DIRECTORY CONTROL TERMINOLOGY

Dir Seg Limit 205





DIRECTORY CONTROL TERMINOLOGY

UNIQUE ID (UID)	A 36-BIT ID (SERIAL NUMBER) ASSIGNED TO EVERY SEGMENT WHEN CREATED
SON:	OF A DIRECTORY. AN IMMEDIATELY INFERIOR (SUBORDINATE) SEGMENT
SON'S LVID:	OF A DIRECTORY. THE ID OF THE LOGICAL VOLUME ON WHICH THE DIRECTORY'S SONS RESIDE (AND WILL RESIDE)
GRANDSON:	OF A DIRECTORY. A SEGMENT INFERIOR BY MORE THAN ONE HIERARCHICAL LEVEL
PARENT:	OF A SEGMENT. THE "CONTAINING" DIRECTORY SEGMENT
ANCESTOR:	OF A SEGMENT. THE PARENT, OR GRANDPARENT, OR GREAT GRANDPARENT, ETC.
BROTHER:	OF A SEGMENT. ANOTHER SEGMENT HAVING THE SAME PARENT

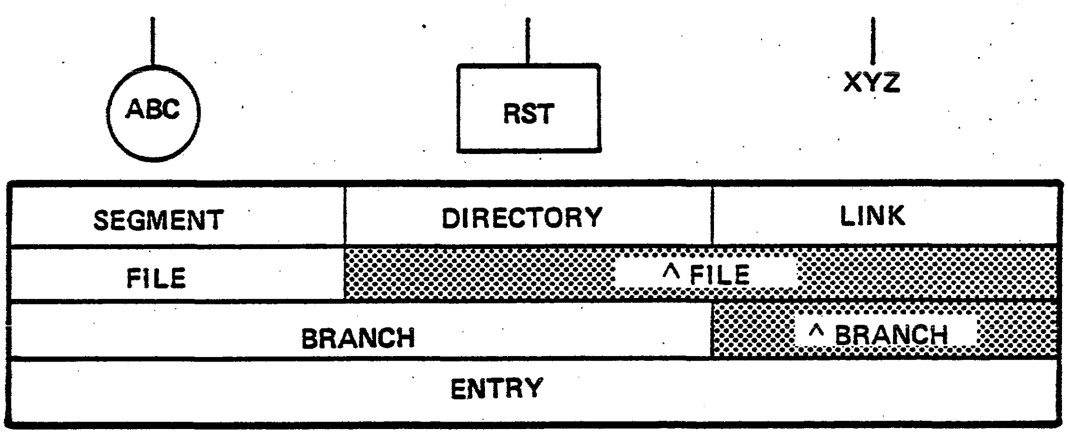
DIRECTORY CONTROL TERMINOLOGY

BRANCH (1): IN A DIRECTORY. A DATA STRUCTURE, CONTAINED IN A DIRECTORY SEGMENT, THAT DESCRIBES AN IMMEDIATELY INFERIOR SEGMENT OR DIRECTORY (BUT NOT A LINK)

BRANCH (2): OF A DIRECTORY. REFERS TO THE ACTUAL SEGMENT OR DIRECTORY IMMEDIATELY INFERIOR TO THE DIRECTORY

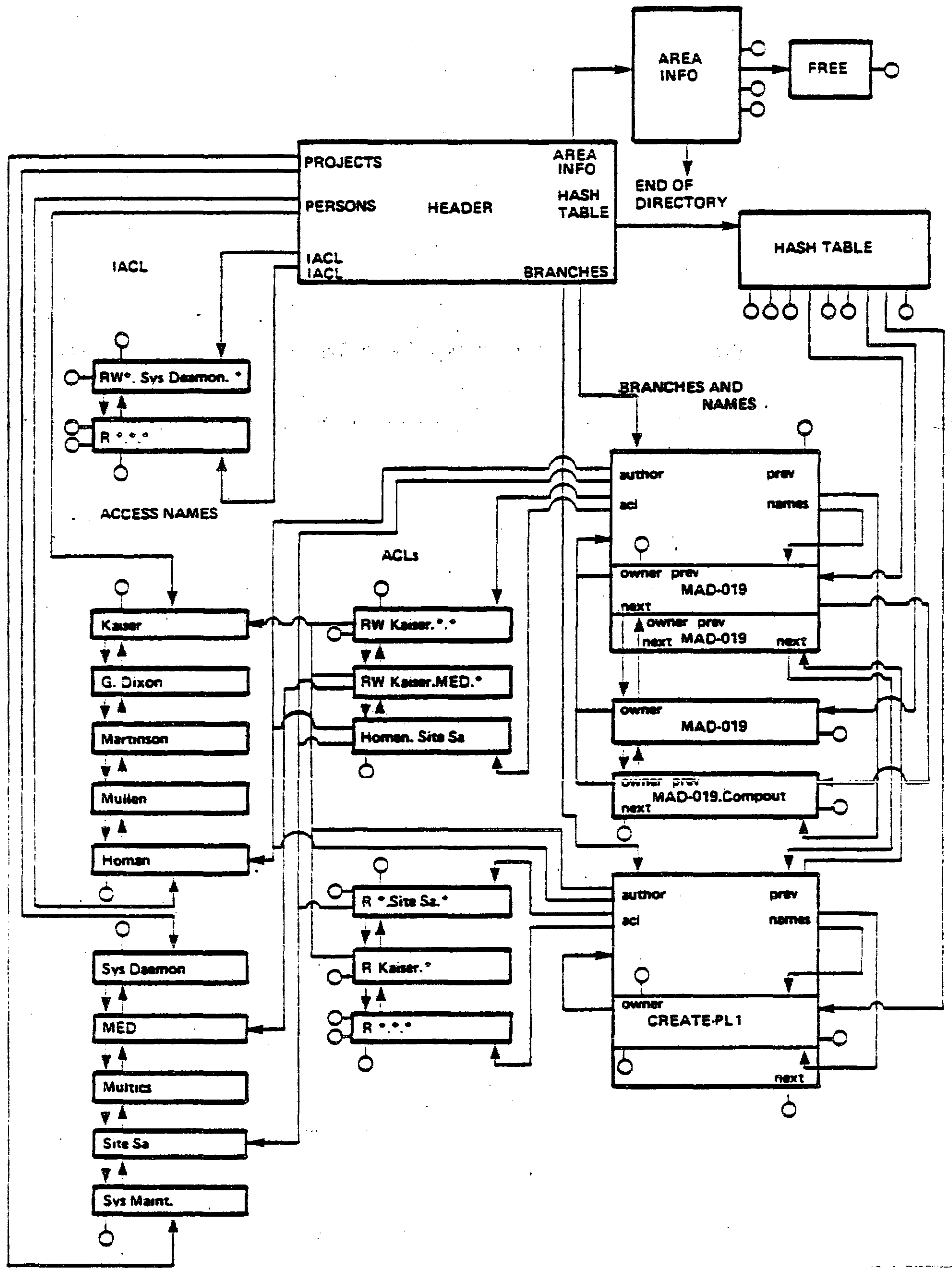
ENTRY (1): IN A DIRECTORY. SAME AS BRANCH (1) BUT INCLUDES LINKS

ENTRY (2): OF A DIRECTORY. SAME AS BRANCH (2) BUT INCLUDES LINKS



STORAGE SYSTEM TERMINOLOGY

DIRECTORY SEGMENTS



DIRECTORY SEGMENT STRUCTURE

<u>DIRECTORY CONTROL DATA BASES</u>

<u>DIRECTORY SEGMENTS</u>

- DIRECTORY SEGMENTS ARE DISK RESIDENT (RLV) DATA BASES MAINTAINED BY DIRECTORY CONTROL

  - ONE DIRECTORY SEGMENT PER DIRECTORY IN THE HIERARCHY

- DIRECTORY SEGMENTS CONTAIN A CATALOG OF STORAGE SYSTEM INFORMATION ABOUT OTHER SEGMENTS, DIRECTORIES AND LINKS

- ALL DIRECTORY SEGMENTS, BY CONVENTION, RESIDE ON THE ROOT LOGICAL VOLUME (RLV)

- DIRECTORY SEGMENTS ARE CREATED BY append MUCH LIKE NORMAL SEGMENTS ARE CREATED

- DIRECTORY SEGMENTS CONTAIN MANY INTER-RELATED COMPLEX DATA STRUCTURES TO INCLUDE THE FOLLOWING (NOT NECESSARILY CONTIGUOUS) REGIONS:

  - DIRECTORY HEADER

    - CONTAINS SELF DESCRIPTIVE INFORMATION LIKE UID, AIM CLASSIFICATION, ETC; AND POINTERS TO OTHER REGIONS

DIRECTORY CONTROL DATA BASES

DIRECTORY SEGMENTS

- HASH TABLE

  - USED TO QUICKLY LOCATE AN ENTRY, GIVEN ITS NAME

- ENTRY LIST

  - CONTAINS ONE DESCRIPTIVE DATA STRUCTURE (AN ENTRY) FOR EACH SEGMENT, DIRECTORY, OR LINK IMMEDIATELY INFERIOR TO THE DIRECTORY (I.E. ALL ENTRIES)

- PERSON_ID AND PROJECT_ID NAME LISTS

  - CONTAINS ALL PERSON_ID'S/PROJECT_ID'S REQUIRED TO DESCRIBE THE ACL, THE AUTHOR, ETC. OF ALL SEGMENTS AND DIRECTORIES IMMEDIATELY INFERIOR TO THE DIRECTORY (I.E. ALL BRANCHES)

- NAME LIST (ONE PER ENTRY)

  - CONTAINS ALL NAMES CURRENTLY ASSIGNED TO THE ENTRY

    - PRIMARY NAME IS ACTUALLY CONTAINED IN ENTRY STRUCTURE ITSELF

- ACCESS CONTROL LIST (ONE PER ENTRY)

  - CONTAINS ALL ACL ENTRIES CURRENTLY ASSOCIATED WITH THE ENTRY

dir w - directory write Behind.
needed for Data mgmt,
otherwise not needed.

Table of who's reading + writing

dirlockt_seg

UID	PID	R
UID	PID	R

n-reader 1-writer locking system

DIRECTORY CONTROL DATA BASES

DIRECTORY HEADER

Word	Bits 0–17	Bits 18–35
0	PROCESS ID OF ~~LAST MODIFIER~~ current modifier	
1	TYPE OF OBJECT (DIR HEADER = 3)	SIZE OF HEADER (64 WORDS)
	6 WORDS USED BY SALVAGER FOR RECORDING DATE TIME CHECKED AND ERRORS DISCOVERED	
8	DIRECTORY'S UNIQUE IDENTIFIER (UID) *	
9	DIRECTORY'S PHYSICAL VOLUME ID (PVID) *	
10	SONS' LOGICAL VOLUME ID (LVID) *	
11–12	ACCESS ISOLATION MECHANISM (AIM) CLASS *	
13	DIRECTORY'S VTOCX *	HEADER VERSION NUMBER
14	ENTRY LIST START R_PTR	
15	ENTRY LIST END R_PTR	
16	PERSON_ID LIST START R_PTR	PROJECT_ID LIST START R_PTR
17	PERSON_ID LIST END R_PTR	PROJECT_ID LIST END R_PTR
18	NUMBER OF SEG ENTRIES	NUMBER OF DIR ENTRIES
19	NUMBER OF LINK ENTRIES	NUMBER OF ACL ENTRIES
20	ALLOCATION AREA R_PTR	PERP, MDIR, FRPV, RENA
	24 WORDS USED FOR INITIAL ACL IMPLEMENTATION	
45	HASH TABLE SIZE	HASH TABLE R_PTR
46	NUMBER OF USED PLACES IN HASH TABLE	
47	DEPTH OF THIS DIRECTORY	
48	DATE TIME SALVAGED (DTS)	
49	UID OF SUPERIOR MASTER DIR	
50	MODIFICATION PSEUDO-CLOCK	
	12 WORDS UNUSED	
62	CHECKSUM (NOT USED)	
63	UID OF PARENT DIR	

(WORDS) (dir_header.incl.pl1)

DIRECTORY HEADER

A DISK RESIDENT (RLV – DIRECTORY SEGMENT) DATA BASE – ONE PER DIRECTORY

(NOT WIRED, NOT ENCACHEABLE)

*COPIED FROM BRANCH WHEN CREATED

DIRECTORY CONTROL DATA BASES

DIRECTORY HEADER

- THE DIRECTORY HEADER IS A DISK RESIDENT DATA BASE CONTAINED AT THE BEGINNING OF A DIRECTORY SEGMENT

  - ONE DIRECTORY HEADER PER DIRECTORY SEGMENT

- THE DIRECTORY HEADER CONTAINS SELF DESCRIPTIVE INFORMATION SUCH AS:

  - THE PROCESS ID OF THE LAST PROCESS TO MODIFY THE DIRECTORY SEGMENT'S CONTENTS

  - THE DIRECTORY'S UID, LVID, PVID, VTOC INDEX, AND AIM CLASSIFICATION

  - RELATIVE POINTERS TO THE BEGINNING AND END OF THE ENTRY LIST, PERSON_ID LIST, PROJECT_ID LIST, AND THE HASH TABLE

  - HIERARCHY DEPTH OF THE DIRECTORY SEGMENT

  - UID OF THE MASTER DIRECTORY AND THE PARENT DIRECTORY

  - SON'S LVID - THE ID OF THE LV ON WHICH INFERIOR NON-DIRECTORY SEGMENTS RESIDE (AND WILL RESIDE)

- THE DIRECTORY HEADER IS ACCESSED AT THE BEGINNING OF DIRECTORY QUERY AND UPDATE OPERATIONS

Attributes vs Data

DTEM vs DTCM

DIRECTORY CONTROL DATA BASES

DIRECTORY ENTRIES

(WORDS)	Bits 0–17	Bits 18–35
0	FORWARD R_PTR	BACKWARD R_PTR
1	TYPE OF ENTRY (SEG = 7)	SIZE OF ENTRY (37 WORDS)
2	ENTRY'S UNIQUE IDENTIFIER (UID)	
3	DATE TIME ENTRY MODIFIED (DTEM)	
4	BRAN (bit 0)	NUMBER OF NAMES
5	NAME LIST START R_PTR	NAME LIST END R_PTR
6	AUTHOR'S PERSON_ID R_PTR	AUTHOR'S PROJECT_ID R_PTR
7	AUTHOR'S TAG	
8–21	14 WORDS CONTAINING PRIMARY NAME	
22	DATE TIME DUMPED (DTD)	
23		
24	BRANCH'S PHYSICAL VOLUME ID (PVID)	
25	BRANCH'S VTOC INDEX	
26	DIR, PERP, COPY, SAFE, MULT, AUD, SOOS, ENTR, MDIR	ENTRY POINT BOUND
27–28	ACCESS ISOLATION MECHANISM (AIM) CLASS	
29	R(1), R(2), R(3), XR(1), XR(2), XR(3)	ACL ENTRY COUNT
30	ACL START R_PTR	ACL END R_PTR
31	BC AUTHOR'S PERSON_ID R_PTR	BC AUTHOR'S PROJECT_ID R_PTR
32	BC AUTHOR'S TAG	BIT COUNT (IF DIR, ≠ 0 IMPLIES MSF)
33	SONS' (NON-DIR'S) LOGICAL VOLUME ID (LVID)	
34		
35	CHECKSUM FROM DTD	
36	UID OF PARENT DIRECTORY	

(dir_entry.incl.pl1)

BRANCH ENTRY

A DISK RESIDENT (RLV – DIRECTORY SEGMENT) DATA BASE – ONE PER BRANCH IN DIRECTORY

(NOT WIRED, NOT ENCACHEABLE)

any info not here is in VTOCE

DIRECTORY ENTRIES

(WORDS)	Bits 0–17	Bits 18–35
0	FORWARD R_PTR	BACKWARD R_PTR
1	TYPE OF ENTRY (LINK = 5)	SIZE OF ENTRY (69 WORDS)
2	LINK'S UNIQUE IDENTIFIER (UID) (NEVER VISIBLE TO USERS)	
3	DATE TIME ENTRY MODIFIED (DTEM)	
4	BRAN	NUMBER OF NAMES
5	NAME LIST START R_PTR	NAME LIST END R_PTR
6	AUTHOR'S PERSON_ID R_PTR	AUTHOR'S PROJECT_ID R_PTR
7	AUTHOR'S TAG	
	14 WORDS CONTAINING PRIMARY NAME	
22	DATE TIME DUMPED (DTD)	
23		
24		PATHNAME SIZE
	42 WORDS CONTAINING THE ABSOLUTE PATHNAME OF THE LINK'S TARGET	
67	UID CHECKSUM	
68	UID OF PARENT DIRECTORY	

Words 0–23: IDENTICAL TO A BRANCH ENTRY

(dir_link.incl.pl1)

LINK ENTRY

A DISK RESIDENT (RLV-DIRECTORY SEGMENT) DATA BASE – ONE PER LINK IN DIRECTORY
(NOT WIRED, NOT ENCACHEABLE)

DIRECTORY ENTRIES

- THE DIRECTORY ENTRY IS A DISK RESIDENT (RLV) DATA BASE CONTAINED WITHIN A DIRECTORY SEGMENT

  - ONE DIRECTORY ENTRY (IN THE DIRECTORY SEGMENT) FOR EACH IMMEDIATELY INFERIOR ENTRY IN THE HIERARCHY

- EACH DIRECTORY ENTRY IS A DATA STRUCTURE DESCRIBING THE ATTRIBUTES OF A SEGMENT, DIRECTORY OR LINK

- DIRECTORY ENTRIES COME IN TWO FLAVORS:

  - LINK ENTRY, (38 OR 72 WORDS) CONTAINING:

    - DATE TIME MODIFIED AND DUMPED (BY THE HIERARCHY DUMPER, NOT VOLUME DUMPER)

    - RELATIVE POINTERS TO THE ENTRY'S NAME LIST AND AUTHOR'S USER ID

    - ABSOLUTE PATHNAME OF THE LINK'S TARGET

    - UID OF PARENT DIRECTORY

DIRECTORY CONTROL DATA BASES

DIRECTORY ENTRIES

- BRANCH ENTRY, (38 WORDS) CONTAINING:

  - DATE TIME MODIFIED AND DUMPED (BY THE HIERARCHY DUMPER, NOT VOLUME DUMPER)

  - RELATIVE POINTERS TO THE ENTRY'S NAME LIST AND AUTHOR'S USER ID

  - BRANCH'S UID, PVID, AND VTOC INDEX

  - AIM CLASSIFICATION, ENTRY POINT BOUND, RING BRACKETS, AND RELATIVE POINTERS TO THE ACL

  - BRANCH'S BIT COUNT AND BIT COUNT AUTHOR

  - SON'S LVID (IF A DIRECTORY) AND PARENT'S UID

  - FLAGS DESCRIBING VARIOUS STATES AND PROPERTIES OF THE ENTRY SUCH AS: DIRECTORY, MASTER DIRECTORY, SECURITY OUT OF SERVICE, COPY AND SAFETY SWITCH, ETC

display branch

- DISPLAY_BRANCH - DISPLAYS BRANCHES IN THE DIRECTORY HIERARCHY

  I OFTEN USEFUL ON CONJUNCTION WITH DUMP_VTOCE Old command
  Display_vtoce

```
! display_branch >udd>Multics>Sibert

        Branch for Sibert in >udd>Multics at 245|20742

   UID 102401170050, is vtocx 63 on root4 (of log vol. root)
   Sibert is a directory.
   Ring brackets (0 0 0)
   Entry modified 02/23/83  1912.1 est Wed
   Dumped 03/20/83  0955.6 est Sun
   9 names.
```

- DISPLAY IS NOT COMPLETE, SO A RING_ZERO_DUMP OF THE SAME DATA IS INCLUDED

words long for an entry

```
! ring_zero_dump >udd>Multics 20742 46 -ch
```

20744

```
020742 021310020604 000004000046 102401170050 446556324757 .......&B.x(....
020746 400000000011 020752021152 001720000532 172000000000 .......j....z...
020752 021010000000 000006000016 020742000233 000000000000 ................
020756 123151142145 162164040040 040040040040 040040040040 Sibert

020762 040040040040 040040040040 040040040040 040040040040
020766 000000000000 102401170050 446752147026 000000000000 ....B.x(..g.....
020772 135240026001 000063000000 400000000000 000000000000 ]....3..........
020776 000000000000 000770000012 021170021300 001720000532 .........x......

021002 172000000000 225072707470 000000000000 000000000000 z....:..........
021006 033023254650 000000000000                           ........
```

To find Branch of segment to delete, if you can't delete in any other way. Try hpdl vtoce or hp_delete

Prz 245 20744 0

UID

TOPIC VI

Volume Management

Page

VOLUME MANAGEMENT OVERVIEW

- FUNCTION

  - VOLUME MANAGEMENT IS RESPONSIBLE FOR THE MANAGEMENT OF PHYSICAL AND LOGICAL VOLUMES

  - ITS TASKS INCLUDE:

    - ACCEPTANCE AND DEMOUNTING OF PHYSICAL VOLUMES

    - MAINTAINING THE ASSOCIATION BETWEEN PHYSICAL VOLUMES, LOGICAL VOLUMES, AND DISK DRIVES

    - ENSURING THE INTEGRITY OF VOLUME CONTENTS

    - MAKING VOLUME CONTENTS ACCESSABLE TO PAGE CONTROL (PAGES) AND SEGMENT CONTROL (VTOC ENTRIES)

  - VOLUME MANAGEMENT IS INVOKED ONLY BY SUBROUTINE CALLS

VOLUME MANAGEMENT OVERVIEW

- MAJOR DATA BASES

  - PHYSICAL VOLUME TABLE (PVT) - ONE PER SYSTEM

    - PHYSICAL VOLUME TABLE ENTRY (PVTE) - ONE PER DISK DRIVE KNOWN TO THE SYSTEM

    - EACH PVTE IDENTIFIES A DRIVE'S DEVICE NUMBER, SUBSYSTEM NAME, DEVICE TYPE, AND INFORMATION ABOUT THE PHYSICAL VOLUME CURRENTLY MOUNTED

    - USED TO MAP REFERENCES TO PAGES OF SEGMENTS INTO AN I/O REQUEST TO THE CORRECT DISK DRIVE

  - LOGICAL VOLUME TABLE (LVT) - ONE PER SYSTEM

    - LOGICAL VOLUME TABLE ENTRY (LVTE) - ONE PER MOUNTED LOGICAL VOLUME

    - EACH LVTE CONTAINS THE LOGICAL VOLUME ID, POINTERS TO MEMBER PVTE'S, AIM CLASS LIMITS, ETC.

    - USED TO DETERMINE A USER'S ACCESS TO A LOGICAL VOLUME (PRIVATE OR PUBLIC) AND TO LOCATE MEMBER PHYSICAL VOLUMES

  - VOLUME HEADER - ONE PER PACK

    - VOLUME LABEL (REGISTRATION AND ACCEPTANCE INFORMATION)

    - VOLUME MAP (OCCUPIED/VACANT INFORMATION FOR VOLUME CONTENTS)

VOLUME MANAGEMENT OVERVIEW

- RECORD STOCKS - ONE PER MOUNTED VOLUME

  - ONLINE CACHE OF INFORMATION ABOUT USED / UNUSED RECORDS ON THE VOLUME

  - THIS INFORMATION IS DERIVED FROM THE VOLUME MAP, BUT KEPT ONLINE TO AVOID THE NECESSITY OF REFERRING TO THE VOLUME MAP ON DISK EVERY TIME A RECORD IS ALLOCATED OR FREED

  - WHEN THE CACHE BECOMES COMPLETELY EMPTY OR COMPLETELY FULL, IT MUST BE UPDATED FROM/TO DISK - A PROTOCOL ENSURES THAT THE COPY ON DISK IS ALWAYS CONSISTENT

  - PROVIDED BY VOLUME MANAGEMENT, BUT USED BY PAGE CONTROL

- VTOCE STOCKS - ONE PER VOLUME

  - SIMILAR TO RECORD STOCKS, BUT MAINTAINS INFORMATION ABOUT USED / UNUSED VTOC ENTRIES ON THE VOLUME

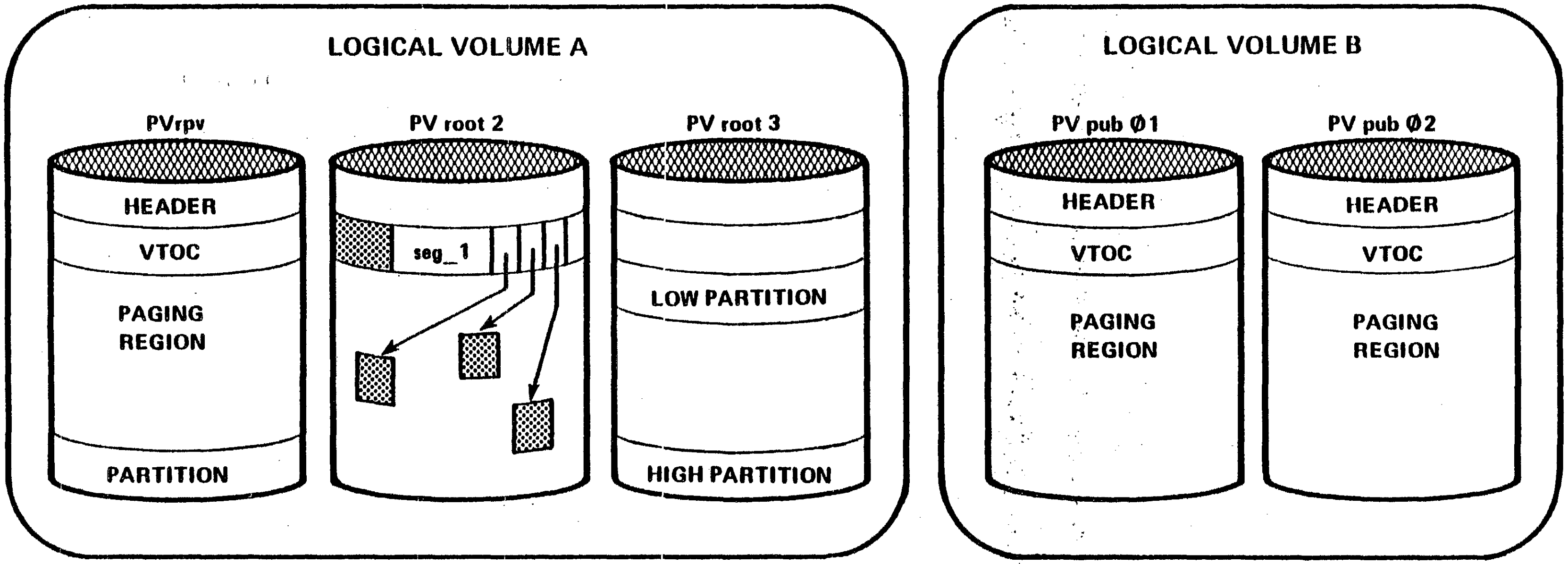
  - PROVIDED BY VOLUME MANAGEMENT, BUT USED BY SEGMENT CONTROL

- PHYSICAL VOLUME HOLD TABLE (PVHT) - ONE PER SYSTEM

  - RECORDS THE COMMENCEMENT OF COMPOUND I/O OPERATIONS UPON A PHYSICAL VOLUME

  - THIS INFORMATION PREVENTS A VOLUME FROM BEING DEMOUNTED WHILE SUCH AN OPERATION IS IN PROGRESS

THE NEW STORAGE SYSTEM



PHYSICAL VOLUME (PV):	**A DISK PACK**
LOGICAL VOLUME (LV):	**A COLLECTION OF ONE OR MORE PHYSICAL VOLUMES GIVEN A SINGLE LOGICAL IDENTITY**
VOLUME HEADER:	**CONTAINS THE LABEL, VOLUME MAP, VTOC MAP, AND VOLUME DUMP MAPS.**
VOLUME TABLE OF CONTENTS (VTOC):	**AN ARRAY OF ENTRIES (VTOCE's) DESCRIBING EACH SEGMENT ON THE PHYSICAL VOLUME**
PAGING REGION:	**THE REGION IN WHICH ALL RECORDS DESCRIBED IN THE VTOC RESIDE (I.E., PAGES OF STORAGE SYSTEM SEGMENTS)**
PARTITION:	**A REGION SET ASIDE FOR SOME PURPOSE OTHER THAN PAGES OF STORAGE SYSTEM SEGMENTS**

THE NEW STORAGE SYSTEM

- SINCE RELEASE 4.0, THE MULTICS STORAGE SYSTEM HAS BEEN ORGANIZED INTO PHYSICAL AND LOGICAL VOLUMES HAVING THE FOLLOWING PROPERTIES

  1 hardware addressable Unit. ie 500's

  - A PHYSICAL VOLUME (PV) IS A DISK PACK (MOUNTED OR NOT) CONTAINING:

    - A LABEL IDENTIFYING ITSELF - INCLUDING A PHYSICAL VOLUME ID (PVID)

    - A VOLUME MAP DESCRIBING WHICH PAGES and VTOCES ARE IN USE AND WHICH ARE FREE.

    - A VOLUME TABLE OF CONTENTS (VTOC) DESCRIBING WHICH SEGMENTS ARE RESIDENT THEREIN - AND THE EXACT LOCATION OF EACH OF THEIR PAGES

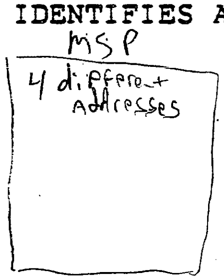
    - THE PAGES OF RESIDENT SEGMENTS (ASSIGNED TO RECORDS OF 1024 WORDS IN SIZE)

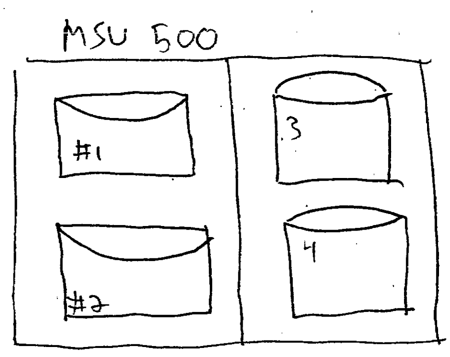
    - AND OPTIONALLY: CONTIGUOUS REGIONS CALLED PARTITIONS, SET ASIDE FOR SPECIAL USE (FDUMP IMAGES, HARDCORE PAGING, ETC)

  - ALL PAGES OF A SEGMENT RESIDE ON A GIVEN PHYSICAL VOLUME

    - THAT IS: EACH NON-ZERO PAGE OF A SEGMENT IS ASSIGNED TO A RECORD OF THE PHYSICAL VOLUME

  - THE PAIR OF PHYSICAL VOLUME ID (PVID) AND VTOC INDEX UNIQUELY IDENTIFIES ANY SEGMENT IN THE STORAGE SYSTEM HIERARCHY





THE NEW STORAGE SYSTEM

- A LOGICAL VOLUME (LV) CONSISTS OF ONE OR MORE PHYSICAL VOLUMES, WHICH ARE:
  - GIVEN ONE LOGICAL VOLUME ID (LVID)
  - ALWAYS MOUNTED AS A SET
- OFFSPRING (SONS, GRANDSONS, ETC) OF A DIRECTORY (NORMALLY) RESIDE WITHIN A GIVEN LOGICAL VOLUME
  - IN OTHER WORDS, A SUB-TREE (NORMALLY) SPANS NO MORE THAN ONE LOGICAL VOLUME
- DIRECTORY SEGMENTS ARE AN EXCEPTION TO THE ABOVE AS ALL DIRECTORY SEGMENTS ARE ASSIGNED TO A LOGICAL VOLUME OF THEIR OWN CALLED THE "ROOT LOGICAL VOLUME" (RLV)
  - THE PHYSICAL VOLUME CONTAINING THE ROOT DIRECTORY IS CALLED "THE ROOT PHYSICAL VOLUME" (RPV)
  - THE RLV IS SPECIAL BECAUSE IT MUST ALWAYS BE MOUNTED, AND IT CONTAINS ALL DIRECTORY SEGMENTS, BUT IT ALSO CONTAINS OTHER SEGMENTS
- SHOULD THE GROWING OF A SEGMENT CAUSE A PHYSICAL VOLUME TO BECOME FULL, A "SEGMENT MOVE" IS AUTOMATICALLY INITIATED
  - THIS IS ONE OF THE MOST COMPLEX AND EXPENSIVE SERVICES PERFORMED BY THE SYSTEM - BUT HAPPENS VERY INFREQUENTLY

THE NEW STORAGE SYSTEM

- SHOULD A LOGICAL VOLUME BECOME FULL:

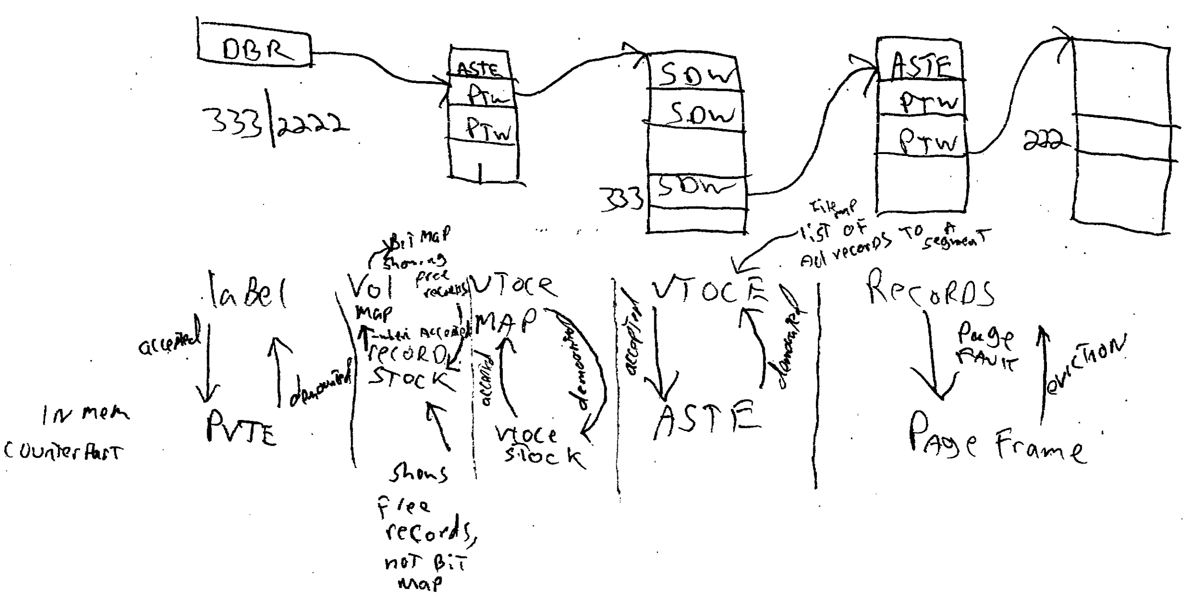
  - USER AND SYSTEM ERROR MESSAGES ARE GENERATED

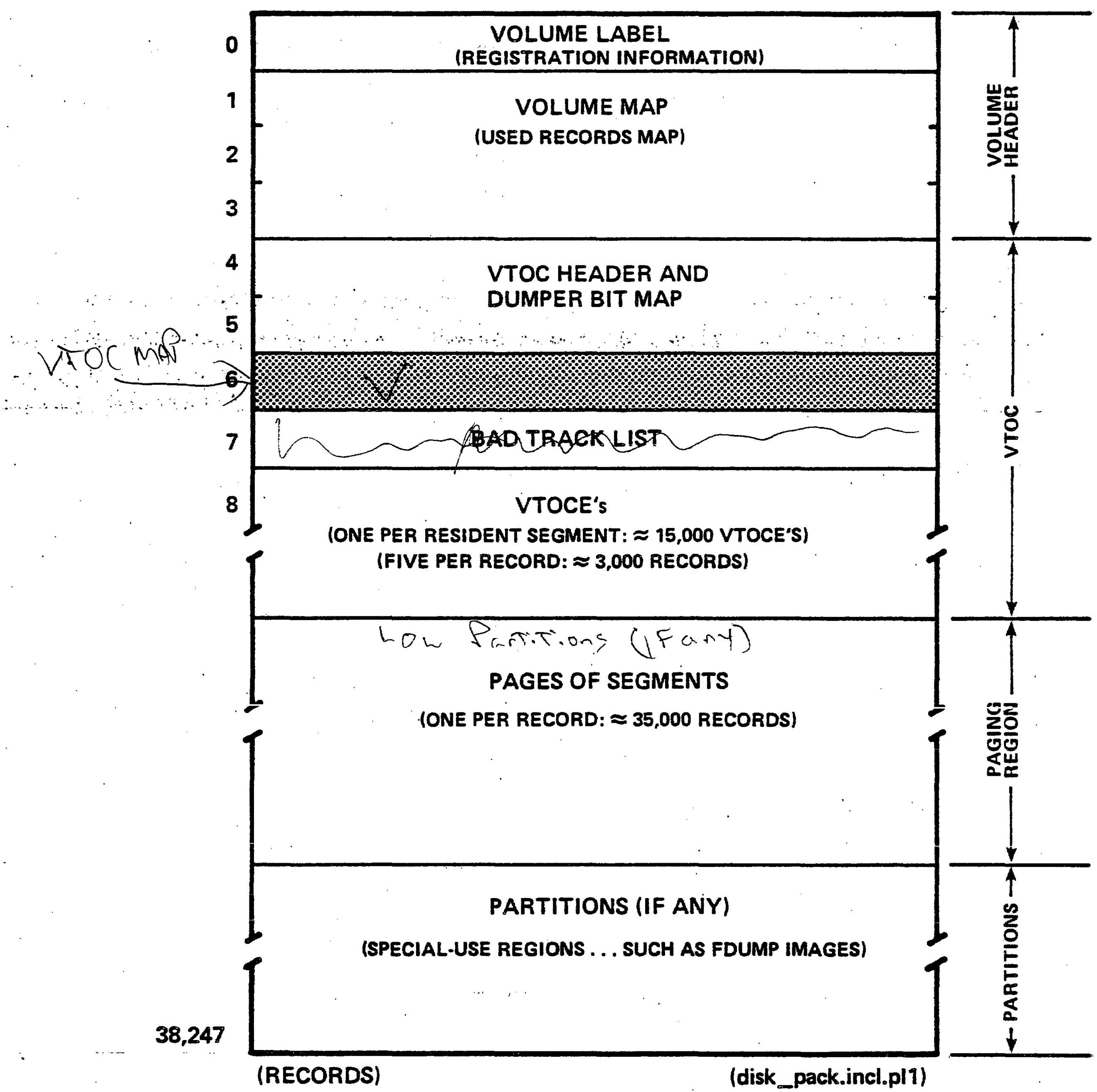
  - SPACE MUST BE OBTAINED ON THE LOGICAL VOLUME BY ADDING MORE PHYSICAL VOLUMES OR BY DELETING OR MOVING SEGMENTS FROM THE LOGICAL VOLUME

  - BECAUSE IT CONTAINS ALL DIRECTORY SEGMENTS, SPACE ON THE RLV IS CRITICAL: IF IT IS USED UP, THE SYSTEM MAY NOT BE ABLE TO CONTINUE OPERATION.

- THE CHOICE OF WHICH PHYSICAL VOLUME TO USE WHEN CREATING A SEGMENT IS MADE IN SUCH A WAY AS TO TRY TO BALANCE THE ALLOCATED SPACE ON ALL THE PHYSICAL VOLUMES OF A LOGICAL VOLUME

KST	RNT (1)	RNT (2)	Dir	AST
UID	ref name	Seg #	name	UID
KSTE (Seg #)	Seg #	ref name	name struct → entry	ASTE

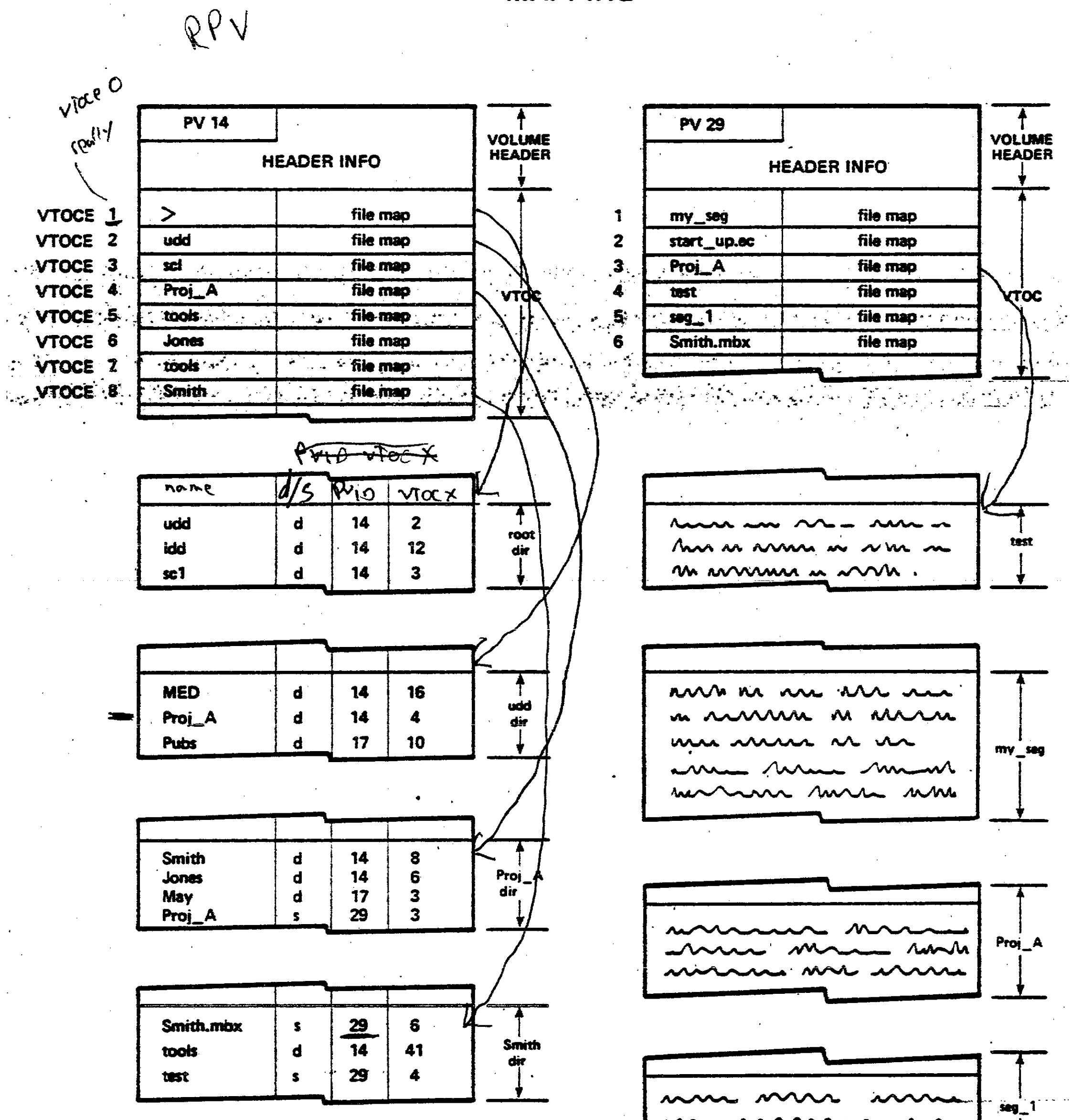




PHYSICAL VOLUME FORMAT

(MSU 451)

HIERARCHY TO STORAGE SYSTEM MAPPING



PHYSICAL VOLUME ID 14
CONTENTS (SIMPLIFIED)

>udd>Proj.A>Smith>Test

PHYSICAL VOLUME ID 29
CONTENTS (SIMPLIFIED)

THE NEW STORAGE SYSTEM

- IF ONE KNOWS WHERE THE ROOT DIRECTORY IS, ALL SEGMENTS IN THE MULTICS HIERARCHY CAN BE FOUND (ASSUMING THE AVAILABILITY OF ALL REQUIRED PHYSICAL VOLUMES)

- MAJOR DESIGN POINT

  - MANY DISK RESIDENT DATA BASES (TO INCLUDE THE PAGES OF SEGMENTS) ARE COPIED INTO MAIN MEMORY AND WRITTEN BACK TO DISK AT SUCH TIMES AS:

    - SYSTEM START-UP/SHUT-DOWN

    - PHYSICAL VOLUME MOUNTING/DEMOUNTING

    - SEGMENT ACTIVATION/DEACTIVATION

    - PAGE FAULTS

  - WHILE IN MAIN MEMORY, THE MEMORY RESIDENT COPY IS CONSIDERED TO BE <u>THE</u> COPY

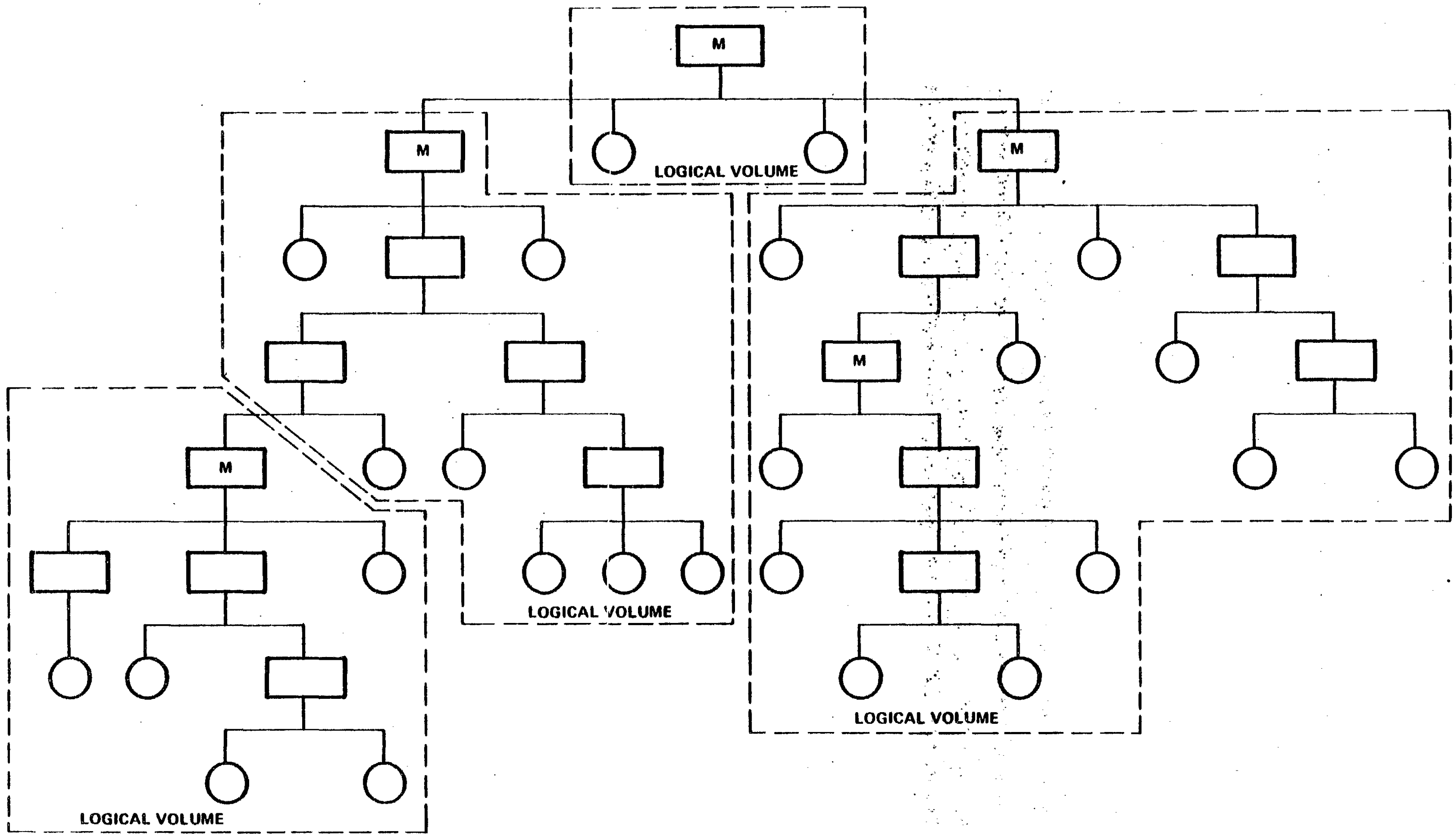
  - WHILE IN MAIN MEMORY, THE DISK RESIDENT COPY IS CONSIDERED TO BE (AND OFTEN IS) WHOLLY INVALID

	SECTOR	VTOCE	RECORD (PAGE)	TRACK	CYLINDER	PACK
WORDS	64	192	1024	3072	58,368	39.2M
SECTORS		3	16	40	752	611.9K
VTOCES			5			
RECORDS (PAGES)				2.5	47	38,247
TRACKS					19	
CYLINDERS						815

MULTICS DISK PACK STATISTICS (MSU 451)

* THE ABOVE FIGURES APPEAR INCONSISTENT IF DATA FORMATTING IS NOT CONSIDERED (E.G.: SOME DISK SPACE IS NOT USED)

fs_dev_types.incl.pl1



LOGICAL VOLUME/MASTER DIRECTORY RELATIONSHIP

* NOTE: ALL DIRECTORY SEGMENTS RESIDE ON ONE DESIGNATED LOGICAL VOLUME CALLED THE ROOT LOGICAL VOLUME (RLV)

VOLUME MANAGEMENT TERMINOLOGY

MOUNT: TO PHYSICALLY PLACE A DISK PACK ON A DRIVE AND CYCLE UP THE DRIVE. (PERFORMED BY THE OPERATOR, NOT BY SOFTWARE)

ACCEPT: AFTER MOUNTING, TO ESTABLISH IN THE SUPERVISOR THE BINDING BETWEEN THE DRIVE AND THE PHYSICAL VOLUME MOUNTED

PUBLIC: A LOGICAL VOLUME ATTRIBUTE INDICATING THAT THE VOLUME IS ATTACHED TO ALL PROCESSES (BY DEFAULT) WHEN ACCEPTED

PRIVATE: A LOGICAL VOLUME ATTRIBUTE INDICATING THAT THE VOLUME IS ATTACHED ONLY TO REQUESTING PROCESSES (SUBJECT TO ACCESS CONTROLS)

PARTITION: A REGION WITHIN A PHYSICAL VOLUME SET ASIDE FOR SPECIAL USE

RECORD: A LOGICAL UNIT OF DISK SPACE, 1024 CONTIGUOUS WORDS IN SIZE. (NUMBERED/ADDRESSED FROM ZERO)
goes with Page Frame which is Kept in memory

SECTOR: A LOGICAL UNIT OF DISK SPACE, 64 CONTIGUOUS WORDS IN SIZE. THE SMALLEST ADDRESSABLE UNIT OF DISK SPACE. A RECORD CONTAINS 16 SECTORS

VOLUME MANAGEMENT TERMINOLOGY

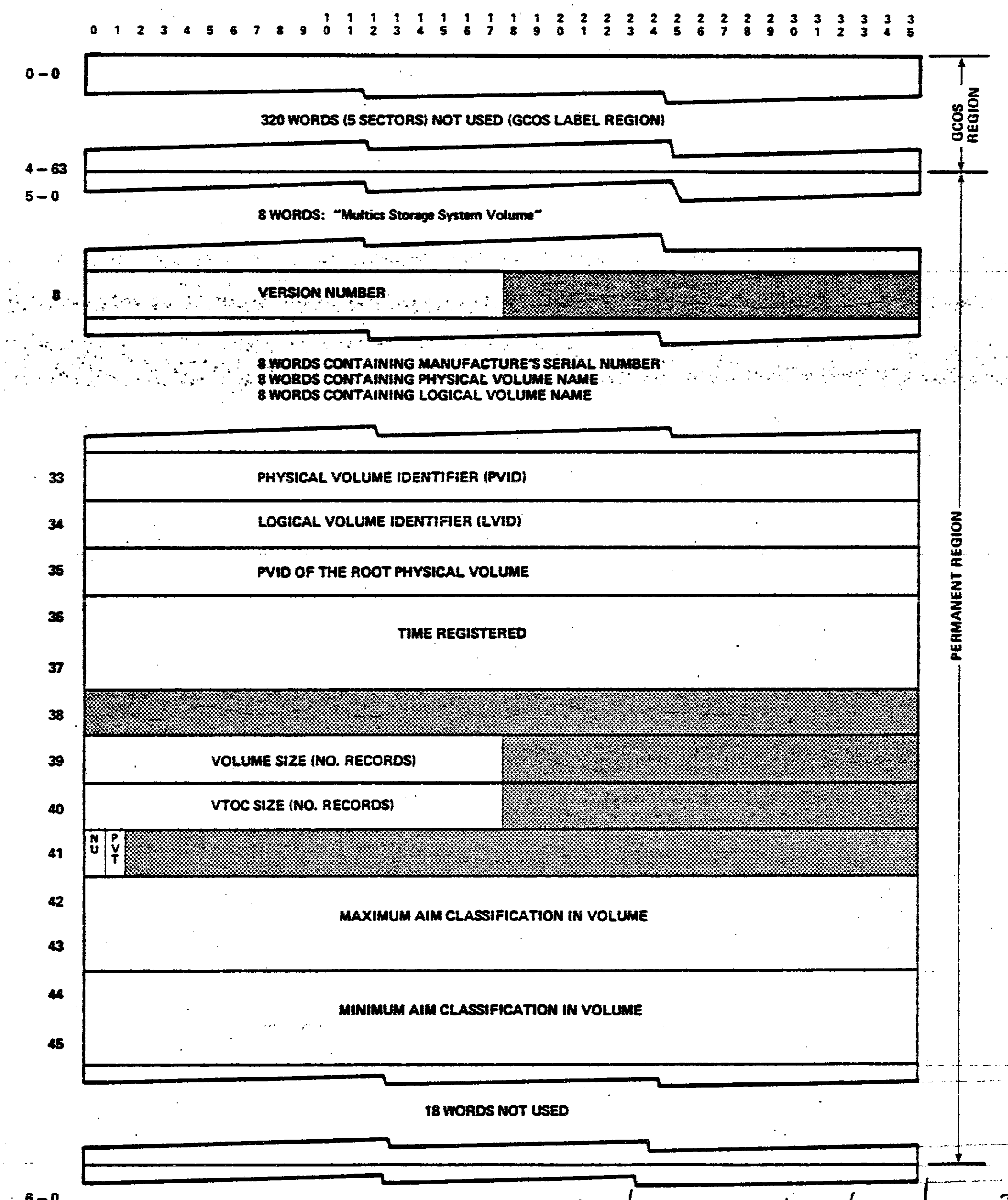
PAGE: A 1024 WORD EXTENT OF DATA STARTING AT A 1024 WORD BOUNDARY OF A SEGMENT. SEGMENTS MUST BE AN INTEGER NUMBER OF PAGES IN SIZE. A PAGE CAN RESIDE IN ONE OR MORE OF THE FOLLOWING LOCATIONS:

MAIN MEMORY FRAME

DISK RECORD

VOLUME MANAGEMENT DATA BASES

VOLUME LABEL



Time_map_updated } equal when demounted.
Time_mounted } If unequal, then disk hasn't demounted, i.e. esd'less crash.

VOLUME LABEL

(SECTOR–WORD)	Contents	
6-0	VOLMAP VERSION (OVERLAYS OLD TIME–UNMOUNTED)	DYNAMIC INFORMATION
6-1		
6-2	TIME SALVAGED	
6-4	TIME OF BOOTLOAD	
6-6	TIME UNMOUNTED	
	8 WORDS USED BY VOLUME DUMPER 48 WORDS UNUSED	
7-0	HERE	ROOT INFORMATION AND VOLUME DESCRIPTION
7-1	VTOC INDEX OF ">" DIRECTORY	
7-2	SHUTDOWN STATE	
7-3		
7-4	DISK_TABLE_ VTOC INDEX	
7-5	DISK_TABLE_ UNIQUE ID	
7-6	ESD STATE	
7-7	VOLUME MAP ORIGIN RECORD	
7-8	VOLUME MAP SIZE	
7-9	VTOC MAP ORIGIN RECORD	
7-10	VTOC MAP SIZE	
7-11	VOLUME MAP SECTION SIZE	
7-12	VTOC ORIGIN RECORD	
7-13	DUMPER BIT MAP RECORD	
	VOLUME INCONSISTENCY COUNT	
	52 WORDS UNUSED	
8–3	NUMBER OF PARTITIONS	
4	PARTITION (1) NAME	PARTITION MAP
5	PARTITION (1) LOCATION	
6	PARTITION (1) SIZE	
7		
	188 WORDS DESCRIBING PARTITIONS 2 THRU 47	
11–0	320 WORDS (5 SECTORS) NOT USED	NOT USED
15 – 63		

(SECTOR–WORD) (fs_vol_label.incl.pl1)

VOLUME LABEL

A DISK RESIDENT DATA BASE – ONE PER PHYSICAL VOLUME

VOLUME MANAGEMENT DATA BASES

VOLUME LABEL

- THE VOLUME LABEL IS A DISK RESIDENT DATA BASE OCCUPYING THE FIRST MULTICS RECORD OF EACH STORAGE SYSTEM PHYSICAL VOLUME

  - ONE VOLUME LABEL PER PHYSICAL VOLUME

- THE LABEL IS GENERATED BY init_disk_pack (init_empty_root IF RPV LABEL) AND CONTAINS REGISTRATION AND STATUS INFORMATION

- THE LABEL IS INSPECTED WHEN THE VOLUME IS ACCEPTED AND UPDATED WHEN DEMOUNTED

- THE LABEL IS DIVIDED INTO SIX SECTORS

  - GCOS REGION (SECTORS 0 TO 4)

    - SKIPPED OVER BY MULTICS TO AVOID ACCIDENTAL OVERWRITING OF GCOS PACKS AND ALLOW FOR FUTURE COMPATABILITY

  - PERMANENT REGION (SECTOR 5)

    - CONTAINS PERMANENT PER-PV INFORMATION (EG: MANUFACTURERS SERIAL NUMBER)

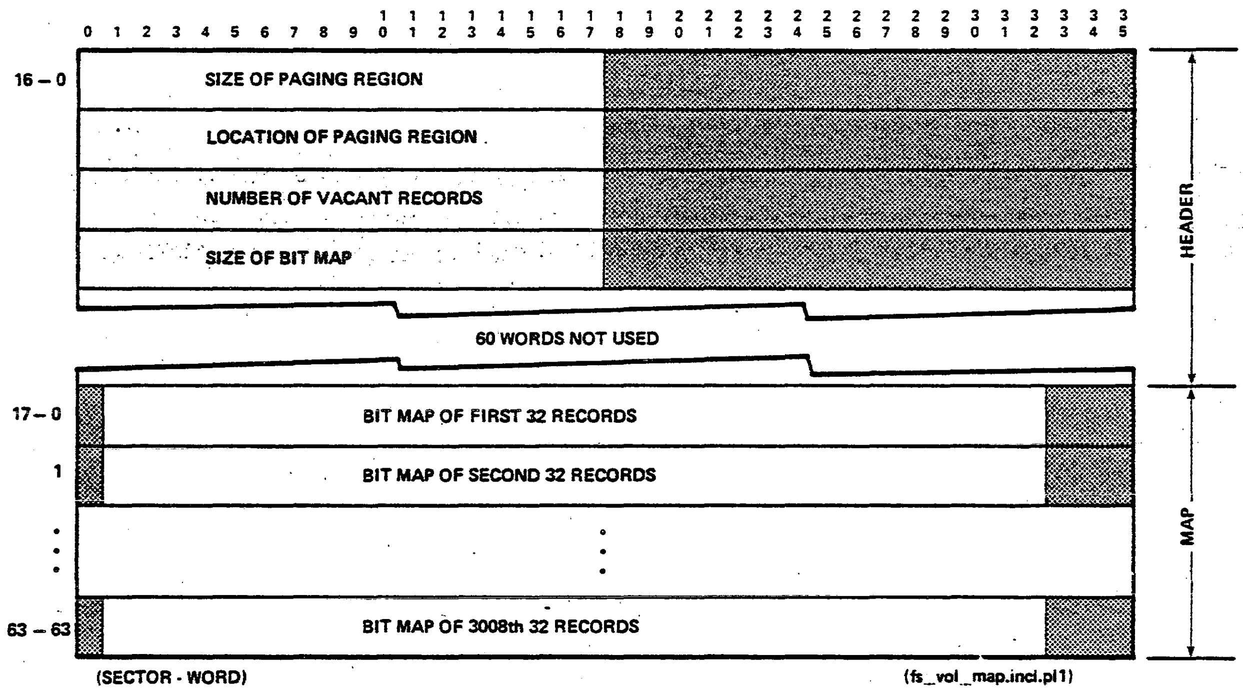
VOLUME MANAGEMENT DATA BASES

VOLUME LABEL

- DYNAMIC INFORMATION REGION (SECTOR 6)
  - CONTAINS INFORMATION RELATING TO THE MOST RECENT MOUNTING OF THE PV (EG: LAST MOUNT TIME)
  - ALLOWS THE STORAGE SYSTEM TO ENSURE THE INTEGRITY OF THE PV
- ROOT INFORMATION REGION (SECTOR 7)
  - DEFINED ONLY FOR THE ROOT PHYSICAL VOLUME
  - CONTAINS DYNAMIC INFORMATION ABOUT THE ENTIRE STORAGE SYSTEM (EG: SHUT DOWN STATE, PAGING DEVICE STATE, ETC)
- PARTITION MAP (SECTOR 8)
  - IDENTIFIES THE LOCATION AND LENGTH OF ANY RESIDENT PARTITIONS
- UNUSED (SECTORS 9 TO 13)

VOLUME MANAGEMENT DATA BASES

VOLUME MAP



VOLUME MAP

A DISK RESIDENT DATA BASE – ONE PER PHYSICAL VOLUME

VOLUME MANAGEMENT DATA BASES

VOLUME MAP

- THE VOLUME MAP IS A DISK RESIDENT DATA BASE

  - OCCUPIES RECORDS 1, 2, AND 3, IMMEDIATELY FOLLOWING THE VOLUME LABEL

  - ONE VOLUME MAP PER PHYSICAL VOLUME

- THE VOLUME MAP IDENTIFIES THE EXTENT OF THE PAGING REGION, THE NUMBER OF VACANT RECORDS, AND THE STATE (VACANT/OCCUPIED) OF EVERY RECORD IN THE VOLUME'S PAGING REGION

  - THIS INFORMATION IS ALSO DERIVABLE FROM AN ANALYSIS OF THE VTOC (AT CONSIDERABLE EXPENSE) - THIS IS DONE WHEN THE VOLUME IS SCAVENGED OR SALVAGED.

- RECORDS ARE TAKEN FROM THE VOLUME MAP DURING OPERATION AND PLACED IN THE RECORD STOCK

  - RECORDS ARE ALLOCATED BY PAGE CONTROL FROM THE RECORD STOCK

  - VOLUME MAP ON DISK IS ALWAYS CONSISTENT

    - RECORDS MARKED FREE ON DISK ARE GUARANTEED TO BE FREE, AND SAFE TO RE-USE

VOLUME MANAGEMENT DATA BASES

VOLUME MAP

never reverse

- RECORDS MARKED AS ALLOCATED MAY NOT ACTUALLY BE IN-USE, IF A CRASH OCCURRED AND DESTROYED THE STOCK CONTENTS

- THIS SITUATION IS BENIGN, AND CORRECTED BY A SCAVENGE OR SALVAGE AT SOME CONVENIENT TIME

- RECORDS FREED ARE PLACED BACK IN THE STOCK

  - IF STOCK FILLS, IT IS WRITTEN BACK TO THE VOLUME MAP

- CONSISTENCY IS ENSURED BY COMPLEX PROTOCOL IN PAGE CONTROL

- RECORD STOCK MECHANISM REPLACES FSMAP SEGMENTS IN PRE-MR10.0 SYSTEMS

DUMPER BIT MAP

0	NUMBER OF VTOCES
1	NUMBER OF FREE VTOCES
2	NUMBER OF WORDS USED IN BIT MAP
3	NUMBER OF LAST RECORD IN VTOC
	4 WORDS UNUSED
8	VTOC BIT MAP ENTRY 0
	. . . .
1023	VTOC BIT MAP ENTRY 1015

VOLUME DUMPER BIT MAP
RECORDS 4 AND 5

0	
	8 WORDS UNUSED
8	VOLUME DUMPER BIT MAP ENTRY 0
	. . . .
2047	VOLUME DUMPER BIT MAP ENTRY 2039

VTOC MAP

RECORD 6

VOLUME MANAGEMENT DATA BASES

DUMPER BIT MAP

- THE DUMPER BIT MAP DESCRIBES WHICH VTOCES ON THE VOLUME HAVE BEEN VOLUME DUMPED

  - ONE PER PHYSICAL VOLUME

  - OCCUPIES RECORDS 4 AND 5, IMMEDIATELY FOLLOWING VOLUME MAP

  - SEPARATE BIT MAPS FOR INCREMENTAL AND CONSOLIDATED VOLUME DUMPS

VOLUME MANAGEMENT DATA BASES

VTOC MAP

- THE VTOC MAP DESCRIBES THE LOCATION AND SIZE OF THE VTOC, AND CONTAINS A BIT MAP OF VTOC ALLOCATIONS

  - OCCUPIES RECORD 6, IMMEDIATELY FOLLOWING THE DUMPER BIT MAP

    - VTOC FOLLOWS, STARTING AT RECORD 8

    - RECORD 7 IS UNUSED

- VTOCES ARE TAKEN FROM THE VTOCE MAP AND PLACED IN AN ONLINE STOCK

  - VTOCE STOCK IS LIKE RECORD STOCK, BUT LESS CRITICAL

    - VTOCES ARE SELF-IDENTIFYING AS TO WHETHER THEY ARE IN USE OR NOT, SO IT IS NOT NECESSARY TO MAINTAIN PERFECT CONSISTENCY IF A CRASH OCCURS

    - VTOCE MAP IS ALSO REBUILT BY SCAVENGE OR SALVAGE OPERATIONS

  - VTOCE MAP REPLACES VTOCE FREE LIST IN PRE-MR10.0 SYSTEMS

VOLUME MANAGEMENT DATA BASES

0	NUMBER OF ENTRIES
1	MAXIMUM NUMBER OF ENTRIES
2	NUMBER OF ENTRIES IN USE
	5 WORDS FOR SHUTDOWN STATUS
8	TIME OF BOOTLOAD
10	9 WORDS OF INFORMATION ABOUT RPV AND RLV
19	21 WORDS OF GLOBAL INFORMATION AND METERING CELLS FOR VOLUME MAPS AND VTOC MAPS
40	ARRAY OF PHYSICAL VOLUME TABLE ENTRIES

PHYSICAL VOLUME TABLE ENTRY (PVTE)

ONE PER PHYSICAL VOLUME
LOCATED IN PHYSICAL VOLUME TABLE

dbm_seg

PVTX

0	PHYSICAL VOLUME ID		
1.	LOGICAL VOLUME ID		
2	DIU DIU DIU	SKIPPED FOR PDIR ALLOCATION	NEXT PV IN LV
3	DISK SUBSYSTEM NAME ie. dska_		
4	DISK DRIVE TYPE	DISK DRIVE NUMBER ie. 1	USED SS PERM TEST BM BDM CRI DI RPV SCAV VOLM BDMZ VAC HCU VLN VIN VTLN
5	FREE VTOCE COUNT	VTOC SCZE (IN RECORDS)	
6	INC. DUMPER BIT MAP POINTER	CONS. DUMPER BIT MAP POINTER	
7	FREE RECORD COUNT	TOTAL RECORD COUNT	
8	DISK DIM INFORMATION		
9	INC. DUMPER VTOC INDEX	CONS. DUMPER VTOC INDEX	
10	COMP. DUMPER VTOC INDEX	TOTAL VTOCE COUNT	
11	FIRST RECORD IN PAGING REGION		
	9 WORDS FOR VOLUME MAP AND VTOC MAP INFORMATION		
22	VOLUME TROUBLE COUNT	SCAVENGER INFO POINTER	

Part of inconsistency

PHYSICAL VOLUME TABLE HEADER

A HARDCORE (WIRED) DATABASE - ONE PER SYSTEM

PHYSICAL VOLUME TABLE (PVT)

- THE PHYSICAL VOLUME TABLE (PVT) IS A HARDCORE, WIRED, PAGED DATA BASE MAINTAINED BY VOLUME MANAGEMENT

  - ONE PVT PER SYSTEM

- THE PVT IS THE MOST IMPORTANT DATA BASE OF VOLUME MANAGEMENT, AND CONTAINS AN ARRAY OF PHYSICAL VOLUME TABLE ENTRIES (PVTE'S)

  - ONE PVTE PER DISK DRIVE KNOWN TO THE SYSTEM

- EACH PVTE DESCRIBES:

  - A DISK DRIVE CONFIGURED TO THE SYSTEM

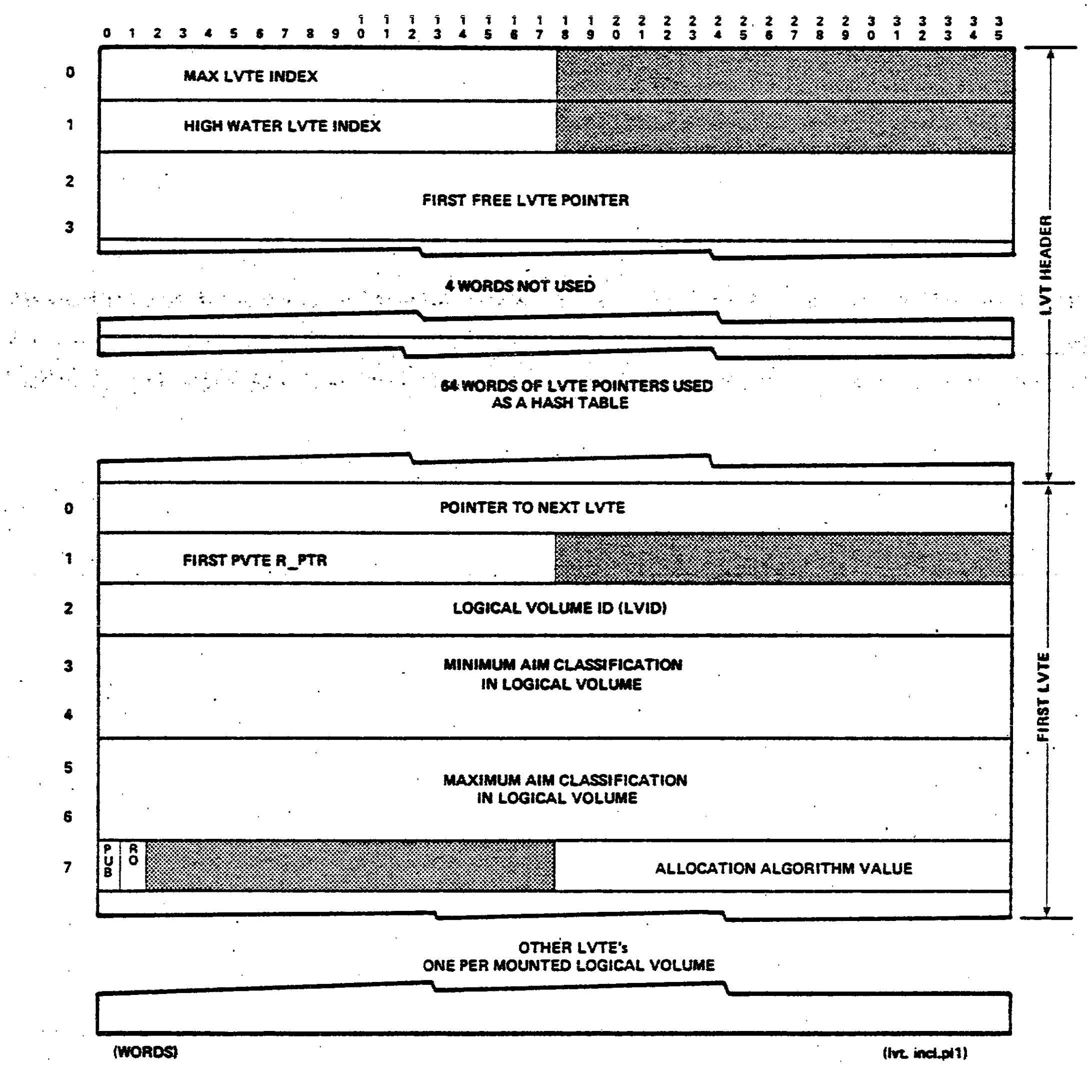
    - INCLUDING THE DEVICE NUMBER, DEVICE TYPE, SUBSYSTEM NAME AND OTHER INFORMATION NEEDED BY THE DISK DIM

  - THE PHYSICAL VOLUME CURRENTLY MOUNTED ON THE DISK DRIVE

    - INCLUDING THE PVID, LVID, AND OTHER INFORMATION TAKEN FROM THE VOLUME HEADER, VOLUME MAP AND VTOC MAP, NEEDED BY PAGE AND SEGMENT CONTROL (THE PV & LV NAMES ARE <u>NOT</u> RECORDED HERE)

    - RECORD AND VTOCE STOCKS ARE LOCATED FROM THE PVT, BUT KEPT IN THE stock_seg

LOGICAL VOLUME TABLE (LVT)



LOGICAL VOLUME TABLE (LVT)

A HARDCORE DATA BASE – ONE PER SYSTEM

LOGICAL VOLUME TABLE ENTRY (LVTE)

ONE PER MOUNTED LOGICAL VOLUME

LOGICAL VOLUME TABLE (LVT)

- THE LOGICAL VOLUME TABLE (LVT) IS A HARDCORE, PAGED DATA BASE MAINTAINED BY VOLUME CONTROL

  - ONE LVT PER SYSTEM

- THE LVT CONTAINS AN ARRAY OF LOGICAL VOLUME TABLE ENTRIES (LVTE'S)

  - ONE LVTE FOR EACH MOUNTED LOGICAL VOLUME

- EACH LVTE DESCRIBES THE LOGICAL VOLUME TO INCLUDE:

  - LVID AND AIM CLASSIFICATION

  - RELATIVE POINTER TO THE THREAD OF PVTE'S OF ACCEPTED PHYSICAL VOLUMES THAT ARE MEMBERS OF THE LOGICAL VOLUME

LOGICAL VOLUME TABLE (LVT)

- THE LVT IS REQUIRED AT THE FOLLOWING TIMES:
  - SEGMENT CREATION
  - SEGMENT MOVING TIME
  - VOLUME MOUNTING AND DEMOUNTING
  - INITIATION AND SEGMENT FAULT TIME (FOR PUBLIC/PRIVATE CHECK)

VOLUME MANAGEMENT DATA BASES

PHYSICAL VOLUME HOLD TABLE

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35

0	PVT R_PTR (1)	APTE R_PTR (1)
1	PVT R_PTR (2)	APTE R_PTR (2)
⋮	⋮	⋮
63	PVT R_PTR (64)	APTE R_PTR (64)

(WORD) (pv_holdt.incl.pl1)

PHYSICAL VOLUME HOLD TABLE

AN INTERNAL STATIC ARRAY IN get_pvtx – ONE PER SYSTEM

- THE PHYSICAL VOLUME HOLD TABLE (PVHT) IS A HARDCORE DATA BASE MAINTAINED BY VOLUME MANAGEMENT

  - ONE PVHT PER SYSTEM

- THE PVHT IDENTIFIES THE PHYSICAL VOLUME AND THE PROCESS ID OF PROCESS THAT HAS STARTED (AND HAS NOT YET COMPLETED) COMPOUND OPERATIONS UPON THE PHYSICAL VOLUME

- THIS INFORMATION PREVENTS A VOLUME FROM BEING DEMOUNTED WHILE SUCH AN OPERATION IS IN PROGRESS

VOLUME MANAGEMENT DATA BASES

PHYSICAL VOLUME HOLD TABLE

- INTERRUPTION OF A COMPOUND OPERATION CAUSES THE VOLUME TO BE MARKED AS CONTAINING AN INCONSISTENCY

- FOR CRASH ANALYSIS, sst.pvthp CONTAINS A POINTER TO THIS TABLE

VOLUME MANAGEMENT OPERATIONS

ACCEPTANCE OF PHYSICAL VOLUMES

- THE ACCEPTANCE OF PHYSICAL VOLUMES IS THE MOST IMPORTANT AND FUNDAMENTAL OPERATION OF VOLUME MANAGEMENT

- PHYSICAL VOLUME ACCEPTANCE IS ACCOMPLISHED BY CALLING initializer_gate_$accept_fs_disk

- THE ROOT PHYSICAL VOLUME (RPV) IS ACCEPTED IN A SPECIAL FASHION DURING COLLECTION 2 OF BOOTLOAD

  - THE RPV IS THE ONLY PV REQUIRED TO BOOTLOAD THE SYSTEM (MORE OF THE RLV WILL BE ACCEPTED BY RING ZERO DURING BOOTLOAD IF POINTED TO BY THE "ROOT" CONFIGURATION CARD)

- ACCEPTANCE INCLUDES:

  - VALIDATE THAT THE DISK PACK MOUNTED IS THE PACK REQUESTED BY THE OPERATOR OR REQUESTING PROCESS VIA label.pvid

  - DETERMINE THAT THE DISK PACK MOUNTED IS IN FACT A MEMBER OF <u>THIS</u> HIERARCHY VIA label.root_pvid

  - INITIALIZING THE APPROPRIATE PVTE WITH DATA FROM THE LABEL, VOLUME MAP, AND VTOC MAP

ACCEPTANCE OF PHYSICAL VOLUMES

- INITIALIZING THE INITIAL CONTENTS OF THE RECORD STOCK AND VTOCE STOCKS

- DETERMINING IF ANY VOLUME INCONSISTENCIES ARE PRESENT, AND LOGGING THIS INFORMATION

  - VOLUME INCONSISTENCIES ARE CAUSED BY EVENTS WHICH MAY MEAN THAT THE DISK RESIDENT COPY OF THE VOLUME MAP OR VTOC MAP IS INCONSISTENT:

    - A CRASH WITHOUT ESD - INDICATED BY label.time_map_updated and label.time_unmounted BEING UNEQUAL, DETECTED AT ACCEPTANCE TIME

    - AN INCONSISTENCY DETECTED ONLINE, SUCH AS AN INVALID VTOC BIT MAP OR A REUSED ADDRESS

    - AN I/O ERROR WHEN WRITING THE VOLUME MAP OR VTOC MAP DURING NORMAL OPERATION

  - A COUNT IS KEPT IN THE LABEL, AND UPDATED AS NECESSARY

  - NORMALLY, INCONSISTENCIES ARE MERELY LOGGED, AND LEFT FOR THE SITE TO TAKE CARE OF AT SOME CONVENIENT TIME

    - IF AN RLV VOLUME CLAIMS ONLY A VERY SMALL NUMBER OF FREE PAGES, A VOLUME SALVAGE IS DONE AUTOMATICALLY TO TRY TO RECOVER ANY LOST DUE TO THE INCONSISTENCY, SINCE A FULL RLV WILL CAUSE SYSTEM CRASHES

- WRITING OUT THE LABEL TO UPDATE label.time_map_updated.

  - NOTE: label.time_map_updated AND label.time_unmounted ARE NOW UNEQUAL

  - THIS INEQUALITY IMPLIES THAT THE VOLUME HAS NOT BEEN PROPERLY SHUT DOWN, AND WILL BE MARKED INCONSISTENT IF ACCEPTED AGAIN IN THIS STATE

- MARKING THE PVTE AS "IN USE" (LAST STEP)

VOLUME MANAGEMENT OPERATIONS

DEMOUNTING OF PHYSICAL VOLUMES

- THE DEMOUNTING OF PHYSICAL VOLUMES INVOLVES REVERSING ALL OF THE STEPS TAKEN AT ACCEPTANCE TIME

- DEMOUNTING IS COMPLICATED BY THE FACT THAT THE PV MAY BE IN USE AT THE TIME

- DEMOUNTING IS ACCOMPLISHED BY CALLING demount_pv ("THE DEMOUNTER")

- ALL VOLUMES ARE DEMOUNTED AT SHUTDOWN TIME

- DEMOUNTING INCLUDES:

  - TURNING ON pvte.being_demounted AND WAITING FOR ALL COMPOUND OPERATIONS TO TERMINATE

  - DEACTIVATING ALL SEGMENTS FROM THE PV WHICH ARE ACTIVE. THIS INCLUDES:

    - FLUSHING MAIN MEMORY AND PAGING DEVICE (IF PRESENT) OF ALL RELEVANT PAGES

DEMOUNTING OF PHYSICAL VOLUMES

- UPDATING THE VTOCE'S FROM THE ASTE'S AND PAGE TABLES

- FLUSHING THE VTOC MANAGER'S BUFFER SEGMENT OF ALL RELEVANT VTOCE-PARTS

- EMPTYING THE RECORD AND VTOCE STOCKS BACK INTO THE VOLUME MAP AND VTOC MAP

- UPDATING THE VOLUME LABEL FROM THE PVTE, PARTICULARLY

  - label.time_unmounted, label.time_map_updated, AND label.inconsistency_count

- PHYSICALLY CYCLING DOWN THE DISK DRIVE

  - NOT DONE AT SYSTEM SHUTDOWN, HOWEVER

- ONLY ONE PV MAY BE DEMOUNTED AT A TIME

VOLUME MANAGEMENT OPERATIONS

LOGICAL VOLUME MANAGEMENT

- LOGICAL VOLUME MANAGEMENT INCLUDES:

  - MAINTAINING THE LOGICAL VOLUME TABLE (LVT) TO REFLECT THE STATE OF THE LOGICAL VOLUMES

  - MAINTAINING, IN THE KNOWN SEGMENT TABLE (KST) OF EACH PROCESS, A TABLE OF PRIVATE LOGICAL VOLUMES MOUNTED TO THE PROCESS

  - ANSWERING THE QUESTION OF WHETHER OR NOT A GIVEN LOGICAL VOLUME IS MOUNTED TO THE CALLING PROCESS

    - OR, IF A PUBLIC LV, MOUNTED AT ALL (TO THE SYSTEM)

  - PROVIDING THE HEAD OF THE PVT CHAIN FOR A GIVEN LV, FOR THE SEGMENT CREATION FUNCTION

- PRINT_CONFIGURATION_DECK - DISPLAYS >sl1>config_deck, WHICH CONTAINS INFORMATION ABOUT DISK LOCATIONS, THE RLV, AND PARTITIONS

  - ONLY THE PART OF THE CONFIG DECK RELEVANT TO VOLUME MANAGEMENT AND DISK CONFIGURATION IS SHOWN HERE

```
root  dska  16.  dskb  25.  dskb  23.  dskb  24.  dska  8.

part  bos  dska  16.
part  dump  dska  16.
part  log  dska  16.

prph  dska  a  20.  2  451.  16.
chnl  dska  a  26.  2  b  24.  2  b  22.  2

prph  dskb  b  20.  2  0  16.  451.  16.
chnl  dskb  b  26.  2  a  24.  2  a  22.  2

prph  dskc  a  28.  2  501.  32.
chnl  dskc  a  30.  2  b  30.  2  b  28.  2

prph  dske  b  32.  2  451.  8.
chnl  dske  b  34.  2

prph  dskf  a  32.  2  501.  16.
chnl  dskf  a  34.  2

mpc  mspa  451.  a  20.  4  a  24.  4
mpc  mspb  451.  b  20.  4  b  24.  4
mpc  mspc  607.  a  28.  4
mpc  mspd  607.  b  28.  4
mpc  mspe  451.  b  32.  4
mpc  mspf  607.  a  32.  4
```

- DISK CONFIGURATION CONFIG CARDS

  - ROOT

    - IDENTIFIES THOSE VOLUMES IN THE ROOT LOGICAL VOLUME WHICH HAVE HC PARTITIONS, USED BY THE SUPERVISOR FOR PAGING OF SUPERVISOR SEGMETNS

  - PART

    - IDENTIFIES THE LOCATIONS OF CERTAIN IMPORTANT PARTITIONS

    - ONLY PARTITIONS NECESSARY FOR MULTICS OPERATIONS ARE IDENTIFIED, NOT ALT PARTITIONS

    - HC PARTITIONS ARE LOCATED BY THE ROOT CARD

  - PRPH DSK_n_, CHNL

    - IDENTIFY PHYSICAL I/O CHANNEL PATHS FOR ACCESSING DISK DRIVES

  - MPC

    - IDENTIFY PHYSICAL CONNECTIONS TO MICROPROGRAMMED DISK CONTROLLERS

list vols

- LIST_VOLS - DISPLAYS A TABLE OF ONLINE VOLUMES, THEIR LOCATION, AND SPACE UTILIZATION

Drive	Records	Left	%	VTOCEs	Left	%	Avg Size	PV Name	PB/PD	LV Name
dskc_17	64504	54730	85	13440	11356	84	4	alpha01	pb pd	Alpha
dskc_18	64504	55389	86	13440	11352	84	4	alpha02	pb pd	Alpha
dska_05	36428	5305	15	8400	2662	32	5	mul03	pb pd	Multics_Pubs
dska_06	36428	4323	12	8400	2365	28	5	mul01	pb pd	Multics_Pubs
dskb_19	36428	4632	13	8400	2813	33	5	mul02	pb pd	Multics_Pubs
dskb_26	36429	13690	38	8400	4326	52	5	mul05	pb pd	Multics_Pubs
dskb_27	36429	4672	13	8400	2333	28	5	mul04	pb pd	Multics_Pubs
dska_01	36308	4680	13	9000	450	5	3	pub01	pb pd	Public
dska_03	36268	3588	10	9200	1017	11	3	pub07	pb pd	Public
dska_04	36268	4500	12	9200	884	10	3	pub04	pb pd	Public
dska_09	36268	4816	13	9200	864	9	3	pub02	pb pd	Public
dskb_17	36269	4281	12	9200	1002	11	3	pub05	pb pd	Public
dskb_18	36268	3840	11	9200	539	6	3	pub08	pb pd	Public
dskc_13	64504	294	0	13440	5539	41	8	rel01		Release
dskc_14	64504	269	0	13440	5214	39	7	rel02		Release
dskc_01	64503	43631	68	13440	10032	75	6	xpub01	pb	Xpublic
dskc_02	64503	45502	71	13440	9838	73	5	xpub02	pb	Xpublic
dskc_03	64503	42374	66	13440	9839	73	6	xpub03	pb	Xpublic
dskc_04	64503	43591	68	13440	9785	73	5	xpub04	pb	Xpublic
dskc_09	64504	58010	90	13440	12394	92	6	xpub05	pb	Xpublic
dskc_10	64504	56786	88	13440	12407	92	7	xpub06	pb	Xpublic
dskc_21	64503	23947	37	13440	6446	48	5	ypub01	pb	Ypublic
dskc_22	64503	23744	37	13440	6194	46	5	ypub02	pb	Ypublic
dskc_29	64503	23794	37	13440	6185	46	5	ypub05	pb	Ypublic
dskc_30	64503	24111	37	13440	6481	48	5	ypub06	pb	Ypublic
dskc_07	64503	11723	18	13440	6149	46	7	zpub01	pb pd	Zpublic
dskc_08	64503	11665	18	13440	6429	48	7	zpub02	pb pd	Zpublic
dskc_23	64504	9777	15	13440	6094	45	7	zpub03	pb pd	Zpublic
dskc_24	64504	11805	18	13440	6141	46	7	zpub04	pb pd	Zpublic
dskc_25	64504	11514	18	13440	7407	55	8	zpub05	pb pd	Zpublic
dskc_26	64504	12958	20	13440	7149	53	8	zpub06	pb pd	Zpublic
dske_06	37089	8053	22	5100	3797	74	22	list01	pb pd	list_1
dska_12	37562	6046	16	2735	957	35	17	list02	pb	list_2
dska_07	37309	6825	18	4000	2107	53	16	list03	pb	list_3
dska_08	36209	3827	11	7000	491	7	4	root5	pb	root
dska_11	36209	8597	24	7000	4181	60	9	root6	pb	root
dska_16	31283	2892	9	9000	3476	39	5	rpv	pb	root
dskb_23	36208	4888	13	7000	455	7	4	root3	pb	root
dskb_24	36209	3097	9	7000	238	3	4	root4	pb	root
dskb_25	36350	4483	12	7000	223	3	4	root2	pb	root

- DISPLAY_LABEL - DISPLAYS THE LABEL OF A STORAGE SYSTEM VOLUME BY READING IT FROM DISK

  - USED / FREE INFORMATION IS COPY ON DISK, AND THEREFORE OUT OF DATE WITH RESPECT TO THE PVTE

```
Label for Multics Storage System Volume rpv on dska_01 d451
          PVID                  220531524345
          Serial                rpv
          Logical Volume        root
            LVID                220531524466

          Registered            01/28/81  1249.5
          Dismounted            03/15/83  0741.9
          Map Updated           03/15/83  0744.6
          Salvaged              10/01/82  0300.3
          Bootload              03/15/83  0743.5
          Reloaded              01/28/81  1510.1
          Dumped
            Incremental         03/17/83  2153.0
            Consolidated        03/16/83  2359.3
            Complete            03/15/83  2353.0

          Inconsistencies             0

          Minimum AIM           0:000000
          Maximum AIM           7:777777

          Volume contains Root (>) at vtocx 0
            disk_table_ at vtocx 100 (uid 033022210261)

Volume Map from Label

          First Rec   (Octal)      Size
                  0         0         8      Label  Region
                  8        10      2000      VTOC   Region
               2008      3730      2008      hc     Partition
               4016      7660     33901      Paging Region
              37917    112035       200      bos    Partition
              38117    112345       141      alt    Partition
                                  38258      Total  Size
```

- DISPLAY_PVTE - DISPLAYS THE PVT ENTRY OF A STORAGE SYSTEM VOLUME

  - PARTITION INFORMATION IS NOT DETAILED IN THE PVTE, BUT USED/FREE INFORMATION IS COMPLETELY UP TO DATE

```
PVTE for Multics Storage System Volume rpv on dska_01 d451 at pvt|50
        PVID                    220531524345
        LVID                    220531524466

        VTOCEs
          Number                   10000
          Left                      3323

        Records
          Number                   33901
          Left                      3796
        Inconsistencies                0

        Volume Map
          volmap_seg ASTE       15|4420
          record stock           76|100
          Page 0 - Base            7660
                   Free            3364
          Page 1 - Base          103660
                   Free            3740
          Page 2 - Base          203660
                   Free               0
          vtoce stock           76|2400

        ON:          storage_system permanent hc_part_used

        OFF:         being_mounted being_demounted being_demounted2
                     scav_check_address device_inoperative vacating
                     dmpr_in_use(incr) dmpr_in_use(cons) dmpr_in_use(comp)

Volume Map from PVTE

        First Rec   (Octal)      Size
            0           0          8      Label  Region
            8          10       2000      VTOC   Region
         2008        3740       2008      Partitions
         4016        7660      33901      Paging Region
        37917      112035        199      Partitions
                               38258      Total  Size
```

VOLUME MANAGEMENT METERS

disk meters

- DISK_METERS - DISPLAYS I/O ACTIVITY TO DISK DRIVES

  - ONLY ONE SUBSYSTEM SHOWN HERE TO CONSERVE SPACE

Total metering time 0:20:12

Subsystem dska	Count	Waits	%Waits	Avg. Wait(ms.)
call locks	26005	217	0.83	0.259
run locks	112	0	0.00	0.000
interrupt locks	25998	239	0.92	0.208
allocations	26001	0	0.00	0.000

Drive	Reads	Writes	Seek Distance	ATB Reads	ATB Writes	ATB I/O
1	269	67	214	4508	18102	3609
3	362	243	109	3350	4991	2004
4	309	131	184	3925	9258	2756
5	547	165	180	2217	7350	1703
6	631	165	161	1922	7350	1523
7	0	0	0	0	0	0
8	5843	2187	122	207	554	151
9	366	116	153	3313	10455	2516
11	3501	1431	200	346	847	245
12	0	0	0	0	0	0
16	7158	2508	135	169	483	125

* DEVICE_METERS - DISPLAYS SUMMARY OF I/O ACTIVITY FOR ALL DISK SUBSYSTEMS

Total metering time 0:20:13

	dska	dskb	dskc	dskd
Prior Page I/O	18571	17743	462	1273
ATB	65.334	68.383	2626.240	953.121
Other Page I/O	6525	5135	16	696
ATB	185.949	236.284	75832.692	1743.280
ATB Page I/O	48.347	53.034	2538.332	616.212
Prior VTOCE I/O	934	895	38	304
ATB	1299.061	1355.668	31929.554	3991.194
ATB I/O	46.612	51.037	2351.401	533.798
% Busy	76	74	0	4
Avg. Page Wait	47.289	46.197	20.341	24.666
Avg. Page ^Wait	176.082	101.023	36.996	61.704
Avg VTOCE Wait	41.138	37.610	38.595	29.090
Avg. Page I/O T	35.619	38.314	20.050	22.482
Avg. VTOCE I/O T	31.139	32.277	37.060	26.606
EDAC Corr. Errs	0	0	0	0
Errors	0	0	0	1
Fatal Errors	0	0	0	0

disk queue

- DISK_QUEUE - DISPLAYS I/O QUEUE FOR A DISK SUBSYSTEM

  - ONLY ONE SUBSYSTEM SHOWN HERE TO CONSERVE SPACE

```
Connects = 2604781, 1359725, 677321, 309367,
           123430, 40159, 10227, 1969.

P  RW VP DV   SECTOR        MEM

0  W  P  24  1350330   27304000
0  W  P   9  1020150    4432000
0  W  P  16  1204130   36246000
0  W  P  16   314370   27306000
0  W  P  16   314430   34166000
```

TOPIC VII

Segment Control

Page

SEGMENT CONTROL OVERVIEW

- FUNCTION

  - SEGMENT CONTROL IS RESPONSIBLE FOR THE MANAGEMENT OF LOGICAL MEMORY

  - ITS TASKS INCLUDE:

    - MAINTAINING THE DISK RESIDENT MAPS OF SEGMENTS (IE: THEIR VTOCE'S)

    - SEGMENT CREATION, TRUNCATION AND DELETION

    - SEGMENT ACTIVATION AND DEACTIVATION (ASTE MULTIPLEXING)

  - SEGMENT CONTROL CAN BE INVOKED EITHER BY SUBROUTINE CALLS OR BY SEGMENT FAULTS

- BASIC PHILOSOPHY OF ACTIVATION/DEACTIVATION

  - OF ALL SEGMENTS RESIDENT WITHIN THE SYSTEM'S MOUNTED PHYSICAL VOLUMES, ONLY A SMALL SUBSET WILL REQUIRE ACCESSING AT ANY ONE TIME. SUCH SEGMENTS WILL BE CALLED "ACTIVE SEGMENTS"

  - A PART OF MAIN MEMORY, CALLED THE "ACTIVE SEGMENT TABLE" (AST), WILL BE RESERVED TO HOLD MANAGEMENT INFORMATION FOR THESE ACTIVE SEGMENTS (IDENTITY, PVT INDEX, LOCATION OF PAGES, ETC.)

SEGMENT CONTROL OVERVIEW

- AS SEGMENTS FALL INTO DISUSE, THEIR "MANAGEMENT INFORMATION" IN THE AST WILL BE REPLACED WITH INFORMATION OF OTHER SEGMENTS REQUIRING ACTIVATION

- USER INTERFACE

  - COMMAND LEVEL

    - create, delete, truncate, etc.

  - SUBROUTINE LEVEL

    - hcs_$append_branch, hcs_$append_branchx, hcs_$delentry_seg, hcs_$delentry_file, hcs_$truncate_seg, hcs_$truncate_file, hcs_$force_write, etc

- MAJOR DATA BASES

  - SYSTEM SEGMENT TABLE (SST) - ONE PER SYSTEM, SHARED WITH PAGE CONTROL. ONE MAJOR COMPONENT IS "OWNED" BY SEGMENT CONTROL:

    - ACTIVE SEGMENT TABLE (AST) - ONE PER SYSTEM

    - THE AST IS A LIST OF ACTIVE (CURRENTLY BEING USED) SEGMENTS

    - ACTIVE SEGMENT TABLE ENTRY (ASTE) - ONE PER ACTIVE SEGMENT

      - ASTES CONTAIN PHYSICAL VOLUME ID'S (PVID'S) AND VTOC INDEX'S (VTOCX'S) OF SEGMENTS. NEEDED BY SEGMENT CONTROL TO FIND THE SEGMENT ON DISK (HARDWARE)

SEGMENT CONTROL OVERVIEW

- AST HASH TABLE

  - ALLOWS EFFICIENT SEARCHING OF ASTE'S

  - LOGICALLY PART OF THE AST, BUT ELSEWHERE FOR HISTORICAL REASONS

- DIRECTORY SEGMENTS

  - CONTAIN LOCATIONS AND ATTRIBUTES OF SEGMENTS. LOCATION INFORMATION FROM DIRECTORY SEGMENTS IS PROVIDED TO SEGMENT CONTROL BY DIRECTORY CONTROL

- VOLUME TABLE OF CONTENTS (VTOC) - ONE PER PHYSICAL VOLUME

  - VOLUME TABLE OF CONTENTS ENTRY (VTOCE) - ONE PER DISK-RESIDENT SEGMENT

  - EACH VTOCE CONTAINS THE SEGMENT'S UNIQUE ID, CURRENT LENGTH, FILE MAP, ETC (NEED TO BUILD ASTE'S AND PT'S)

  - VTOCES ARE READ AND WRITTEN ONLY BY SEGMENT CONTROL

- VTOCE STOCKS - FROM VOLUME MANAGEMENT

  - USED WHEN CREATING AND DELETING VTOCES FOR SEGMENTS

SEGMENT CONTROL TERMINOLOGY

MULTIPLEXING: CONTROLLED SHARING OF A REUSABLE RESOURCE

VTOC: VOLUME TABLE OF CONTENTS (ONE PER PV). AN ARRAY OF VTOCE'S IDENTIFYING ALL SEGMENTS RESIDENT ON THE PHYSICAL VOLUME

VTOCE: VOLUME TABLE OF CONTENTS ENTRY (ONE PER RESIDENT SEGMENT). CONTAINS IDENTIFICATION AND LOCATOR INFORMATION ABOUT A SEGMENT RESIDENT WITHIN THE PHYSICAL VOLUME

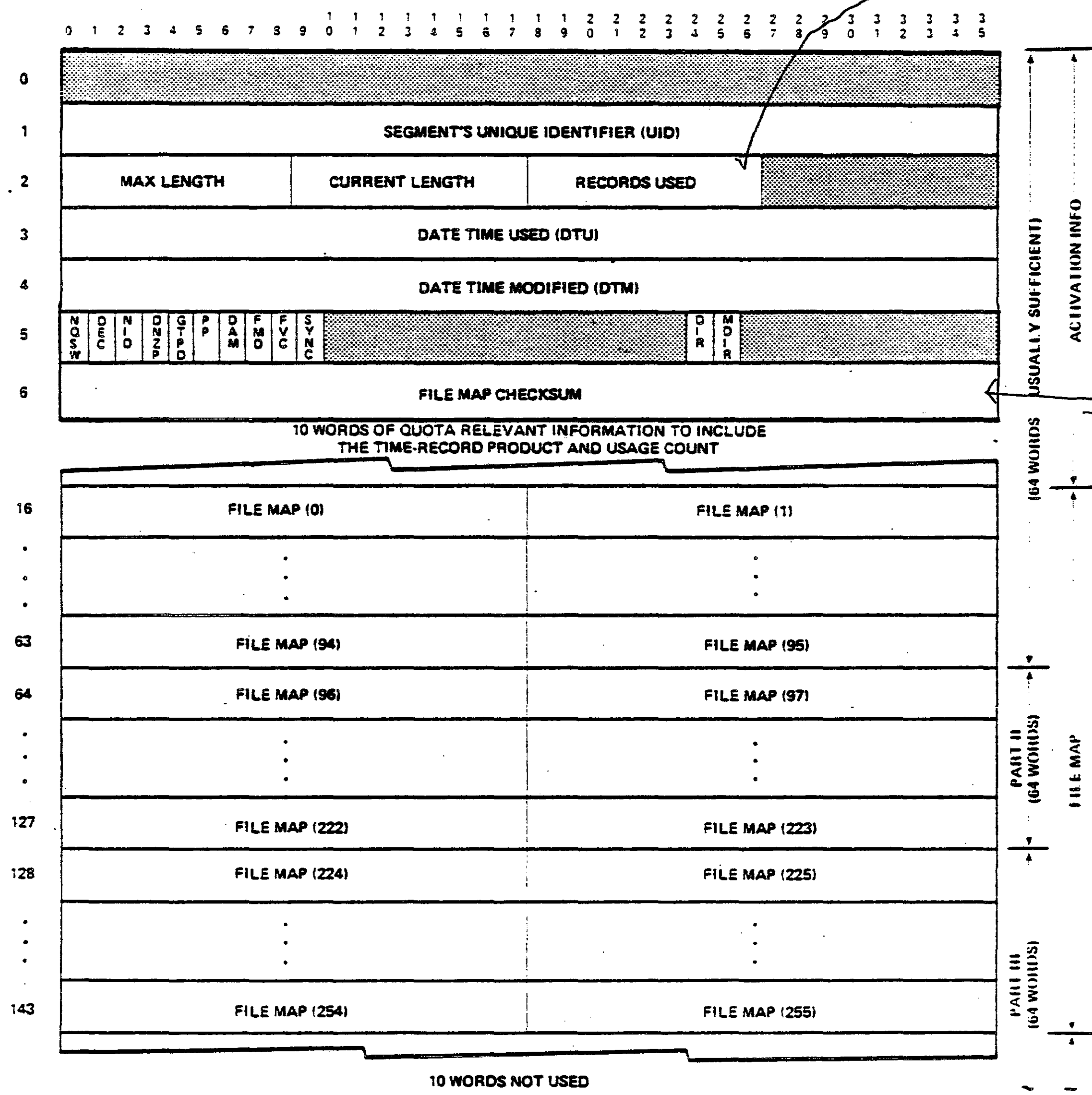
SEGMENT: A COLLECTION OF INFORMATION (PROCEDURE OR DATA) GROUPED TOGETHER UNDER THE SAME ACCESS CONTROL CONSTRAINTS. EACH SEGMENT IS GIVEN ONE OR MORE NAMES AND A COLLECTION OF ATTRIBUTES INCLUDING LENGTH, ACCESS PERMISSIONS, ETC

SEMI-PERMANENT ACTIVATION: A SEGMENT AND TURNING ON ITS ENTRY HOLD SWITCH (aste.ehs) PREVENTING NORMAL (ASTE CONTENTION) DEACTIVATION

SEGMENT CONTROL DATA BASES

VOLUME TABLE OF CONTENTS (VTOC)

number of non zero records



Allows you to use a disk after eddless crash.

Connection failure

directory entry + vtoce Broken

SEGMENT CONTROL DATA BASES

VOLUME TABLE OF CONTENTS (VTOC)

SEGMENT CONTROL DATA BASES

VOLUME TABLE OF CONTENTS VTOCE

154	NCD	
155	DATE TIME DUMPED (DTD)	
156	VOLUME ID (1)	(INCREMENTAL DUMP TAPE ID)
157	VOLUME ID (2)	(CONSOLIDATED DUMP TAPE ID)
158	VOLUME ID (3)	(COMPLETE DUMP TAPE ID)
159	UID OF MASTER DIRECTORY	
160	UID OF SUPERIOR DIRECTORY (0)	
	. . .	. . .
175	UID OF SUPERIOR DIRECTORY (15)	

8 WORDS CONTAINING THE SEGMENT'S ORIGINAL PRIMARY NAME

184	DATE_TIME VTOCE CREATED	
185	PVID OF VOLUME OF PARENT DIRECTORY	
186	PARENT VTOC INDEX	SEGMENT BRANCH OFFSET IN PARENT
187	TIME THIS VTOCE CHECKED FOR CONNECTION FAILURE	
188	ACCESS ISOLATION MECHANISM (AIM) CLASS	
190		
191	PVID OF VOLUME FOR THIS VTOCE	

PART III (64 WORDS)

PERMANENT INFO

(WORDS)

(vtoce.incl.pl1)

VOLUME TABLE OF CONTENTS ENTRY (VTOCE)

A DISK RESIDENT DATA BASE – ONE PER SEGMENT

SEGMENT CONTROL DATA BASES

VOLUME TABLE OF CONTENTS (VTOC)

- THE "VOLUME TABLE OF CONTENTS" (VTOC) IS A DISK RESIDENT DATA BASE CONTAINING (OF INTEREST HERE) AN ARRAY OF ENTRIES KNOWN AS "VOLUME TABLE OF CONTENTS ENTRIES" (VTOCE'S)

  - ONE VTOC PER PHYSICAL VOLUME

  - ONE VTOCE PER SEGMENT

- EACH VTOCE CONTAINS RESIDENCY INFORMATION (AND SOME ATTRIBUTE INFORMATION) OF A PARTICULAR SEGMENT

- EACH VTOCE IS ADDRESSED BY INDEXING INTO THE ARRAY OF VTOCE'S

  - CONSEQUENTLY, THE PAIR OF PVID AND VTOC INDEX UNIQUELY IDENTIFIES ANY SEGMENT IN THE STORAGE SYSTEM HIERARCHY

- EACH VTOCE IS 192 WORDS LONG AND IS DIVIDED INTO THREE LOGICAL PARTS:

  - ACTIVATION INFORMATION (16 WORDS)

    - CONTAINS ALL INFORMATION (EXCLUDING THE FILE MAP) NEEDED TO <u>USE</u> THE SEGMENT, OR MORE TECHNICALLY, TO ACTIVATE THE SEGMENT

SEGMENT CONTROL DATA BASES

VOLUME TABLE OF CONTENTS (VTOC)

    - INCLUDES: UID, CURRENT LENGTH, RECORDS USED, MAXIMUM LENGTH, RECORDS USED, ETC

  - ALL INFORMATION LIKELY TO CHANGE BECAUSE OF THE ACTIVATION

    - INCLUDES: DATE TIME MODIFIED AND USED, QUOTA CELLS (IF A DIRECTORY), ETC

- FILE MAP (128 WORDS)

  - AN ARRAY OF 256 RECORD ADDRESS OR NULL ADDRESS DETAILING WHERE EACH PAGE OF THE SEGMENT RESIDES

  - A NULL ADDRESS (NOT TO BE CONFUSED WITH A NULLED ADDRESS -- DISCUSSED LATER) INDICATES THAT NO RECORD OF THE VOLUME IS ASSIGNED TO THAT PAGE OF THE SEGMENT

  - A RECORD ADDRESS IS THE ADDRESS OF THE RECORD ASSIGNED TO THAT PAGE OF THE SEGMENT (I.E., THE DISK RESIDENT HOME OF THE PAGE)

  - NOTE: PAGE CONTROL ENSURES THAT NO RECORD ADDRESS EVER APPEARS (OR REMAINS) IN THE FILE MAP UNLESS THE PAGE ACTUALLY APPEARS ON THE VOLUME

- PERMANENT INFORMATION (48 WORDS)

  - CONTAINS ATTRIBUTES WHICH RARELY (IF EVER) CHANGE SUCH AS:

    - UID'S OF SUPERIOR DIRECTORIES, AIM CLASSIFICATION, DATE TIME DUMPED (BY PHYSICAL VOLUME DUMPER)

VOLUME TABLE OF CONTENTS (VTOC)

- EACH VTOCE IS ALSO DIVIDED INTO THREE PHYSICAL PARTS:
  - FIRST SECTOR (WORDS 0-63):
    - CONTAINS ALL "ACTIVATION INFORMATION" AND THE FIRST PORTION OF THE FILE MAP
  - SECOND SECTOR (WORDS 64-127):
    - CONTAINS THE BULK OF THE FILE MAP
  - THIRD SECTOR (WORDS 128-191):
    - CONTAINS THE END OF THE FILE MAP AND ALL "PERMANENT INFORMATION"

VOLUME TABLE OF CONTENTS (VTOC)

- VTOCE I/O

  - USING RECORD I/O (IN UNITS OF 1024 WORDS) TO ACCESS A VTOCE (192 WORDS) WOULD HAVE EXCESSIVE OVERHEAD FOR BUFFERS

  - FURTHERMORE, BECAUSE MOST SEGMENTS ARE SMALL, MOST VTOCE ACCESSING IS ONLY CONCERNED WITH ACTIVATION INFORMATION AND THE FIRST PORTION OF THE FILE MAP, I.E., THE FIRST SECTOR (64 WORDS)

  - TO TAKE ADVANTAGE OF THESE FACTS, VTOCE'S ARE ACCESSED VIA SECTOR I/O, NOT RECORD I/O

  - A LARGE MECHANISM KNOWN AS THE VTOC MANAGER (vtoc_man) EXIST TO EFFICIENTLY MANAGE THIS SECTOR I/O AND ITS BUFFERING

  - VTOCE I/O IS THE ONLY NON-PAGE I/O DONE TO DISK

  - VTOCE I/O IS DONE IN PARTS (SECTORS)

    - FOR A SEGMENT, OR A DIRECTORY WITHOUT TERMINAL QUOTA, PARTS ONE AND TWO CAN BE WRITTEN ENTIRELY FROM INFORMATION DERIVED FROM THE ASTE, AND NEED NOT BE READ IN FIRST

    - PART THREE MUST ALWAYS BE READ BEFORE BEING WRITTEN, AS MUST PART ONE FOR A DIRECTORY WITH QUOTA

    - ALL THREE PARTS OF A VTOCE ARE ALWAYS READ WHENEVER ANY PART IS REQUESTED, IN CASE THE OTHERS ARE NEEDED SOON AFTERWARDS

SEGMENT CONTROL DATA BASES

ACTIVE SEGMENT TABLE (AST)

Word	Bits 0–17	Bits 18–35
0	FORWARD R_PTR	BACKWARD R_PTR
1	NEXT BROTHER R_PTR	FIRST SON R_PTR
2	SYSTEM TRAILER R_PTR	PARENT R_PTR
3	SEGMENT'S UNIQUE IDENTIFIER (UID)	
4	MAX LENGTH, PV TABLE INDEX	VTOC INDEX
5	USED, INIT, GTUS, GTMS, HC, HCSD, AAO, WAO, INHC, EOOK, DEA, HCP, FMD, (unused), DIUS, NID	(unused), EHS, NQSW, DIR, MDIR, IVMS, TQSW, (unused)
6	DATE TIME USED (DTU)	
7	DATE TIME MODIFIED (DTM)	
8	SEGMENT QUOTA USAGE	DIRECTORY QUOTA COUNT
9	SEGMENT QUOTA USED	DIRECTORY QUOTA USED
10	CURRENT LENGTH, FMC, FMS, NPFS, (unused), DNZP, PERP, DDNP, (unused)	RECORDS USED, NO. OF PAGES
11	HASH TABLE R_PTR	FMCI, DAM, POVF, PTSI, (unused), PTSI, MARKER

(WORDS)

ACTIVE SEGMENT TABLE ENTRY (ASTE)

(aste.incl.pl1)

A WIRED (SST) DATA BASE – ONE PER ACTIVE SEGMENT

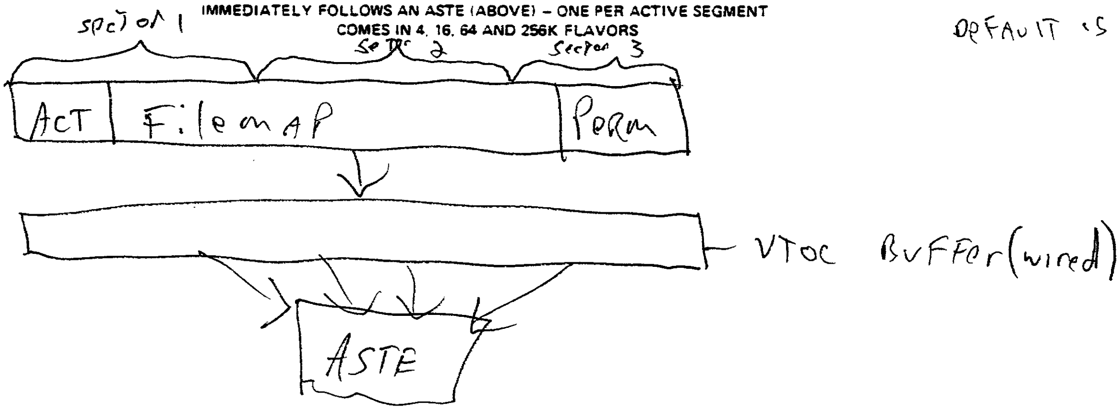
Word	Bits 0–17	Bits 18–35
0	ADDRESS	ADD_TYPE, FRST, PROC, (unused), UNUS, PHU, (unused), PHM, PHUI, WIRE, OS, VALID, DFNO
⋮		
13	MAIN MEMORY ADDRESS, 0000	1000, :
14	0 NULL ADDRESS – REPRESENTS PAGE OF ZEROES	0000, :

(WORDS) (PTW.INCL.PL1)

A 16K PAGE TABLE (PT)

IMMEDIATELY FOLLOWS AN ASTE (ABOVE) – ONE PER ACTIVE SEGMENT
COMES IN 4, 16, 64 AND 256K FLAVORS

VToc BuFFers
PARm VTB 64
DEFAULT is higher



SEGMENT CONTROL DATA BASES

ACTIVE SEGMENT TABLE (AST)

- THE ACTIVE SEGMENT TABLE (AST) IS A HARDCORE, WIRED, UNPAGED, DATA BASE LOCATED WITHIN THE SYSTEM SEGMENT TABLE (SST), AND CONSISTS OF AN ARRAY OF PAIRED ENTRIES KNOWN AS ACTIVE SEGMENT TABLE ENTRIES (ASTE'S) AND PAGE TABLES (PT'S)

  - ONE AST PER SYSTEM

  - ONE ASTE/PT PAIR PER ACTIVE SEGMENT

- IN ORDER FOR A SEGMENT TO BE ACCESSED VIA THE HARDWARE, VTOCE INFORMATION MUST BE BROUGHT INTO MAIN MEMORY

- THE 12 WORD ASTE (AND ITS ASSOCIATED PAGE TABLE) CAN BE THOUGHT OF AS THE MAIN MEMORY RESIDENT IMAGE OF THE VTOCE

  - SPECIFICALLY, THE ASTE CONTAINS:

    - THE VTOCE'S "ACTIVATION INFORMATION" SUCH AS THE SEGMENTS UID, CURRENT LENGTH, MAX LENGTH, DTU, DTM, QUOTA DATA

    - AND NON-VTOCE INFORMATION SUCH AS: PVT INDEX, VTOC INDEX, VARIOUS FLAGS AND POINTERS

  - THE PAGE TABLE CONTAINS THE RECORD ADDRESS (TAKEN FROM THE VTOCE'S "FILE MAP") OF EACH NONZERO PAGE OF THE SEGMENT

SEGMENT CONTROL DATA BASES

ACTIVE SEGMENT TABLE (AST)

- A SEGMENT HAVING AN ASTE AND A PAGE TABLE IS SAID TO BE ACTIVE

  - "ACTIVATING" A SEGMENT IS THE PROCESS OF ALLOCATING (AND FILLING IN) AN ASTE AND A PAGE TABLE FOR THE SEGMENT

  - CONVERSELY, "DEACTIVATING" A SEGMENT INVOLVES FREEING ITS ASTE AND PAGE TABLE FOR FURTHER USE

  - BEING ACTIVE DOES NOT IMPLY THAT THE SEGMENT IS ACTUALLY IN USE BY ANY PROCESS

- SINCE THE AST IS A PART OF A SINGLE SEGMENT (HAVING FINITE SIZE), THE NUMBER OF ASTE/PT PAIRS IS FINITE, IMPLYING:

  - ONLY A FINITE NUMBER OF SEGMENTS MAY BE ACTIVE AT ONE TIME

  - WHEN A NON-ACTIVE SEGMENT IS REFERENCED, AND THERE ARE NO FREE ASTE'S AVAILABLE, SOME SEGMENT MUST BE DEACTIVATED

    - THIS ASTE/PT MULTIPLEXING IS THE PRIME RESPONSIBILITY OF SEGMENT CONTROL

  - BEING A FINITE (AND A CRITICAL) SYSTEM RESOURCE, THE NUMBER OF ASTE/PT PAIRS CAN DRAMATICALLY AFFECT THE COMPETITION FOR ASTE'S, AND CONSEQUENTLY SYSTEM PERFORMANCE

SEGMENT CONTROL DATA BASES

ACTIVE SEGMENT TABLE (AST)

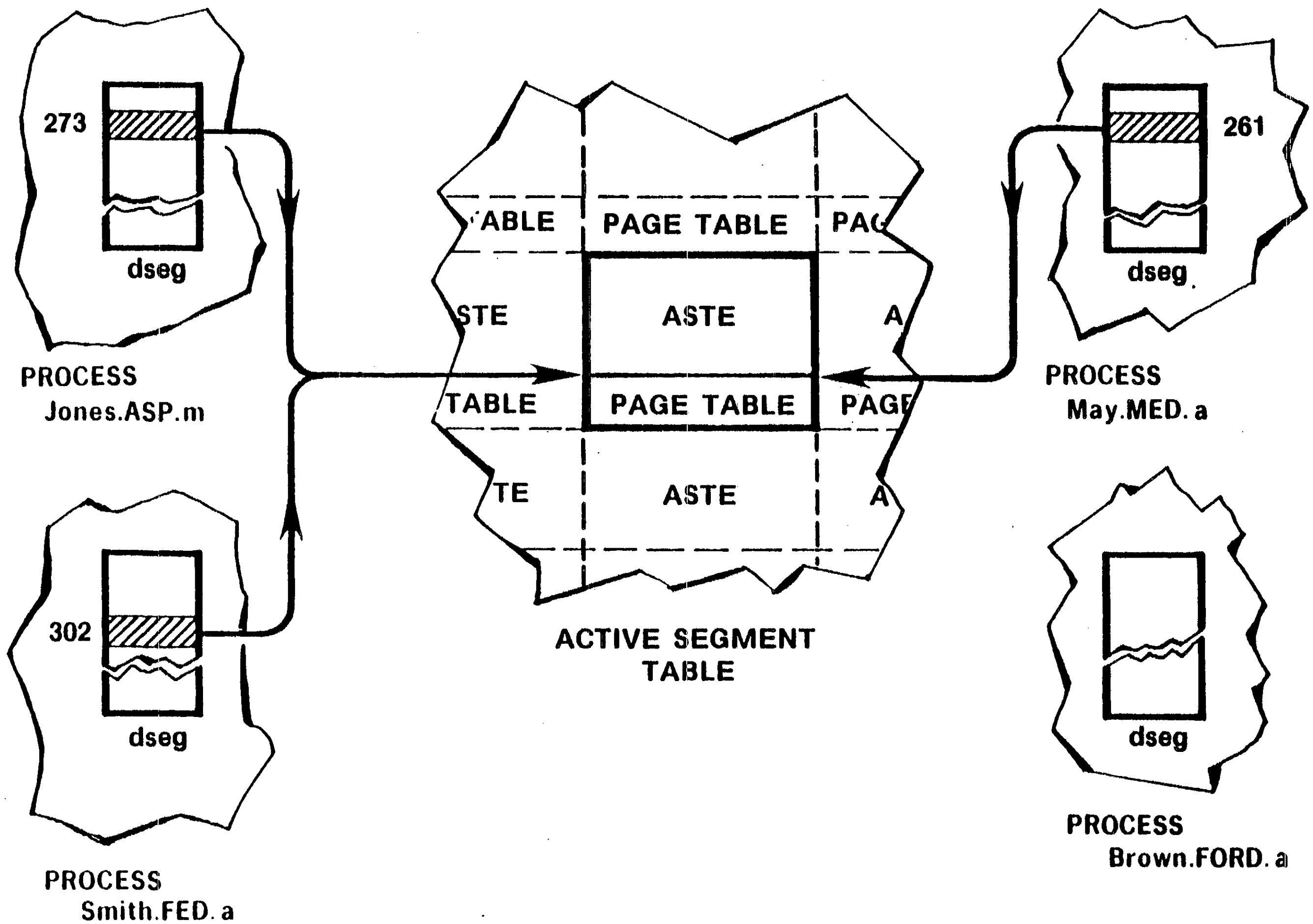
- TOO FEW ASTE/PT PAIRS WILL CAUSE "SEGMENT THRASHING"

- TOO MANY ASTE/PT PAIRS WILL OCCUPY MAIN MEMORY FRAMES THAT MIGHT BETTER BE UTILIZED FOR NORMAL PAGING TRAFFIC, PERHAPS LEADING TO "PAGE THRASHING"

- CONSEQUENTLY, THE NUMBER OF ASTE/PT PAIRS IS A CRITICAL SYSTEM PARAMETER (SET ON THE SST CONFIG CARD)

  - OF THE TWO POSSIBILITIES, TOO MANY OR TOO FEW, TOO FEW IS BY FAR THE WORSE

CONNECTING AND FAULTING SDW'S



SEGMENT CONTROL DATA BASES

ACTIVE SEGMENT TABLE (AST)

- IN ORDER TO MAXIMIZE THE NUMBER OF ASTE/PT PAIRS WITHIN AN AST OF A GIVEN SIZE, ASTE/PT PAIRS COME IN FOUR FIXED SIZES:

  - ASTE + A 4 WORD PAGE TABLE (16 WORDS TOTAL)

    - FOR 0-4K SEGMENTS

  - ASTE + A 16 WORD PAGE TABLE (28 WORDS TOTAL)

    - FOR 5-16K SEGMENTS

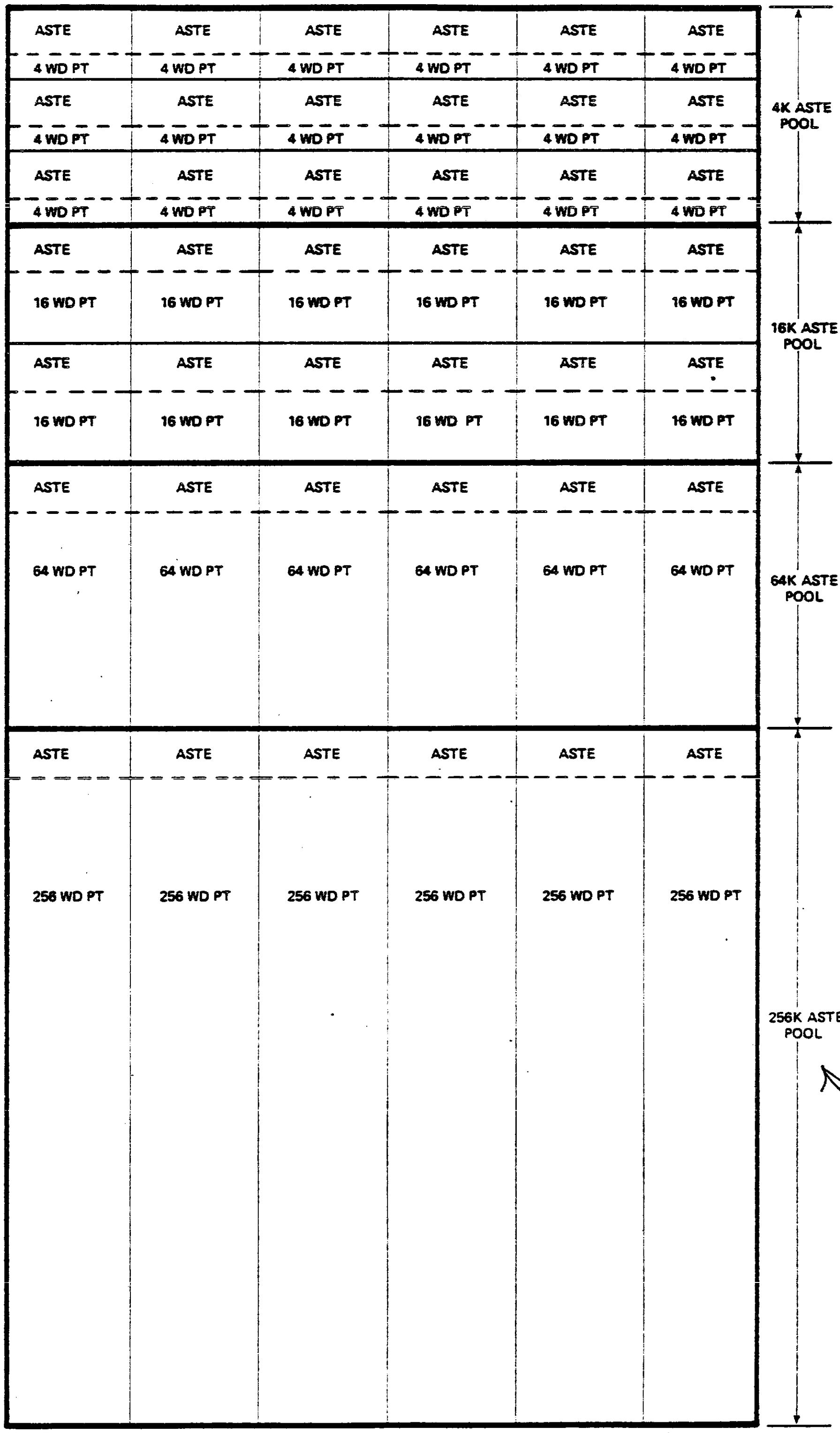
  - ASTE + A 64 WORD PAGE TABLE (76 WORDS TOTAL)

    - FOR 17-64K SEGMENTS

  - ASTE + A 256 WORD PAGE TABLE (268 WORDS TOTAL)

    - FOR 65-256K SEGMENTS

SEGMENT CONTROL DATA BASES



MRDS uses Alot of 256k Pools

ACTIVE SEGMENT TABLE (AST)

A HARDCORE (SST) UN-PAGED DATA BASE – ONE PER SYSTEM
(NOT DRAWN TO SCALE)

SEGMENT CONTROL DATA BASES

ACTIVE SEGMENT TABLE (AST)

- THE SIZE OF EACH OF THE FOUR ASTE POOLS IS DETERMINED AT SYSTEM INITIALIZATION BY THE SST CONFIG CARD AND IS A CRITICAL SYSTEM TUNING PARAMETER

- SINCE THE FREQUENCY OF SMALL SEGMENTS IS HIGHER THAN THE FREQUENCY OF LARGE SEGMENTS, THE DISTRIBUTION OF ASTE'S IS NORMALLY AS FOLLOWS:

  - 4K > 16K > 64K > 256K

  - ON SYSTEM-M, IN PHOENIX, A 6 CPU, 8MW MEMORY, 200 USER SYSTEM, THE ASTE DISTRIBUTION IS NORMALLY:

    3500 1500 750 250

    ON MIT-MULTICS, A 3 CPU, 3.5MW MEMORY, 110 USER SYSTEM, THE ASTE DISTRIBUTION IS NORMALLY:

    1700 600 220 75

- A SEGMENT NORMALLY REMAINS ACTIVE (FOR >200 SECONDS) UNTIL FORCED TO GIVE UP ITS ASTE/PT PAIR TO ANOTHER SEGMENT (DEACTIVATION)

ACTIVE SEGMENT TABLE (AST)

- THIS DEACTIVATION CONSISTS OF:

  - MAKING THE SEGMENT INACCESSIBLE TO USER PROCESSES

    - DONE BY "CUTTING TRAILERS", IN THE PROGRAM setfaults.pl1

    - A LIST OF ALL SEGMENTS CONNECTED TO (USING) THE SEGMENT IS KEPT FOR THIS PURPOSE

  - EVICTING ALL PAGES OF THE SEGMENT FROM MAIN MEMORY

    - ONLY MODIFIED PAGES MUST BE WRITTEN BACK TO DISK. UNMODIFIED PAGES ARE SIMPLY OVERWRITTEN

  - UPDATING THE VTOCE BY WRITING THE (POSSIBLY MODIFIED) ACTIVATION INFORMATION BACK TO THE VTOCE

  - FREEING THE ASTE/PT PAIR

- SINCE ACTIVATING/DEACTIVATING SEGMENTS IS EXPENSIVE, THE CHOICE OF A SEGMENT FOR DEACTIVATION IS IMPORTANT, AND BELONGS TO THE SEGMENT REQUIRING ACTIVATION FURTHEST IN THE FUTURE

SEGMENT CONTROL DATA BASES

ACTIVE SEGMENT TABLE (AST)

- THE ALGORITHM WHICH CHOOSES A "BEST" SEGMENT FOR DEACTIVATION IS IMPLEMENTED IN THE PROGRAM get_aste, AND CONSIDERS SUCH FACTORS AS:

  - THE PRESENCE OF ACTIVE INFERIORS (IF A DIRECTORY)

  - THE NUMBER OF PAGES CURRENTLY IN MAIN MEMORY (SINCE WORK IS REQUIRED TO EVICT THE MODIFIED FRACTION OF SUCH PAGES AND BECAUSE THIS INDICATES "USED RECENTLY" IN SOME SENSE)

  - IT LOOKS FIRST FOR A SEGMENT WITH NO INFERIORS, AND NO PAGES IN MEMORY, AND ALMOST ALWAYS SUCCEEDS - BUT IF IT FAILS, IT TRIES TO MAKE A "GOOD CHOICE", AND DEACTIVATES INFERIORS, AND/OR FLUSHES PAGES TO DISK IF NECESSARY

A aste.used

B aste.ehs entry-hold-switch

aste.sons each parent dir of active segment is active

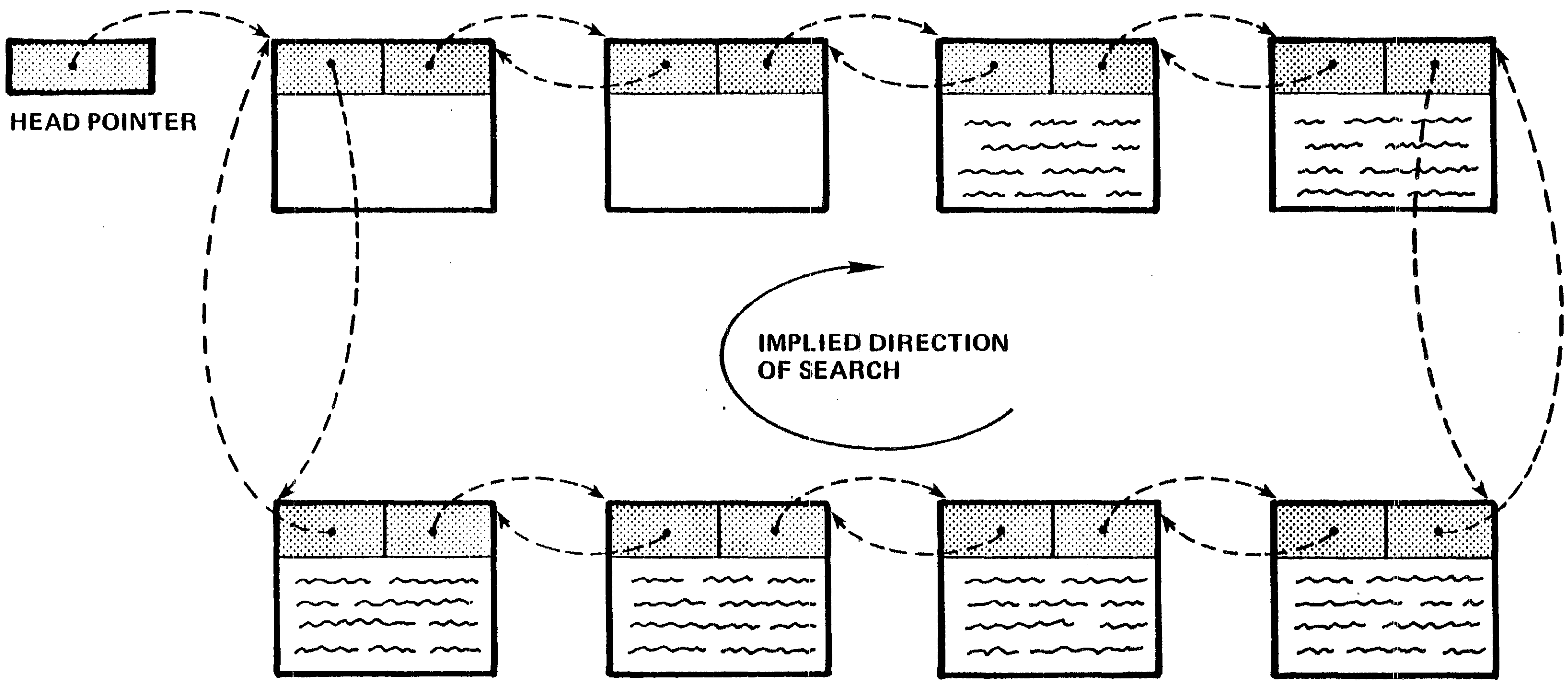
Set by page control

aste.np number pages in main memory

IF np = 0 Then seg not in use, so find out if "recently used"

aste.init IF 0 not recently used so we can deactivate it.

first turns off init bits so that on 3rd pass we can deactivate segs in case 2nd pass didn't find any not in use.



TYPICAL "USED LIST"

A DOUBLE THREADED LIST OF SIMILAR OBJECTS GENERALLY IMPLEMENTING REPLACEMENT ALGORITHMS. CONTAINS BOTH FREE AND IN–USE OBJECTS, WITH FREE OBJECTS MAINTAINED AT THE HEAD OF THE LIST.

ACTIVE SEGMENT TABLE (AST)

- NOTE: WHILE INSPECTING THE ASTE'S, OPPORTUNITY IS TAKEN TO NOTICE ASTE'S WHOSE FILE MAPS HAVE CHANGED AND TO UPDATE THEIR VTOCE'S

  - KNOWN AS "AST TRICKLE"

  - THIS IS DONE TOTALLY AS A HEDGE AGAINST A FATAL CRASH, AS A SUCCESSFUL SHUTDOWN UPDATES ALL VTOCE'S OF ACTIVE SEGMENTS

  - THIS IS NOT DONE FOR PROCESS DIRECTORY SEGMENTS, SINCE THEIR CONTENTS ARE OF LITTLE USE AFTER A CRASH

  - THE "AST TRICKLE" IS ALSO FORCED TO OCCUR EVERY FIFTEEN MINUTES WHEN THE SYSTEM IS LIGHTLY LOADED, BECAUSE OTHERWISE VTOCES MIGHT REMAIN UNUPDATED FOR HOURS

SEGMENT CONTROL DATA BASES

ACTIVE SEGMENT TABLE (AST)

- AST HIERARCHY

  - MIRRORS A SUBSET OF THE STORAGE SYSTEM HIERARCHY

  - THE ROOT DIRECTORY (>) CANNOT BE DEACTIVATED

  - NO SEGMENT (EXCEPT THE ROOT) MAY BE ACTIVE UNLESS ITS PARENT IS ACTIVE. THE REASONS FOR THIS ARE:

    - PARENT MUST BE ACTIVATED IN ORDER TO FIND THE SON

    - ACTIVATION OF OTHER SONS IS EASIER IF THE PARENT REMAINS ACTIVE

    - THE QUOTA ACCOUNT AGAINST WHICH AN ACTIVE SEGMENT'S "RECORDS USED" IS TALLIED SHOULD BE IMMEDIATELY AVAILABLE WHEN A SEGMENT CHANGES SIZE. THE QUOTA ACCOUNT IS FOUND IN ONE OF THE ANCESTORS' ASTES

    - DATE TIME MODIFIED (DTM) FOR A DIRECTORY IS THE DTM OF THE LAST MODIFIED SEGMENT IN THE SUBTREE. DTM OF ALL ANCESTOR'S SHOULD BE IMMEDIATELY AVAILABLE WHEN A SEGMENT IS MODIFIED. DTM IS FOUND IN THE ASTE OF THE ANCESTORS. (USED BY THE HIERARCHY DUMPER)

  - SUCH UPDATES TO THE ASTE'S OF PARENTS ARE PERFORMED BY PAGE CONTROL

  - EACH ASTE HAS A POINTER TO ITS PARENT'S ASTE FOR THE ABOVE REASONS. (THIS POINTER IMPLEMENTS THE AST HIERARCHY)

SEGMENT CONTROL DATA BASES

ACTIVE SEGMENT TABLE (AST)

- ASTE'S MAY BE THREADED ONTO ONE OF SIX LISTS VIA THE RELATIVE POINTERS aste.fp and aste.bp

  - FOUR USED LISTS: THREADS ALL FREE AND REPLACEABLE ASTE'S OF EACH POOL SIZE (sst.level.ausedp POINTS TO THE FIRST FREE ASTE IN THE LIST)

  - INIT AND TEMP LISTS: USED AT SYSTEM INITIALIZATION TO RECEIVE (AND DELETE) INITIALIZATION AND TEMPORARY SEGMENTS

- THERE ALSO EXIST SEVERAL AUXILIARY LISTS SUCH AS THE HASH THREAD AND FATHER-SON, AND BROTHERS LISTS

- ALL ACTIVE SEGMENTS IN THE HIERARCHY ARE IN THE FOUR USED LISTS - EXCEPT FOR SEGMENTS IN THE HARDCORE PARTITION (THE PAGED SUPERVISOR), AND A SMALL CLASS OF SEGMENTS WHICH MAY NOT BE DEACTIVATED

  - SEGMENTS ARE SOMETIMES UNTHREADED FROM THEIR USED LIST TEMPORARILY IN ORDER TO KEEP THEM OUT OF REACH WHILE SOME COMPLEX OPERATION IS PERFORMED

CREATING SEGMENTS

- SEGMENT CREATION IS PERFORMED BY THE PROCEDURE create_vtoce

  - INPUT: A POINTER TO THE BRANCH ENTRY IN A DIRECTORY SEGMENT

  - OUTPUT: PVID AND VTOC INDEX OF THE CREATED SEGMENT

- create_vtoce MAY BE CALLED BY append (NORMAL SEGMENT CREATION) OR segment_mover (DUE TO PACK OVERFLOW)

- PRINCIPAL STEPS OF create_vtoce:

  - CREATE A LOCAL IMAGE OF THE VTOCE TO BE CREATED

    - FILL IN MOST ACTIVATION AND PERMANENT INFORMATION FROM THE BRANCH ENTRY

    - CREATE A NULL FILE MAP

    - DETERMINE THE UID PATH (UID'S OF SUPERIOR DIRECTORIES)

CREATING SEGMENTS

- SELECT AN APPROPRIATE PV WITHIN THE LV SPECIFIED BY THE sons_lvid OF THE DIRECTORY IN WHICH THE BRANCH ENTRY APPEARS

  - SELECTION GOAL IS TO EVENLY DISTRIBUTE SEGMENTS OVER ALL PV'S OF THE LV, THEREBY REDUCING DISK CONTENTION

  - SELECTION ALGORITHM WALKS THE CHAIN (THROUGH pvte.brother_pvtx) OF PV'S IN THE LV AND SELECTS THE PV HAVING THE HIGHEST PERCENTAGE OF UNUSED RECORDS IN ITS PAGING REGION

  - NO PV IS ACCEPTED IF pvte.vacating IS ON, SIGNIFYING THAT sweep_pv IS TRYING TO VACATE, OR INHIBIT CREATION UPON, THE PV

  - AN EXCEPTION IS MADE FOR PER PROCESS SEGMENTS (entry.per_process IS ON)

    - SINCE SUCH SEGMENTS ARE ALL HEAVILY USED, A ROUND ROBIN ALGORITHM EVENLY DISTRIBUTES THESE SEGMENTS ACROSS ALL PV'S IN THE LV

- INVOKE THE VTOC MANAGER (vtoc_man$alloc_and_put_vtoce) TO ALLOCATE AND WRITE THE VTOCE IMAGE ON THE SELECTED PV

  - VTOC_MAN ATTEMPTS TO ALLOCATE A VTOCE FROM THE VTOCE STOCK FOR THE VOLUME

    - IF THE STOCK IS EMPTY, IT REFILLS IT FROM THE VTOCE MAP ON DISK (SEE vtoce_stock_man.pl1)

    - BECAUSE IT IS PERMISSIBLE TO TAKE PAGE FAULTS IN THE VTOC_MAN ENVIRONMENT, THE VTOCE STOCK IS ACCESSED WITHOUT ANY SPECIAL PAGE CONTROL PROTOCOLS

  - RETURNS THE VTOC INDEX OF THE ALLOCATED VTOCE

CREATING SEGMENTS

- RETURN THE PVID AND VTOC INDEX OF THE NEW SEGMENT TO THE CALLER (WHO RECORDS SAME IN entry.pvid AND entry.vtocx)

SEGMENT ACTIVATION

- SEGMENT ACTIVATION IS PERFORMED BY THE PROCEDURE "activate"

  - INPUT: A POINTER TO THE BRANCH ENTRY IN A DIRECTORY SEGMENT

  - OUTPUT: AN ASTE POINTER

- activate IS PRINCIPALLY CALLED BY seg_fault (OF ADDRESS/NAME SPACE MANAGEMENT) WHO HAS LOCATED THE SEGMENT'S BRANCH ENTRY, VALIDATED THE USER'S ACCESS, AND CHECKED THE PRESENCE OF THE LV

- PRINCIPAL STEPS OF activate:

  - LOCK THE AST LOCK AND CHECK IF THE SEGMENT IS ALREADY ACTIVE. IF SO, UNLOCK THE AST AND RETURN ITS ASTE POINTER

  - IF THE SEGMENT IS NOT ACTIVE, UNLOCK THE AST AND READ IN ALL REQUIRED PARTS OF THE VTOCE AND COMPARE UID'S FOR CONNECTION FAILURE (IN WHICH CASE, DO NOT ACTIVATE AND RETURN AN ERROR)

  - ENSURE THAT THE SEGMENT'S PARENT IS ACTIVE

    - THIS IS DONE BY REFERENCING THE PARENT DIRECTORY (PERHAPS CAUSING A RECURSIVE SEGMENT FAULT AND ACTIVATION) AND SETTING THE aste.ehs BIT FOR IT TEMPORARILY WHILE ACTIVATION IS TAKING PLACE

SERVICES OF SEGMENT CONTROL

SEGMENT ACTIVATION

   - NOTE THAT THE ONLY EXPLICIT ACTION TAKEN TO ACTIVATE THE PARENT IS TO REFERENCE IT; THE POSSIBLE RECURSIVE SEGMENT FAULT TAKES CARE OF EVERYTHING AND ALLOWS THAT REFERENCE TO PROCEED

- OBTAIN AN ASTE FOR THE SEGMENT BY CALLING get_aste. THIS MAY INVOLVE DEACTIVATING SOME OTHER SEGMENT - BUT HOPEFULLY NOT THE PARENT! (ENSURED BY THE aste.ehs BEING ON)

- THREAD THE ASTE INTO THE INFERIOR LIST OF THE PARENT'S ASTE (THIS WILL KEEP HIM ACTIVE), AND RESET THE PARENT'S aste.ehs

- FILL IN THE ASTE WITH THE VTOCE'S ACTIVATION INFORMATION AND INITIAL FLAG VALUES

- CALL pc$fill_page_table (PASSING THE VTOCE'S FILE MAP) TO INITIALIZE THE PAGE TABLE AND OTHER PAGE CONTROL INFORMATION

- PLACE THE UID IN THE ASTE AND HASH IT INTO THE AST HASH TABLE

   - AFTER A SYSTEM FAILURE, ESD USES A ZERO UID AS A CUE THAT THE ASTE IS INVALID, AND DOES NOT INVOKE A VTOCE UPDATE FOR THE ASTE

SERVICES OF SEGMENT CONTROL

SEGMENT TRAILERS

- WHEN A PROCESS IS USING A SEGMENT, AND IT HAS A VALID SDW FOR THAT SEGMENT, A RECORD MUST BE KEPT IN CASE IT IS NECESSARY TO REVOKE THAT SDW (WHEN THE SEGMENT IS DELETED, WHEN THE ASTE IS RE-USED, ETC)

  - THIS IS DONE BY THE SEGMENT TRAILER MECHANISM

    - EACH PROCESS WHICH HAS A VALID SDW FOR A SEGMENT HAS A "TRAILER ENTRY" WHICH RECORDS ITS PROCESS IDENTIFICATION AND THE SEGMENT NUMBER IT IS USING FOR THE SEGMENT → NORM
      config ⇒ Parm STR 64. → default 16 pages
      or limit of lastpool 256 maybe

    - TRAILER ENTRIES ARE KEPT IN THE str_seg, A PAGED SUPERVISOR SEGMENT

    - THERE IS A LINKED LIST OF TRAILER ENTRIES FOR EACH SEGMENT; THE HEAD IS POINTED TO BY aste.strp

  - TRAILERS ARE ONLY KEPT FOR SEGMENTS WHICH MAY BE DEACTIVATED: ORDINARY SEGMENTS AND DIRECTORIES, BUT NOT SUPERVISOR SEGMENTS

  - A TRAILER IS ATTACHED FOR A SEGMENT BY seg_fault.pl1 BEFORE IT PLACES THE SDW INTO THE DSEG FOR THE PROCESS

  - TRAILERS ARE USED LATER BY setfaults.pl1 FOR:

    - DEACTIVATION - THE SDW IS ENTIRELY REVOKED (ZEROED)

    - ACCESS CHANGES, DELETION - THE SDW IS MADE INVALID, BUT ITS PAGE TABLE ADDRESS IS LEFT UNCHANGED, INDICATING TO seg_fault.pl1 THAT ALL IT MUST DO IS RECALCULATE THE ACCESS

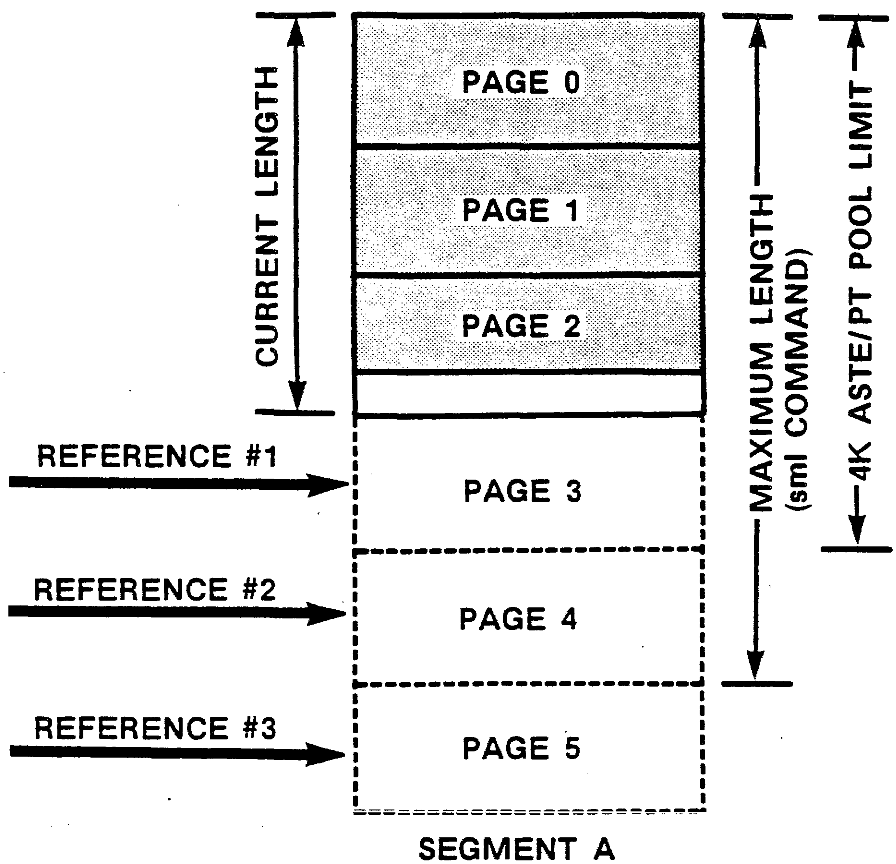
SERVICES OF SEGMENT CONTROL

SEGMENT TRAILERS

- PROCESS TERMINATION - ALL SDWS A PROCESS HAD ARE REVOKED AT TERMINATION

- CACHE CONTROL - WHEN THE ENCACHABILITY OF A SEGMENT CHANGES, THE SDWS WHICH REFER TO IT MUST HAVE THEIR CACHE CONTROL BITS UPDATED

- TRAILERS ARE ONLY REMOVED WHEN THE ASSOCIATED SDW IS REVOKED COMPLETELY



BOUNDS FAULT HANDLING

1. PAGE FAULT. PAGE 3 "CREATED" IN MAIN MEMORY AS A PAGE OF ZEROS (CALLED A "NEW PAGE")

2. BOUNDS FAULT. PROMOTE THE SEGMENT TO THE 16K ASTE/PT POOL..CONTINUE REFERENCE AS IN CASE #1

3. BOUNDS FAULT. ERROR: SIGNAL "out-of-bounds" CONDITION

SERVICES OF SEGMENT CONTROL

BOUNDSFAULT HANDLING

- THE BOUNDSFAULT HANDLER IS THE PROCEDURE "boundsfault", INVOKED BY THE FAULT INTERCEPTER MODULE, FIM, WHEN THE BOUNDSFAULT IS DETECTED BY THE APPENDING UNIT HARDWARE

- BASIC STEPS OF BOUNDSFAULT

  - USING THE SEGMENT NUMBER IN THE (SAVED) MACHINE CONDITIONS, FIND AND LOCK THE PARENT DIRECTORY, AND FIND THE BRANCH ENTRY

  - LOCK THE AST AND FIND THE SEGMENT'S ASTE VIA get_ptrs_$given_segno. IF ATTEMPTED REFERENCE IS BEYOND THE MAXIMUM LENGTH (aste.msl) THEN CAUSE "out_of_bounds" TO BE SIGNALLED

  - MAKE THE SEGMENT INACCESSIBLE TO USERS BY "CUTTING TRAILERS"

  - TURN ON THE PARENT'S aste.ehs BIT AND CALL get_aste TO OBTAIN A LARGER ASTE

  - CALL PAGE CONTROL'S pc$move_page_table TO MOVE ALL ASTE/PT INFORMATION TO THE NEW ASTE

  - RETHREAD ALL INFERIOR LIST AND PARENT POINTERS AFFECTED AND TURN OFF THE PARENT'S aste.ehs BIT

    - NOTE: IF THE SEGMENT IS A DIRECTORY, ALL FATHER POINTERS OF INFERIOR SEGMENTS MUST BE UPDATED

BOUNDSFAULT HANDLING

- THIS IS THE ONLY REASON FOR THE EXISTENCE OF INFERIOR LIST IN THE AST

- REMOVE THE OLD ASTE FROM THE AST HASH TABLE AND HASH IN THE NEW

- CALL put_aste TO FREE THE OLD ASTE

- UNLOCK THE AST AND RETURN A ZERO STATUS CODE TO THE FIM

SEGMENT DEACTIVATION

- SEGMENT DEACTIVATION IS PERFORMED BY THE PROCEDURE "deactivate"

  - INPUT: POINTER TO AN ASTE

- deactivate MAY BE CALLED BY:

  - get_aste WHEN AN ASTE MUST BE FREED TO MAKE ROOM FOR A NEW SEGMENT

  - delete_vtoce AS PART OF SEGMENT DELETING

  - demount_pv IN ORDER TO UPDATE THE VTOCE'S (AND SEGMENTS) OF A DISK BEING DEMOUNTED

- PRINCIPAL STEPS OF deactivate:

  - MAKE THE SEGMENT INACCESSIBLE TO USERS BY "CUTTING TRAILERS"

  - CALL pc$cleanup TO REMOVE ALL PAGES OF THE SEGMENT FROM BULK STORE AND MAIN MEMORY, WRITING ALL MODIFIED PAGES TO DISK

SERVICES OF SEGMENT CONTROL

SEGMENT DEACTIVATION

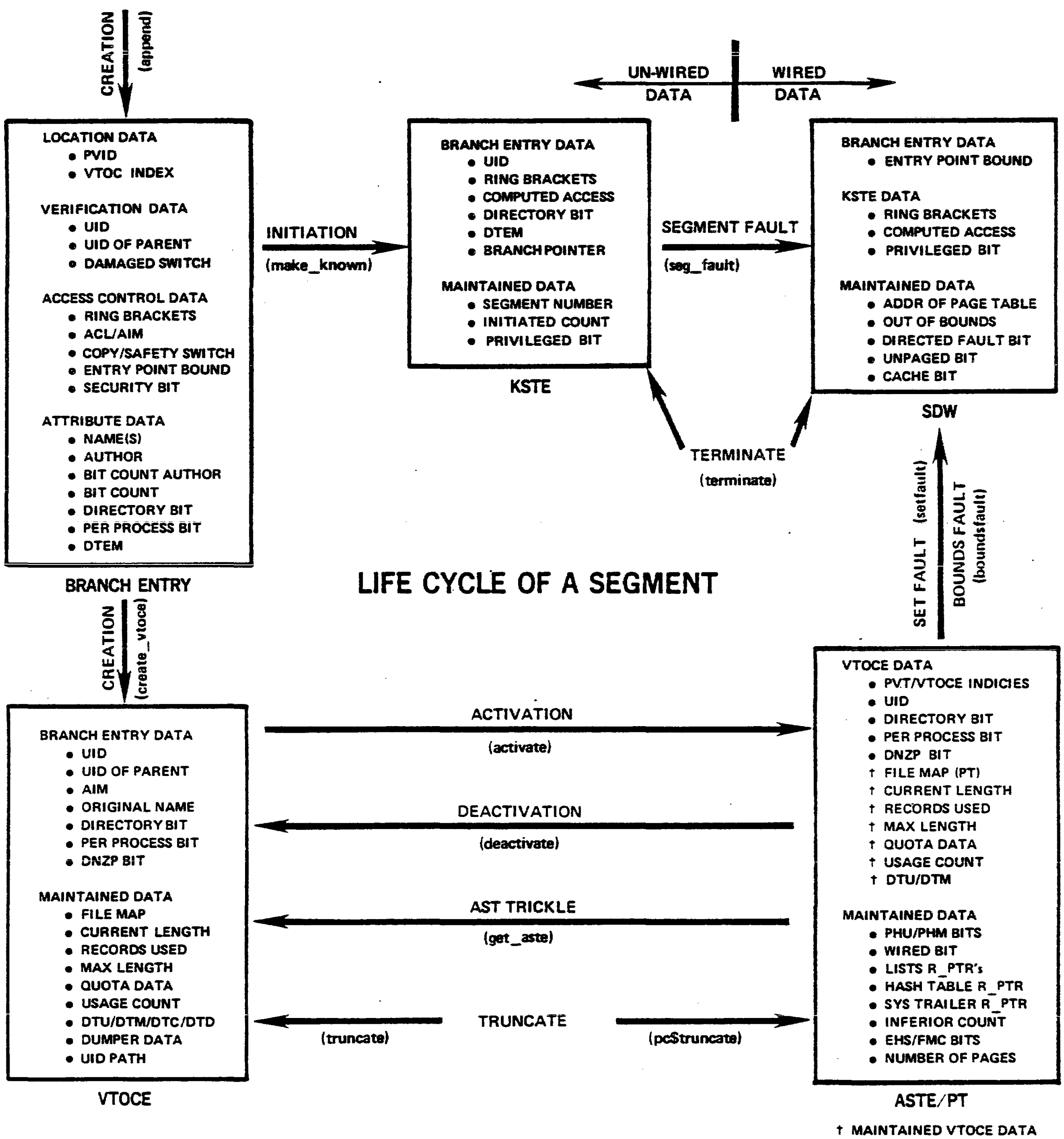
- UPDATE THE VTOCE FROM THE NOW QUIESCENT ASTE

- THREAD THE ASTE OUT OF PARENT'S INFERIOR LIST, AND OUT OF THE AST HASH TABLE

- CALL put_aste TO CLEAR AND INITIALIZE THE ASTE/PT PAIR, AND THREAD THE ASTE AT THE HEAD OF THE APPROPRIATE USED LIST

SERVICES OF SEGMENT CONTROL

SUMMARY OF MAJOR SERVICES



TRUNCATING SEGMENTS

- SEGMENT TRUNCATION (IE: PAGE REMOVAL) IS PERFORMED BY THE PROCEDURE truncate_vtoce

  - INPUT: A POINTER TO THE BRANCH ENTRY IN A DIRECTORY SEGMENT, AND A PAGE NUMBER FROM WHICH TO START TRUNCATING

- truncate_vtoce IS CALLED BY:

  - truncate (DIRECTORY CONTROL) WHO HAS LOCATED THE SEGMENTS BRANCH ENTRY AND VALIDATED THE USER'S ACCESS

  - delete_vtoce (SEGMENT CONTROL) WHO REQUIRES TRUNCATION (FROM PAGE #0) PRIOR TO VTOCE DELETION

- PRINCIPAL STEPS OF truncate_vtoce:

  - IF SEGMENT IS ACTIVE, CALL pc$truncate WHO MARKS THE DEVICE ADDRESS OF ALL PAGES ABOVE THE PAGE NUMBER SPECIFIED AS "NULLED" ADDRESSES (DISCUSSED IN "PAGE CONTROL" TOPIC)

  - IF SEGMENT IS NOT ACTIVE:

    - READ IN ALL REQUIRED PARTS OF THE VTOCE AND COMPARE UID'S FOR CONNECTION FAILURE (RETURN AN ERROR IF SO)

TRUNCATING SEGMENTS

- CALL get_pvtx$hold_pvtx TO PREVENT A DEMOUNT (IF CALLED BY truncate)

- COPY ALL ADDRESSES OF PAGES TO BE TRUNCATED FROM THE FILE MAP AND REPLACE THEM WITH "NULL" ADDRESSES

- FABRICATE A NEW VTOCE AND WRITE THE VTOCE BACK BY CALLING vtoc_man$put_vtoce

- IF ANY REAL ADDRESSES WERE COPIED FROM THE FILE MAP, AWAIT THE SUCCESSFUL COMPLETION OF THE VTOCE WRITE BY CALLING vtoc_man$await_vtoce

- CALL pc$deposit_list TO DEPOSIT (FREE) THESE REAL RECORD ADDRESSES

- CALL get_pvtx$release_pvtx TO AGAIN PERMIT DEMOUNTING (IF CALLED BY truncate). THIS CREATES AN ENTRY IN THE PV HOLD TABLE (SEE TOPIC 6, VOLUME MANAGEMENT)

DELETING SEGMENTS

- SEGMENT DELETION IS PERFORMED BY THE PROCEDURE delete_vtoce

  - INPUT: A POINTER TO THE BRANCH ENTRY IN A DIRECTORY SEGMENT

- delete_vtoce IS CALLED BY delentry (OF DIRECTORY CONTROL FAME) WHO HAS LOCATED THE SEGMENT'S BRANCH ENTRY AND VALIDATED THE USER'S ACCESS

- PRINCIPAL STEPS OF delete_vtoce:

  - CALL get_pvtx$hold_pvtx TO PREVENT A VOLUME DEMOUNT IN THE MIDDLE OF THE DELETION

  - IF ACTIVE, MAKE THE SEGMENT INACCESSIBLE TO USERS (SEE "ADDRESS AND NAME SPACE MANAGEMENT", TOPIC 5)

  - TRUNCATE THE SEGMENT TO ZERO LENGTH (SEE "TRUNCATING SEGMENTS" IN THIS TOPIC), FREEING ALL DISK, BULK STORE, AND MAIN MEMORY PAGES OCCUPIED BY THE SEGMENT

  - IF THE SEGMENT IS A DIRECTORY SEGMENT HAVING A QUOTA ACCOUNT, CALL THE QUOTA MOVE PRIMITIVE (quotaw$mq) TO RELINQUISH THE QUOTA TO ITS SUPERIOR

DELETING SEGMENTS

- NOTE: DIRECTORY CONTROL IS RESPONSIBLE FOR DELETING ALL INFERIOR SEGMENTS BEFORE REQUESTING DELETION OF THE DIRECTORY - ENSURING A CONSISTENT HIERARCHY AND RECOVERY OF ALL INFERIOR QUOTA ACCOUNTS

- IF THE SEGMENT IS ACTIVE, DEACTIVATE IT, RELEASING ITS ASTE

- FREE THE VTOCE WITH A CALL TO vtoc_man$free_vtoce

- CALL get_pvtx$release_pvtx TO AGAIN PERMIT VOLUME DEMOUNTING

OTHER SERVICES

- OTHER SERVICES PERFORMED BY SEGMENT CONTROL INCLUDE:

  - SEGMENT MOVING

    - REQUIRED WHEN AN ATTEMPT IS MADE TO GROW A SEGMENT AND THERE IS NO MORE ROOM ON THE PHYSICAL VOLUME

    - THE ENTIRE SEGMENT MUST BE MOVED TO ANOTHER PV WITHIN THE SAME LV --- TRANSPARENT TO THE USER AND DIRECTORY CONTROL

    - THIS IS THE SINGLE MOST INVOLVED AND ESOTERIC SERVICE OF SEGMENT CONTROL

  - SEMI-PERMANENT ACTIVATION

    - ACTIVATING A SEGMENT INTO AN ASTE OF A GIVEN SIZE AND TURNING ON ITS aste.ehs (DONE BY grab_aste)

  - SERVICES FOR sweep_pv

    - LISTING THE VTOC OF A PACK (IE: REPORTING THE PATHNAMES OF ALL SEGMENTS OWNING VTOCE'S)

    - THE LOCATING AND DELETING OF ORPHAN VTOCE'S (VTOCE'S NOT DESCRIBED IN ANY BRANCH)

    - REBALANCING OR VACATING PACKS VIA DEMAND SEGMENT MOVING

SERVICES OF SEGMENT CONTROL

OTHER SERVICES

- SERVICES AT DEMOUNT/SHUTDOWN TIME

  - DEACTIVATION OF ALL SEGMENTS ON THE VOLUME BEING DEMOUNTED AND WRITING OUT THE LABEL, ETC

- SERVICES FOR ADDRESS/NAME SPACE MANAGEMENT (SEE TOPIC 4)

  - DESCRIPTOR SEGMENT (DSEG), PROCESS DATA SEGMENT (PDS) AND KNOWN SEGMENT TABLE (KST) MANAGEMENT

  - SEGMENT FAULT HANDLING (seg_fault), CREATION, ENTRY HOLDING

- FILE_SYSTEM_METERS - DISPLAYS MISCELLANEOUS METERING INFORMATION FOR THE FILE SYSTEM

  - ONLY PARTS RELEVANT TO SEGMENT CONTROL INCLUDED HERE; SEE TOPIC 8 (PAGE CONTROL) FOR THE REST

Total metering time 0:20:02

		#	ATB — AVG Time Between	
	Activations	1043	1.153 sec.	
	segfault	969	1.241 sec.	92.905% of all
	makeknown	74	16.251 sec.	7.095% of all
	directories	96	12.527 sec.	9.204% of all
	Deactivations	1056	1.139 sec.	
	Demand deactivate attempts	3	400.857 sec.	
	Seg Faults	5080	0.237 sec.	
	fault	4311	0.279 sec.	84.862% of Seg Faults
	call	769	1.564 sec.	15.138% of Seg Faults
	activations	969	1.241 sec.	19.075% of Seg Faults
	Bound Faults	220	5.466 sec.	
going to SDW marking it invalid	Setfaults	4484	268.191 msec.	
	access	42	28.633 sec.	0.937% of setfaults
	ASTE Trickle	139	8.652 sec.	
	Steps	4279	281.040 msec.	
	Skips	3016	0.399 sec.	70.484% of Steps
	ehs — entry hold switch	271	4.438 sec.	8.985% of Skips
	mem in main mem	1083	1.110 sec.	35.909% of Skips
	init	1662	0.724 sec.	55.106% of Skips
	Searches	0	0.000 sec.	
Getting Seg out of mem to deactivate it. equal to deactivations	Cleanups	1056	1.139 sec.	0.1 % of real time
	Force writes	3	400.857 sec.	
	pages written	3	400.857 sec.	
	Lock AST	18422	0.065 sec.	

SEGMENT CONTROL METERS

file system meters

Can Be disk bottle neck

	AVE/lock	%
AST locked	4.833 msec.	7.4
AST lock waiting	1.601 msec.	2.5

÷ By 100 should Be < .8 or 80

AST Sizes	4	16	64	256
Number	1701	601	221	74
Need	819	202	208	34
Steps	2341	645	1139	154
Ave Steps	2.9	3.2	5.5	4.5
Lap Time(sec)	873.8	1120.5	233.3	577.9

$\frac{2341}{819} \approx 2.9$

seconds to go around last.
Should Be > 200

Seg Fault – referencing SDW with Valid Bit OFF
Put PTW in SDW
or
look in AST + Activate it
from reading VTOCE

Sort command can cause segment thrashing for awhile.
Because it chooses 17 Page size, Putting it in 64k Pool.

Flaw in Segment replacing Algorythm
light load, lots of memory, All Active segs in memory, ~~Since~~ Hence it Thinks it is in use + has to go Around for Awhile until it has to Force it Out. Causes slow response Time on light load.

c VTOC_BUFFER_METERS - DISPLAYS VTOC BUFFER MANAGER ACTIVITY

Total metering time: 65:21:12

Routine	# calls	ATB (sec)	
get_vtoce	1346752	0.17	
put_vtoce	0	0.00	
alloc_and_put_vtoce	77378	3.04	
free_vtoce	75664	3.11	
await_vtoce	93370	2.52	
GET_BUFFERS	2732265	0.09	1656952 Hits (60.6% of calls)
WAIT	946437	0.25	946435 TC Waits (100.0% of calls)

Buffer Allocation

	#	ATB (sec)	
Steps	1354279	0.17	
Skips	278866	0.84	20.6% of steps
os	240348	0.98	86.2% of skips
hot	0	0.00	0.0% of skips
wait	38518	6.11	13.8% of skips

Disk I/Os

	#	ATB (sec)
Reads	836941	0.28
Writes	548573	0.43

Display-ASTE

- PRINT_ASTE_PTP - DISPLAYS INFORMATION FROM AN ASTE

```
ASTE for >udd>Multics>Sibert at 115244 in sst_seg
 077550100664 041540056140 165706076310 102401170050
 315015000063 640000044000 446736250032 446736250272
 003720000000 003164000110 006200006000 073770000102

uid = 102401170050, vtocx = 63 on pvtx 15 (root4 = dskd_13)
max len 205, 6 recs used, 0 in core, cur len 6 (decimal)
Used 03/18/83  0116.0 est Fri
Modified 03/18/83  0116.2 est Fri
Par astep = 76310, Son = 56140, brother = 41540
Trailer thread = 165706
Aste for a directory.
     Quota (S D) = (2000 0)
     QUsed (S D) = (1652 72)

Flags: usedf init seg-tqsw fms

PAGE       PT          DEVADD

   0   063540200041    63540
   1   063546200041    63546
   2   063547200041    63547
   3   063550200041    63550
   4   063567200041    63567
   5   063571200041    63571
   6   377020000001    null
====
  17   377020000001    null
```

* DUMP_VTOCE - READS A VTOCE FROM DISK AND DISPLAYS ITS CONTENTS

```
vtoce Sibert (Directory), vtocx 63 on pvtx 15 (root4)-03/18/83  0116.7 est Fri

Uid = 102401170050, msl/csl/rec = 205   6   6
     Quota (S D) = (2000 0)
     Quota used (S D) = (1423 44)
     Quota received (S D) = (2000 0)
Created     08/16/81  1204.0 est Sun
Dumped      03/18/83  0106.7 est Fri
Used        03/17/83  1737.8 est Thu
Modified    03/15/83  2138.3 est Tue

Switches:  fm_checksum_valid

Activation information:

   0 000000000000 102401170050 315006006000 446734564570
   4 446723404647 001000400000 063560063567 003720000000
  10 002617000054 003720000000 000000000000 015364402005
  14 000000000000 000652154327 446734634556 445673737671

File map:

  20 063540063546 063547063550 063567063571 777020777020
  24 777020777020 777020777020 777020777020 777020777020
  30 777776777776 777776777776 777776777776 777776777776
======
 214 777776777776 777776777776 777776777776 777776777776

Permanent information:

 220 000000000000 000000000000 000000000000 000000000000
======
 230 000000000000 000000000000 000000000000 446736227316
 234 126104000547 126104000525 126104000764 000000000000
 240 777777777777 033022237767 033023254650 000000000000
 244 000000000000 000000000000 000000000000 000000000000
======
 260 123151142145 162164040040 040040040040 040040040040
 264 040040040040 040040040040 040040040040 040040040040
 270 441200557477 135240026001 004370027106 000000000000
 274 000000000000 000000000000 000000000000 000000000000
```

TOPIC VIII

Page Control

Page

PAGE CONTROL OVERVIEW

- FUNCTION

  - PAGE CONTROL IS RESPONSIBLE FOR THE MANAGEMENT OF PHYSICAL MEMORY TO INCLUDE THE MULTIPLEXING OF MAIN MEMORY FRAMES, AND THE MANAGEMENT OF DISK STORAGE

  - ITS TASKS INCLUDE:

    - TRANSFERRING THE PAGES OF SEGMENTS BETWEEN THE MEMORY DEVICES, AND RECORDING THE LOCATION OF "THE" COPY OF THESE PAGES

    - REPORTING THE STATUS AND FILE MAPS OF SEGMENTS TO SEGMENT CONTROL

  - PAGE CONTROL IS LARGELY CODED IN MULTICS ASSEMBLER LANGUAGE (ALM)

  - PAGE CONTROL CAN BE INVOKED EITHER BY SUBROUTINE CALLS OR BY PAGE FAULTS

  - THERE ARE NO EXPLICIT USER INTERFACES TO PAGE CONTROL

PAGE CONTROL OVERVIEW

- BASIC PHILOSOPHY

  - OF ALL THE SEGMENTS ACTIVE AT A GIVEN TIME, ONLY A SMALL SUBSET OF THEIR TOTAL PAGES WILL BE REQUIRED FOR ACCESSING

  - PAGES WILL BE READ INTO MAIN MEMORY AS THEY ARE REQUIRED

  - THE READING OF A PAGE INTO MAIN MEMORY WILL (PROBABLY) REQUIRE THE EVICTION OF A PREVIOUSLY REQUIRED PAGE

  - THE CHOICE OF A PAGE FOR EVICTION WILL BE BASED ESSENTIALLY UPON A "LEAST RECENTLY USED" CRITERIA

  - AN EVICTED PAGE NEED BE WRITTEN BACK TO DISK <u>ONLY</u> IF IT WAS MODIFIED DURING ITS RESIDENCY IN MAIN MEMORY

- MAJOR DATA BASES

  - PHYSICAL VOLUME TABLE (PVT) - ONE PER SYSTEM. PROVIDED BY VOLUME MANAGEMENT

    - PHYSICAL VOLUME TABLE ENTRY (PVTE) - ONE PER DISK DRIVE CONFIGURED

    - EACH PVTE CONTAINS:

      - THE DEVICE ID (DISK DRIVE ID) AND THE ID OF THE PHYSICAL VOLUME (DISK PACK) CURRENTLY MOUNTED

PAGE CONTROL OVERVIEW

  - THE NUMBER OF RECORDS LEFT UNALLOCATED ON THE PHYSICAL VOLUME, POINTER TO THE RECORD STOCK, ETC

- RECORD STOCKS - ONE PER MOUNTED PHYSICAL VOLUME, PROVIDED BY VOLUME MANAGEMENT

  - CONTAINS AN IN-MEMORY CACHE OF THE IN-USE STATUS OF RECORDS ON THE VOLUME, FROM THE VOLUME MAP, USED WHEN ALLOCATING OR FREEING PAGES

  - ACCESSED BY A COMPLEX MECHANISM WHICH USES NORMAL PAGE I/O BUT HAS A PROTOCOL TO ENSURE SYNCHRONIZATION OF DISK CONTENTS AND RECORD STOCK CONTENTS

- SYSTEM SEGMENT TABLE (SST) - ONE PER SYSTEM. SHARED WITH SEGMENT CONTROL. CONTAINS THE FOLLOWING FIVE DATA BASES USED BY PAGE CONTROL:

  - SYSTEM SEGMENT TABLE (SST) HEADER - ONE PER SYSTEM

    - CONTAINS A LARGE NUMBER OF COUNTERS AND POINTERS VITAL TO THE MAINTENANCE AND METERING OF THE STORAGE SYSTEM

    - CONTAINS LOCKWORDS USED TO SYNCHRONIZE PAGE CONTROL AND SEGMENT CONTROL OPERATIONS

  - CORE MAP - THE core_map SEGMENT - ONE PER SYSTEM

    - CORE MAP ENTRY (CME) - ONE PER FRAME (1024 WORDS) OF CONFIGURED MAIN MEMORY

    - EACH CME REPRESENTS A FRAME OF MAIN MEMORY AND IDENTIFIES THE CURRENT OCCUPANT OF THAT FRAME

    -

PAGE CONTROL OVERVIEW

NOT PART OF THE SST SEGMENT ANY MORE, BUT LOGICALLY PART OF THE SST

- ACTIVE SEGMENT TABLE (AST) - ONE PER SYSTEM
  - ACTIVE SEGMENT TABLE ENTRY (ASTE) - ONE PER ACTIVE SEGMENT
  - LIST OF ACTIVE (CURRENTLY BEING USED) SEGMENTS
- PAGE TABLES (PT) - ONE PER ACTIVE SEGMENT, THE OTHER HALF OF EACH ASTE
  - PAGE TABLE WORD (PAGE PTW) - EITHER 4, 16, 64, OR 256 PER PAGE TABLE
  - EACH PTW DEFINES THE CURRENT LOCATION OF A PAGE OF THE SEGMENT: DISK, MAIN MEMORY ADDRESS, OR NULL

PAGE CONTROL TERMINOLOGY

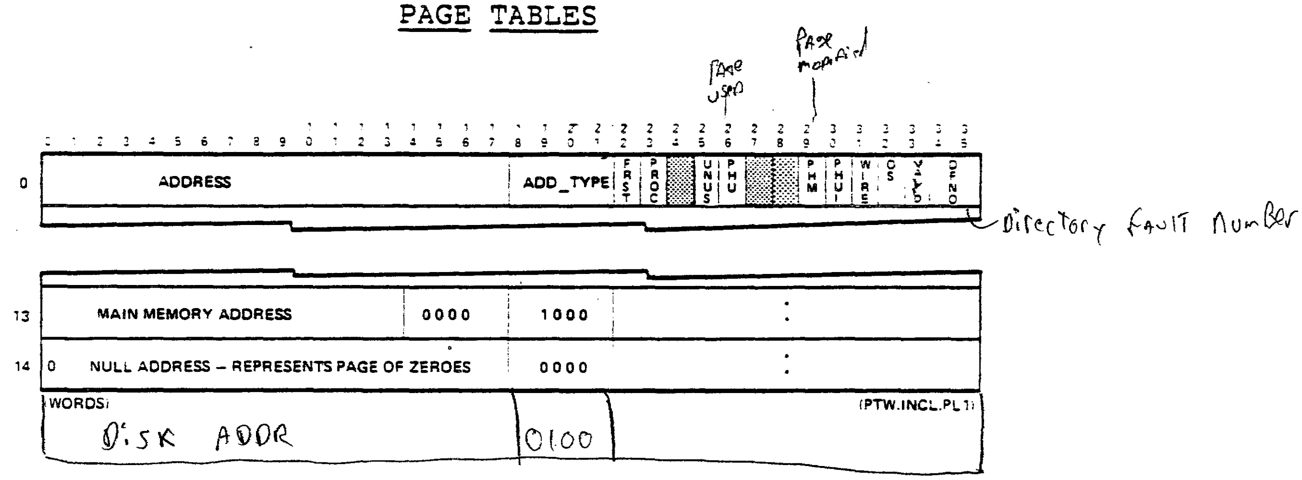
PAGING: THE PROCESS OF TRANSFERRING PAGES OF DATA BETWEEN DISK STORAGE AND MAIN MEMORY (CORE) TO ACHIEVE THE EFFECT OF ALL DATA BEING IN MEMORY ALL THE TIME

CORE: AN OBSOLETE TERM USED FREQUENTLY TO REFER TO MAIN MEMORY (WHICH IS MOS TECHNOLOGY, NOT CORE TECHNOLOGY)

PAGE FAULT: AN EXCEPTION CONDITION DETECTED BY THE PROCESSOR HARDWARE (IN THE APPENDING UNIT) WHEN AN ATTEMPT IS MADE TO USE A PTW SPECIFYING THAT ITS PAGE IS NOT IN MAIN MEMORY

PAGE CONTROL DATA BASES

PAGE TABLES



A 16K PAGE TABLE (PT)

IMMEDIATELY FOLLOWS AN ASTE (ABOVE) – ONE PER ACTIVE SEGMENT
COMES IN 4, 16, 64 AND 256K FLAVORS

- THE PAGE TABLES (PT'S) ARE HARDCORE (SST), UNPAGED, DATA BASES EACH CONSISTING OF AN ARRAY OF PAGE TABLE WORDS (PTW'S)

  - ONE PAGE TABLE PER ACTIVE SEGMENT

  - 4, 16, 64, OR 256 PTW PER PAGE TABLE

PAGE CONTROL DATA BASES

PAGE TABLES

- ALL PAGE TABLES ARE ASSOCIATED WITH, AND IMMEDIATELY FOLLOW AN ASTE IN THE AST REGION OF THE SST

- EACH PTW DESCRIBES THE STATUS OF ONE PAGE OF THE SEGMENT CURRENTLY IN POSSESSION OF THE ASSOCIATED ASTE, INCLUDING:

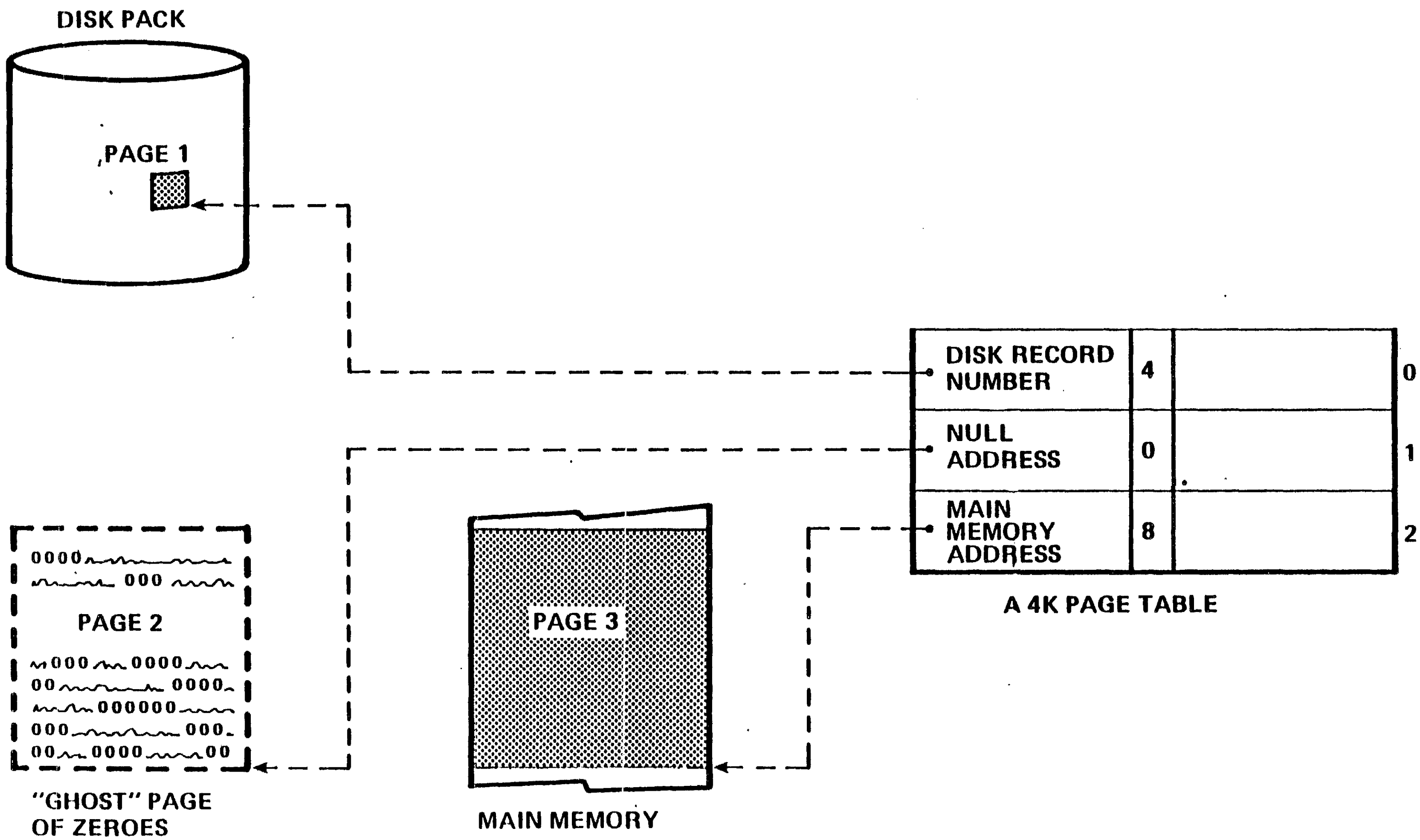
  - THE DEVICE ADDRESS OF THE COPY OF THE PAGE

  - PTW VALID INDICATOR AND FAULT NUMBER (FAULT #1)

  - FLAGS INDICATING VARIOUS STATES AND PROPERTIES OF THE PAGE SUCH AS I/O IN PROGRESS, WIRED, USED, MODIFIED

- THE ADDRESS PORTION OF EACH PTW IS INITIALIZED FROM THE SEGMENT'S VTOCE FILE MAP AT SEGMENT ACTIVATION TIME

PAGE TABLE DEVICE ADDRESSES



F80A

PAGE CONTROL DATA BASES

CORE MAP

(WORDS)	Bits 0–17	Bits 18–35
0	FORWARD REL POINTER	BACKWARD REL PTR
1	DEVICE ADDRESS	TYPE, SYH, IO, ER, RMV, ABSW, ABSU, NOTR, PHMH, SCU
2	PTW OFFSET IN SST	A STE OFFSET IN SST
3	PIN COUNTER	PAGE SYNCHRONIZATION REL PTR

CORE MAP ENTRY (CME) (cmp.incl.alm)

A HARDCORE DATABASE, LOCATED IN THE CORE_MAP SEGMENT. ONE CME PER CONFIGURED FRAME OF MAIN MEMORY.

- THE CORE MAP IS A PERMANENTLY WIRED, UNPAGED, SEGMENT CONTAINING AN ARRAY OF CORE MAP ENTRIES (CME'S)
  - ONE CORE MAP PER SYSTEM
  - ONE CME PER ADDRESSABLE MAIN MEMORY FRAME
    - IF THE CONFIGURATION HAS HOLES IN THE MEMORY ADDRESS ASSIGNMENTS, OR MEMORIES WHICH ARE TURNED OFF, THOSE CMES ARE PRESENT ANYWAY (BUT UNUSED) (0 TO HIGHEST FRAME ADDRESS)

PAGE CONTROL DATA BASES

CORE MAP

- EACH CME DESCRIBES THE STATUS OF ONE PAGE FRAME IN MAIN MEMORY INCLUDING:

  - THE DISK ADDRESS OF THE PAGE CURRENTLY OCCUPYING THE FRAME

  - ADDRESS OF THE ASTE AND PTW OF THE OCCUPANT

  - FLAGS INDICATING VARIOUS STATES AND PROPERTIES OF THE FRAME AND ITS OCCUPANT SUCH AS I/O IN PROGRESS, NOTIFICATION REQUESTED, AND PIN WEIGHT

- THE CME'S ARE KEPT IN A DOUBLE-THREADED CIRCULAR LIST POINTED TO BY sst.usedp

  - CME'S FOR FRAMES UNDERGOING I/O ARE TEMPORARILY THREADED OUT OF THE LIST

  - CME'S FOR FRAMES CONFIGURED BUT NOT PHYSICALLY PRESENT ARE ALSO THREADED OUT BUT WITH THREAD WORD "777777777777" OCTAL

  - THE REMAINING CME'S REPRESENT MAIN MEMORY FRAMES ACTIVELY IN USE - AND SUBJECT TO EVICTION BY THE PAGE REPLACEMENT ALGORITHM

PAGE CONTROL DATA BASES

SYSTEM SEGMENT TABLE (SST) HEADER

- THE FIRST 512 WORDS OF THE SST IS CALLED THE SST HEADER AND CONTAINS:
  - A LARGE NUMBER OF GLOBAL VARIABLES VITAL TO THE OPERATION OF THE STORAGE SYSTEM AND ITS SUBSYSTEMS
  - NUMEROUS CELLS USED TO METER THE STORAGE SYSTEM

- AMONG THOSE OF INTEREST TO PAGE CONTROL ARE THE FOLLOWING:
  - GLOBAL VARIABLES:
    - PAGE TABLE LOCK (sst.ptl)
    - NUMBER OF MAIN MEMORY FRAMES AVAILABLE FOR PAGING ACTIVITIES (sst.nused) AND NUMBER WIRED (sst.wired)
    - POINTERS TO THE BASE OF THE CME ARRAY AND TO THE CME OF THE "BEST" CANDIDATE PAGE FOR REPLACEMENT
    - PVT INDEX OF THE RPV (USED DURING INITIALIZATION)
  - METERS
    - THRASHING, POST-PURGE-TIME, PAGE FAULTS ON DIRECTORIES, RING 0 PAGE FAULTS, LOOP LOCK TIME, SEGMENT MOVES

PAGE CONTROL DATA BASES

SYSTEM SEGMENT TABLE (SST) HEADER

- PAGING METERS REPORTED BY file_system_meters SUCH AS STEPS, NEEDS, CEILING, SKIPS

OTHER DATA BASES

- ALTHOUGH BASICALLY DATA BASES OF VOLUME MANAGEMENT, THE FOLLOWING CONTAIN INFORMATION REQUIRED BY PAGE CONTROL (AS INDICATED)

- PHYSICAL VOLUME TABLE (PVT) - ONE PER SYSTEM

  - INFORMATION REQUIRED BY THE DISK DIM FOR I/O

  - INFORMATION USED BY THE DISK RECORD ALLOCATOR/DEALLOCATOR (free_store) SUCH AS:

    - THE NUMBER OF UNALLOCATED RECORDS LEFT ON THE VOLUME

    - THE LOCATION OF THE RECORD STOCK FOR THE VOLUME

- RECORD STOCKS

  - RECORD STOCKS ARE KEPT IN A WIRED SEGMENT: stock_seg

  - THE RECORD STOCK FOR A VOLUME IS A LIST OF SOME OF THE RECORDS WHICH ARE FREE ON THE VOLUME

    - WHEN THERE ARE NO MORE ENTRIES AVAILABLE IN THE STOCK, IT IS UPDATED FROM THE VOLUME MAP

OTHER DATA BASES

- IF THE STOCK BECOMES FULL, SOME OF ITS ENTRIES ARE UPDATED TO THE VOLUME MAP AND REMOVED FROM THE STOCK

- A COMPLEX MECHANISM (SEE volmap.alm, volmap_page.alm) MAKES IT POSSIBLE TO REFERENCE THE VOLUME MAP PAGES WHILE SATISFYING A PAGE FAULT

SERVICES OF PAGE CONTROL

PAGE FAULT HANDLING

- WITHIN ANY DEMAND PAGING ENVIRONMENT THE CHOICE OF WHICH PAGE TO EVICT IS CRUCIAL TO SYSTEM PERFORMANCE

- ONE OF THE BETTER CHOICES FOR EVICTION IS THE "LEAST RECENTLY USED" PAGE....OR (BECAUSE OF EFFICIENCY), THE "LEAST RECENTLY NOTICED AS BEING USED" PAGE.

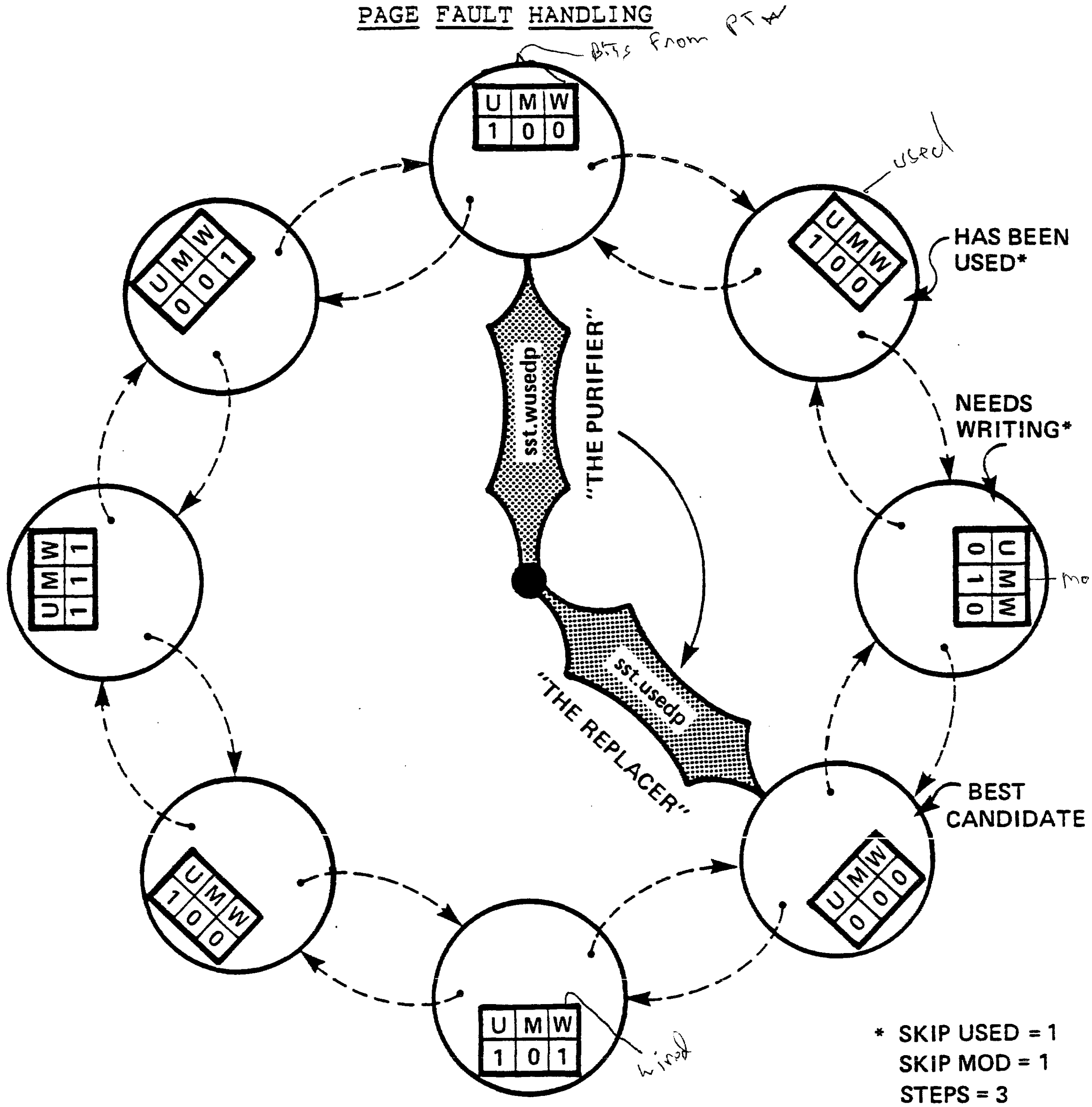
- THE MULTICS PAGE REPLACEMENT ALGORITHM (PRA), KNOWN IN THE LITERATURE AS THE "CLOCK" ALGORITHM WAS ONE OF THE FIRST EVER TO BE IMPLEMENTED

- THE VERSION AS IT EXISTS TODAY IS A DIRECT DESCENDANT OF Corbato's ORIGINAL ALGORITHM (SEE SECTION 5 OF THE "MULTICS STORAGE SYSTEM PLM", AN61, FOR A BIBLIOGRAPHY)

- PAGES ARE KEPT IN A CIRCULAR LIST, THE CORE USED LIST, IMPLEMENTED BY THE DOUBLY THREADED CME'S

- A POINTER, MAINTAINED IN THE SST, (sst.usedp) POINTS TO THE LOGICAL HEAD OF THIS LIST AS FOLLOWS:

PAGE FAULT HANDLING



THE CLOCK ALGORITHM

1. THE "REPLACER" SEARCHES FOR THE FIRST PAGE WHICH IS NEITHER WIRED NOR MODIFIED, AND HAS NOT BEEN USED SINCE LAST INSPECTED, MAKING THAT PAGE IMMEDIATELY AVAILABLE TO THE REQUESTOR.

2. THE "PURIFIER" THEN CATCHES UP, INITIATING WRITES FOR ALL "NOT USED-BUT MODIFIED" PAGES PASSED OVER BY THE "REPLACER" AND TURNING OFF THE USED FLAG FOR ALL USED PAGES.

SERVICES OF PAGE CONTROL

PAGE FAULT HANDLING

- PAGE FAULT HANDING IS THE MOST VISIBLE AND CRUCIAL SERVICE OF PAGE CONTROL

- A PAGE FAULT OCCURS WHEN A USER REFERENCES A PAGE OF SOME SEGMENT THAT IS NOT IN MAIN MEMORY

  - OR MORE SPECIFICALLY: HARDWARE ATTEMPTS TO USE A PTW THAT INDICATES ITS PAGE IS NOT IN THE MAIN MEMORY

- PAGE FAULT HANDLING IS IMPLEMENTED IN THE ALM PROGRAM page_fault WHICH IS INVOKED DIRECTLY BY THE FAULT VECTOR CODE (page_fault IS THE FAULT INTERCEPTOR FOR PAGE FAULTS)

- THE PRINCIPAL STEPS OF page_fault ARE:

  - SAVE ALL MACHINE CONDITIONS, MASK AGAINST INTERRUPTS, AND ESTABLISH A STACK FRAME ON THE BASE OF THE PROCESSOR DATA SEGMENT (PRDS), WHICH IS USED AS THE STACK FOR INTERRUPTS AND PAGE FAULTS

  - CHECK FOR ILLEGAL CONDITIONS AND CRASH IF SO

SERVICES OF PAGE CONTROL

PAGE FAULT HANDLING

- ATTEMPT TO LOCK THE PAGE TABLE LOCK (sst.ptl) AND WAIT IF UNSUCCESSFUL

- LOCATE THE RESPONSIBLE PTW AND ITS ASTE. THIS IS OFTEN THE MOST DIFFICULT TASK

  - IT IS DIFFICULT BECAUSE IT REQUIRES FETCHING THE SDW FROM THE DSEG, WHICH IS, ITSELF, PAGED, AND NOT GUARANTEED TO BE IN MEMORY

- CHECK FOR TWO WINDOW SITUATIONS INVOLVING SOME OTHER PROCESS HANDLING A PAGE FAULT FOR THE SAME PAGE:

  - IF PAGE IS NOW IN, THEN UNLOCK THE LOCK AND RESTART THE MACHINE CONDITIONS

  - IF PAGE IS BEING READ IN NOW, DEVELOP THE WAIT EVENT FOR THE PTW AND SKIP THE NEXT THREE STEPS

- INVOKE read_page TO FIND THE LEAST RECENTLY (NOTICED AS BEING) USED MAIN MEMORY FRAME, <u>BEGIN</u> THE PAGE-READING FUNCTION, AND DEVELOP THE WAIT EVENT

- EXECUTE THE REPLACEMENT ALGORITHM'S WRITE-BEHIND (PURIFIER) FUNCTION, CAUSING PASSED OVER WRITE REQUESTS TO BE QUEUED

- METER THE PAGE FAULT TO INCLUDE: TIME SPENT; MAIN MEMORY USAGE OF THIS PROCESS; RING ZERO, DIRECTORY, AND PER-PROCESS FAULTS

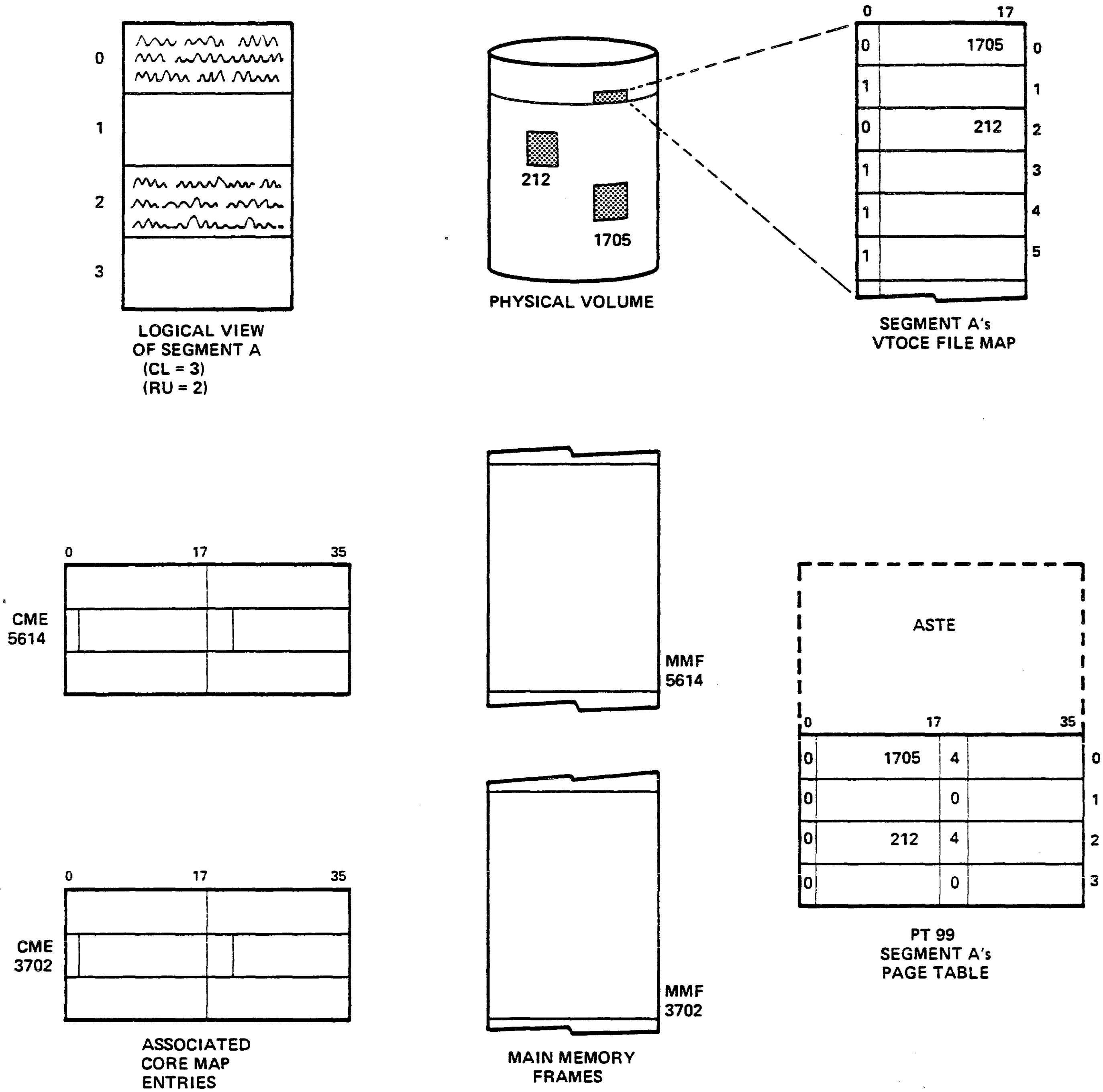
PAGE FAULT HANDLING

- TRANSFER TO THE TRAFFIC CONTROLLER, WHO PLACES THE PROCESS IN THE WAIT STATE, UNLOCKS THE PAGE TABLE LOCK, AND ABANDONS THE ENVIRONMENT (SEE "TRAFFIC CONTROL", TOPIC 10)

- WHEN THE PAGE READING I/O IS COMPLETE, THE EVENT WILL BE POSTED. THE WAITING PROCESS WILL BE GIVEN THE PROCESSOR AGAIN AND TRAFFIC CONTROLLER WILL TRANSFER THE FAULTING PROCESS TO page_fault$wait_return TO RESTART THE MACHINE CONDITIONS

SERVICES OF PAGE CONTROL

PAGE FAULT HANDLING

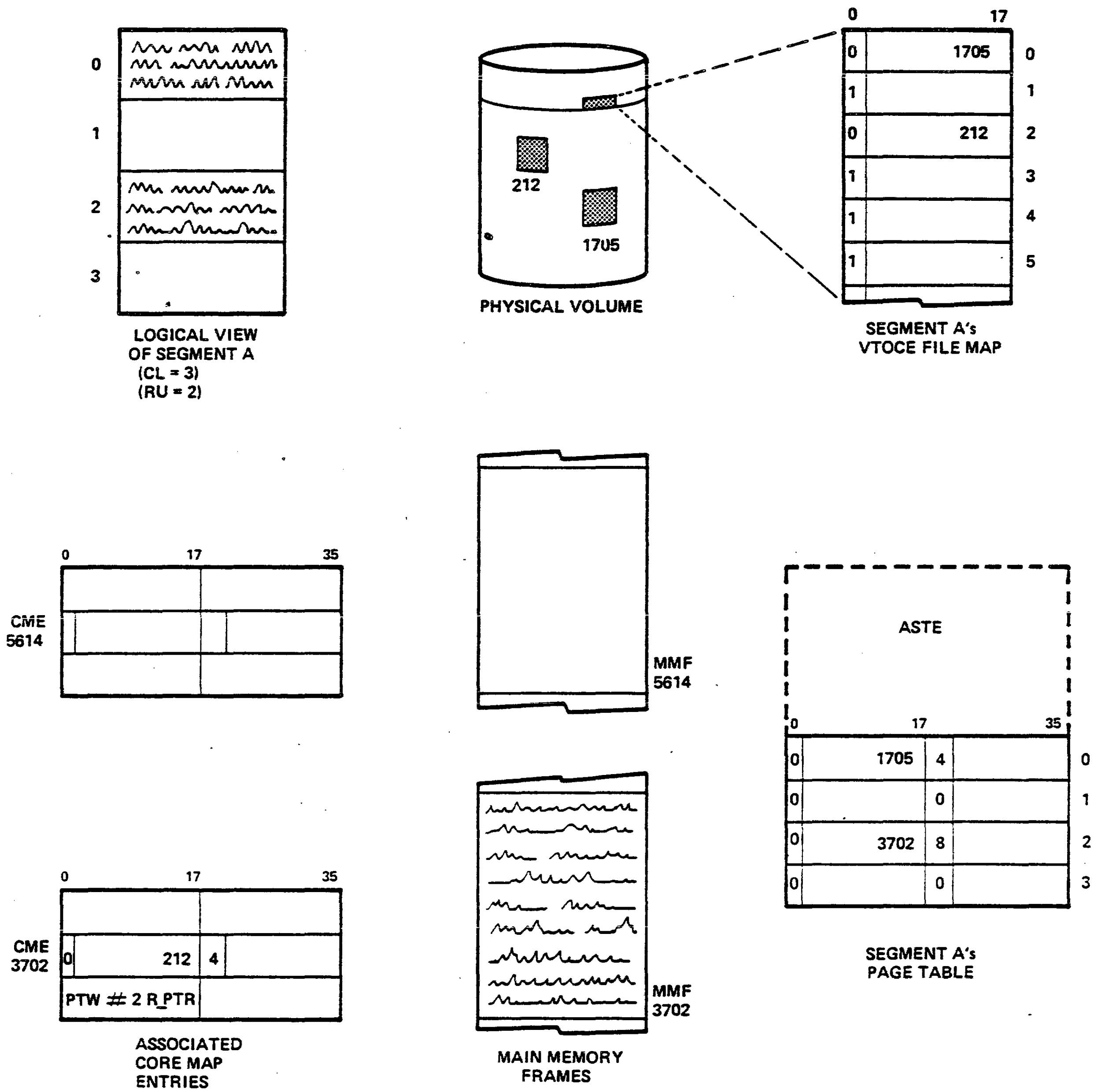


PAGE FAULT SCENARIO

SCENE 1: AFTER SEGMENT ACTIVATION

1. ASTE/PT PAIR FREED AND ALLOCATED TO SEGMENT A
2. VTOCE'S ACTIVATION DATA COPIED INTO ASTE
3. VTOCE'S FILE MAP DATA COPIED INTO PTW

PAGE FAULT HANDLING

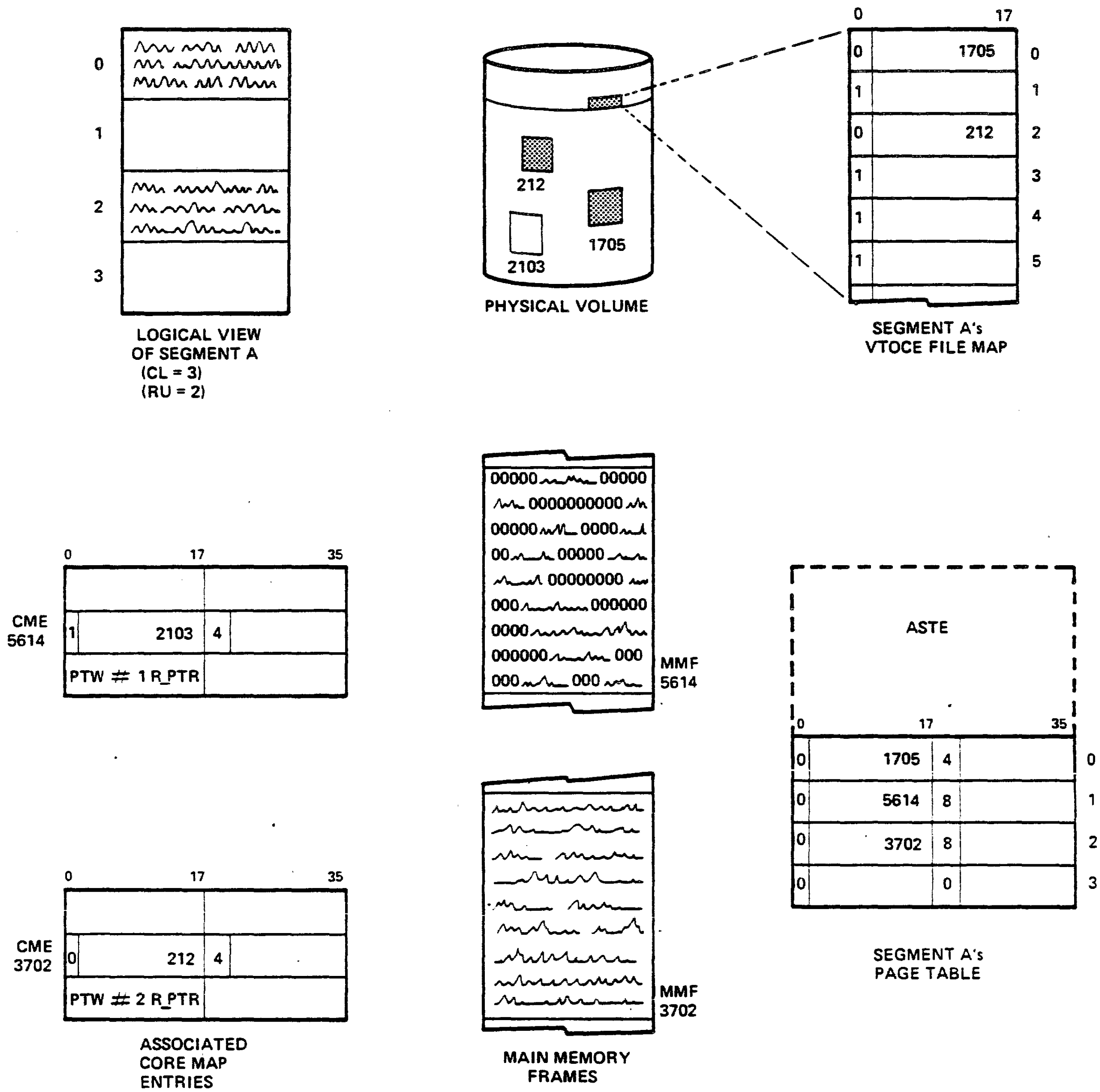


PAGE FAULT SCENARIO

SCENE 2: AFTER PAGE FAULT ON PAGE NO.2

1. FRAME 3702 FREED AND ALLOCATED TO PAGE NO. 2
2. PAGE NO. 2 COPIED INTO FRAME 3702
3. ADDRESS IN PTW NO. 2 COPIED TO CME 3702
4. ADDRESS IN PTW NO. 2 REPLACED WITH MAIN MEMORY ADDRESS

PAGE FAULT HANDLING



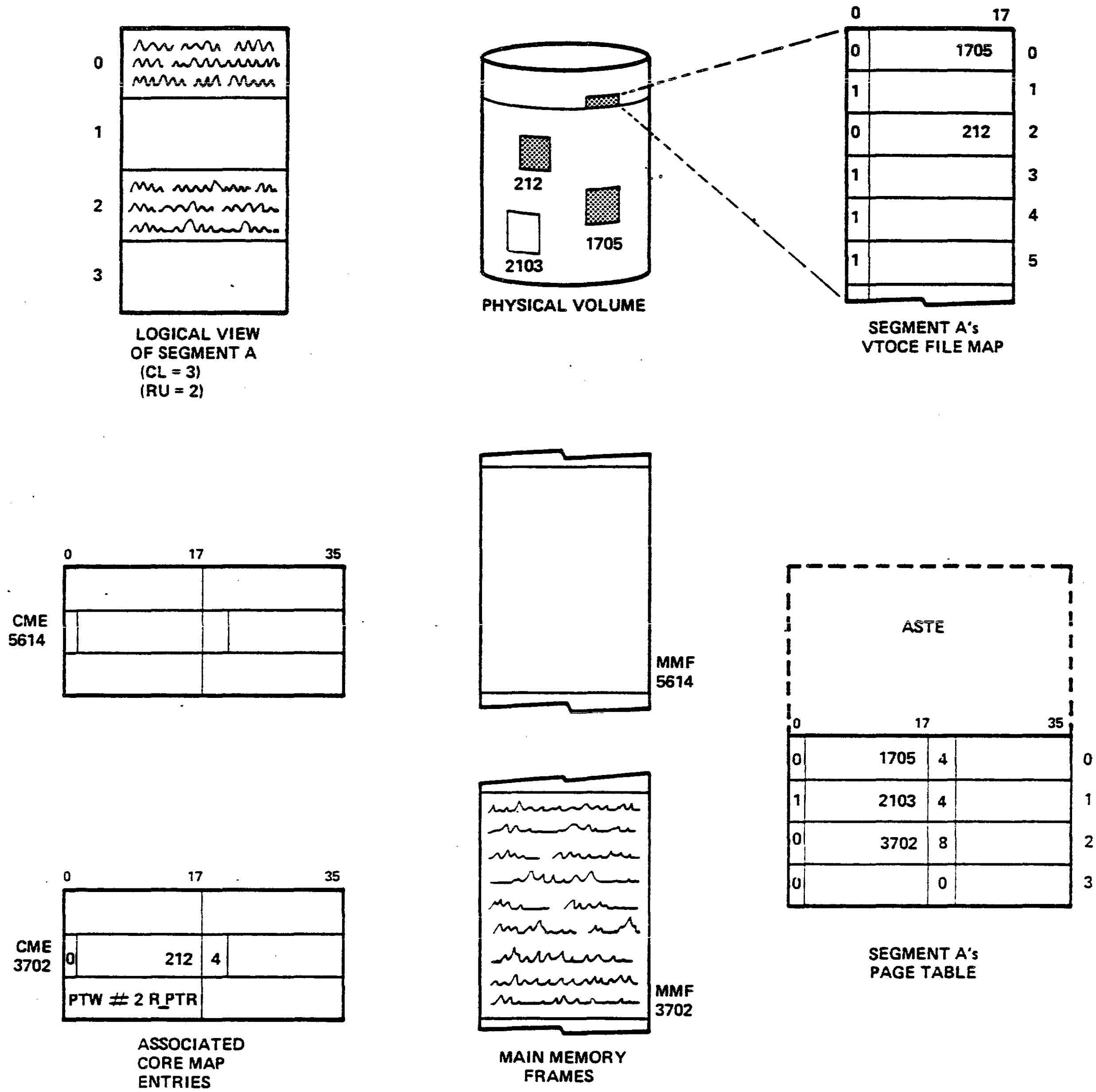
PAGE FAULT SCENARIO

SCENE 3: AFTER PAGE FAULT ON PAGE NO. 1

1. FRAME 5614 FREED AND ALLOCATED TO PAGE NO. 1
2. FRAME 5614 ZEROED BECAUSE OF PTW NO.1's NULL ADDRESS
3. RECORD 2103 ALLOCATED TO PAGE NO.1
4. ADDRESS FOR RECORD 2103 WRITTEN INTO CME 5614 AS A NULLED ADDRESS

SERVICES OF PAGE CONTROL

PAGE FAULT HANDLING

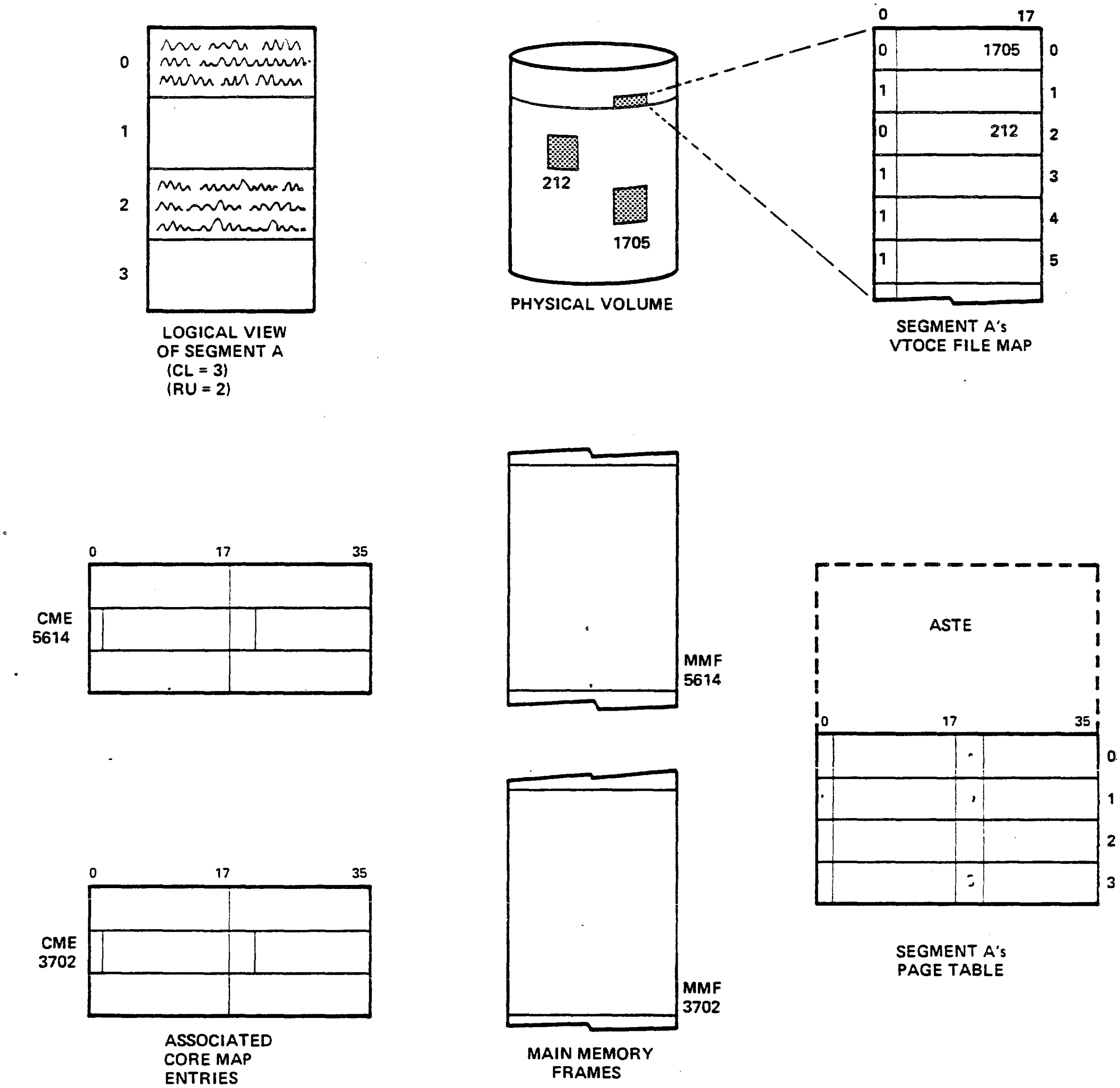


PAGE FAULT SCENARIO

SCENE 4A: PAGE NO.1 (UNMODIFIED) EVICTED FROM MAIN MEMORY

1. NULLED ADDRESS IN CME 5614 COPIED TO PTW NO.1
2. CME 5614 FREED

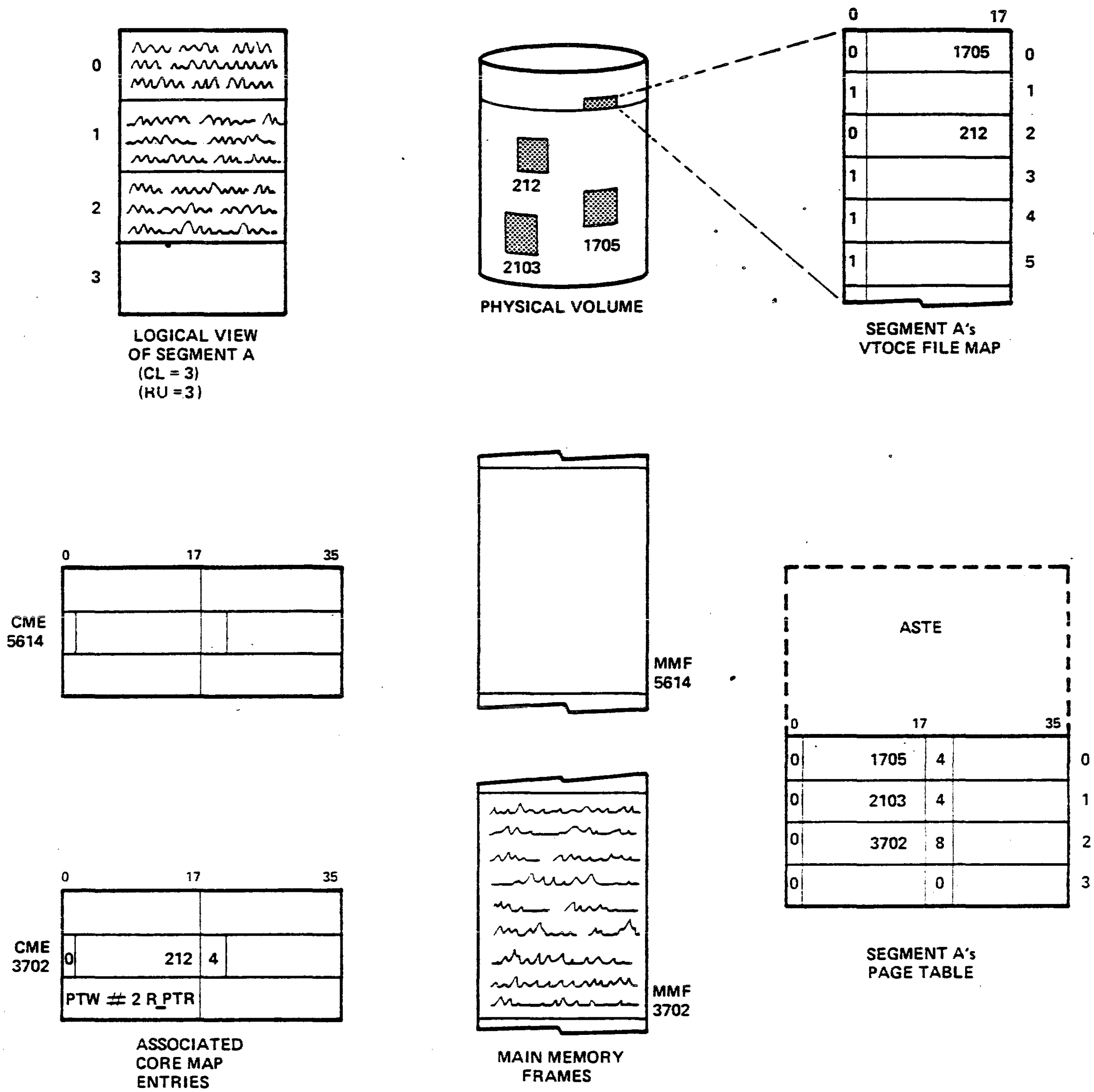
PAGE FAULT HANDLING



PAGE FAULT SCENARIO

SCENE 5A: AFTER SEGMENT DEACTIVATION

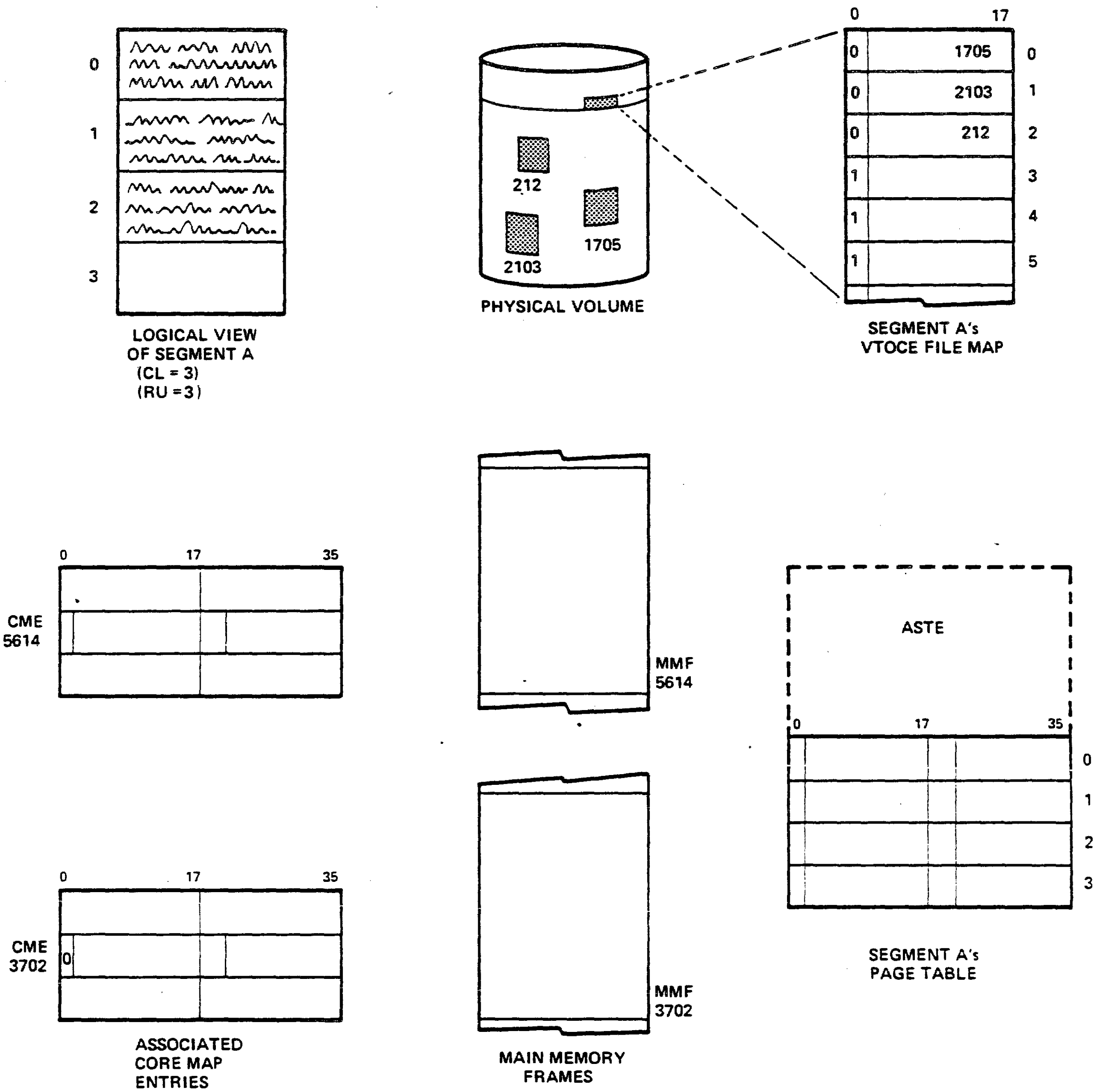
1. PAGE NO. 2 EVICTED FROM MAIN MEMORY AFTER ADDRESS IN CME 3702 COPIED TO PTW NO. 2
2. PTW ADDRESSES WRITTEN TO VTOCE FILE MAP WITH NULLED ADDRESS IN PTW NO. 1 CONVERTED TO A NULL ADDRESS
3. RECORD 2103 FREED
4. ASTE/PT PAIR FREED



PAGE FAULT SCENARIO

SCENE 4B: PAGE NO.1(MODIFIED) EVICTED FROM MAIN MEMORY
(AND DISK I/O KNOWN TO BE COMPLETE)

1. FRAME 5614 WRITTEN TO RECORD 2103
2. NULLED ADDRESS IN CME 5614 RESURRECTED AND COPIED TO PTW NO. 1
3. CME 5614 FREED



PAGE FAULT SCENARIO

SCENE 5B: AFTER SEGMENT DEACTIVATION

1. PAGE NO. 2 EVICTED FROM MAIN MEMORY
2. PTW ADDRESSES WRITTEN TO VTOCE FILE MAP
3. ASTE/PT PAIR FREED

SERVICES OF PAGE CONTROL

POST PURGING

- POST PURGING IS PERFORMED BY THE PROCEDURE post_purge

- POST PURGING IS AN OPTIONAL SERVICE USED TO OPTIMIZE THE PAGE REPLACEMENT ALGORITHM

- POST PURGING:

    - FAVORS THE REPLACING OF PAGES USED BY A PROCESS WHICH HAS JUST LOST ELIGIBILITY (SEE "TRAFFIC CONTROL", TOPIC 9)

    - IS A WORK CLASS SETTABLE ATTRIBUTE

- POST PURGING IS LARGELY USELESS TODAY, BECAUSE MAIN MEMORY SIZE IS SO MUCH GREATER

PAGE CONTROL METERS

file system meters

- FILE_SYSTEM_METERS - DISPLAYS MISCELLANEOUS METERING INFORMATION FOR THE FILE SYSTEM

  - ONLY PARTS RELEVANT TO PAGE CONTROL INCLUDED HERE; SEE TOPIC 7 (SEGMENT CONTROL) FOR THE REST

```
Total metering time    0:20:02

                      #           ATB
Needc             62654       19.194 msec.
Ring 0 faults                 16.639 %
PDIR faults                   50.607 %
Level 2 faults                21.556 %
DIR faults                     7.645 %
New Pages                     14.661 %
Volmap_seg            0        0.000 msec.
Zero pages          770     1561.779 msec.
Laps                105       11.453 sec.
Steps            361483        3.327 msec.
Skip             322555        3.728 msec.  89.231% of Steps
   wired          11057      108.761 msec.   3.428% of Skip
   used          109719       10.960 msec.  34.016% of Skip
   mod           140336        8.569 msec.  43.508% of Skip
   fc pin         37717       31.884 msec.  11.693% of Skip
   cl pin         23726       50.686 msec.   7.356% of Skip

3419 pages, 139 wired.
Average steps          5.770
```

close to # page faults (Need core)

From HC Partition on RLV

Level 2 dirs i.e. >sss >tools libraries etc

45% faults were from RLV

95% 5% for user dir segs.

Directory Segments

Page faults for null pages

write or evict zero page

around chain core map entries for page replacement Algorithm

low then page thrashing 5-6 seconds

Page rep Algorithm stats

Pin waits

Total +

Pin waits → Page Replacement Algorythm
favor pages used by
initializer to stay
in memory longer.
set for work classes.

Init Pinwait = 3
Users = 0
now default = 1
not big effect
Changed by TWC

Try reducing
initializors pin wait.

If used bit = 0
then look
at pinwait.
Pin wait gives
it extra laps

TOPIC IX

Traffic Control

Page

TRAFFIC CONTROL OVERVIEW

- FUNCTION

  - TRAFFIC CONTROL (OR THE "TRAFFIC CONTROLLER") IS RESPONSIBLE FOR MANAGING THE ASSIGNMENT OF PHYSICAL PROCESSORS TO MULTICS PROCESSES AND IMPLEMENTING THE SYSTEM'S WAIT/NOTIFY AND INTERPROCESS COMMUNICATION PRIMITIVES

  - THE FUNCTIONS ASSUMED BY THE TRAFFIC CONTROLLER ARE KNOWN AS MULTIPROGRAMMING, MULTIPROCESSING, SCHEDULING, DISPATCHING, PROCESSOR MANAGEMENT, AND INTERPROCESS COMMUNICATION.

  - ITS MAJOR FUNCTION IS ALLOWING PROCESSES TO AWAIT THE COMPLETION OF FILE SYSTEM OPERATIONS, SUCH AS PAGE I/O

  - TRAFFIC CONTROL CAN BE INVOKED BY SUBROUTINE CALLS AND INTERRUPTS

  - THERE ARE NO IMPORTANT USER SUBROUTINE INTERFACES, BUT THERE ARE PRIVILEGED SUBROUTINE INTERFACES FOR PROCESS CREATION, ADJUSTMENT OF SCHEDULING PARAMETERS, ETC.

- MAJOR DATA BASES

  - TC_DATA SEGMENT - ONE PER SYSTEM. CONTAINS THE FOLLOWING FOUR DATA BASES:

    - TC_DATA HEADER - ONE PER SYSTEM

      - CONTAINS VARIOUS METERS, COUNTERS AND POINTERS USED BY THE TRAFFIC CONTROLLER

TRAFFIC CONTROL OVERVIEW

- ACTIVE PROCESS TABLE (APT) - ONE PER SYSTEM
  - ACTIVE PROCESS TABLE ENTRY (APTE) - ONE OCCUPIED PER ACTIVE PROCESS (TOTAL NUMBER IS DETERMINED BY CONFIG DECK)
  - EACH APTE CONTAINS VARIOUS ATTRIBUTES OF AN ACTIVE PROCESS INCLUDING THE PROCESS ID, STATE, THE VALUE OF ITS DESCRIPTOR BASE REGISTER (DBR), SCHEDULING PARAMETERS, AND A POINTER TO THE PROCESS'S ITT ENTRIES
  - THE APTE CONTAINS ALL INFORMATION THE SUPERVISOR NEEDS TO KNOW ABOUT A PROCESS WHEN THE PROCESS IS NOT RUNNING
- INTERPROCESS TRANSMISSION TABLE (ITT) - ONE PER SYSTEM
  - ITT ENTRY - ONE OCCUPIED PER OUTSTANDING IPC WAKEUP
  - A QUEUE FOR TEMPORARILY STORING IPC WAKEUP INFORMATION (CHANNEL NAME, RANDOM DATA, PROCESS ID, ETC)
- WORK CLASS TABLE (WCT) - ONE PER SYSTEM
  - WORK CLASS TABLE ENTRY (WCTE) - ONE PER WORKCLASS
  - EACH WCTE CONTAINS ADMINISTRATOR DEFINED PARAMETERS OF THE WORKCLASS, VARIOUS METERS AND POINTERS

TRAFFIC CONTROL TERMINOLOGY

PROCESS:

AN ADDRESS SPACE AND AN EXECUTION POINT WITHIN THAT ADDRESS SPACE

MULTIPROGRAMMING:

PERTAINING TO THE CONCURRENT EXECUTION OF TWO OR MORE PROGRAMS BY INTERLEAVING THEIR EXECUTION

MULTIPROCESSING:

PERTAINING TO THE SIMULTANEOUS EXECUTION OF TWO OR MORE PROGRAMS BY A MULTIPROCESSOR SYSTEM (PARALLEL PROCESSING)

a single process is not on more than 1 cpu.

ELIGIBLE:

AN ADJECTIVE DESCRIBING THOSE PROCESSES ACTIVELY COMPETING FOR A PROCESSOR. ALL PROCESSES ARE EITHER ELIGIBLE OR INELIGIBLE

traffic control

making eligible

SCHEDULING:

PERTAINS TO THE ACT OF CHOSING AND PROMOTING AN "INELIGIBLE" PROCESS TO "ELIGIBLE" STATUS

Running state

DISPATCHING:

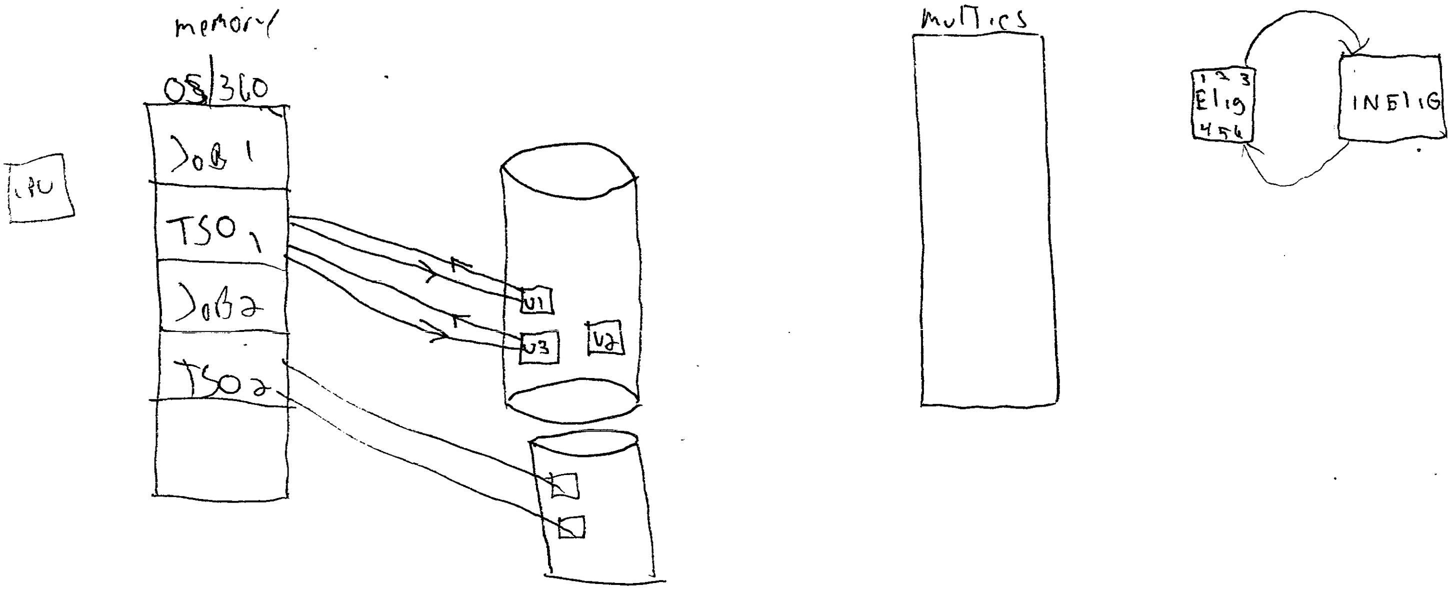
PERTAINS TO THE ACT OF CHOSING AND PLACING AN ELIGIBLE PROCESS IN THE "RUNNING" STATE (IE: EXECUTING INSTRUCTIONS ON A PROCESSOR)

TRAFFIC CONTROL TERMINOLOGY

WORK CLASS: A WELL DEFINED SET OF USERS (USUALLY CONSISTING OF ONE OR MORE PROJECTS) HAVING COMMON PERFORMANCE PARAMETERS. THE USER COMMUNITY MAY BE DIVIDED INTO NO MORE THAN 16 WORK CLASSES

IDEA

WORKING SET: Collection of pages needed in core at any 1 time. What they pay for.
THE SET OF PAGES A PROCESS TOUCHES DURING A GIVEN INTERVAL. THE SIZE OF A CURRENT WORKING SET IS PREDICTIVE OF FUTURE MEMORY REQUIREMENTS



working set:
memory units referred to as Frankstons
Developed Visicalc on Multics

loading getting processes ① seg wired in ~~the~~ memory

Time slice ie: Quantum — Amount of VCPU you can use while your eligible. Once you use it your marked ineligible

Prempt - one process on CPU voluntarily gives it up even though they need it.

VCPU - CPU billed to users, doesn't count interrupts & most faults.

TC DATA

tc_data HEADER
WORK CLASS TABLE (WCT) (0:16)
ACTIVE PROCESS TABLE (APT) (0:CONFIGURED)
INTERPROCESS TRANSMISSION TABLE (ITT) (0:CONFIGURED)

TC_DATA

A WIRED DATA BASE – ONE PER SYSTEM

TRAFFIC CONTROL DATA BASES

TC DATA

Word	Bits 0–17	Bits 18–35
0	FORWARD R_PTR	BACKWARD R_PTR
1	MBZ, WAKE, STOP, PE, HPRO, LOAD, ELIG, IDLE, INTE, PEP, DPR, RTB, AL, DBRL, SL, PROC, PAGE, INIT	EXECUTION STATE
2	NUMBER OF PAGE FAULTS	
3	PROCESS IDENTIFIER	
4	TIME HAS BEEN ELIGIBLE (TE)	
5	TIME SINCE TI CHANGED (TS)	
6	TIME SINCE LAST INTERACTION (TI)	
7	MAXIMUM VALUE FOR TI (TIMAX)	
8	IPC EVENT THREAD	
9	IPS MESSAGE(S)	
10	R_PTR TO ASTE OF PDS	R_PTR TO ASTE OF DSEG
11	R_PTR TO ASTE OR PRDS (WHILE RUNNING)	EXECUTION POINT (IN TRAFFIC CONTROL)
12	PROCESS IDENTIFIER TO NOTIFY ON TERMINATION	
13	LOCK IDENTIFIER	
14–15	CPU TIME USED	
16	WAIT EVENT	
17	WORK CLASS TABLE INDEX	PRIS, SP WAKEUPS, ABS, PR TAG
18–19	TIME OF LAST STATE CHANGE	

TRAFFIC CONTROL DATA BASES

TC DATA

20–21	WAKEUP EVENT FOR ALARM CLOCK MANAGER
22	ALARM TIME THREAD
22–23	ALARM TIME
24–25	EVENT CHANNEL TO NOTIFY ON PROCESS TERMINATION
26	WORKING SET ESTIMATE
27	MAXIMUM VALUE FOR TE (TEMAX)
28–29	DEADLINE SCHEDULING TIME
30	APTE LOCK
31	CPU MONITOR
32–33	PAGING MEASURE
34–35	ACCESS ISOLATION MECHANISM (AIM) CLASS
36–37	DESCRIPTOR BASE REGISTER VALUE (DBR)
38–39	VIRTUAL CPU TIME
40	ITT MESSAGES SENT AND NOT READ OUT
41	ITT MESSAGES RECEIVED AND NOT READ OUT
	8 WORDS FOR RESPONSE TIME METERING
50	SAVED VALUE OF TEMAX
51	PROCS REQUIRED
	12 WORDS OF PADDING (TO 64 WORDS)

ACTIVE PROCESS TABLE (APT)

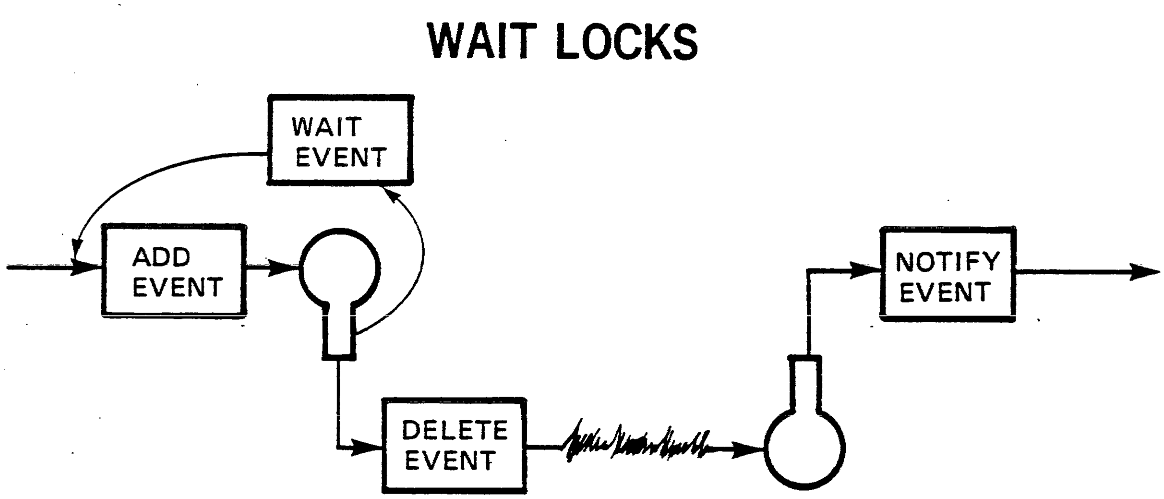
A HARDCORE (tc_data) DATA BASE – ONE PER SYSTEM

CONSISTS OF AN ARRAY OF APTE'S AS ABOVE

ACTIVE PROCESS TABLE ENTRY (APTE)

ONE PER ACTIVE PROCESS

WAIT LOCKS



SERVICES OF TRAFFIC CONTROL

PROCESSOR MULTIPLEXING

- SINCE THE NUMBER OF ACTIVE PROCESSES GENERALLY EXCEEDS THE NUMBER OF PROCESSORS (OFTEN 50:1) THE PROCESSORS MUST BE MULTIPLEXED

- PROCESSOR MULTIPLEXING IS THE PRIMARY RESPONSIBILITY OF THE TRAFFIC CONTROLLER

- THE MULTICS ARCHITECTURE DICTATES THE FOLLOWING AXIOMS:

  - ALL PROCESSORS ARE SYMMETRICAL

    - AN INTERRUPT IS SEEN BY ALL PROCESSORS AND IS SERVICED BY THE PROCESSOR THAT CLAIMS IT FIRST

      - NOT ACTUALLY TRUE, DUE TO HARDWARE CONNECTION LIMITATIONS

    - THERE ARE NO MASTER OR SLAVE PROCESSORS. ONLY A PROCESSOR DESIGNATED TO PERFORM BOOTLOAD AND SHUTDOWN (THE "BOOTLOAD PROCESSOR")

      - BOOTLOAD PROCESSOR CAN BE CHANGED AT ANY TIME BY DYNAMIC RECONFIGURATION

  - A PROCESSOR MAY BE "IN" AT MOST ONE PROCESS AT A TIME

  - A PROCESS MAY EXECUTE ON ONE AND ONLY ONE PROCESSOR AT A TIME. THE PROCESS MAY, HOWEVER, "RANDOMLY" MIGRATE FROM ONE PROCESSOR TO ANOTHER

SERVICES OF TRAFFIC CONTROL

PROCESSOR MULTIPLEXING

- A PROCESSOR WILL, AT ALL TIMES, BE "IN" A PROCESS

- PROCESSES MAY EXIST IN ONE OF SIX STATES - ONE OF WHICH IS "RUNNING" (IE: EXECUTING ON A PROCESSOR)

- THE TRAFFIC CONTROLLER CODE PERFORMS ALL STATE CHANGES. THE STATE CHANGES FALL INTO TWO CATEGORIES: SELF IMPOSED AND EXTERNALLY IMPOSED.

  - SELF IMPOSED STATE CHANGES (THE PROCESS MUST HAVE A PROCESSOR)

    - RUNNING -> READY (#1 -> #2)

      - THE PROCESS WAS TOLD (BY EITHER A CONNECT FAULT FROM ANOTHER PROCESSOR OR A TIMER RUNOUT) TO GIVE UP ITS PROCESSOR

      - THE PROCESS IS NOW WAITING FOR NO OTHER RESOURCE THAN A PROCESSOR

    - RUNNING -> WAITING (#1 -> #3)

      - THE PROCESS ISSUED A REQUEST FOR A "SYSTEM EVENT" (EG: A PAGE FAULT OR A WAIT LOCK)

      - "SYSTEM EVENTS" OCCUR AFTER A PREDICTABLY SHORT PERIOD OF TIME AND ARE HANDLED BY THE WAIT/NOTIFY MECHANISM

      - THE PROCESS IS NOW WAITING FOR A NOTIFY INDICATING THE COMPLETION OF THE EVENT (EG: THE ARRIVAL OF THE PAGE IN MAIN MEMORY)

PROCESSOR MULTIPLEXING

- RUNNING -> PAGE TABLE LOCK WAITING (#1 -> #6)
  - THE PROCESS ATTEMPTED TO LOCK THE PAGE-TABLE LOCK AND FOUND IT ALREADY LOCKED
  - THE PROCESS IS NOW WAITING FOR A NOTIFY INDICATING THE UNLOCKING (A SYSTEM EVENT)

- RUNNING -> BLOCKED (#1 -> #4)
  - THE PROCESS ISSUED A REQUEST FOR A "USER EVENT" (EG: A READ FROM THE TERMINAL)
  - "USER EVENTS" OCCUR AFTER A PREDICTABLY LONG PERIOD OF TIME AND ARE HANDLED BY THE BLOCK/WAKEUP MECHANISM
  - THE PROCESS IS NOW WAITING FOR A WAKE_UP INDICATING THE COMPLETION OF THE EVENT (EG: A LINE_FEED GENERATES A WAKE_UP) ARE HANDLED BY THE BLOCK/WAKE_UP MECHANISM
  - A PROCESS GOING BLOCKED GIVES UP ITS RING ZERO STACK

- RUNNING -> STOPPED (#1 -> #5)
  - THE PROCESS EXECUTED THE logout OR new_proc COMMAND, EVENTUALLY CALLING hcs_$stop_process
  - THE PROCESS IS PROHIBITED FROM RUNNING AGAIN AND WILL QUICKLY BE DESTROYED BY THE INITIALIZER

SERVICES OF TRAFFIC CONTROL

PROCESSOR MULTIPLEXING

- EXTERNALLY IMPOSED STATE CHANGES (THE STATE IS CHANGED BY ANOTHER PROCESS)

  - READY -> RUNNING (#2 -> #1)

    - THE PROCESS (WHICH WAS WAITING FOR NO OTHER RESOURCE THAN A PROCESSOR) WAS CHOSEN AS SUCCESSOR BY A PROCESS RELINQUISHING A PROCESSOR

    - THE PROCESS IS NOW EXECUTING

  - BLOCKED -> READY (#4 -> #2)

    - SOME OTHER PROCESS SENT A WAKEUP INDICATING THE COMPLETION OF THE EVENT THIS PROCESS WAS WAITING ON, AND CHANGE THE STATE OF THIS PROCESS

    - THE PROCESS IS NOW WAITING FOR NO OTHER RESOURCE THAN A PROCESSOR

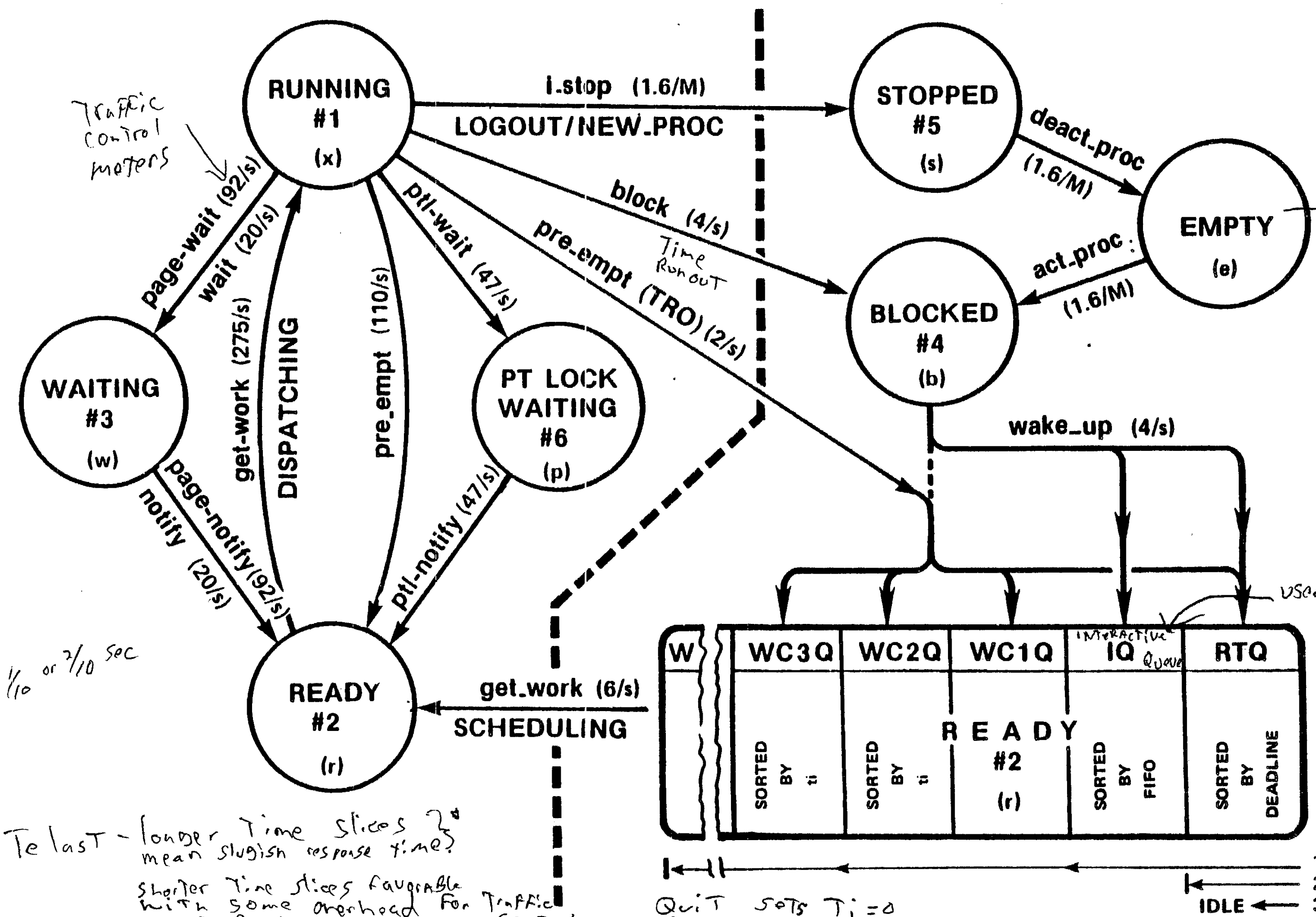
  - WAITING -> READY (#6 -> #2)
    PTL-WAITING -> READY (#3 -> #2)

    - SOME OTHER PROCESS SENT A NOTIFY INDICATING THE COMPLETION OF THE EVENT THIS PROCESS WAS WAITING ON, AND CHANGED THE STATE OF THIS PROCESS

    - THE PROCESS IS NOW WAITING FOR NO OTHER RESOURCE THAN A PROCESSOR

Blocked - waiting for something doesn't know how long

TRAFFIC CONTROL STATES



Te last - 1/10 or 3/10 sec

Te last - longer Time slices mean sluggish response time?

Shorter Time slices favorable with some overhead for Traffic control won't penalize long users noticibly.

ELIGIBLE PROCESSES (13) **INELIGIBLE PROCESSES (87)**

* ALL FIGURES ARE BASED ON METERS FROM A 100 USER, 3 CPU, 2.5M MEMORY, MAXE = 16 SYSTEM.

TRAFFIC CONTROL SCENARIO

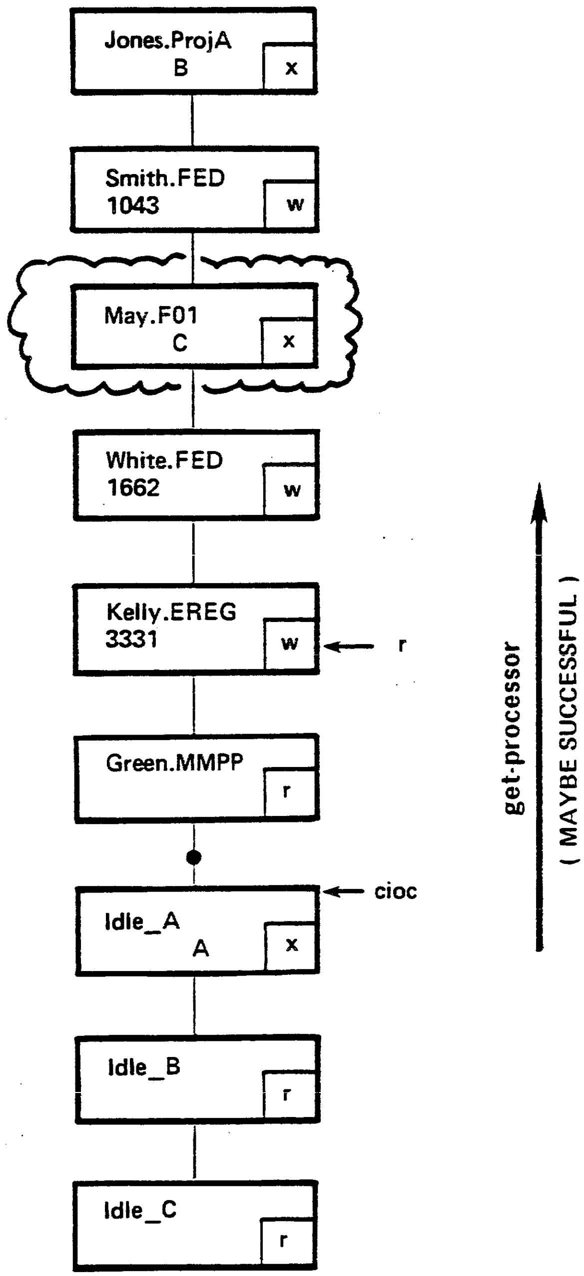
SCENE 1: CPU "C" TAKES AN INTERRUPT

1. EVENT 3331 OCCURES. CPU "C" TAKES THE INTERRUPT AND STORES MACHINE CONDITIONS

2. May.F01 NOTIFIES Kelly.EREG BY CHANGING HIS STATE TO READY (r)

3. May.F01 CALLS get-processor WHICH LOOKS FOR AN EXECUTING (x) PROCESS LOWER THAN Kelly.EREG, FINDS IDLE_A, AND SENDS CPU "A" A CONNECT FAULT (cioc INSTRUCTION)

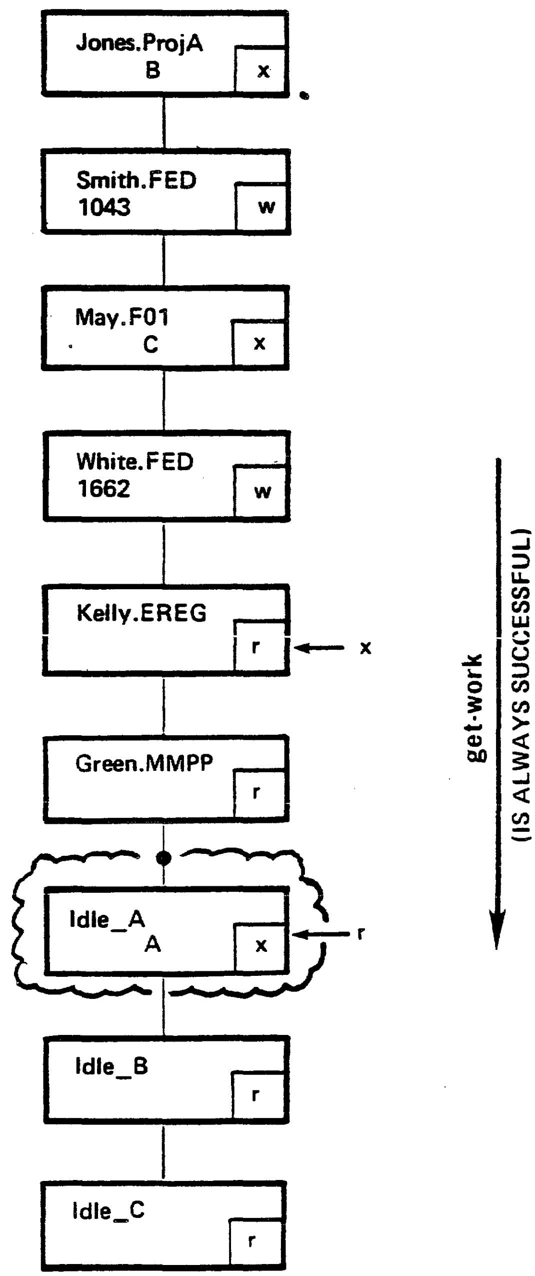
4. May.F01 RESUMES PRIOR ACTIVITY BY RESTORING MACHINE CONDITIONS



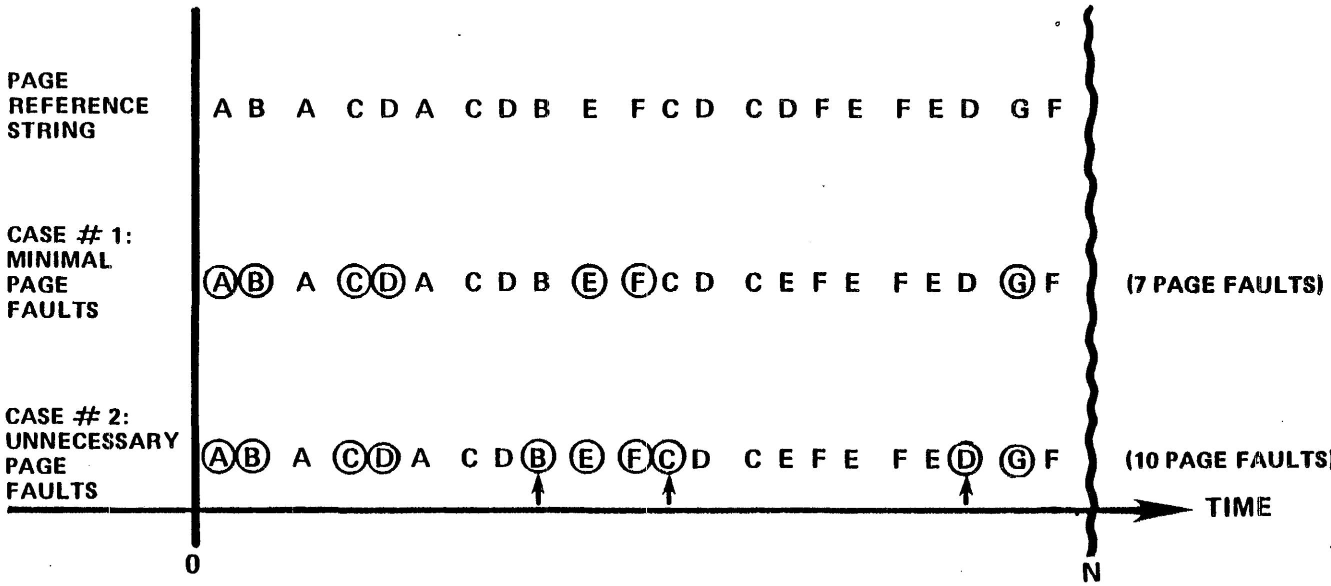
TRAFFIC CONTROL SCENARIO

SCENE 2: CPU "A" RECEIVES CONNECT FAULT

1. CPU "A" RECEIVES CONNECT FAULT AND STORES MACHINE CONDITIONS
2. Idle_A BEGINS PRE-EMPTION BY CHANGING STATE TO READY (r)
3. Idle_A CALLS get-work WHICH LOOKS FOR THE HIGHEST READY (r) PROCESS, FINDS Kelly.EREG, CHANGES HIS STATE TO EXECUTING (x) AND CPU TAG TO "A", AND LOADS HIS DBR
4. Kelly.EREG RESUMES PRIOR ACTIVITY BY RESTORING MACHINE CONDITIONS



WORKING SETS AND PAGE THRASHING



WORKING SET = 7 PAGES

- CASE # 1 : THRASHING = $\frac{0}{7}$ = 0%
- CASE # 2 : THRASHING = $\frac{3}{10}$ = 30%

IDLE CATEGORIES

		ELIGIBLE			NOT ELIGIBLE		
		RUNNING	WAITING	READY	READY	READY, BUT IN OVERUSED WORKCLASS	
no eligible proc's & none in queues.	ZERO IDLE	0	0	0	0	0	good idle
Non multi-programming. Some eligible, but not enough to keep all processors busy	NMP IDLE	1 0 1	1 2 0	0 0 0	0 0 0	0 0 0	
would have been used by someone but they didn't have dseg in memory.	LOADING IDLE	1	2*	0	0	0	
Multi-programming users eligible but we are at MAXE meaning, MAXE too low or another problem	MP IDLE	1 0	2 3	0 0	>0 >0	≥0 ≥0	Bad idle
CPU gives time to idle instead of giving it to process at MAX% for the work class	WORKCLASS IDLE	1 0 1	1 2 0	0 0 0	0 0 0	>0 >0 >0	Administrative idle

example: 2 CPU's MAXE=3

* last process being loaded.

TRAFFIC CONTROL METERS

total time meters

read from Bottom

* TOTAL_TIME_METERS - OVERVIEW OF HOW THE SYSTEM IS USING ITS RESOURCES, ALSO MEASURED AGAINST NON-IDLE TIME

Total metering time 0:20:36 Better View

non idle

	%	%NI	AVE	
		normal	normal	
Page Faults	6.55	7.19	2130.424	high is seg fault Too few entries is AST
PC Loop Locks	0.39	0.43	1993.209	never or ignore These
PC Queue	0.86	0.94	348.946	
Seg Faults	1.74	1.90	9579.170	Activating or deactivating segs
Bound Faults	0.15	0.16	17426.208	
disk I/O — Interrupts	9.02	9.89	1713.504	related To page fault activity
handling of connect faults → Other Fault	8.45	9.27		
Getwork (part of other fault)	4.54	4.98	660.550	
TC Loop Locks	0.20	0.22	247.788	
Post Purging	0.36	0.39	1407.132	
MP Idle	0.70	0.77		Time lost because maxE should be < 1 or 2 %
Work Class Idle	0.98	1.08		
Loading Idle	0.16	0.17		
NMP Idle	8.84			good idle
Zero Idle	0.00			
Other Overhead	0.03	0.04		fudge factor
Virtual CPU Time	62.53	68.59		Billed To users

TRAFFIC CONTROL METERS

traffic control meters

- TRAFFIC_CONTROL_METERS - DISPLAY THE STATE OF THE SCHEDULER
  - OUTPUT COMES IN THREE PARTS, SHOWN OUT OF ORDER HERE:
    - QUEUE LENGTHS AND RESPONSE TIME - THESE ARE WEIGHTED AVERAGES OVER THE LAST FIFTEEN SECONDS.
    - ACTIVITIES VERSUS DEPTH - HOW DEEP THE TRAFFIC CONTROLLER HAD TO SEARCH TO FIND A SCHEDULABLE PROCESS
    - MISCELLANEOUS COUNTERS AND FREQUENCIES OF VARIOUS EVENTS

```
Total metering time     0:20:34

Ave queue length      16.52
Ave eligible          13.31
Response time          0.264 sec
```

Avg procs not blocked. current load

Avg procs

crude measure, ~~not~~ really what

time in queue to time eligible. not really what user sees.

DEPTH	%PF	TBPF	%GTW	TBS	%CPU
1	12.0	22.8	10.8	11.6	8.1
2	11.8	21.0	9.2	12.2	7.4
3	10.8	24.9	8.7	14.1	8.0
4	10.2	27.9	8.5	15.2	8.4
5	9.5	29.8	8.3	15.5	8.4
6	8.4	33.5	7.8	16.5	8.4
7	7.4	36.3	7.3	16.8	8.0
8	30.0	48.6	39.5	16.8	43.3

response_meters good indication of response time.

TRAFFIC CONTROL METERS

traffic control meters

COUNTER	TOTAL	ATB	#/INT
Interactions	7977	0.155 sec	
Loadings	12161	0.102 sec	1.525
Blocks	14082	0.088 sec	
Wakeups	36078	0.034 sec	
Schedulings	12591	0.098 sec	1.578
Lost priority	1	1234.756 sec	
Priority boosts	0	0.000 sec	
I/O boosts	578	2.136 sec	
Wait Page	127040	9.719 msec	15.926
Wait PTL	75691	16.313 msec	9.489
Wait Other	31912	38.693 msec	4.001
Total Waits	234643	5.262 msec	29.415
Notify Page	128954	9.575 msec	
Notify PTL	75691	16.313 msec	
Notify Other	25330	48.747 msec	
Total Notifies	229975	5.369 msec	
Get Processor	245856	5.022 msec	
Pre-empts	94235	13.103 msec	11.813
Getwork	338802	3.644 msec	
Retry getwork	4988	0.248 sec	
Extra notifies	2949	0.419 sec	
Last EN event	000000000071		
Last NTO event	033022237767		

⊗ ALARM_CLOCK_METERS - DISPLAYS INFORMATION ABOUT THE USER ALARM TIMER FACILITY (HARDCORE INTERFACE FOR timer_manager_)

```
Total metering time     0:20:31

No. alarm clock sims.    2171
Simulation lag              5.245 msecs.
Max. lag                  1.7340314e4 msecs.
```

TRAFFic_control
PXSS.alm
Processor exchange + STACK Storing

help hardcore_wait

TRAFFIC CONTROL METERS

traffic control queue

SNAPSHOT

- TRAFFIC_CONTROL_QUEUE - DISPLAYS THE CURRENT CONTENTS OF THE SCHEDULER QUEUES, USEFUL FOR GETTING AN IDEA OF WHAT THE USER PROCESSES ARE DOING

Time elg - Time slice 1 → 15 sec

Time eligible - how much of time slice consumed

what process is waiting for

I FIRST PART OF OUTPUT IS ELIGIBLE QUEUE:

Time since state change

Total CPU Time → delta Time used

delta Page faults

since last TCQ

Time since interaction updated AT end of slice

System ~~Time of last change~~ Time of last ACTIVITY

device - absolete

working set - ⊖

work class

avq = 13, elapsed time = 1247 sec, 64 active last 15 sec.

flags	dtu	dpf	temax	te	ts	ti	tssc	event	d	ws	wc	process
rWLE(d)	148	6946	2097	37	0	0	-0.001	0	0	6	0	initializer
xLED(c)	15	1111	1000	910	2012	1897	0.001	0	0	11	6	Sibert
rLE(d)	14	1370	500	89	0	0	-0.009	0	0	0	4	Diaz
rLE(d)	15	823	500	13	0	0	-0.009	0	0	64	8	JCrow
wLE(d)	13	634	500	422	0	0	0.018	316537	0	3	3	Gintell
xWLED(a)	6	108	1000	495	2054	2004	0.010	0	0	13	6	Brunelle
wLE(d)	16	864	1000	85	0	510	0.016	504031	0	46	6	WFeck
wWLE(b)	468	2734	1000	315	3010	8000	0.013	224641	0	4	3	Spratt
wLE(d)	17	686	500	60	0	0	0.007	165111	0	3	4	RTowle
wLE(b)	12	520	500	50	0	0	0.005	71	0	3	3	Kress
wLE(d)	69	1895	500	21	0	0	0.001	177022	0	0	2	OPCTL
xLED(b)	12	872	500	85	0	0	0.005	0	0	0	3	Pandolf
rLE(d)	67	3279	500	0	0	0	-0.006	0	0	0	3	Lackey

ready — running — waiting — "W" wakeup waiting — loaded — APT — eligible

→ real Time head of Queue

lock in MCS

events
400000 → AST lock (small clue
digits → disk reads ; Address of PTW

TRAFFIC CONTROL METERS

traffic control queue

! SECOND PART IS REALTIME, INTERACTIVE, AND ALL WORKCLASS QUEUES:

```
REALTIME QUEUE:

INTERACTIVE QUEUE:

WORKCLASS  2 QUEUE: credits =   576 ms.

WORKCLASS  3 QUEUE: credits =   242 ms.
r        289   5326  1000    0     0   503   0.218     0 0   22   3 Dupuis
r        131   2513  1000    0  4010  8000   0.128     0 0    0   3 Falksenj

WORKCLASS  4 QUEUE: credits =  2601 ms.

WORKCLASS  5 QUEUE: credits =  4000 ms.

WORKCLASS  6 QUEUE: credits =  -563 ms.

WORKCLASS  7 QUEUE: credits =  3962 ms.
rW         5    166   500    0     0     0   0.192     0 0    2   7 Saccuci

WORKCLASS  8 QUEUE: credits =  3934 ms.

WORKCLASS  9 QUEUE: credits =  2216 ms.

WORKCLASS 10 QUEUE: credits =  4000 ms.

WORKCLASS 11 QUEUE: credits =  4000 ms.
```

used to give out correct time to work classes

TRAFFIC CONTROL METERS reset AT shift change

work class meters

* WORK_CLASS_METERS - DISPLAY THE VARIOUS WORKCLASS PARAMETERS, AND SHOW WHAT RESOURCES EACH WORKCLASS IS CONSUMING.

(Any Time scewed % TOTAL CPU Time

VCPU/eligibility not interesting

interactive queue

Total metering time 0:20:38 P.n Wait

load ctrl groups

WC	%GUAR	%MAX	%TCP	V/ELIG	PW	IRESP	IQUANT	RESP	QUANT	P	M	R	I	LCG
0			3.	0.12	3	0.26	2.10	0.26	2.10	P	O	R	I	Init
1			3.	0.09	1	0.25	0.75	0.50	1.00	P	O	R	I	RTime
2	7.		15.	0.44	1					P	O		I	System SysAdm OPR FED
3	32.		44.	0.49	1					P	O		I	SysProg SysDev
4	9.	14.	4.	0.26	1					P	O			SEngr
5	20.		2.	0.46	1					P	O		I	HEngr
6	12.	16.	8.	0.25	1					P	O			MktUS MktFor MktEd
7	3.	7.	4.	0.36	1					P	O			DS-CC
8	6.		0.	0.18	1					P	O		I	OffAuto
9	4.	8.	2.	0.62	1					P	O			Misc Mfg
10	3.		0.	0.55	1					P	O		I	Other
11	4.		2.	0.16	1					P	O		I	Special

TCPU percents (%GUAR) control non-realtime work_classes.

IT Traffic control goes up
from 2 TO 5 % OK.

IF YOU have Short Te last
would normally Push up
Page faults BUT not IF
you have alot OF Memory.
IT makes longer JOBS stay
in Que longer. Shorter JOBS
run through Quicker

TRAFFIC CONTROL METERS

respons meters

IMPORTANT

- RESPONSE_METERS - DISPLAYS RESPONSE TIME, BASED ON TERMINAL INTERACTIONS, ON A PER-WORKCLASS BASIS
  response Seen By users

Total metering time 0:20:36 Time leaving STATE until Blocked Time returning

WC	---Thinks/-- ---Queues--- #	Avg	----Response Times by VCPU Range---- -VCPU Range- From	To	# Int	Avg VCPU	Avg RT	Resp Fact	Load Control Group
		Time Spent			Inter Actions				UPTO 10 is reasonable.
0	86	2.70	0.00	0.50	113	0.04	0.42	9.34	Init
	92	0.15	0.50	1.00	3	0.55	5.51	10.00	
			1.00	10.00	3	2.43	14.96	6.15	
			-----	-----	119	0.12	0.91	7.76	
1	35	11.78	0.00	0.50	34	0.11	0.96	8.74	RTime
	39	0.21	0.50	1.00	2	0.83	4.06	4.87	
			-----	-----	36	0.15	1.13	7.55	
2	593	14.90	0.00	0.50	620	0.05	0.49	10.50	System SysAdm OPR FED
	612	0.15	0.50	1.00	28	0.71	3.61	5.10	
			1.00	10.00	39	1.77	8.39	4.74	
			-----	-----	687	0.17	1.07	6.22	.66/.08 = 7.96
3	2496	6.38	0.00	0.50	2993	0.08	0.66	7.96	SysProg SysDev
	2622	0.17 Buckets	0.50	1.00	117	0.68	4.22	6.16	
			1.00	10.00	66	2.46	13.47	5.49	
			10.00	99.99	10	56.85	99.99	3.33	
			-----	-----	3186	0.33	1.65	4.96	
4	581	15.82	0.00	0.50	663	0.05	0.71	12.94	SEngr
	590	0.26	0.50	1.00	13	0.64	4.64	7.30	
			1.00	10.00	8	3.47	32.07	9.24	
			-----	-----	684	0.11	1.15	10.88	
5	133	29.51	0.00	0.50	148	0.06	0.83	12.95	HEngr
	141	0.17	0.50	1.00	3	0.69	4.64	6.70	
			1.00	10.00	5	3.15	10.24	3.25	
			10.00	99.99	2	26.69	48.84	1.83	
			-----	-----	158	0.51	1.81	3.55	

Times Paused Between interactions

Setting Te first for WC3
1) look AT 0.08 SO IF
Te first .20 → .50 Probably .4 we want mid range.

~~heavily loaded~~ -

First Place To look

TRAFFIC CONTROL METERS

respons meters

6	1180	11.65	0.00	0.50	977	0.05	1.13	20.91	MktUS MktFor MktEd
	1211	0.58	0.50	1.00	24	0.64	4.95	7.74	
			1.00	10.00	16	1.89	11.88	6.29	
			-----	-----	1017	0.10	1.39	14.38	
7	259	14.65	0.00	0.50	292	0.08	1.53	20.17	DS-CC
	287	0.98	0.50	1.00	9	0.71	9.03	12.66	
			1.00	10.00	17	2.04	14.76	7.24	
			10.00	99.99	1	12.22	99.99	9.45	
			-----	-----	319	0.24	2.81	11.86	
8	73	2.69	0.00	0.50	79	0.06	0.31	5.45	OffAuto
	74	0.05	0.50	1.00	2	0.60	6.40	10.62	
			1.00	10.00	1	3.20	6.73	2.10	
			-----	-----	82	0.11	0.54	4.94	
9	94	41.91	0.00	0.50	80	0.11	1.23	11.07	Misc Mfg
	96	0.30	0.50	1.00	11	0.69	4.74	6.91	
			1.00	10.00	13	2.99	12.82	4.30	
			-----	-----	104	0.53	3.05	5.74	
10	7	99.99	0.00	0.50	10	0.11	1.45	13.82	Other
	8	0.21	1.00	10.00	1	3.88	39.11	10.09	
			-----	-----	11	0.45	4.88	10.89	
11	417	12.71	0.00	0.50	445	0.03	0.50	15.32	Special
	420	0.16	0.50	1.00	5	0.57	3.61	6.29	
			1.00	10.00	4	1.65	11.40	6.90	
			-----	-----	454	0.05	0.63	11.93	
All	5954	11.13	0.00	0.50	6454	0.07	0.77	11.40	
	6192	0.29	0.50	1.00	217	0.68	4.54	6.70	
			1.00	10.00	173	2.31	13.11	5.68	
			10.00	99.99	13	48.77	99.99	3.32	
			-----	-----	6857	0.24	1.51	6.39	

86797 calls to meter_response_time   283 invalid transitions.
Overhead =   0.09% (0.052 ms./call)

post purge meters

Always Q

- POST_PURGE_METERS - DISPLAY THE STATE OF POST-PURGE ACTIVITY.

  - CONSIDERABLY MORE DETAILED METERS ARE KEPT BY RING ZERO, BUT NOT REPORTED AND NOT PARTICULARLY INTERESTING.

```
Total metering time     0:20:29

Post purge time          1.41 msec. (0.36% of system)
Ave list size           17.76 entries
Ave working set          5.59 pages
Working set factor       0.50
Working set addend          0
Thrashing percentage     1.34 %
Ave post in core        10.65           (59.94 %)
```

print tuning parameters also CTP

* PRINT_TUNING_PARAMETERS - PRINT VALUES FOR SYSTEM CONTROL PARAMETERS. MOST CONTROL THE SCHEDULER

Current system tuning parameters:

how often a process will attempt to prempt itself get. process

normally ?: rethreaded to top of wclass ie: cheat. ?; * quit priority so if quit priority =1 it won't change anything.

notify timeout - waiting forevent to complete + be notified. if interval exceeded, it notifies them anyway.

Syserr_codes.incl. PL1

tefirst	0.5 seconds
telast	1. seconds
timax	8. seconds
priority_sched_inc	80. seconds
min_eligible	2.
max_eligible	20.
max_batch_elig	0 ABS jobs active at 1 time
working_set_factor	0.5
working_set_addend	0
deadline_mode	off benchmarks makes all wcs realtime
int_q_enabled	on global → norm won't follow to closely if you use interactive que
post_purge	on default is off - useless
pre_empt_sample_time	0.04 seconds
gp_at_notify	off use prempt sample
gp_at_ptlnotify	off time instead
process_initial_quantum	2. seconds to get logged in fast
quit_priority	0.
gv_integration governing	4. seconds period of time remembering workclasses for enforcing percentages.
realtime_io_priority improves response for tape users	on off default
realtime_io_deadline	0. seconds
realtime_io_quantum	0.005 seconds
notify_timeout_interval	30. seconds
notify_timeout_severity	0 → print on console, log, alarm
write_limit	724

* THESE ARE "INTERNAL", NORMALLY NEVER CHANGED, AND ONLY PRINTED IF THE -all CONTROL ARGUMENT IS GIVEN

stack_truncation	on
stack_truncation_always	off
stk_trunc_block_avg_factor	0.25
trap_invalid_masked	off
meter_ast_locking	off
checksum_filemap	on

PRINT_APT_ENTRY - INTERPRETS AND DUMPS AN APT ENTRY

exception rather than rule

```
! print_apt_entry Sibert -dump

  Sibert.Multics.a b.h126 at 10300 in tc_data, >pdd>!BblCpbBbBBBBBB
  PID:010300356001 TRM:000447007410 407777000460
  Running for 0.067825 (since 01:28:53).
  Usage: cpu 8:40.8; vcpu 6:04.6; pf 23622.
  te/s/i/x: 0.411 0.000 0.647 32.000.
  Flags: loaded,eligible,mbz11,dbr_loaded.
  Alarm in 31.244 (at 01:29:24).

   0  005400003000 014225000001 000000056106 010300356001
   4  000001443141 000000000000 000002356631 000172044000
  10  000000000000 000000000000 005200013740 000000002076
  14  003000777777 115641364232 000000000000 003702747134
  20  000000000000 001360000002 000000111567 457572337662
  24  000000000000 000000000000 011700111567 457761705103
  30  000447007410 407777000460 000000000000 000001720440
  34  000000111567 457736532472 003322000000 000000000000
  40  000000000000 000004325430 000000000000 000000000000
  44  035117540004 001775100023 000000000000 002556711030
  50  000000000000 000000000000 000000000004 000000004441
  54  000000111567 457572506651 000000000000 014274070015
  60  000000000000 002557051053 000001720440 776000000000
  64  000000000000 000000000000 000000000000 000000000000
  70  000000000000 000000000000 000000000000 000000000000
  74  000000000000 000000000000 000000000000 000000000000
```

unique chars gives PD

TOPIC X

Fault and Interrupt Handling

<u>FAULT AND INTERRUPT HANDLING OVERVIEW</u>

- FUNCTION

  - RESPONSIBLE FOR HANDLING ALL EXCEPTIONS IN A CPU WHETHER INTERNAL TO THE PROCESSOR (REFERRED TO AS <u>FAULTS</u>) OR EXTERNAL (REFERRED TO AS <u>INTERRUPTS</u>)

  - ESTABLISHES THE SUPERVISOR ENVIRONMENT AT FAULT AND INTERRUPT TIME. SAVES THE MACHINE CONDITIONS AND TRANSFERS TO THE APPROPRIATE HANDLER

  - MAJOR COMPONENTS: THE FAULT INTERCEPT MODULE (fim), WIRED-FAULT INTERCEPT MODULE (wired_fim), I/O INTERRUPT HANDLER (io_interrupt), sys_trouble, page_fault
    Store control unit

- MAJOR DATA BASES

  - INTERRUPT VECTORS - ONE SET PER SYSTEM (WIRED)

    - INTERRUPT PAIR (2 INSTRUCTIONS) - ONE PAIR PER DEFINED INTERRUPT TYPE

    - LOCATED AT ABSOLUTE ADDRESS 0. A HARDWARE RECOGNIZED DATA BASE

    - DESCRIBE WHERE TO SAVE THE CONTEXT, AND WHERE TO TRANSFER TO TO PROCESS THE INTERRUPT (ALWAYS io_interrupt)

FAULT AND INTERRUPT HANDLING OVERVIEW

- FAULT VECTORS - ONE SET PER SYSTEM (WIRED)
  - VECTOR PAIR (2 INSTRUCTIONS) - ONE PAIR PER DEFINED FAULT TYPE
  - LOCATED AT ABSOLUTE ADDRESS 100 (OCTAL) IMMEDIATELY ABOVE THE INTERRUPT VECTORS. A HARDWARE RECOGNIZED DATA BASE
  - DESCRIBE WHERE TO SAVE THE CONTEXT, AND WHERE TO TRANSFER TO TO PROCESS THE FAULT (fim, wired_fim, page_fault)
- <u>PROCESS</u> DATA SEGMENT (<u>PDS</u>) - ONE PER PROCESS (WIRED WHEN ELIGIBLE)
  - CONTAINS PROCESS RELEVANT INFO SUCH AS PROCESS ID, USER ID, HOME/WORKING/PROCESS DIRECTORIES, AIM CLASSIFICATION, INITIAL RING, ETC
  - CONTAINS ALL INFORMATION ABOUT THE PROCESS NEEDED BY THE SUPERVISOR CODE WHEN THE PROCESS IS RUNNING
  - CONTAINS SAVE AREAS FOR CONTEXT INFORMATION ABOUT FAULTS WHICH CAN RESULT IN GIVING UP THE PROCESSOR: PAGE FAULTS, SEGMENT FAULTS, AND ALL FAULTS NOT HANDLED BY THE SUPERVISOR
- <u>PROCESSOR</u> DATA SEGMENT (<u>PRDS</u>) - ONE PER CONFIGURED CPU (WIRED)
  - SERVES AS RING-ZERO STACK FOR PAGE CONTROL AND TRAFFIC CONTROL
  - ALSO CONTAINS SAVE AREAS FOR CONTEXT INFORMATION ABOUT FAULTS WHICH USUALLY DO NOT MEAN GIVING UP THE PROCESSOR: CONNECT FAULTS AND INTERRUPTS.

FAULT AND INTERRUPT HANDLING OVERVIEW

- FIM_TABLE

  - A TABLE IN THE FIM PROGRAM WHICH DESCRIBES THE ACTION TO BE TAKEN FOR VARIOUS TYPES OF FAULTS

FAULT AND INTERRUPT DATA BASES

FAULT AND INTERRUPT VECTORS

- ALL FAULTS AND INTERRUPTS ARE HANDLED IN A CENTRALIZED FASHION

  - FOR EACH FAULT OR INTERRUPT, THERE ARE TWO INSTRUCTIONS:

    - AN scu INSTRUCTION, TO STORE THE ABSOLUTELY ESSENTIAL DATA NEEDED TO RESTART FROM THE FAULT

    - A tra INSTRUCTION, TO TRANSFER TO TO THE APPROPRIATE FAULT HANDLER

  - ASSOCIATED WITH EACH OF THESE INSTRUCTIONS, THERE IS A POINTER

    - AN SCU DATA POINTER, POINTING TO ONE OF SIX REGIONS WHERE FAULT DATA GOES

    - A FAULT HANDLER POINTER, INDICATING ONE OF THE PROCEDURES USED TO HANDLE FAULTS

  - THESE INSTRUCTIONS AND POINTERS ARE STORED IN THE fault_vector

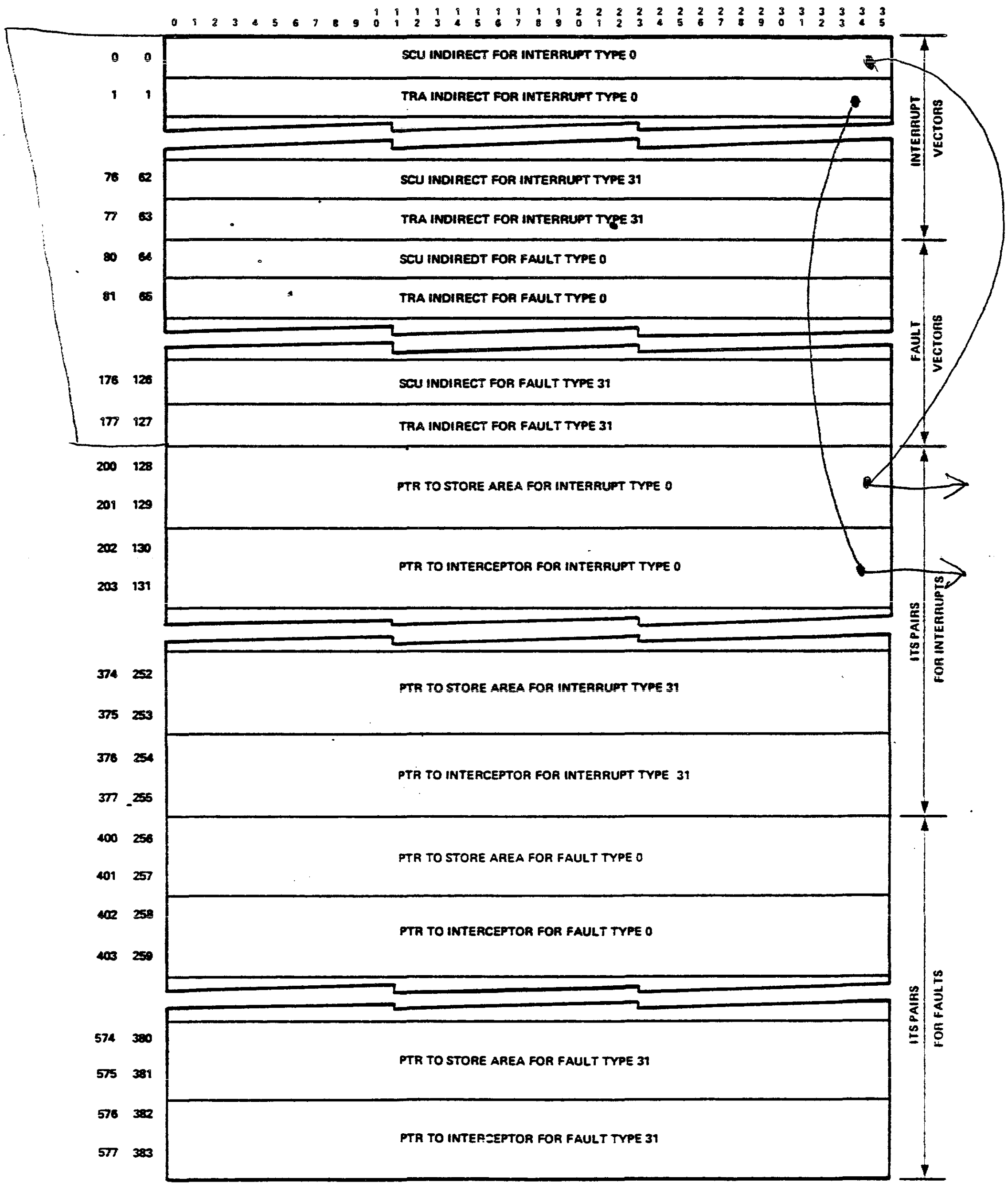
  - THE DATA STORED BY scu IS ONLY THE "CONTROL UNIT DATA" - EACH HANDLER MUST IMMEDIATELY SAVE THE PROGRAM VISIBLE REGISTERS, THE EIS POINTER & LENGTH DATA, AND SO FORTH. THIS DATA IS ALWAYS STORED IN THE SAME FORMAT, THE MACHINE CONDITIONS STRUCTURE (mc.incl.pl1)

ITS - indirect_TO_segment pointer

Appending mode
ABSolute mode

FAULT AND INTERRUPT DATA BASES

FAULT AND INTERRUPT VECTORS



FAULT AND INTERRUPT VECTORS

A HARDWARE RECOGNIZED DATA BASE LOCATED
AT ABSOLUTE ADDRESS 0-577

FAULT AND INTERRUPT DATA BASES

FAULT DATA SAVE AREAS

- THERE ARE SIX REGIONS WHERE MACHINE CONDITIONS ARE STORED.

  - THEY ARE SELECTED TO MINIMIZE THE NUMBER OF TIMES WHEN FAULT DATA MUST BE MOVED. USUALLY, ONCE FAULT DATA HAS BEEN STORED IN A PARTICULAR PLACE, IT CAN BE RESTORED DIRECTLY FROM THERE, BUT SOMETIMES IT MUST BE MOVED TO ANOTHER PLACE

    - prds$interrupt_data - USED FOR INTERRUPTS, ONLY

    - prds$fim_data - USED FOR FAULTS SUCH AS CONNECT FAULTS, WHICH WILL BE HANDLED ENTIRELY USING THE WIRED RING ZERO STACK (PRDS)

    - prds$sys_trouble_data - USED FOR THE FAULT THAT CRASHED THE SYSTEM. NO MACHINE CONDITIONS ARE EVER STORED HERE DIRECTLY, ONLY MOVED HERE.

    - pds$fim_data - USED FOR FAULTS WHICH WILL BE HANDLED IN RING ZERO, USING THE RING ZERO STACK, WHERE PAGE FAULTS MIGHT BE TAKEN WHILE THE OTHER FAULT IS BEING HANDLED

    - pds$page_fault_data - USED FOR PAGE FAULTS AND TIMER RUNOUTS (WHICH INDICATE THE END OF A QUANTUM) - BOTH ARE EVENTS WHICH ALMOST ALWAYS RESULT IN GIVING UP THE PROCESSOR, BUT WHICH ARE HANDLED ON THE WIRED RING ZERO STACK (PRDS)

    - pds$signal_data - USED FOR FAULTS WHICH WILL BE SIGNALLED OUT FOR THE USER RING TO HANDLE. IF AN ERROR OCCURS PROCESSING SOME FAULT IN RING ZERO, ITS FAULT DATA IS MOVED HERE BEFORE SIGNALLING

  - THERE ARE ALSO SPECIAL STACK FRAMES CREATED BY THE FIM USED FOR FAULT SIGNALLING, AND THE MACHINE CONDITIONS ARE COPIED THERE.

IMPORTANT TYPES OF FAULTS

- CERTAIN FAULTS DESERVE SPECIAL DISCUSSION, AS THEY ARE USED TO IMPLEMENT IMPORTANT SUPERVISOR SERVICES

  - LINKAGE FAULT

    - OCCUR WHEN A POINTER CONTAINING 46 OCTAL IN THE LOW SIX BITS OF THE FIRST WORD IS USED

    - USED TO IMPLEMENT DYNAMIC LINKING (SEE NAME/ADDRESS SPACE MANAGEMENT, TOPIC 4)

    - USES pds$fim_data, IS HANDLED BY fim.alm, WHICH INVOKES link_man.pl1, HANDLED ENTIRELY ON THE stack_0

  - SEGMENT FAULT

    - OCCURS WHEN AN NON-ACTIVE SEGMENT IS REFERENCED (SEE SEGMENT CONTROL, TOPIC 7)

    - USES pds$fim_data, IS HANDLED BY fim.alm, WHICH INVOKES seg_fault.pl1, HANDLED ENTIRELY ON THE stack_0

    - A SEGMENT FAULT MAY OCCUR WHILE ANOTHER IS BEING HANDLED, AND IT WILL BE HANDLED RECURSIVELY

  - PAGE FAULT

    - OCCURS WHEN A PAGE NOT IN MEMORY IS REFERENCED (SEE PAGE CONTROL, TOPIC 8)

IMPORTANT TYPES OF FAULTS

- USES pds$page_fault_data, IS HANDLED DIRECTLY BY page_fault.alm, AND IS HANDLED ENTIRELY ON THE PRDS.

- TIMER RUNOUT   Ring 0 ignores TRO   uses ring alarm register

  - OCCURS WHEN THE TIMER REGISTER IS DECREMENTED THROUGH ZERO, INDICATING THAT THE RUNNING PROCESS HAS NOW OVERSTAYED ITS WELCOME, AND SHOULD LOSE ELIGIBILITY

    - TIMER RUNOUT FAULTS ARE ALSO USED INTERNAL TO TRAFFIC CONTROL TO IMPLEMENT "PRE-EMPT" SAMPLING; IN THIS MODE, NOW THE DEFAULT, THE TIMER GOES OFF EVERY FEW MILLISECONDS AND THE PROCESS CHECKS TO SEE WHETHER A HIGHER PRIORITY PROCESS WANTS THE PROCESSOR; THIS DOES NOT MAKE THE RUNNING PROCESS INELIGIBLE, HOWEVER, UNLESS ITS QUANTUM HAS ALSO RUN OUT

    - A PROCESS RUNNING IN RING ZERO NEED NEVER GIVE UP ELIGIBILITY - WHEN A TIMER RUNOUT HAPPENS, IT REMEMBERS, AND SETS THE RING ALARM REGISTER, WHICH WILL CAUSE A RING ALARM FAULT LATER ON WHEN IT LEAVES RING ZERO
      Then makes un ineligible

  - USES pds$page_fault_data, IS HANDLED DIRECTLY BY pxss.alm, AND IS HANDLED ENTIRELY ON THE PRDS.

- RING ALARM FAULT

  - OCCURS WHEN A PROCESS RETURNS TO AN OUTER RING FROM AN INNER RING, AND THE RING ALARM REGISTER HAS BEEN SET

  - (1) USED TO DEFER ACTION ON TIMER RUNOUTS AND CONNECT FAULTS (SEE BELOW) UNTIL A PROCESS LEAVES RING ZERO

  - (2) USED TO ENSURE THAT THE SOFTWARE VALIDATION LEVEL (SET WITH cu_$level_set) IS NEVER LOWER THAN THE RING OF EXECUTION - SETTING THE VALIDATION LEVEL ALSO SETS THE RING ALARM REGISTER, AND THE RING ALARM FAULT CAUSES THE VALIDATION LEVEL TO BE RESET

FAULT AND INTERRUPT DATA BASES

IMPORTANT TYPES OF FAULTS

- USES pds$page_fault_data, IS HANDLED DIRECTLY BY ring_alarm.alm AND IS HANDLED ENTIRELY ON THE PRDS.

- IS ACTUALLY A SUBTYPE OF ACCESS VIOLATION

- CONNECT FAULT   Communication between CPUS usually to clear Associative memory when PTWS are changed. ie. page fault or seg fault

  - OCCURS WHEN ONE PROCESSOR SENDS A "connect" TO ANOTHER, USING A cioc INSTRUCTION

    - RESEMBLES A SOFTWARE SENDABLE INTERRUPT; ALTHOUGH PROCESSORS CAN SEND INTERRUPTS TO EACH OTHER, LIMITATIONS OF THE HARDWARE MAKE CONNECT FAULTS EASIER TO USE

  - USED FOR ALL INTERPROCESSOR SIGNALLING -

    - (1) TO CAUSE ANOTHER PROCESSOR TO SELECTIVELY CLEAR ITS CACHE OR ASSOCIATIVE MEMORY

    - (2) TO PRE-EMPT A PROCESS RUNNING ON ANOTHER CPU (WHEN PRE-EMPT SAMPLING IS NOT IN USE)

    - (3) TO INFORM ANOTHER PROCESSOR THAT THE SYSTEM IS CRASHING

    - (4) TO INFORM ANOTHER PROCESSOR THAT IT IS BEING REMOVED FROM THE CONFIGURATION

  - USES prds$fim_data, IS SOMETIMES HANDLED BY wired_fim.alm, AND IS HANDLED ENTIRELY ON THE PRDS.

    - FOR TYPE 1 CONNECTS (CACHE CLEAR), THE FAULT IS HANDLED VERY SPECIALLY BY CODE WHICH IS ACTUALLY EXECUTED FROM WITHIN THE PRDS (SEE fast_connect_init.alm), AND RUNS SOMEWHAT FASTER. THE COMPLICATED CASES ARE LEFT TO WIRED_FIM

  System Communication Segment for CPU Communication using connect fault

  - CELLS IN THE scs ARE USED TO DISTINGUISH BETWEEN THE DIFFERENT TYPES OF CONNECT FAULTS; A PROCESSOR SETS THE APPROPRIATE CELLS BEFORE SENDING THE CONNECT.

IMPORTANT TYPES OF FAULTS

- OTHER FAULTS

  - PARITY - RUN AUTOMATIC PARITY ERROR LOGGING AND DIAGNOSIS (TO CHIP LEVEL, FOR CACHE) ROUTINES

  - OP_NOT_COMPLETE, COMMAND, SHUTDOWN, STARTUP, STORE, TROUBLE - RUN HARDWARE ERROR LOGGING ROUTINES

  - OVERFLOW, UNDERFLOW - CAN BE SET UP TO AUTOMATICALLY SET A SPECIFIED (VERY LARGE OR VERY SMALL) VALUE AND RESTART WITHOUT INTERRUPTING THE RUNNING PROGRAM

  - DERAIL - USED WHEN CRASHING THE SYSTEM OR VOLUNTARILY RETURNING TO BOS TO BEGIN EXECUTION IN BOS. AN ORDINARY SIGNALLABLE FAULT AT ALL OTHER TIMES (AND USED THAT WAY BY THE gtss EMULATOR). other systems used to have supervisor do something. ie. ~~meme 1~~ MME1 fault

  - EXECUTE - USED TO FORCE A SYSTEM CRASH

  - ACCESS VIOLATION - CAN AUTOMATICALLY LOG ACCESS VIOLATIONS FOR SECURITY AUDITS

  - OTHERS - HANDLED BY fim.alm, WHICH MAPS THE HARDWARE FAULTS ONTO THE MULTICS ENVIRONMENT CONDITION NAMES

    - CONSIDERABLE INTERPRETATION IS SOMETIMES REQUIRED; FOR INSTANCE, A NULL POINTER IS ACTUALLY SIGNALLED BY THE HARDWARE AS AN ACCESS VIOLATION, OUT OF BOUNDS ON DSEG, FAULT

- INTERRUPT

  - INTERRUPTS ARE USED TO ANNOUNCE THE COMPLETION OF ALL I/O OPERATIONS, AND ALSO (RARELY) USED DURING DYNAMIC RECONFIGURATION TO START PROCESSORS

IMPORTANT TYPES OF FAULTS

- USES prds$interrupt_data, IS HANDLED DIRECTLY BY iom_interrupt.alm OR init_processor.alm, AND IS HANDLED ENTIRELY ON THE PRDS.

FAULT/INTERRUPT METERS

fim meters

⊠ FIM_METERS - COUNTS FOR ALL FAULT PROCESSING

Total metering time: 0:20:28

fault type	count
shutdown	0
store	0
mme1	0
fault_tag_1	2
timer_runout	12907
command	5
derail	0
lockup	0
connect	742501
parity	0
illegal_procedure	0
op_not_complete	0
startup	0
overflow	0
divide_check	0
execute	0
segment_fault	6697
page_fault	151772
directed_fault_2	0
directed_fault_3	0
access_violation	21120
mme2	0
mme3	0
mme4	0
linkage_fault	21058
fault_tag_3	0
trouble	0

1 CPU - uses connect faults ~~or~~ lightly loaded use idle processes.

* INTERRUPT_METERS - COUNTS & TIMING FOR ALL I/O INTERRUPTS

Total metering time 0:20:28

IOM	Ch	Int	Avg Time	% CPU	Name
A	6.	11	2.043	0.00	IOM A special
A	10.	379	1.553	0.01	prtb
A	13.	9253	3.436	0.65	fnp c
A	14.	19	0.788	0.00	opc
A	16.	26248	1.347	0.72	tapa
A	17.	128	1.328	0.00	tapa
A	18.	13097	5.383	1.43	fnp b
A	20.	21041	0.860	0.37	dska
A	21.	2881	0.860	0.05	dska
A	22.	8486	0.877	0.15	dskb
A	23.	257	0.964	0.01	dskb
A	24.	17896	0.895	0.33	dskb
A	25.	2573	0.943	0.05	dskb
A	26.	10486	0.857	0.18	dska
A	27.	255	0.899	0.00	dska
A	28.	24717	0.845	0.43	dskc
A	29.	2319	0.849	0.04	dskc
A	30.	9915	0.845	0.17	dskc
A	31.	215	0.767	0.00	dskc
B	14.	236	0.877	0.00	fnp d
B	15.	6917	3.351	0.47	fnp f
B	18.	3547	3.543	0.26	fnp a
B	19.	13667	3.435	0.96	fnp e
B	20.	22013	0.881	0.39	dskb
B	21.	4745	0.863	0.08	dskb
B	22.	6054	0.861	0.11	dska
B	23.	60	0.925	0.00	dska
B	24.	15946	0.867	0.28	dska
B	25.	989	0.841	0.02	dska
B	26.	12757	0.863	0.22	dskb
B	27.	1020	0.903	0.02	dskb
B	28.	5178	0.838	0.09	dskc
B	29.	30	0.653	0.00	dskc
B	30.	16582	0.823	0.28	dskc
B	31.	825	0.841	0.01	dskc
B	32.	1764	0.825	0.03	dske
Chan		262506	1.464	7.82	
Ovhd		258932	0.230	1.21	
Total		258932	1.713	9.03	

TOPIC XI

System Initialization/Shutdown

SYSTEM INITIALIZATION OVERVIEW

- FUNCTION

  - PREPARE THE SYSTEM TO OPERATE, STARTING FROM A COMPLETELY EMPTY MACHINE

  - READS IN SUPERVISOR PROGRAMS FROM SYSTEM TAPE, SNAPS LINKS BETWEEN SUPERVISOR COMPONENTS, VERIFIES AND INITIALIZES HARDWARE CONFIGURATION, SETS UP SYSTEM DATABASES, ACCEPTS STORAGE SYSTEM DISKS AND PREPARES THEM FOR USE BY THE FILE SYSTEM

  - MOST PROGRAMS IN SYSTEM INITIALIZATION ARE DELETED AFTER INITIALIZATION IS COMPLETE.

  - SUPERVISOR PROGRAMS ARE LOADED IN THREE "COLLECTIONS", EACH OF WHICH DEPENDS ON THE MECHANISMS SET UP BY THE PREVIOUS ONE

- MAJOR DATA BASES

  - THESE DATA BASES ARE ALL BUILT DURING THE PROCESS OF INITIALIZATION (EXCEPT FOR THE CONFIG DECK) AND KEPT AFTER INITIALIZATION IS FINISHED

  - SEGMENT LOADING TABLE (>sl1>slt)

    - CONTAINS AN ENTRY DESCRIBING THE ATTRIBUTES OF EACH SEGMENT IN THE SUPERVISOR

SYSTEM INITIALIZATION OVERVIEW

- NAME TABLE (>sl1>name_table)

  - CONTAINS A LIST OF NAMES FOR EACH OF THE SEGMENTS IN THE SUPERVISOR

- DEFINITIONS SEGMENT (>sl1>definitions_)

  - CONTAINS THE DEFINITIONS SECTIONS FOR ALL THE SEGMENTS IN THE SUPERVISOR, WHICH ARE USED IN ORDER TO SNAP LINKS BETWEEN THE SUPERVISOR MODULES

- CONFIG DECK (>sl1>config_deck)

  - CONTAINS A DESCRIPTION OF THE HARDWARE CONFIGURATION AND CERTAIN SOFTWARE PARAMETERS

  - PROVIDED TO SYSTEM INITIALIZATION BY BOS

- SHUTDOWN -- TERMINATES THE ACTIVITIES OF THE SYSTEM IN AN ORDERLY FASHION

  - TWO TYPES OF SHUTDOWN:

    - NORMAL -- REQUESTED BY THE INITIALIZER, RUNS IN THE USUAL SUPERVISOR ENVIRONMENT

    - EMERGENCY -- USED AFTER A CRASH, MUST MAKE THE SUPERVISOR ENVIRONMENT OPERABLE BEFORE PROCEEDING

SYSTEM INITIALIZATION OVERVIEW

- BOTH TYPES EXIST PRIMARILY TO SHUT DOWN THE FILE SYSTEM -- THAT IS, TO WRITE ALL DATA IN MEMORY INTO ITS PROPER HOME ON DISK

  - INCLUDES PAGES OF SEGMENTS, VTOCES, VOLUME AND VTOC MAPS

- SHUTDOWN ESSENTIALLY RUNS THE STEPS OF INITIALIZATION BACKWARDS, BUT WITH A LOT OF SHORTCUTS

SYSTEM INITIALIZATION TERMINOLOGY

CONFIG DECK: A SET OF CARDS OR CARD IMAGES USED TO INFORM THE SOFTWARE ABOUT THE OPERATIONAL READINESS OF THE HARDWARE PRESENT, SWITCH SETTINGS AND SPECIFICATIONS OF SOME SOFTWARE DATA BASES (SIZE, LOCATION, ETC)

BOS: THE BOOTLOAD OPERATING SYSTEM. A SIMPLE OPERATING SYSTEM (A ONE CPU, UNPAGED ENVIRONMENT) OF 8 SEGMENTS OCCUPYING THE FIRST 16K OF MAIN MEMORY. BOS RESIDES IN THE BOS PARTITION WHEN "MULTICS" IS RUNNING

MST: THE MULTICS SYSTEM TAPE CONTAINS PRECISELY ENOUGH INFORMATION (PROCEDURES AND DATA BASES) TO BRING A BARE HARDWARE SYSTEM TO MULTICS COMMAND LEVEL (ACTUALLY, ONLY ENOUGH OF COMMAND LEVEL TO PERFORM A RELOAD)

BOOT: THE OPERATIONAL PROCEDURE OF READING THE SEGMENTS FROM THE MST AND EXECUTING THE PROCEDURE/SEGMENTS THEREIN

WARM/COLD BOOT: BOOTING THE SYSTEM WITH/WITHOUT A HIERARCHY PRESENT

HARDCORE PARTITION: A RLV PARTITION FOR PAGING HARDCORE SEGMENTS CREATED DURING BOOTLOAD

SYSTEM INITIALIZATION TERMINOLOGY

DECIDUOUS SEGMENT:

A SEGMENT READ IN AS PART OF THE BOOTLOAD TAPE AND PLACED INTO THE HIERARCHY. DECIDUOUS SEGMENTS ARE PART OF THE INITIALIZER'S HARDCORE ADDRESS SPACE AND RESIDE ENTIRELY IN THE HARDCORE PARTITION. THEY ARE PUT INTO THE HIERARCHY (>sl1) IN ORDER TO BE ACCESSIBLE FROM THE USER RINGS

NON DECIDUOUS HARDCORE SEGMENT:

A PAGED HARDCORE SEGMENT NOT IN THE HIERARCHY (AND THUS HAS NO PATHNAME)

SEGMENT CATEGORIES

ALL OTHER SEGMENTS		ORDINARY SEGMENTS: > sss > bound_pl1_ > udd > MED > Kaiser > Kaiser.mbx > udd > Site Sa > Homan > test_pl1 [pd] > stack_4	NON-SUPERVISOR	HIERARCHY
ALL OTHER SEGMENTS		DIRECTORY SEGMENTS: > udd > udd > Site Sa > sss > udd > MED	NON-SUPERVISOR	HIERARCHY
ALL OTHER SEGMENTS		REVERSE-DECIDUOUS: (per-process, per-bootload, ring 0 only) [pd] > dseg > sl1 > stack_0.nnn [pd] > kst [pd] > pds	NON-SUPERVISOR	HIERARCHY
MST SEGMENTS	COLLECTION # 3	NON-SUPERVISOR PROGRAMS: > sl1 > bound_command_loop_ > sl1 > bound_rcp_	NON-SUPERVISOR	HIERARCHY
MST SEGMENTS	COLLECTIONS # 0, # 1, AND # 2	DECIDUOUS: > sl1 > bound_sss_wired_ > sl1 > sys_info > sl1 > hcs_ (all ring 0 gates)	PERMANENT SUPERVISOR AND INIT-SEGS	HIERARCHY
MST SEGMENTS	COLLECTIONS # 0, # 1, AND # 2	DISK OVERLAY ABS-SEGS: volmap_abs_seg syserr_log	PERMANENT SUPERVISOR AND INIT-SEGS	NON-HIERARCHY
MST SEGMENTS	COLLECTIONS # 0, # 1, AND # 2	NON-DECIDUOUS PAGED: bound_file_system bound_system_faults str_seg	PERMANENT SUPERVISOR AND INIT-SEGS	NON-HIERARCHY
MST SEGMENTS	COLLECTIONS # 0, # 1, AND # 2	UNPAGED: sst_seg tc_data pxss bound_page_cotnrol bound_to_wired	PERMANENT SUPERVISOR AND INIT-SEGS	NON-HIERARCHY

INITIALIZATION DATA BASES

- THE TERM INITIALIZATION REFERS TO THE ACTIONS REQUIRED TO CREATE THE MULTICS ENVIRONMENT GIVEN THE EXISTENCE OF A CONFIGURATION DECK, AND HARDWARE CONTAINING NO OTHER DATA THAN FIRMWARE AND BOS

- INITIALIZATION IS ACCOMPLISHED BY AN ORDERLY LOADING AND PROCESSING OF THE SEGMENTS RESIDING ON THE MULTICS SYSTEM TAPE (MST)

- THE SEGMENTS OF THE MST MAY BE DIVIDED INTO THREE CATEGORIES:

  - INITIALIZATION SEGMENTS

    - PROCEDURES USED ONLY FOR INITIALIZATION AND SUBSEQUENTLY DISCARDED

  - SUPERVISOR SEGMENTS

    - DATA BASES USED DURING INITIALIZATION THAT ULTIMATELY BECOME DATA BASES OF Initializer.SysDaemon.z

    - PROCEDURES AND DATA BASES THAT CONSTITUTE MULTICS HARDCORE SUPERVISOR IN ITS ENTIRETY

  - NON-SUPERVISOR SEGMENTS

    - THE SEGMENTS OF COLLECTION THREE ARE PRECISELY THE NON-SUPERVISOR SEGMENTS OF THE MST. THESE SEGMENTS ARE LOADED DIRECTLY INTO >system_library_1, AND ARE NOT PART OF THE RING ZERO SUPERVISOR

INITIALIZATION DATA BASES

- BOOTLOAD PROCESSOR

  - ONE PROCESSOR (THE BOOTLOAD CPU) PERFORMS ALL OF INITIALIZATION RUNNING EXCLUSIVELY IN RING ZERO

  - IN THE MOST OF INITIALIZATION (COLLECTION ONE AND MOST OF COLLECTION TWO), THERE ARE NO PROCESSES, AS SUCH. THE ENVIRONMENT WHICH RUNS THERE EVENTUALLY BECOMES THE Initializer.SysDaemon PROCESS

  - NOTE: SINCE THE Initializer.SysDaemon DOES NOT LOGIN LIKE OTHER USERS, IT DOES NOT APPEAR IN THE NORMAL USER TABLE, CONSEQUENTLY IS NOT VISIBLE TO THE who COMMAND

- STRATEGY OF INITIALIZATION: BOOTSTRAPPING

  - THE FIRST PROCEDURES RUN IN AN ENVIRONMENT DEVOID OF ALL SOFTWARE ASSISTANCE

  - EACH NEW MECHANISM (SEGMENTATION, STACKS, SYMBOLIC LINKING, PAGING, ETC) IS MADE OPERATIVE AS SOON AS POSSIBLE TO ENRICH THE ENVIRONMENT IN WHICH FURTHER MECHANISMS ARE MADE OPERATIVE

  - MANY MECHANISMS HAVE SUBSYSTEMS THAT CONTROL THEM AND THESE SUBSYSTEMS ARE NORMALLY INITIALIZED BY CALLING A SPECIAL ENTRY POINT IN THE SUBSYSTEM WHICH PERFORM SUCH TASKS AS:

    - CREATING TABLES WHOSE SIZES ARE DETERMINED BY DATA SPECIFIED IN THE "CONFIG" DECK

    - THREADING OF RELEVANT LISTS

INITIALIZATION DATA BASES

- SEGMENTS ON THE MST ARE ARRANGED IN SUCH AN ORDER THAT THE EARLIER SEGMENTS ALLOW AS MANY MECHANISMS AS POSSIBLE TO BE USED IN LOADING AND PROCESSING OF THE LATER SEGMENTS

- FOR THIS PURPOSE (AND BECAUSE THE SIZE OF THE MST IS POTENTIALLY LARGER THAN MAIN MEMORY), THE MST IS DIVIDED INTO FOUR PARTS KNOWN AS COLLECTION ZERO, ONE, TWO AND THREE

- INITIALIZATION CAN BE VIEWED AS THE LOADING AND PROCESSING OF COLLECTION ONE, COLLECTION TWO, AND COLLECTION THREE, IN TURN

- THE ADDRESS SPACE OF INITIALIZATION (MINUS THE INITIALIZATION SEGMENTS) BECOMES THE GLOBAL SUPERVISOR ADDRESS SPACE OF MULTICS

  - THIS ADDRESS SPACE IS "CLONED" TO BECOME THE INITIAL ADDRESS SPACE OF NEWLY CREATED PROCESSES BY DUPLICATING THE DSEG

ENVIRONMENT PASSED TO INITIALIZATION

- THE FIRST SEGMENT OF COLLECTION ONE IS THE BOUND SEGMENT bound_bootload_1

- AT THE TIME CONTROL IS TRANSFERRED TO bound_bootload_1, IT IS REQUIRED THAT BOS HAS INITIALIZED MAIN MEMORY AS FOLLOWS:

  - THE IOM MAILBOX MUST BE AT LOCATION 1400 AND CONTAIN THE CHANNEL AND DEVICE NUMBER OF THE TAPE DRIVE ON WHICH THE MST IS MOUNTED

  - THE CONFIG DECK (AS PRODUCED BY BOS) MUST RESIDE AT LOCATION 14000 THRU 15777

  - THE BOS TOEHOLD AND FLAGBOX MUST BE AT LOCATION 10000 THRU 11777

    - THIS ONE PAGE CONSTITUTES ALL KNOWLEDGE THAT MULTICS HAS OF BOS

    - TRANSFERRING CONTROL TO THE START OF THE BOS TOEHOLD WILL CAUSE:

      - FIRST 64K OF MAIN MEMORY TO BE FLUSHED OUT TO THE BOS PARTITION

      - THE BOS OPERATING SYSTEM TO BE READ INTO THE FIRST 64K OF MAIN MEMORY

      - CONTROL GIVEN TO BOS

  - THE REMAINDER OF MAIN MEMORY MUST CONTAIN ZEROES

COLLECTION 0

- THIS CONSISTS OF THE SEGMENTS WHICH ARE DEFINED TO BE PRESENT IN THE INITIAL ADDRESS SPACE

  - THESE SEGMENTS ARE EMPTY, AND OVERLAY SPECIFIC REGIONS OF MAIN MEMORY. THEY ARE DEFINED SO THAT bound_bootload_1 CAN KNOW WHAT SEGMENT NUMBERS TO USE FOR WHAT DATA

  - THE COLLECTION ZERO SEGMENTS ARE dseg, fault_vector, iom_mailbox, config_deck, dn355_mailbox, bos_toehold, flagbox, slt, and name_table

    - SEE SECTION 5, NAME / ADDRESS SPACE MANAGEMENT, FOR THEIR DESCRIPTIONS

COLLECTION 1

- COLLECTION 1 CONTAINS ALL OF THE PROCEDURES AND DATA BASES NECESSARY TO MAKE PAGING OPERATIVE

- BASIC STEPS OF COLLECTION 1 LOADING AND INITIALIZATION:

  - bound_bootload_1 GAINS CONTROL FROM BOS, IN ABSOLUTE MODE, AND PERFORMS THE FOLLOWING:

    - LOADS THE REMAINDER OF ITSELF INTO MAIN MEMORY

    - ESTABLISHES INTERIM FAULT AND INTERRUPT VECTORS

    - INITIALIZES THE INITIALIZATION DSEG, AND ENTERS APPENDING MODE

    - READS THE REMAINDER OF COLLECTION 1 INTO MAIN MEMORY (INCLUDING A SEGMENT NAMED bootstrap2)

    - bound_bootload_1 TRANSFERS TO bootstrap2

  - bootstrap2 PERFORMS THE FOLLOWING:

    - CREATES A STACK FRAME IN THE SEGMENT "inzr_stk0"

    - CALL THE APPROPRIATE PROCEDURES TO PRELINK THE SEGMENTS OF COLLECTION 1

    - SETS UP THE PL/I ENVIRONMENT AND CALLS THE FIRST PL/I PROCEDURE "initializer"

COLLECTION 1

- initializer (A SUPERVISOR SEGMENT), ACTUALLY CALLS real_initializer TO DO THE REAL WORK. ALL THE REST OF INITIALIZATION TAKES PLACE VIA CALLS IN real_initializer
  - FOR DEBUGGING PURPOSES, THERE IS A MECHANISM IN real_initializer WHICH CAN BE USED TO STOP AT ANY OF THOSE CALLS BY SETTING A VALUE IN THE PROCESSOR SWITCHES

- THE MAJOR INITIALIZATIONS PERFORMED IN THE REST OF COLLECTION ONE ARE:
  - INITIALIZE THE SCU, CLOCK, AND CPU CONTROL MECHANISMS, CHECKING THE SWITCHES AND THE ADDRESSABILITY OF MEMORY
  - INITIALIZE FAULT AND INTERRUPT PROCESSING. INITIALIZE THE CONSOLE AND SYSERR MECHANISMS
  - INITIALIZE PRIMITIVE TRAFFIC CONTROL (WAIT FOR SINGLE EVENTS). INITIALIZE THE SST
  - INITIALIZE AND CHECK THE DISK CONFIGURATION. AT THIS POINT, IT BECOMES POSSIBLE TO TAKE PAGE FAULTS, AND ALL FURTHER DISK I/O IS DONE BY PAGING
  - CHECK THE ROOT VOLUMES SPECIFIED ON THE ROOT CARD, AND INITIALIZE THE HARDCORE PARTITION MECHANISM. THIS ALSO INCLUDES CREATING THE PARTITIONS AND VTOC IN A COLD BOOT
  - MAKE SEGMENTS PAGED -- AT THIS POINT, ALL PAGED SUPERVISOR SEGMENTS ARE COPIED INTO THE HARDCORE PARTITION, AND ACCESSED BY THE NORMAL PAGE FAULT MECHANISM. COLLECTION ONE ENDS HERE

- COLLECTION TWO - THE REST OF SUPERVISOR INITIALIZATION

  - READ IN THE REST OF THE SUPERVISOR SEGMENTS FROM TAPE AND INITIALIZE THE REST OF THE SUPERVISOR DATABASES

  - WHEN COLLECTION TWO IS FINISHED, THE INITIALIZATION ENVIRONMENT HAS BECOME Initializer PROCESS, AND IT CALLS OUT TO RING ONE TO START UP THE ANSWERING SERVICE

  - THE FOLLOWING MAJOR STAGES TAKE PLACE IN COLLECTION TWO:

    - COLLECTION TWO IS READ FROM TAPE (STILL USING A SPECIAL PROCEDURE, tape_reader, WHICH DOUBLE-BUFFERS)

    - THE AND CONDITION SIGNALLING AND HIGHER LEVEL FAULT MECHANISMS ARE INITIALIZED

    - HIGH LEVEL FILE SYSTEM MECHANISMS ARE INITIALIZED: THE VTOC MANAGER, VOLUME DUMPER BIT MAP, SCAVENGER, SEGMENT TRAILERS, LOGICAL VOLUME MANAGEMENT; DIRECTORY LOCKING

    - TRAFFIC CONTROL IS FURTHER INITIALIZED, AND AN IDLE PROCESS IS CREATED FOR THE RUNNING CPU

    - SYSERR LOGGING (TO THE SYSERR PARTITION) IS INITIALIZED. ALL SYSERR MESSAGES GENERATED AFTER THIS POINT WILL GO IN THE LOG

    - THE FILE SYSTEM ON THE RPV IS "ACCEPTED", AND THE ROOT DIRECTORY INSPECTED (OR CREATED, IF THIS IS A COLD BOOT)

      - AFTER THIS POINT, THE CONTENTS OF THE HARDCORE PARTITIONS ARE FIXED, AND ALL FURTHER RECORD ALLOCATION AND FREEING IS DONE FROM THE PAGING REGIONS OF THE FILE SYSTEM VOLUMES

COLLECTION 2

- DECIDUOUS SEGMENTS ARE SPLICED INTO THE HIERARCHY

- STACK SHARING IS INITIALIZED

- THE REST OF THE SEGMENTS, COMPRISING COLLECTION THREE, ARE READ, DIRECTLY INTO THEIR PLACES IN THE HIERARCHY (>sl1)

- THE USER VISIBLE SUPERVISOR I/O MECHANISM (ioi) IS INITIALIZED, AND THE SPECIAL SUPERVISOR TAPE READER IS SHUT DOWN

- TRAFFIC CONTROL IS FULLY INITIALIZED BY CREATING IDLE PROCESSES FOR THE OTHER CPUS AND STARTING THEM

- ALL INITIALIZATION SEGMENTS ARE NOW DISCARDED, AND WHAT HAS BECOME THE Initializer.SysDaemon PROCESS CALLS init_proc AND BEGINS EXECUTION IN RING ONE

- AT THIS POINT, SUPERVISOR INITIALIZATION IS FINISHED

- COLLECTION THREE IS NOT PROPERLY A "COLLECTION" AT ALL, BUT JUST A TERM USED TO DESCRIBE THE BEGINNING OF Initializer AND ANSWERING SERVICE INITIALIZATION

  - MANY THINGS HAPPEN BETWEEN FIRST LEAVING RING ZERO AND LOGGING IN THE FIRST USER. MOST HAVE NOTHING TO DO WITH THE FILE SYSTEM. THE FEW MOST INTERESTING ONES ARE:

    - VOLUME ACCEPTANCE - THE NON-ROOT VOLUMES ARE INSPECTED AND ACCEPTED INTO THE HIERARCHY. THEIR CONFIGURATION CAN BE CHANGED BY USE OF OPERATOR COMMANDS

    - delete_old_pdds - THE PROCESS DIRECTORY AND DECIDUOUS SEGMENTS LEFT OVER FROM THE PREVIOUS BOOTLOAD(S) ARE DELETED

      - THIS MUST BE DONE AFTER THE HIERARCHY IS FULLY AVAILABLE, SINCE NON-RLV VOLUMES MAY HAVE BEEN USED FOR PROCESS DIRS

    - VOLUME SCAVENGER AND QUOTA SALVAGER PROCESSES MAY BE STARTED AT THIS TIME, IF THERE ARE INCONSISTENCIES IN THE HIERARCHY CAUSED BY PREVIOUS CRASHES

NORMAL SHUTDOWN

- WHEN THE SYSTEM IS SHUT DOWN, IT MUST BE DONE IN AN ORDERLY MANNER

  - THIS IS ACCOMPLISHED, MORE OR LESS, BY RUNNING THE STEPS IN INITIALIZATION BACKWARDS:

    - CRAWLOUTS ARE DISABLED. ONCE SHUTDOWN BEGINS, IT CAN'T BE STOPPED

    - THE Initializer SWITCHES TO RUNNING ON THE BOOTLOAD CPU, TRAFFIC CONTROL IS DISABLED, AND THE OTHER CPUS ARE STOPPED AND DELETED.

      - LOCKING IS DISABLED AT THIS POINT, SINCE THERE IS NOW ONLY ONE PROCESS AND ONE PROCESSOR RUNNING

    - ALL THE DISK DRIVES ARE EXERCISED, TO DETERMINE IF ANY ARE BROKEN AND CANNOT BE SHUT DOWN

    - ANY VOLUME SCAVENGES IN PROGRESS ARE STOPPED AND ABANDONED

  - AT THIS POINT, NORMAL SHUTDOWN IS READY TO SHUT DOWN THE FILE SYSTEM, AND ALL THE NORMAL MECHANISMS ARE ASSUMED TO BE OPERATING.

  - THE Initializer SWITCHES TO inzr_stk0 (WHERE IT ALL STARTED) AND CALLS shutdown_file_system

FILE SYSTEM SHUTDOWN

- FILE SYSTEM SHUTDOWN CONSISTS OF FORCING ALL DATA OUT OF MEMORY TO ITS HOME ON DISK:

  - ALL PAGES ARE WRITTEN

  - SEGMENTS ARE DEACTIVATED, AND THEIR VTOCES UPDATED

  - VOLUMES ARE DEMOUNTED, AND THEIR LABELS UPDATED

  - FILE SYSTEM SHUTDOWN TAKES THE FOLLOWING STEPS:

    - ALL MODFIED PAGES ARE WRITTEN TO DISK. THIS IS DONE SEVERAL TIMES DURING THE COURSE OF SHUTDOWN

    - STACK_0 SEGMENTS ARE DEACTIVATED AND DISCARDED

    - THE deactivate_for_demount PROCEDURE IS CALLED TO DEACTIVATE ALL OTHER SEGMENTS AND UPDATE THEIR VTOCES

      - THIS IS DONE BY WALKING THE AST HIERARCHY FROM THE BOTTOM UP, DEACTIVATING A SEGMENT, ITS SIBLINGS, AND ITS PARENTS, ETC.

      - THIS IS DONE TO ENSURE CONSISTENT QUOTA VALUES IN VTOCES AFTER SHUTDOWN, BECAUSE QUOTA MUST BE UPDATED FROM THE BOTTOM UP

    - ALL VOLUMES ARE DEMOUNTED, THEIR VOUME AND VTOC MAPS UPDATED, AND LABELS CHANGED TO INDICATE SUCCESSFUL DEMOUNT

      - THE ORDER IS NOT IMPORTANT, EXCEPT THAT THE RPV GOES LAST

<u>FILE SYSTEM SHUTDOWN</u>

- MEMORY IS FLUSHED AGAIN
- IF ANY DRIVES WERE INOPERATIVE, THIS IS ANNOUNCED, AND THE RPV IS <u>NOT</u> DEMOUNTED
  - THIS MAKES IT POSSIBLE TO FIX THE BROKEN DRIVE AND DO AN EMERGENCY SHUTDOWN TO FINISH SHUTDOWN
- IF THERE WERE NO PROBLEMS, THE RPV IS DEMOUNTED AND MEMORY IS FLUSHED ONE LAST TIME. ALL RELEVANT INFORMATION IS .NOT ON DISK

- AT THE END OF FILE SYSTEM SHUTDOWN, ALL CONSOLE MESSAGES ARE ALLOWED TO COMPLETE, AND THE SYSTEM RETURNS TO BOS

EMERGENCY SHUTDOWN

- EMERGENCY SHUTDOWN IS DONE AFTER A CRASH OR SHUTDOWN FAILURE

  - LIKE NORMAL SHUTDOWN, THE MISSION OF ESD IS TO SHUT DOWN THE FILE SYSTEM, AND IT DOES THIS BY MAKING THE SYSTEM WORK WELL ENOUGH TO CALL shutdown_file_system

  - UNLIKE NORMAL SHUTDOWN, IT CANNOT ASSUME THAT NORMAL MECHANISMS ARE OPERATIONAL, AND MUST MAKE THEM WORK FIRST

  - AFTER THE SUPERVISOR IS MADE OPERATIONAL, EMERGENCY SHUTDOWN TRANSFERS TO THE NORMAL FILE SYSTEM SHUTDOWN

  - THE FOLLOWING STEPS ARE TAKEN TO REANIMATE THE SUPERVISOR:

    - EMERGENCY SHUTDOWN STARTS OUT RUNNING IN ABSOLUTE MODE

    - IT ENTERS APPENDING MODE, FINDS THE PRDS FOR THE PROCESSOR IT IS RUNNING ON, SETS IT UP AS ITS STACK

    - ALL CRITICAL LOCKS (PAGE TABLE, APT) ARE FORCIBLY UNLOCKED. TRAFFIC CONTROL IS DISABLED. THE PROCESSOR RUNNING IS NOW THE ONLY ONE, AND LOCKS ARE UNNECESSARY

    - THE CONSOLE AND SYSERR MECHANISMS ARE RESET

    - SUPERVISOR I/O SUPPORT, IN PARTICULAR THE DISK DIM, IS REINITIALIZED

    - THE STATE OF PAGE TABLES AND THE CORE MAP IS MADE CONSISTENT

      - THIS IS DONE BY pc_recover_sst AND CAN BE DONE ONLY BECAUSE ALL OF PAGE CONTROL IS CODED TO FOLLOW PROTOCOLS ABOUT THE ORDER IN WHICH TO UPDATE RELATED DATA

- BECAUSE OF THIS, IF PAGE CONTROL IS INTERRUPTED AT ANY POINT, IT IS POSSIBLE TO DETERMINE WHAT IT WAS DOING AND COMPLETE THE OPERATION

- THIS STEP IS CRUCIAL TO BEING ABLE TO TAKE PAGE FAULTS LATER IN SHUTDOWN

- BECAUSE THERE IS NO GUARANTEE THAT THE PROCESS WHICH CRASHED THE SYSTEM (AND THUS, THE ADDRESS SPACE WHERE ESD IS RUNNING) IS NOT DEFECTIVE, ESD REBUILDS ITS PDS FROM THE template_pds

- ESD SWITCHES TO THE inzr_stk0 AND CALLS wired_shutdown

  - ALL THE ABOVE STEPS WERE DONE IN ALM. wired_shutdown IS A PL/I PROCEDURE

- THE VTOC BUFFER IS CHECKED TO SEE WHETHER ANY OPERATIONS WERE IN PROGRESS. IF SO, THOSE VOLUMES ARE MARKED (IN THE PVT) TO INDICATE THAT THEY MAY HAVE INCONSISTENCIES

- MEMORY IS FLUSHED

- THE VTOC MANAGER IS REINITIALIZED

- AT THIS POINT, THE SUPERVISOR SHOULD BE WORKING WELL ENOUGH TO RUN shutdown_file_system. FROM NOW ON, EMERGENCY SHUTDOWN FOLLOWS THE SAME PATH AS NORMAL SHUTDOWN

- IF EMERGENCY SHUTDOWN FAILS, FOR TAKING A FAULT OR SOME OTHER REASON, IT CAN BE RETRIED INDEFINITELY

- SHUTDOWN IS MARKED COMPLETE IN VOLUME LABELS, AND THESE ARE NOT UPDATED UNTIL THE VERY END. THUS, NO HARM CAN COME FROM RETRYING ARBITRARILY

  - ESD TRIES TO CORRECT ALL THE PROBLEMS THAT MIGHT ARISE. BECAUSE OF THE COMPLEXITY OF THE SUPERVISOR, IT DOES NOT ALWAYS GET ALL OF THEM

TOPIC XII

File System Salvagers

Page

OVERVIEW OF SALVAGERS

- FUNCTION

  - ENSURE THE CONSISTENCY OF THE FILE SYSTEM DATABASES AND PERFORM PERIODIC PREVENTIVE MAINTENANCE OPERATIONS

  - THERE ARE SEVERAL SALVAGERS, EACH WITH A DIFFERENT FUNCTION

  - BECAUSE OF THE COMPLICATED INTERACTIONS THEY HAVE WITH THE REST OF THE FILE SYSTEM, THE SALVAGERS ARE PERHAPS THE MOST COMPLICATED SINGLE PROGRAMS IN THE SUPERVISOR

  - SOME SALVAGING IS DONE AUTOMATICALLY, WHEN THE SYSTEM DETECTS AN INCONSISTENCY. OTHER SALVAGE OPERATIONS ARE EXPLICITLY REQUESTED, BY PRIVILEGED USERS.

  - EXCEPT FOR SUPERVISOR BUGS, THE ONLY TIME DAMAGE OCCURS THAT REQUIRES SALVAGING TO FIX IS AFTER A CRASH WHERE EMERGENCY SHUTDOWN FAILS

- THE SALVAGERS:

  - DIRECTORY SALVAGER

    - CORRECTS INCONSISTENCIES IN DIRECTORY SEGMENTS BY REBUILDING THEM

    - THIS IS THE ONLY SALVAGER INVOKED AUTOMATICALLY IN USER PROCESSES: ANY ATTEMPT TO LEAVE RING ZERO WITH A DIRECTORY LOCKED FOR WRITING WILL CAUSE IT TO BE SALVAGED

OVERVIEW OF SALVAGERS

- DIRECTORY SALVAGING ALSO RECLAIMS WASTED SPACE IN THE DIRECTORY, AND IS RUN PERIODICALLY TO COMPACT DIRECTORIES

- QUOTA SALVAGER

  - CORRECTS INCONSISTENCIES IN THE QUOTA SYSTEM
    TO FIX ASTES AFTER esd loss crash.

- PHYSICAL VOLUME SCAVENGER

  - RECONSTRUCTS RECORD AND VTOCE STOCK INFORMATION FROM THE VTOCES ON A VOLUME, THEREBY RECLAIMING ANY RECORDS OR VTOCES WHICH MIGHT HAVE BEEN LOST

  - RUNS ENTIRELY ONLINE WHILE THE SYSTEM IS UP FOR USERS (NEW IN MR10.1)

  - THIS TYPE OF DAMAGE IS USUALLY BENIGN, SO RUNNING THE SCAVENGER CAN BE DELAYED.

- PHYSICAL VOLUME SALVAGER

  - RECONSTRUCTS RECORD AND VTOCE STOCK INFORMATION

  - RUNS ONLY DURING INITIALIZATION, AND THEREFORE DELAYS CRASH RECOVERY

  - NOW USED ONLY FOR RARE CASES WHERE THERE IS NOT ENOUGH FREE SPACE LEFT FOR THE SCAVENGER TO RUN. IN THESE RARE CASES, IT IS INVOKED AUTOMATICALLY BY SYSTEM INITIALIZATION.

- SWEEP_PV

OVERVIEW OF SALVAGERS

- DELETES UNUSED VTOC ENTRIES WHICH HAVE NO DIRECTORY ENTRY POINTING TO THEM

- RUNS ENTIRELY IN USER RING, EXCEPT FOR ACTUALLY READING VTOC ENTRIES AND DIRECTORY ENTRIES

- PURELY A HOUSEKEEPING FUNCTION, AND RUN ONLY RARELY.

DIRECTORY SALVAGER

- THE DIRECTORY SALVAGER IS USED FOR TWO MAIN PURPOSES: DAMAGE CORRECTION AND STORAGE RECLAIMATION

- DIRECTORY DAMAGE occurs as needed

  - DAMAGE CAN OCCUR IN A DIRECTORY FOR SEVERAL REASONS:

    - DISK I/O ERROR WRITING BAD DATA TO A DIRECTORY SEGMENT

    - SUPERVISOR BUG

      But any crash can cause inconsistent dir.

    - CRASH WITHOUT ESD, WHERE THE DIRECTORY WAS UPDATED, BUT NOT FULLY WRITTEN TO DISK

  - ACTUAL DAMAGE TO DIRECTORIES IS COMPARATIVELY RARE, BUT:

    - IF A FAULT OCCURS FOR ANY REASON WHILE A USER HAS A DIRECTORY LOCKED, THE SYSTEM ASSUMES THE DIRECTORY COULD BE AT FAULT, AND SALVAGES transparent

    - THIS SORT OF SALVAGING IS DONE BY THE online_salvager PROGRAM, WHICH IS INVOKED DYNAMICALLY BY verify_lock IF A PROCESS ATTEMPTS TO LEAVE RING ZERO ("CRAWL OUT") WITH A DIRECTORY LOCKED user message

    - ORDINARILY, THERE IS NOTHING WRONG, AND THE SALVAGER JUST CHECKS OVER THE DIRECTORY

    - IF THE DIRECTORY IS DAMAGED, IT IS "REPAIRED" AS WELL AS POSSIBLE

DIRECTORY SALVAGER

Only applies to process_dirs. Most dirs get updated at least every 15 min

- UNFORTUNATELY, THE MOST COMMON FORMS OF DAMAGE DESTROY THE DIRECTORY HEADER SO THAT NO REPAIR IS POSSIBLE, AND THE DIRECTORY MUST BE REINITIALIZED AS EMPTY

  Not really true, only for process dirs.

STORAGE RECLAIMATION

- DIRECTORY SPACE CANNOT ALWAYS BE EFFICIENTLY RE-USED, AND UNUSABLE SPACE ACCUMULATES AS ENTRIES ARE CREATED AND DELETED

  - IT IS NOT PRACTICAL FOR THE SYSTEM TO COMPACT DIRECTORIES WHEN THIS HAPPENS, SINCE THIS COULD CAUSE MAJOR RESOURCE CONSUMPTION, AND CAUSE EXPENSE FOR RANDOM PROCESSES

  - INSTEAD, THE DIRECTORY SALVAGER IS PERIODICALLY RUN IN Salvager.SysDaemon PROCESSES TO REBUILD ALL THE DIRECTORIES IN THE HIERARCHY, COLLECTING ALL FREE SPACE

    - THIS IS DONE BY CALLS TO hphcs_$salv_directory

  - IF THE Salvager PROCESSES DETECT DAMAGE, IT IS CORRECTED AS WELL

  - DEMAND DIRECTORY SALVAGING IS USUALLY ALSO INSTRUCTED TO LOOK FOR AND CORRECT "CONNECTION FAILURES"

    - A CONNECTION FAILURE IS A DIRECTORY BRANCH WHICH INDICATES A FREE VTOCE, OR A VTOCE WITH A DIFFERENT UID THAN THE BRANCH. THIS MEANS THAT THE DIRECTORY AND VTOC ARE INCONSISTENT

    - CONNECTION FAILURES ARE USUALLY DELETED BY DEMAND SALVAGING (THIS IS AN OPTION)

    - ONLINE SALVAGING DOES NOT DELETE CONNECTION FAILURES BECAUSE IT IS NOT ALWAYS POSSIBLE TO IDENTIFY THEM PROPERLY DURING AN ONLINE SALVAGE

- DEMAND DIRECTORY SALVAGING CAN ALSO BE USED BY SYSTEM MAINTENANCE PERSONNEL WHEN THERE APPEAR TO BE DIRECTORY PROBLEMS

Salquota - dirs + QUOTA
quota - Quota only Quick

QUOTA SALVAGING

X repair salvquota -esdless
quota esd

- QUOTA SALVAGING CORRECTS INCONSISTENCIES IN THE HIERARCHY OF QUOTA-USED VALUES

  - THESE INCONSISTENCIES ARISE AFTER A CRASH WHERE ESD FAILS:

    - A SEGMENT (>udd>a>b) IS DELETED, FREEING PAGES, AND THE ASTES OF ITS PARENT (>udd>a) AND ITS PARENT'S PARENT (>udd) ARE UPDATED TO REFLECT THIS DECREASE IN QUOTA USED .

    - THE VTOCE FOR >udd>a IS WRITTEN TO DISK DURING NORMAL OPERATION. THE VTOCE FOR >udd IS NOT

    - THE SYSTEM CRASHES, AND NO ESD IS PERFORMED

    - WHEN THE SYSTEM COMES BACK UP, THE VTOCE FOR >udd IS INACCURATE: IT CLAIMS THAT RECORDS ARE STILL IN USE, WHEN IN FACT THEY WERE FREED

    - IF A NEW SEGMENT (>udd>a>c) IS NOW CREATED, IT MAY SPURIOUSLY CAUSE A RECORD QUOTA OVERFLOW ON >udd.

  - QUOTA SALVAGING IS PERFORMED ENTIRELY ONLINE, WHILE THE SYSTEM IS RUNNING, USUALLY IN A Salvager.SysDaemon PROCESS

    - THE FUNDAMENTAL PRINCIPLE BEHIND THE QUOTA SALVAGER IS THAT QUOTA INCONSISTENCIES CANNOT ARISE DURING NORMAL OPERATION: IF QUOTA IS CONSISTENT, OR INCONSISTENT BY <u>n</u> RECORDS, IT WILL STAY THAT WAY UNLESS EXPLICITLY CORRECTED

    - THE QUOTA SALVAGER PROCESS WALKS THE HIERARCHY FROM THE BOTTOM UP, USING do_subtree, CORRECTING INCONSISTENCIES IN A DIRECTORY AND ALL ITS SIBLINGS, THEN IN THEIR PARENT, AND SO ON UP

QUOTA SALVAGING

- AN hphcs_ ENTRY IS CALLED TO CORRECT EACH DIRECTORY, ON THE ASSUMPTION THAT ALL ITS CHILDREN ARE CONSISTENT

  - THE ACTUAL CORRECTION MECHANISM INVOLVES COMPLICATED MANIPULATION OF VARIOUS LOCKS. REFER TO THE Multics Storage System PLM (AN61) FOR A DESCRIPTION

PHYSICAL VOLUME SCAVENGER

- THIS IS ALSO PRIMARILY USEFUL AFTER A CRASH WITHOUT ESD

  - IT REBUILDS THE VOLUME MAP AND VTOC MAP FOR A VOLUME BY EXAMINING ALL THE VTOCES AND ASTES FOR SEGMENTS ON THE VOLUME

  - IT RUNS WHILE THE VOLUME IS IN USE BY USERS, AND NO INTERRUPTION OF SERVICE OCCURS

  - DAMAGE TO THE MAPS OCCURS AFTER A CRASH WHEN THE STOCKS, IN MEMORY, DO NOT GET PROPERLY UPDATED TO THE MAPS ON DISK

    - BECAUSE OF THE STOCK MANAGEMENT POLICIES, THIS IS ALWAYS BENIGN: RECORDS MAY BE MARKED IN-USE THAT ARE NOT, BUT IT IS NOT POSSIBLE FOR THE MAP TO INDICATE A RECORD AS FREE (REUSABLE) WHEN IT BELONGS TO A SEGMENT ON THE VOLUME

  - FOR RECONSTRUCTING THE VOLUME MAP, THE SCAVENGER WORKS BY AN INTERACTION WITH THE PAGE CONTROL RECORD ALLOCATION MECHANISM

    - IT BUILDS A MAP, IN scavenger_data, THAT SAYS WHAT VTOCE OWNS EACH RECORD ON THE VOLUME, AND THEN RESOLVES THE INCONSISTENCIES BETWEEN THAT AND THE MAP ON DISK

      - THE DATABASE IS WIRED FOR THE DURATION OF THE SCAVENGE, AND CAN BE QUITE LARGE (64K FOR AN MSU0501)

      - WHILE A SCAVENGER IS RUNNING, PAGE CONTROL KEEPS THE DATABASE UP TO DATE AS IT ALLOCATES AND FREES RECORDS

  - FOR VTOCES, THE PROBLEM IS MUCH SIMPLER: THE VTOC IS SIMPLY SCANNED, AND ALL IN-USE VTOCES ARE RECORDED

PHYSICAL VOLUME SCAVENGER

- vtoc_man ALSO KEEPS THE TABLE UP TO DATE AS IT FREES AND ALLOCATES VTOCES ON THE VOLUME

- A VTOCE THAT APPEARS IN-USE MAY NOT NECESSARILY BE PART OF THE HIERARCHY -- sweep_pv IS RUN TO TAKE CARE OF THAT

- AFTER THE NEW MAPS ARE CONSTRUCTED, THEY ARE WRITTEN TO DISK

  - THE SCAVENGE RUNS ENTIRELY IN RING ZERO, BUT IT CAN BE SAFELY INTERRUPTED AND RESTARTED FROM THE BEGINNING AT ANY TIME

    - THE SCAVENGER ACCESSES THE VTOC USING vtoc_man

A Started Per SSU.ec

- THE SCAVENGER TAKES ABOUT FIFTEEN MINUTES PER VOLUME, BUT THE SYSTEM IS UP WHILE IT DOES SO

ESD-less from root Down # daemons

After ESD'less Crash Evening Salvage dirs. Also weekly or Monthly

X repair Quota > 2 usually done in couple of minutes fixes quota.

X repair SalvQuota > 2 -compact -check_vtoce -{dcF} delete connection Failures. ie; directory entry But no vtoce.

MESSAGE: seg_fault seg contents not in Volume Table OF Contents.

Sweep_PV All pvs -gc -dl -force deletes reverse connection failures ie; vtoce But no directory entry.

Good ESD

Can DO repair quota But not necessary.

PHYSICAL VOLUME SALVAGER

- THIS DOES THE SAME JOB AS THE SCAVENGER, BUT CAN RUN ONLY WHEN THE VOLUME IS UNAVAILABLE TO USERS

  - BECAUSE IT HAS THE VOLUME ALL TO ITSELF, IT IS MUCH SIMPLER:

    - IT SCANS THE VTOC, BUILDING THE SAME SORT OF DATABASES AS THE SCAVENGER

    - WHEN FINISHED, IT RESOLVES INCONSISTENCIES, AND WRITES THE MAPS BACK TO DISK

      - THE VOLUME SALVAGER ACCESSES THE VTOC USING salv_abs_seg_NN

  - THE VOLUME SALVAGER TAKES ABOUT A MINUTE AND A HALF PER VOLUME, BUT ALL VOLUMES MUST BE SALVAGED BEFORE THE SYSTEM CAN RUN AGAIN

    - AFTER A CRASH WITHOUT ESD, ALL VOLUMES MOUNTED AT THE TIME OF THE CRASH ARE PRESUMED TO BE INCONSISTENT

  - THE VOLUME SALVAGER IS LARGELY OBSOLETE TODAY. IT IS RETAINED TO DEAL WITH RARE SITUATIONS WHICH MAKE IT IMPOSSIBLE TO BRING THE SYSTEM UP FAR ENOUGH TO RUN THE SCAVENGER WITHOUT FIRST REBUILDING THE FILE MAP TO FREE UNUSED SPACE

SWEEP PV

- sweep_pv ELIMINATES ORPHAN VTOCES

  - AN ORPHAN VTOCE IS ONE WHICH APPEARS TO DESCRIBE A VALID SEGMENT, BUT DOES NOT APPEAR IN ANY DIRECTORIES

    - ORPHANS ARE USUALLY CREATED BY CRASHES OR DIRECTORY DAMAGE WHERE THE DIRECTORY COULD NOT BE REPAIRED

    - ORPHANS ARE ALSO SOMETIMES CALLED "REVERSE CONNECTION FAILURES"

  - sweep_pv WORKS BY INSPECTING EVERY VTOCE ON A VOLUME, AND ATTEMPTING TO FIND ITS PARENT

    - THERE IS A "UID PATHNAME" IN PART 3 OF ALL VTOCES WHICH CONTAINS THE UIDS OF ALL ITS PARENT DIRECTORIES

    - BY STARTING FROM THE ROOT, AND SEARCHING EACH DIRECTORY IN THE PATH FOR THE UID OF THE NEXT ONE, THE VTOCE CAN BE FOUND

      - IF IT CAN'T BE FOUND, IT IS AN ORPHAN, AND SWEEP_PV DELETES IT

  - IT IS POSSIBLE TO "ADOPT" ORPHANS INTO A DIFFERENT PLACE IN THE HIERARCHY IF IT IS IMPORTANT TO RECOVER THEIR CONTENTS, USING THE adopt_seg TOOL

  - THE UID PATHNAME OF A VTOCE CAN BE INTERPRETED MANUALLY BY USING THE vtoc_pathname TOOL

Sweep_Pv -adopt → not really a good idea

TOPIC XIII

The Initializer.SysDaemon Process

Page

INITIALIZER OVERVIEW

- FUNCTION

  - THE SYSTEM'S INITIALIZATION, ADMINISTRATIVE AND CONTROL PROCESS (Initializer.SysDaemon.z), RESPONSIBLE FOR:

    - INITIALIZING THE OPERATING SYSTEM AT BOOTLOAD, FOLLOWING SUCCESSFUL INITIALIZATION OF THE SUPERVISOR

    - ANSWERING SERVICE (login and logout)

    - PROCESS CREATION AND DESTRUCTION

    - MESSAGE COORDINATOR (DAEMON COORDINATION)

    - SYSTEM ADMINISTRATION FUNCTIONS

    - SYSTEM ACCOUNTING FUNCTIONS

- MAJOR DATA BASES, ALL KEPT IN >sc1

  - ANSWER_TABLE

  - ABSENTEE_USER_TABLE

  - DAEMON_USER_TABLE

INITIALIZER OVERVIEW

- MASTER GROUP TABLE (MGT)
- CHANNEL DEFINITION TABLE (CDT)
- SYSTEM ADMINISTRATION TABLE (SAT)
- PERSON NAME TABLE (PNT)
- PROJECT DEFINITION TABLES (PDT'S)

SERVICES OF INITIALIZER

- THE Initializer.SysDaemon PROCESS PERFORMS VARIOUS SERVICES NECESSARY FOR THE SUPERVISOR. MANY OF THEM ARE PERFORMED BY THE INITIALIZER PRIMARILY FOR CONVENIENCE, SINCE IT IS EASIER TO CONTROL RESOURCES IN A CONTROLLED ENVIRONMENT SUCH AS A SINGLE PROCESS THAN TO CREATE A MECHANISM WHICH CAN BE RUN IN ALL PROCESSES

  - SYSTEM INITIALIZATION AND SHUTDOWN

    - THE Initializer PROCESS IS CREATED FROM THE ENVIRONMENT THAT INITIALIZES THE SUPERVISOR

    - WHEN THE SYSTEM IS SHUT DOWN, THE Initializer COORDINATES THE ORDERLY CESSATION OF SYSTEM ACTIVITY AND CALLS hphcs_$shutdown TO SHUT DOWN THE SUPERVISOR

  - LOGICAL VOLUME MANAGEMENT

    - THE Initializer PROCESS HANDLES ALL REQUESTS FROM USER PROCESSES TO MOUNT AND UNMOUNT PRIVATE LOGICAL VOLUMES

    - IT ALSO MAKES THE NECESSARY CHECKS TO ACCEPT A VOLUME INTO THE HIERARCHY OR REMOVE IT FOR DEMOUNTING

  - RESOURCE CONTROL

    - THE Initializer DOES ALL THE CONTROL, ASSIGNMENT, AND ACCESS CHECKING OF RESOURCES (SUCH AS I/O DEVICES) CONTROLLED BY RCP

  - PROCESS CREATION AND DESTRUCTION

SERVICES OF INITIALIZER

- THE Initializer CREATES AND DESTROYS ALL PROCESSES, SCHEDULES ABSENTEE PROCESSES, AND HANDLES "CONSOLE" I/O FROM DAEMON PROCESSES

- AN IMPORTANT SERVICE OF PROCESS MANAGEMENT IS PROCESS DIR VOLUME MANAGEMENT, IN WHICH THE Initializer PICKS THE LOGICAL VOLUMES USED FOR PROCESS DIRECTORIES

- THE Initializer ALSO HANDLES ALL PROCESS ACCOUNTING AND LOAD CONTROL

- COMMUNICATIONS

  - THE Initializer MANAGES LOADING / DUMPING OF FNPs AND SOFTWARE COMMUNICATIONS MULTIPLEXERS

  - IT ALSO HANDLES ALLOCATION OF COMMUNICATION CHANNELS, BOTH FOR PROCESSES LOGGING IN AND REQUESTS MADE THROUGH dial_manager_

- DYNAMIC RECONFIGURATION

  - THE Initializer RUNS THE DYNAMIC RECONFIGURATION SOFTWARE, AND UPDATES THE SYSTEM LOAD LIMITS WHEN THE CONFIGURATION CHANGES

  - ACTUALLY, ANY PROCESS WITH hphcs_ ACCESS CAN USE THE RECONFIGURATION COMMANDS, BUT ONLY THE Initializer CAN KEEP ALL THE ACCOUNTING DATABASES UP TO DATE

TOPIC XIV

Metering and Tuning

Page

METER AND TUNING OVERVIEW

Probably 1970 overhead.

- WHILE NOT A SUBSYSTEM ITSELF, METERING AND TUNING IS A POLICY AND CAPABILITY COMMON TO ALL OF THE SUPERVISOR'S SUBSYSTEMS

- FUNCTION

  - METERING (CONSISTS OF THREE ACTIVITIES)

    - ACCUMULATING DATA: THIS IS PERFORMED THROUGHOUT THE SUPERVISOR BY CODE WHICH

      - RECORDS THE NUMBER OF TIMES AN EVENT HAPPENS OR A PARTICULAR PIECE OF CODE IS EXECUTED; AND/OR

      - RECORDS THE TIME REQUIRED TO PERFORM A TASK

      - SUCH DATA IS STORED IN AREAS REFERRED TO AS "METERING CELLS"

    - EXTRACTING DATA: THIS IS PERFORMED BY NUMEROUS METERING COMMANDS WHICH (WHEN INVOKED)

      - READ AND STORE THE CURRENT VALUES OF RELEVANT METERING CELLS

    - REPORTING THE DATA: THIS IS PERFORMED BY THE METER COMMANDS WHICH (WHEN INVOKED)

      - COMPARE CURRENT METERING CELL VALUES WITH PREVIOUSLY READ VALUES

      - PERFORM THE APPROPRIATE ARITHMETIC COMPUTATIONS UPON THE DATA IN ORDER TO ARRIVE AT THE DESIRED STATISTIC

      - ARRANGE THE DATA IN A USEFUL FORMAT (A REPORT OR DIAGRAM) AND PRINT IT

METER AND TUNING OVERVIEW

- TUNING

  - CHANGING THE SYSTEM'S OPERATING PARAMETERS AND/OR CONFIGURATION BASED UPON THE DATA AND INSIGHTS FROM THE SYSTEM'S METERS

- MAJOR DATA BASES

  - SST HEADER, TC_DATA HEADER, ETC.

ANALYZING PERFORMANCE PROBLEMS

- MULTICS IS VERY HEAVILY INSTRUMENTED WITH MANY METERING COMMANDS

  - MOST SOLVABLE PERFORMANCE PROBLEMS SHOW UP QUITE DIRECTLY IN THE METERS

  - IF THE SYSTEM IS TOO SLOW, THERE ARE GOOD RULES OF THUMB TO FOLLOW TO LOOK FOR WHEN PROBLEM

  - NO TWO SYSTEMS ARE IDENTICAL - THE MOST IMPORTANT INFORMATION YOU HAVE IS HOW THE METERS ARE DIFFERENT FROM THE WAY THEY WERE WHEN THE SYSTEM WAS WORKING BETTER

  - NOT ALL TUNING PROBLEMS CAN BE RESOLVED WITH SOFTWARE. OFTEN IT SIMPLY INDICATES THAT THERE IS NOT ENOUGH MEMORY, OR NOT ENOUGH DISK CAPACITY. IT IS OFTEN DIFFICULT TO DETERMINE WHAT HARDWARE CHANGES WOULD BE MOST COST-EFFECTIVE

- total_time_meters - THE FIRST STEP

  - total_time_meters MAY INDICATE EXCESSIVE TIME SPENT IN SEVERAL AREAS:

    - PAGE FAULTS - TOO MANY PAGE FAULTS MEAN NOT ENOUGH MEMORY, TOO MANY ELIGIBLE PROCESSES (max_eligible), WHICH CAUSE THRASHING, INSUFFICIENT DISK CAPACITY, OR INEFFICIENT APPLICATIONS

      - LOOK TO file_system_meters, device_meters, AND disk_meters FOR MORE HELP

- SEGMENT FAULTS - TOO MANY ALMOST ALWAYS MEAN THAT THE AST POOLS (sst CONFIG CARD) ARE TOO SMALL
  - LOOK TO file_system_meters (AST Pool grace time) FOR MORE HELP

- INTERRUPT - USUALLY MEANS EXCESSIVE INTERRUPT ACTIVITY EITHER FOR FNPS OR BECAUSE OF EXCESSIVE PAGING, BUT MAY INDICATE HARDWARE PROBLEMS
  - LOOK TO interrupt_meters TO LOCALIZE IT, THEN TO file_system_meters and system_comm_meters

- OTHER FAULT - GENERALLY INDICATES TOO MANY CONNECT OR TIMER RUNOUTS FAULTS, INDICATING EXCESSIVE TRAFFIC CONTROL ACTIVITY
  - LOOK TO THE TUNING PARAMETERS (ptp) AND traffic_control_meters FOR MORE HELP

- MP IDLE - INDICATES TOO MUCH TRAFFIC CONTROL ACTIVITY, USUALLY BECAUSE THERE ARE TOO MANY ELIGIBLE PROCESSES (max_eligible) AND/OR NOT ENOUGH MEMORY
  - LOOK TO THE TUNING PARAMETERS (ptp), file_system_meters, device_meters, disk_meters, AND traffic_control_meters FOR MORE HELP

- WORK CLASS IDLE - THIS IS CPU TIME WASTED BECAUSE GOVERNED WORKCLASSES WERE NOT PERMITTED TO USE IT, AND NO OTHER TAKERS WANTED IT
  - FREQUENTLY NOT A PROBLEM, BUT MAY INDICATE A NEED TO READJUST WORKCLASSES (work_class_meters)

- NMP IDLE, ZERO IDLE - INDICATE THAT CPU TIME HAS GONE TO WASTE BECAUSE NO PROCESSES WANTED IT. NMP IDLE MEANS THAT THERE ARE MORE PROCESSORS THAN PROCESSES THAT WANT CPU TIME, AND ZERO IDLE MEANS THERE ARE NO PROCESSES AT ALL

ANALYZING PERFORMANCE PROBLEMS

- NORMALLY INSIGNIFICANT QUANTITIES - THE FOLLOWING NUMBERS SHOULD ALWAYS BE VERY SMALL; LARGE VALUES PROBABLY INDICATE A HARDWARE OR SOFTWARE PROBLEM

  PC LOOP LOCKS
  BOUND FAULTS
  TC LOOP LOCKS
  POST PURGING
  LOADING IDLE
  OTHER OVERHEAD

DETAILED PROBLEM ANALYSIS

- THE MORE DETAILED METERING COMMANDS CAN BE USED TO PIN DOWN A PROBLEM

  - DETAILED DESCRIPTIONS CAN BE FOUND IN THE Multics Metering Manual (AN52)

  - THE DETAILED METERS SHOULD ALSO BE CHECKED IF THERE IS NO OBVIOUS PROBLEM SHOWN BY total_time_meters

  - AGAIN, THE REALLY IMPORTANT THING IS TO BE ABLE TO COMPARE AGAINST PREVIOUS DATA FOR YOUR SITE

- SOME GOOD PLACES TO LOOK ARE:

  - interrupt_meters

    - UNUSUALLY LONG DISK INTERRUPTS MAY INDICATE LOCKING PROBLEMS - SEE disk_meters

    - TOO MUCH TOTAL TIME SPENT WITH DISK INTERRUPTS USUALLY MEANS TOO MANY PAGE FAULTS, WHICH WILL BE SHOWN IN MORE DETAIL BY file_system_meters

    - FNP INTERRUPTS ARE TYPICALLY MUCH LONGER THAN OTHERS. TOO MUCH FNP INTERRUPT TIME <u>MAY</u> INDICATE A BAD FRONT END CHANNEL, BUT THIS IS RARE

DETAILED PROBLEM ANALYSIS

- file_system_meters

  - A SHORT PAGE LAP TIME MEANS THAT PAGES ARE NOT STAYING IN MEMORY LONG ENOUGH; THAT IS, THERE IS NOT ENOUGH MEMORY

  - FREQUENT PAGE CLAIM RUNS INDICATE THAT THE WRITE LIMIT IS TOO LOW

  - IF THE AVERAGE PAGE FAULT DURATION IS HIGH, IT CAN MEAN THAT THE WRITE LIMIT IS TOO HIGH, AND THAT DISK ALLOCATION LOCKS ARE A LIMITATION; SEE disk_meters TO CHECK

  - SHORT AST POOL GRACE TIMES MEANS THAT THE POOL SIZES SHOULD BE INCREASED. THIS IS USUALLY ALSO INDICATED BY TOO HIGH A FREQUENCY OF SEGMENT FAULTS

- disk_meters

  - WHEN THE ATB I/O FOR A DRIVE IS TOO LOW, THE DRIVE IS A SERIOUS BOTTLENECK.

    - THIS OFTEN HAPPENS FOR RLV DRIVES, BECAUSE OF DIRECTORIES

    - IF PARM DIRW IS ON, IT SHOULD BE TURNED OFF

  - TOO HIGH A PERCENTAGE OF ALLOCATION LOCKS INDICATE THAT TOO MANY WRITES ARE BEING QUEUED SIMULTANEOUSLY, AND MAKING IT IMPOSSIBLE TO GET A QUEUE ENTRY FOR READING

    - THIS USUALLY MEANS THAT WRITE_LIMIT IS TOO HIGH

- device_meters

  - IF THE SUBSYSTEM BUSY PERCENTAGE IS TOO CLOSE TO THE DISK CHANNEL CAPACITY, IT USUALLY MEANS THAT THERE ARE TOO FEW PHYSICAL DISK CHANNELS

OUTPUT FROM METERING COMMANDS

total time meters

- TOTAL_TIME_METERS - OVERVIEW OF HOW THE SYSTEM IS USING ITS RESOURCES, ALSO MEASURED AGAINST NON-IDLE TIME

Total metering time 0:20:36

	%	%NI	AVE
Page Faults	6.55	7.19	2130.424
PC Loop Locks	0.39	0.43	1993.209
PC Queue	0.86	0.94	348.946
Seg Faults	1.74	1.90	9579.170
Bound Faults	0.15	0.16	17426.208
Interrupts	9.02	9.89	1713.504
Other Fault	8.45	9.27	
Getwork	4.54	4.98	660.550
TC Loop Locks	0.20	0.22	247.788
Post Purging	0.36	0.39	1407.132
MP Idle	0.70	0.77	
Work Class Idle	0.98	1.08	
Loading Idle	0.16	0.17	
NMP Idle	8.84		
Zero Idle	0.00		
Other Overhead	0.03	0.04	
Virtual CPU Time	62.53	68.59	

- Avg CPU Time Per Page FAULT
DPS 8 → 1500.

Other FAULT – (Getwork, TC loop locks, Post Purging) equals Connect faults

Check AVE first

Page FAULTS >12% is hi. 1) Check IoP_Times for Pages in File_System_meters for Page Thrashing i.e. hi. MAXE, TE last

2) First Check AVE IF >1800 Then Bottle neck. Then check disk_meters

Seg FAULTS >2-3% Then
Check file_system meters, Possibly hi.
IF FSM OK Possibly disk Bottle neck.

Interrupts >12% check interrupt meters.
Check FNP's. IF interrupts changes dramaticly, check for a particular FNP.

getwork >3% check TE first → short or TE last

MP idle >1% Check for low MAKE or high disk I/O + Paging
So check MP idle Because if you set MAKE it will make it worse.

OUTPUT FROM METERING COMMANDS

interrupt meters

* INTERRUPT_METERS - COUNTS & TIMING FOR ALL I/O INTERRUPTS

Total metering time 0:20:28

IOM	Ch	Int	Avg Time	% CPU	Name
A	6.	11	2.043	0.00	IOM A special
A	10.	379	1.553	0.01	prtb
A	13.	9253	3.436	0.65	fnp c
A	14.	19	0.788	0.00	opc
A	16.	26248	1.347	0.72	tapa
A	17.	128	1.328	0.00	tapa
A	18.	13097	5.383	1.43	fnp b
A	20.	21041	0.860	0.37	dska
A	21.	2881	0.860	0.05	dska
A	22.	8486	0.877	0.15	dskb
A	23.	257	0.964	0.01	dskb
A	24.	17896	0.895	0.33	dskb
A	25.	2573	0.943	0.05	dskb
A	26.	10486	0.857	0.18	dska
A	27.	255	0.899	0.00	dska
A	28.	24717	0.845	0.43	dskc
A	29.	2319	0.849	0.04	dskc
A	30.	9915	0.845	0.17	dskc
A	31.	215	0.767	0.00	dskc
B	14.	236	0.877	0.00	fnp d
B	15.	6917	3.351	0.47	fnp f
B	18.	3547	3.543	0.26	fnp a
B	19.	13667	3.435	0.96	fnp e
B	20.	22013	0.881	0.39	dskb
B	21.	4745	0.863	0.08	dskb
B	22.	6054	0.861	0.11	dska
B	23.	60	0.925	0.00	dska
B	24.	15946	0.867	0.28	dska
B	25.	989	0.841	0.02	dska
B	26.	12757	0.863	0.22	dskb
B	27.	1020	0.903	0.02	dskb
B	28.	5178	0.838	0.09	dskc
B	29.	30	0.653	0.00	dskc
B	30.	16582	0.823	0.28	dskc
B	31.	825	0.841	0.01	dskc
B	32.	1764	0.825	0.03	dske
Chan		262506	1.464	7.82	
Ovhd		258932	0.230	1.21	
Total		258932	1.713	9.03	

OUTPUT FROM METERING COMMANDS

file system meters

- FILE_SYSTEM_METERS - DISPLAYS MISCELLANEOUS METERING INFORMATION FOR THE FILE SYSTEM

  I ONLY PARTS RELEVANT TO SEGMENT CONTROL INCLUDED HERE; SEE TOPIC 8 (PAGE CONTROL) FOR THE REST

```
Total metering time          0:20:02

                        #                    ATB

Activations            1043      1.153 sec.
   segfault             969      1.241 sec.  92.905% of all
   makeknown             74     16.251 sec.   7.095% of all
   directories           96     12.527 sec.   9.204% of all
Deactivations          1056      1.139 sec.
Demand deactivate
   attempts               3    400.857 sec.
Seg Faults             5080      0.237 sec.
   fault               4311      0.279 sec.  84.862% of Seg Faults
   call                 769      1.564 sec.  15.138% of Seg Faults
   activations          969      1.241 sec.  19.075% of Seg Faults
Bound Faults            220      5.466 sec.
Setfaults              4484    268.191 msec.
   access                42     28.633 sec.   0.937% of setfaults
ASTE Trickle            139      8.652 sec.
Steps                  4279    281.040 msec.
Skips                  3016      0.399 sec.  70.484% of Steps
   ens                  271      4.438 sec.   8.985% of Skips
   mem                 1083      1.110 sec.  35.909% of Skips
   init                1662      0.724 sec.  55.106% of Skips
Searches                  0      0.000 sec.
Cleanups                   1056      1.139 sec.   0.1 % of real time
Force writes              3    400.857 sec.
   pages written          3    400.857 sec.
Lock AST              18422      0.065 sec.
```

OUTPUT FROM METERING COMMANDS

file system meters

	AVE/lock	%
AST locked	4.833 msec.	7.4
AST lock waiting	1.601 msec.	2.5

AST Sizes	4	16	64	256
Number	1701	601	221	74
Need	819	202	208	34
Steps	2341	645	1139	154
Ave Steps	2.9	3.2	5.5	4.5
Lap Time(sec)	873.8	1120.5	233.3	577.9

OUTPUT FROM METERING COMMANDS

file system meters

- FILE_SYSTEM_METERS - DISPLAYS MISCELLANEOUS METERING INFORMATION FOR THE FILE SYSTEM

  - ONLY PARTS RELEVANT TO PAGE CONTROL INCLUDED HERE; SEE TOPIC 7 (SEGMENT CONTROL) FOR THE REST

```
Total metering time    0:20:02

                   #           ATB

Needc           62654      19.194 msec.
Ring 0 faults              16.639 %
PDIR faults                50.607 %
Level 2 faults             21.556 %
DIR faults                  7.645 %
New Pages                  14.661 %
Volmap_seg          0       0.000 msec.
Zero pages        770    1561.779 msec.
Laps              105      11.453 sec.
Steps          361483       3.327 msec.
Skip           322555       3.728 msec.  89.231% of Steps
   wired        11057     108.761 msec.   3.428% of Skip
   used        109719      10.960 msec.  34.016% of Skip
   mod         140336       8.569 msec.  43.508% of Skip
   fc pin       37717      31.884 msec.  11.693% of Skip
   cl pin       23726      50.686 msec.   7.356% of Skip

3419 pages, 139 wired.
Average steps          5.770
```

* DEVICE_METERS - DISPLAYS SUMMARY OF I/O ACTIVITY FOR ALL DISK SUBSYSTEMS

Total metering time 0:20:13

	dska	dskb	dskc	dskd
Prior Page I/O	18571	17743	462	1273
ATB	65.334	68.383	2626.240	953.121
Other Page I/O	6525	5135	16	696
ATB	185.949	236.284	75832.692	1743.280
ATB Page I/O	48.347	53.034	2538.332	616.212
Prior VTOCE I/O	934	895	38	304
ATB	1299.061	1355.668	31929.554	3991.194
ATB I/O	46.612	51.037	2351.401	533.798
% Busy	76	74	0	4
Avg. Page Wait	47.289	46.197	20.341	24.666
Avg. Page ^Wait	176.082	101.023	36.996	61.704
Avg VTOCE Wait	41.138	37.610	38.595	29.090
Avg. Page I/O T	35.619	38.314	20.050	22.482
Avg. VTOCE I/O T	31.139	32.277	37.060	26.606
EDAC Corr. Errs	0	0	0	0
Errors	0	0	0	1
Fatal Errors	0	0	0	0

disk meters

- DISK_METERS - DISPLAYS I/O ACTIVITY TO DISK DRIVES

  - ONLY ONE SUBSYSTEM SHOWN HERE TO CONSERVE SPACE

old copy

Total metering time 0:20:12

If non 0 Then check % Then check I/O's for subsystems

Subsystem dska	Count	Waits	%Waits	Avg. Wait(ms.)
call locks	26005	217	0.83	0.259
run locks	112	0	0.00	0.000
interrupt locks	25998	239	0.92	0.208
allocations	26001	0	0.00 <2%	0.000

Times ~~tried~~ tried to put something in Que

Average Time between

Drive	Reads	Writes	Seek Distance	ATB Reads	ATB Writes	ATB I/O
1	269	67	214	4508	18102	3609 → 3.5 sec not busy
3	362	243	109	3350	4991	2004
4	309	131	184	3925	9258	2756
5	547	165	180	2217	7350	1703
6	631	165	161	1922	7350	1523
7	0	0	0	0	0	0
8	5843	2187	122	207	554	151
9	366	116	153	3313	10455	2516
11	3501	1431	200	346	847	245
12	0	0	0	0	0	0
16	7158	2508	135	169	483	125 - very busy

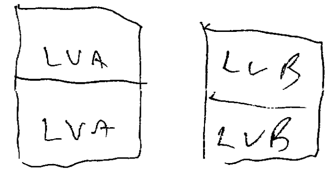
2 configs

OUTPUT FROM METERING COMMANDS

501

disk queue

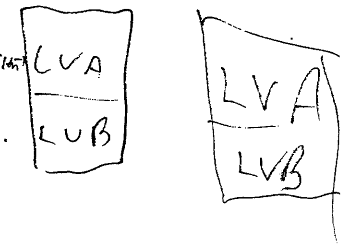
IF you lose one spindle only lose 1 system but not as efficient 1.



* DISK_QUEUE - DISPLAYS I/O QUEUE FOR A DISK SUBSYSTEM

  * ONLY ONE SUBSYSTEM SHOWN HERE TO CONSERVE SPACE

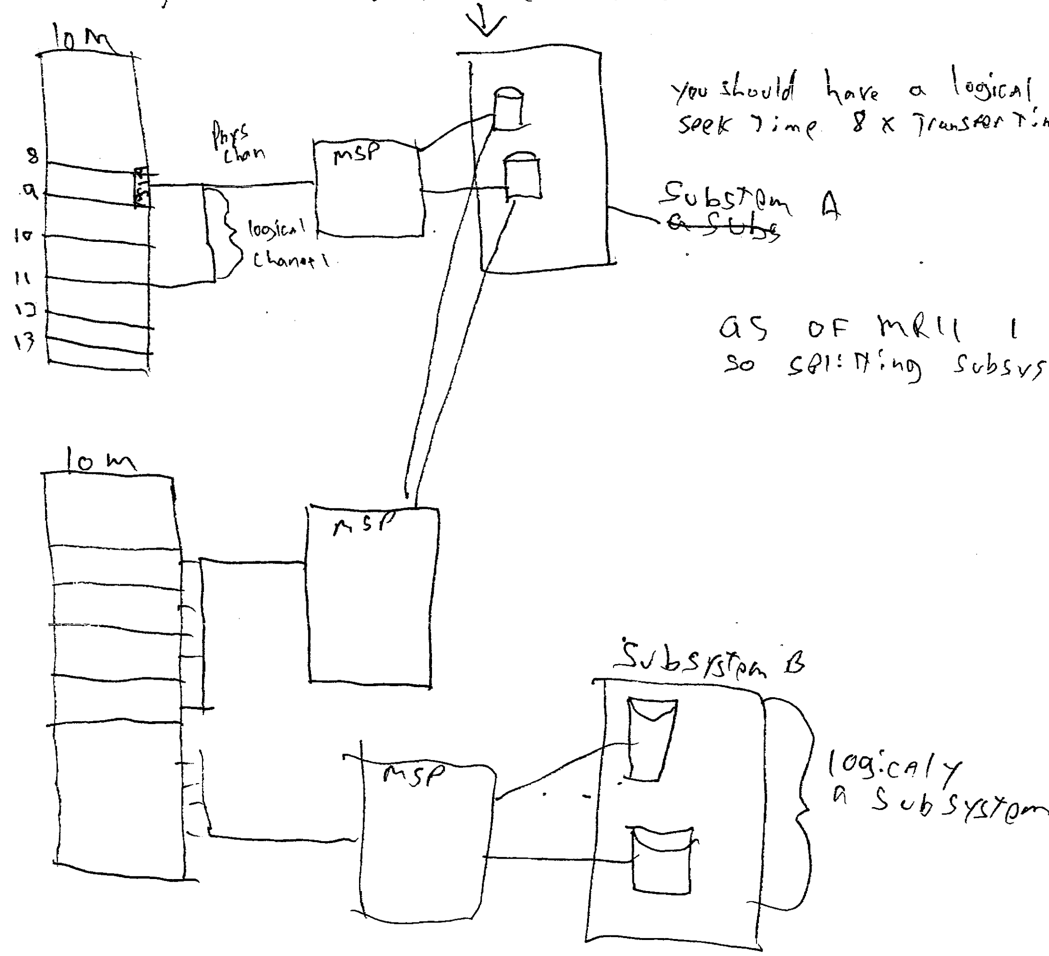
efficient but less reliable.



Connects = 2604781, 1359725, 677321, 309367,
123430, 40159, 10227, 1969.

P	RW	VP	DV	SECTOR	MEM
0	W	P	24	1350330	27304000
0	W	P	9	1020150	4432000
0	W	P	16	1204130	36246000
0	W	P	16	314370	27306000
0	W	P	16	314430	34166000

goes with total time meters



you should have a logical channell for each arm
seek time 8 x transfer time.

as of MR11 1 disk queue
so splitting subsystems isn't necessary.

print configuration deck

* PRINT_CONFIGURATION_DECK - DISPLAYS >sl1>config_deck, WHICH CONTAINS INFORMATION ABOUT DISK LOCATIONS, THE RLV, AND PARTITIONS

  - ONLY THE PART OF THE CONFIG DECK RELEVANT TO VOLUME MANAGEMENT AND DISK CONFIGURATION IS SHOWN HERE

```
root  dska  16.  dskb  25.  dskb  23.  dskb  24.  dska  8.

part  bos  dska  16.
part  dump  dska  16.
part  log  dska  16.

prph  dska  a  20.  2  451.  16.
chnl  dska  a  26.  2  b  24.  2  b  22.  2

prph  dskb  b  20.  2  0  16.  451.  16.
chnl  dskb  b  26.  2  a  24.  2  a  22.  2

prph  dskc  a  28.  2  501.  32.
chnl  dskc  a  30.  2  b  30.  2  b  28.  2

prph  dske  b  32.  2  451.  8.
chnl  dske  b  34.  2

prph  dskf  a  32.  2  501.  16.
chnl  dskf  a  34.  2

mpc  mspa  451.  a  20.  4  a  24.  4
mpc  mspb  451.  b  20.  4  b  24.  4
mpc  mspc  607.  a  28.  4
mpc  mspd  607.  b  28.  4
mpc  mspe  451.  b  32.  4
mpc  mspf  607.  a  32.  4
```

print configuration deck

- DISK CONFIGURATION CONFIG CARDS

  - ROOT

    - IDENTIFIES THOSE VOLUMES IN THE ROOT LOGICAL VOLUME WHICH HAVE HC PARTITIONS, USED BY THE SUPERVISOR FOR PAGING OF SUPERVISOR SEGMETNS

  - PART

    - IDENTIFIES THE LOCATIONS OF CERTAIN IMPORTANT PARTITIONS

    - ONLY PARTITIONS NECESSARY FOR MULTICS OPERATIONS ARE IDENTIFIED, NOT ALT PARTITIONS

    - HC PARTITIONS ARE LOCATED BY THE ROOT CARD

  - PRPH DSK<u>n</u>, CHNL

    - IDENTIFY PHYSICAL I/O CHANNEL PATHS FOR ACCESSING DISK DRIVES

  - MPC

    - IDENTIFY PHYSICAL CONNECTIONS TO MICROPROGRAMMED DISK CONTROLLERS

list vols

- LIST_VOLS - DISPLAYS A TABLE OF ONLINE VOLUMES, THEIR LOCATION, AND SPACE UTILIZATION

Drive	Records	Left	%	VTOCEs	Left	%	Avg Size	PV Name	PB/PD	LV Name
dskc_17	64504	54730	85	13440	11356	84	4	alpha01	pb pd	Alpha
dskc_18	64504	55389	86	13440	11352	84	4	alpha02	pb pd	Alpha
dska_05	36428	5305	15	8400	2662	32	5	mul03	pb pd	Multics_Pubs
dska_06	36428	4323	12	8400	2365	28	5	mul01	pb pd	Multics_Pubs
dskb_19	36428	4632	13	8400	2813	33	5	mul02	pb pd	Multics_Pubs
dskb_26	36429	13690	38	8400	4326	52	5	mul05	pb pd	Multics_Pubs
dskb_27	36429	4672	13	8400	2333	28	5	mul04	pb pd	Multics_Pubs
dska_01	36308	4680	13	9000	450	5	3	pub01	pb pd	Public
dska_03	36268	3588	10	9200	1017	11	3	pub07	pb pd	Public
dska_04	36268	4500	12	9200	884	10	3	pub04	pb pd	Public
dska_09	36268	4816	13	9200	864	9	3	pub02	pb pd	Public
dskb_17	36269	4281	12	9200	1002	11	3	pub05	pb pd	Public
dskb_18	36268	3840	11	9200	539	6	3	pub06	pb pd	Public
dskc_13	64504	294	0	13440	5539	41	8	rel01		Release
dskc_14	64504	269	0	13440	5214	39	7	rel02		Release
dskc_01	64503	43631	68	13440	10032	75	6	xpub01	pb	Xpublic
dskc_02	64503	45502	71	13440	9838	73	5	xpub02	pb	Xpublic
dskc_03	64503	42374	66	13440	9839	73	6	xpub03	pb	Xpublic
dskc_04	64503	43591	68	13440	9785	73	5	xpub04	pb	Xpublic
dskc_09	64504	58010	90	13440	12394	92	6	xpub05	pb	Xpublic
dskc_10	64504	56786	88	13440	12407	92	7	xpub06	pb	Xpublic
dskc_21	64503	23947	37	13440	6446	48	5	ypub01	pb	Ypublic
dskc_22	64503	23744	37	13440	6194	46	5	ypub02	pb	Ypublic
dskc_29	64503	23794	37	13440	6185	46	5	ypub05	pb	Ypublic
dskc_30	64503	24111	37	13440	6481	48	5	ypub06	pb	Ypublic
dskc_07	64503	11723	18	13440	6149	46	7	zpub01	pb pd	Zpublic
dskc_08	64503	11665	18	13440	6429	48	7	zpub02	pb pd	Zpublic
dskc_23	64504	9777	15	13440	6094	45	7	zpub03	pb pd	Zpublic
dskc_24	64504	11805	18	13440	6141	46	7	zpub04	pb pd	Zpublic
dskc_25	64504	11514	18	13440	7407	55	8	zpub05	pb pd	Zpublic
dskc_26	64504	12958	20	13440	7149	53	8	zpub06	pb pd	Zpublic
dske_06	37089	8053	22	5100	3797	74	22	list01	pb pd	list_1
dska_12	37562	6046	16	2735	957	35	17	list02	pb	list_2
dska_07	37309	6825	18	4000	2107	53	16	list03	pb	list_3
dska_08	36209	3827	11	7000	491	7	4	root5	pb	root
dska_11	36209	8597	24	7000	4181	60	9	root6	pb	root
dska_16	31283	2892	9	9000	3476	39	5	rpv	pb	root
dskb_23	36208	4888	13	7000	455	7	4	root3	pb	root
dskb_24	36209	3097	9	7000	238	3	4	root4	pb	root
dskb_25	36350	4483	12	7000	223	3	4	root2	pb	root

OUTPUT FROM METERING COMMANDS

traffic control queue

- TRAFFIC_CONTROL_QUEUE - DISPLAYS THE CURRENT CONTENTS OF THE SCHEDULER QUEUES, USEFUL FOR GETTING AN IDEA OF WHAT THE USER PROCESSES ARE DOING

I FIRST PART OF OUTPUT IS ELIGIBLE QUEUE:

avq = 13, elapsed time = 1247 sec, 64 active last 15 sec.

flags	dtu	dpf	temax	te	ts	ti	tssc	event	d	ws	wc	process
rWLE(d)	148	6946	2097	37	0	0	-0.001	0	0	6	0	Initializer
xLED(c)	15	1111	1000	910	2012	1897	0.001	0	0	11	6	Sibert
rLE(d)	14	1370	500	89	0	0	-0.009	0	0	0	4	Diaz
rLE(d)	15	823	500	13	0	0	-0.009	0	0	64	8	JCrow
wLE(d)	13	634	500	422	0	0	0.018	316537	0	3	3	Gintell
xWLED(a)	6	108	1000	495	2054	2004	0.010	0	0	13	6	Brunelle
wLE(d)	16	864	1000	85	0	510	0.016	504031	0	46	6	WPeck
wWLE(b)	468	2734	1000	315	3010	8000	0.013	224641	0	4	3	Spratt
wLE(d)	17	686	500	60	0	0	0.007	165111	0	3	4	RTowle
wLE(b)	12	520	500	50	0	0	0.005	71	0	3	3	Kress
wLE(d)	69	1895	500	21	0	0	0.001	177022	0	0	2	OPCTL
xLED(b)	12	872	500	85	0	0	0.005	0	0	0	3	Pandolf
rLE(d)	67	3279	500	0	0	0	-0.006	0	0	0	3	Lackey

! SECOND PART IS REALTIME, INTERACTIVE, AND ALL WORKCLASS QUEUES:

```
REALTIME QUEUE:

INTERACTIVE QUEUE:

WORKCLASS  2 QUEUE: credits =   576 ms.

WORKCLASS  3 QUEUE: credits =   242 ms.
r         289   5326  1000    0     0   503   0.218     0 0   22   3 Dupuis
r         131   2513  1000    0  4010  8000   0.128     0 0    0   3 Falksenj

WORKCLASS  4 QUEUE: credits =  2601 ms.

WORKCLASS  5 QUEUE: credits =  4000 ms.

WORKCLASS  6 QUEUE: credits =  -563 ms.

WORKCLASS  7 QUEUE: credits =  3962 ms.
rW          5    166   500    0     0     0   0.192     0 0    2   7 Saccuci

WORKCLASS  8 QUEUE: credits =  3934 ms.

WORKCLASS  9 QUEUE: credits =  2216 ms.

WORKCLASS 10 QUEUE: credits =  4000 ms.

WORKCLASS 11 QUEUE: credits =  4000 ms.
```

OUTPUT FROM METERING COMMANDS

traffic control meters

- TRAFFIC_CONTROL_METERS - DISPLAY THE STATE OF THE SCHEDULER

  - OUTPUT COMES IN THREE PARTS, SHOWN OUT OF ORDER HERE:

    - QUEUE LENGTHS AND RESPONSE TIME - THESE ARE WEIGHTED AVERAGES OVER THE LAST FIFTEEN SECONDS.

    - ACTIVITIES VERSUS DEPTH - HOW DEEP THE TRAFFIC CONTROLLER HAD TO SEARCH TO FIND A SCHEDULABLE PROCESS

    - MISCELLANEOUS COUNTERS AND FREQUENCIES OF VARIOUS EVENTS

```
Total metering time    0:20:34

Ave queue length    16.52
Ave eligible        13.31
Response time        0.264 sec
```

DEPTH	%PF	TBPF	%GTW	TBS	%CPU
1	12.0	22.8	10.8	11.6	8.1
2	11.8	21.0	9.2	12.2	7.4
3	10.8	24.9	8.7	14.1	8.0
4	10.2	27.9	8.5	15.2	8.4
5	9.5	29.8	8.3	15.5	8.4
6	8.4	33.5	7.8	16.5	8.4
7	7.4	36.3	7.3	16.8	8.0
8	30.0	48.6	39.5	16.8	43.3

OUTPUT FROM METERING COMMANDS

traffic control meters

COUNTER	TOTAL	ATB	#/INT
Interactions	7977	0.155 sec	
Loadings	12161	0.102 sec	1.525
Blocks	14082	0.088 sec	
Wakeups	36078	0.034 sec	
Schedulings	12591	0.098 sec	1.578
Lost priority	1	1234.756 sec	
Priority boosts	0	0.000 sec	
I/O boosts	578	2.136 sec	
Wait Page	127040	9.719 msec	15.926
Wait PTL	75691	16.313 msec	9.489
Wait Other	31912	38.693 msec	4.001
Total Waits	234643	5.262 msec	29.415
Notify Page	128954	9.575 msec	
Notify PTL	75691	16.313 msec	
Notify Other	25330	48.747 msec	
Total Notifies	229975	5.369 msec	
Get Processor	245856	5.022 msec	
Pre-empts	94235	13.103 msec	11.813
Getwork	338802	3.644 msec	
Retry getwork	4988	0.248 sec	
Extra notifies	2949	0.419 sec	
Last EN event	000000000071		
Last NTO event	033022237767		

⊠ ALARM_CLOCK_METERS - DISPLAYS INFORMATION ABOUT THE USER ALARM TIMER FACILITY (HARDCORE INTERFACE FOR timer_manager_)

```
Total metering time    0:20:31

No. alarm clock sims.   2171
Simulation lag             5.245 msecs.
Max. lag                1.7340314e4 msecs.
```

print tuning parameters

- PRINT_TUNING_PARAMETERS - PRINT VALUES FOR SYSTEM CONTROL PARAMETERS. MOST CONTROL THE SCHEDULER

```
Current system tuning parameters:

tefirst                        0.5 seconds
telast                         1. seconds
timax                          8. seconds
priority_sched_inc             80. seconds
min_eligible                   2.
max_eligible                   20.
max_batch_elig                 0
working_set_factor             0.5
working_set_addend             0
deadline_mode                  off
int_q_enabled                  on
post_purge                     on
pre_empt_sample_time           0.04 seconds
gp_at_notify                   off
gp_at_ptlnotify                off
process_initial_quantum        2. seconds
quit_priority                  0.
gv_integration                 4. seconds
realtime_io_priority           on
realtime_io_deadline           0. seconds
realtime_io_quantum            0.005 seconds
notify_timeout_interval        30. seconds
notify_timeout_severity        0
write_limit                    724
```

- THESE ARE "INTERNAL", NORMALLY NEVER CHANGED, AND ONLY PRINTED IF THE -all CONTROL ARGUMENT IS GIVEN

```
stack_truncation               on
stack_truncation_always        off
stk_trunc_block_avg_factor     0.25
trap_invalid_masked            off
meter_ast_locking              off
checksum_filemap               on
```

work class meters

- WORK_CLASS_METERS - DISPLAY THE VARIOUS WORKCLASS PARAMETERS, AND SHOW WHAT RESOURCES EACH WORKCLASS IS CONSUMING.

Total metering time 0:20:38

WC	%GUAR	%MAX	%TCP	V/ELIG	PW	IRESP	IQUANT	RESP	QUANT	P	M	R	I	LCG
0			3.	0.12	3	0.26	2.10	0.26	2.10	P	O	R	I	Init
1			3.	0.09	1	0.25	0.75	0.50	1.00	P	O	R	I	RTime
2	7.		15.	0.44	1					P	O		I	System SysAdm OPR FED
3	32.		44.	0.49	1					P	O		I	SysProg SysDev
4	9.	14.	4.	0.26	1					P	O			SEngr
5	20.		2.	0.46	1					P	O		I	HEngr
6	12.	16.	8.	0.25	1					P	O			MktUS MktFor MktEd
7	3.	7.	4.	0.36	1					P	O			DS-CC
8	6.		0.	0.18	1					P	O		I	OffAuto
9	4.	8.	2.	0.62	1					P	O			Misc Mfg
10	3.		0.	0.55	1					P	O		I	Other
11	4.		2.	0.16	1					P	O		I	Special

TCPU percents (%GUAR) control non-realtime work_classes.

OUTPUT FROM METERING COMMANDS

respons meters

- RESPONSE_METERS - DISPLAYS RESPONSE TIME, BASED ON TERMINAL INTERACTIONS, ON A PER-WORKCLASS BASIS

Total metering time 0:20:36

WC	---Thinks/-- ---Queues--- #	Avg	----Response Times by VCPU Range---- -VCPU Range- From	To	# Int	Avg VCPU	Avg RT	Resp Fact	Load Control Group
0	86	2.70	0.00	0.50	113	0.04	0.42	9.34	Init
	92	0.15	0.50	1.00	3	0.55	5.51	10.00	
			1.00	10.00	3	2.43	14.96	6.15	
			-----	-----	119	0.12	0.91	7.76	
1	35	11.78	0.00	0.50	34	0.11	0.96	8.74	RTime
	39	0.21	0.50	1.00	2	0.83	4.06	4.87	
			-----	-----	36	0.15	1.13	7.55	
2	593	14.90	0.00	0.50	620	0.05	0.49	10.50	System SysAdm OPR FED
	612	0.15	0.50	1.00	28	0.71	3.61	5.10	
			1.00	10.00	39	1.77	8.39	4.74	
			-----	-----	687	0.17	1.07	6.22	
3	2496	6.38	0.00	0.50	2993	0.08	0.66	7.96	SysProg SysDev
	2622	0.17	0.50	1.00	117	0.68	4.22	6.16	
			1.00	10.00	66	2.46	13.47	5.49	
			10.00	99.99	10	56.85	99.99	3.33	
			-----	-----	3186	0.33	1.65	4.96	
4	581	15.82	0.00	0.50	663	0.05	0.71	12.94	SEngr
	590	0.26	0.50	1.00	13	0.64	4.64	7.30	
			1.00	10.00	8	3.47	32.07	9.24	
			-----	-----	684	0.11	1.15	10.88	
5	133	29.51	0.00	0.50	148	0.06	0.83	12.95	HEngr
	141	0.17	0.50	1.00	3	0.69	4.64	6.70	
			1.00	10.00	5	3.15	10.24	3.25	
			10.00	99.99	2	26.69	48.84	1.83	
			-----	-----	158	0.51	1.81	3.55	

OUTPUT FROM METERING COMMANDS

respons meters

6	1180	11.65	0.00	0.50	977	0.05	1.13	20.91	MktUS MktFor MktEd
	1211	0.58	0.50	1.00	24	0.64	4.95	7.74	
			1.00	10.00	16	1.89	11.88	6.29	
			-----	-----	1017	0.10	1.39	14.38	
7	259	14.65	0.00	0.50	292	0.08	1.53	20.17	DS-CC
	287	0.98	0.50	1.00	9	0.71	9.03	12.66	
			1.00	10.00	17	2.04	14.76	7.24	
			10.00	99.99	1	12.22	99.99	9.45	
			-----	-----	319	0.24	2.81	11.86	
8	73	2.69	0.00	0.50	79	0.06	0.31	5.45	OffAuto
	74	0.05	0.50	1.00	2	0.60	6.40	10.62	
			1.00	10.00	1	3.20	6.73	2.10	
			-----	-----	82	0.11	0.54	4.94	
9	94	41.91	0.00	0.50	80	0.11	1.23	11.07	Misc Mfg
	96	0.30	0.50	1.00	11	0.69	4.74	6.91	
			1.00	10.00	13	2.99	12.82	4.30	
			-----	-----	104	0.53	3.05	5.74	
10	7	99.99	0.00	0.50	10	0.11	1.45	13.82	Other
	8	0.21	1.00	10.00	1	3.88	39.11	10.09	
			-----	-----	11	0.45	4.88	10.89	
11	417	12.71	0.00	0.50	445	0.03	0.50	15.32	Special
	420	0.16	0.50	1.00	5	0.57	3.61	6.29	
			1.00	10.00	4	1.65	11.40	6.90	
			-----	-----	454	0.05	0.63	11.93	
All	5954	11.13	0.00	0.50	6454	0.07	0.77	11.40	
	6192	0.29	0.50	1.00	217	0.68	4.54	6.70	
			1.00	10.00	173	2.31	13.11	5.68	
			10.00	99.99	13	48.77	99.99	3.32	
			-----	-----	6857	0.24	1.51	6.39	

86797 calls to meter_response_time   283 invalid transitions.
Overhead =   0.09% (0.052 ms./call)

OUTPUT FROM METERING COMMANDS

system performance graph SPG

- SYSTEM_PERFORMANCE_GRAPH - GIVES A ROUGH SNAPSHOT OF SYSTEM ACTIVITY, SAMPLED PERIODICALLY

S, m, i, t, p, - overhead.

*, white space   good idle   VCPU white space

```
                                        04/02/83 | 1114.3 est Sat
   up= 04/01/83  0855.6 est, sys_hours= 26.3, cpu_hours= 78.9
       cpu= 3, pages= 3419, min_e= 3, max_e= 16, wsa= 0, wsf= 0.50,
       tefirst=  0.13, telast=  0.50, timax= 32.0
1115.00 Qe      .  |   *******|**+******|******mitpS y .   DV
1116.00 Qe     .****|*********|**+******|**iiits    |y   . S DV
1117.00 Q*e*******|***iittpp|  + .    y|     .      |    .  D V
1118.00 Q*e*******|*******iittps+. y   |     .      |    .  D V
1119.00 Qe********|*********|***+*iit  |y    .      |    .   DV
1120.00 Qe********|*********|***+**it  |y    .      |    .   DV
1121.01 Q*e*******|******iitpp  +y     |     .      |    .  D V
1122.00 Qe********|*********|****+iitp|y     .      |    S   DV
1123.00 Qe********|*********|**mi+ts   y     .      |    S   DV
1124.00 Q*e*******|*********|*iii+ps   |y    .      |    .   DV
1125.00 Q*e*******|**********miii+tps  |y    .      |    . D  V
1126.00 Q*e*******|****iiiit|tps+.  y  |     .      |    . D  V
1127.00 Q*e*******|*********|*ii+tp  y|     .      |    .  D V
1128.00 Q*e*******|*********|***+*iiittp     y      |    .  D V
1129.00 Qe *******|*********|***+*****|****iittp|y       .  D V
1130.00 Qe *******|*********|***+*****|**iiittp |y       .   DV
1131.00 Qe *******|*********|***+*****|****iiitt| y      .   DV
1132.00 Qe********|*********|***+*****|****iiit |y       .   DV
1133.00 rQ********|*********|**+******|*iiitts   y       .  D V
1134.00 Qe********|*********|**+******|***iitp   y       .   DV
1135.00 Qe    .   |   *******|**+******|********iit  y.   DV
1136.00 Qe********|*********|***+******iiitts   y|       .  D V
1137.00 rQ********|*********|***+*****|****iitts|y       .  D V
1138.01 Qe********|*********|***+*****iiiitp     y       .   DV
1139.00 Qe********|*********|****+*****iitps     y       S   DVP
1140.02 rQ********|*********|***+*****iiittpp    y       . D VP
```

Qe → eligible processes

r → response time from traffic control meters

→ # quits per some time (minute

+ ~~load~~ ~~units~~ ~~...~~ # users } usually the same

\- load units

Bulk store

Voice I/O

Disk I/O R/W activity per sec

← measured right to left

q = # users in queue

OUTPUT FROM METERING COMMANDS

system performance graph

- SOME CHARACTERS INDICATES PERCENTAGES OF SYSTEM TIME SPENT IN VARIOUS ACTIVITIES. OTHERS INDICATE ACTUAL VALUES

  - PERCENTAGES

    - BLANK - USER PROCESSING. RING ZERO TIME BETWEEN "y" AND THE RIGHT MARGIN, USER RING BETWEEN "s" and "y"

    - SYSTEM SERVICES - SEGMENT FAULTS ("s"), PAGE FAULTS ("p"), TRAFFIC CONTROL ("t"), INTERRUPTS ("i")

    - IDLE TIME - MP IDLE ("m"), NMP IDLE ("*"), ZERO IDLE (BLANKS ON THE LEFT)

  - OTHER VALUES (RELATIVE TO THE LEFT MARGIN)

    - TRAFFIC CONTROL QUEUE LENGTHS - READY QUEUE ("q"), ELIGIBLE QUEUE ("e")

    - USER COUNTS - NUMBER OF USERS ("+"), LOAD UNITS ("-")

    - TRAFFIC CONTROL VALUES - RESPONSE TIME ("r"), QUITS PER MINUTE ("Q"), SCHEDULINGS IN TEN SECONDS ("S")

  - OTHER VALUES (RELATIVE TO THE RIGHT MARGIN)

    - I/O TRAFFIC - DISK I/O PER 100 MILLISECONDS ("D"), VTOC I/O PER 100 MILLISECONDS ("V")

OUTPUT FROM METERING COMMANDS

meter gate

- METER_GATE - SHOWS TIME SPENT CALLING THROUGH SUPERVISOR GATES (HCS_, ETC.)

```
Metering since 04/01/83  0855.6 est Fri.
Total non-idle time at 04/02/83  1101.9 est Sat = 39 hr. 36 min. 57 sec.

Gate meters for hcs_:                      total calls = 6294028.
7 hr. 35 min. 54 sec. or 19.180% spent in calls through gate.

  calls   pcnt     avg   pwait  entry name

 902768   4.50    7.11   0.02  tty_write
  13857   2.29  236.01   2.28  list_dir
 442912   1.19    3.85   0.01  tty_read
 435510   1.01    3.30   0.02  tty_order
 165154   0.97    8.35   0.10  initiate
 120204   0.83    9.84   0.76  make_ptr
  20620   0.51   34.95   0.48  make_seg
  60796   0.50   11.69   0.14  truncate_seg
  45464   0.45   14.25   0.16  status_long
 337615   0.40    1.69   0.01  set_alarm_timer
 532081   0.35    0.95   0.00  read_events
   4085   0.35  122.90   3.57  star_dir_list_
  42080   0.33   11.17   0.02  list_acl
  59354   0.32    7.76   0.01  status_for_backup
 103029   0.32    4.44   0.03  tty_get_line
  61851   0.32    7.33   0.03  tty_write_whole_string
  48621   0.31    9.05   0.35  status_minf
  31637   0.26   11.64   0.04  set_max_length_seg
 150703   0.26    2.41   0.03  terminate_noname
  67267   0.20    4.19   0.03  tty_read_echoed
  14851   0.20   18.77   0.08  quota_get
  21051   0.18   12.48   0.72  get_link_target
  30337   0.18    8.25   0.40  initiate_count
   6288   0.17   39.36   0.25  delentry_seg
 247420   0.16    0.95   0.03  wakeup
  14024   0.16   15.99   0.43  status_
 476759   0.16    0.47   0.00  level_set
  13601   0.13   13.69   0.00  dir_quota_read
  14860   0.13   12.05   0.02  list_dir_acl
  13496   0.13   13.22   0.01  get_max_length
  28194   0.12    6.15   0.41  make_entry
  23725   0.11    6.65   0.09  set_bc_seg
   1416   0.11  108.00   2.08  star_
  13941   0.10   10.68   0.40  get_access_class
  28696   0.10    5.03   0.04  status_mins
  13598   0.09    9.76   0.02  list_inacl_all
```

meter gate

12881	0.08	8.50	0.39	get_user_effmode
46180	0.08	2.33	0.01	tty_read_with_mark
206	0.07	493.77	3.79	star_list_
1109005	0.07	0.09	0.00	level_get

OUTPUT FROM METERING COMMANDS

meter_gate

I HPHCS_ AND IOI_ ARE THE OTHER SUPERVISOR GATES THAT CONSUME SIGNIFICANT RESOURCES

```
Metering since 04/01/83  0855.6 est Fri.
Total non-idle time at 04/02/83  1102.7 est Sat = 39 hr. 37 min. 59 sec.

Gate meters for hphcs_:                    total calls = 70604.
0 hr. 29 min. 51 sec. or  1.256% spent in calls through gate.

  calls   pcnt      avg    pwait  entry name

    869   0.57   928.77     5.40  destroy_process_finish
    911   0.24   381.04    15.09  create_proc
   5944   0.20    48.34     1.04  dir_quota_read
   8326   0.05     8.47     0.04  tty_write_force
   5946   0.05    11.31     0.00  quota_read
    105   0.04   500.68     0.00  flush_core
    246   0.04   208.83     0.17  flush_ast_pool
   1753   0.02    16.94     0.76  star_
   5595   0.02     4.12     0.00  set_backup_dump_time
  31893   0.01     0.60     0.00  set_kst_attributes
      1   0.01 14598.71     8.00  add_scu


Metering since 04/01/83  0855.6 est Fri.
Total non-idle time at 04/02/83  1103.0 est Sat = 39 hr. 38 min. 22 sec.

Gate meters for ioi_:                      total calls = 245985.
0 hr. 12 min. 14 sec. or  0.515% spent in calls through gate.

  calls   pcnt      avg    pwait  entry name

 243976   0.50     2.94     0.00  connect
    248   0.01    65.91     2.09  workspace
    793   0.00     0.58     0.00  get_special_status
    533   0.00     0.54     0.00  set_event
    322   0.00     0.59     0.14  set_status
    105   0.00     0.55     0.00  timeout
      3   0.00     4.38     0.00  set_channel_required
      5   0.00     0.65     0.00  get_detailed_status
```

TOPIC XV

Evolution of Memory Addressing/Management

CONVENTIONAL MEMORY

STRUCTURE

- 1-DIMENSIONAL ADDRESS SPACE USED BY THE SYSTEM

CONVENTIONAL MEMORY

CHARACTERISTICS

- SPACE DIVIDED INTO REGIONS OF VARIOUS SIZES, ONE REGION PER PROCESS/JOB/USER

- REGIONS ARE:

  - PROTECTED FROM ONE ANOTHER BY A BAR OR BY PROTECT KEYS

  - SUBDIVIDED INTO POOLS OF STORAGE USED FOR: PROGRAMS, I/O BUFFERS, MEMORY ALLOCATION AREA, AUTOMATIC VARIABLES, STATIC VARIABLES

- 1-DIMENSIONAL ADDRESS SPACE WIRED DIRECTLY ONTO REAL MEMORY

- PROGRAMS MUST BE LOADED INTO REAL MEMORY

  - MEMORY ALLOCATION FOR REGION

  - PROGRAMS AND BUFFERS ALLOCATED WITHIN REGION

  - PREPARATORY ADDRESS MODIFICATION (ADDRESSES MUST BE MODIFIED TO REFLECT LOCATION WITHIN THE REGION)

CONVENTIONAL MEMORY

CHARACTERISTICS

- LINKAGE EDITING REQUIRED

  - ALL SYMBOLIC REFERENCES MUST BE RESOLVED - IMPLYING THAT ALL REFERENCED PROGRAMS MUST BE LOADED REGARDLESS OF WHETHER OR NOT THEY ARE ACTUALLY NEEDED AT RUN TIME

- EXAMPLES: GCOS, IBM OS/MFT, IBM OS/MVT

CONVENTIONAL MEMORY

PROBLEMS

- SYSTEM ADDRESS SPACE LIMITED TO SIZE OF REAL MEMORY

- USER ADDRESS SPACES (REGIONS) ARE SMALL. THIS MEANS:

  - PROGRAMS MUST BE WRITTEN TO FIT INTO SMALL REGIONS

  - SMALL REGION PROGRAMS OFTEN OPERATE LESS EFFICIENTLY IN THEIR USE OF CPU TIME THAN A SIMILAR PROGRAM DESIGNED FOR A LARGE REGION

  - BECAUSE SMALL REGION OPERATION IS PROGRAMMED IN, SUCH PROGRAMS CANNOT TAKE ADVANTAGE OF MORE MEMORY WHEN IT CAN BE MADE AVAILABLE

  - PROGRAMMER MUST WASTE INGENUITY (AND SYSTEM RESOURCES) TO MAKE PROGRAMS RUN IN A SMALL REGION:

    - WRITING OVERLAY PROGRAMS

    - DIVIDING REGION INTO OPTIONAL-SIZE POOLS

    - PROGRAMMING MECHANISMS TO EXTEND OR SWAP-OUT POOLS WHEN THEY OVERFLOW

  - PROGRAMMER TIME WASTED WHEN PROGRAMS MUST BE CONVERTED TO TAKE ADVANTAGE OF LARGER REGIONS WHEN CONFIGURATION IS INCREASED

CONVENTIONAL MEMORY

PROBLEMS

- REAL MEMORY IS USED INEFFICIENTLY

  - WITHIN A REGION, PROGRAM COMPONENTS OR DATA AREAS NOT REFERENCED STILL OCCUPY REAL MEMORY

  - UNUSED SPACE BETWEEN REGIONS WASTES REAL MEMORY (FRAGMENTATION)

- SCHEMES FOR USING REAL MEMORY MORE EFFICIENTLY ARE COSTLY

  - CPU COST OF MOVING REGIONS TO REDUCE FRAGMENTATION

  - CPU AND I/O COSTS TO SWAP OUT ENTIRE REGIONS TO SHARE REAL MEMORY AMONG MORE USERS

- NO PROTECTION OF DATA WITHIN A REGION

  - PROGRAMMING ERRORS CAN CAUSE UNWANTED WRITING INTO PROGRAM OR DATA AREAS

CONVENTIONAL MEMORY

PROBLEMS

- OVERHEAD AND INCONVENIENCE OF LOADING AND LINKAGE EDITING

- NO SHARING OF PROGRAMS AND DATA BETWEEN REGIONS

  - EACH REGION MUST CONTAIN A COPY OF SHARED DATA (INEFFICIENT USE OF MEMORY)

  - MODIFICATIONS TO SHARED DATA CANNOT EASILY BE REFLECTED IN ALL COPIES

- PHYSICAL INPUT/OUTPUT OPERATIONS ON DISK FILES OFTEN BECOME THE RESPONSIBILITY OF THE PROGRAMMER (REDUCING PRODUCTIVITY)

SINGLE VIRTUAL MEMORY

STRUCTURE

- A LARGE ADDRESS SPACE USED BY THE SYSTEM (EG, 4M WORDS)

- THE ADDRESS SPACE IS LOGICALLY DIVIDED INTO REGIONS OF VARIOUS SIZES, AS IN CONVENTIONAL MEMORY

- THE VIRTUAL MEMORY IS PHYSICALLY DIVIDED INTO PAGES HAVING A FIXED SIZE

- THE ADDRESS SPACE AND THE VIRTUAL MEMORY HAVE THE SAME SIZE

- THE 1-DIMENSIONAL ADDRESS SPACE IS MAPPED DIRECTLY ONTO THE PAGED, 1-DIMENSIONAL VIRTUAL MEMORY

- THE VIRTUAL MEMORY IS MAPPED BY A PAGING ALGORITHM ONTO A SMALLER (EG, 256K WORDS) REAL MEMORY

- PROGRAMS MUST BE LOADED INTO THE VIRTUAL MEMORY AS IN CONVENTIONAL MEMORY SYSTEMS

SINGLE VIRTUAL MEMORY

CHARACTERISTICS

- LINKAGE EDITING REQUIRED AS IN CONVENTIONAL MEMORY SYSTEMS

- EXAMPLES:

  - IBM OS/VS-1 OR OS/VS-2 RELEASE 1

SINGLE VIRTUAL MEMORY

SOLVED PROBLEMS

- THE SYSTEM'S ADDRESS SPACE IS MUCH LARGER THAN REAL MEMORY
  - THE SYSTEM CAN RUN MORE AND/OR LARGER USER REGIONS
- THE USER REGIONS CAN BE LARGER
  - PROGRAMS CAN BE WRITTEN FOR A LARGE REGION TO TAKE ADVANTAGE OF ADDITIONAL MEMORY WHEN IT IS AVAILABLE
- PROGRAMMER PRODUCTIVITY IMPROVES
  - PROGRAMMERS WORRY LESS ABOUT OPTIMIZING MEMORY USAGE
  - CONVERTING PROGRAMS TO USE LARGER MEMORY CONFIGURATIONS OFTEN UNNECESSARY - LESS COMPETITION
- REAL MEMORY USED MORE EFFICIENTLY
  - FEWER UNREFERENCED AREAS OF ADDRESS SPACE OCCUPY REAL MEMORY
  - PAGING ALGORITHM SIMPLIFIES MEMORY MANAGEMENT SCHEMES

SINGLE VIRTUAL MEMORY

PROBLEMS

- MEMORY SWAPPING MAY STILL BE NECESSARY TO SHARE THE ADDRESS SPACE AMONG MANY USERS

- USER REGIONS STILL TOO SMALL TO HANDLE EVERY APPLICATION. SOME MEMORY MANAGEMENT STILL REQUIRED

- NO PROTECTION OF DATA WITHIN A REGION

- OVERHEAD AND INCONVENIENCE OF LOADING AND LINKAGE EDITING

- NO SHARING OF DATA BETWEEN REGIONS

- EXPLICIT DISK I/O STILL REQUIRED TO ACCESS FILES

- THE ADVANTAGES OF SOLVING THE PROBLEMS ABOVE MAY BE OUTWEIGHED BY COSTS IN HARDWARE AND SOFTWARE OF THE PAGING OVERHEAD

MULTIPLE VIRTUAL MEMORIES

STRUCTURE

- THE SYSTEM USES MANY LARGE (4M WORD), 1-DIMENSIONAL ADDRESS SPACES, ONE PER USER REGION

- EACH ADDRESS SPACE CONTAINS THE SUPERVISOR PROGRAMS PLUS ONE LARGE USER REGION (DIVIDED INTO POOLS), PLUS PROGRAMS AND DATA SHARED AMONG ALL REGIONS

- EACH ADDRESS SPACE IS MAPPED DIRECTLY ONTO ITS OWN, PAGED VIRTUAL MEMORY

- EACH VIRTUAL MEMORY IS MAPPED BY A PAGING ALGORITHM ONTO THE SINGLE, SMALLER REAL MEMORY

- PROGRAMS MUST BE LOADED INTO THE VIRTUAL MEMORY AS IN CONVENTIONAL MEMORY SYSTEMS

- LINKAGE EDITING REQUIRED AS IN CONVENTIONAL MEMORY SYSTEMS

MULTIPLE VIRTUAL MEMORIES

CHARACTERISTICS

- EXAMPLES:

  - IBM OS/VS-2 RELEASE 2 (MVS)

MULTIPLE VIRTUAL MEMORIES

SOLVED PROBLEMS

- THE LARGE USER REGIONS CAN HANDLE ALL BUT THE LARGEST PROGRAMS WITHOUT SPECIAL MEMORY MANAGEMENT. PROGRAMMER MUST STILL DIVIDE REGIONS INTO POOLS, HOWEVER, AND SOMETIMES PROVIDE POOL OVERFLOW MECHANISMS

- MEMORY SWAPPING IS NOW UNNECESSARY. PREVIOUSLY SWAPPED REGIONS NOW OCCUPY THEIR OWN ADDRESS SPACES AND ARE PAGED IN AND OUT

- PROGRAMS AND DATA (USUALLY READ-ONLY) CAN BE SHARED BETWEEN REGIONS IN A LIMITED WAY BY OCCUPYING THE SAME POOL IN EVERY ADDRESS SPACE

- NEWLY SOLVED PROBLEMS MAKE PAGING OVERHEAD MORE WORTHWHILE

MULTIPLE VIRTUAL MEMORIES

PROBLEMS

- NO PROTECTION OF DATA WITHIN THE MAJORITY OF A USER'S ADDRESS SPACE (REGION)

- OVERHEAD AND INCONVENIENCE OF LOADING AND LINKAGE EDITING

- GENERAL SHARING OF READ-WRITE DATA STILL NOT POSSIBLE

- EXPLICIT DISK I/O STILL REQUIRED TO ACCESS FILES

MULTICS VIRTUAL MEMORY

STRUCTURE

- THE SYSTEM USES MANY, VERY LARGE (EG, 256M WORDS), 2-DIMENSIONAL ADDRESS SPACES, ONE PER USER PROCESS (REGION)

- EACH ADDRESS SPACE IS DIVIDED INTO SEGMENTS WHICH PERFORM THE SAME FUNCTION AS POOLS IN MULTIPLE VIRTUAL MEMORY SYSTEMS

- SEGMENTS:

  - HAVE VARYING SIZES

  - ARE EXTENDABLE

  - ARE FILES IN THE MULTICS STORAGE SYSTEM

  - ARE ACCESSED AS READ-WRITE, READ-ONLY, EXECUTABLE OR CALLABLE DATA, WITH ACCESS CONTROLLED BY AN ACL, RING BRACKETS AND AN AIM CLASSIFICATION

  - ARE SHARED AMONG ADDRESS SPACES, WITH EACH ADDRESS SPACE HAVING ITS OWN PERMISSION TO ACCESS THE SEGMENT

MULTICS VIRTUAL MEMORY

CHARACTERISTICS

- EACH ADDRESS SPACE INTERSECTS IN VARYING DEGREES WITH EVERY OTHER ADDRESS SPACE

- EACH ADDRESS SPACE IS MAPPED DIRECTLY ONTO ITS OWN PAGED, SEGMENTED VIRTUAL MEMORY OF THE SAME SIZE

- EACH VIRTUAL MEMORY IS MAPPED BY A PAGING ALGORITHM ONTO THE SINGLE, SMALLER REAL MEMORY

- NO LOADING IS REQUIRED SINCE ALL ADDRESSES ARE INTERPRETED AS OFFSETS WITHIN SEGMENTS

- NO LINKAGE EDITING IS REQUIRED SINCE ALL SYMBOLIC REFERENCES ARE RESOLVED AT RUN TIME IF AND WHEN THEY ARE ENCOUNTERED (DYNAMIC LINKING)

MULTICS VIRTUAL MEMORY

SOLVED PROBLEMS

- DATA WITHIN THE ADDRESS SPACE PROTECTED

    - READ OR READ-EXECUTE DATA STORED IN SEPARATE SEGMENTS WHICH ARE PROTECTED FROM MODIFICATION

    - PROGRAMMING ERRORS REFERENCING OUTSIDE ARRAY BOUNDS CANNOT REFERENCE DATA IN ANOTHER SEGMENT

- DATA CAN BE SHARED IN A GENERAL WAY BETWEEN ADDRESS SPACES

    - EACH SEGMENT (NOT A COPY OF THE SEGMENT) CAN APPEAR IN SEVERAL ADDRESS SPACES

    - DIFFERENT PROCESSES CAN HAVE DIFFERENT ACCESS TO THE SAME SEGMENT IN THEIR ADDRESS SPACE

- OVERHEAD AND INCONVENIENCE OF LOADING (SOFTWARE) REPLACED BY AN ADDRESS FORMATION SCHEME (HARDWARE)

    - UNREFERENCED PROGRAMS (AND/OR PAGES OF PROGRAMS) DO NOT REQUIRE MAIN MEMORY SPACE

MULTICS VIRTUAL MEMORY

SOLVED PROBLEMS

- PROGRAMS NEVER REQUIRE PREPARATORY ADDRESS MODIFICATION

- EXPLICIT I/O IS NOT REQUIRED TO ACCESS FILES (SEGMENTS)

  - SEGMENTS CAN BE ACCESSED BY MAKING THEM KNOWN TO THE ADDRESS SPACE AND REFERENCING THE SEGMENT

  - THIS IS CALLED VIRTUAL FILE I/O

- OVERHEAD AND INCONVENIENCE OF LINKAGE EDITING REPLACED BY DYNAMIC LINKING

  - UNREFERENCED PROGRAMS DO NOT REQUIRE LINKING

- ADVANTAGES OF VERY LARGE ADDRESS SPACE, SHARED FILES, DATA PROTECTION, AND VIRTUAL I/O DEFINITELY OUTWEIGH THE COSTS OF PAGING AND SEGMENTATION OVERHEAD

MULTICS VIRTUAL MEMORY

PROBLEMS

- MULTICS FILE I/O IS OFTEN LESS EFFICIENT THEN SPECIAL CASED METHODS BASED ON KNOWN ACCESS PATTERNS

- THE SYSTEM'S SEGMENT SIZE DOES NOT GENERALIZE UPWARD TO HANDLE VERY LARGE DATA BASES. ALTERNATE (KLUDGY) METHODS MUST BE USED (SUCH AS MSF'S)

- THE OVERHEAD TO TOUCH A PAGE OF ONE HUNDRED DIFFERENT SEGMENTS IS CONSIDERABLY MORE THAN THE OVERHEAD TO TOUCH ONE HUNDRED PAGES OF THE SAME SEGMENT. (IE: SPARSE AND INFREQUENT ACCESSING IS EXPENSIVE)

- LACK OF EXPLICIT I/O CONTROL MAKES OVERALL SYSTEM RELIABILITY SUFFER SINCE FULL RECOVERY FROM A SYSTEM CRASH REQUIRES THE SUCCESSFUL FLUSHING OF ALL PAGES FROM MAIN MEMORY

  - hcs_$force_write MAY BE USED BY THOSE APPLICATIONS REQUIRING SUCH RELIABILITY

Massachusetts Institute of Technology

Information Processing Center

Printout of the 59 Entries

of the

Libraries

include.**, hard.source

Which Match the Search Names

add_type.incl.pl1, aim_template.incl.pl1, apte.incl.pl1, aste.incl.pl1, bos_dump.incl.pl1, cmp.incl.pl1, dbm.incl.pl1, dir_acl.incl.pl1, dir_allocation_area.incl.pl1, dir_entry.incl.pl1, dir_header.incl.pl1, dir_ht.incl.pl1, dir_link.incl.pl1, dir_name.incl.pl1, dirlockt.incl.pl1, disk_pack.incl.pl1, dskdcl.incl.pl1, ect_structures.incl.pl1, event_wait_list.incl.pl1, fault_vector.incl.pl1, fgbx.incl.pl1, fs_dev_types.incl.pl1, fs_types.incl.pl1, fs_vol_label.incl.pl1, hc_lock.incl.pl1, itt_entry.incl.pl1, kst.incl.pl1, lock_array.incl.pl1, lvt.incl.pl1, mc.incl.pl1, null_addresses.incl.pl1, ptw.l68.incl.pl1, pv_holdt.incl.pl1, pvt.incl.pl1, pvte.incl.pl1, rnt.incl.pl1, scavenger_data.incl.pl1, scs.incl.pl1, sdw.l68.incl.pl1, sdw_info.incl.pl1, signaller_stack.incl.pl1, slt.incl.pl1, slte.incl.pl1, sst.incl.pl1, sstnt.incl.pl1, stack_0_data.incl.pl1, stack_frame.incl.pl1, stack_header.incl.pl1, stock_seg.incl.pl1, str.incl.pl1, tcm.incl.pl1, vol_map.incl.pl1, vtoc_buffer.incl.pl1, vtoc_header.incl.pl1, vtoc_map.incl.pl1, vtoce.incl.pl1, pds.cds, prds.cds, tc_data.cds

Printed on: 04/01/83 0037.0

Printed by: Sibert.Multics.a

Descriptor: multics_libraries_

APPENDIX A

Hardcore Include Files

This Page Intentionally Left Blank

```
        /* BEGIN INCLUDE FILE add_type.incl.pl1 */
        /* 02/26/75 by Bernard S. Greenberg */

        /* This file provides a structure for checking
                PTW/CME address type fields in PL/I */

dcl 1 add_type unaligned static internal,
          2 core bit (4) init ("1000"b),                /* in core- S/B only in PTW */
          2 disk bit (4) init ("0100"b),                /* Disk address */
          2 pd   bit (4) init ("0010"b),                /* Paging Device */
          2 reserved bit (4) init ("0001"b),            /* Reserved */
          2 non_null bit (4) init ("1111"b);            /* Not null address */

dcl 1 badd_type unaligned based,
    2 (core, disk, pd, reserved) bit (1) unaligned;

        /* END INCLUDE FILE add_type.incl.pl1 */
```

```
/* BEGIN INCLUDE FILE aim_template.incl.pl1 */

/* Created 740723 by PG */
/* Modified 06/28/78 by C. D. Tavares to add rcp privilege */

/* This structure defines the components of both an access
   class and an access authorization as interpreted by the
   Access Isolation Mechanism. */


dcl  1 aim_template aligned based,                      /* authorization/access class template */
       2 categories bit (36),                           /* access categories */
       2 level fixed bin (17) unaligned,                /* sensitivity level */
       2 privileges unaligned,                          /* special access privileges (in authorization only) */
        (3 ipc,                                         /* interprocess communication privilege */
         3 dir,                                         /* directory privilege */
         3 seg,                                         /* segment privilege */
         3 soos,                                        /* security out-of-service privilege */
         3 ring1,                                       /* ring 1 access privilege */
         3 rcp) bit (1),                                /* RCP resource access privilege */
         3 pad bit (12);


/* END INCLUDE FILE aim_template.incl.pl1 */
```

```
/*  BEGIN INCLUDE FILE ... apte.incl.pl1 */


dcl  aptep pointer;

dcl 1 apte based (aptep) aligned,                        /* APT entry declaration for an active (known) process */
      2 thread unaligned,                                /* List thread */
        3 fp bit (18),                                   /* Forward pointer */
        3 bp bit (18),                                   /* Backward pointer */
      2 flags unaligned,                                 /* Flags and miscellaneous */
        3 mbz bit (1),                                   /* This bit must be zero (sentinel bit) */
        3 wakeup_waiting bit (1),                        /* ON if process has received wakeup */
        3 stop_pending bit (1),                          /* ON if process has received stop connect */
        3 pre_empted bit (1),                            /* ON if process is being pre-empted by get_processor */
        3 hproc bit (1),                                 /* ON if process is hardcore process */
        3 loaded bit (1),                                /* ON if required per-process pages are in memory and wired */
        3 eligible bit (1),                              /* ON if process is eligible */
        3 idle bit (1),                                  /* ON if  this is an idle process */
        3 interaction bit (1),                           /* ON if process has interacted recently */
        3 pre_empt_pending bit (1),                      /* ON if process has received pre-empt connect */
        3 default_procs_required bit (1),                /* ON if apte.procs_required is system default */
        3 realtime_burst bit (1),                        /* ON if next eligibility is realtime */
        3 always_loaded bit (1),                         /* ON if process is not to be unloaded */
        3 dbr_loaded bit (1),                            /* ON if DBR is loaded on some CPU */
        3 being_loaded bit (1),                          /* ON if somebody loading this process */
        3 shared_stack_0 bit (1),                        /* ON if a shared stack_0 is assigned */
        3 page_wait_flag bit (1),                        /* flag ON if waiting for page */
        3 firstsw bit (1),                               /* OFF until process is intialized */
        3 state bit (18),                                /* execution state */
      2 page_faults fixed bin (35),                      /* total page faults for the process */
      2 processid bit (36),                              /* bit 0-17: offset of ATPE */
                                                         /* bit 18-35: sequential number */
      2 te fixed bin (35),                               /* virtual time since eligibility award */
      2 ts fixed bin (35),                               /* virtual time since scheduling */
      2 ti fixed bin (35),                               /* virtual time since interaction */
      2 timax fixed bin (35),                            /* maximum value allowed for apte.ti */

/* * * * * * * * * */

      2 ipc_pointers unaligned,
        3 event_thread bit (18),                         /* relative pointer to ITT list */
        3 pad3 bit (18),
      2 ips_message bit (36),                            /* IPS signals pending */
      2 asteps unaligned,                                /* relative ASTE pointers */
        3 pds bit (18),                                  /* PDS (per-process) */
        3 dseg bit (18),                                 /* DSEG (per-process) */
        3 prds bit (18),                                 /* PRDS (per-processor) */
      2 savex7 bit (18) unaligned,                       /* x7 at call to getwork (return point in pxss) */
      2 term_processid bit (36),                         /* process to send wakeup at temination */
```

```
    2 lock_id bit (36),                                 /* File System unqieu ID associated with process */
    2 time_used_clock fixed bin (71),                   /* Total CPU time when process last lost CPU */

/* * * * * * * * * */

    2 wait_event bit (36) aligned,                      /* Event ID process awaiting */
    2 wct_index bit (18) unaligned,                     /* rel offset of WCTE */
    2 flags2 unaligned,
      3 priority_scheduling bit (1),                    /* ON if guaranteed eligibility */
      3 special_wakeups bit (6),                        /* Special wakeup channels */
      3 pad7 bit (7),
      3 batch bit (1),                                  /* ON if absentee */
      3 pr_tag bit (3),                                 /* CPU tag running or last run */
    2 state_change_time fixed bin (71),                 /* Time apte.state last changed */
    2 alarm_event fixed bin (71),                       /* wakeup event for alarm clock manager */
    2 alarm_time_thread bit (18) unaligned,             /* thread of processes with pending alarms */
    2 alarm_time bit (54) unaligned,                    /* wakeup time for alarm */

/* * * * * * */

    2 term_channel fixed bin (71),                      /* wakeup event for account overflow */
    2 ws_size fixed bin,                                /* working set estimate for the process */
    2 temax fixed bin (35),                             /* maximum eligibility slice (vcpu) */
    2 deadline fixed bin (71),                          /* time of next run */
    2 lock bit (18) unaligned,                          /* 0 => APTE locked, unlocked => return point of last unlock */
    2 unusable bit (18) unaligned,                      /* locking routines destroy */
    2 cpu_monitor fixed bin (35),                       /* if not 0, send wakeup to term_processid when virtual cpu
                                                        /* reaches this (units = 1/1024 sec) */
    2 paging_measure fixed bin (71),                    /* cumulative memory units */
    2 access_authorization bit (72),                    /* authorization of this process */
    2 dbr fixed bin (71),                               /* DBR value (constant since DSEG entry-held) */

    2 virtual_cpu_time fixed bin (71),                  /* cumulative virtual CPU time for the process */
    2 ittes_sent fixed bin (18),                        /* Unprocessed ITTs sent by this process */
    2 ittes_got fixed bin (18),                         /* Unprocessed ITTs received by this process */

/*  Cells used to drive and instrument finite-state model for response time
    measurement. Maintained by meter_response_time */

    2 current_response_state fixed bin (17) unaligned,  /* Process state in modle */
    2 pad18 bit (18) unaligned,
    2 number_processing fixed bin (35),                 /* Number interactions */
    2 last_response_state_time fixed bin (71),          /* Clock time at last response state change */
    2 total_processing_time fixed bin (71),             /* Total interaction processing time */

/* * * * * * */

    2 begin_interaction_vcpu fixed bin (71),            /* Virtual cpu at beginning of last interaction */

/*  End of cells for finite-state model */

    2 saved_temax fixed bin (35),                       /* temax at eligibility award */
    2 procs_required bit (8) unaligned,                 /* bit mask of CPUs this process can run */
    2 apad (12) fixed bin (35);
```

```
/*  END INCLUDE FILE ... apte.incl.pl1 */
```

```
/*        BEGIN INCLUDE FILE ...aste.incl.pl1 ... */

/* Template for an AST entry. Length = 12 words. */

/* Words 0 to 7, and 11 are read by PC; they are read and modified by SC.
   Words 8, 9 and 10 are modified by PC; they should never be modified without locking the PC lock */

dcl  astep ptr;

dcl 1 aste based (astep) aligned,

    (2 fp bit (18),                               /* forward  used list rel pointer */
    2 bp bit (18),                                /* backward used list rel pointer */

    2 infl bit (18),                              /* ptr to NEXT  in list of ASTE's of my brothers */
    2 infp bit (18),                              /* ptr to FIRST in list of ASTE's of my children */

    2 strp bit (18),                              /* rel pointer to process trailer */
    2 par_astep bit (18),                         /* rel pointer to parent aste */

    2 uid bit (36),                               /* segment unique id */

    2 msl bit (9),                                /* maximum segment length in 1024 word units */
    2 pvtx fixed bin (8),                         /* physical volume table index */
    2 vtocx fixed bin (17),                       /* vtoc entry index */

    2 usedf bit (1),                              /* ast entry is being used if non-zero */
    2 init bit (1),                               /* used bit - insure 1 lap */
    2 gtus bit (1),                               /* global transparent usage switch */
    2 gtms bit (1),                               /* global transparent modified switch */
    2 hc bit (1),                                 /* hard core segment */
    2 hc_sdw bit (1),                             /* aste with sdw for hardcore seg if non-zero */
    2 any_access_on bit (1),                      /* any sdw allows access, unless write_access_on */
    2 write_access_on bit (1),                    /* any sdw allows write access */
    2 inhibit_cache bit (1),                      /* flag not to reset above bits */
    2 explicit_deact_ok bit (1),                  /* set if user can deactivate seg */
    2 deact_error bit (1),                        /* set if error occurred while deactivating */
    2 hc_part bit (1),                            /* set if pages are in a hardcore partition */
    2 fm_damaged bit (1),                         /* set if filemap checksum was ever bad */
    2 pad1 bit (3),                               /* 00000 */
    2 dius bit (1),                               /* dumper in use switch */
    2 nid bit (1),                                /* if on prevents addtion to incremental dump map */
    2 dmpr_pad bit (1),
    2 ehs bit (1),                                /* entry hold switch */
    2 nqsw bit (1),                               /* no quota switch - no checking for pages of this seg */
    2 dirsw bit (1),                              /* directory switch */
    2 master_dir bit (1),                         /* master dir - a root for the log volume */
    2 volmap_seg bit (1),                         /* volmap_seg for some volume */
    2 tqsw (0:1) bit (1),                         /* terminal quota switch - (0) for non dir pages */
```

```
    2 pad_ic bit (10),                                     /* Used to be aste.ic */

    2 dtu bit (36),                                        /* date and time segment last used */

    2 dtm bit (36),                                        /* date and time segment last modified */


    2 quota (0:1) fixed bin (18) unsigned,                 /* sec storage quota - (0) for non dir pages */

    2 used (0:1) fixed bin (18) unsigned,                  /* sec storage used  - (0) for non dir pages */

    2 csl bit (9),                                         /* current segment length in 1024 words units */
    2 fmchanged bit (1),                                   /* turned on by page if file map changed */
    2 fms bit (1),                                         /* file modified switch */
    2 npfs bit (1),                                        /* no page fault switch */
    2 gtpd bit (1),                                        /* global transparent paging device switch */
    2 dnzp bit (1),                                        /* don't null out if zero page switch */
    2 per_process bit (1),                                 /* use master quota for this entry */
    2 ddnp bit (1),                                        /* don't deposit nulled pages */
    2 pad2 bit (2),
    2 records bit (9),                                     /* number of records used by the seg in sec storage */
    2 np bit (9),                                          /* number of pages in core */


    2 ht_fp bit (18),                                      /* hash table forward rel pointer */
    2 fmchanged1 bit (1),                                  /* value of "fmchanged" saved by pc$get_file_map */
    2 damaged bit (1),                                     /* PC declared segment unusable */
    2 pack_ovfl bit (1),                                   /* page fault on seg would cause pack overflow */
    2 synchronized bit (1),                                /* Data Management synchronized segment */
    2 pad3 bit (6),                                        /* 000000000 */
    2 ptsi bit (2),                                        /* page table size index */
    2 marker bit (6)) unaligned;                           /* marker to indicate last word of ASTE */

dcl  asta (0 : 8000) bit (36*12 /* sst-> sst.astsize */) based aligned;


dcl 1 aste_part aligned based (astep),

    2 one bit (36) unaligned,                              /* fp and bp */
    2 two bit (36*11 - 8) unaligned,                       /* part that has to be zeroed when ASTE is freed */
    2 three bit (8) unaligned;                             /* ptsi and marker */


dcl 1 seg_aste based (astep) aligned,                      /* Overlay because quota is only for dirs */
    2 pad1 bit (8*36),
    2 usage fixed bin (35),                                /* page fault count: overlays quota */
    2 pad2 bit (3*36);

/*        END INCLUDE FILE ... aste.incl.pl1 */
```

```
/* BEGIN INCLUDE FILE ... bos_dump.incl.pl1 ... */
/* Modified 1 September 1976 */
/* Modified 11/11/80 by J. A. Bush for the DPS8/70M CPU */
/* Modified 6/12/81 by Rich Coppola to extend the dps8 extended fault reg to
   15 bits  */
/* Modified 02/23/81, W. Olin Sibert, to describe old and new FDUMP styles */


dcl  dumpptr ptr;                                  /* pointer to following structure */

dcl 1 dump based (dumpptr) aligned,                /* header of dump by fdump */
    2 dump_header aligned like dump_header,

    2 segs (1008),                                 /* segment array */
      3 segno bit (18) unal,                       /* segment number */
      3 length bit (18) unal,                      /* length of segment in sector sized blocks */

    2 amptwregs (0 : 63) bit (36),                 /* assoc. mem. page table word regs */
    2 amptwptrs (0 : 63) bit (36),                 /* assoc. mem. page table word pointers */
    2 amsdwregs (0 : 63) bit (72),                 /* assoc. mem. segment descriptor word registers */
    2 amsdwptrs (0 : 63) bit (36),                 /* assoc. mem. segment descriptor word pointers */

    2 ouhist (0 : 63) bit (72),                    /* operations unit history registers */
    2 cuhist (0 : 63) bit (72),                    /* control unit history registers */
    2 duhist (0 : 63) bit (72),                    /* decimal unit history registers */
    2 auhist (0 : 63) bit (72),                    /* appending unit history registers */

    2 prs (0 : 7) ptr,                             /* pointer registers */

    2 regs aligned like dump_registers,            /* assorted machine registers */

    2 low_order_port bit (3),                      /* from which clock is read */
    2 pad4 bit (36),
    2 mctime fixed bin (52),                       /* time conditions were taken */
    2 pad5 (0 : 3) bit (36),

    2 misc_registers like dump_misc_registers,     /* Assorted registers & processor data */

    2 ptrlen (0 : 7) bit (36),                     /* pointers and lengths for EIS */

    2 coreblocks (0 : 7),
      3 num_first bit (18) unal,                   /* first addr in coreblock */
      3 num_blocks bit (18) unal,                  /* number of blocks used */
    2 pad7 (112) fixed bin;


dcl 1 dump_header aligned based,                   /* Standard header for FDUMP */
    2 words_dumped fixed bin (35),                 /* total words in dump */
    2 valid bit (1),                               /* = 1 if there is a 6180 dump to be had */
```

```
    2 time fixed bin (71),                              /* time of dump */
    2 erfno fixed bin (18),                             /* Error Report Form Number */
    2 num_segs fixed bin,                               /* number of segments dumped */
    2 valid_355 bit (1),                                /* = 1 if there is a dn355 dump to be had */
    2 dumped_355s bit (4),                              /* indicates which 355s were dumped */
    2 time_355 fixed bin (71),                          /* time of 355 dump */
    2 version fixed bin,                                /* currently 2 */
    2 pad0 (5) fixed bin;                               /* pad0 to 16 words */

dcl 1 dump_registers aligned based,                     /* Standard (SREG) arrangement of registers in dump */
   (2 x (0 : 7) bit (18),                               /* index registers */
    2 a bit (36),                                       /* the a register */
    2 q bit (36),                                       /* the q register */
    2 e bit (8),                                        /* the e register */
    2 pad2 bit (28),                                    /* pad */
    2 t bit (27),                                       /* timer register */
    2 pad3 bit (6),                                     /* pad */
    2 ralr bit (3)) unaligned;                          /* ring alarm register */

dcl 1 dump_misc_registers aligned based,
    2 scu (0 : 7) bit (36),                             /* from store control unit instr. */
    2 mcm (0 : 7) bit (72),                             /* memory controller masks every 64 K */
    2 dbr bit (72),                                     /* descriptor segment base register */
    2 intrpts bit (36),                                 /* interrupts */
    2 bar bit (36),                                     /* base address register */
    2 modereg bit (36),                                 /* mode register */
    2 cmodereg bit (36),                                /* cache mode register */
    2 faultreg bit (36),                                /* fault register */
    2 ext_fault_reg bit (15) unaligned,                 /* DPS8 extended fault register */
    2 pad6 bit (21) unaligned;


dcl 1 v1_dump aligned based (dumpptr),                  /* Old version of FDUMP (pre March, 1981) */
    2 dump_header aligned like dump_header,

    2 segs (688),                                       /* segment array */
      3 segno bit (18) unal,                            /* segment number */
      3 length bit (18) unal,                           /* length of segment in sector sized blocks */

    2 amsdwregs (0 : 15) bit (72),                      /* assoc. mem. segment descriptor word registers */
    2 amsdwptrs (0 : 15) bit (36),                      /* assoc. mem. segment descriptor word pointers */
    2 amptwregs (0 : 15) bit (36),                      /* assoc. mem. page table word regs */
    2 amptwptrs (0 : 15) bit (36),                      /* assoc. mem. page table word pointers */
    2 pad1 (0 : 15) bit (36),

    2 ouhist (0 : 15) bit (72),                         /* operations unit history registers */
    2 cuhist (0 : 15) bit (72),                         /* control unit history registers */
    2 auhist (0 : 15) bit (72),                         /* appending unit history registers */
    2 duhist (0 : 15) bit (72),                         /* decimal unit history registers */

    2 prs (0 : 7) ptr,                                  /* pointer registers */

    2 regs aligned like dump_registers,                 /* assorted machine registers */
```

```
    2 mctime fixed bin (52),                                  /* time conditions were taken */
    2 pad4 (0 : 5) bit (36),

    2 misc_registers aligned like dump_misc_registers,        /* Assorted registers */

    2 pad5 bit (36),
    2 ptrlen (0 : 7) bit (36),                                /* pointers and lengths for EIS */
    2 pad6 (15) bit (36),
    2 low_order_port bit (3),                                 /* from which clock was read */

    2 coreblocks (0 : 7),
      3 num_first bit (18) unal,                              /* first addr in coreblock */
      3 num_blocks bit (18) unal;                             /* number of blocks used */

dcl  DUMP_VERSION_1 fixed bin internal static options (constant) init (1);
dcl  DUMP_VERSION_2 fixed bin internal static options (constant) init (2);

/* END INCLUDE FILE ... bos_dump.incl.pl1 ... */
```

```
/* BEGIN INCLUDE FILE cmp.incl.pl1 --- October 1982 */
/* Note: This include file has an ALM counterpart NOT made with cif (for historical reasons). Keep it up to date */

dcl  cmep ptr;                                           /* pointer to core map entry */

dcl 1 cme based (cmep) aligned,                          /* core map entry */
    2 fp bit (18) unaligned,                             /* forward pointer to next entry */
    2 bp bit (18) unaligned,                             /* backward pointer to previous entry */

    2 devadd bit (22) unaligned,                         /* device address of page in the core block */
    2 pad5 bit (1) unaligned,
    2 synch_held bit (1) unaligned,                      /* Page of synchronized seg held in memory */
    2 io bit (1) unaligned,                              /* input/output indicator 1=output, 0=input */
    2 pad2 bit (1) unaligned,
    2 er bit (1) unaligned,                              /* indicates error in previous IO activity */
    2 removing bit (1) unaligned,                        /* core is being removed by reconfiguration */
    2 abs_w bit (1) unaligned,                           /* absolute address must not be changed for page */
    2 abs_usable bit (1) unaligned,                      /* page may be assigned with fixed absolute address */
    2 notify_requested bit (1) unaligned,                /* notify requested on I/O completion */
    2 pad3 bit (1) unaligned,
    2 phm_hedge bit (1) unaligned,                       /* on => pc$flush_core ought write. */
    2 contr bit (3) unaligned,                           /* controller in which core block is located */

    2 ptwp bit (18) unaligned,                           /* pointer to page table word for the page */
    2 astep bit (18) unaligned,                          /* relative AST entry pointer of page */
    2 pin_counter fixed bin (17) unaligned,              /* number of times to skip eviction */
    2 synch_page_entryp bit (18) unaligned;              /* relp to synch.page entry */


dcl 1 cma (0: 1) based aligned like cme;                 /* Core map array */

dcl 1 mcme based (cmep) aligned,                         /* core map entry for extracting DID */
    2 pad bit (36) unaligned,
    2 record_no bit (18) unaligned,                      /* record number of device */
    2 add_type bit (4) unaligned,                        /* see add_type.incl.pl1 */
    2 flags bit (14) unal,
    2 pad1 bit (18) unal;


/* END INCLUDE FILE  cmp.incl.pl1 */
```

```
/* BEGIN INCLUDE FILE ... dbm.incl.pl1  ... Feb 1976 */

dcl  dbm_seg$ ext;
dcl  dbmp ptr;

dcl 1 dbm based (dbmp) aligned,
    2 lock_data,
      3 lock bit (36),
      3 event bit (36),
      3 notify bit (1),
    2 control,
      3 init bit (1) unal,
      3 pad1 bit (35) unal,
    2 stats,
      3 sets fixed bin unal,
      3 resets fixed bin unal,
      3 allocs fixed bin unal,
      3 frees fixed bin unal,
    2 pad2 (2) bit (36),
    2 area area (255*1024 -8);

/* END INCLUDE FILE ... dbm.incl.pl1 */
```

```
/*      BEGIN INCLUDE FILE ... dir_acl.incl.pl1 ... last modified Nov 1975 for nss */

/* Template for an ACL entry. Length = 8 words */

dcl aclep ptr;

dcl 1 acl_entry based (aclep) aligned,                  /* length is 8 words */
      2 frp bit(18) unaligned,                          /* rel ptr to next entry */
      2 brp bit(18) unaligned,                          /* rel ptr to previous entry */

      2 type bit (18) unaligned,                        /* type = dir acl */
      2 size fixed bin (17) unaligned,                  /* size of acl entry */


      2 name unaligned,                                 /* user name associated with this ACL entry */
        3 pers_rp bit(18) unaligned,                    /* name of user */
        3 proj_rp bit(18) unaligned,                    /* project of user */
        3 tag char(1) unaligned,                        /* tag of user */
      2 mode bit (3) unaligned,                         /* mode for userid */
      2 pad24 bit(24) unaligned,

      2 ex_mode bit(36),                                /* extended access modes */

      2 checksum bit (36),                              /* checksum from acl_entry.name */
      2  owner bit (36);                                /* uid of owning entry */

/* Template for a person or project name on ACL. Length = 14 words. */

dcl 1 access_name aligned based,                        /* person or project name */
      2 frp bit(18) unaligned,                          /* rel ptr to next name structure */
      2 brp bit(18) unaligned,                          /* rel ptr to prev name structure */

      2 type bit (18) unaligned,                        /* type = access name */
      2 size fixed bin (17) unaligned,                  /* size of access name */

      2 salv_flag fixed bin(17) unaligned,              /* used by salvager to check for ascii names */
      2 usage fixed bin(17) unaligned,                  /* number of ACL entries that refer to this name */

      2 pad1 bit (36),

      2 name char(32) aligned,                          /* person or project name itself */

      2 checksum bit (36),                              /* checksum from salv_flag */

      2 owner bit (36);                                 /* uid of containing directory */

/*      END INCLUDE FILE ... dir_acl.incl.pl1 */
```

```
/*      BEGIN INCLUDE FILE ... dir_allocation_area.incl.pl1 ... last modified December 1973 */

dcl  areap ptr;

dcl 1 area based (areap) aligned,
      2 nsizes fixed bin (18),                          /* Number of types. */
      2 lu fixed bin (18),                              /* Next available word in area. */
      2 lw fixed bin (18),                              /* Last usable word. */
      2 array (100) aligned,                            /* Array of types. */
        3 fptr bit (18) unaligned,                      /* Free pointer for this size. */
        3 size fixed bin (17) unaligned;                /* Size. */

/*      END INCLUDE FILE ... dir_allocation_area.incl.pl1 */
```

```
/*        BEGIN INCLUDE FILE ... dir_entry.incl.pl1 ...last modified August 1974 for nss */

/* Template for an entry. Length = 38 words */

dcl  ep ptr;

dcl 1 entry based (ep) aligned,

    (2 efrp bit (18),                              /* forward rel ptr to next entry */
    2 ebrp bit (18)) unaligned,                   /* backward rel ptr to previous entry */

    2 type bit (18) unaligned,                     /* type of object = dir entry  */
    2 size fixed bin (17) unaligned,               /* size of dir entry */

    2 uid bit (36),                                /* unique id of entry */

    2 dtem bit (36),                               /* date-time entry modified */

    (2 bs bit (1),                                 /* branch switch = 1 if branch */
    2 pad0 bit (17),
    2 nnames fixed bin (17),                       /* number of names for this entry */

    2 name_frp bit (18),                           /* rel pointer to start of name list */
    2 name_brp bit (18),                           /* rel pointer to end of name list */

    2 author,                                      /* user who created branch */
      3 pers_rp bit (18),                          /* name of user who created branch */
      3 proj_rp bit (18),                          /* project of user who created branch */

      3 tag char (1),                              /* tag of user who created branch */
      3 pad1 char (3),

    2 primary_name bit (504),                      /* first name on name list */

    2 dtd bit (36),                                /* date time dumped */

    2 pad2 bit (36),

/* the declarations below are for branch only */

    2 pvid bit (36),                               /* physical volume id */

    2 vtocx fixed bin (17),                        /* vtoc entry index */
    2 pad3 bit (18),

    2 dirsw bit (1),                               /* = 1 if this is a directory branch */
```

```
    2 oosw bit (1),                                   /* out of service switch  on = 1 */
    2 per_process_sw bit (1),                         /* indicates segment is per process */
    2 copysw bit (1),                                 /* = 1 make copy of segment whenever initiated */
    2 safety_sw bit (1),                              /* if 1 then entry cannot be deleted */
    2 multiple_class bit (1),                         /* segment has multiple security classes */
    2 audit_flag bit (1),                             /* segment must be audited for security */
    2 security_oosw bit (1),                          /* security out of service switch */
    2 entrypt_sw bit (1),                             /* 1 if call limiter is to be enabled */
    2 master_dir bit (1),                             /* TRUE for master directory */
    2 tpd bit (1),                                    /* TRUE if this segment is never to go on the PD */
    2 pad4 bit (11),
    2 entrypt_bound bit (14)) unaligned,              /* call limiter */

    2 access_class bit (72) aligned,                  /* security attributes : level and category */

    (2 ring_brackets (3) bit (3),                     /* ring brackets on segment */
    2 ex_ring_brackets (3) bit (3),                   /* extended ring brackets */
    2 acle_count fixed bin (17),                      /* number of entries on ACL */

    2 acl_frp bit (18),                               /* rel ptr to start of ACL */
    2 acl_brp bit (18),                               /* rel ptr to end of ACL */

    2 bc_author,                                      /* user who last set the bit count */
      3 pers_rp bit (18),                             /* name of user who set the bit count */
      3 proj_rp bit (18),                             /* project of user who set the bit count */

      3 tag char (1),                                 /* tag of user who set the bit count */
      3 pad5 bit (2),
    2 bc fixed bin (24)) unaligned,                   /* bit count for segs, msf indicator for dirs */

    2 sons_lvid bit (36),                             /* logical volume id for immediat inf non dir seg */

    2 pad6 bit (36),

    2 checksum bit (36),                              /* checksum from dtd */

    2 owner bit (36);                                 /* uid of containing directory */

/*        END INCLUDE FILE ... dir_entry.incl.pl1 ... */
```

```
/*        BEGIN INCLUDE FILE ... dir_header.incl.pl1 */
/* Modified 8/74 for NSS */
/* Modified 8/76 to add version number and hash table rel pointer for variable hash table sizes */
/* Modified 3/82 BIM for change pclock */
/* format: style3 */

/* Template for the directory header. Length = 64 words. */

dcl  dp ptr;

dcl 1 dir based (dp) aligned,

   2 modify bit (36),                                  /* Process ID of last modifier */
   2 type bit (18) unaligned,                 /* type of object = dir header */
   2 size fixed bin (17) unaligned,                    /* size of header in words */
    2 dtc (3),                                         /* date-time checked by salvager array */
      3 date bit (36),                                 /* the date */
      3 error bit (36),                                /* what errors were discovered */

    2 uid bit (36),                                    /* uid of the directory           - copied from branch */

    2 pvid bit (36),                                   /* phys vol id of the dir         - copied from branch */

    2 sons_lvid bit (36),                              /* log vol id for inf non dir seg - copied from branch */

    2 access_class bit (72),                           /* security attributes of dir     - copied from branch */

    (2 vtocx fixed bin (17),                           /* vtoc entry index of the dir    - copied from branch */
    2 version_number fixed bin (17),                   /* version number of header */

    2 entryfrp bit (18),                               /* rel ptr to beginning of entry list */
    2 pad2 bit (18),

    2 entrybrp bit (18),                               /* rel ptr to end of entry list */
    2 pad3 bit (18),

    2 pers_frp bit (18),                               /* rel ptr to start of person name list */
    2 proj_frp bit (18),                               /* rel ptr to start of project name list */

    2 pers_brp bit (18),                               /* rel ptr to end of person name list */
    2 proj_brp bit (18),                               /* rel ptr to end of project name list */

    2 seg_count fixed bin (17),                        /* number of non-directory branches */
    2 dir_count fixed bin (17),                        /* number of directory branches */

    2 lcount fixed bin (17),                           /* number of links */
    2 acle_total fixed bin (17),                       /* total number of ACL entries in directory */

    2 arearp bit (18),                                 /* relative pointer to beginning of allocation area */
```

```
    2 per_process_sw bit (1),                    /* indicates dir contains per process segments */
    2 master_dir bit (1),                        /* TRUE if this is a master dir */
    2 force_rpv bit (1),                         /* TRUE if segs must be on RPV */
    2 rehashing bit (1),                         /* TRUE if hash table is being constructed */
    2 pad4 bit (14),

    2 iacl_count (0:7),
      3 seg fixed bin (17),                      /* number of initial acl entries for segs */
      3 dir fixed bin (17),                      /* number of initial acl entries for dir */

    2 iacl (0:7),                                /* pointer to initial ACLs for each ring */
      3 seg_frp bit (18),                        /* rel ptr to start of initial ACL for segs */
      3 seg_brp bit (18),                        /* rel ptr to end of initial ACL for segs */

      3 dir_frp bit (18),                        /* rel ptr to start of initial for dirs */
      3 dir_brp bit (18),                        /* rel ptr to end of initial ACL for dirs */

    2 htsize fixed bin (17),                     /* size of hash table */
    2 hash_table_rp bit (18),                    /* rel ptr to start of hash table */

    2 htused fixed bin (17),                     /* no. of used places in hash table */
    2 pad6 fixed bin (17),

    2 tree_depth fixed bin (17),                 /* number of levels from root of this dir */
    2 pad7 bit (18)) unaligned,

    2 dts bit (36),                              /* date-time directory last salvaged */

    2 master_dir_uid bit (36),                   /* uid of superior master dir */
    2 change_pclock fixed bin (35),              /* up one each call to sum$dirmod */
    2 pad8 (11) bit (36),                        /* pad to make it a 64 word header */
    2 checksum bit (36),                         /* checksummed from uid on */
    2 owner bit (36);                            /* uid of parent dir */

dcl version_number_2 fixed bin int static options (constant) init (2);

/*        END INCLUDE FILE ... dir_header.incl.pl1 */
```

```
/*      BEGIN INCLUDE FILE ... dir_ht.incl.pl1 */


dcl htp ptr;

dcl 1 hash_table based (htp) aligned,                   /* htp = ptr(dp,active_hardcore_data$htrp) */
    2 modify bit (36) unal,
    2 type bit (18) unal,                                /* type = dir hash table */
    2 size fixed bin (17) unal,                          /* size of current dir hash table entry */
    2 name_rp (0:1) bit(18) unal,                        /* rel ptr of name entry  */
    2 checksum bit (36) unal,
    2 owner bit (36) unal;
                                                         /* otherwise rel ptr to name */

/*      END INCLUDE FILE ... dir_ht.incl.pl1 */
```

```
/*      BEGIN INCLUDE FILE ... dir_link.incl.pl1 ... last modified August 1974 for nss */

/* Template for link. Note that it is identical to entry for first 24 words. */


dcl 1 link based (ep) aligned,

    (2 efrp bit (18),                                   /* forward rel ptr to next entry */
    2 ebrp bit (18),                                    /* backward rel ptr to previous entry */

    2 type bit (18),                                    /* type = dir link */
    2 size fixed bin (17),                              /* size of link in words */

    2 uid bit (36),                                     /* unique id of entry */

    2 dtem bit (36),                                    /* date-time entry modified */

    2 bs bit (1),                                       /* entry switch = 1 if entry */
    2 pad0 bit (17),
    2 nnames fixed bin (17),                            /* number of names for this entry */

    2 name_frp bit (18),                                /* rel pointer to start of name list */
    2 name_brp bit (18),                                /* rel pointer to end of name list */

    2 author,                                           /* user who created entry */
      3 pers_rp bit (18),                               /* name of user who created entry */
      3 proj_rp bit (18),                               /* project of user who created entry */

      3 tag char (1),                                   /* tag of user who created entry */
      3 pad1 char (3),

    2 primary_name bit (504),                           /* first name on name list */

    2 dtd bit (36),                                     /* date time dumped */

    2 pad2 bit (36),


/* the declarations below are only applicable to links */

    2 pad3 bit (18),
    2 pathname_size fixed bin (17),                     /* number of characters in pathname */

    2 pathname char (168 refer (pathname_size))) unaligned, /* pathname of link */

    2 checksum bit (36),                                /* checksum from uid */

    2 owner bit (36);                                   /* uid of containing directory */
```

```
/*      END INCLUDE FILE ... dir_link.incl.pl1 */
```

```
/*     BEGIN INCLUDE FILE ... dir_name.incl.pl1 ... last modified Nov 1975 for nss */

/* Template for names of branches or links. Length = 14 words. */

dcl np ptr;

dcl 1 names based aligned,                          /* based on ptr(dp,ep->entry.name_frp) */
      2 fp bit(18) unaligned,                       /* rel ptr to next name */
      2 bp bit(18) unaligned,                       /* rel ptr to prev name */

      2 type bit (18) unaligned,                    /* type = dir name */
      2 size fixed bin (17) unaligned,              /* size of dir name */

      2 entry_rp bit(18) unaligned,                 /* rel ptr to entry */
      2 ht_index fixed bin(17) unaligned,           /* index of hash table entry */

      2 hash_thread bit (18) unal,                  /* relative ptr to next hash  entry */
      2 pad3 bit (18) unal,

      2 name char(32) aligned,

      2 checksum bit (36),                          /* checksum from entry_rp */

      2 owner bit (36);                             /* uid of entry */


/*     END INCLUDE FILE ... dir_name.incl.pl1 */
```

```
/*        BEGIN INCLUDE FILE ..... dirlockt.incl.pl1 ..... */
/* Modified BIM 1/83 cleanup to multi-read lock */

/* format: style3,idind25 */

dcl       dirlockt_seg$            ext;                     /* name of the segment containing the directory locks */

dcl       dirlocktp                ptr;                     /* pointer to the dirlock table */


dcl       1 dirlockt               based (dirlocktp) aligned,
                                                            /* Table of locks for direcyories */
            2 lock                 bit (36),                /* Lock for the table itself */
            2 ind                  fixed bin (35),          /* Event for the above lock */
            2 notify_sw            bit (1),
            2 last                 fixed bin (17),          /* Index of the last entry currently used */
            2 highest_last         fixed bin (17),          /* Highest index ever used */
            2 counter              (1:59) fixed bin (35),   /* count(i) = number of times entry i was used */
            2 dirlock              (1:10000) aligned like dir_lock;
                                                            /* entry for a directory lock */

declare   dir_lock_ptr             pointer;
declare   1 dir_lock               aligned based (dir_lock_ptr),
            2 pid                  bit (36),                /* pid of the process that locked the dir for write */
            2 ind                  bit (36) aligned,        /* uid of the directory - also used as event id */
            2 notify_sw            bit (1) unaligned,       /* ON if one or more processes are waiting for the lock */
            2 salvage_sw           bit (1) unaligned,       /* ON if dir was locked for salvage */
            2 pad1                 bit (34) unaligned,
            2 lock_count           fixed bin (35);          /* POSITIVE --> write_lock */
                                                            /* NEGATIVE --> -number of lockers */
                                                            /* ZERO --> not locked */

/* ..... END dirlockt.incl.pl1 ..... */
```

```
/*       BEGIN INCLUDE FILE...disk_pack.incl.pl1         Last Modified January 1982 for new volume map   */


/*
   All disk packs have the standard layout described below:

   Record  0                   : contains the label, as declared in fs_vol_label.incl.pl1.
   Record  1 to 3              : contains the volume map, as declared in vol_map.incl.pl1
   Record  4 to 5              : contains the dumper bit map, as declared in dumper_bit_map.incl.pl1
   Record  6                   : contains the vtoc map, as declared in vtoc_map.incl.pl1
   Record  7                   : formerly contained bad track list; no longer used.
   Records 8 to n-1            : contain the array of vtoc entries; ( n is specified in the label)
   each record contains 5 192-word vtoc entries. The last 64 words are unused.
   Records n to N-1            : contain the pages of the Multics segments. ( N is specified in the label)

   Sundry partitions may exist within the region n to N-1, withdrawn or not as befits the meaning
   of the particular partition.



   A conceptual declaration for a disk pack could be:

   dcl 1 disk_pack,
   2 label_record             (0 : 0)             bit(36 * 1024),
   2 volume_map_record        (1 : 3)             bit(36 * 1024),
   2 dumper_bit_map_record    (4 : 5)             bit(36 * 1024),
   2 vtoc_map_record          (6 : 6)             bit(36 * 1024),
   2 spare_record             (7 : 7)             bit(36 * 1024),
   2 vtoc_array_records       (8 : n-1),
   3 vtoc_entry ( 5 )                             bit(36 * 192),
   3 unused                                       bit(36 * 64),
   2 Multics_pages_records    (n : N-1)           bit(36 * 1024);



*/

dcl (LABEL_ADDR              init (0),                       /* Address of Volume Label */
     VOLMAP_ADDR             init (1),                       /* Address of first Volume Map record */
     DUMPER_BIT_MAP_ADDR     init (4),                       /* For initial release compaitiblity */
     VTOC_MAP_ADDR           init (6),                       /* Address of first VTOC Map Record */
     VTOC_ORIGIN             init (8),                       /* Address of first record of VTOC */
     SECTORS_PER_VTOCE       init (3),
     VTOCES_PER_RECORD       init (5),
     DEFAULT_HCPART_SIZE     init (1000),                    /* Size of Hardcore Partition */
     MAX_VTOCE_PER_PACK      init (31774))                   /* Limited by size of VTOC Map */
                             fixed bin (17) int static options (constant);
```

```
/*        END INCLUDE FILE...disk_pack.incl.pl1   */
```

```
/* Begin include file ...... dskdcl.incl.pl1 */
/* Structures used by the Disk DIM */

/* format: style4,delnl,insnl,tree,ifthenstmt,indnoniterend */

dcl  disk_seg$ ext;                                         /* disk data segment */

dcl  disksp ptr,                                            /* pointer to disk subsystem info */
     diskp ptr;                                             /* pointer to disk DIM info structure */

dcl  1 disk_data based (disksp) aligned,                    /* disk subsystem information */
       2 subsystems fixed bin,                              /* number of subsystems */
       2 free_offset bit (18),                              /* offset of first unused location in segment */
       2 status_mask bit (36),                              /* mask for checking for disk error */
       2 last_queue_time fixed bin (71),                    /* for dump analysis, to interpret Q times */
       2 pad (2) fixed bin,
       2 array (32),                                        /* per subsystem info */
         (
         3 offset bit (18),                                 /* location of data for this subsystem */
         3 pad bit (18),
         3 name char (4)
         ) unal;                                            /* name of subsystem */

dcl  1 disktab based (diskp) aligned,                       /* control structure for DIM's */
       2 lock bit (36) unal,                                /* data base lock */
       2 nchan fixed bin,                                   /* number of disk channels */
       2 ndrives fixed bin,                                 /* highest disk drive number */
       2 channels_online fixed bin,                         /* number of disk-channels actually in use */
       2 dev_busy bit (64),                                 /* busy bit for each device */
       2 dev_queued bit (64),                               /* requests queued bit for each device */
       2 wq (0:1) like qht,                                 /* wait queue head/tail */
       2 free_q like qht,                                   /* free queue head/tail */
       2 abs_mem_addr fixed bin (26) unsigned,              /* absolute memory address of this structure */
       2 pad fixed bin,
       2 errors fixed bin,                                  /* error count */
       2 ferrors fixed bin,                                 /* fatal error count */
       2 edac_errors fixed bin,                             /* count of EDAC correctable errors */
       2 pg_io_count (0:1) fixed bin,                       /* count of page I/O operations */
       2 vt_io_count (0:1) fixed bin,                       /* count of VTOCE I/O operations */
       2 call_lock_meters like disk_lock_meters,            /* lock meters for call side of DIM */
       2 int_lock_meters like disk_lock_meters,             /* lock meters for interrupt side of DIM */
       2 alloc_wait_meters like disk_lock_meters,           /* meters for queue entry allocations */
       2 run_lock_meters like disk_lock_meters,             /* lock meters for run calls */
       2 pg_wait (0:1) fixed bin (52),                      /* total time spent waiting for page I/O */
       2 vt_wait (0:1) fixed bin (52),                      /* total time spent waiting for VTOCE I/O */
       2 pg_io (0:1) fixed bin (52),                        /* total time spent doing page I/O */
       2 vt_io (0:1) fixed bin (52),                        /* total time spent doing VTOCE I/O */
       2 queue (64) like quentry,                           /* queue entries */
       2 chantab (8) like chantab,                          /* channel information table */
```

```
       2 devtab (0 refer (disktab.ndrives)) like devtab;    /* device information table */
%page;
dcl   qp ptr,                                               /* pointer to queue entry */
      cp ptr;                                               /* pointer to channel information table */

dcl   1 quentry based (qp) aligned,                         /* queue entry */
        (
        2 next bit (18),                                    /* index to next queue entry */
        2 write_sw bit (1),                                 /* non-zero for write operation */
        2 sect_sw bit (1),                                  /* non-zero for single sector operation */
        2 testing bit (1),                                  /* non-zero if quentry is for disk ready test */
        2 retry bit (1),                                    /* non-zero if retry has been performed on broken device */
        2 used bit (1),                                     /* non-zero if queue entry in use */
        2 swap bit (1),
        2 cylinder fixed bin (11),                          /* disk cylinder number */
        2 pdi unsigned fixed bin (6),                       /* pdi of device */
        2 coreadd bit (24),                                 /* memory address for data transfer */
        2 dev unsigned fixed bin (6),                       /* disk device code */
        2 sector bit (21),                                  /* disk sector address */
        2 pad bit (9),
        2 n_sectors fixed bin (6) unsigned,                 /* number of sectors for sector I/O */
        2 time fixed bin (36) unsigned                      /* low-order microsecond clock at queue */
        ) unal;                                             /* time entry was queued */

dcl   1 chantab based (cp) aligned,                         /* channel information table */
        2 chx fixed bin (35),                               /* io_manager channel index */
        2 ioi_ctx fixed bin (35),                           /* ioi channel table index */
        2 statusp ptr,                                      /* pointer to hardware status word */
        2 chanid char (8),                                  /* channel name */
        (
        2 pad0 bit (18),
        2 in_use bit (1),                                   /* non-zero if channel being used */
        2 active bit (1),                                   /* non-zero if channel active */
        2 rsr bit (1),                                      /* non-zero if RSR in progress */
        2 prior bit (1),                                    /* priority of current request */
        2 ioi_use bit (1),                                  /* non-zero if channel usurped by IOI */
        2 inop bit (1),                                     /* non-zero if channel inoperative */
        2 broken bit (1),                                   /* non-zero if channel broken */
        2 action_code bit (2),                              /* saved from status */
        2 pad1 bit (9)
        ) unal,
        (
        2 qrp bit (18),                                     /* rel ptr to queue entry */
        2 pad2 bit (3),
        2 command bit (6),                                  /* peripheral command */
        2 erct fixed bin (8)
        ) unal,                                             /* error retry count */
        2 select_data,                                      /* data passed to IOM on select */
          (
          3 limit bit (12),                                 /* limit on number of sectors */
          3 mbz bit (3),
          3 sector bit (21)
          ) unaligned,                                      /* sector address */
        2 connect_time fixed bin (52),                      /* time of last connect */
        2 connects fixed bin,                               /* count of connects performed */
```

```
        2 detailed_status (0:17) bit (8) unal,          /* detailed status bytes */
        2 rstdcw bit (36),                               /* restore command */
        2 scdcw bit (36),                                /* select command */
        2 sddcw bit (36),                                /* select data xfer */
        2 dcdcw bit (36),                                /* command to read or write */
        2 dddcw bit (36),                                /* data xfer DCW */
        2 dscdcw bit (36),                               /* RSR command */
        2 dsddcw bit (36),                               /* RSR data xfer */
        2 rssdcw bit (36),                               /* RSS command */
        2 status bit (36) aligned;                       /* saved status */
%page;
dcl    1 qht aligned based,                              /* queue head/tail structure */
        2 (head, tail) bit (18) unal;

dcl    dp ptr,                                           /* pointer to device information table */
       pvtdip ptr;                                       /* pointer to dim_info in PVT entry */

dcl    1 devtab based (dp) aligned,                      /* device information table */
        (
        2 pvtx fixed bin (8),                            /* index of PVT entry for device */
        2 inop bit (1),                                  /* device inoperative */
        2 was_broken bit (1),                            /* device previously broken */
        2 broken bit (1),                                /* device down */
        2 abandoned bit (1),                             /* device lost and gone forever */
        2 pad bit (11),
        2 buddy unsigned fixed bin (6),                  /* other device on this spindle or 0 */
        2 pdi unsigned fixed bin (6)
        ) unal,                                          /* primary device index */
        2 queue_count fixed bin (8),                     /* count of requests queued for device */
        2 cylinder fixed bin (11),                       /* current cylinder position */
        2 seek_distance fixed bin (35, 18),              /* average seek distance */
        2 read_count fixed bin,                          /* count of reads */
        2 write_count fixed bin,                         /* count of writes */
        2 time_inop fixed bin (52);                      /* time drive became inoperative */

dcl    1 pvtdi based (pvtdip) aligned,                   /* disk DIM info in PVT entry */
        (
        2 sx fixed bin (11),                             /* structure index */
        2 usable_sect_per_cyl fixed bin (11),            /* # of usable sectors on disk cylinder */
        2 unused_sect_per_cyl fixed bin (11)
        ) unal;                                          /* # of unused sectors at end of cylinder */

dcl    1 disk_lock_meters based aligned,                 /* lock meters for disk DIM */
        2 count fixed bin,                               /* total number of attempts */
        2 waits fixed bin,                               /* number of attempts which required waiting */
        2 wait_time fixed bin (52);                      /* total time spent waiting */

dcl    (
       RST_LISTX init (1),                               /* listx for restore */
       SC_LISTX init (2),                                /* listx for select */
       DSC_LISTX init (6),                               /* listx for RSR */
       RSS_LISTX init (8)
       ) fixed bin (12) static options (constant);      /* listx for RSS */

/* End of include file ...... dskdcl.incl.pl1 */
```

```
/*       BEGIN INCLUDE FILE ... ect_structures.incl.pl1 ... Jan 1981 */

/* format: style3 */
dcl      ect_ptr             ptr;                           /* points to base of Event Channel Table header */
dcl      ectep               ptr;                           /* points to event channel table entry */

dcl      1 ect_header        aligned based (ect_ptr),       /* structure of the Event Channel Table header */
           2 ect_areap       ptr,                           /* pointer to area in which ect entries are allocated */
           2 ect_area_size   fixed bin (19),                /* number of words in ect area */
           2 flags,
             3 call_priority bit (1) unal,                  /* = "0"b if wait chns have priority - default */
                                                            /* = "1"b if call chans have priority */
             3 unused        bit (17) unal,
             3 mask_call_count
                             fixed bin (17) unal,           /* number times event call chans masked */
           2 count           (0:5) fixed bin,               /* totals of entries allocated */
                                                            /* 0 = number of entries, 1 = number of wait channels */
                                                            /* 2 = number of call channels */
                                                            /* 3 = number of call channel messages */
                                                            /* 4 = number of itt messages, 5 = number of messages */
           2 entry_list_ptrs (4),                           /* head and tail of lists in ECT */
                                                            /* 1 = wait channels, 2 = call channels */
                                                            /* 3 = call channel messages, 4 = itt messages */
             3 firstp        ptr,                           /* head of list */
             3 lastp         ptr,                           /* tail of list */
           2 meters,
             3 total_wakeups fixed bin (33),                /* total wakeups sent on all channels */
             3 total_wait_wakeups
                             fixed bin (33),                /* wakeups sent on wait channels */
             3 total_call_wakeups
                             fixed bin (33),                /* wakeups sent on call channels */
           2 seed            fixed bin (33),                /* used to generate uid portion of channel name */
           2 ittes_tossed    fixed bin (33),                /* number invalid ITT messages received, ignored */
           2 fill            (5) fixed bin;                 /* pad to 36 words */

dcl      TOTAL               fixed bin static options (constant) init (0);
dcl      WAIT                fixed bin static options (constant) init (1);
dcl      CALL                fixed bin static options (constant) init (2);
dcl      EV_CALL_MESSAGE     fixed bin static options (constant) init (3);
                                                            /* used to index count and entry_list_ptrs arrays */
dcl      ITT_MESSAGE         fixed bin static options (constant) init (4);
dcl      EV_MESSAGE          fixed bin static options (constant) init (5);


dcl      1 wait_channel      aligned based (ectep),         /* Event wait channel - type = WAIT */
           2 word_0,
             3 unused1       fixed bin (17) unal,
             3 type          fixed bin (17) unal,           /* = WAIT */
           2 next_chanp      ptr unal,                      /* pointer to next wait channel */
```

```
     2 prev_chanp       ptr unal,                        /* pointer to previous wait channel */
     2 word_3,
       3 unused2        bit (1) unal,
       3 inhibit_count  fixed bin (16) unal,             /* number of times message reception has been inhibited */
       3 wakeup_count   fixed bin (18) unal unsigned,    /* number of wakeups received over this channel */
     2 name             bit (72),                        /* event channel name associated with this channel */
     2 first_ev_msgp    ptr unal,                        /* pointer to first message in queue */
     2 last_ev_msgp     ptr unal,                        /* pointer to last message in queue */
     2 unused3          (4) fixed bin;                   /* pad to 12 words */

dcl     1 call_channel      aligned based (ectep),          /* Event call channel - type = CALL */
     2 word_0,
       3 priority       fixed bin (17) unal,             /* indicated priority relative to other call chns */
       3 type           fixed bin (17) unal,             /* = CALL */
     2 next_chanp       ptr unal,                        /* pointer to next call channel */
     2 prev_chanp       ptr unal,                        /* pointer to prev call channel */
     2 word_3,
       3 call_inhibit   bit (1) unal,                    /* = "1"b if call to associated proc in progress */
       3 inhibit_count  fixed bin (16) unal,             /* number of times message reception has been inhibited */
       3 wakeup_count   fixed bin (18) unal unsigned,    /* number of wakeups received over this channel */
     2 name             bit (72),                        /* event channel name associated with this channel */
     2 first_ev_msgp    ptr unal,                        /* pointer to first message in queue */
     2 last_ev_msgp     ptr unal,                        /* pointer to last message in queue */
     2 data_ptr         ptr unal,                        /* pointer to associated data base */
     2 procedure_value,                                  /* procedure to call when message arrives */
       3 procedure_ptr  ptr unal,                        /* pointer to entry point */
       3 environment_ptr
                        ptr unal,                        /* pointer to stack frame */
     2 unused           fixed bin;                       /* pad to 12 words */

dcl     1 event_message     aligned based,                  /* Event message - type = EV_MESSAGE */
     2 word_0,
       3 priority       fixed bin (17) unal,             /* priority of call channel */
       3 type           fixed bin (17) unal,             /* = EV_MESSAGE */
     2 next_ev_msgp     ptr unal,                        /* pointer to next message for this channel */
     2 message_data     like event_message_data aligned,
                                                         /* event message as returned from ipc_$block */
     2 chanp            ptr unal,                        /* pointer to associated event channel */
     2 next_call_msgp   ptr unal,                        /* pointer to next event call channel message */
     2 unused2          (2) fixed bin;                   /* pad to 12 words */

dcl     1 itt_message       aligned based,                  /* Itt message - type = ITT_MESSAGE */
     2 word_0,
       3 unused1        fixed bin (17) unal,
       3 type           fixed bin (17) unal,
     2 next_itt_msgp    ptr unal,                        /* pointer to next itt message entry in ECT currently */
     2 message_data     like event_message_data aligned,
     2 unused2          (4) fixed bin;                   /* pad to 12 words */

dcl     1 event_channel_name
                        aligned based,                   /* description of name of channel */
     2 ecte_ptr         ptr unal,                        /* pointer to channel entry in ECT */
                                                         /* = null if fast channel */
     2 ring             fixed bin (3) unal unsigned,     /* ring number of ECT */
     2 unique_id        fixed bin (33) unal unsigned;    /* identified unique to the process */
```

```
dcl       1 fast_channel_name aligned based,               /* description of name of of fast channel */
            2 ecte_ptr        ptr unal,                    /* = null fast channel  */
                                                           /* ^= null full event channel */
            2 ring            fixed bin (3) unal unsigned, /* target ring number */
            2 mbz             bit (15) unal,
            2 channel_index   fixed bin (17) unal;         /* number of special channel */

dcl       1 event_message_data
                              aligned based,               /* template for event message */
            2 channel_id      fixed bin (71),              /* event channel name */
            2 message         fixed bin (71),              /* 72 bit message associated with wakeup */
            2 sender          bit (36),                    /* process if of sender */
            2 origin,
              3 dev_signal    bit (18) unal,               /* "1"b if device signal */
                                                           /* "0"b if user event */
              3 ring          fixed bin (17) unal;         /* ring of sending process */

/*        END INCLUDE file ... ect_structures.incl.pl1 */
```

```
/* BEGIN INCLUDE FILE ... event_wait_list.incl.pl1 */

/* ipc_$block wait list structure -- Must begin on an even word boundary.

   Written 9-May-79 by M. N. Davidoff.
*/

     declare event_wait_list_n_channels
                                 fixed binary;
     declare event_wait_list_ptr    pointer;

     declare 1 event_wait_list      aligned based (event_wait_list_ptr),
               2 n_channels         fixed binary,            /* number of channels in wait list */
               2 pad                bit (36),
               2 channel_id         (event_wait_list_n_channels refer (event_wait_list.n_channels)) fixed binary (71);
                                                             /* event channels to wait on */

/* END INCLUDE FILE ... event_wait_list.incl.pl1 */
```

```
/* BEGIN INCLUDE FILE ... fault_vector.incl.pl1 ... last modified February 1981 */

dcl  fvp ptr;                                            /* pointer to the fault and interrupt vectors */

dcl 1 fv based (fvp) aligned,                            /* fault and interrupt vectors */
    2 ipair (0: 31),                                     /* interrupt pairs */
      3 scu bit (36),                                    /* SCU instruction */
      3 tra bit (36),                                    /* TRA instruction */
    2 fpair (0: 31),                                     /* fault pairs */
      3 scu bit (36),                                    /* SCU instruction */
      3 tra bit (36),                                    /* TRA instruction */
    2 i_tra_ptr (0: 31) ptr,                             /* ITS pair for interrupt TRA instruction */
    2 i_scu_ptr (0: 31) ptr,                             /* ITS pair for interrupt SCU instruction */
    2 f_tra_ptr (0: 31) ptr,                             /* ITS pairs for fault TRA instruction */
    2 f_scu_ptr (0: 31) ptr;                             /* ITS pairs for fault SCU instruction */

/* Fault Types by fault number                                                                  */

   dcl (FAULT_NO_SDF  init (0),                          /* Shutdown                            */
        FAULT_NO_STR  init (1),                          /* Store                     */
        FAULT_NO_MME  init (2),                          /* Master Mode Entry 1                 */
        FAULT_NO_F1   init (3),                          /* Fault Tag 1                         */
        FAULT_NO_TRO  init (4),                          /* Timer Runout                        */
        FAULT_NO_CMD  init (5),                          /* Command                             */
        FAULT_NO_DRL  init (6),                          /* Derail                              */
        FAULT_NO_LUF  init (7),                          /* Lockup                              */
        FAULT_NO_CON  init (8),                          /* Connect                             */
        FAULT_NO_PAR  init (9),                          /* Parity                              */
        FAULT_NO_IPR  init (10),                         /* Illegal Procedure                   */
        FAULT_NO_ONC  init (11),                         /* Operation Not Complete              */
        FAULT_NO_SUF  init (12),                         /* Startup                             */
        FAULT_NO_OFL  init (13),                         /* Overflow                            */
        FAULT_NO_DIV  init (14),                         /* Divide Check                        */
        FAULT_NO_EXF  init (15),                         /* Execute                             */
        FAULT_NO_DFO  init (16),                         /* Directed Fault 0 (Segment Fault)    */
        FAULT_NO_DF1  init (17),                         /* Directed Fault 1 (Page Fault)       */
        FAULT_NO_DF2  init (18),                         /* Directed Fault 2                    */
        FAULT_NO_DF3  init (19),                         /* Directed Fault 3                    */
        FAULT_NO_ACV  init (20),                         /* Access Violation                    */
        FAULT_NO_MME2 init (21),                         /* Master Mode Entry 2                 */
        FAULT_NO_MME3 init (22),                         /* Master Mode Entry 3                 */
        FAULT_NO_MME4 init (23),                         /* Master Mode Entry 4                 */
        FAULT_NO_F2   init (24),                         /* Fault Tag 2 (Linkage Fault)         */
        FAULT_NO_F3   init (25),                         /* Fault Tag 3                         */
                                                         /* Fault Numbers 26-30 unassigned      */
        FAULT_NO_TRB  init (31)                          /* Trouble                             */

                               ) fixed bin (17) int static options (constant);
```

```
/* END INCLUDE FILE ... fault_vector.incl.pl1 */
```

```
/* BEGIN INCLUDE FILE ... fgbx.incl.pl1 */
/* last modified 5/3/77 by Noel I. Morris        */
/* Modified 8/79 by R.J.C. Kissel to add FNP blast message. */
/* Modified 7/82 BIM for recognizable sentinel field */

/* The contents of this segment are data shared by Multics and BOS.
   This segment occupies the 2nd, 3rd, 4th, and 5th 16-word blocks of the BOS toehold. */

dcl  flagbox$ ext;
dcl  fgbxp ptr;

dcl 1 fgbx based (fgbxp) aligned,
    2 flags (36) bit (1) unal,                          /* communications switches */
    2 slt_segno bit (18),                               /* segment # of the SLT */
    2 pad1 fixed bin,
    2 rtb,                                              /* return to BOS info */
     (3 ssenb bit (1),                                  /* "1"b if storage system enabled */
      3 call_bos bit (1),                               /* "1"b if BOS called by operator */
      3 shut bit (1),                                   /* "1"b if BOS called after shutdown */
      3 mess bit (1),                                   /* "1"b if message has been provided */
      3 alert bit (1),                                  /* "1"b if audible alarm to be sounded */
      3 pad bit (25),
      3 bos_entry fixed bin (5)) unal,                  /* type of entry into BOS
                                                           0 => XED 10002 (BOS entry)
                                                           1 => XED 10004 (Multics entry)
                                                           2 => XED 10000 (manual entry) */
    2 sentinel char (32) aligned,                       /* set by BOS (for now) */
    2 sst_sdw bit (72),                                 /* set by init_sst */
    2 hc_dbr bit (72),                                  /* set by start_cpu, idle DBR */
    2 message char (64),                                /* message for return to BOS */
    2 fnp_blast char (128);                             /* message for FNP use when Multics is down. */

declare FLAGBOX_SENTINEL char (32) init ("Flagbox & Toehold Valid") int static options (constant);

/* END INCLUDE FILE ... fgbx.incl.pl1 */
```

```
/* Begin include file ...... fs_dev_types.incl.pl1 */
/* Modified 5/19/76 by N. I. Morris */
/* Modified 12/27/78 by Michael R. Jordan to correct MSS0500 information */
/* Modified 4/79 by R.J.C. Kissel to add msu0501 information. */

dcl (maxdevt init (7),                                  /* maximum legal devt */
     bulkdevt init (1),                                 /* bulk store devt */
     msu0500devt init (2),                              /* MSU0500 device type */
     msu0451devt init (3),                              /* MSU0451 device type */
     msu0450devt init (3),                              /* MSU0450 device type */
     msu0400devt init (4),                              /* MSU0400 device type */
     dsu191devt init (4),                               /* DSU191 device type */
     dsu190devt init (5),                               /* DSU190 device type */
     dsu181devt init (6),                               /* DSU181 device type */
     msu0501devt init (7)                               /* MSU0501 device type */
     ) fixed bin (4) static options (constant);

dcl  MODEL (10) fixed bin static options (constant) init    /* Known device model numbers */
    (0, 500, 451, 450, 400, 402, 191, 190, 181, 501);

dcl  MODELX (10) fixed bin static options (constant) init   /* translation from model number to device type */
    (1, 2, 3, 3, 4, 4, 4, 5, 6, 7);

dcl  MODELN (7) fixed bin static options (constant) init    /* translation from device type to model number */
    (0, 500, 451, 400, 190, 181, 501);

dcl  device_names (7) char (4) aligned static options (constant) init ( /* device names indexed by device type */
     "bulk", "d500", "d451", "d400", "d190", "d181", "d501");

dcl  media_removable (7) bit (1) static options (constant) init /* ON => demountable pack on device */
    ("0"b, "0"b, "1"b, "1"b, "1"b, "1"b, "0"b);

dcl  shared_spindle (7) bit (1) static options (constant) init /* ON => 2 devices per spindle */
    ("0"b, "1"b, "0"b, "0"b, "0"b, "0"b, "1"b);

dcl  rec_per_dev (7) fixed bin static options (constant) init /* table of # of records on each device */
    (0, 38258, 38258, 19270, 14760, 4444, 67200);

dcl  cyl_per_dev (7) fixed bin static options (constant) init /* table of # of cylinders on each device */
    (0, 814, 814, 410, 410, 202, 840);

dcl  rec_per_cyl (7) fixed bin static options (constant) init /* table of # of records per cylinder on each device */
    (0, 47, 47, 47, 36, 22, 80);

dcl  sect_per_cyl (7) fixed bin static options (constant) init /* table of # of sectors per cylinder on each device */
    (0, 760, 760, 760, 589, 360, 1280);

dcl  sect_per_rec (7) fixed bin static options (constant) init /* table of # of sectors per record on each device */
    (0, 16, 16, 16, 16, 16, 16);
```

```
dcl  tracks_per_cyl (7) fixed bin static options (constant) init /* table of # of tracks per cylinder on each device */
    (0, 19, 19, 19, 19, 20, 20);

dcl  sect_per_track (7) fixed bin static options (constant) init /* table of # of sectors per track on each device */
    (0, 40, 40, 40, 31, 18, 64);

dcl  words_per_sect (7) fixed bin static options (constant) init /* table of # of words per sector on each device */
    (0, 64, 64, 64, 64, 64, 64);

dcl  first_rec_num (7) fixed bin static options (constant) init /* table of # of first record on each device */
    (0, 0, 0, 0, 0, 0, 0);

dcl  last_rec_num (7) fixed bin (18) static options (constant) init /* table of # of last record on each device */
    (0, 38257, 38116, 19128, 14651, 4399, 67199);

dcl  first_sect_num (7) fixed bin (24) static options (constant) init /* table of # of first sector for each device */
    (0, 0, 0, 0, 0, 0, 0);

dcl  last_sect_num (7) fixed bin (24) static options (constant) init /* table of # last sector number for each device */
    (0, 618639, 616359, 309319, 239722, 71999, 1075199);

dcl  first_alt_sect_num (7) fixed bin (24) static options (constant) init /* table of # of first sector of alt partition */
    (0, 638400, 616360, 309320, 239723, 72000, 1075200);

dcl  last_alt_sect_num (7) fixed bin (24) static options (constant) init /* table of # of last sector of alt partition */
    (0, 639919, 618639, 311599, 241489, 72719, 1077759);

dcl  last_physical_sect_num (7) fixed bin (24) static options (constant) init /* table of # of last sector on device (includes T&D c
ylinders) */
    (0, 639919, 619399, 312359, 242249, 72359, 1077759);

dcl  dev_time (7) float bin (27) static options (constant) init /* table of average access times for each device */
    (384e0, 33187e0, 33187e0, 34722e0, 46935e0, 52631e0, 33187e0);

/* End of include file ...... fs_dev_types.incl.pl1 */
```

```
/* BEGIN INCLUDE FILE ... fs_types.incl.pl1 */

dcl  ACCESS_NAME_TYPE bit (18) static options (constant) init ("000001"b3);
dcl  ACLE_TYPE bit (18) static options (constant) init ("000002"b3);
dcl  DIR_HEADER_TYPE bit (18) static options (constant) init ("000003"b3);
dcl  DIR_TYPE bit (18) static options (constant) init ("000004"b3);
dcl  LINK_TYPE bit (18) static options (constant) init ("000005"b3);
dcl  NAME_TYPE bit (18) static options (constant) init ("000006"b3);
dcl  SEG_TYPE bit (18) static options (constant) init ("000007"b3);
dcl  HASH_TABLE_TYPE bit (18) static options (constant) init ("000013"b3);

dcl  access_name_type fixed bin static options (constant) init (1);
dcl  acle_type fixed bin static options (constant) init (2);
dcl  dir_header_type fixed bin static options (constant) init (3);
dcl  dir_type fixed bin static options (constant) init (4);
dcl  link_type fixed bin static options (constant) init (5);
dcl  name_type fixed bin static options (constant) init (6);
dcl  seg_type fixed bin static options (constant) init (7);
dcl  hash_table_type fixed bin static options (constant) init (11);

/* END INCLUDE FILE ... fs_types.incl.pl1 */
```

```
/* BEGIN INCLUDE FILE ... fs_vol_label.incl.pl1 .. last modified January 1982 for new volume map format  */

/* This is the label at fixed location of each physical volume. Length 1 page */

dcl  labelp ptr;

dcl 1 label based (labelp) aligned,

/* First comes data not used by Multics.. for compatibility with GCOS */

    2 gcos (5*64) fixed bin,

/* Now we have the Multics label */

    2 Multics char (32) init ("Multics Storage System Volume"), /* Identifier */
    2 version fixed bin,                              /* Version 1 */
    2 mfg_serial char (32),                           /* Manufacturer's serial number */
    2 pv_name char (32),                              /* Physical volume name. */
    2 lv_name char (32),                              /* Name of logical volume for pack */
    2 pvid bit (36),                                  /* Unique ID of this pack */
    2 lvid bit (36),                                  /* unique ID of its logical vol */
    2 root_pvid bit (36),                             /* unique ID of the pack containing the root. everybody must agree. */
    2 time_registered fixed bin (71),                 /* time imported to system */
    2 n_pv_in_lv fixed bin,                           /* # phys volumes in logical */
    2 vol_size fixed bin,                             /* total size of volume, in records */
    2 vtoc_size fixed bin,                            /* number of recs in fixed area + vtoc */
    2 not_used bit (1) unal,                          /* used to be multiple_class */
    2 private bit (1) unal,                           /* TRUE if was registered as private */
    2 flagpad bit (34) unal,
    2 max_access_class bit (72),                      /* Maximum access class for stuff on volume */
    2 min_access_class bit (72),                      /* Minimum access class for stuff on volume */
    2 password bit (72),                              /* not yet used */
    2 pad1 (16) fixed bin,
    2 time_mounted fixed bin (71),                    /* time mounted */
    2 time_map_updated fixed bin (71),                /* time vmap known good */

/* The next two words overlay time_unmounted on pre-MR10 systems. This
   forces a salvage if an MR10 pack is mounted on an earlier system.
 */
    2 volmap_version fixed bin,                       /* version of volume map (currently 1) */
    2 pad6 fixed bin,

    2 time_salvaged fixed bin (71),                   /* time salvaged */
    2 time_of_boot fixed bin (71),                    /* time of last bootload */
    2 time_unmounted fixed bin (71),                  /* time unmounted cleanly */
    2 last_pvtx fixed bin,                            /* pvtx in that PDMAP */
    2 pad1a (2) fixed bin,
    2 err_hist_size fixed bin,                        /* size of pack error history */
    2 time_last_dmp (3) fixed bin (71),               /* time last completed dump pass started */
```

```
     2 time_last_reloaded fixed bin (71),              /* what it says */
     2 pad2 (40) fixed bin,
     2 root,
       3 here bit (1),                                 /* TRUE if the root is on this pack */
       3 root_vtocx fixed bin (35),                    /* VTOC index of root, if it is here */
       3 shutdown_state fixed bin,                     /* Status of hierarchy */
       3 pad7 bit (1) aligned,
       3 disk_table_vtocx fixed bin,                   /* VTOC index of disk table on RPV */
       3 disk_table_uid bit (36) aligned,              /* UID of disk table */
       3 esd_state fixed bin,                          /* State of esd */
     2 volmap_record fixed bin,                        /* Begin record of volume map */
     2 size_of_volmap fixed bin,                       /* Number of records in volume map */
     2 vtoc_map_record fixed bin,                      /* Begin record of VTOC map */
     2 size_of_vtoc_map fixed bin,                     /* Number of records in VTOC map */
     2 volmap_unit_size fixed bin,                     /* Number of words per volume map section */
     2 vtoc_origin_record fixed bin,                   /* Begin record of VTOC */
     2 dumper_bit_map_record fixed bin,                /* Begin record of dumper bit-map */
     2 vol_trouble_count fixed bin,                    /* Count of inconsistencies found since salvage */
     2 pad3 (52) fixed bin,
     2 nparts fixed bin,                               /* Number of special partitions on pack */
     2 parts (47),
       3 part char (4),                                /* Name of partition */
       3 frec fixed bin,                               /* First record */
       3 nrec fixed bin,                               /* Number of records */
       3 pad5 fixed bin,
     2 pad4 (5*64) fixed bin;

dcl  Multics_ID_String char (32) init ("Multics Storage System Volume") static;

/* END INCLUDE FILE fs_vol_label.incl.pl1 */
```

```
/* Begin include file hc_lock.incl.pl1 BIM 2/82 */
/* Lock format suitable for use with lock$lock_fast, unlock_fast */

/* format: style3 */

declare   lock_ptr            pointer;
declare   1 lock              aligned based (lock_ptr),
            2 pid             bit (36) aligned,           /* holder of lock */
            2 event           bit (36) aligned,           /* event associated with lock */
            2 flags           aligned,
              3 notify_sw     bit (1) unaligned,
              3 pad           bit (35) unaligned;         /* certain locks use this pad, like dirs */

/* End include file hc_lock.incl.pl1 */
```

```
/*       BEGIN INCLUDE FILE ... itt_entry.incl.pl1 ... Feb 1981 */

/* format: style3 */
dcl      itte_ptr            ptr;                           /* pointer to entry in ITT */

dcl      1 itt_entry         aligned based (itte_ptr),      /* declaration of single entry in the ITT */
           2 next_itt_relp   bit (18) unaligned,            /* thread of relative pointers */
           2 pad             bit (18) unaligned,
           2 sender          bit (36),                      /* id of sending process */
           2 origin,                                        /* origin of event message */
             3 dev_signal    bit (18) unaligned,            /* 0 = user-event, 1 = device-signal */
             3 ring          fixed bin (17) unaligned,      /* if user-event, sender's validation ring */
           2 target_id       bit (36),                      /* target process' id */
           2 channel_id      fixed bin (71),                /* target process' event channel */
           2 message         fixed bin (71);                /* event message */

/*       END INCLUDE FILE ... itt_entry.incl.pl1 */
```

```
/*  BEGIN INCLUDE FILE  - - - kst.incl.pl1 - - - last modified March 1976 by R. Bratt */

dcl  pds$kstp ext ptr,
    (kstp, kstep) ptr;

dcl 1 kst aligned based (kstp),                        /* KST header declaration */
    2 lowseg fixed bin (17),                           /* lowest segment number described by kst */
    2 highseg fixed bin (17),                          /* highest segment number described by kst */
    2 highest_used_segno fixed bin (17),               /* highest segment number yet used  */
    2 lvs fixed bin (8),                               /* number of private LVs this process is connected to */
    2 time_of_bootload fixed bin (71),                 /* bootload time during prelinking */
    2 garbage_collections fixed bin (17) unaligned,    /* KST garbage collections */
    2 entries_collected fixed bin (17) unaligned,      /* KST entries recovered by garbage collection */
    2 free_list bit (18) unaligned,                    /* relative pointer to first free kste */
    2 prelinked_ring (7) bit (1) unaligned,            /* rings prelinked in process */
    2 template bit (1) unaligned,                      /* this is a template kst if set */
    2 allow_256K_connect bit (1) unaligned,            /* can use 256K segments */
    2 unused_2 bit (9) unaligned,
    2 uid_hash_bucket (0 : 127) bit (18) unaligned,    /* hash buckets */
    2 kst_entry (0 refer (kst.lowseg):0 refer (kst.highseg)) aligned like kste, /* kst entries */
    2 lv (1:256) bit (36),                             /* private logical volume connection list */
    2 end_of_kst bit (36);

dcl 1 kste based (kstep) aligned,                      /* KST entry declaration */
    2 fp bit (18) unaligned,                           /* forward rel pointer */
    2 segno fixed bin (17) unaligned,                  /* segment number of this kste */
    2 usage_count (0:7) fixed bin (8) unaligned,       /* outstanding initiates/ring */
    2 entryp ptr unaligned,                            /* branch pointer */
    2 uid bit (36) aligned,                            /* unique identifier */
    2 access_information unaligned,
      3 dtbm bit (36),                                 /* date time branch modified */
      3 extended_access bit (33),                      /* extended access from the branch */
      3 access bit (3),                                /* rew */
      3 ex_rb (3) bit (3),                             /* ring brackets from branch */
    2 hdr bit (3) unaligned,                           /* highest detectable ring */
    2 flags unaligned,
      3 dirsw bit (1),                                 /* directory switch */
      3 allow_write bit (1),                           /* set if initiated with write permission */
      3 priv_init bit (1),                             /* privileged initiation */
      3 tms bit (1),                                   /* transparent modification switch */
      3 tus bit (1),                                   /* transparent usage switch */
      3 tpd bit (1),                                   /* transparent paging device switch */
      3 audit bit (1),                                 /* audit switch */
      3 explicit_deact_ok bit (1),                     /* set if I am willing to have a user force deactivate */
      3 pad bit (3),
    2 infcount fixed bin (12) unaligned;               /* if dirsw then inferior count else lv index */

/*  END INCLUDE FILE - - - - - - - - - - - - - kst.incl.pl1 - - - - - - - - - - - - */
```

```
/*     BEGIN INCLUDE FILE ... lock_array.incl.pl1 */
/* modified BIM 2/82 to clean up */
/* format: style3 */

dcl       1 pds$lock_array    (0:19) external aligned like pds_entry;

dcl       1 pds_entry         based,
            2 lock_ptr        pointer,
            2 event           bit (36) aligned,
            2 flags           aligned,
              3 dir_lock      bit (1) unaligned, /* This is a dir read lock */
              3 one_word      bit (1) unaligned, /* lock$wait */
              3 pad           bit (34) unaligned, /* else lock$lock_fast */
            2 caller_ptr      pointer;


/*     END INCLUDE FILE ... lock_array.incl.pl1 */
```

```
/* BEGIN INCLUDE FILE lvt.incl.pl1 -- Written Jan. 1976 by R. Bratt       */
/*
   This include file defines the format of the hardcore Logical Volume Table (LVT).
*/

dcl  lvt$ ext;
dcl  lvtp ptr;
dcl  lvtep ptr;

dcl 1 lvt aligned based (lvtp),
    2 max_lvtex fixed bin (17),                        /* maximum number of LVs describable */
    2 high_water_lvtex fixed bin (17),                 /* highest LVT index assigned */
    2 free_lvtep ptr,                                  /* pointer to  first free lvte  */
    2 pad1 (4) bit (36),
    2 ht (0:63) ptr unal,                              /* lvid hash table */
    2 lvtes (1:1 refer (lvt.max_lvtex)) like lvte;     /* LVT entries */

dcl 1 lvte aligned based (lvtep),                      /* logical volume table entry */
    2 lvtep ptr unaligned,                             /* lvid hash thread */
    2 pvtex fixed bin (17),                            /* thread of mounted PVs */
    2 lvid bit (36),                                   /* logical volume id */
    2 access_class aligned,                            /* access isolation mechanism stuff */
      3 min bit (72),                                  /* minimum access class allowed on LV */
      3 max bit (72),                                  /* maximum access class allowed on volume */
    2 flags unaligned,                                 /* flags */
      3 public bit (1),                                /* => anyone can connect to this LV */
      3 read_only bit (1),                             /* => no writes on this LV */
      3 pad bit (16),
      3 cycle_pvtx fixed bin (17);                     /* pvtx for next per_process seg */

/* END INCLUDE FILE lvt.incl.pl1                                          */
```

/*

```
*/
/* BEGIN INCLUDE FILE  mc.incl.pl1   Created Dec 72 for 6180 - WSS. */
/* Modified 06/07/76 by Greenberg for mc.resignal */
/* Modified 07/07/76 by Morris for fault register data */
/* Modified 08/28/80 by J. A. Bush for the DPS8/70M CVPU */


/* words 0-15 pointer registers */

dcl  mcp ptr;

dcl 1 mc based (mcp) aligned,
      2 prs (0:7) ptr,                                        /* POINTER  REGISTERS */
     (2 regs,                                                 /* registers */
        3 x (0:7) bit (18),                                   /* index registers */
        3 a bit (36),                                         /* accumulator */
        3 q bit (36),                                         /* q-register */
        3 e bit (8),                                          /* exponent */
        3 pad1 bit (28),
        3 t bit (27),                                         /* timer register */
        3 pad2 bit (6),
        3 ralr bit (3),                                       /* ring alarm register */

      2 scu (0:7) bit (36),

      2 mask bit (72),                                        /* mem controller mask at time of fault */
      2 ips_temp bit (36),                                    /* Temporary storage for IPS info */
      2 errcode fixed bin (35),                               /* fault handler's error code */
      2 fim_temp,
        3 unique_index bit (18) unal,                         /* unique index for restarting faults */
        3 resignal bit (1) unal,                              /* recompute signal name with fcode below */
        3 fcode bit (17) unal,                                /* fault code used as index to FIM table and SCT */
      2 fault_reg bit (36),                                   /* fault register */
      2 pad2 bit (1),
      2 cpu_type fixed bin (2) unsigned,                      /* L68 = 0, DPS8/70M = 1 */
      2 ext_fault_reg bit (15),                               /* extended fault reg for DPS8/70M CPU */
      2 fault_time bit (54),                                  /* time of fault */

      2 eis_info (0:7) bit (36)) unaligned;


dcl (apx fixed bin init (0),
     abx fixed bin init (1),
     bpx fixed bin init (2),
     bbx fixed bin init (3),
     lpx fixed bin init (4),
     lbx fixed bin init (5),
     spx fixed bin init (6),
     sbx fixed bin init (7)) internal static;
```

```
dcl  scup ptr;

dcl 1 scu based (scup) aligned,                          /* SCU DATA */


/*        WORD (0)            */

   (2 ppr,                                               /* PROCEDURE POINTER REGISTER */
      3 prr bit (3),                                     /* procedure ring register */
      3 psr bit (15),                                    /* procedure segment register */
      3 p bit (1),                                       /* procedure privileged bit */

    2 apu,                                               /* APPENDING UNIT STATUS */
      3 xsf bit (1),                                     /* ext seg flag - IT modification */
      3 sdwm bit (1),                                    /* match in SDW Ass. Mem. */
      3 sd_on bit (1),                                   /* SDW Ass. Mem.  ON */
      3 ptwm bit (1),                                    /* match in PTW Ass. Mem. */
      3 pt_on bit (1),                                   /* PTW Ass. Mem.  ON */
      3 pi_ap bit (1),                                   /* Instr Fetch or Append cycle */
      3 dsptw bit (1),                                   /* Fetch of DSPTW */
      3 sdwnp bit (1),                                   /* Fetch of SDW non paged */
      3 sdwp bit (1),                                    /* Fetch of SDW paged */
      3 ptw bit (1),                                     /* Fetch of PTW */
      3 ptw2 bit (1),                                    /* Fetch of pre-paged PTW */
      3 fap bit (1),                                     /* Fetch of final address paged */
      3 fanp bit (1),                                    /* Fetch of final address non-paged */
      3 fabs bit (1),                                    /* Fetch of final address absolute */

    2 fault_cntr bit (3),                                /* number of retrys of EIS instructions */


/*        WORD (1)            */

    2 fd,                                                /* FAULT  DATA */
      3 iro bit (1),                                     /* illegal ring order */
      3 oeb bit (1),                                     /* out of execute bracket */
      3 e_off bit (1),                                   /* no execute */
      3 orb bit (1),                                     /* out of read bracket */
      3 r_off bit (1),                                   /* no read */
      3 owb bit (1),                                     /* out of write bracket */
      3 w_off bit (1),                                   /* no write */
      3 no_ga bit (1),                                   /* not a gate */
      3 ocb bit (1),                                     /* out of call bracket */
      3 ocall bit (1),                                   /* outward call */
      3 boc bit (1),                                     /* bad outward call */
      3 inret bit (1),                                   /* inward return */
      3 crt bit (1),                                     /* cross ring transfer */
      3 ralr bit (1),                                    /* ring alarm register */
      3 am_er bit (1),                                   /* associative memory fault */
      3 oosb bit (1),                                    /* out of segment bounds */
      3 paru bit (1),                                    /* processor parity upper */
      3 parl bit (1),                                    /* processor parity lower */
      3 onc_1 bit (1),                                   /* op not complete type 1 */
```

```
      3 onc_2 bit (1),                               /* op not complete type 2 */

    2 port_stat,                                     /* PORT  STATUS */
      3 ial bit (4),                                 /* illegal action lines */
      3 iac bit (3),                                 /* illegal action channel */
      3 con_chan bit (3),                            /* connect channel */

    2 fi_num bit (5),                                /* (fault/interrupt)  number */
    2 fi_flag bit (1),                               /*  1 => fault,  0 => interrupt  */


/*        WORD (2)              */

    2 tpr,                                           /* TEMPORARY POINTER REGISTER */
      3 trr bit (3),                                 /* temporary ring register */
      3 tsr bit (15),                                /* temporary segment register */

    2 pad2 bit (9),

    2 cpu_no bit (3),                                /* CPU number */

    2 delta bit (6),                                 /* tally modification DELTA */


/*        WORD (3)              */

    2 word3 bit (18),

    2 tsr_stat,                                      /* TSR STATUS for 1,2,&3 word instructions */
      3 tsna,                                        /* Word 1 status */
        4 prn bit (3),                               /* Word 1  PR  number */
        4 prv bit (1),                               /* Word 1  PR  valid bit */
      3 tsnb,                                        /* Word 2  status */
        4 prn bit (3),                               /* Word 2  PR  number */
        4 prv bit (1),                               /* Word 2  PR  valid bit */
      3 tsnc,                                        /* Word 3  status */
        4 prn bit (3),                               /* Word 3  PR  number */
        4 prv bit (1),                               /* Word 3  PR  valid bit */

    2 tpr_tbr bit (6),                               /* TPR.TBR  field */


/*        WORD (4)              */

    2 ilc bit (18),                                  /* INSTRUCTION  COUNTER */

    2 ir,                                            /* INDICATOR  REGISTERS */
      3 zero bit (1),                                /* zero indicator */
      3 neg bit (1),                                 /* negative indicator */
      3 carry bit (1),                               /* carryry indicator */
      3 ovfl bit (1),                                /* overflow indicator */
      3 eovf bit (1),                                /* eponent overflow */
      3 eufl bit (1),                                /* exponent underflow */
      3 oflm bit (1),                                /* overflow mask */
      3 tro bit (1),                                 /* tally runout */
```

```
        3 par bit (1),                                  /* parity error */
        3 parm bit (1),                                 /* parity mask */
        3 bm bit (1),                                   /* bar mode */
        3 tru bit (1),                                  /* truncation mode */
        3 mif bit (1),                                  /* multi-word instruction mode */
        3 abs bit (1),                                  /* absolute mode */
        3 pad bit (4),

/*        WORD (5)              */

      2 ca bit (18),                                    /* COMPUTED  ADDRESS */

      2 cu,                                             /* CONTROL UNIT STATUS */
        3 rf bit (1),                                   /* on first cycle of repeat instr */
        3 rpt bit (1),                                  /* repeat instruction */
        3 rd bit (1),                                   /* repeat double instruction */
        3 rl bit (1),                                   /* repeat link instruciton */
        3 pot bit (1),                                  /* IT  modification */
        3 pon bit (1),                                  /* return type instruction */
        3 xde bit (1),                                  /* XDE from Even location */
        3 xdo bit (1),                                  /* XDE from Odd  location */
        3 poa bit (1),                                  /* operation preparation */
        3 rfi bit (1),                                  /* tells CPU to refetch instruction */
        3 its bit (1),                                  /* ITS modification */
        3 if bit (1),                                   /* fault occured during instruction fetch */

      2 cpu_tag bit (6)) unaligned,                     /* computed tag field */

/*        WORDS (6,7)           */

      2 even_inst bit (36),                             /* even instruction of faulting pair */

      2 odd_inst bit (36);                              /* odd  instruction of faulting pair */


/*        ALTERNATE   SCU  DECLARATION            */

dcl 1 scux based (scup) aligned,

     (2 pad0 bit (36),

      2 fd,                                             /* GROUP II  FAULT DATA */
        3 isn bit (1),                                  /* illegal segment number */
        3 ioc bit (1),                                  /* illegal op  code  */
        3 ia_am bit (1),                                /* illegal address - modifier */
        3 isp bit (1),                                  /* illegal slave  procedure */
        3 ipr bit (1),                                  /* illegal procedure */
        3 nea bit (1),                                  /* non existent address */
```

```
      3 oobb bit (1),                              /* out of bounds */
      3 pad bit (29),

    2 pad2 bit (36),

    2 pad3a bit (18),

    2 tsr_stat (0:2),                              /* TSR  STATUS  as an  ARRAY */
      3 prn bit (3),                               /* PR  number */
      3 prv bit (1),                               /* PR  valid bit */

    2 pad3b bit (6)) unaligned,

    2 pad45 (0:1) bit (36),

    2 instr (0:1) bit (36);                        /* Instruction ARRAY */


/*  END INCLUDE FILE   mc.incl.pl1 */
```

```
/* BEGIN INCLUDE FILE  null_addresses.incl.pl1 */

dcl (pc_move_page_table_1_null_addr init          ("3770070"b3),
     pc_move_page_table_2_null_addr init          ("3770100"b3),
     get_aste_null_addr init                      ("3770110"b3),
     make_sdw_null_addr init                      ("3770120"b3),
     put_aste_null_addr init                      ("3770130"b3),
     page_bad_pd_null_addr init                   ("3770150"b3),
     list_deposit_null_addr init                  ("3770160"b3),
     get_file_map_null_addr init                  ("3770170"b3),
     fill_page_table_null_addr init               ("3770200"b3),
     init_sst_null_addr init                      ("3770210"b3),
     get_file_map_vt_null_addr init               ("3770220"b3),
     unprotected_null_addr init                   ("3770230"b3),
     page_parity_null_addr init                   ("3770260"b3),
     page_devparity_null_addr init                ("3770270"b3),
     get_file_map_dumper_non_null_addr init       ("3777720"b3),
     page_bad_null_addr init                      ("3770240"b3),
     page_problem_null_addr init                  ("3770250"b3)) bit (22) aligned static options (constant);

dcl create_vtoce_four_null_addrs fixed bin (71) int static init (-1);/* 777777 777777 777777 777777 */

dcl (create_vtoce_null_addr init                  ("777777"b3),
     update_vtoce_null_addr init                  ("777776"b3),
     truncate_vtoce_fill_null_addr init           ("777775"b3),
     truncate_vtoce_null_addr init                ("777002"b3),
     pv_salv_null_addr init                       ("777004"b3),
     pv_scav_null_addr init                       ("777006"b3),
     volume_reloader_null_addr init               ("777774"b3),
     volume_retriever_null_addr init              ("777773"b3),
     salv_truncate_null_addr init                 ("777005"b3)) bit (18) aligned static options (constant);

/* END INCLUDE FILE null_addresses.incl.pl1 */
```

```
/* ***********************************************************
   *                                                         *
   * Copyright, (C) Honeywell Information Systems Inc., 1982 *
   *                                                         *
   *********************************************************** */

/* PDS - The Process Data Segment

   Last modified (Date and reason):
   2/6/76 by S. Webber Initial coding
   9/17/76 by R. Bratt to add seg_fault, bounds_fault, vtoc_read, and vtoc_write meters.
   11/03/76 by M. Weaver to extend stack header
   04/20/77 by M. Weaver to delete rntp and 7/77 to add name template_pds
   06/07/78 by E. Donner to add ring_events (to prevent delayed ipc wakeups)
   05/10/79 by B. Margulies to eliminate exmode_level
   05/09/79 by Mike Grady to use shared ring 0 stacks
   08/17/79 by J. A. Bush for exp under/overflow restart switches & cache parity diagnostics
   02/28/80 by B. Margulies to use the include file for the default overflow
   08/26/80 by J. A. Bush  for the DPS8/70M CPU
   value.
   02/23/81 by J. Bongiovanni to remove temp_mode_reg (moved to prds$mode_reg_enabled)
   03/81 by E. Donner to remove next_itt and ect_pointers
    3/82 BIM for lock_array cleanup.
   11/82 by J. Bongiovanni to make force_write_limit per-ring
    2/83 by E. N. Kittlitz for hfp_exponent_enabled.
*/


/* ******************************************************
   *                                                    *
   *                                                    *
   * Copyright (c) 1972 by Massachusetts Institute of   *
   * Technology and Honeywell Information Systems, Inc. *
   *                                                    *
   *                                                    *
   ****************************************************** */


/* format: style3,idind25 */
pds:
     proc;

/* This program creates the pds data base */

/* Automatic */

dcl       1 cdsa                   aligned like cds_args;
dcl       code                     fixed bin (35);
```

```
/* Constants */

dcl     pdsname                 char (3) aligned static init ("pds") options (constant);
dcl     exclude_pad             (1) char (32) aligned static options (constant) init ("pad*");

/* Builtins */

dcl     (addr, bin, bit, hbound, mod, null, rel, size, string, unspec)
                                builtin;

/* Entries */

dcl     com_err_                entry options (variable);
dcl     create_data_segment_    entry (ptr, fixed bin (35));
dcl     get_temp_segment_       entry (char (*), ptr, fixed bin (35));
dcl     release_temp_segment_   entry (char (*), ptr, fixed bin (35));
dcl     hcs_$chname_file        entry (char (*), char (*), char (*), char (*), fixed bin (35));
dcl     get_wdir_               entry () returns (char (168));

/* External Static */

dcl     error_table_$segnamedup fixed bin (35) ext;
```

```
dcl     pdsp                    ptr;

dcl     1 pds                   aligned based (pdsp),
          2 page_fault_data     like mc,                /* MC for page faults and timer runouts */
          2 fim_data            like mc,                /* MC for normal faults */
          2 signal_data         aligned like mc,        /* storage for MC being signalled */
          2 history_reg_data    (64) fixed bin (71),    /* this must follow signal data */
          2 process_group_id    char (32),              /* user id for current process */
          2 cpu_time            fixed bin (52),         /* number that when subtracted from clock reading gives
                                                           virtual cpu time */
          2 virtual_delta       fixed bin (52),         /* temporary used in calculating VCPU time */
          2 virtual_time_at_eligibility
                                fixed bin (52),         /* temporary used in calculation of VCPU time */
          2 temp_1              fixed bin (71),         /* temporary */
          2 temp_2              fixed bin (71),         /* temporary */
          2 time_1              fixed bin (52),         /* page fault metering time */
          2 time_v_temp         fixed bin (52),         /* temporary used in calculating VCPU time */
          2 fim_v_temp          fixed bin (52),         /* VCPU temporary for the FIM */
          2 fim_v_delta         fixed bin (71),         /* VCPU temporary for the FIM */
          2 save_history_regs   bit (1) aligned,        /* = "1"b if history registers are to be saved */
          2 hregs_saved         bit (1) aligned,        /* = "1"b if history regs were saved */
          2 last_sp             ptr,                    /* stack pointer at getwork time */
          2 apt_ptr             ptr,                    /* pointer to this process's APT entry */
          2 arg_1               fixed bin (71),         /* argument for pxss */
          2 arg_2               fixed bin (71),         /* argument for pxss */
          2 arg_3               fixed bin (71),         /* argument for pxss */
          2 arg_4               fixed bin (71),         /* argument for pxss */
          2 access_authorization aligned like aim_template,
                                                        /* access authorization for the process */
          2 base_addr_reg       bit (18) aligned,       /* for BAR mode use */
          2 alarm_ring          fixed bin (3),          /* setting for ring alarm register */
          2 pxss_args_invalid   bit (36) aligned,       /* used by pxss masking/arg copying code */
          2 processid           bit (0) unaligned,      /* process ID (added segdef) */
          2 process_id          bit (36) aligned,       /* process ID */
          2 vtime_count         fixed bin,              /* depth counter used in VCPU calculation */
          2 pstep               bit (0) unaligned,      /* (added segdef for dstep) */
          2 dstep               bit (18) aligned,       /* rel pointer to ASTE for dseg */
          2 wakeup_flag         bit (36) aligned,       /* flag indicating type of wakeup */
          2 pc_call             bit (36) aligned,       /* flag saying type of wait */
          2 audit_flags         bit (36) aligned,       /* bits indicating types of auditing to do */
          2 quota_inhib         bit (36) aligned,       /* ON if quota checking to be inhibited */
          2 pd_page_faults      fixed bin,              /* faults from paging device */
          2 page_waits          fixed bin,              /* page faults */
          2 number_of_pages_in_use
                                fixed bin,              /* used in calculating memory units */
          2 post_purged         fixed bin,              /* number of post purgings */
          2 connect_pending     bit (1) aligned,        /* turned on for delayed connects to be resent by fim */
          2 segment_faults      fixed bin (35),         /* count of segment faults taken by this process */
          2 bounds_faults       fixed bin (35),         /* count of bounds faults taken by this process */
          2 vtoc_reads          fixed bin (35),         /* vtoc read I/Os done for this process */
          2 vtoc_writes         fixed bin (35),         /* vtoc write I/Os done for this process */
          2 mc_trace_seg        fixed bin,              /* seg number of object segment being traced */
          2 mc_trace_sw         bit (2) aligned,        /* switch for M. C. Tracing "11"b => trace on */
```

```
2 stack_0_sdwp          ptr aligned,              /* ptr to stack sdw in dseg */
2 stack_0_ptr           ptr aligned,              /* ptr to base of ring 0 stack (wired for esd) */
2 tc_argp               ptr,                      /* arg ptr used by tc */
2 tc_mask               bit (72) aligned,         /* save tc mask */
2 exp_undfl_rest        bit (1) aligned,          /* switch for restarting exp underflows from the fim */
2 exp_ovfl_rest         bit (1) aligned,          /* switch for restarting exp overflows from the fim */
2 eovfl_value           bit (72) aligned,         /* value to load when restarting exp overflows from the fim */
2 cpar_err_data         bit (72) aligned,         /* cache parity error data (from cache) */
2 cpar_mem_data         bit (72) aligned,         /* cache parity error data (from memory) */
2 cpar_info             bit (36) aligned,         /* diagnose flag, cache level and absaddr # */
2 hfp_exponent_enabled  bit (1) aligned,          /* user allowed to set IR hex exp bit */
2 pad_for_trace_mod16   (14) fixed bin,
2 trace                 (306) fixed bin (71),     /* system trace data */
                                                  /* pds$trace + 16 defines the pds for idle procs */
2 timer_time_out        fixed bin (52),           /* time out time for the process */
2 timer_channel         fixed bin (71),           /* event channel for time out event */
2 term_channel          fixed bin (71),           /* channel used to signal process termination */
2 term_proc             bit (36) aligned,         /* process ID of process to signal term process */
2 pl1_machine           fixed bin,                /* nonzero if we do pl1-like things */
2 validation_level      fixed bin (3),
2 condition_name        aligned,                  /* ACC string for condition name */
  3 len                 fixed bin (8) unaligned,
  3 chars               char (31) unaligned,
2 pad_obsolete          bit (36) aligned,
2 ips_mask              (0:7) bit (35) aligned,   /* IPS masks */
2 auto_mask             (0:7) bit (36) aligned,   /* array of automatic masks for IPS signals */
2 ring_alarm_val        (0:7) fixed bin,          /* used in checking validation level changes */
2 lock_id               bit (36) aligned,         /* UID used in some locking */
2 mc_trace_buf          ptr unaligned,            /* packed ptr to mc_trace wired buffer */
2 pad_end_of_page_0     bit (0) unaligned,
2 pathname_am           aligned like pam,         /* pathname associative memory */
2 initial_procedure     ptr,                      /* first procedure executed in a new process */
2 account_id            char (32) aligned,        /* not used yet */
2 lock_array            (0:19) aligned like pds_entry,
2 access_name           aligned,                  /* alternate form of process group id */
  3 user                char (32) aligned,
  3 project             char (32) aligned,
  3 tag                 char (32) aligned,
2 home_dir              char (168) aligned,       /* home directory */
2 process_dir_name      char (32) aligned,        /* name of process directory */
2 wdir                  (0:7) ptr,                /* pointers to per-ring working directories */
2 wdir_uid              (0:7) bit (36) aligned,   /* UID of per-ring working directories */
2 transparent           bit (36) aligned,         /* transparent usage, mod, pd switch */
2 itt_head              bit (18) aligned,         /* top of present ITT list */
2 max_access_authorization
                        aligned like aim_template,
                                                  /* max authorization this user can attain */
2 stacks                (0:7) ptr,                /* per-ring stack pointers */
2 kstp                  ptr,                      /* pointer to start of KST */
2 events_pending        bit (36) aligned,         /* special wakeups pending */
2 special_channels      bit (36) aligned,         /* special channels assigned */
2 event_masks           (7) bit (36) aligned,     /* per-ring mask for special channels */
2 initial_ring          fixed bin (3),            /* initial ring of execution for the process */
2 interrupt_ring        fixed bin (3),            /* lowest ring in which IPS interrupts are allowed */
2 highest_ring          fixed bin (3),            /* highest ring in which process can run */
```

```
          2 prelinked_ring        bit (8) aligned,          /* bit(i) is ON if ring (i) is prelinked */
          2 unique_scu_index      bit (36) aligned,         /* used to tag MC */
          2 max_lot_size          (0:7) fixed bin,          /* sizes lots can grow to */
          2 lot_stack_size        (0:7) fixed bin,          /* size of lot in stack (0 -> lot not in stack) */
          2 clr_stack_size        (0:7) fixed bin,          /* size of CLR in stack */
          2 network_ptbl_idx      fixed bin,                /* index into NCP's process table */
          2 link_meters_bins      (4) fixed bin,            /* histograms of linkage faults */
          2 link_meters_times     (4) fixed bin (30),       /* histogram of linkage fault times */
          2 link_meters_pgwaits   (4) fixed bin,            /* histogram of linkage faults PF's */
          2 dmpr_copy_dirsegp     ptr,                      /* ptr to temp segment into which dirs are copied */
          2 dmpr_pvid             bit (36),                 /* pvid of volume being dumped */
          2 dmpr_pvtx             fixed bin,                /* pvtx of volume being dumped  */
          2 first_call            fixed bin,                /* ON until leave ring zero once */
          2 mc_save_area          bit (18) aligned,         /* rel pointer to start of saved MC area */
          2 mc_save_ptr           bit (18) aligned,         /* ptr to next mc save place */
          2 mc_save_limit         bit (18) aligned,         /* max address where MC can be saved */
          2 useable_lot           bit (8) aligned,          /* indicates whether lot can be referenced */
          2 ring_events           bit (36) aligned,         /* per-ring indicator that itt messages copied to ect */
          2 force_write_limit (0:7)
                                  fixed bin,                /* limit on force-writing */
                                                            /* Following must be doubleword aligned! */
          2 ipc_vars              aligned,                  /* holds state of fast_hc_ipc at block */
            3 ap                  pointer unal,
            3 retsw               fixed bin (35),
            3 save_entry_ret      fixed bin (35),
            3 truncated_stacks    fixed bin (35),
            3 chan                fixed bin (71),
            3 block_start_steps   fixed bin (35),
            3 stk_temp            fixed bin (35),
          2 ipc_block_return      bit (36),                 /* ipc block return address */
          2 avg_block_steps       fixed bin (35, 18),
          2 block_lock_count      fixed bin,                /* count of locks held */
          2 pad_for_data_mod16    (13) fixed bin (35),
          2 data                  bit (0) aligned;          /* to mark end of PDS for MC save area */

%include pathname_am;
%include exponent_control_info;
%include lock_array;
```

```
        call get_temp_segment_ ("pds", pdsp, code);

/* Now begins the initialization */

        pds.process_group_id = "Initializer.SysDaemon.z";

        pds.access_authorization.categories = (18)"0"b;
        pds.access_authorization.level = 0;
        pds.access_authorization.dir = "1"b;              /* for initializer */
        pds.access_authorization.seg = "1"b;
        pds.access_authorization.rcp = "1"b;
        pds.access_authorization.ipc = "1"b;
        pds.access_authorization.soos = "1"b;             /* .. */

        pds.max_access_authorization.categories = (18)"1"b || (18)"0"b;
        pds.max_access_authorization.level = 7;

        pds.vtime_count = -1;
        pds.process_id = (36)"1"b;
        pds.lock_id = (36)"1"b;
        pds.pl1_machine = 1;
        pds.ips_mask (*) = (35)"1"b;
        pds.force_write_limit (*) = 1;

        pds.save_history_regs, pds.hregs_saved = "0"b;
        pds.history_reg_data (*) = 0;

        pds.mc_trace_buf = null;
        pds.mc_trace_sw = "0"b;
        pds.mc_trace_seg = 0;

        pds.eovfl_value = unspec (Default_exponent_control_overflow_value);
                                                          /* set default exp overflow restart value */
        pds.exp_ovfl_rest, pds.exp_undfl_rest = "0"b;

        pds.stack_0_sdwp = null;
        pds.stack_0_ptr = null;
        pds.pad_for_trace_mod16 (*) = 0;

        trace_ptr = addr (pds.trace);
        trace.last_available = divide (hbound (pds.trace, 1) * size (page_trace_entry) - 8, 2, 17, 0);

        pds.initial_procedure = null;
        pds.lock_array (*).lock_ptr = null;
        pds.lock_array (*).caller_ptr = null;
        pds.lock_array (*).event = ""b;

        pds.access_name.user = "Initializer";
        pds.access_name.project = "SysDaemon";
        pds.access_name.tag = "z";

        pds.home_dir = ">system_control_1";
        pds.process_dir_name = ">process_dir_dir>!zzzzzzzbBBBBBB";
```

```
	pds.wdir (*) = null;
	pds.wdir_uid (*) = "0"b;


	pds.stacks (*) = null;

	pds.dmpr_pvid = "0"b;
	pds.dmpr_pvtx = 0;
	pds.dmpr_copy_dirsegp = null;

	pds.kstp = null;
	pds.first_call = 1;
	pds.initial_ring = 1;
	pds.interrupt_ring = 4;
	pds.highest_ring = 7;

	pds.max_lot_size (*) = 1024;

	pds.mc_save_area = rel (addr (pds.data));
	pds.mc_save_ptr = rel (addr (pds.data));
	pds.mc_save_limit = bit (bin (4096, 18), 18);	/* Allow for as many as fit in 4K. */

/* Now make some checks on alignment of certain variables */

	call check (addr (pds.ipc_vars), "ipc_vars", 2);
	call check (addr (pds.page_fault_data), "page_fault_data", 16);
	call check (addr (pds.trace), "trace", 16);
	call check (addr (pds.signal_data), "signal_data", 16);
	call check (addr (pds.lock_array), "lock_array", 2);
	call check (addr (pds.data), "data", 16);
	if bin (rel (addr (pds.pad_end_of_page_0)), 18) ^= 1024
	then call com_err_ (0, pdsname, "Wired portion must end at 1024");

/* Now set up call to create data base */

	cdsa.sections (1).p = addr (pds);
	cdsa.sections (1).len = size (pds);
	cdsa.sections (1).struct_name = "pds";

	cdsa.seg_name = "pds";
	cdsa.num_exclude_names = 1;
	cdsa.exclude_array_ptr = addr (exclude_pad);

	string (cdsa.switches) = "0"b;
	cdsa.switches.have_text = "1"b;

	call create_data_segment_ (addr (cdsa), code);

	call release_temp_segment_ ("pds", pdsp, code);

	call hcs_$chname_file (get_wdir_ (), "pds", "", "template_pds", code);
	if code ^= 0
	then if code ^= error_table_$segnamedup
```

```
then call com_err_ (code, pdsname, "Unable to add name template_pds.");
```

```
check:
     proc (where, message, modulo);

dcl       where                    ptr;
dcl       message                  char (*);
dcl       modulo                   fixed bin;
dcl       remainder                fixed bin;

          remainder = mod (bin (rel (where), 18), modulo);
          if remainder ^= 0
          then call com_err_ (0, pdsname, "The variable ^a is ^d words away from being aligned on a ^d-word boundary.",
                    message, (modulo - remainder), modulo);

     end check;
```

```
%include cds_args;
```

```
%include sys_trace;

%include aim_template;

%include mc;

     end pds;
```

```
/* PRDS - The Processor Data Segment and Processor Stack.
   /* Last modified (Date and reason):
   2/6/76 by S. Webber Initial coding
   6/15/77 by M. Weaver to  null signal and sct pointers
   8/25/80 by J. A. Bush for the dps8/70m cpu
   2/22/81 by J. Bongiovanni for fast_connect_code
   6/27/81 by J. Bongiovanni for idle_temp
*/


/* ******************************************************
   *                                                    *
   *                                                    *
   * Copyright (c) 1972 by Massachusetts Institute of   *
   * Technology and Honeywell Information Systems, Inc. *
   *                                                    *
   *                                                    *
   ****************************************************** */


prds: proc;

/* This program creates the prds data base */

/* Automatic */

dcl  i fixed bin;
dcl 1 cdsa aligned like cds_args;
dcl  code fixed bin (35);

/* Static */

dcl  prdsname char (4) aligned static init ("prds") options (constant);
dcl  exclude_pad (1) char (32) aligned static options (constant) init ("pad*");

/* The following must correspond to the size of the fast connect code in
   fast_connect_init                                                        */

dcl  FAST_CONNECT_CODE_WORDS init (62) fixed bin int static options (constant);

/* Builtins */

dcl (addr, baseptr, bin, mod, null, ptr, rel, size, string, unspec) builtin;

/* Entries */

dcl  com_err_ entry options (variable);
dcl  create_data_segment_ entry (ptr, fixed bin (35));
dcl  get_temp_segment_ entry (char (*), ptr, fixed bin (35));
```

```
dcl  release_temp_segment_ entry (char (*), ptr, fixed bin (35));
```

```
dcl  prdsp ptr;

dcl 1 prds aligned based (prdsp),
    2 header aligned like stack_header,             /* standard stack header */
    2 interrupt_data aligned like mc,                /* MC for interrupts */
    2 fim_data aligned like mc,                      /* MC for connect faults, timer runouts */
    2 sys_trouble_data aligned like mc,              /* MC for saved sys trouble data */
    2 ignore_data aligned like scu,                  /* for SCU data to be ignored at certain times */
    2 iitemp fixed bin (71),                         /* temporary used by ii (surprise!) */
    2 last_recorded_time fixed bin (71),             /* used by traffic control */
    2 idle_ptr ptr,                                  /* pointer to idle process APTE for this processor */
    2 simulated_mask fixed bin (71),                 /* simulated system controller mask register */
    2 am_data bit (0),                               /* to get addr of associative memory data block */
    2 ptw_am_regs (4*16) fixed bin (35),             /* page table regs (4 sets of 16 for dps8/70m) */
    2 ptw_am_ptrs (4*16) fixed bin (35),             /* page table pointers (4 sets of 16 for dps8/70m) */
    2 sdw_am_regs (4*16) fixed bin (71),             /* segment desc. regs (4 sets of 16 for dps8/70m) */
    2 sdw_am_ptrs (4*16) fixed bin (35),             /* segment desc. pointers (4 sets of 16 for dps8/70m) */
    2 processor_pattern bit (8) aligned,             /* 1 bit ON for this processor */
    2 processor_tag fixed bin (3),                   /* CPU tag from maintenance panel */
    2 last_timer_setting bit (27) aligned,           /* last timer value loaded for this CPU */
    2 depth fixed bin,                               /* depth in eligible queue for running process */
    2 mode_reg bit (36) aligned,                     /* mode register for this processor */
    2 cache_luf_reg bit (36) aligned,                /* cache mode register for this CPU */
    2 fault_reg bit (72) aligned,                    /* place to store the fault register */
    2 apt_ptr ptr,                                   /* -> apte running on this cpu                    */
    2 idle_temp fixed bin (71),                      /* used by idle process                           */

/*  The following contains code used for handling connect faults for this processor               */

    2 fast_connect_code (FAST_CONNECT_CODE_WORDS) bit (36) aligned,
    2 fast_connect_code_end bit (36) aligned,        /* marker for fast_connect_init                   */
    2 mode_reg_enabled bit (36) aligned,             /* used to set mode register                      */
    2 pad_mod_16 (3) fixed bin,
    2 processor_stack aligned like stack_frame;      /* first stack frame location */
```

```
        call get_temp_segment_ ("prds", prdsp, code);

        unspec (prds) = ""b;


/* Now make some checks on alignment of certain variables */

        call check (addr (prds.idle_ptr), "idle_ptr", 2);
        call check (addr (prds.processor_stack), "processor_stack", 16);
        call check (addr (prds.ptw_am_regs), "ptw_am_regs", 16);
        call check (addr (prds.sdw_am_regs), "sdw_am_regs", 32);
        call check (addr (prds.fast_connect_code), "fast_connect_code", 2);


/* Now set up call to create data base */

        cdsa.sections (1).p = addr (prds);
        cdsa.sections (1).len = size (prds);
        cdsa.sections (1).struct_name = "prds";

        cdsa.seg_name = "prds";
        cdsa.num_exclude_names = 1;
        cdsa.exclude_array_ptr = addr (exclude_pad);

        string (cdsa.switches) = "0"b;
        cdsa.switches.have_text = "1"b;

        call create_data_segment_ (addr (cdsa), code);

        call release_temp_segment_ ("prds", prdsp, code);
```

```
check:    proc (where, message, modulo);

dcl  where ptr;
dcl  message char (*);
dcl  modulo fixed bin;

          if mod (bin (rel (where), 18), modulo) ^= 0
          then call com_err_ (0, prdsname, "The variable ^a is not aligned on a ^d-word boundary.", message, modulo);

     end check;
```

```
% include cds_args;
```

```
% include stack_header;
```

```
% include stack_frame;

% include mc;

     end prds;
```

```
/* BEGIN INCLUDE FILE ... ptw.168.incl.pl1 ... 02/26/81, for ADP conversion */
/* Note: This include file has an ALM counterpart made with cif. Keep it up to date */

dcl 1 168_core_ptw aligned based (ptp),                  /* In-core page descriptor */
      2 frame fixed bin (14) unsigned unaligned,          /* Core frame number */
      2 pad1 bit (4) unaligned,
      2 flags unaligned like 168_ptw_flags;

dcl 1 168_ptw aligned based (ptp),                       /* General declaration for out-of-core PTW */
      2 add bit (18) unaligned,
      2 flags like 168_ptw_flags unaligned;

dcl 1 168_special_ptw aligned based (ptp) like 168_ptw;    /* Page is somewhere peculiar -- add_type = "01"b */
dcl 1 168_real_disk_ptw aligned based (ptp) like 168_ptw;  /* PTW for page actually on disk -- add_type = "10"b */
dcl 1 168_null_disk_ptw aligned based (ptp) like 168_ptw;  /* PTW for page not yet on disk -- add_type = "11"b */

dcl 1 168_ptw_flags unaligned based,                     /* Various software/hardware flags */
     (2 add_type bit (4),                                /* 0000=null, 1000=core, 0100=disk, 0010=pd, 0001=swap */
      2 first bit (1),                                   /* the page has not yet been written out */
      2 er bit (1),                                      /* error on last page I/O (also used by post-purge as temp) */

      2 pad1 bit (1),
      2 unusable1 bit (1),                               /* can't be used because hardware resets this bit */
      2 phu bit (1),                                     /* page has been used bit */

      2 phm1 bit (1),                                    /* Cumulative OR of hardware phm's */
      2 nypd bit (1),                                    /* must be moved to paging device */
      2 phm bit (1),                                     /* page has been modified bit */

      2 phu1 bit (1),                                    /* page has been used in the quantum */
      2 wired bit (1),                                   /* page is to remain in core */
      2 os bit (1),                                      /* page is out-of-service (I/O in progress) */
      2 valid bit (1),                                   /* directed fault if this is 0 (page not in core) */
      2 df_no bit (2)) unaligned;                        /* directed fault number for page faults */

/* END INCLUDE FILE ... ptw.168.incl.pl1 */
```

```
/*        BEGIN INCLUDE FILE ... pv_holdt.incl.pl1 ... */

dcl pv_holdtp ptr;

dcl 1 pv_holdt (1 : 64) based (pv_holdtp) aligned,

    2 pvtx   fixed bin(17) unaligned,
    2 apterp bit(18) unaligned;

/*        END INCLUDE FILE ... pv_holdt.incl.pl1 ...*/
```

```
/*        BEGIN INCLUDE FILE ... pvt.incl.pl1 ... last modified January 1982 */


/* The physical volume table (PVT) is a wired-down table.
   It has one entry for each spindle present, be it for
   Storage System or "I/O" use.
*/

     dcl     pvt$                     ext,
             pvtp                     ptr;


     dcl     1 pvt                    based (pvtp) aligned,

               2 n_entries            fixed bin (17),          /* number of PVT entries */
               2 max_n_entries        fixed bin (17),          /* max number of PVT entries */
               2 n_in_use             fixed bin (17),          /* number of PVT entries in use */
               2 rwun_pvtx            fixed bin,               /* rewind_unloading pvtx */
               2 shutdown_state       fixed bin,               /* state of previous shutdown */
               2 esd_state            fixed bin,               /* state of ESD, >0 iff in ESD */
               2 prev_shutdown_state fixed bin,                /* shutdown state of previous bootload */
               2 prev_esd_state       fixed bin,               /* ESD state of previous bootload */

               2 time_of_bootload fixed bin (71),              /* Time of bootload */
               2 root_lvid            bit (36) aligned,        /* Logical volume ID of Root Logical Volume (RLV) */
               2 root_pvid            bit (36) aligned,        /* Physical volume ID of Root Physical Volume (RPV) */
               2 root_pvtx            fixed bin,               /* Index to PVTE for Root Physical Volume (RPV) */
               2 root_vtocx           fixed bin,               /* VTOCE index for root (>) */
               2 disk_table_vtocx     fixed bin,               /* VTOCE index for disk table on RPV */
               2 disk_table_uid       bit (36) aligned,        /* File System UID for disk_table */

               2 rpvs_requested       bit (1) aligned,         /* RPVS keyword given on BOOT */
               2 rpv_needs_salv       bit (1) aligned,         /* RPV required (not requested) salvage */
               2 rlv_needs_salv       bit (1) aligned,         /* RLV required (not requested) salvage */
               2 volmap_lock_wait_constant bit (36) aligned,/* For constructing wait event: OR pvte_rel into lower */
               2 volmap_idle_wait_constant bit (36) aligned,/* For constructing wait event: OR pvte_rel into lower */
               2 vtoc_map_lock_wait_constant bit (36) aligned, /* For constructing wait event: OR pvte_rel into lower */
               2 n_volmap_locks_held fixed bin (17),           /* Current number of volmap locks held */
               2 n_vtoc_map_locks_held fixed bin (17),         /* Current number of VTOC Map locks held */

               2 last_volmap_time     fixed bin (71),          /* Time a volmap was last locked/unlocked */
               2 last_vtoc_map_time fixed bin (71),            /* Time a VTOC Map was last locked/unlocked */
               2 total_volmap_lock_time fixed bin (71),        /* Total time volmap's were locked (integral) */
               2 total_vtoc_map_lock_time fixed bin (71),      /* Total time VTOC Maps were locked (integral) */

               2 n_volmap_locks       fixed bin (35),          /* Number times a volmap was locked */
               2 n_vtoc_map_locks     fixed bin (35),          /* Number times a vtoc_map was locked */
               2 volmap_lock_nowait_calls fixed bin (35),      /* Number calls to lock volmap, no wait */
               2 volmap_lock_nowait_fails fixed bin (35),      /* Number times lock failed */
```

```
        2 volmap_lock_wait_calls fixed bin (35),      /* Number calls to lock volmap, wait */
        2 volmap_lock_wait_fails fixed bin (35),      /* Number times lock failed */
        2 pad (2) bit (36) aligned,

        2 array               fixed bin (71);         /* Array of PVTE's -- must be double-word aligned */


/*        END INCLUDE FILE ...pvt.incl.pl1 */
```

```
/*  START OF:       pvte.incl.pl1        July 1982  * * * * * * * * * * * * * * * * * */

     dcl     pvt$array               aligned external;
     dcl     pvt$max_n_entries       fixed bin external;

     dcl     pvt_arrayp              ptr;
     dcl     pvtep                   ptr;

     dcl     1 pvt_array             (pvt$max_n_entries) aligned like pvte based (pvt_arrayp);



     dcl     1 pvte                  based (pvtep) aligned,

               2 pvid                bit (36),                /* physical volume ID */

               2 lvid                bit (36),                /* logical volume ID */

               2 dmpr_in_use         (3) bit (1) unaligned,   /* physical volume dumper interlock */
               2 pad3                bit (6) unaligned,
               2 skip_queue_count    fixed bin (18) unsigned unaligned, /* number of times this pv skipped for per-proc allocation du
e to saturation */
               2 brother_pvtx        fixed bin (8) unaligned,/* next pvte in lv chain */


               2 devname             char (4),                /* device name */

               (2 device_type        fixed bin (8),           /* device type */
               2 logical_area_number fixed bin (8),           /* disk drive number */
               2 used                bit (1),                 /* TRUE if this entry is used */
               2 storage_system      bit (1),                 /* TRUE for storage system (vs io disk) */
               2 permanent           bit (1),                 /* TRUE if cannot be demounted */
               2 testing             bit (1),                 /* Protocol bit for read_disk$test */
               2 being_mounted       bit (1),                 /* TRUE if the physical volume is being mounted */
               2 being_demounted     bit (1),                 /* TRUE if the pysical volume is being demounted */
               2 check_read_incomplete bit (1),               /* page control should check read incomplete */
               2 device_inoperative  bit (1),                 /* TRUE if disk_control decides dev busted */
               2 rpv                 bit (1),                 /* TRUE if this is the root physical volume */
               2 scav_check_address
                                     bit (1),                 /* TRUE is page control should check deposits/withdrawals against scaven
ger table */
               2 deposit_to_volmap   bit (1),                 /* TRUE if deposits should got to volume map, not stock */
               2 being_demounted2    bit (1),                 /* No more vtoc I/O during demount */
               2 pad5                bit (1),
               2 vacating            bit (1),                 /* don't put new segs on this vol */
               2 hc_part_used        bit (1),                 /* HC part set up by init_pvt */
```

```
        2 volmap_lock_notify bit (1) unal,          /* TRUE if notify required when volmap lock is unlocked */
        2 volmap_idle_notify bit (1) unal,          /* TRUE if notify required when volmap state is idle */
        2 vtoc_map_lock_notify bit (1) unal,        /* TRUE if notify required when vtoc map lock is unlocked */


        2 n_free_vtoce       fixed bin (17),        /* number of free VTOC entries */
        2 vtoc_size          fixed bin (17),        /* size of the VTOC part of the disk - in records */

        2 dbmrp              (2) bit (18),          /* rel ptr to dumber bit maps for this volume */

        2 nleft              fixed bin (17),        /* number of records left */
        2 totrec             fixed bin (17)) unaligned, /* Total records in this map */

        2 dim_info           bit (36),              /* Information peculiar to DIM */

        2 curn_dmpr_vtocx    (3) fixed bin unaligned,/* current vtocx being dumped */
        2 n_vtoce            fixed bin unaligned,   /* number of vtoce on this volume */

        2 baseadd            fixed bin (18) uns unaligned, /* Base of paging region */
        2 pad2               bit (18) unaligned,

        2 volmap_seg_sdw     fixed bin (71),        /* SDW describing volmap_seg */


        2 volmap_astep       ptr unal,              /* Packed pointer to ASTE for volmap_seg */

        2 volmap_offset      bit (18) unal,         /* Offset in volmap_seg of volume map */
        2 vtoc_map_offset    bit (18) unal,         /* Offset in volmap_seg of VTOC map */


        2 volmap_lock        bit (36) aligned,      /* Lock on volume map operations */

        2 vtoc_map_lock      bit (36) aligned,      /* Lock on VTOC map operations */

        2 volmap_stock_ptr   ptr unal,              /* Packed pointer to record stock */

        2 vtoc_map_stock_ptr ptr unal,              /* Packed pointer to VTOCE stock */

        2 volmap_async_state fixed bin (17) unaligned, /* Asynchronous update state of Volume Map */
        2 volmap_async_page  fixed bin (17) unaligned, /* Page number for asynchronous update */

        2 vol_trouble_count fixed bin (17) unaligned,  /* Count of inconsistencies since last salvage */
        2 scavenger_block_rel bit (18) unaligned;   /* Offset to scavenger block,  ^0 => scavenging */


   dcl     (VOLMAP_ASYNC_IDLE    init (0),              /* for volmap_async_state */
           VOLMAP_ASYNC_READ     init (1),
           VOLMAP_ASYNC_WRITE    init (2)) fixed bin int static options (constant);

/*  END OF:        pvte.incl.pl1                         * * * * * * * * * * * * * * * * */
```

```
/* BEGIN INCLUDE FILE RNT.INCL.PL1 - WRITTEN SEPTEMBER 1974 BY R. BRATT */
/* modified July 1976 by R. Bratt; updated March 1977 by M. Weaver */
/* modified November 1977 by M. Weaver to use PL/I offsets instead of pointers */

dcl (rntp, rntep) ptr;
dcl lth fixed bin (17);
dcl based_rnt_area area based;

dcl 1 rnt aligned based (rntp),
    2 areap ptr,                                        /* pointer to area for rnte allocations */
    2 meters,
      3 insert,
        4 trys fixed bin (17) unaligned,
        4 wins fixed bin (17) unaligned,
      3 get_segno like insert,
      3 get_refnames like insert,
      3 delete_segno like insert,
      3 delete_name like insert,
    2 rnt_area_size fixed bin,
    2 srulep ptr,
    2 name_hash_table (0:127) offset (rnt.areap -> based_rnt_area),
    2 segno_hash_table (0:127) offset (rnt.areap -> based_rnt_area);

dcl 1 rnte aligned based (rntep),
    2 name_fp offset (rnt.areap -> based_rnt_area),
    2 segno_fp offset (rnt.areap -> based_rnt_area),
    (2 segno fixed bin (17),
    2 length fixed bin (17),
    2 name char (lth refer (rnte.length)))unaligned;

/* --------------------END RNT.INCL.PL1-------------------- */
```

```
/*  START OF:       scavenger_data.incl.pl1   November 1982       *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  */

/* format: style3 */
dcl       scavenger_data$     external;

dcl       scavenger_datap     ptr;
dcl       sc_metersp          ptr;
dcl       sc_process_tablep   ptr;
dcl       scavenger_blockp    ptr;
dcl       record_blockp       ptr;
dcl       scavenger_optionsp  ptr;

dcl       sc_n_processes      fixed bin;
dcl       scavenger_n_records fixed bin;
dcl       scavenger_n_ovfl    fixed bin;


dcl       1 scavenger_data    aligned based (scavenger_datap),
            2 lock            aligned,                      /* Lock on scavenger_data */
              3 lock_word     bit (36) aligned,
              3 wait_event    bit (36) aligned,
              3 notify_sw     bit (1) aligned,
            2 process_table_ptr
                              ptr,                          /* Pointer to scavenger process table */
            2 error_severity  fixed bin,                    /* Severity of unexpected error condition */
            2 meters          aligned like sc_meters,
            2 free            bit (1) aligned;              /* Available region */


dcl       1 sc_meters         aligned based (sc_metersp),   /* Meters */
            2 n_scavenge      fixed bin (35),               /* Number of volume scavenges */
            2 pf              fixed bin (35),               /* Total page faults */
            2 vcpu            fixed bin (71),               /* Total virtual CPU time */
            2 clock_time      fixed bin (71),               /* Total clock time */
            2 n_vtoces        fixed bin (35),               /* Number VTOCEs examined */
            2 n_vtoces_damaged
                              fixed bin (35),               /* Number VTOCEs damaged by scavenge */
            2 n_vtoces_per_proc
                              fixed bin (35),               /* Number per-process VTOCEs freed */
            2 n_vtoces_per_boot
                              fixed bin (35),               /* Number per-bootload VTOCEs freed */
            2 n_vtoces_freed  fixed bin (35),               /* Total number VTOCEs freed */
            2 n_vtoces_fmd    fixed bin (35),               /* Number VTOCEs with fm_damaged reset */
            2 n_records       fixed bin (35),               /* Number non-null filemap entries examined */
            2 n_conflicts     fixed bin (35),               /* Number potential conflicts detected */
            2 n_fmd_conflicts fixed bin (35),               /* Number potential conflicts due to fm_damaged */
            2 n_real_conflicts
                              fixed bin (35),               /* Number real conflicts */
            2 n_lost_records  fixed bin (35);               /* Number lost records freed */
```

```
dcl        1 sc_process_table  aligned based (sc_process_tablep),
             2 max_n_processes fixed bin,                     /* Number of table entries */
             2 n_processes     fixed bin,                     /* Number active entries */
             2 process         (sc_n_processes refer (sc_process_table.max_n_processes)) aligned,
               3 processid     bit (36) aligned,              /* Owner. 0=>empty */
               3 pvtep         ptr unal,                      /* PVTE of volume being scavenged */
               3 blockp        ptr unal,                      /* Block w/i scavenger_data */
               3 first_block_page
                               fixed bin,                     /* Index of first page of block */
               3 n_block_pages fixed bin;                     /* Number of pages in block */


dcl        1 scavenger_block   aligned based (scavenger_blockp),
             2 n_records       fixed bin,                     /* Number of record addresses */
             2 n_ovfl          fixed bin,                     /* Number of overflow blocks */
             2 ovfl_free_ix    fixed bin,                     /* Index of first free overflow block */
             2 records         (scavenger_n_records refer (scavenger_block.n_records)) aligned like record_block,
             2 overflow        (scavenger_n_ovfl refer (scavenger_block.n_ovfl)) aligned like record_block;


dcl        1 record_block      aligned based (record_blockp),/* One per record address */
             2 vtocx           fixed bin (15) uns unal,       /* Owning VTOCE index */
             2 pageno          fixed bin (8) uns unal,        /* Owning page number */
             2 state           fixed bin (2) uns unal,        /* State */
             2 lock            bit (1) unal,                  /* Lock bit on this block */
             2 ovflx           fixed bin (10) uns unal;       /* Index of first overflow block on chain */

dcl        1 scavenger_data_pages
                               aligned based (scavenger_datap),
             2 page            (0:255) aligned,
               3 word          (1024) bit (36) aligned;

dcl        1 scavenger_options aligned based (scavenger_optionsp),
             2 print_meters    bit (1) unaligned,             /* ON => meter each scavenge into the log */
             2 debug           bit (1) unaligned,             /* ON => do special debugging things */
             2 dump            bit (1) unaligned,             /* ON => dump bad VTOCEs into syserr log */
             2 trap            bit (1) unaligned,             /* ON => trap to BOS for debug */
             2 no_optimize     bit (1) unaligned;             /* ON => no VTOCE read-ahead */


dcl        (
           STATE_UNSEEN        init (0),
           STATE_FREE          init (1),
           STATE_IN_USE        init (2),
           STATE_CONFLICT      init (3)
           )                   fixed bin int static options (constant);


/*  END OF:        scavenger_data.incl.pl1            *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  */
```

```
/* BEGIN INCLUDE FILE scs.incl.pl1 ... April 1982 */

/* Information about system controllers */

dcl  1 scs$controller_data (0:7) aligned ext,             /* per-controller info */
       2 size fixed bin (17) unaligned,                   /* size (in 1024 word blocks) of this controller */
       2 base fixed bin (17) unaligned,                   /* abs address (0 mod 1024) for base of this controller */
       2 eima_data (4) unaligned,                         /* EIMA information for this controller */
         3 mask_available bit (1) unaligned,              /* ON if corresponding mask exists */
         3 mask_assigned bit (1) unaligned,               /* ON if mask assigned to a port */
         3 mbz bit (3) unaligned,
         3 mask_assignment fixed bin (3) unaligned,       /* port to which mask is assigned */
       2 info aligned,
         3 online bit (1) unaligned,                      /* ON if controller is online */
         3 offline bit (1) unaligned,                     /* ON if controller is offline but can be added */
         3 store_a_online bit (1) unaligned,              /* ON if store A is online */
         3 store_a1_online bit (1) unaligned,             /* ON if store A1 is online */
         3 store_b_online bit (1) unaligned,              /* ON if store B is online */
         3 store_b1_online bit (1) unaligned,             /* ON if store B1 is online */
         3 store_b_is_lower bit (1) unaligned,            /* ON if store B is lower */
         3 ext_interlaced bit (1) unaligned,              /* ON if this SCU is interlaced with other SCU */
         3 int_interlaced bit (1) unaligned,              /* ON if this SCU is internally interlaced */
         3 four_word bit (1) unaligned,                   /* ON if external interlace is 4-word */
         3 cyclic_priority (7) bit (1) unaligned,         /* Cyclic priority for adjacent ports */
         3 type bit (4) unaligned,                        /* Model number for this controller */
         3 abs_wired bit (1) unaligned,                   /* ON if controller can have abs_wired pages */
         3 program bit (1) unaligned,                     /* PROGRAM/MANUAL switch setting */
         3 mbz bit (13) unaligned,
       2 lower_store_size fixed bin (17) unaligned,       /* size (in 1024 word blocks) of lower store */
       2 upper_store_size fixed bin (17) unaligned;       /* size (in 1024 word blocks) of upper store */

/* Information about CPUs */

dcl  1 scs$processor_data (0:7) aligned ext,              /* information about CPUs in the system */
       (
       2 online bit (1),                                  /* "1"b if CPU is online */
       2 offline bit (1),                                 /* "1"b if CPU is offline but can be added */
       2 release_mask bit (1),                            /* "1"b is this CPU is to give up its mask */
       2 accept_mask bit (1),                             /* "1"b if this CPU is to grap mask in idle loop */
       2 delete_cpu bit (1),                              /* "1"b if this CPU is to delete itself */
       2 interrupt_cpu bit (1),                           /* "1"b if this CPU takes hardware interrupts */
       2 halted_cpu bit (1),                              /* "1"b if this CPU has stopped itself (going to BOS) */
       2 cpu_type fixed bin (2) unsigned,                 /*  0 => DPS or L68, 1 => DPS8 */
       2 mbz bit (21),
       2 expanded_port bit (1),                           /* "1"b = on expanded port */
       2 expander_port fixed bin (2) unsigned,            /* The actual expander port */
       2 controller_port fixed bin (3) unsigned
       ) unaligned;                                       /* Port on controller */
```

```
dcl  1 scs$port_data (0:7) aligned external static,          /* Info about what is connected to each SCU port */
       2 assigned fixed bin (4) unsigned unaligned,          /* Type of device on this port                   */
       2 expander_port bit (1) unaligned,                    /* "1"b => this port has a port expander */
       2 expanded_cpu (0:3) bit (1) unaligned,               /* "1"b => this expander port has a CPU attached */
       2 iom_number fixed bin (3) unsigned unaligned,        /* IOM number of IOM attached to this port */
       2 cpu_number (0:3) fixed bin (3) unsigned unaligned,  /* CPU number of CPU(s) attached to this port */
                                                             /* cpu_number (0) is only one if expander_port is "0"b */
       2 pad bit (12) unaligned;

dcl  1 scs$cow (0:7) aligned external,                       /* Actual connect words */
       2 pad bit (36) aligned,                               /* Expander COW's must be odd-word */
       2 cow,
         3 sub_mask bit (8) unaligned,                       /* Expander sub-port mask */
         3 mbz1 bit (13) unaligned,
         3 expander_command bit (3) unaligned,               /* Expander command. */
         3 mbz2 bit (2) unaligned,
         3 expanded_port bit (1) unaligned,                  /* "1"b = on expanded port */
         3 expander_port fixed bin (3) unsigned unaligned,   /* Port on expander for cioc */
         3 mbz3 bit (3) unaligned,
         3 controller_port fixed bin (3) unaligned unsigned;/* controller port for this CPU */

dcl  1 scs$cow_ptrs (0:7) external aligned,                  /* Pointers to COW's */
       2 rel_cow_ptr bit (18) unal,                          /* Relative pointer to COW */
       2 pad bit (12) unal,
       2 tag bit (6) unal;                                   /* Better be zero. */

dcl  1 scs$reconfig_general_cow aligned external,            /* Used during reconfig ops. */
       2 pad bit (36) aligned,
       2 cow,                                                /* Connect operand word, in odd location. */
         3 sub_mask bit (8) unaligned,                       /* Expander sub-port mask */
         3 mbz1 bit (13) unaligned,
         3 expander_command bit (3) unaligned,               /* Expander command. */
         3 mbz2 bit (9) unaligned,
         3 controller_port fixed bin (3) unaligned unsigned;/* controller port for this CPU */

/* MASKS and PATTERNS */

dcl  scs$sys_level bit (72) aligned ext;                     /* mask used while handling I/O interrupts */
dcl  scs$open_level bit (72) aligned ext;                    /* mask used during normal operation */
dcl  scs$processor_start_mask bit (72) aligned ext;          /* mask used when starting up a CPU */
dcl  scs$cpu_test_mask bit (72) aligned ext;                 /* mask used for ISOLTS CPU testing */
dcl  scs$number_of_masks fixed bin ext;                      /* number of masks (starting at sys_level) */
dcl  scs$processor_start_pattern bit (36) aligned ext;       /* SMIC pattern used to send processor start interrupt */
dcl  scs$cpu_test_pattern bit (36) aligned ext;              /* SMIC pattern used for ISOLTS processor testing */

/* CAM and CACHE clear info */

dcl  scs$cam_pair fixed bin (71) ext;                        /* instructions XEDd when CAMing and clearing CACHE */
dcl  scs$cam_wait bit (8) aligned ext;                       /* Used when evicting pages from main memory */

/* MASKING INSTRUCTIONS & POINTERS */

dcl  scs$set_mask (0:7) bit (36) aligned ext;                /* instructions to set mask (STAQ or SMCM) */
dcl  scs$read_mask (0:7) bit (36) aligned ext;               /* instructions to read mask (LDAQ or RMCM) */
```

```
dcl  scs$mask_ptr (0:7) ptr unaligned ext;                  /* pointers for real or simulated masks */

/* MISCELLANEOUS */

dcl  1 scs$processor_test_data aligned ext,                 /* info used for cpu testing */
       (
       2 active bit (1),                                    /* = "1"b if cpu currently under test */
       2 scu_state bit (2),                                 /* state of scu being used for testing (see definition below) */
            2 pad1 bit (15),
       2 cpu_tag fixed bin (5),                             /* tag of cpu under test */
       2 scu_tag fixed bin (5),                             /* tag of scu being used for cpu testing */
       2 mask_cpu fixed bin (5)
       ) unaligned;                                         /* tag of active cpu that has mask asigned to above scu */

/* scu_state = "00"b => SCU defined by scs$processor_test_data.scu_tag not yet effected */
/* scu_state = "01"b => all core removed from SCU, port mask not yet changed */
/* scu_state = "10"b => all core removed from SCU, port mask changed */
/* scu_state = "11"b => only 64k at base of SCU being used for testing, original port mask restored */

dcl  scs$idle_aptep (0:7) ptr unaligned ext;                /* pointer to idle process APTE for each processor */

dcl  scs$connect_lock bit (36) aligned ext;                 /* lock for sending connects */
dcl  scs$reconfig_lock bit (36) aligned ext;                /* Lock used during reconfiguration */
dcl  scs$trouble_flags bit (8) aligned ext;                 /* checkoff flags for sys_trouble stopping */
dcl  scs$bos_restart_flags bit (8) aligned ext;             /* checkoff flags for restarting after sys_trouble */
dcl  scs$nprocessors fixed bin ext;                         /* number of runnung processors */
dcl  scs$bos_processor_tag fixed bin (3) ext;               /* CPU tag of processor running BOS */
dcl  scs$faults_initialized bit (1) aligned ext;            /* ON after faults have been enabled */
dcl  scs$sys_trouble_pending bit (1) aligned ext;           /* sys_trouble event is pending in the system */
dcl  scs$fast_cam_pending (0:7) bit (36) aligned ext;       /* checkoff cells for cam connect          */
dcl  scs$interrupt_controller fixed bin (3) ext;            /* port number of low order controller */
dcl  scs$processor_start_int_no fixed bin (5) ext;          /* interrupt cell for starting a processor */
dcl  scs$processor bit (8) aligned ext;                     /* bits ON for online CPUs */
dcl  scs$processor_start_wait bit (8) aligned ext;          /* checkoff flags for waiting for new processor */

dcl  scs$trouble_dbrs (0:7) fixed bin (71);                 /* DBR values at system crash time */

dcl  scs$port_addressing_word (0:7) bit (3) aligned ext;    /* active module port number for each controller */

dcl  scs$cfg_data (0:7) fixed bin (71) aligned ext;         /* RSCR-CFG data from each controller */

dcl  scs$cfg_data_save fixed bin (71) aligned ext;          /* RSCR-CFG save area for ISOLTS CPU testing */

dcl  scs$expanded_ports bit (1) unaligned dim (0:7) external;
                                                            /* Which ports have expanders */

dcl  scs$processor_switch_data (0:4) bit (36) aligned ext;  /* raw data from RSW 0 thru 4 */
dcl  scs$processor_switch_template (0:4) bit (36) aligned ext;/* expected data from RSW 0 thru 4 */
dcl  scs$processor_switch_compare (0:4) bit (36) aligned ext; /* discrepancies from expected data */
dcl  scs$processor_switch_mask (0:4) bit (36) aligned ext;  /* masks for comparing switch data */

dcl  scs$processor_data_switch_value bit (36) aligned ext;  /* Correct value for CPU data switches */

dcl  scs$controller_config_size (0:7) fixed bin (14) aligned ext;
                                                            /* Controller size on config card */
```

```
dcl  scs$reconfig_locker_id char (32) aligned ext;          /* process group ID of process doing reconfiguration */

dcl  scs$scas_page_table (0:31) bit (36) aligned external static;
                                                            /* PTWs for SCAS pages */

dcl  scs$cycle_priority_template bit (7) aligned ext;       /* template for setting anti-hog switches        */
dcl  scs$set_cycle_switches bit (1) aligned ext;            /* flag to set ant-hog switches                  */


dcl  (
     IOM_PORT init (1),
     CPU_PORT init (2),
     BULK_PORT init (3)
     ) fixed bin int static options (constant);             /* values for scs$port_data.assigned             */

/* END INCLUDE FILE scs.incl.pl1 */
```

```
/* BEGIN INCLUDE FILE ... sdw.l68.incl.pl1 ... Updated for ADP conversion 03/01/81 */
/* Note: This include file has an ALM counterpart made with cif. Keep it up to date */

dcl 1 l68_sdw based (sdwp) aligned,                      /* Level 68 Segment Descriptor Word */

    (2 add bit (24),                                      /* main memory address of page table */
     2 rings,                                             /* ring brackets for the segment */
       3 r1 bit (3),
       3 r2 bit (3),
       3 r3 bit (3),
     2 valid bit (1),                                     /* directed fault bit (0 => fault) */
     2 df_no bit (2),                                     /* directed fault number */

     2 pad1 bit (1),
     2 bound bit (14),                                    /* boundary field (in 16 word blocks) */
     2 access,                                            /* access bits */
       3 read bit (1),                                    /* read permission bit */
       3 execute bit (1),                                 /* execute permission bit */
       3 write bit (1),                                   /* write permission bit */
       3 privileged bit (1),                              /* privileged bit */
     2 unpaged bit (1),                                   /* segment is unpaged if this is 1 */
     2 not_a_gate bit (1),                                /* if this is 0 the entry bound is checked by hardware */
     2 cache bit (1),                                     /* cache enable bit */
     2 entry_bound bit (14)) unaligned;                   /* entry bound */

/* END INCLUDE FILE ... sdw.l68.incl.pl1 */
```

```
/* BEGIN INCLUDE FILE ... sdw_info.incl.pl1 ... 12/16/80, for ADP conversion */
/* Note: This include file has an ALM counterpart made with cif. Keep it up to date */

dcl  sdw_info_ptr pointer;

dcl 1 sdw_info aligned based (sdw_info_ptr),          /* Structure describing SDW contents */
      2 address fixed bin (26),                        /* Address of seg base or of page table */
      2 size fixed bin (19),                           /* Max length of segment (NOT offset of last word) */

      2 access unaligned,                              /* REWP */
        3 read bit (1) unaligned,
        3 execute bit (1) unaligned,
        3 write bit (1) unaligned,
        3 privileged bit (1) unaligned,

      2 pad1 bit (32) unaligned,

      2 rings unaligned,                               /* Ring brackets */
        3 r1 bit (3) unaligned,
        3 r2 bit (3) unaligned,
        3 r3 bit (3) unaligned,

      2 pad2 bit (27) unaligned,

      2 flags aligned,
        3 paged bit (1) unaligned,                     /* "1"b => Segment is paged */
        3 faulted bit (1) unaligned,                   /* "1"b => SDW has fault set */
        3 cache bit (1) unaligned,                     /* "1"b => Segment is encacheable */
        3 pad3 bit (33) unaligned,

      2 gate_entry_bound fixed bin (14);               /* Number of entrypoints in gate, or zero */

/* END INCLUDE FILE ... sdw_info.incl.pl1 */
```

```
/*        BEGIN INCLUDE FILE ... signaller_stack.incl.pl1 ... Created Feb 79 by D.Spector */

/*        This file matches signaller_stack.incl.alm and is currently used only by verify_lock */

declare   1 signaller_stack based unaligned,
            2 pad (8) bit (36),                   /* Make machine conditions 0 mod 16 */
            2 mach_cond (48) bit (36),            /* Machine conditions */
            2 mc_ptr ptr aligned,                 /* Pointer to machine conditions */
            2 null_ptr ptr aligned,               /* Null pointer */
            2 string_descriptor bit (36),         /* Condition name descriptor */
            2 ptr_descriptor bit (36),            /* M.C. ptr descriptor */
            2 arglist (18) bit (36),              /* Arg list for call to signal */
            2 signal_string char (32),            /* Condition name */
            2 on_unit (16) bit (36),              /* Must be at 128 in stack frame */
            2 history_registers (128) bit (36);

/* on_unit must start at 128 because trap_caller_caller_ sets up a stack frame
   assuming this to be so.  Similarly mach_cond must start at 48. */

/*        END INCLUDE FILE ... signaller_stack.incl.pl1 ... */
```

```
/* BEGIN INCLUDE FILE slt.incl.pl1 --- Last modified 2/76 SHW */

/* Declarations for Segment Loading Table header and array.

   Used by Initialization and MST Checker subroutines */

dcl sltp ptr,                                           /* pointer to base of SLT segment */
    names_ptr ptr,                                      /* pointer to base of SLT names segment */
    namep ptr,                                          /* pointer to segment name list block */
    pathp ptr,                                          /* pointer to segment's directory path name */
    aclp ptr;                                           /* pointer to acl structure */

declare 1 slt based (sltp) aligned,                     /* declaration of Segment Loading Table (SLT) */
          2 name_seg_ptr ptr,                           /* words 0-1, pointer (ITS pair) to name segment */
          2 free_core_start fixed bin (24),             /* word 2, start of free core after perm-wired */
          2 first_sup_seg fixed bin (18),               /* word 3, first supervisor segment number */
          2 last_sup_seg fixed bin (18),                /* word 4, last supervisor segment number */
          2 first_init_seg fixed bin (18),              /* word 5, first initializer segment number */
          2 last_init_seg fixed bin (18),               /* word 6, last initializer segment number */
          2 free_core_size fixed bin (24),              /* size (in words) of free core after perm-wired */
          2 seg (0:8191) aligned,                       /* segment entries (4 words each) */
            3 slte (4) fixed bin (35);                  /* Space for SLT entries */

/* auxiliary segment of SLT for storing of segment names and directory path names */

declare 1 name_seg based (names_ptr) aligned,           /* name segment header */
          2 pad bit (18) unal,
          2 next_loc bit (18) unal,                     /* Next available free location in name seg */
          2 ht (0:127) bit (18) aligned;                /* Names hash table */

declare 1 segnam based (namep) aligned,                 /* declaration for segment name block */
          2 count fixed bin (17),                       /* number of segment names in this block */
          2 names (50 refer (segnam.count)),            /* segment name array */
            3 hp bit (18) unal,                         /* hash thread pointer */
            3 ref bit (1) unal,                         /* "1"b if name referenced */
            3 pad bit (5) unal,
            3 segno bit (12) unal,                      /* segment number associated with this name */
            3 name char (32) unal;                      /* space for name (max 32 characters) */

declare 1 path based (pathp) aligned,                   /* declaration for directory path name */
          2 size fixed bin (17),                        /* length of pathname */
          2 name char (168 refer (path.size)) unal,     /* directory path name */
          2 acls fixed bin;                             /* ACL list starts here */

declare 1 acls based (aclp) aligned,                    /* declaration for acl list */
          2 count fixed bin,                            /* number of entries in acl list */
          2 acl (50 refer (acls.count)),                /* array of acl entries */
            3 userid char (32),                         /* user specification */
            3 mode bit (36) aligned,                    /* mode for the specified user */
```

```
		3 pad bit (36) aligned,
		3 code fixed bin;

/* END INCLUDE FILE slt.incl.pl1 */
```

```
/* BEGIN INCLUDE FILE slte.incl.pl1 */
/* Declaration for Segment Loading Table Entry structure.
   Used by Initialization, MST Generation, and MST Checker subroutines */
/* last modified 5/4/76 by Noel I. Morris         */
/* format: style3 */

dcl       sltep              ptr;

dcl       1 slte_uns         based (sltep) aligned,
          ( 2 names_ptr      bit (18),                  /* rel pointer to thread of names */
            2 path_ptr       bit (18),                  /* rel pointer to pathname (if present) */
/**** End of word 1 */
            2 access         bit (4),                   /* SDW access bit (REWP) */
            2 cache          bit (1),                   /* Segment to be allowed in cache */
            2 abs_seg        bit (1),                   /* segment is an abs seg if ON */
            2 firmware_seg   bit (1),                   /* load in low 256 */
            2 layout_seg     bit (1),                   /* mailbox & such */
            2 pad1           bit (4),                   /* unused */
            2 wired          bit (1),                   /* segment is wired if ON */
            2 paged          bit (1),                   /* segment is paged if ON */
            2 per_process    bit (1),                   /* segment is per-process if ON */
            2 pad3           bit (2),
            2 acl_provided   bit (1),                   /* ON if acl structure follows path_name on MST */
/**** End of 1st half of word 2 */
            2 pad4           bit (3),
            2 branch_required bit (1),                  /* path name supplied if ON */
            2 init_seg       bit (1),                   /* segment is init_seg if ON */
            2 temp_seg       bit (1),                   /* segment is temp_seg if ON */
            2 link_provided  bit (1),                   /* linkage segment provided if ON */
            2 link_sect      bit (1),                   /* segment is linkage segment if ON */
            2 link_sect_wired bit (1),                  /* linkage segment is wired if ON */
            2 combine_link   bit (1),                   /* linkage is combined if ON */
            2 pre_linked     bit (1),                   /* lot entry has been made if ON */
            2 defs           bit (1),                   /* segment is definitions segment if ON */
/***** End of word 2 */
            2 pad5           bit (6),
            2 cur_length     fixed bin (9) uns,         /* current length of segment (in 1024 word blocks) */
            2 ringbrack      (3) fixed bin (3) uns,     /* ringbrackets */
            2 segno          fixed bin (18) uns,        /* text/link segment number */
/***** End of word 3 */
            2 pad7           bit (3),
            2 max_length     fixed bin (9) uns,         /* maximum length for segment */
            2 bit_count      fixed bin (24) uns
            )                unaligned;                 /* bitcount of segment */

dcl       1 slte             based (sltep) aligned,
          ( 2 names_ptr      bit (18),                  /* rel pointer to thread of names */
            2 path_ptr       bit (18),                  /* rel pointer to pathname (if present) */
            2 access         bit (4),                   /* SDW access bit (REWP) */
```

```
          2 cache          bit (1),                    /* Segment to be allowed in cache */
          2 abs_seg        bit (1),                    /* segment is an abs seg if ON */
          2 firmware_seg   bit (1),
          2 layout_seg     bit (1),
          2 pad2           bit (4),
          2 wired          bit (1),                    /* segment is wired if ON */
          2 paged          bit (1),                    /* segment is paged if ON */
          2 per_process    bit (1),                    /* segment is per-process if ON */
          2 pad3           bit (2),
          2 acl_provided   bit (1),                    /* ON if acl structure follows path_name on MST */
          2 pad4           bit (3),
          2 branch_required bit (1),                   /* path name supplied if ON */
          2 init_seg       bit (1),                    /* segment is init_seg if ON */
          2 temp_seg       bit (1),                    /* segment is temp_seg if ON */
          2 link_provided  bit (1),                    /* linkage segment provided if ON */
          2 link_sect      bit (1),                    /* segment is linkage segment if ON */
          2 link_sect_wired bit (1),                   /* linkage segment is wired if ON */
          2 combine_link   bit (1),                    /* linkage is combined if ON */
          2 pre_linked     bit (1),                    /* lot entry has been made if ON */
          2 defs           bit (1),                    /* segment is definitions segment if ON */
          2 pad5           bit (6),
          2 cur_length     bit (9),                    /* current length of segment (in 1024 word blocks) */
          2 ringbrack      (3) bit (3),                /* ringbrackets */
          2 segno          bit (18),                   /* text/link segment number */
          2 pad6           bit (3),
          2 max_length     bit (9),                    /* maximum length for segment */
          2 bit_count      bit (24)
          )                unaligned;                  /* bitcount of segment */

/* END INCLUDE FILE slte.incl.pl1 */
```

```
/*   BEGIN INCLUDE FILE ... sst.incl.pl1 ... January 1971 */
/* Note: This include file has an ALM counterpart made with cif. Keep it up to date */


dcl  sst_seg$ external;
dcl  sstp ptr;

dcl 1 sst based (sstp) aligned,
    2 space (8) fixed bin,                              /* empty space to watch for bugs */

/* SST HEADER */

    2 pre_page_time fixed bin (71),                     /* total time spent pre-paging */
    2 post_purge_time fixed bin (71),                   /* total time spent post-purging */
    2 post_in_core fixed bin,                           /* total pages in core (and in list) at purge time */
    2 thrashing fixed bin,                              /* meter of thrashing being done on system */
    2 npfs_misses fixed bin,                            /* meter of times npfs was on when pre-paging */
    2 salv fixed bin,                                   /* flag which is ^=0 if and only if salvaging */

    2 ptl bit (36),                                     /* global page table loop lock */
    2 astl bit (36),                                    /* global ast allocation block lock */
    2 astl_event bit (36),                              /* event used when waiting for AST lock */
    2 astl_notify_requested bit (1) aligned,            /* flag to notify AST lock */
    2 nused fixed bin,                                  /* number of pages on used list */
    2 ptwbase fixed bin (24),                           /* absolute address of page table array */
    2 tfreep ptr,                                       /* pointer to first trailer on free list */

    2 astap ptr,                                        /* aste array pointer */
    2 bulk_pvtx fixed bin (8) aligned,                  /* pvtx of bulk store, zero if none */
    2 ptl_wait_ct fixed bin,                            /* pxss: number is >= # of processes waiting to ptl */
    2 astsize fixed bin,                                /* size of an AST entry */
    2 cmesize fixed bin,                                /* size of a CME entry */
    2 root_astep ptr,                                   /* pointer to the root AST entry */

    2 pts (0: 3) fixed bin,                             /* array of page table sizes */
    2 level (0:3),                                      /* per-list information about ASTE's */
      3 (ausedp, no_aste) bit (18) unaligned,           /* used list and count of number of entries */

    2 (atempp, atempp1) bit (18) unal,                  /* temp seg list pointer */
    2 dm_enabled bit (1) aligned,                       /* ON => journal seg exists */
    2 (ainitp, ainitp1) bit (18) unal,                  /* init seg list pointer */
    2 strsize fixed bin,                                /* Trailer size in words. */

/* CORE MAP HEADER */

    2 cmp ptr,                                          /* pointer to start of core map */
    2 usedp bit (18),                                   /* pointer to first used core block */
    2 wtct fixed bin,                                   /* count of pages being written */
```

```
     2 startp bit (18),                                 /* pointer to solid page for lap counting (fsdct) */
     2 removep bit (18),                                /* pointer to list of pages being removed from use */
                                                        /* MISC */

     2 double_write fixed bin,                          /* trigger for store through scheme */
                                                        /* 0 = no double writes,
                                                           1 = all non-pd pages get written,
                                                           2 = all directories get written */
     2 temp_w_event bit (36) aligned,                   /* wait event for temp wiring lock */

     2 root_pvtx fixed bin,                             /* pvtx or rpv */
     2 ptw_first bit (1) aligned,                       /* flag controlling when pages go to pd */
     2 nolock bit (1),                                  /* if on, don't lock ptl on interrupts */
     2 x_fsdctp bit (18),                               /* removed by thvv */

     2 fc_skips_pinned fixed bin (35),                  /* number of skips over pinned page in find_core */
     2 cl_skips_pinned fixed bin (35),                  /* number of skips over pinned page in claim_mod_core */
     2 ast_ht_ptr ptr,                                  /* AST hast table pointer */
     2 ast_ht_n_buckets fixed bin,                      /* number of buckets in AST hash table */
     2 ast_ht_uid_mask bit (36) aligned,                /* mask to strip out low-order bits of uid */
     2 meter_ast_locking fixed bin,                     /* non-zero enables AST lock meters */
     2 checksum_filemap fixed bin,                      /* non-zero enables filemap checksumming */

/* 100 octal */

     2 page_read_errors fixed bin,                      /* read errors posted to page control */
     2 page_write_errors fixed bin,                     /* write errors posted to page control */
     2 rws_read_errors fixed bin,                       /* read-side rws errors */
     2 rws_write_errors fixed bin,                      /* write-side rws errors */

     2 cycle_pv_allocation fixed bin,                   /* flag to cycle VTOCE allocation among PVs */

     2 n_trailers fixed bin,                            /* Number of trailer entries in str_seg */
     2 synch_activations fixed bin (35),                /* Activation attempts for synchronized segs */
     2 synch_skips fixed bin (35),                      /* get_aste skips because not synchronized */

     2 lock_waits fixed bin,                            /* Number of times we had to wait for a lock */
     2 total_locks_set fixed bin,                       /* Total number of block locks set */
     2 pdir_page_faults fixed bin,                      /* total page faults off >pdd */
     2 level_1_page_faults fixed bin,                   /* total page faults in sys libes */
     2 dir_page_faults fixed bin,                       /* Total page faults on directories */
     2 ring_0_page_faults fixed bin,                    /* page faults in ring 0 */
     2 rqover fixed bin (35),                           /* errcode for record quota overflow */
     2 pc_io_waits fixed bin,                           /* Number of times pc had to wait on io */

/*  The following (until pdmap) used to be the 'cnt' in cnt.incl.pl1 */

     2 steps fixed bin,                                 /* number of steps taken around used list */
     2 needc fixed bin,                                 /* number of times core page needed */
     2 ceiling fixed bin,                               /* number of times ceiling hit */
     2 ctwait fixed bin,                                /* number of times write counter was full */
     2 wired fixed bin,                                 /* number of pages wired by pc */
     2 laps fixed bin,                                  /* number of times around used list */
     2 skipw fixed bin,                                 /* number of pages skiped because they were wired */
```

```
    2 skipu fixed bin,                                      /* because of being used */

    2 skipm fixed bin,                                      /* because of being modified */
    2 skipos fixed bin,                                     /* because out of service */
    2 skipspd fixed bin,                                    /* number of times a block of core was skipped for active rws */
    2 aused fixed bin,                                      /* number of AST entries on used list */
    2 damaged_ct fixed bin,                                 /* count of segments that system damaged */
    2 deact_count fixed bin,                                /* count of deactivations */
    2 demand_deact_attempts fixed bin,                      /* user requested deactivations */
    2 demand_deactivations fixed bin,                       /* user instigated deactivations */

    2 reads (8) fixed bin,                                  /* number of reads for each did */
    2 writes (8) fixed bin,                                 /* number of writes for each did */

    2 short_pf_count fixed bin,                             /* count of page faults on out of service pages */
    2 loop_locks fixed bin,                                 /* count of times locked PTL */
    2 loop_lock_time fixed bin (71),                        /* time spent looping on PTL */
    2 cpu_sf_time fixed bin (71),                           /* cpu time spent in seg_fault */
    2 total_sf_pf fixed bin,                                /* total page faults while in seg_fault */
    2 total_sf fixed bin,                                   /* total number of seg_faults */
    2 pre_page_size fixed bin,                              /* total pre-pagings expected */
    2 post_list_size fixed bin,
    2 post_purgings fixed bin,                              /* total number of post-purgings */
    2 post_purge_calls fixed bin,                           /* total number of calls to post-purge */
    2 pre_page_calls fixed bin,                             /* total number of calls tp pre-page */
    2 pre_page_list_size fixed bin,
    2 pre_page_misses fixed bin,                            /* total number of misses in pre-page list */
    2 pre_pagings fixed bin,                                /* total number of pre-pagings */

/* 200 octal */

/*  TEMPORARY WIRED PROCEDURE INFO  */

    2 wire_proc_data (8) fixed bin (71),                    /* data for wire_proc */

/* MAIN MEMORY USAGE INFORMATION */

    2 abs_wired_count fixed bin,                            /* count of abs-wired pages */
    2 system_type fixed bin,                                /* ADP_SYSTEM or L68_SYSTEM */
    2 wired_copies fixed bin,                               /* number of times a wired page was copied */
    2 recopies fixed bin,                                   /* number of times recopied because modified */
    2 first_core_block fixed bin,                           /* core map index for first block of core */
    2 last_core_block fixed bin,                            /* core map index for last block of core */
    2 fw_retries fixed bin (35),                            /* force_write retries due to ASTE move */
    2 pvhtp ptr unaligned,                                  /* ptr to PV hold table for debugging */

/*  AST METERS  */

    2 askipsize (0: 3) fixed bin,                           /* array of skips because wrong AST size */
    2 aneedsize (0: 3) fixed bin,                           /* array of times needed each size */

    2 stepsa fixed bin,                                     /* count of steps taken looking for an AST entry */
    2 askipsehs fixed bin,                                  /* count of skips because EHS was ON */
    2 asearches fixed bin,                                  /* count of full searches made */
    2 askipslevel fixed bin,                                /* count of skips because pages were in core */
```

```
    2 askipsinit fixed bin,                   /* count of times turned OFF init switch */
    2 acost fixed bin,                        /* cumulative cost of deactivations */
    2 askipslock fixed bin,                   /* count of skips because couldn't lock parent */
    2 askipdius fixed bin,                    /* count of skips because DIUS was on */

    2 alaps fixed bin,                        /* lap counter for AST list */
    2 updates fixed bin,                      /* calls to updateb */
    2 setfaults_all fixed bin,                /* setfaults done to the entire SDW */
    2 setfaults_acc fixed bin,                /* setfaults done to the access field */
    2 total_bf fixed bin,                     /* count of bound faults */
    2 total_bf_pf fixed bin,                  /* page faults during bound faults */
    2 cpu_bf_time fixed bin (71),             /* cpu time spent in bound fault */

    2 asteps (0: 3) fixed bin,                /* per-size AST step counters */

    2 ast_locked_at_time fixed bin (71),      /* clock reading when ast last locked */
    2 ast_locked_total_time fixed bin (71),   /* total real time the ast lock was locked */
    2 ast_lock_wait_time fixed bin (71),      /* total real time of all waiting on ast lock */
    2 ast_locking_count fixed bin (35),       /* number of times ast was locked */
    2 cleanup_count fixed bin,                /* calls to pc$cleanup */
    2 cleanups_with_any_rws fixed bin,        /* ditto, with >0 rws's */
    2 cleanup_rws_count fixed bin,            /* total rws's started by cleanup */
    2 cleanup_real_time fixed bin (71),       /* total real time in pc$cleanup */

/* PRE-PAGE METERS */

/* 300 octal */

    2 tree_count (0: 63) fixed bin,           /* counters for pre-page decisions */

/* 400 octal */

    2 pp_meters (0: 63) fixed bin,            /* counters for measuring pre-page success */

/* End of old cnt include file */

/* 500 octal */

    2 wusedp bit (18) aligned,                /* Relative cmep to next cme for writing */
    2 write_hunts fixed bin,                  /* Times claim_mod_core invoked */
    2 claim_skip_cme fixed bin,               /* Times unacceptable cme found by c_m_c */
    2 claim_skip_free fixed bin,              /* Times free cme passed by c_m_c */
    2 claim_notmod fixed bin,                 /* Times c_m_c passed pure page */
    2 claim_passed_used fixed bin,            /* Times used page seen */
    2 claim_skip_ptw fixed bin,               /* Times c_m_c saw unacceptable ptw */
    2 claim_writes fixed bin,                 /* Writes queued by c_m_c */
    2 claim_steps fixed bin,                  /* Steps passed in core claiming */
    2 rws_reads_os fixed bin,                 /* RWS reads outstanding, in SST for debugging */
    2 pd_updates fixed bin,                   /* done_ time pd writes */
    2 pre_seeks_failed fixed bin,             /* counter of times quick find_core_ failed */
    2 pd_desperation_steps fixed bin,         /* steps of allocate_pd finding pdme */
    2 pd_desperations fixed bin,              /* times allocate_pd needed to force one free */
    2 skips_nypd fixed bin,                   /* find_core skips for nypd pages */
    2 pd_writeahead bit (1) aligned,          /* "1"b => allocate_pd at disk done time */
    2 pd_desperations_not_mod fixed bin,      /* desperations on pure pages */
```

```
     2 resurrections fixed bin,                          /* nulled addresses reinstated */
     2 volmap_seg_page_faults fixed bin (35),            /* Pseudo-page faults on volmap_seg */
     2 oopv fixed bin,                                   /* out-of-physical-volume page faults */
     2 pdflush_replaces fixed bin,                       /* addresses "corrected" by post-crash pd flush */
     2 pcrsst_statptr ptr unal,                          /* ptr to damage table of pc_recover_sst */
     2 pd_resurrections fixed bin,                       /* addresses resurrected at RWS time */
     2 dblw_resurrections fixed bin,                     /* addresses resurrected by double-writing */
     2 sgm_time fixed bin (71),                          /* Time (VCPU) in seg mover */
     2 sgm_pf fixed bin,                                 /* Page faults in seg moving */
     2 bad_sgms fixed bin,                               /* Seg moves that failed */
     2 sgm_sgft fixed bin,                               /* Seg faults in seg moves */
     2 good_sgms fixed bin,                              /* Seg moves that completed */
     2 claim_runs fixed bin,                             /* Times claim_mod_core had to run */
     2 activations fixed bin,                            /* total count of activations */
     2 dir_activations fixed bin,                        /* count of directory activations */
     2 hedge_updatevs fixed bin,                         /* call-in updatevs */
     2 hedge_writes fixed bin,                           /* call in core flush writes */
     2 evict_recover_data,                               /* see evict_page.alm */
      3 evict_ptp bit (18) unal,                         /* ptp of page being moved */
      3 evict_phmbit bit (18) unal,                      /* N/Z if page was mod */

/* Data for metering force_write facility 08/19/78 */

     2 force_swrites fixed bin,                          /* Calls on segments to force write */
     2 force_pwrites fixed bin,                          /* Mod pages so written */
     2 fw_none fixed bin,                                /* Force write wrote none */
     2 force_updatevs fixed bin,                         /* Updatev's so forced */

     2 pf_pd_loop_time fixed bin (71),                   /* Time looping on pd on page faults */
     2 pf_unlock_ptl_time fixed bin (71),                /* Time unlocking ptln page faults */
     2 pf_pd_loop_meterings fixed bin,
     2 pf_unlock_ptl_meterings fixed bin,

     2 makeknown_activations fixed bin (35),             /* activations at makeknown time */
     2 backup_activations fixed bin (35),                /* activations for backup */
     2 metering_flags aligned,                           /* small chunks of misc. information */
       3 activate_activated bit (1) unal,                /* ON => last call to activate entry actually activated something */
       3 pad bit (35) unal,
     2 seg_fault_calls fixed bin (35),                   /* number calls to seg_fault for explicit activation */

/* METERS FOR STACK TRUNCATION */

     2 (stk_truncate_should_didnt,                       /* counts */
        stk_truncate_should_did,
        stk_truncate_shouldnt_didnt,
        stk_truncate_shouldnt_did) fixed bin (35),
     2 stk_pages_truncated fixed bin (35),
     2 stk_pages_truncated_in_core fixed bin (35),

     2 padder (8) fixed bin,

/* the following data is used by page multilevel */

/* 600 octal */
```

```
     2 pdmap ptr,                                  /* pointer to the pd map */
     2 pdhtp ptr,                                  /* pointer to the pd hash table */
     2 pd_id fixed bin (8) aligned,                /* pvt index of paging device, 0 if none */
     2 pdsize fixed bin,                           /* the number of words in a paging device map entry */
     2 pdme_no fixed bin,                          /* the number of entries in the paging device map */
     2 pdusedp bit (18) unaligned,                 /* pointer to head of paging device used list */
     2 pd_first fixed bin,                         /* first usable record of paging device */
     2 pd_map_addr fixed bin,                      /* core address of base of paging device map */
     2 nrecs_pdmap fixed bin,                      /* number of records in pd map */
     2 pd_free fixed bin,                          /* number of free records on the paging device */
     2 pd_using fixed bin,                         /* actual number of pd records being used */
     2 pd_wtct fixed bin,                          /* number of read/write sequences queued */
     2 pd_writes fixed bin,                        /* total number of read/write sequences ever made */
     2 pd_ceiling fixed bin,                       /* number of times too many rws active at once */
     2 pd_steps fixed bin,                         /* total steps taken around the pd map */
     2 pd_skips_incore fixed bin,                  /* number of entries skipped because page was in core */
     2 pd_skips_rws fixed bin,                     /* number of entries skipped because a rws was active */
     2 pd_needed fixed bin,                        /* total number of pd records needed */
     2 mod_during_write fixed bin,                 /* times a page was modified while it was being written */
     2 pd_write_aborts fixed bin,                  /* number of pd writes aborted */
     2 pd_rws_active fixed bin,                    /* count of current number of active rws's */
     2 pd_no_free fixed bin,                       /* number of times couldn't find a free pd record */
     2 pd_read_truncates fixed bin,                /* number of truncated pages during read of rws */
     2 pd_write_truncates fixed bin,               /* number of truncated pages during write of rws */
     2 pd_htsize fixed bin,                        /* number of entries in pd hash table */
     2 pd_hash_mask bit (18),                      /* mask used in pd hashing algorithm */
     2 pdmap_astep ptr,                            /* pointer to temporary segment for pdmap copying */
     2 zero_pages fixed bin,                       /* count of pages truncated because all zero */
     2 pd_zero_pages fixed bin,                    /* as above except also on paging device */
     2 trace_sw aligned,                           /* tracing control flags */
       3 pad_trace bit (32) unaligned,
       3 pc_trace_pf bit (1) unaligned,            /* tracing for page faults, done, etc. */
       3 tty_trace bit (1) unaligned,
       3 pc_trace bit (1) unaligned,               /* flag used by page control primitives */
       3 sc_trace bit (1) unaligned,               /* flag used by segment control primitives */
     2 new_pages fixed bin,                        /* newly created pages */
     2 rws_time_temp fixed bin (71),               /* temporary used for rws metering */
     2 rws_time_start fixed bin (71),              /* time spent initiating rws */
     2 rws_time_done fixed bin (71),               /* time spent finishing up rws */
     2 pd_time_counts (4) fixed bin,               /* number of hits in the following bins */
     2 pd_time_values (4) fixed bin (71),          /* total residency time for the 4 bins */
     2 pd_no_free_gtpd fixed bin,                  /* times pages written to disk because gtpd ON */
     2 pd_page_faults fixed bin,                   /* total page faults from pd */
     2 pd_no_free_first fixed bin,                 /* times pages were written to disk because first  write */
     2 update_index fixed bin,                     /* temporary used during paging device map update */
     2 last_update fixed bin (71),                 /* time last paging device update was performed */
     2 count_pdmes fixed bin,                      /* if non-zero, pdme statistics will be kept */
     2 bucket_overflow fixed bin,                  /* counter for overflows */
     2 buckets (0:63) fixed bin,                   /* buckets for pdme stats */
     2 ast_track bit (1) aligned,                  /* "1"b => keep SST name table */
     2 dirlock_writebehind fixed bin,              /* =1 to flush modified dir pages in lock$unlock */
     2 write_limit fixed bin,                      /* Max # of outstanding writes by page control */
     2 pad4 (1) fixed bin;                         /* padding to 512 words (1000)8 */

/* END INCLUDE FILE sst.incl.pl1 */
```

```
/* Begin include file sstnt.incl.pl1   */

/* Created 10/03/74 by Bernard Greenberg */
/* modified 08/24/79 by J. A. Bush  for easier calculation of size of sstnt */

dcl  sst_names_$ ext;                                      /* Segment containing sst name table */

dcl  sstnp ptr;                                            /* Pointer to sst name segment */

dcl 1 sstnt based (sstnp) aligned,                         /* Major structure */
    2 valid bit (1) aligned,                               /* 1 => structure filled by Multics */
    2 multics_or_bos char (4) aligned,                     /* Origin of data in table */
    2 nentries fixed bin,                                  /* number of entries in the sstnt */
    2 pad1 (5) fixed bin,

    2 (ast_sizes,                                          /* Sizes of ASTE's at each level */
     ast_name_offsets,                                     /* Starting index for names at each level */
     ast_offsets,                                          /* Starting rel addr of each AST region */
     pad2) (0 : 3) fixed bin,

    2 names (0 : 0 refer (sstnt.nentries)) char (32) varying; /* Names of AST entries */

dcl (sstnmx, ptsi_a) fixed bin (17);                       /* Index into name table */

dcl  nm_astep ptr;                                         /* astep to be used */

/* End include file sstnt.incl.pl1 */
```

```
/* BEGIN INCLUDE FILE ... stack_0_data.incl.pl1 */

/* Created 790509 by Mike Grady */

dcl  stack_0_data$ fixed bin ext;                        /* shared stack 0 data base seg */
dcl  stack_0_data_init_number_of_stacks fixed bin;       /* Make PL/I work */
dcl  sdtp ptr;

dcl 1 sdt aligned based (sdtp),                          /* stack 0 database */
    2 lock bit (36),                                     /* lock before changing threads */
    2 num_stacks fixed bin,                              /* number of stacks in pool */
    2 freep bit (18),                                    /* head of free thread, managed LIFO */
    2 pad fixed bin,
    2 stacks (stack_0_data_init_number_of_stacks
         refer (sdt.num_stacks)) like sdte;

dcl  sdtep ptr;

dcl 1 sdte aligned based (sdtep),                        /* stack data table entry */
    2 nextp bit (18) unal,                               /* thread to next free entry (if free) */
    2 pad bit (18) unal,
    2 astep bit (18) unal,                               /* ptr to ASTE for this stack seg */
    2 aptep bit (18) unal,                               /* ptr to APTE of process using this stack, if not free */
    2 sdw bit (72);                                      /* SDW for this stack seg */

/* END INCLUDE FILE ... stack_0_data.incl.pl1 */
```

```
/*       BEGIN INCLUDE FILE ... stack_frame.incl.pl1 ... */

/* Modified: 16 Dec 1977, D. Levin - to add fio_ps_ptr and pl1_ps_ptr */
/* Modified:  3 Feb 1978, P. Krupp - to add run_unit_manager bit & main_proc bit */
/* Modified: 21 March 1978, D. Levin - change fio_ps_ptr to support_ptr */

dcl  sp pointer;                                        /* pointer to beginning of stack frame */

dcl  stack_frame_min_length fixed bin static init(48);


dcl 1 stack_frame based(sp) aligned,
    2 pointer_registers(0 : 7) ptr,
    2 prev_sp pointer,
    2 next_sp pointer,
    2 return_ptr pointer,
    2 entry_ptr pointer,
    2 operator_and_lp_ptr ptr,                          /* serves as both */
    2 arg_ptr pointer,
    2 static_ptr ptr unaligned,
    2 support_ptr ptr unal, /* only used by fortran I/O */
    2 on_unit_relp1 bit(18) unaligned,
    2 on_unit_relp2 bit(18) unaligned,
    2 translator_id bit(18) unaligned,                  /* Translator ID
                                                           0 => PL/I version II
                                                           1 => ALM
                                                           2 => PL/I version I
                                                           3 => signal caller frame
                                                           4 => signaller frame */
    2 operator_return_offset bit(18) unaligned,
    2 x(0: 7) bit(18) unaligned,                        /* index registers */
    2 a bit(36),                                        /* accumulator */
    2 q bit(36),                                        /* q-register */
    2 e bit(36),                                        /* exponent */
    2 timer bit(27) unaligned,                                    /* timer */
    2 pad bit(6) unaligned,
    2 ring_alarm_reg bit(3) unaligned;


dcl 1 stack_frame_flags based(sp) aligned,
    2 pad(0 : 7) bit(72),                               /* skip over prs */
    2 xx0 bit(22) unal,
    2 main_proc bit(1) unal,                            /* on if frame belongs to a main procedure */
    2 run_unit_manager bit(1) unal,                     /* on if frame belongs to run unit manager */
    2 signal bit(1) unal,                               /* on if frame belongs to logical signal_ */
    2 crawl_out bit(1) unal,                            /* on if this is a signal caller frame */
    2 signaller bit(1) unal,                            /* on if next frame is signaller's */
    2 link_trap bit(1) unal,                            /* on if this frame was made by the linker */
    2 support bit(1) unal,                              /* on if frame belongs to a support proc */
```

```
    2 condition bit(1) unal,                    /* on if condition established in this frame */
    2 xx0a bit(6) unal,
    2 xx1 fixed bin,
    2 xx2 fixed bin,
    2 xx3 bit(25) unal,
    2 old_crawl_out bit (1) unal,               /* on if this is a signal caller frame */
    2 old_signaller bit(1) unal,                /* on if next frame is signaller's */
    2 xx3a bit(9) unaligned,
    2 xx4(9) bit(72) aligned,
    2 v2_pl1_op_ret_base ptr,                   /* When a V2 PL/I program calls an operator the
                                                *  operator puts a pointer to the base of
                                                *  the calling procedure here. (text base ptr)  */
    2 xx5 bit(72) aligned,
    2 pl1_ps_ptr ptr;                           /* ptr to ps for this frame; also used by fio. */

/*        END INCLUDE FILE ... stack_frame.incl.pl1 */
```

```
/*      BEGIN INCLUDE FILE ... stack_header.incl.pl1 .. 3/72 Bill Silver  */
/*      modified 7/76 by M. Weaver for *system links and more system use of areas */
/*      modified 3/77 by M. Weaver to add rnt_ptr */


dcl  sb    ptr;                                          /* the  main pointer to the stack header */

dcl 1 stack_header      based (sb) aligned,

    2 pad1 (4)          fixed bin,              /*  (0) also used as arg list by outward_call_handler  */
    2 old_lot_ptr       ptr,                    /*  (4)  pointer to the lot for current ring (obsolete) */
    2 combined_stat_ptr ptr,                    /*  (6)  pointer to area containing separate static */

    2 clr_ptr           ptr,                    /*  (8)  pointer to area containing linkage sections */
    2 max_lot_size      fixed bin(17) unal,     /*  (10) DU  number of words allowed in lot */
    2 main_proc_invoked fixed bin (11) unal,    /*  (10) DL  nonzero if main procedure invoked in run unit */
    2 run_unit_depth    fixed bin(5) unal,      /*  (10) DL  number of active run units stacked */
    2 cur_lot_size      fixed bin(17) unal,     /*  (11)  number of words (entries) in lot */

    2 system_free_ptr   ptr,                    /*  (12)  pointer to system storage area */
    2 user_free_ptr     ptr,                    /*  (14)  pointer to user storage area */

    2 null_ptr          ptr,                    /*  (16)  */
    2 stack_begin_ptr   ptr,                    /*  (18)  pointer to first stack frame on the stack */
    2 stack_end_ptr     ptr,                    /*  (20)  pointer to next useable stack frame */
    2 lot_ptr           ptr,                    /*  (22)  pointer to the lot for the current ring */

    2 signal_ptr        ptr,                    /*  (24)  pointer to signal procedure for current ring */
    2 bar_mode_sp       ptr,                    /*  (26)  value of sp before entering bar mode */
    2 pl1_operators_ptr ptr,                    /*  (28)  pointer to pl1_operators_$operator_table */
    2 call_op_ptr       ptr,                    /*  (30)  pointer to standard call operator */

    2 push_op_ptr       ptr,                    /*  (32)  pointer to standard push operator */
    2 return_op_ptr     ptr,                    /*  (34)  pointer to standard return operator */
    2 return_no_pop_op_ptr ptr,                 /*  (36)  pointer to standard return / no pop operator */
    2 entry_op_ptr      ptr,                    /*  (38)  pointer to standard entry operator */

    2 trans_op_tv_ptr   ptr,                    /*  (40)  pointer to translator operator ptrs */
    2 isot_ptr          ptr,                    /*  (42)  pointer to ISOT */
    2 sct_ptr           ptr,                    /*  (44)  pointer to System Condition Table */
    2 unwinder_ptr      ptr,                    /*  (46)  pointer to unwinder for current ring */

    2 sys_link_info_ptr ptr,                    /*  (48)  pointer to *system link name table */
    2 rnt_ptr           ptr,                    /*  (50)  pointer to Reference Name Table */
    2 ect_ptr           ptr,                    /*  (52)  pointer to event channel table */
    2 assign_linkage_ptr ptr,                   /*  (54)  pointer to storage for (obsolete) hcs_$assign_linkage */
    2 pad3 (8)          bit (36) aligned;       /*  (56)  for future expansion */
```

```
/*        The following offset refers to a table within the  pl1  operator table.  */

  dcl     tv_offset            fixed bin        init(361)  internal static;  /* (551) octal */

/*        The following constants are offsets within this transfer vector table.  */

  dcl    (call_offset          fixed bin        init(271),
          push_offset          fixed bin        init(272),
          return_offset        fixed bin        init(273),
          return_no_pop_offset fixed bin        init(274),
          entry_offset         fixed bin        init(275))  internal static;



/*        The following declaration  is an overlay of the whole stack header.   Procedures which
          move the whole stack header should use this overlay.
*/

  dcl     stack_header_overlay (size(stack_header))        fixed bin      based (sb);


/*        END INCLUDE FILE ... stack_header.incl.pl1 */
```

```
/*  START OF:       stock_seg.incl.pl1           *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  */

     dcl     stock_segp            ptr;
     dcl     record_stockp         ptr;
     dcl     vtoce_stockp          ptr;
     dcl     stock_seg$            ext;

     dcl     n_in_record_stock     fixed bin;
     dcl     n_volmap_pages        fixed bin;
     dcl     n_in_vtoce_stock      fixed bin;


     dcl     1 stock_seg           aligned based (stock_segp),

               2 meters            aligned like rsmeters,

               2 record_stock_entries fixed bin,          /* Number of entries in a record stock */
               2 vtoce_stock_entries fixed bin,           /* Number of entries in a VTOCE stock */
               2 record_stock_size  fixed bin,            /* Size of a record stock in words */
               2 vtoce_stock_size   fixed bin,            /* Size of a VTOCE stock in words */
               2 n_stock_entries    fixed bin,            /* Number of stocks of each type */
               2 record_stock_arrayp ptr,                 /* Record stock region */
               2 vtoce_stock_arrayp ptr;                  /* VTOCE stock region */


     dcl     1 record_stock        aligned based (record_stockp),

               2 pvtep             ptr unal,              /* PVTE for this stock */

               2 n_in_stock        fixed bin (18) uns unal,/* Max number of addresses in stock */
               2 n_volmap_pages    fixed bin (18) uns unal,/* Number of pages in Volume Map */

               2 n_free_in_stock   fixed bin (18) uns unal,/* Number addresses currently free */
               2 n_os_in_stock     fixed bin (18) uns unal,/* Number addresses currently out-of-service */

               2 low_threshold     fixed bin (18) uns unal,/* Low threshold for withdrawing from volmap */
               2 high_threshold    fixed bin (18) uns unal,/* High threshold for depositing to volmap */

               2 target            fixed bin (18) uns unal,/* Target for stock */
               2 stock_offset      bit (18) unal,         /* Offset of stock in this structure */

               2 n_words_in_stock  fixed bin (18) uns unal,/* Number of words = Number of entries / 2 */
               2 search_index      fixed bin (18) uns unal,/* Roving pointer */

               2 old_volmap_page   (3) aligned,           /* N_OLD_VOLMAP_PAGES (cif) */
                 3 last            fixed bin (18) uns unal,/* Roving pointer */
                 3 pad             bit (18) unal,

               2 volmap_page       (n_volmap_pages refer (record_stock.n_volmap_pages)) aligned,
```

```
            3 n_free            fixed bin (18) uns unal,/* Number free records in this volmap page */
            3 baseadd           fixed bin (17) unal,    /* First record address described by this page */

          2 stock               (n_in_record_stock refer (record_stock.n_in_stock)) bit (18) unal; /* Stock array of addresses *
/
                                                        /* bit 0 ON => out-of-service */


     dcl     1 vtoce_stock              aligned based (vtoce_stockp),
          2 pvtep               ptr unal,               /* PVTE for this stock */
          2 n_in_stock          fixed bin (18) uns unal,/* Max number indices in stock */
          2 n_free_in_stock     fixed bin (18) uns unal,/* Number indices currently free */
          2 target              fixed bin (18) uns unal,/* Target when withdrawing/depositing */
          2 search_index        fixed bin (18) uns unal,/* Roving pointer */
          2 stock               (n_in_vtoce_stock refer (vtoce_stock.n_in_stock)) fixed bin (17) unal; /* Stock array of VTOCE i
ndices */


     dcl     1 rsmeters                 aligned based,

          2 async_read_calls    fixed bin (35),         /* Number of asynchronous read attempts */
          2 async_page_reads    fixed bin (35),         /* Number of times page read was required */
          2 async_post_io_calls fixed bin (35),         /* Number of times read or write posted */
          2 deposit_calls       fixed bin (35),         /* Number of times deposit called */
          2 async_post_io_time  fixed bin (71),         /* CPU time posting I/Os (interrupt side) */
          2 deposit_time        fixed bin (71),         /* CPU time in deposit (call side) */
          2 low_thresh_detected fixed bin (35),         /* Number of times stock below low threshold */
          2 high_thresh_detected fixed bin (35),        /* Number of times stock above high threshold */
          2 low_thresh_fails    fixed bin (35),         /* Number of times no records in volmap */
          2 withdraw_stock_steps fixed bin (35),        /* Number steps thru stock in withdraw */
          2 withdraw_stock_losses fixed bin (35),       /* Number lockless losses */
          2 n_withdraw_attempt  fixed bin (35),         /* Number attempts to withdraw a page */
          2 n_withdraw_range    fixed bin (35),         /* Number attempts to withdraw within range */
          2 n_pages_withdraw_stock fixed bin (35),      /* Number pages withdrawn from stock */
          2 n_pages_withdraw_async fixed bin (35),      /* Number pages withdrawn from volmap */
          2 n_v_withdraw_attempts fixed bin (35),       /* Number attempts to withdraw from volmap */
          2 withdraw_volmap_steps fixed bin (35),       /* Number steps thru volmap in withdraw */
          2 deposit_stock_steps fixed bin (35),         /* Number steps thru stock in deposit */
          2 deposit_stock_losses fixed bin (35),        /* Number lockless losses */
          2 n_deposit_attempt   fixed bin (35),         /* Number attempts to deposit a page */
          2 n_pages_deposit_stock fixed bin (35),       /* Number pages deposited to stock */
          2 n_pages_deposit_volmap fixed bin (35),      /* Number pages deposited to volmap */
          2 n_v_deposit_attempts fixed bin (35),        /* Number attempts to deposit to volmap */
          2 reset_os_calls      fixed bin (35),         /* Number calls to reset_os */
          2 reset_os_losses     fixed bin (35),         /* Number lockless losses */
          2 withdraw_calls      fixed bin (35),         /* Number calls to withdraw */
          2 withdraw_time       fixed bin (71),         /* CPU time in withdraw (page-fault) */
          2 pc_deposit_time     fixed bin (71),         /* CPU time in pc_deposit */
          2 pc_deposit_calls    fixed bin (35),         /* Number calls to pc_deposit */
          2 pc_deposit_pages    fixed bin (35),         /* Number pages deposited by pc_deposit */
          2 get_free_vtoce_calls fixed bin (35),        /* Number calls to get_free_vtoce */
          2 return_free_vtoce_call fixed bin (35),      /* Number calls to return_free_vtoce */
          2 deposit_vstock_calls fixed bin (35),        /* Number attempts to deposit to vtoce stock */
          2 deposit_vstock_fails fixed bin (35),        /* Number times deposit failed */
          2 withdraw_vstock_calls fixed bin (35),       /* Number attempts to withdraw from vtoce stock */
```

```
        2 withdraw_vstock_fails fixed bin (35),       /* Number times withdraw failed */
        2 deposit_vtoc_map fixed bin (35),            /* Number times vtoce deposited to map */
        2 withdraw_check_scav fixed bin (35),         /* Number times withdraw checked an address for scavenge */
        2 withdraw_conflict fixed bin (35),           /* Number times conflict found */
        2 pad (11)          fixed bin (35);

dcl     N_OLD_VOLMAP_PAGES     fixed bin init (3) int static options (constant);
dcl     DEFAULT_N_IN_RECORD_STOCK fixed bin init (104) int static options (constant);
dcl     DEFAULT_N_IN_VTOCE_STOCK fixed bin init (10) int static options (constant);

/*  END OF:        stock_seg.incl.pl1              *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  */
```

```
/*  BEGIN INCLUDE FILE ... str.incl.pl1 ... last modified March 1970 */

dcl str_seg$ ext,
    strp ptr;

dcl 1 str based (strp) aligned,                /* segment or process trailer declaration */

   (2 fp bit (18),                             /* forward ast trailer rel pointer */
    2 bp bit (18),                             /* backward ast trailer rel pointer*/

    2 segno bit (18),                          /* segment number*/
    2 dstep bit (18)) unaligned;               /* rel pointer to ring 0 dste */

dcl stra (0:8000) bit (72) based (strp) aligned;

/*  END INCLUDE FILE ... str.incl.pl1 */
```

```
/* **********************************************************
   *                                                        *
   * Copyright, (C) Honeywell Information Systems Inc., 1982 *
   *                                                        *
   ********************************************************** */
/* TC_DATA - This is the Traffic Controller Database. */
/* Last modified (Date and reason):
   2/6/76 by S. Webber Initial coding
   6/20/79 by Mike Grady to init max_maxe
   3/4/81 by J. Bongiovanni not to set prds_length (it's done from the header
         or the TBLS Config Card)
   3/21/81 by J. Bongiovanni for max_stopped_stack_0, initialization NTO,
         response time metering
   6/27/81 by J. Bongiovanni for tuning parameter changes (-tcpu, +pre_empt_sample_time,
         gp_at_notify and gp_at_ptlnotify off by default
   1/82 BIM for stk truncation tuning parms.
   4/27/82 by J. Bongiovanni to change post_purge to OFF
   August 1982, J. Bongiovanni, for realtime_io parameters
*/

/* ****************************************************
   *                                                  *
   *                                                  *
   * Copyright (c) 1972 by Massachusetts Institute of *
   * Technology and Honeywell Information Systems, Inc. *
   *                                                  *
   *                                                  *
   **************************************************** */

tc_data: proc;

/* This program creates the tc_data base */

/* Automatic */

dcl 1 cdsa aligned like cds_args;
dcl  code fixed bin (35);
dcl  big_time fixed bin (71);

/* Based */

dcl 1 tc_data aligned like tcm based (tcmp);

/* Static */

dcl  exclude_pad (1) char (32) aligned static options (constant) init ("pad*");

/* Builtins */
```

```
dcl (addr, bin, null, rel, size, string, unspec) builtin;

/* Entries */

dcl  com_err_ entry options (variable);
dcl  create_data_segment_ entry (ptr, fixed bin (35));
dcl  get_temp_segment_ entry (char (*), ptr, fixed bin (35));
dcl  release_temp_segment_ entry (char (*), ptr, fixed bin (35));
```

```
        call get_temp_segment_ ("tc_data", tcmp, code);

/* Check offsets assumed by BOS */

        call check_offset_for_bos (addr (tc_data.apt_offset), 171, "apt_offset");
        call check_offset_for_bos (addr (tc_data.apt_size), 203, "apt_size");
        call check_offset_for_bos (addr (tc_data.apt_entry_size), 215, "apt_entry_size");


        tc_data.apt_offset = rel (addr (tc_data.apt));
        tc_data.apt_lock = -1;                          /* -1 = unlocked */
        tc_data.metering_lock = -1;                     /* 0 = locked */
        tc_data.working_set_factor = 1;
        tc_data.ncpu = 0;
        tc_data.itt_size = 155;
        tc_data.dst_size = 155;
        tc_data.initializer_id = (36)"1"b;
        tc_data.max_eligible = 6*262144;
        tc_data.max_max_eligible = 16*262144;
        tc_data.max_stopped_stack_0 = 4;
        tc_data.apt_entry_size = size (apte);
        tc_data.pds_length = 1024;

        tc_data.interactive_q.fp = rel (addr (tc_data.interactive_q));
        tc_data.interactive_q.bp = rel (addr (tc_data.interactive_q));
        tc_data.interactive_q.sentinel = (36)"1"b;

        tc_data.max_hproc_segno = 191;                  /* largest (default) hardcore segment number */
        tc_data.dst_ptr = null;
        tc_data.old_user = null;
        tc_data.tefirst = 2000000;
        tc_data.telast = 2000000;
        tc_data.timax = 8000000;
        tc_data.process_initial_quantum = 2000000;
        tc_data.gp_at_notify = 0;                       /* off by default */
        tc_data.gp_at_ptlnotify = 0;                    /* off by default */
        tc_data.pre_empt_sample_time = 40000;           /* 40 milliseconds */
        tc_data.max_timer_register = 40000;

        tc_data.sort_to_elhead = 1;
        tc_data.auto_tune_ws = 1;
        tc_data.ocore = .01b;
        tc_data.stk_truncate = "1"b;
        tc_data.stk_truncate_always = "0"b;

/* See fast_hc_ipc, but the rolling average of steps/block is calculated */
/* as NEW_AVERAGE = factor*NEW_VALUE + (1-factor)*OLD_AVERAGE */

        tc_data.stk_trunc_avg_f1 = 0.9375; /* 15/16 */
        tc_data.stk_trunc_avg_f2 = 1b - tc_data.stk_trunc_avg_f1;

        tc_data.lock_error_severity = 3; /* BEEP BEEP but no crash */
```

```
tc_data.realtime_q.fp = rel (addr (tc_data.realtime_q));
tc_data.realtime_q.bp = rel (addr (tc_data.realtime_q));
tc_data.realtime_q.sentinel = (36)"1"b;

tc_data.eligible_q_head.fp = rel (addr (tc_data.eligible_q_tail));
tc_data.eligible_q_head.bp = "0"b;
tc_data.eligible_q_head.sentinel = (36)"1"b;

tc_data.eligible_q_tail.fp = rel (addr (tc_data.idle_tail));
tc_data.eligible_q_tail.bp = rel (addr (tc_data.eligible_q_head));
tc_data.eligible_q_tail.sentinel = (36)"1"b;

tc_data.idle_tail.fp = "0"b;
tc_data.idle_tail.bp = rel (addr (tc_data.eligible_q_tail));
tc_data.idle_tail.sentinel = (36)"1"b;

tc_data.min_eligible = 2*262144;
tc_data.guaranteed_elig_inc = 250000;
tc_data.priority_sched_inc = 80000000;
tc_data.int_q_enabled = 1;
tc_data.fnp_buffer_threshold = 30;                /* fnp tries to keep > this many free buff */

unspec (big_time) = "000000000000000000111111111111111111111111111111111111111111111111111111"b;
tc_data.end_of_time = big_time;
tc_data.next_alarm_time = big_time;               /* gets zeroed by tc_init$part_2 */
tc_data.priority_sched_time = big_time;
tc_data.tty_polling_time = big_time;
tc_data.disk_polling_time = big_time;
tc_data.tape_polling_time = big_time;
tc_data.imp_polling_time = big_time;
tc_data.mos_polling_time = big_time;
tc_data.volmap_polling_time = big_time;
tc_data.realtime_io_deadline = 0;
tc_data.realtime_io_quantum = 5000;               /* 5 milliseconds */

tc_data.max_channels = 6;

tc_data.init_wait_timeout = 5000000;              /* 5 second NTO during initialization            */
tc_data.init_timeout_severity = 0;                /* beep                                          */
tc_data.vcpu_response_bounds_size = VCPU_RESPONSE_BOUNDS;
tc_data.vcpu_response_bounds (1) = 500000;        /* 1/2 second                                    */
tc_data.vcpu_response_bounds (2) = 1000000;       /* 1 second                                      */
tc_data.vcpu_response_bounds (3) = 10000000;      /* 10 seconds                                    */


tc_data.default_procs_required = (8) "1"b;        /* all CPUs */
/* Now set up call to create data base */

cdsa.sections (1).p = addr (tc_data);
cdsa.sections (1).len = size (tc_data);
cdsa.sections (1).struct_name = "tc_data";

cdsa.seg_name = "tc_data";
cdsa.num_exclude_names = 1;
cdsa.exclude_array_ptr = addr (exclude_pad);
```

```
string (cdsa.switches) = "0"b;
cdsa.switches.have_text = "1"b;

call create_data_segment_ (addr (cdsa), code);

call release_temp_segment_ ("tc_data", tcmp, code);
```

```
check_offset_for_bos:
          proc (item_ptr, bos_offset, item_name);

          dcl item_ptr ptr;                                    /* pointer to item in tc_data                  */
          dcl bos_offset fixed bin (18);                       /* location assumed by BOS                     */
          dcl item_name char (*);                              /* name of item in structure                   */

          if bin (rel (item_ptr)) ^= bos_offset
               then call com_err_ (0, "tc_data", "^a not at BOS-assumed offset (^d=^oo)",
               item_name, bos_offset, bos_offset);


     end check_offset_for_bos;
```

```
% include cds_args;
```

```
% include apte;
```

```
% include tcm;

     end tc_data;
```

```
/* BEGIN INCLUDE FILE ... tcm.incl.pl1 ... used to generate tc_data cds */
/* NOTE -- This include file has TWO counterparts in ALM: tc_meters.incl.alm and */
/* wcte.incl.alm. They cannot be produced with cif, and must be kept up to date manually. */


dcl  tcmp ptr;

dcl 1 tcm aligned based (tcmp),
    2 pad_base (3) fixed bin (18),                     /* stat moved from here .. */
    2 cid2 fixed bin (18),
    2 cid3 fixed bin (18),
    2 cid4 fixed bin (18),
    2 depth_count fixed bin (18),                      /* depth last process run */
    2 loadings fixed bin (18),                         /* number of process loadings */

    2 blocks fixed bin (18),                           /* number of calls to block */
    2 wakeups fixed bin (18),                          /* number of calls to wakeup */
    2 waits fixed bin (18),                            /* number of calls to wait */
    2 notifies fixed bin (18),                         /* number of calls to notify */
    2 schedulings fixed bin (18),
    2 interactions fixed bin (18),                     /* number of interactive schedulings */
    2 avequeue fixed bin (35, 18),                     /* recent time average of number in queue */
    2 te_wait fixed bin (18),                          /* times te called from wait */

    2 te_block fixed bin (18),                         /* times te updated from block */
    2 te_i_stop fixed bin (18),                        /* times te updated from i_stop */
    2 te_pre_empt fixed bin (18),                      /* times te updated from pre_empt */
    2 p_interactions fixed bin,                        /* times interaction bit turned off because of high priority */
    2 idle fixed bin (71),                             /* total idle time */
    2 mp_idle fixed bin (71),                          /* multi-programming idle */

    2 nmp_idle fixed bin (71),                         /* non-multi-programming idle time */
    2 zero_idle fixed bin (71),                        /* zero idle time */
    2 last_time fixed bin (71),                        /* last time a process was run */
    2 loop_locks fixed bin (18),                       /* times looped on the APT lock */
    2 loop_lock_time fixed bin (18),                   /* time looping on the APT lock */
    2 ave_eligible fixed bin (35, 18),                 /* average length of eligible queue */
    2 sort_to_elhead fixed bin (18),                   /* 0=> no one,1 => int've only, 2 => everybody */
    2 processor_time fixed bin (71),                   /* total processor time on system */
    2 response_time fixed bin (71),                    /* estimate of response time */
    2 eligible_time fixed bin (71),                    /* estimate of eligible time */
    2 response_count fixed bin,                        /* count of response meters */
    2 eligible_count fixed bin,                        /* count of eligible meters */
    2 quit_counts (0:5) fixed bin,                     /* array of buckets indexed by state */
    2 loading_idle fixed bin (71),                     /* loading_idle time */
    2 delta_vcpu fixed bin (71),                       /* delta virtual CPU time for the system */
    2 post_purge_switch fixed bin,                     /* ON if post purging is to be done */
    2 time_out_severity fixed bin,                     /* syserr first arg for notify time outs */
```

```
    2 notify_check fixed bin,                       /* obsolete */
    2 quit_priority fixed bin,                      /* factor for scheduler quit response */
    2 iobm_polling_time fixed bin (71),             /* time to poll iobm */
    2 end_of_time fixed bin (71),                   /* very large time */
    2 gp_at_notify fixed bin (18),                  /* 0 => just do get_idle_processor */
    2 gp_at_ptlnotify fixed bin (18),               /* 0 => just do get_idle_processor */
    2 int_q_enabled fixed bin (18),                 /* 0 => no intv q in percent mode */
    2 fnp_buffer_threshold fixed bin (18),          /* if fewer free buffs then stingy alloc strategy */
                                                    /* set this to >= half n_ttylines/fnp for safety */

/* 100 octal */

    2 depths (8) fixed bin (18),                    /* histogram of run depths */
    2 tdepths (8) fixed bin (71),                   /* histogram of times run per depth */
    2 pfdepth (8) fixed bin (18),                   /* histogram of page faults per depth */

    2 ptl_not_waits fixed bin (18),                 /* times ptl_wait noticed ptl was unlocked */
    2 gw_gp_window_count fixed bin (18),            /* times window noticed */
    2 metering_lock fixed bin (18),                 /* 0=locked, else unlocked */
    2 ptl_waits fixed bin (18),                     /* num calls to ptl_wait */
    2 gp_start_count fixed bin (18),                /*  to detect gw_gp window lossage */
    2 gp_done_count fixed bin (18),
    2 nto_check_time fixed bin (71),                /* next time at which nto code will be called */
    2 nto_delta fixed bin (35),                     /* microsec between nto checks */
    2 nto_count fixed bin (18),                     /* number of times nto detected */
    2 tcpu_scheduling fixed bin (18),               /* obsolete                              */
    2 nto_event bit (36),                           /* last event which NTO'd */
    2 page_notifies fixed bin (18),
    2 notify_nobody_count fixed bin (18),
    2 notify_nobody_event bit (36),
    2 system_type fixed bin,                        /* used to be tcm.inter */

    2 stat (0:15) fixed bin (18),                   /* num apte's in each state */

/* 200 octal */

    2 wait (8),
      3 time fixed bin (18),                        /* histogram of page fault waiting times versus did */
      3 count fixed bin (18),

    2 ready (8),
      3 time fixed bin (18),                        /* histogram of times in ready queue */
      3 count fixed bin (18),

    2 total_pf_time fixed bin (71),                 /* total time spent from start to end of
                                                       all page faults */
    2 total_pf_count fixed bin (18),                /* total number of page faults metered */
    2 auto_tune_ws fixed bin (18),                  /* 0=> dont, atherwise compensate for quantum len */
    2 ocore_delta fixed bin (18),                   /* number of pages reserved for int users */
    2 ws_sum fixed bin (18),                        /* total of eligible's ws_sizes */
    2 nonidle_force_count fixed bin (18),           /* count of eligibilities forced */
    2 itt_list_lock bit (36) aligned,               /* Lock on ITT free list */
    2 cpu_pf_time fixed bin (71),                   /* total cpu time spent handling page faults */
    2 cpu_pf_count fixed bin (18),                  /* total count of cpu time meterings */
    2 special_offsets unaligned,
```

```
        3 apt_offset bit (18),
        3 pad bit (18),
      2 getwork_time fixed bin (71),                       /* total time spent in getwork */
      2 getwork_count fixed bin (18),                      /* total times through getwork */
      2 short_pf_count fixed bin (18),                     /* number of short page faults */
      2 interrupt_time fixed bin (71),                     /* total time spent in interrupt */
      2 interrupt_count fixed bin (71),                    /* total number of metered interrupts */
      2 ocore fixed bin (35, 18),                          /* fraction of core for int've users */
      2 pre_empt_flag bit (36) aligned,                    /* controls whether preempting at done time */
      2 cumulative_memory_usage fixed binary (71),         /* total number of memory usage units */
      2 processor_time_at_define_wc fixed bin (71),        /* value of processor_time when WC's last defined */
      2 boost_priority fixed bin,                          /* number of times priority process given high priority */
      2 lost_priority fixed bin,                           /* number of times priority process lost eligibility */
      2 total_clock_lag fixed bin (71),                    /* sum of all simulated clock delays */
      2 clock_simulations fixed bin,                       /* number of times alarm clock interrupt was simulated */
      2 max_clock_lag fixed bin,                           /* largest simulated alarm clock delay */

/* 300 octal */

      2 pdscopyl fixed bin (18),                           /* amount of pds to copy for new process */
      2 max_hproc_segno fixed bin,                         /* largest allowed hardcore segment number */
      2 prds_length fixed bin (18),                        /* length of PRDS */
      2 pds_length fixed bin (18),                         /* length of PDS */
      2 lock fixed bin (18),                               /* process id generator lock */
      2 id fixed bin (18),                                 /* next processid to be given out */
      2 system_shutdown fixed bin (18),
      2 working_set_factor fixed bin (35, 18),             /* working set factor */

      2 ncpu fixed bin (18),                               /* number of processors currently being used */
      2 last_eligible bit (18),                            /* last process to gain eligibility */
      2 apt_lock fixed bin (35),                           /* + write; 0 hidden; -1 unlocked; -(N+1) Nreaders */
      2 apt_size fixed bin (18),                           /* number of APT entries */
      2 realtime_q aligned like based_sentinel,            /* processes with realtime deadlines */
      2 aht_size fixed bin (18),                           /* APT hash table size */
      2 itt_size fixed bin (18),                           /* number of ITT entries */

      2 dst_size fixed bin (18),                           /* number of allowed DST entries */
      2 itt_free_list bit (18),                            /* pointer to ITT free list */
      2 used_itt fixed bin (18),                           /* number of used ITT entries */
      2 initializer_id bit (36) aligned,                   /* process id of initializer */
      2 n_eligible fixed bin (18),                         /* number of processes eligible */
      2 max_eligible fixed bin (30),                       /* maximum allowed number of eligible processes */
      2 wait_enable fixed bin (18),                        /* turned on when waiting mechanism works */
      2 apt_entry_size fixed bin (18),                     /* size of an APT entry */

      2 interactive_q aligned like based_sentinel,         /* head of interactive queue */
      2 dst_ptr ptr,                                       /* pointer to device signal table */
      2 old_user ptr,                                      /* last process to run (apt ptr ) */
      2 initialize_time fixed bin (71),                    /* time of initialization */

      2 init_event fixed bin (18),                         /* wait event during initialization */
      2 oldt fixed bin (18),                               /* timer reading from previous process */
      2 newt fixed bin (18),                               /* timer setting for new process */
      2 tefirst fixed bin (30),                            /* first eligible time */
      2 telast fixed bin (30),                             /* last eligible time */
```

```
    2 timax fixed bin (35),                                   /* time in queue for lowest level */
    2 empty_q bit (18),                                       /* thread of empty APT entries */
    2 working_set_addend fixed bin (18),                      /* additive working set parameter */
    2 ready_q_head bit (0) aligned,                           /* for added segdef */
    2 eligible_q_head aligned like based_sentinel,            /* head of eligible queue */
    2 ready_q_tail bit (0) aligned,                           /* for added segdef */
    2 eligible_q_tail aligned like based_sentinel,            /* tail of eligible queue */
    2 idle_tail aligned like based_sentinel,                  /* tail of idle list */
    2 min_eligible fixed bin (30),
    2 alarm_timer_list bit (18) aligned,                      /* rel pointer to apt entry for next alarm timer */
    2 guaranteed_elig_inc fixed bin (35),                     /* amount of guaranteed eligibility time in microsecs. */
    2 priority_sched_inc fixed bin (35),                      /* amount of block time before process is given priority */
    2 next_alarm_time fixed bin (71),                         /* clock time for next alarm timer */
    2 priority_sched_time fixed bin (71),                     /* time for priority process to be given priority */
    2 tty_polling_time fixed bin (71),                        /* time to poll TTY DIM */
    2 disk_polling_time fixed bin (71),                       /* time to poll disk DIM */
    2 tape_polling_time fixed bin (71),                       /* time to poll tape DIM */
    2 imp_polling_time fixed bin (71),                        /* time to poll imp */
    2 imp_polling_lock fixed bin (18),                        /* do not poll if lock set */
    2 max_channels fixed bin (18),                            /* num special channels per process */

/* 400 octal */

    2 system_virtual_time fixed bin (71),                     /* non-idle virtual time */
    2 credit_bank fixed bin (71),                             /* credits not yet passed out */
    2 min_wct_index bit (18) aligned,                         /* offset of initializer work class table entry */
    2 max_wct_index bit (18) aligned,                         /* offset of highest wcte currently defined */
    2 delta_vt fixed bin (71),                                /* temp used by pxss.compute_virtual_clocks */
    2 gross_idle_time fixed bin (71),                         /* idle time_used_clock */
    2 credits_per_scatter fixed bin (35),                     /* total number of credits awarded at once */
    2 best_credit_value fixed bin (18),                       /* temp for pxss.find_next_eligible */
    2 define_wc_time fixed bin (71),                          /* clock time when workclasses last degined */
    2 max_batch_elig fixed bin (35),
    2 num_batch_elig fixed bin (35),
    2 deadline_mode fixed bin (35),                           /* 0=> ti sorts, else deadline sorts */
    2 credits_scattered fixed bin (35),
    2 max_max_eligible fixed bin (30),                        /* Maximum of maxe */
    2 max_stopped_stack_0 fixed bin (35),                     /* Maximum stack_0's suspended by stopped procs    */
    2 stopped_stack_0 fixed bin (35),                         /* Number stack_0's suspended by stopped procs     */
    2 mos_polling_interval fixed bin (35),                    /* for heals */
    2 mos_polling_time fixed bin (71),                        /* for heals */
    2 vcpu_response_bounds (VCPU_RESPONSE_BOUNDS) fixed bin (35),
    2 vcpu_response_bounds_size fixed bin (35),
    2 meter_response_time_calls fixed bin (35),
    2 meter_response_time_invalid fixed bin (35),
    2 meter_response_time_overhead fixed bin (71),
    2 init_wait_time fixed bin (71),                          /* used by wait/notify during initialization      */
    2 init_wait_timeout fixed bin (71),                       /* notify-timeout interval during initialization  */
    2 init_timeout_severity fixed bin,                        /* notify-timeout severity during initialization  */
    2 init_timeout_recurse fixed bin,                         /* count of NTO recursion during initialization   */
    2 max_timer_register fixed bin (71),                      /* max cpu burst = # cpus x pre_empt_sample_time  */
    2 pre_empt_sample_time fixed bin (35),                    /* tuning parameter - max time between samples    */
    2 governing_credit_bank fixed bin (35),                   /* used for limiting eligibility on governed work classes*/
    2 process_initial_quantum fixed bin (35),                 /* eligibility quantum first eligibility          */
    2 default_procs_required bit (8) aligned,                 /* default mask of CPUs required */
```

```
    2 work_class_idle fixed bin (71),                         /* idle time due to work class restrictions */

/* Tuning Parameters for Stack Truncation */

    2 stk_truncate bit (1) aligned,
    2 stk_truncate_always bit (1) aligned,
    2 stk_trunc_avg_f1 fixed bin (35, 18),
    2 stk_trunc_avg_f2 fixed bin (35, 18),
    2 lock_error_severity fixed bin,                          /* syserr severity */

    2 gv_integration fixed bin (35),                          /* Integration interval for governing */
    2 gv_integration_set bit (1) aligned,                     /* ON => gv_integration set by ctp */
    2 pad8 fixed bin (35),
    2 volmap_polling_time fixed bin (71),
    2 pad9 fixed bin (71),
    2 realtime_io_priority_switch fixed bin,                  /* 0 => give I/O interrupt wakeups realtime priotiry */
    2 realtime_io_deadline fixed bin (35),                    /* Delta to clock for I/O realtime deadline */
    2 realtime_io_quantum fixed bin (35),                     /* Quantum for I/O realtime burst */
    2 realtime_priorities fixed bin (35),                     /* Count for metering */
    2 relinquishes fixed bin (35),                            /* Calls to relinquish_priority */
    2 abort_ips_mask bit (36) aligned,                        /* IPS mask for tc_util$check_abort */

/* 500 octal */

    2 pad5 (192) fixed bin (35),                              /* room for expansion compatibly                  */

/* 1000 octal */

    2 pad7 (64) fixed bin (35),

/* 1100 octal */

    2 pad6 (8) fixed bin (35),
    2 work_class_table aligned,                               /* array of per workclass information */
      3 wcte (0:16) aligned like wct_entry,

/* 3000 octal */

    2 apt fixed bin;

dcl  wctep ptr;

dcl 1 wct_entry aligned based (wctep),                        /* Work class entry */
    2 thread unaligned,                                       /* Ready list */
      3 fp bit (18),                                          /* Head of ready list */
      3 bp bit (18),                                          /* Tail of ready list */
    2 flags unaligned,
      3 mnbz bit (1),                                         /* Sentinel bit must not be zero. */
      3 defined bit (1),
      3 io_priority bit (1),
      3 governed bit (1),
      3 interactive_q bit (1),
      3 pad bit (31),
    2 credits fixed bin (35),                                 /* Current worthiness of group */
    2 minf fixed bin (35),                                    /* min fraction of cpu */
```

```
    2 pin_weight fixed bin (35),                          /* number of cycles to pin pages */
    2 eligibilities fixed bin (35),                       /* Count of eligibilities awarded */
    2 cpu_sum fixed bin (71),                             /* CPU used by members */
    2 resp1 fixed bin (71),
    2 resp2 fixed bin (71),
    2 quantum1 fixed bin (35),
    2 quantum2 fixed bin (35),
    2 rmeter1 fixed bin (71),
    2 rmeter2 fixed bin (71),
    2 rcount1 fixed bin (35),
    2 rcount2 fixed bin (35),
    2 realtime fixed bin (35),
    2 purging fixed bin (35),
    2 maxel fixed bin (35),
    2 nel fixed bin (35),
    2 number_thinks fixed bin (35),                       /* number times process entered "think" state     */
    2 number_queues fixed bin (35),                       /* number times process entered "queued" state    */
    2 total_think_time fixed bin (71),
    2 total_queue_time fixed bin (71),

/*  The next three arrays  correspond to the array vcpu_response_bounds                                      */

    2 number_processing (VCPU_RESPONSE_BOUNDS+1) fixed bin (35), /* number times entered "processing" state   */
    2 total_processing_time (VCPU_RESPONSE_BOUNDS+1) fixed bin (71),
    2 total_vcpu_time (VCPU_RESPONSE_BOUNDS+1) fixed bin (71),
    2 maxf fixed bin (35),                                /* maximum fraction of cpu time                   */
    2 governing_credits fixed bin (35),                   /* for limiting cpu resources                     */
    2 pad1 (4) fixed bin (35);


dcl 1 based_sentinel aligned based,                       /* format of pxss-style sentinel */
    2 fp bit (18) unal,
    2 bp bit (18) unal,
    2 sentinel bit (36) aligned;

dcl VCPU_RESPONSE_BOUNDS fixed bin init (3) int static options (constant);

/* END INCLUDE FILE tcm.incl.pl1 */
```

```
/* BEGIN INCLUDE FILE ... vol_map.incl.pl1 */

dcl   vol_mapp      ptr;

dcl 1 vol_map based (vol_mapp) aligned,

    2 n_rec fixed bin(17),                       /* number of records represented in the map */
    2 base_add fixed bin(17),                    /* record number for first bit in bit map */
    2 n_free_rec fixed bin(17),                  /* number of free records */
    2 bit_map_n_words fixed bin(17),             /* number of words of the bit map */
    2 pad (60) bit(36),                          /* pad to 64 words */
    2 bit_map (3*1024 - 64) bit(36) ;            /* bit map - the entire vol map occupies 3 records */

/* END INCLUDE ... vol_map */
```

```
/*  START OF:       vtoc_buffer.incl.pl1   November 1982      * * * * * * * * * * * * * * * * */

     dcl      vtoc_buffer_seg$        ext;

     dcl      vtoc_buffer_segp        ptr;
     dcl      vtoc_buf_descp          ptr;
     dcl      vtoc_bufp               ptr;
     dcl      vtoc_buf_desc_arrayp    ptr;
     dcl      vtoc_buf_arrayp         ptr;

     dcl      vtoc_buf_n_buffers      fixed bin;
     dcl      vtoc_buf_n_buckets      fixed bin;


     dcl      1 vtoc_buffer           aligned based (vtoc_buffer_segp),

                2 lock,                                 /* Global lock for VTOC buffers */
                  3 processid         bit (36) aligned,     /* Owner */
                  3 wait_event        bit (36) aligned,     /* For lock */
                  3 notify_sw         bit (1) aligned,      /* ON => notify on unlock */

                2 n_bufs              fixed bin,            /* Number of full VTOCE buffers */
                2 n_hash_buckets      fixed bin,            /* Number of hash table buckets */
                2 hash_mask           bit (36) aligned,     /* Mask for hash algorithm */
                2 abs_addr            fixed bin (24),       /* Absolute address of vtoc_buffer_seg */
                2 wait_event_constant fixed bin (36) uns unal,   /* Constant to add to part index to form wait event */
                2 buf_desc_offset     bit (18),             /* Offset of buf_desc */
                2 buf_offset          bit (18),             /* Offset of buf */
                2 hash_table_offset   bit (18),             /* Offset of hash_table */
                2 search_index        fixed bin,            /* Roving pointer for buffer selection */
                2 unsafe_pvtx         fixed bin,            /* PVTE index with update in progress */
                2 scavenger_free_p_clock
                                      fixed bin (35),       /* Pseudo-Clock for scavenger-free-other-allocate race */
                2 meters,
                  3 call_get          fixed bin (35),       /* Calls to get_vtoce */
                  3 call_put          fixed bin (35),       /* Calls to put_vtoce */
                  3 call_alloc        fixed bin (35),       /* Calls to alloc_and_put_vtoce */
                  3 call_free         fixed bin (35),       /* Calls to free_vtoce */
                  3 call_await        fixed bin (35),       /* Calls to await_vtoce */
                  3 steps             fixed bin (35),       /* Steps through buffer allocation */
                  3 skip_os           fixed bin (35),       /* Skipped because out-of-service */
                  3 skip_hot          fixed bin (35),       /* Skipped because buffer hot */
                  3 skip_wait         fixed bin (35),       /* Skipped because notify_sw set */
                  3 disk_reads        fixed bin (35),       /* Number of same */
                  3 disk_writes       fixed bin (35),       /* Number of same */
                  3 get_buffer_calls  fixed bin (35),       /* Number of calls to GET_BUFFER */
                  3 get_buffer_hits   fixed bin (35),       /* Number times VTOCE in buffer */
                  3 wait_calls        fixed bin (35),       /* Number of calls to WAIT */
                  3 wait_os           fixed bin (35),       /* Number of times had to wait */
```

```
            3 scavenger_free_checks
                                  fixed bin (35),         /* Number of times had to check pseudo-clock */
            3 scavenger_free_losses
                                  fixed bin (35),         /* Number of times race lost between scavenger freeing and other allocat
e */
            3 pad (15)            fixed bin (35),

          2 hash_table            (vtoc_buf_n_buckets refer (vtoc_buffer.n_hash_buckets)) bit (18) aligned,


          2 buf_desc              (vtoc_buf_n_buffers refer (vtoc_buffer.n_bufs)) aligned like vtoc_buf_desc,

          2 buffer                (vtoc_buf_n_buffers refer (vtoc_buffer.n_bufs)) aligned like vtoce_buffer;



dcl     1 vtoc_buf_desc_array   (vtoc_buffer.n_bufs) aligned based (vtoc_buf_desc_arrayp) like vtoc_buf_desc;


dcl     1 vtoc_buf_desc         aligned based (vtoc_buf_descp),
          2 pvtx                fixed bin (17) unal,    /* PVTE index */
          2 vtocx               fixed bin (17) unal,    /* VTOCE Index */
          2 parts_used          bit (3) unal,           /* Mask of parts used or os */
          2 err                 bit (1) unal,           /* ON => I/O error on buffer */
          2 notify_sw           bit (1) unal,           /* ON => notify requied on I/O completion */
          2 write_sw            bit (1) unal,           /* ON => write I/O */
          2 os                  bit (1) unal,           /* ON => I/O in progress */
          2 ioq                 bit (1) unal,           /* ON => I/O has been requested */
          2 used                bit (1) unal,           /* ON => this descriptor is in use */
          2 pad                 bit (9) unal,
          2 wait_index          fixed bin (17) unal,    /* Buffer index for forming wait event */
          2 ht_thread           bit (18) unal,          /* Offset of next entry in hash table */
          2 buf_rel             bit (18) unal;          /* Offset of buffer in segment */

dcl     1 vtoce_buffer_array    (vtoc_buffer.n_bufs) aligned based (vtoc_buf_arrayp) like vtoce_buffer;

dcl     1 vtoce_buffer          aligned based (vtoc_bufp),
          2 parts               (3) aligned,
            3 words             (64) bit (36) aligned;


dcl     N_PARTS_PER_VTOCE       fixed bin int static options (constant) init (3);
dcl     VTOCE_PART_SIZE         fixed bin int static options (constant) init (64);
dcl     VTOCE_BUFFER_SIZE       fixed bin int static options (constant) init (3 * 64);
dcl     N_VTOCE_PER_RECORD      fixed bin int static options (constant) init (5);
dcl     N_SECTOR_PER_VTOCE      fixed bin int static options (constant) init (3);

/*  END OF:         vtoc_buffer.incl.pl1                    *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  */
```

```
/* BEGIN INCLUDE FILE ... vtoc_header.incl.pl1 */

dcl  vtoc_headerp ptr;

dcl 1 vtoc_header based (vtoc_headerp) aligned,

    2 version fixed bin (17),                      /* version number. The current version number is 1. * */
    2 n_vtoce fixed bin (17),                      /* number of vtoc entries */
    2 vtoc_last_recno fixed bin (17),              /* record number of the last record of the vtoc */
    2 n_free_vtoce fixed bin (17),                 /* number of free vtoc entries */
    2 first_free_vtocx fixed bin (17),             /* index of the first vtoce in the free list */
    2 pad (3) bit (36),
    2 dmpr_bit_map (2048 - 8) bit (36);            /* space for dmpr bit map */

/* END INCLUDE ... vtoc_header */
```

```
/*  START OF:       vtoc_map.incl.pl1    ... March 1982 ...    * * * * * * * * * * * * * * * * * */

     dcl     vtoc_mapp              ptr;
     dcl     bit_map_wordp          ptr;

     dcl     1 vtoc_map             aligned based (vtoc_mapp),
               2 n_vtoce            fixed bin,             /* Number of VTOCEs on the device */
               2 n_free_vtoce       fixed bin,             /* Number of free VTOCEs */
               2 bit_map_n_words    fixed bin,             /* Number of words in the bit map below */
               2 vtoc_last_recno    fixed bin,             /* Last record number in VTOC */
               2 pad                (4) fixed bin,
               2 bit_map            (0:1024 - 9) bit (36); /* This structure consumes exactly 1 page */

     dcl     1 bit_map_word         aligned based (bit_map_wordp),
               2 pad1               bit (1) unal,
               2 bits               bit (32) unal,         /* 32 VTOCES ON => free */
               2 pad2               bit (3) unal;


/*  END OF:         vtoc_map.incl.pl1                     * * * * * * * * * * * * * * * * * */
```

```
/*        BEGIN INCLUDE FILE ...vtoce.incl.pl1 ... last modified September 1982 */

/* Template for a VTOC entry. Length = 192 words. (3 * 64). */

dcl  vtocep ptr;

dcl 1 vtoce based (vtocep) aligned,


    (2 pad_free_vtoce_chain bit (36),                  /* Used to be pointer to next free VTOCE */

     2 uid bit (36),                                   /* segment's uid - zero if vtoce is free */

     2 msl bit (9),                                    /* maximum segment length in 1024 word units */
     2 csl bit (9),                                    /* current segment length - in 1024 word units */
     2 records bit (9),                                /* number of records used by the seg in second storage */
     2 pad2 bit (9),

     2 dtu bit (36),                                   /* date and time segment was last used */

     2 dtm bit (36),                                   /* date and time segment was last modified */

     2 nqsw bit (1),                                   /* no quota switch - no checking for pages of this seg */
     2 deciduous bit (1),                              /* true if hc_sdw */
     2 nid bit (1),                                    /* no incremental dump switch */
     2 dnzp bit (1),                                   /* Dont null zero pages */
     2 gtpd bit (1),                                   /* Global transparent paging device */
     2 per_process bit (1),                            /* Per process segment (deleted every bootload) */
     2 damaged bit (1),                                /* TRUE if contents damaged */
     2 fm_damaged bit (1),                             /* TRUE if filemap checksum bad */
     2 fm_checksum_valid bit (1),                      /* TRUE if the checksum has been computed */
     2 synchronized bit (1),                           /* TRUE if this is a data management synchronized segment */
     2 pad3 bit (8),
     2 dirsw bit (1),                                  /* directory switch */
     2 master_dir bit (1),                             /* master directory - a root for the logical volume */
     2 pad4 bit (16)) unaligned,                       /* not used */

     2 fm_checksum bit (36) aligned,                   /* Checksum of used portion of file map */

    (2 quota (0:1) fixed bin (18) unsigned,            /* sec storage quota - (0) for non dir pages */

     2 used (0:1) fixed bin (18) unsigned,             /* sec storage used  - (0) for non dir pages */

     2 received (0:1) fixed bin (18) unsigned,         /* total amount of storage this dir has received */

     2 trp (0:1) fixed bin (71),                       /* time record product - (0) for non dir pages */

     2 trp_time (0:1) bit (36),                        /* time time_record_product was last calculated */
```

```
    2 fm (0:255) bit (18),                          /* file map - 256 entries - 18 bits per entry */

    2 pad6 (10) bit (36),                           /* not used */

    2 ncd bit (1),                                  /* no complete dump switch */
    2 pad7 bit (17),
    2 pad8 bit (18),

    2 dtd bit (36),                                 /* date-time-dumped */

    2 volid (3) bit (36),                           /* volume ids of last incremental, consolidated, and complete dumps */

    2 master_dir_uid bit (36),                      /* superior master directory uid */



    2 uid_path (0:15) bit (36),                     /* uid pathname of all parents starting after the root */

    2 primary_name char (32),                       /* primary name of the segment */

    2 time_created bit (36),                        /* time the segment was created */

    2 par_pvid bit (36),                            /* physical volume id of the parent */

    2 par_vtocx fixed bin (17),                     /* vtoc entry index of the parent */
    2 branch_rp bit (18)) unaligned,                /* rel pointer of the branch of this segment */

    2 cn_salv_time bit (36),                        /* time branch - vtoce connection checked */

    2 access_class bit (72),                        /* access class in branch */
    2 perm_flags aligned,
      3 per_bootload bit (1) unal,                  /* ON => deleted each bootload */
      3 pad9 bit (35) unal,
    2 owner bit (36);                               /* pvid of this volume */

dcl  vtoce_parts (3) bit (36 * 64) aligned based (vtocep);

dcl 1 seg_vtoce based (vtocep) aligned,             /* Overlay for vtoce of segments, which don't have quota */
    2 pad1 bit (7*36),
    2 usage fixed bin (35),                         /* page fault count: overlays quota */
    2 pad2 bit (184*36);

/*        END INCLUDE FILE vtoce.incl.pl1 */
```

APPENDIX B

MULTICS TECHNICAL PAPERS

This page has intentionally
been left blank.

```
0556.00 S     .   +|    .    |    .    |    .    |    .    |    .    |    .    |    .    |    .    |      .   *V
0601.00 QS    .    +    .    |    .    |    .    |    .    |    .    |    .    |    .  **|*********|*****t y V
0606.00 Q     S   +|    .    |    .    |    .    |  . ***|*********|*********|*********|*p y.     |      .  DV
0611.00 QS    .   +|    .    |    .    |    .    |    .    |    .    |    .    |    .    |    .    |    . ***V
0616.00 QS    .   +|    .    |    .    |    .    |    .    |    .    |    .    |    .    |    .    |    *****V
0621.00 QS    .   +|    .    |    .    |    .    |    .    |    .    |    .    |    .    |    .    |    . ***V
0626.00 QS    .   +|    .    |    .    |    .    |    .    |    .    |    .    |    .    |    .    |  ***** V
0631.00 S     .   +|    .    |    .    |    .    |    .    |    .    |    .    |    .    |    .    |    . ***V
0636.00 Q     .    |  S .    +    .    |    .    |    .    |    .    |*******|******tss|  . y    |    .    V
0641.00 Qe    .    |    .   **-*+******|*********|****S****|*********|*********|********iitps .  |y   .  DVP
0646.00 Q     .    |    .    |- + .    |    S    |    *****|*********|*********|*********|********its   y  VP
0651.00 Q     .    |  S .    |  - +.   |    .    |    .    |    .    |    .    |    .    |********|*****t y V
0656.00 Q     .    |S   .    |  - +.   |    .    |    .    |    .    |    .    |    .    |    .  ********syV
0701.00 Q     .    |   S .   |  - +.   |    .    |    .    |    .    |    .    |    .    |    .   |******ty V
0706.00 Q     .   S     .    |    -.+  |    .    |    .    |    .    |    .    |    .    |*******|****i y  V
0711.00 Q     .  S |    .    |    -.+  |    .    |    .    |    .    |    .    |    .    |   .  **|******iy V
0716.00 Q     .   S     .    |    -.+  |    .    |    .    |    .    |    .    |    .    |   .****|******ty V
0721.00 Q     .    |    . S  |    -.+  |    .    |    .    |    .    |    .    |    .    |*******|******py V
0726.00 Qe    .    |    .S   |     - + |    .    |    .    |    .    |    .    |*******|*********|**t . y  V
0731.00 Q     .    |    . S  |    -.+  |    .    |    .    |    .    |    .    |*********|*********|***t. y  V
0736.00 Qe    .    |    .    |     - + |    .    |    .    |    .    |*********|*********|******its|y   .   DV
0741.00 Q     .    |    .    |    .- + |S   .    |    .    |    .  **|*********|*********|*****its |y   .   DV
0746.00 Qe    .    |    .    |    .   -|+   .    |S   .  **|*********|*********|*********|***iips   | y  .   DV
0751.00 Qe    .    |    .    |    .    |- + .    |    *****|*S*******|*********|*********|****iitps |  y .  D V
0756.00 Qe    .    |    .    |    .   **-*+*****|*********|*********|*********|****iiiitpss  . y |    .  D V
0801.01 Q  e  .    |    .   *|**********-*+******|*********|*********|*****miiiitttppss  |  y.   |   .D   V
0806.01 Q**e******|*********|*********|**-*+****|*********|*********iiiitStppppssssss  |  y .   |   D.   VP
0811.01 Q***e*****|*********|*********|*-*+*****|***iiiiiittppppssss|sS  .   y|    .    |    .    |  D .   VP
0816.01 *Q***eq***|*********|*********|******-i+iiitttppppssssssss S|   y     |    .    |    .    | D  .   VP
0821.01 rQ***eq***|*********|*********|*****-*+*|*miiiiittppppsssSss|sss .   |y   .    |    .    |  D .   VP
0826.01 *Q**e*****|**********miiiitttp|pssss-s+ss   .y   |   .  S  |    .    |    .    |    .    |  D .   VP
0831.01 Qr****e***|*********|*********|*mmmmiii-i+itttppppssssssSss|   . y   |    .    |    .   D     .   VP
0836.01 Q**e******|*********|*********|*********-*+*******|*******Smiiiiitttpp|psssss   |y   .    | D   .   VP
0841.01 Q****e****|*********|*********|*********|*-*+******mmmiiiiit|tppppppss|ss  .  y |    S   | D   .   VP
0846.01 r*Q*e*****|*********|*********|*********|***-*+**miiiiitttppSpppsssss    y     |    .    | D   .   VP
0851.11 rQ****e***|*********|********m|miiiiiiiiittt-t+pppp|pppppppss|s   .  y |    .    |    .   D    .   V
0856.13 Qr*******eq*********|*********|*****mmmmmiiii-i+tt|tpppppppp|pppssssss|   . y   |    .   D    .   VP
0901.09 Q**r****e**qmmmmmmmm|mmmmmiiii|iiitttttpp|ppppp-p+s|sssss    |y  .     S  .    | . D  |    .   VP
0906.05 Q***mmrmmemmmmmqmmii|iiiiiitttpppppppppp|ppp-p+pss|sss .    |y  . S   |    .    |  . D |    .   VP
0911.02 *Q*r***e**|**mmmmmmm|mmmmmmiii|iiiitttttp|ppp-p+ppp|ppssssss |   .y   |    .    | S . D |    .   VP
0916.00 Q*****eq**|*********|*********|**********mmmm-i+iiittttppppp|ppppssssss   .y    |    .   D    .   VP
0921.02 Qr*****e*q*********|*********|********m|mmii-i+it|tpSppppss|ssss.   | y  .    |    .   D    .   VP
0926.01 Q***e*****|*********|*********|*********|**-*+****miiiiiittp|pppssssssS   .y    |    .   |D   .   VP
0931.01 Qr*****e*q*********|*********|*********|**-*+***m|mmiiiiitttppppppssss|s   .   S|    .    |  D  .   VP
0936.01 Q*e*******|*********|*********|*********-*+*******|*********|*miiiiStp|ppssssss  |  y.     |   D.   VP
0941.01 Q**e******|*********|*********|*********-*+*******|*******ii|iiitttpppssss .   y   .    |   D.   V
0946.12 rQ***eq***|*********|*********|*******-*+*********mmiiiiitttppppsssS  |   . y   |    .    |  D .  VP
0951.01 Q**eq*****|*********|*********|*********-*+*******|****iiiiiittppppssss   S y   |    .    |  D .  VP
0956.01 rQ**e*****|*********|*********|*********|*-i+iiiit|tppppssss|   S y   |    .    |    .    | D  .   VP
```

```
SNAPSHOT 09/17/86 08:59


Multics MR10.2, load 45.0/60.0; 47 users, 24 interactive, 22 daemons.
Absentee users 1/4
avq = 11, elapsed time = 930 sec, 30 active last 15 sec.
Virtual CPU Time 29.02  41.46
Zero Idle          0.00
NMP Idle          30.02
  1   1487   4.87       0.00  0.50    2263  0.00   0.14 70.09   VADIS
All   1994   8.76       0.00  0.50    2828  0.01   0.39 27.70
 0               4.     0.35  3    0.26  2.10     0.26  2.10 P  0  R  I  Init
 1              16.     0.09  1    1.00  0.50     1.00  1.00 P  0  R  I  VADIS
 2   10.         0.     0.95  1                                P  0     I  System Sy
 3   21.        26.     0.80  1                                P  0     I  Other
 4              11.     0.02  1    0.13  1.00     0.50  1.00 P  0  R  I  IO
xLEID(b) 460       0       0     0       0       0    0.008     0 0    0 -1 Idle
rLEI(a)  341       0       0     0       0       0    0.000     0 0    0 -1 Idle
wLE      156   13765    1000     8       0       0    0.020 34465 0    0  4 Retriev
b        117   11693    2000     0    5267    6000   48.105     0 0    0  3 Alonso
wwLE     114   13513    2000   118    2003    6000    0.084 273211 0   0  3 Cintro
wLE       70    4502    1000     3       0    6000    0.189 -400000000000 0  0  1
b         70     817    2000     0      49       0  162.744     0 0    0  3 Figuero
b         47    1356    2097     0      78       0    2.271     0 0    0  0 Rios
b         39    2223    1000     0      87       0    0.161     0 0    0  1 IslaGra
b         34    2158    1000     0     161       0    1.442     0 0    0  1 Caguas
Page Faults       11.74  16.78     2717.319
Laps              124         7.479 sec.
allocations             33127         0        0.00       0.000
allocations             68230      9515       13.95      11.462
 9           261        10500        47         3553          88         86
 1          7005         3081       167          132         300         91
 7          5000         5091       128          185         182         91
15          4736         4294       158          195         215        102
15          5296         3526       202          175         263        105
 6          4787         3641       131          193         254        110
 5          4099         3881       161          226         238        116
 2          6264          592       200          148        1566        135
-- -- -- -- -- --
10          2714          182       307          341        5095        320
14          2619          209       291          354        4437        327
 5           617          327        73         1502        2835        982
11           255          158       150         3636        5869       2245
12            27            0        19        34346           0      34346
Seg Faults         5.11   7.30    18332.578
Lap Time(sec)      191.9     706.6     1973.2       84.0
Other Fault       10.29  14.71
 Getwork           4.53   6.48      498.232
MP Idle            5.92   8.46
```

```
clok  4  ast
root  dska  1  dskb  2
schd  400000  10  20  60  2  10
tbls  sstn  32.  str  64.
tcd  75.  299.
parm  ttyb  14080.  ttyq  1536.
part  bos  dska  1
part  dump  dska  1
part  log  dskb  2
mpc  mspa  451.  a  20  4  a  24  3
mpc  mspb  451.  a  30  4  a  34  2
mpc  mtpa  601.  a  16  1  a  17  1
mpc  urpa  600.  a  10  4
fnp  a  a  14
fnp  b  a  15
cpu  a  7  on  dps8  70.  8.
cpu  b  6  on  dps8  70.  8.
chnl  dska  a  34  2
chnl  dskb  a  30  4
sst  600.  400.  300.  750.
salv  dcf
prph  dska  a  20  4  451.  20
prph  dskb  a  24  4  451.  20
prph  tapa  a  16  2  500.  10
prph  prta  a  10  1600.  600.  136.
prph  prtb  a  11  1600.  600.  136.
prph  rdra  a  12  301.
prph  opca  a  37  6601.  80.  on
prph  puna  a  13  300.
intk  warm  4
iom  a  0  nsa  on
mem  d  512.  on
mem  c  512.  on
mem  b  512.  on
mem  a  2048.  on
fnp  c  a  36
```

Drive	Records	Left	%	VTOCEs	Left	%	Avg Size	PV Name	PB/PD	LV Name
dska_05	36584	1354	4	7625	4605	60	11	cis2	pb	cis
dska_11	36584	2455	7	7625	4814	63	12	cis1	pb	cis
dska_12	36584	6226	17	7625	4154	54	8	ldd1	pb	ldd
dskb_09	36720	27412	75	7650	7592	99	160	progra	pb	progra
dskb_05	36720	2521	7	7650	1986	26	6	public01	pb pd	public
dskb_06	36720	1954	5	7650	1717	22	5	public03	pb pd	public
dskb_07	36720	2373	6	7650	1791	23	5	public02	pb pd	public
dska_01	29811	1121	4	17400	12774	73	6	rpv	pb	root
dskb_02	35837	1968	5	7575	2971	39	7	root2	pb	root
dskb_03	36584	2805	8	7625	2987	39	7	root3	pb	root
dska_15	36584	3604	10	7625	5209	68	13	vadis_misc1	pb pd	vadis_mis
dskb_15	36584	5178	14	7625	5478	72	14	vadis_misc2	pb pd	vadis_mis
dska_04	37967	182	0	710	562	79	255	vadis_pf7	pb	vadis_pf
dska_06	37967	168	0	710	560	79	251	vadis_pf5	pb	vadis_pf
dska_10	37967	433	1	710	563	79	255	vadis_pf3	pb	vadis_pf
dska_13	37967	398	1	710	561	79	252	vadis_pf2	pb	vadis_pf
dskb_08	38094	299	1	710	561	79	253	vadis_pf1	pb	vadis_pf
dskb_14	37967	60	0	710	561	79	254	vadis_pf6	pb	vadis_pf
dskb_16	37967	310	1	710	561	79	252	vadis_pf4	pb	vadis_pf

Current system tuning parameters:

tefirst	1. seconds
telast	2. seconds
timax	6. seconds
priority_sched_inc	80. seconds
min_eligible	2.
max_eligible	8.
max_batch_elig	0
working_set_factor	0.5
working_set_addend	0
deadline_mode	off
int_q_enabled	on
post_purge	off
pre_empt_sample_time	0.04 seconds
gp_at_notify	off
gp_at_ptlnotify	off
process_initial_quantum	2. seconds
quit_priority	0.
notify_timeout_interval	30. seconds
notify_timeout_severity	3
write_limit	100
gv_integration	8. seconds
realtime_io_priority	off
realtime_io_deadline	0. seconds
realtime_io_quantum	0.005 seconds

CTP → don't change it in config, put it in ec.

! answer yes -bf CTP Telast .2 -silent
↳ suppresses msgs on oper console for init.

notify timeout severity,

proc_id.

look in Tc_data

003000 777777 initializer
3100 555555 idle process

Event that did not occur → hardcore_events.gi.info

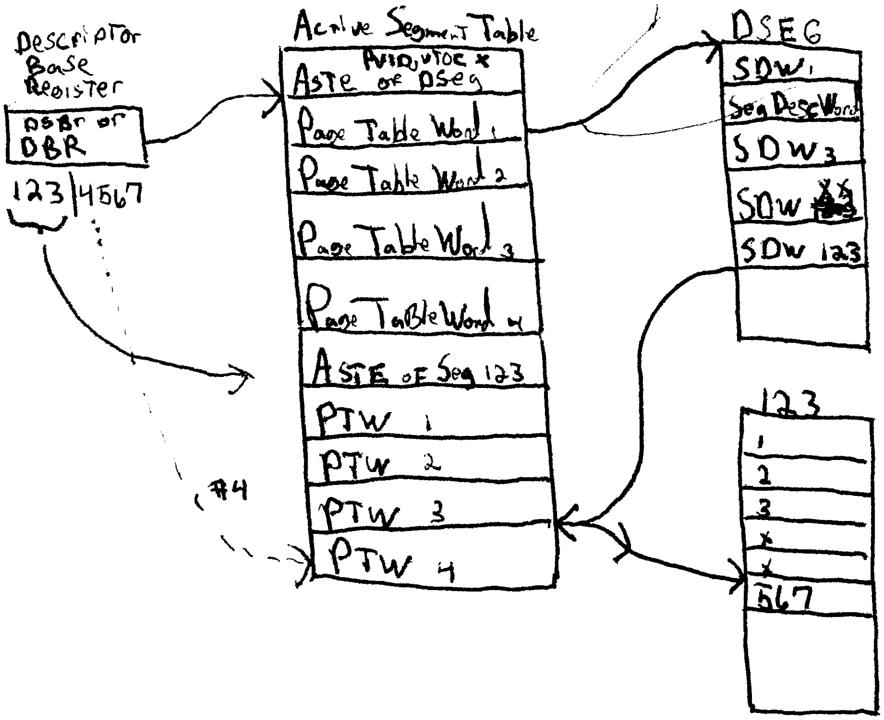
If low overhead 5% then adding CPU helps

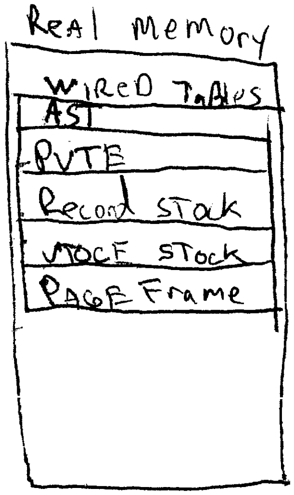
look at avg CPU per interaction + buckets in meters

SALVAGERS → checks damage to vtoce + turns on damage switch.
Directory salvaging → online salvager maybe do one later
Quta " → can be done right away.
Scavenger → if ~~uncos~~ inconsistency switch is on - online scavenge. Can be delayed. Does 1 volume at a time. 15 min per volume.

Physical Volume Salvager - Offline salvaging - BOOT RLVS or RPVS
Basicly same as scavenge.

sweep_pv - connection failures + reverse connection failures.
runs once a month.





DSBT is Register in CPU
Contains Current Process ID
To initiate Segment 123

The DBR points to ASTE of DSeg. AST is always in memory. There is an ASTE + Page Table in the AST for each ~~segment~~ active segment. There is also a Segment Descriptor word (SDW) inside the DSeg for each segment known to the process. The PageTable of Dseg points to the base of DSeg. Using hash tables the SDW of the segment number is searched for. If found the ~~first~~ offset which is in the SDW is used to point to the offset of the ASTE for that segment. If not found the info is put into the AST, the PTW pointed to by the first # of the offset. The PTW ~~to~~ uses the rest of the offset to find location inside the segment.

DBR info comes from APT
when process is eligible, Page of DSeg is temp wired.
DSEG is made active at Proc creation Semi hold switch set on for process life
SDW allocated making segment known
SDW filled in at Seg fault time
ASTE ~~#~~ for Seg is filled in from VTOCE at segment activation.
PTW gets filled in at Page fault time

Seg Active

ASTE entries filled in By VTOCE

APTE'S chained Together.

As you come in, you are at end of list ie: low Priority

If we scan & don't find ready Process, we use idle Process which are always at end of elgible Queue

notifying a Process just makes it ready, doesn't make it run immediately

when a Processed is elgible it looks for first 2 Pages of DSeg + PDS, if They are in memory They are Marked wired. If ~~w~~ They are not, Then a Disk read is Done. when They are in memory Then marked wired.

No Stack_0 in Proc_dir for Ring 0. You can't Become inelgible while in Ring 0. while in Ring 4 nothing is in any othe ring. This means ring 0 stacks Can Be shared. from a Pool. This Prevents Page faults for Ring 0 stack.

(MAXE) MAX elgible Paramater, related To # Ring 0 STacks.

The working set of System (all Procs) Can't exceed main memory. IT will cause Page Thrashing.

2 Mw memory

$$\frac{2MW}{1\text{ to } 200} \quad 2 ~~0~~ \text{ or } 12 \quad 10$$

$$\frac{2000 - 500}{150 \text{ or } 200} = 8 \text{ or } 10$$

~ 5 ~ 6 MAXE Procs per MW

NORM

8 mw X 5 = ME of 40

MAXE, Balance of CPU usage vs Available memory

MAX-MAXE

# of Ring 0 Stacks To Create. CTP won't let you change MAXE > This setting

Page Trace

losing elgibility.

runs Post_Purge which uses PDS To find # Page Faults

To find The working set ie: # of Pages Touched during elgibility.

This Breaks down Because of ~~Page eviction done while~~ Pages remain in memory

~~waiting~~ while you go To inelgible.

By default Post Purge is OFF.

Useless PARAMS

MIN ~~Mid~~_elgible

Post_Purge

MINE ?? Used To Be = # CPU's

WSF = GWS + (ws * wsF + Twsa)
SFactor

WSA
at end

MPidle - idle time From idle Process,
AT MAXE & had elgible Processes
which means low MAXE.

Ques

First Time in you go Through IQ (interactive Q)

(Interactive + SysDaemons
Ti max normally = 8 sec used up all of There Ti.

The more Times you cycle in a Work class, The lower you sink

abs TIMAX normally = 16 lets interactive favorable over ABSentee.

Do NOT PUT ABSentee's in ~~some~~ seperate work class By Themselves,

They will All Accumulate + get More Time. IT is better

To change abs Ti max. Real Time: mainly used for Printer Daemons

Percentages of work classes

Real Time. doesn't use Percentages, IT is higher Priority.

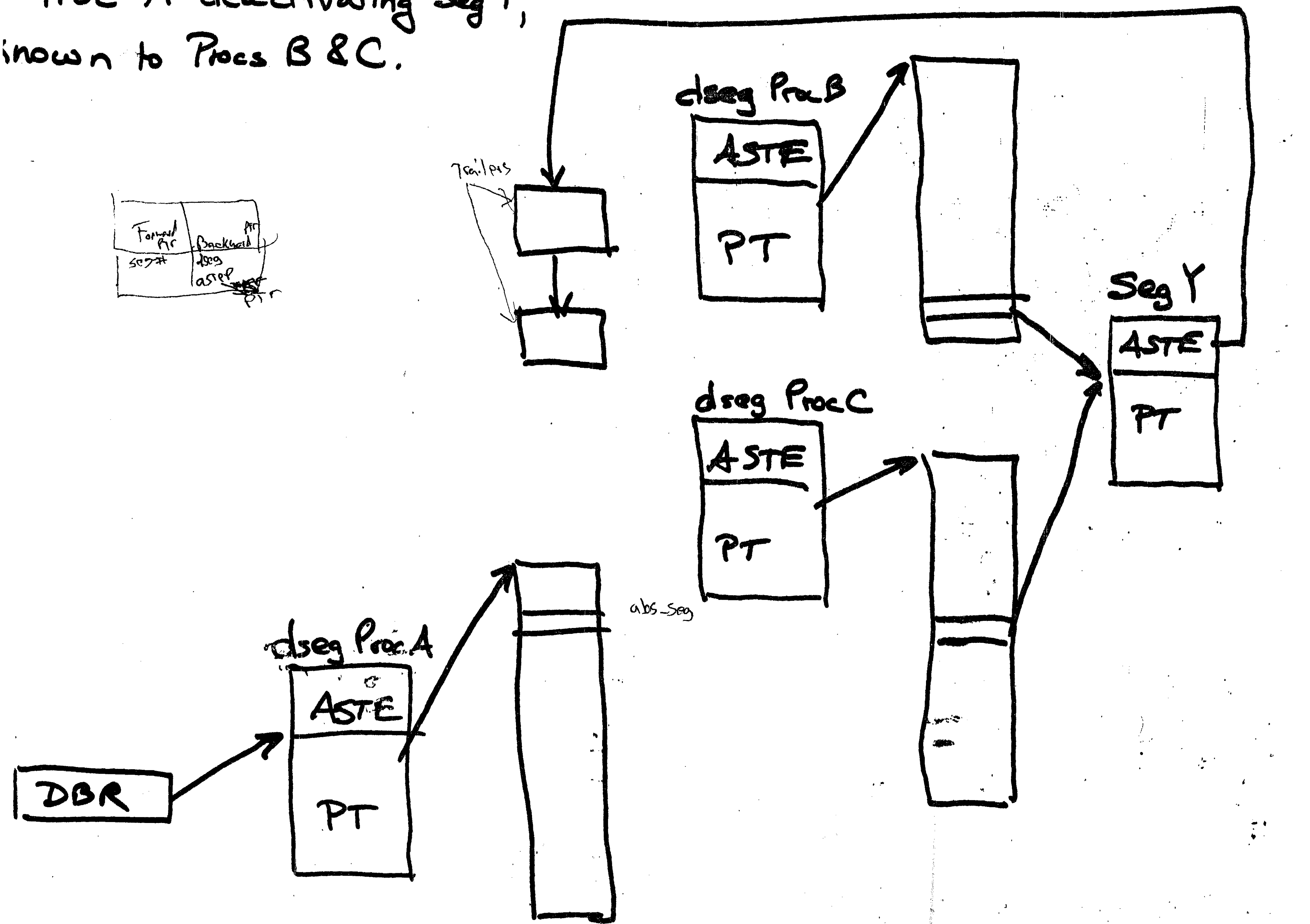
Process ~~is made~~ follows Ir → IQ

R Q

deadline = clock() + Ir. once deadline occurs, They have higher priority.

also They start AT TOP OF Que Instead of Bottom.

Proc A deactivating Seg Y,
known to Procs B & C.



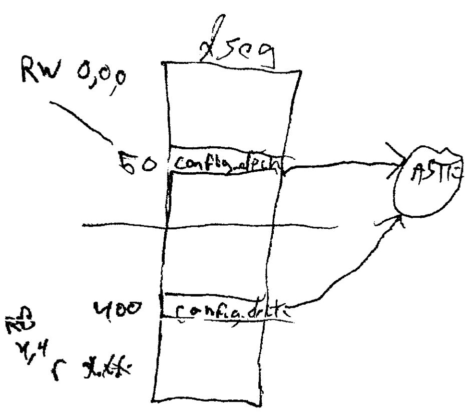
abs_seg supervisor seg used to plug in whatever SDW we need.
Address of ASTE from trailer is plugged in abs_seg. 2nd proc ~~which~~ uses offset to point to dseg.

deciduous - ref by seg#, supervisor segments dir entry is fabricated even though it is not in hiarchy. Fabrication is in >SL1>config_deck

hack to avoid gates.

other examples

rb 0,0,5 >sl1>hcs_
re *.*.*



Things in PD created by init are reverse deciduous. Because deciduous are created as supervisor made to look like hierarchy, reverse are hierarchy made to look like supervisor

Small MAKE → hi pi
mine small or large → no effect

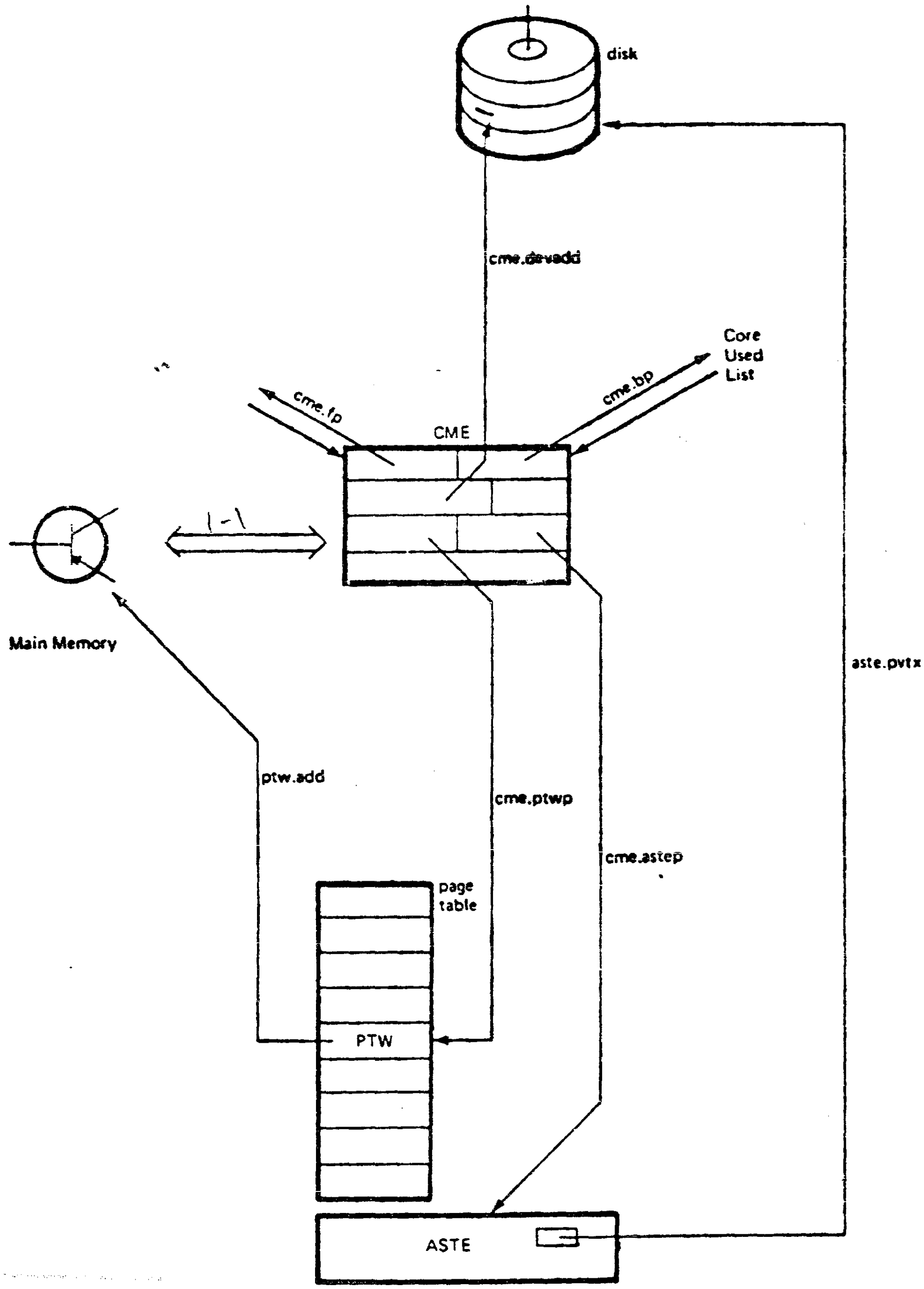


Figure 6-2. Page Control Data Bases
Page in main memory, not on paging device

1-1 relation Between CME'S
& Page Frames

Work Classes

~~limited To~~

MAX % Per WClass, There is A Bit So That They Do not go Through Interactive Queu.

load control Groups → MGT

decides who can login when system AT MAXU.

Project get assigned To LCG's which get assigned WClass.

PTL - Global Page Table lock

when unlocked, only 1 Process is notified instead of All ~~of~~ Processes waiting

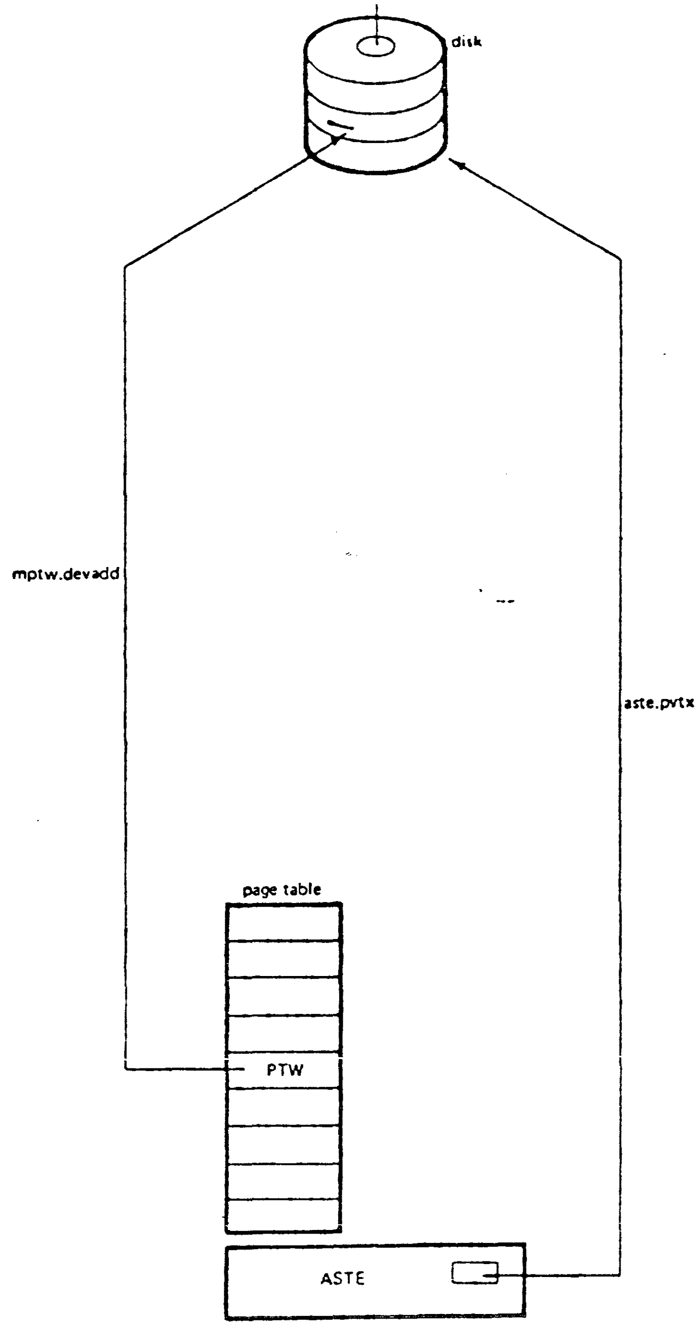
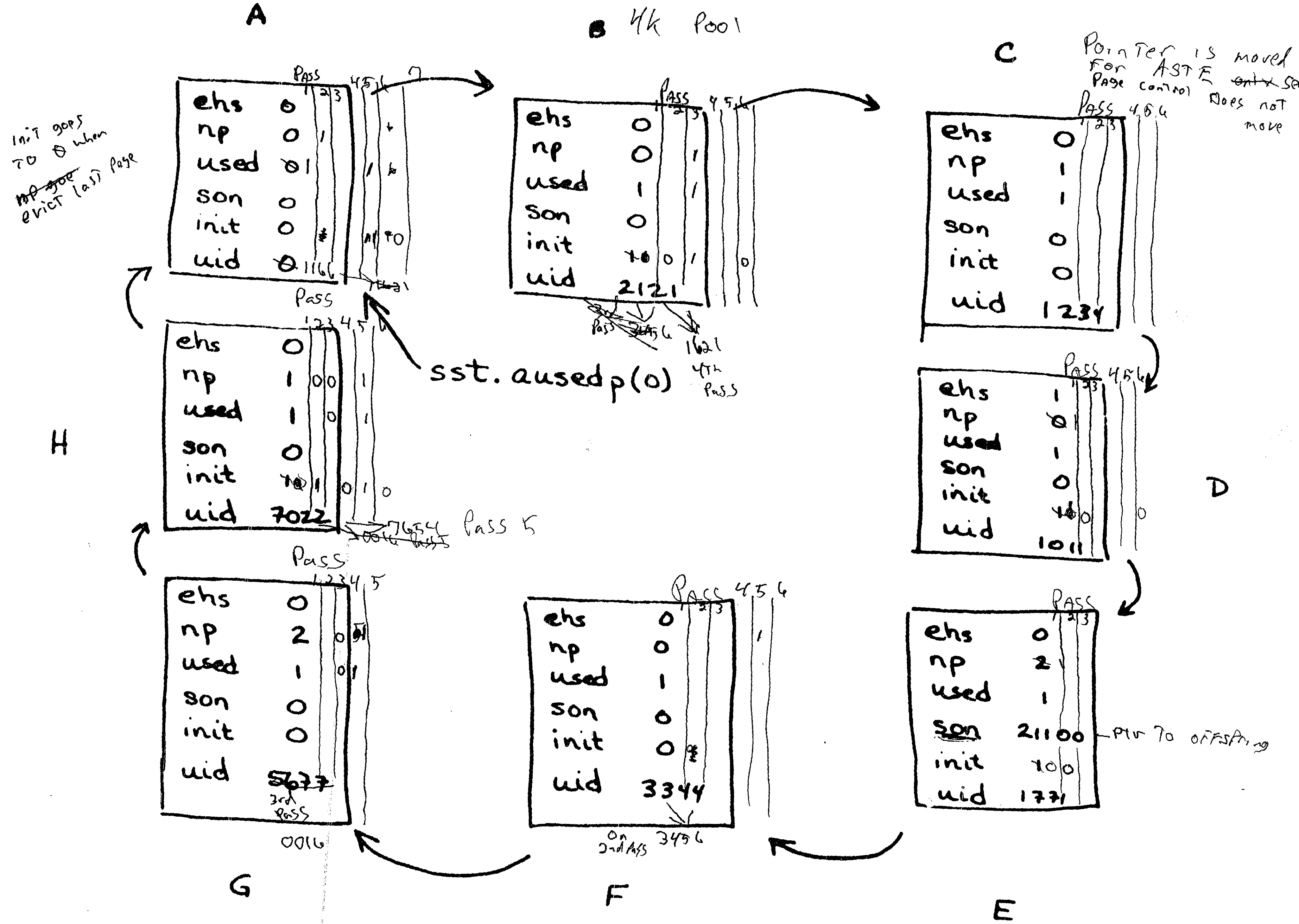


Figure 6-1. Page Control Data Bases
Page not in main memory or on paging device


A
B 4k Pool
C
D
E
F
G
H
Pointer is moved for ASTE
Page control Does not move
init goes to 0 when evict last page
sst. ausedp(0)
ehs
np
used
son
init
uid
2121
1234
1011
1771
3344
7022
Pass 5
4th Pass
3rd Pass
On 2nd Pass
ptr to offspring

1. Activate seg, UID 1166
2. Read Page 0 of 1011
   (should this be 1 for 2nd page 0 being first?
3. Evict Page 2 of 1771
4. Read Page 0 of 1166
5. Activate 3456
6. Evict Page 1 of 7022
7. Read Page 5 (!) of 5677 Bounds fault goes to 16K pool. Also gets rethreaded so it is after pointer so it is next one seen.
8. Activate 0016
9. Read Page 1 of 0016
10. Activate 1621
11. Read Page 0 of 3456
12. Evict Page 0 of 1166
13. Activate 7654

COMMON MULTICS PERFORMANCE PROBLEMS

System Tasks Taking Too Much Resources

MTAR
Too much backup during prime time
Directory salvaging after ESDless crash
System personnel favored by workclass parameters

Parameters Set wrong

Work Class Parameters
Absentees in separate workclass eats up time

Traffic Control Parameters ~~TC first too high~~
telast too high
maxe wrong, usually too low

AST Size

Segment thrashing due to small pool size(s)

Hardware Configuration

Not enough memory
Not enough disk channels
Not enough logical channels
Not enough disk arms
Insufficient CPU power

Disk I/O

Unbalanced disk I/O

Communications

HASP lines
X.25 lines cause interrupts, not much you can do about it.
breakall on dialout lines

Misc

Bootload console looping
Initializer time at maxu users
Application causing segment thrashing
Application using too much CPU time
Bad write/notify cables → makes CPU clear cache memory
Check_CPU will check this