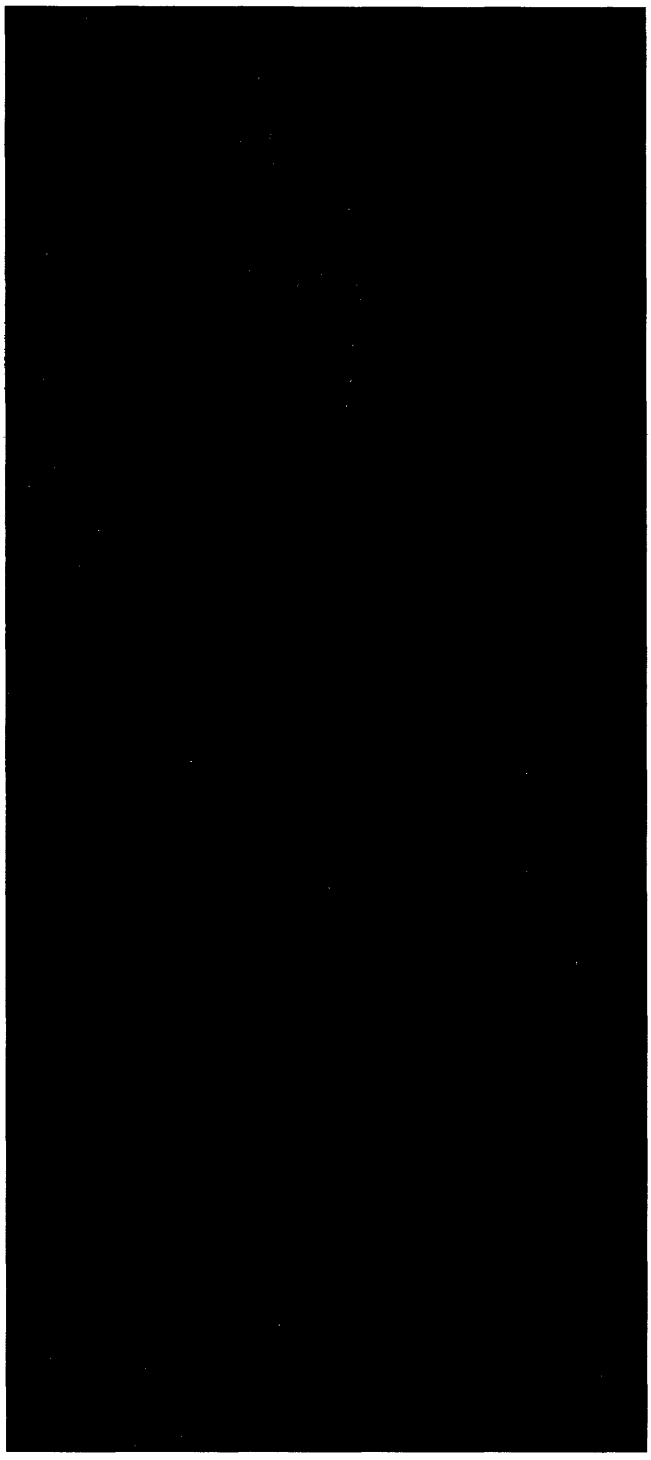
Honeywell

SYSTEM ADMINISTRATOR'S MANUAL

MULTICS

PRELIMINARY EDITION

SOFTWARE



Honeywell

SYSTEM ADMINISTRATOR'S MANUAL

MULTICS

PRELIMINARY EDITION

SUBJECT:

Description of the Capabilities and Functions of the Multics System Administrator.

SPECIAL INSTRUCTIONS:

This manual corresponds to MIT Revision 3, dated January 9, 1973.

DATE:

February 1973

ORDER NUMBER:

AK50, Rev. 0

This manual was written by Thomas H. Van Vleck of the Programming Development Office of the Massachusetts Institute of Technology.

File No.: 1L13

Table of Contents

Table of Contents

Table of Contents

Chapter 1: Introduction

1.1 Purpose

This document is a manual for system administrators on the Multics system. It does not describe the internal structure of the system-administration data bases or the programs which manipulate them: consult the MPM SPS and the MSPM for these details. This manual is intended to be a useful guide for the man who has to set up and run a Multics system.

1.2 Resource Control

The overall job of the Multics system-administration programs is to control the use of system resources and to keep records about how they were used. They must also support rationing of resources, provide system security services, and produce usage reports and bills as required.

Multics resources are used by logged-in users, each of whom has (at least one) process under his control. The user is identified, when he attempts to log in, by his person ID and his project ID. A process is then created for him, and his resource usage is metered by various hard-core modules. These meter readings are collected and kept in the Multics storage system, and are made available to the resource-monitoring and billing programs.

In order to log a user in, some process must execute a program which reads the supplied login request line and password, and checks them agains the list of valid users, also kept in the Multics storage system. To provide maximum protection for this segment, the process in which the login program executes is a fictitious user of the system, called the "answering service." (This process is also called the "initializer process," because it is the process created during the system initialization sequence, and the "system control process.") Almost all resource-usage metering functions are performed in the initializer process. The generic term for the thirty or so programs which deal with system resource control is "user control" or "system control."

The administrative and resource-control functions of Multics comprise a sizable subsystem, which, like the rest of the operating system, is designed to continue to grow, and to allow many optional facilities which a particular installation may decide to bypass.

The system administration facilities are organized into several functional areas, with well-defined interfaces. There are six major areas, each of which contains several subsystems.

1. Hardcore resource multiplexing and metering. The central part of the supervisor, especially the Traffic Control, Page Control, and fault and interrupt service modules, has primary responsibility for providing users with access to the system's resources. These modules also keep extremely fine-grained meters which record the usage of each process.

2. Hardcore interfaces to the user. The Multics Storage System provides Directory Control services which organize data for a user and control access to data, and keep fine-grained meters and quotas on storage usage. The Interprocess Communication facilities allow for synchronization of and cooperation between processes.

3. Other supervisor services. Other tape and data-management services are also provided by the Multics supervisor to allow users access to system resources and to control the manner in which these resources are used.

4. System control. User identification and login control, basic accounting functions, file backup, control of shared devices such as line printers, and operator communication are provided by system control.

5. System Administration. User registration services and billing operations are performed by the system administrators. This manual is primarily a guide to the operation of programs in the system administration area.

6. User project administration. Some groups of users may have the ability to manage some of their own resources. This manual, and the Multics Project Administrator's Manual, describe operations in the user project administration area.

1.3 System administrators

There are three kinds of system administrators:

1. Unrestricted system administrators.

2. Restricted system administrators.

3. Project administrators.

The system administrators are users of Multics like any other users. They must log in the same way that other users do, and when they log in a process is created for them in the same way as for other users. They differ from other users of Multics in that

1. They have access to certain segments which regular users do not. It is possible, but not necessary, for them to have access to all segments in the system.

2. They use special programs to manipulate the accounting data bases. In general, these programs do not make privileged calls: they are ordinary PL/1 programs which manipulate data in ordinary ways. The data for these programs, though, are accounting records and control segments which normal users have no access to.

3. The system will grant certain requests for them which it will not grant for normal users. In particular, the initializer process will install system tables for system administrators, to change passwords or add new users.

4. Some privileged calls are available to the system administrators. A system administrator is quite often a very experienced system programmer, and so some special abilities, such as the privilege of being able to patch the system, are available to the system administrators. Almost no use of this ability is made in the design of the system-administration package.

The restricted and unrestricted system administrators are users on the project "SysAdmin". Since they therefore have access to the system-administration segments and programs, they can completely control the system's resources. A restricted system administrator has a special initial procedure,

accounts_overseer_

which provides him with a special set of commands designed for system administration. Although a restricted system administrator has access control privileges which would

potentially enable him to destroy any segment in the system, he is trapped in his special command system, and so can only perform specific accounting functions.

An unrestricted system administrator can use any Multics command. (He may also use the special command system.) Because the unrestricted system administrator can cause damage to the segment hierarchy, circumvent access control restrictions, and destroy resource accounting, only careful and carefully-trained programmers with a good knowledge of Multics should operate as unrestricted system administrators.

Project administrators are in charge of a particular project. A system administrator can function as a project administrator for any project. Initially, all projects are set up with only the system administrators. If a project needs more control over its usage than the system administrators can provide, they may delegate control of the project to a project administrator. Project administrators, too, can cause damage to segments, but only those of their projects. They should therefore be responsible individuals who are willing to do the necessary work to keep projects running smoothly. Project administrators are usually normal Multics programmers who have taken on additional responsibilities: they are given no special command system, but certain commands will work for them which will not work for non-administrators.

1.4 Definitions

A person is a human being or something (such as a daemon) that is treated like one by the system. Each registered person is identified by a string of one to 22 characters, beginning with a capital letter, called a person ID, which is unique within the installation. Usually, the person ID is the person's last name. Each person has several attributes which the system remembers, including a personal password and a default project ID. These items, along with information about the user such as full name and title, mailing address, and programmer number, are maintained by the system administrators.

The term user is used in two different ways. We often speak of a logged-in person as a user, and, sometimes, use the term to refer to the process which the user is controlling from his terminal. Espcially in the context of user control, however, we apply the term user to any member of the set of users (in the first sense) who could log in. In other words, we speak of a user as a registered identity.

A project is a grouping of users for resource control purposes. Each project is identified by a string of from one to nine characters, beginning with a capital letter or a digit, called a Project ID, which is unique within the installation.

Each user is associated with a project. The user is identified for access control purposes by the concatenation of his person ID and his project ID. A person may be registered as a user on more than one project: we consider him to be two different users in this case.

Since the project ID makes up the second component of each user's access control name, permission to log in on a project confers the ability to reference certain segments.

The list of persons who may log in on a project is contained in a segment within the Multics storage system. This segment is accessible only to the system and to the system administrator. The segment is known as a project definition table (pdt). There is one entry in the pdt for each user, which contains the user's attributes and usage information for the current month.

Each project has an administrator associated with it. The administrator for a project may be the system administrator, or the project administrator may be some other user of Multics. Project administrators will receive monthly usage reports describing the activity of users of the project.

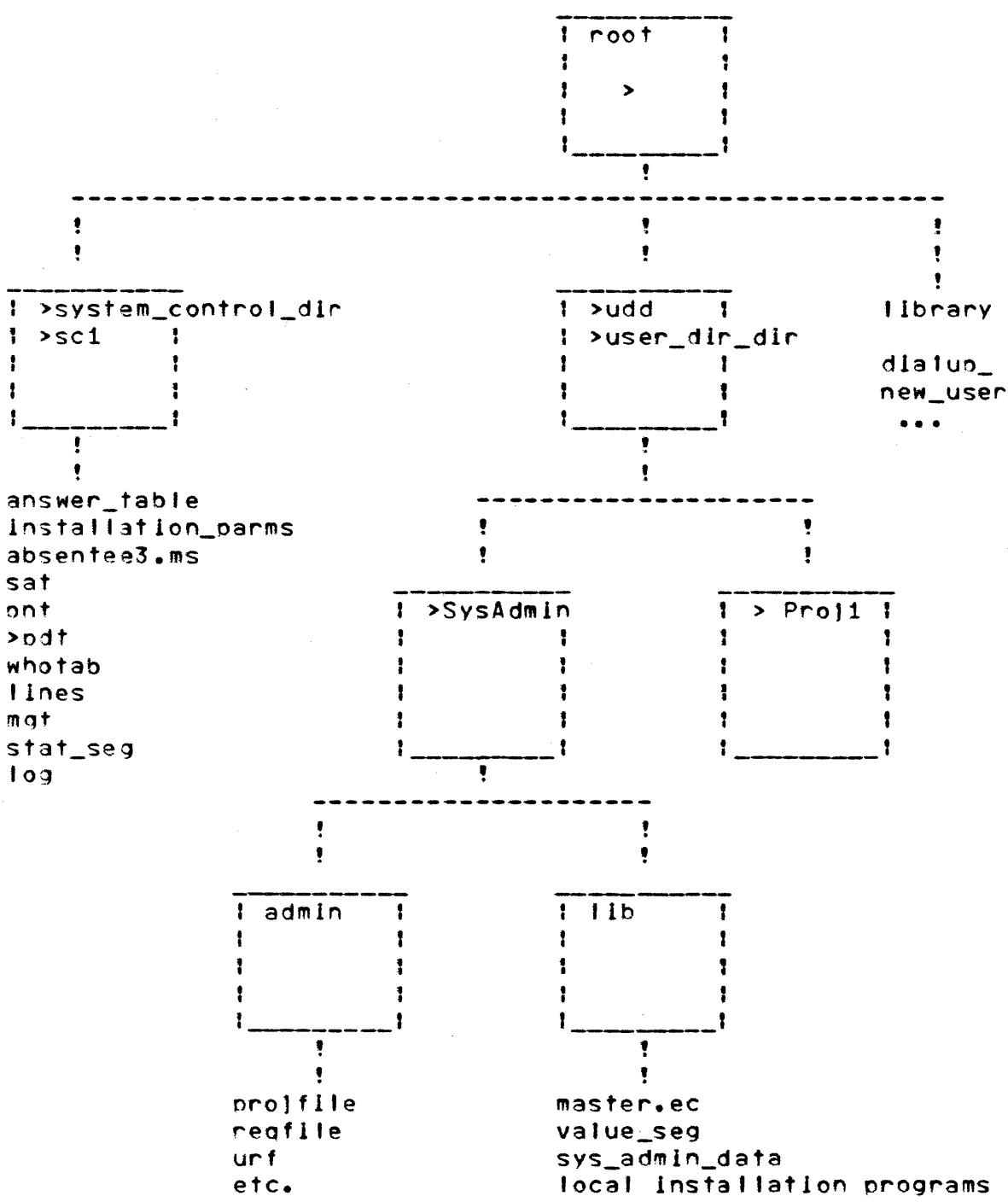
A project directory is established for each project. This

directory is contained in the directory ">user_dir_dir". It usually contains a home directory for each user of the project, although the project administrator may specify some other arrangement. The project directory has a disk quota, which limits the total amount of disk storage which the project may occupy. The project administrator may subdivide this quota among directories inferior to the project directory, or he may elect to allow some or all of the directories inferior to the project directory to have a zero quota, and to charge their usage to the project directory quota.

Each project is a member of a load control group, which is a grouping of projects which share a guaranteed access to the system. Each load control group has a quota of "primary" load units which represents a guaranteed number of users from the group who will be able to log in. Users in excess of the group's quota will be allowed to log in if the system is not full, but will be subject to preemption by primary users from other groups. The project administrators for the projects of a group may cooperate to control who from the group may have primary status, and for how long.

The load control group feature can be used to satisfy large customers of the system, while protecting the small user's ability to be able to access the system. This feature of Multics is optional, though: an installation may simply put all users in the same group and bypass the facility entirely.

The following figure is an overview of the system directory structure.



2.1 >system_control_dir

The most important directory for the system-administration programs is the initializer's home directory,

>system_control_dir

This directory, right off the root, contains those segments referenced by the initializer during the processes of

1. Logging in a user
2. Logging out a user
3. Accounting for user usage
4. Installing new system-administration tables
5. System startup

The contents of this directory include the following special segments:

lines	the list of GIOC channels to accept dialups and logins from, referenced at system start-up.
sat	the System Administrator's Table, containing an entry for each legal project plus some per-system quantities. This table is referenced at every login, to validate the user's project id.
installation_parms	installation-defined constants, including shift definitions and prices
pnt	the Person Name Table, containing the identification of all registered persons. This table is accessed at most logins, to determine the user's password.
IO__accounting	the IO daemon accounting records, one "card" per print or punch operation
stat_seg	on-line system statistics sampled every accounting update
answer_table	a table which has an entry for each terminal line, with the name and accounting data and process id of the user logged in on it
whotab	the public list of logged-in users
absentee_user_tab	the "answer_table" for absentees

absenteeN.ms	the absentee user job queue for queue N
ont.ht	a hash table for the ont
message_of_the_day	a message typed by the user process during the normal process startup
proj_admin_seg	a segment publishing the initializer's event channel used to signal updates of system-control tables by administrators
log	the initializer's log of events of interest. Many initializer messages are typed to the operator and also recorded in this segment.
admin.ec	a special segment giving sequences of commands which operators may be allowed to execute as macro-operations.
login_help	a segment which is typed if a user says "help" instead of "login"
master_group_table	a table giving the guaranteed load units for each load control group and listing the groups' current occupancy. This table is accessed at each login to determine whether the attempted login would overload the system.

In addition, other segments may be kept in >system_control_dir, such as segments necessary for system reloads.

There are two subdirectories of >system_control_dir which are important. The first directory,

>system_control_dir>pdt

contains a pdt for each legal project. Whenever a user logs in, his pdt entry is located in his project's pdt, and the user's attributes are used to initialize his process. The usage data figures in these pdt's are updated while the system is running.

The second subdirectory,

>system_control_dir>install

is used when an administrator requests the installation of a new copy of a system table.

2.2 >user_dir_dir>SysAdmin

Another important directory is the system administration's project directory,

>user_dir_dir>SysAdmin

and the directories and segments inferior to it.

2.2.1 >user_dir_dir>SysAdmin>admin

All data segments pertaining to system administration should be kept in a single directory inferior to >udd>SysAdmin. At MIT, this directory is called

. >udd>SysAdmin>admin

It contains the following:

urf	the user registration segment, giving names and addresses, and person ID's. This segment is used when a person is registered, to guarantee that his person ID is unique at the installation.
urf.ht	a hash table for this segment
reqfile	the requisition segment, containing account number, billing address, total charges, and requisition number for each usage account.
projfile	the project segment, containing descriptive information about each project, including title, supervisor address, and disk storage usage.
miscfile	the miscellaneous charges and credits journal
pmf.archive	an archive of all pmf's for projects run by the system administrator
PNT.pnt	a copy of the system pnt
pnt.ht	a hash table for this segment
smf.cur	the current smf This is converted to a sat by "cv_smf"

smf.cur.sat	a copy of the current sat
disk_stat	a segment giving disk quota and usage for all directories with quota
installation_parms	a backup copy of the same segment in >system_control_dir
lines	a backup copy of the same segment in >system_control_dir
smf.new	the next version of the smf. some commands make changes to this segment. All changes can be put into effect by typing "new_smf"
cutrpt	a report, showing accounts which are cut off, prepared daily
sumry	a report of account status, prepared daily
diskreport	a report, giving project disk usage, prepared on request
delegated_pmf.archive	an archive of all pmf's for delegated projects, giving the pmf at the time of delegation. Changes by project administrator are <u>not</u> reflected.
daemon_backup.ec	this segment becomes an "absin" segment at the end of the month which lists all daemon use
meter_data	a copy of the contents of >system_control_1>stat_seg, made every day, for use in generating system statistical reports.
today.use_totals	a statistical data base which describes month-to-date system resource availability and usage.
yesterday.use_totals	the previous day's copy of today.use_totals.
daily_report.control	a segment which describes which projects will be summarized under each category in the daily and monthly statistical reports.
system.report	the daily statistical report.
oldpmf.archive	an archive of pmf's for deleted projects

PNT.pnt*n*	previous copies of the pnt, in case of crash
smf.old*n*	previous copies of the smf, in case of crash
daily_log_*N*	extracts from the system log, prepared for printing each day
log_select_file	a segment which controls the program "daily_log_process" in its selection of messages from the log for daily printing
suffix_list	a segment which controls the program "disk_usage_stat" in its selection of segment names to summarize
termseg	a segment which lists terminals by terminal id code, used by "console_report"
termuseg	a segment which lists the users of each terminal created by "console_report"

Other data segments kept in this directory include backup copies of billing data for previous months and other temporary segments. It is in this directory that the monthly bills and statistical reports are written.

"Filled" and processed IO_accounting segments and log files for the current month are kept in a special subdirectory,

>udd>SysAdmin>admin>history

until the end of the month.

Every night, the system_control_dir copies of the sat and the pdt's are copied into the directory

>udd>SysAdmin>admin>safe_pdts

to provide backup billing data. The copies are also used at the end of a billing period, to reset the usage charges in the system pdt's.

Copies of the usage totals, "projfile", and "reqfile" for previous months are kept in another special subdirectory,

>udd>SysAdmin>admin>HF

for at least a month, so that the billing can be re-run if necessary. These segments should be dumped to tape and deleted after a reasonable period.

2.2.2 >user_dir_dir>SysAdmin>lib

Those segments which are not sensitive, such as system-administration programs not contained in the library, are kept in a different directory. This facilitates dumping of these segments separately, and aids in the startup of new installations. At MIT, this second directory is called

>udd>SysAdmin>lib

This "accounting library" directory may contain:

master.ec	the main exec_com segment for the administration
err.ec	the exec_com segment invoked when errors occur
biller.ec	the exec_com segment invoked for billing
sys_admin_data	the interlock to prevent multiple editing of the same segment
value_seg	a symbol-table segment which gives the value of certain constants used in the system-administration operations

Installation-dependent system administration tools are also kept here. All system administrators should do a "set_search_dir" so that they use the programs in this library.

Chapter 3: Administrator Commands

The following section provides operating instructions for the functions available to a system administrator. If the system administrator is operating in "restricted" mode, under control of "accounts_overseer_", then all he needs to do is type the function name followed by its arguments. If a system administrator is operating under control of the standard Multics overseer, "process_overseer_", he must precede each command by the string "ec master", e.g.,

```
ec master func args ...
```

The examples given below assume that the administrator is restricted. Lines prefixed by "u)" are typed by the user and lines prefixed by "s)" are typed by the system.

add_anon

Operation: add anonymous user to project

Command: add_anon

Usage: This function adds an anonymous user to a non-delegated project with given home directory and initial procedure and (optional) password.

Example: To add an anonymous user to the project "Proj5", type

```
u) add_anon Proj5 limited_service_system_ >udd>Proj5
s) Warning: anonymous user has no password
s) r 1557 12.558 40+34
s) Signal from System Control process: PDT installed
```

bill

Operation: run monthly bills

Command: bill

Usage: This function calls "biller.ec" to aid in billing operations.

Example: To prepare for billing, type

u) bill prepare

To run the bills, type

u) bill run

To accept a bill, type

u) bill accept

To clean up after the bills have been written, type

u) bill delete

See Chapter 4 (page 57) for a complete explanation of the billing process.

chaddr

Operation: change person's mailing address

Command: chaddr

Usage: This function calls the "change" function to change a person's address. This function is used to change just the address for a single person. It does not install the PNT: to do this, use "install PNT.pnt" or do a regular change and answer yes. The address must be enclosed in quotes. If the address is not supplied, the old address will be displayed, and a change accepted.

Example: The address must be enclosed in quotes. To change the address for "Jones", type

```
u) chaddr Jones "MIT 39-895"
s) r 1557 12.558 40+34
```

---

chalias

Operation: change person's alias

Command: chalias (obsolete)

Usage: This function has been removed from the system. The alias feature is not currently recommended, because we do not see a good way to implement it consistently. Design discussions are going on about the alias feature, and it will be either completely removed or fully supported.

change

Operation: modify user registration data

Command: change

Usage: This function is used to review and possibly modify user registration data in the urf and pnt. Each item is typed out, and then the administrator may leave it unchanged by typing "no" or a carriage return, or replace it by new data.

Example: To change the default project for the user "Smith", type

```
u) change
s) Enter userid
u) Smith
s) If you want to change any item, type the new data.
s) Otherwise type "no" or hit carriage return.
s) Name:              Smith, John
u)
s) Address:                       MIT 3-001
u)
s) Programmer number:             1234
u)
s) Project:                       Quark
u) Physics
s) Change password?
u) no
s) More users to change?
u) no
s) Install now?
u) yes
s) Installing new pnt.
s) Installation completed.
s) r 1557 12.558 40+34
s) Signal from System Control process: PNT installed
```

Notes: The user's name should be given last name first, then a comma, then first name, a space, and middle initial followed by a period. Names of the form "Smith, J. Alfred" are also allowed. If the user has a title, such as "Prof.", the title should follow the name and be separated from the name by a colon, as in "Smith, John J.:Prof.". You may type "stop" at any time to abort all the changes you have made to a particular user and start over. In response to the query "Change password?" you may answer "yes" and then give a new password, or "verify" to

change

check that you have the correct password, or "no" to leave the password as is. The last name may be up to 32 characters long. The first name and middle initial field may be up to 24 characters. The address field may be up to 32 characters total: use slashes to separate lines of address. The programmer number and title may each be up to 8 characters long.

chdf_proj

Operation: change person's default project

Command: chdf_proj

Usage: This function calls the "change" function to change a person's default project. This function is used to change just the default project for a single person. It does not install the PNT: to do this, either type "install PNT.ont" or do a regular change and answer yes. If the default project is not currently in the project segment, a warning message will be printed. If the default project is not supplied, the old value will be displayed, and a change accepted.

If the user has changed his own default project, with the "-change_default_project" argument to login, this change will not be effective. A special tool is available to unrestricted system administrators to force the change of a default project: this program is called "pass_util".

Example: To change the person-id "Smith" so that his default project is "Proj3", type

```
u) chdf_proj Smith Proj3
s) r 1557 12.558 40+34
```

---

check_log

Operation: check log for info on user

Command: check_log

Usage: Sometimes a user will complain that he can't log in. To find out why, you can scan the log segment for the messages placed there by the initializer which give the reason.

Example: To see why "Jones" can't log in, type

```
u) check_log Jones
s) 702 09/22/71 1333.2 0 lg_ctl_: no  1  Jones.Multics  2741
022 chn tty302 reason bad_pass
s) r 1557 12.558 40+34
```

This message says that the user could not log in becuase he gave the wrong password. His terminal ID was 022, and his channel was tty302. This message has severity 0 (so it was not typed to the operator, just logged), and is the 702nd entry in the current log.

The following is a list of reasons for not logging a user in:

bad_pass	bad password
badpers	person not in pnt, either not added or mistyped
no_name	no name given
pwlocked	password locked by administrator
bad_proj	project does not exist in sat. may be mistyped
no_pdt	project's pdt is not in >sc1>pdt. message with stars also typed online
not_pdt	user not in pdt for project. did you forget to do an "upmf?"
anon_pw	bad anonymous-user password
already	user already logged in, and no "multip" flag (absentee does not count)
no_acct	no default account ID
acct_no	project's account out of funds or past termination date
hd_path	bad path name for home directory
hd_nodir	what should be home directory is non-dir branch
hd_err	some error in getting status of home directory
hd_make	error creating home directory branch. message in log
hd_where	home directory missing, does not seem to be below project dir. we will not create.

sys_full	system full according to load units
saturate	system full according to max. number of users
cant_bum	system full, user cannot find anyone to bump
groupmax	load control group at absolute maximum
no_group	load control group in sat not in mgt
nf_nosec	system not full, group full, no secondary
sysgrpfl	system full, group full, cannot bump
prog_err	programming error in lg_ctl_

chname

Operation: change person's mailing name

Command: chname

Usage: This function calls the "change" function to change a person's mailing name. This function is used to change just the name for a single person. It does not install the PNT: to do this, use "install PNT.pnt" or do a regular change and answer yes. The full name must be enclosed in quotes. If the name argument is not given, the old name will be displayed, and a change accepted.

Example: The name must be enclosed in quotes. To change the mailing name for the person-id "Jones", type

```
u) chname Jones "Jones,W. Alfred"
s) r 1557 12.558 40+34
```

chpass

Operation: change person's password

Command: chpass

Usage: This function calls the "change" function to change a person's password. This function is used to change the password for a single person. It does not install the PNT: to do this, either type "install PNT.pnt" or do some other change and answer yes. This function can also be used to verify a person's password. Answer "v" or "verify" to the first question, and the program will ask for a password. If the password you give matches the person's password, the command will type "ok". Otherwise, the command will type "wrong" and ask if you want to change (or try again to verify) the password.

If the user has changed his own password, with the "-change_password" argument to login, this change will not be effective. A special tool is available to unrestricted system administrators to force the change of such a password: this program is called "pass_util".

Example: To change the password for the person-id "Jones", type

```
u) chpass Jones
s) Change password?
u) yes
s) Password:
u) newpass                          (the printer is turned off)
s) r 1557 12.558 40+34
```

---

chprog

Operation: change person's programmer number

Command: chprog

Usage: This function calls the "change" function to change a person's programmer number. This function is used to change just the programmer number for a single person. It does not install the PNT: to do this, either type "install PNT.pnt" or do a regular change and answer yes. The programmer number must be all numeric. It may be up to 8 characters long. Some installations use the "man number" assigned by the company in this field. If the programmer number is not supplied, the old value will be displayed and a change will be accepted.

Example: To chage the programmer number for the person-id "Smith", type

```
u) chprog Smith 7399
s) r 1557 12.558 40+34
```

cu

Operation: create user directory

Command: cu

Usage: This function is used to create a user's home directory. Home directories are created when the user first logs in, and the "upmf" function calls this function, but in rare cases, it may be necessary to execute this function separately.

Example: To create the home directory for "Jones.Gamma", type

```
u) cu Jones Gamma
s) r 1557 12.558 40+34
```

---

day, dayok

Operation: print daily billing output

Command: day, dayok

Usage: Every night, a self-rescheduling absentee job called the "crank" runs which does the daily charging of users and checks for users who should be cut off.

The "day" command prints the output from the crank on the console, and asks whether or not to delete the output. The "dayok" command just asks whether to delete the output. Unless there was an error, the absentee output segment should be deleted.

Example: The output from the crank should look like this:

```
u) day
s)
s) Absentee user Accountant.SysAdmin logged in ...
s) r 0330 12.558 40+34
s)
s) loacur - 731 cards, 20 charged, 708 previously collected,
3 next period.
s) Total charge: 345.67
s) Begin charging for 7/31/71 2355.0 to 8/9/71 2345.1
s) cut 3, warned 7, total charge $45678.90
s) r 1557 12.558 40+34
s)
s) Absentee user Accountant.SysAdmin logged out ...
s) Delete?
u) yes
```

See section 5.3 (page 63) for a complete explanation of the daily processing.

delegate

Operation: give project to project administrator

Command: delegate

Usage: This function is used to give a project administrator control over a project's pmf. Once this is done, the project may add and delete its own users, and set resource-usage limits on users in the project.

The command moves the project's pmf from "pmf.archive" to a specified directory (keeping a spare copy in "delegated_pmf.archive"), edits the smf to show that the project administrator may install pdt's for this project, and sets access on "proj_admin_seg" and ">system_control_dir>update" so that the administrator may install his pdt.

A delegated project is under control of the project administrator only. The system administrator will receive an error message if he attempts to execute the "dpmf", "upmf", or "pmf" functions on a delegated project. Use the "undelegate" command if you wish to take a project back from a project administrator, either in order to have it controlled by the system administrators again, or so that it may be delegated to some new user.

Example: To give "Jones.Gamma" control of project "Beta", placing the pmf in ">udd>Beta", type

```
u) delegate Beta >udd>Beta Jones.Gamma
s) archive: Beta appended to delegated_pmf.archive
s) $ do a "new_smf"
s) r 1557 12.558 40+34
```

disk_report

Operation: write disk usage report

Command: disk_report, drp, disk_auto

Usage: This function is used to calculate a disk usage report. The administrator may type "disk_report" to cause a manual disk usage calculation. Normally, though, disk usage is done automatically every night by the absentee job "dodrp", which executes the "disk_auto" function. (See chapter 5 for details on how the absentee use is managed.)

The program "sweep" is used to get quota for all directories into the data segment "disk_stat". The program "charge_disk" then charges these usage figures to projects in their "proj file" entries. A printable segment, called "diskreport", is produced but not automatically printed. To print copies of this segment, type "rqm diskreport".

Example: To run a disk report, type

```
u) disk_report
s) $ Creating disk usage report.
s) $ Following figure is total quota/current use
s) 75500/64432
s) Charged 906 directories out of 910 to 108 projects
s) r 1557 12.558 40+34
```

dpmf

Operation: remove user from project

Command: dpmf

Usage: This function deletes a user from a project master file for a project which is managed by the system administrators.

Example: To delete user "Black" from project "Gamma", type

```
u) dpmf Gamma Black
s) r 1557 12.558 40+34
s) Signal from System Control process: PDT installed
```

dproj

Operation: delete project

Command: dproj

Usage: This function is used to delete a project. It edits the smf to remove the project entry, calls "epro" and "erf" to set the date off for the project, moves the pmf for the project to "oldpmf.archive" in case it is ever needed again, and calls deletedir to delete the project directory and all of its contents.

Example: To delete project "Delta", type

```
u) dproj Delta
s) archive: Delta added to oldpmf.archive
s) type
s) type
s) dd: do you want to delete the directory >udd>Delta??
u) yes
s) $ do a "new_smf"
s) r 1557 12.558 40+34
```

Notes: If you answer "yes" to the question about deleting the directory, the project directory and all segments and directories inferior to the project directory will be deleted. The project will be charged for disk usage until the project directory is deleted. If you answer "no", the project's directory and segments will not be deleted, and the project will continue to accumulate storage charges. Do not answer "no" unless there is some exceptional reason for doing so, since the project directory will have to be deleted later by an unrestricted system administrator.

install

Operation: Install system table

Command: install

Usage: This function is used to install a system table. It can be used when the table has already been converted, in case a previous installation request failed due to a crash.

Example: To install the current pnt, type

```
u) install PNT.pnt
s) r 1557 12.558 40+34
s) Signal from System Control process: PNT installed
```

lson

Operation: see if person is registered

Command: lson

Usage: This function types "true" or "false" depending on whether a person is registered in the pnt or not. It then lists all users in the user registration file who have a last name which matches the argument.

Example: To check whether the user "Jones" is registered, type

```
u) lson Jones
s) true
s) Userid for "Jones, Herbert R." is "HJones"
s) Userid for "Jones, Peter" is "Jones"
s) Userid for "Jones, W. Alfred" is "AJones"
s) Number of users with last name "Jones" is 3
s) r 1557 12.558 40+34
```

labels

Operation: print mailing labels

Command: labels

Usage: This function prints a set of mailing labels offline.

Example: To run the labels, type

```
u) labels
s) r 1557 12.558 40+34
```

---

misc

Operation: process charges and credits

Command: misc

Usage: This function is used to input miscellaneous charges and credits. For each transaction, the project ID, the amount (negative for a credit), and an explanation are required. Each transaction will result in a separate line on the monthly bill. The transactions are stored in "miscfile", and the total is also placed in "projfile". All three input items for a transaction may be put on the same line, or they may be supplied one at a time. The program will indicate the type of item to be input next. Typing "x" instead of a project ID exits from this function.

Example: To credit project "Alpha" for a crash, and charge project "Beta" for some manuals, type

```
u) misc
s) proj
u) Alpha
s) amt
u) -10.55
s) explanation
u) system crash 6/23 Jones
s) proj
u) Beta 4.50 Manuals Smith
s) proj
u) x
s) r 1557 12.558 40+34
```

new_proj

Operation: add new project

Command: new_proj

Usage: this function is used to add a new project.

Example: To add new project "Gamma", giving the user "Fooch.Gamma" project administrator status (i. e. write access on the project directory - the project is not delegated), and setting an initial quota of 100 records, type

```
u) new_proj Gamma Fooch.Gamma 100
s) archive: Gamma appended to pmf.archive
```

First, the system asks for the project title. This field should be a short description of the purpose of the project. It may be up to 52 characters long.

```
s) type type title
u) Gamma Ray Research
```

Next, the system asks for the name and address of the principal investigator. This is the individual to whom the usage report will be addressed. Both name and address may be up to 32 characters long.

```
s) inv
u) Prof. Q. X. Jones
s) inv_addr
u) MIT 39-895
```

Next, the system asks for the name, address, and telephone of the project supervisor. This is the individual in direct contact with the project's day-to-day activities. He may be the same person as the principal investigator, or he may not. Often, he is registered as a user or project administrator for the project. The name and address may each be up to 32 characters long. The telephone number may be up to 16 characters long.

```
s) sup
u) Mr. Melvin Fooch
s) sup_addr
u) MIT 39-896
s) sup_phone
u) MIT x1234
```

---

new_proj

At this point, the system has finished adding the project to "projfile". You are now asked if you wish to review and correct the entry (by typing "p" and then perhaps "c") or if you wish to continue with the registration procedure (by typing "file").

```
s) type
u) file
```

Now, the system adds an entry for the project to the requisition file, "reqfile", first asking for the external account number and requisition or purchase order number.

```
s) type account
u) 11792
s) req
u) 123456
```

The system next asks for the cutoff limits for the project. The funds limit is a dollar amount (enter "0" to show an "open" amount). The date cutoff limit is a date, expressed as either "mm/dd/yy" or "mmddyy". If the project exceeds either of these limits, all users on the project will be prevented from logging in, but the project will continue to incur disk and registration charges until you delete the project.

```
s) funds
u) 0
s) cutoff
u) 7/30/72
```

Next, the system asks for the name and address to which charges incurred by the project should be reported. The name and address may each be up to 32 characters long.

```
s) name
u) Fiscal Office, Attn: L. Spottswood
s) addr
u) MIT 1-101
```

At this point the project has been added to the requisition file. You are now asked whether you wish to review and edit the entry (by typing "p" and perhaps "c") or if you wish to finish registering the project (by typing "file").

```
s) type
```

new_proj

```
u) file
s) $ do a "new_smf"
s) r 1557 12.558 40+34
```

new_smf

Operation: convert smf to sat and install

Command: new_smf

Usage: This function is used after any change is made to the smf. It converts the smf to a binary sat, reruns "daily_summary" to insert cutoff codes, and requests the initializer to install the new sat. Those functions which make changes to the smf will remind the administrator to do a "new_smf". Since this function is time-consuming, it should be done at the end of a bunch of changes.

Example:

```
u) new_smf
s) r 1557 12.558 40+34
s) Signal from System Control process: SAT installed
```

pmf

Operation: Edit and convert pmf, install pdt

Command: pmf

Usage: This function is used to do arbitrary editing of a pmf. It throws the administrator into the qedx editor after reading in the segment, and allows him to make any changes he wants. Because qedx is complicated to learn, this function is not for beginners. After the administrator exits from qedx, the function converts the pmf to a pdt, and signals the initializer to install the pmf. The function takes care of updating "pmf.archive" as well.

Example: To edit the segment "Operator.pmf", type

```
u) pmf Operator
s) Edit.
u) (editing commands)
u) w
u) q
s) r 1557 12.558 40+34
s) Signal from System Control process: PDT installed
```

---

proj_mtd

Operation: month-to-date report

Command: proj_mtd

Usage: This command types a month-to-date report for any project's usage. The report lists all users on the project and their dollar totals, as well as disk and miscellaneous charges.

Example: To get a report of the usage for project "ALPHA", type

```
u) proj_mtd ALPHA
s) Month to date for proj ALPHA

s) Name              logins    charge

s) White                 11 $  133.41
s) Brown                  0 $    0.00

s) 2 users               11 $  133.41

s) registration            $   20.00
s) misc                    $    0.00
s) disk                    $  176.08

s) Total                   $  329.49

s) r 1557 12.558 40+34
```

recov

Operation: generate pmf from pdt

Command: recov

Usage: This function is used if a project's PMF is destroyed. The function generates a new PMF from the system's binary PDT.

Example: Suppose the project "Proj2" loses its pmf. To get a new one, type

```
u) recov Proj2
s) r 1557 12.558 40+34
```

register

Operation: Register new person

Command: register

Usage: This function is used to register a new person-id. It enters him in the "urf" and in the "pnt". If a person is already a user of Multics, or if he was once registered and was not removed, this function should not be used since the user will still be in the urf and pnt.

Enter "stop" at any time to abort the processing of the current user, for instance if you have misspelled his last name, or if he is already registered.

Example: To register a user, the dialogue goes like this:

```
u) register
s) Enter full name (Last,First I.)
u) Smith,Robert M.:Prof.
s) Enter mailing address
u) MIT 39-575
s) Enter programmer number or "none"
u) 6789
s) Enter default project
u) Language
s) Password:
u) wugga                (the printer is turned off)
```

The system will attempt to generate a unique person ID for the person being registered, by trying first his last name alone, and then his last name prefixed by his initials. If either of these guesses succeeds, the system makes a tentative assignment and asks if the person ID is acceptable. If neither of the guesses succeeds, and if you reject the system's guess, the system will ask you to specify a person ID, and will then check to make sure that the ID is unique.

```
s) Userid "Smith" is already used by "Smith, Frank X."
s) Trying "RMSmith" for userid.
s) Userid assigned is "RMSmith"
s) Is this ok?
u) no
s) Please suggest a userid for "Smith, Robert M."
u) RSmith
s) Userid assigned is "RSmith"
```

```
s) Is this ok?
u) yes
```

At this point, the user has been added to the user registration file and the PNT. You may now add more users, or if you are finished registering users, you may install the PNT immediately or later.

```
s) More users to add?
u) no
s) Install now?
u) yes
s) Installing new pnt.
s) Installation completed.
s) r 1557 12.558 40+34
s) Signal from System Control process: PNT installed
```

---

rename_proj

Operation: rename project

Command: rename_proj

Usage: This command is used to rename a project.

Example: To rename project "AAA" to "BBB", type

```
u) rename_proj AAA BBB
s) Now do a 'new_smf' and then a 'pmf BBB'
s) also change default project for users on BBB
s) r 1557 12.558 40+34

u) new_smf
s) r 1557 12.558 40+34
s) Signal from System Control Process: SAT installed

u) pmf BBB
s) Edit.
u) q
s) r 1557 12.558 40+34
s) Signal from System Control Process: PDT installed
```

rqm

Operation: request statistics to be printed

Command: rqm

Usage: This function requests the dprinting of a report for all administrators. The single argument is the name of a segment to be dprinted.

Example: To send a copy of the disk usage report to all administrators, type

```
u) rqm diskreport
s) r 1557 12.558 40+34
```

---

setcrank, setdisk

Operation: Schedule absentee jobs

Commands: 3.34 setcrank, setdisk

Usage: These commands do the initial scheduling of the absentee jobs described in chapter 5 (page 62) which perform daily accounting. Unless the absentee jobs crashed or the absentee job queues were lost, there is no need to execute these commands. to check whether a job is scheduled, type

```
lar -queue 1 -long
```

If the jobs "dodrp" or "crank" are not scheduled, execute "setcrank" or "setdisk" as appropriate.

undelegate

Operation: Undelegate a project

Command: undelegate

Usage: Execute this command if you have delegate a project to some user, and wish to regain control of it or delegate it to some other user.

Example: To change the project "BLAH" from control by Smith.BLAH to control by Jones.BLAH, type

```
u) undelegate BLAH
s) archive: BLAH appended to pmf.archive
s) $$$ do a new_smf
s) r 1557 12.558 40+34

u) delegate BLAH >udd>BLAH>Jones Jones.BLAH
s) archive: BLAH appended to delegated_pmf.archive
s) $$$ do a "new_smf"
s) r 1557 12.558 40+34

u) new_smf
s) r 1557 12.558 40+34
s) Signal from System Control Process: SAT installed
```

upmf

Operation: add user to project

Command: upmf

Usage: This function adds a user to a project. It is used only on projects which are not delegated. The function extracts the project's pmf from "pmf.archive", edits the pmf to add the new entry, converts the pmf to a pdt, signals the initializer to install the pdt, and replaces the edited pmf in "pmf.archive".

Example: To add user "Jones" to project "Gamma", type

```
u) upmf Gamma Jones
s) r 1557 12.558 40+34
s) Signal from System Control process: PDT installed
```

Other commands

Operation: other commands

Commands: epro, erf, mail, help, logout, status, listacl, listcacl, lar, car, mq, gq

Usage: Consult the MPM and the MPM SPS for details on the usage of these commands. A short explanation of each follows.

car	cancel absentee request. Use this function if an absentee job is scheduled too many times by mistake.
epro	edit "projfile". Use this function to change a project's supervisor name, title, etc.
erf	edit "reqfile". Use this function to change a project's account number, requisition number, funds, cutoff date, etc.
gq	getquota. Use this function to print a directory's disk quota and usage.
help	print information on command usage.
lar	list absentee requests. Use this function (usually with the "-q 1" argument) to list your absentee requests.
listacl	list access control list. Use this function to check accessibility of a segment.
listcacl	list common access control list. Use this function to check accessibility of the contents of a directory.
logout	log out from Multics.
mail	read or send mail. To read mail sent to you, type mail To send mail to some other user, type mail * Person Project and then type your message, ending with a line consisting only of ".".

Other commands

mq	movequota. Use this function to shift quota to a directory from its superior.
status	print information about a segment. Use this function to see when a segment was created.

Chapter 4: Billing

The following is a short explanation of the monthly billing process and the information used in monthly billing.

The most important thing to remember is that the monthly billing process consists mostly of a print pass over the saved PDT's and a print pass over "reqfile". Other information is also taken from "projfile", "miscfile", and "billing_footnote". No pricing out of usage is done during the billing process: this has all been done during the daily processing.

4.1 Output from billing

The reports output from the monthly billing process are:

1. long_bill - this report is a complete breakdown and justification of charges for each project. For each project, the bill will have from one to four sections:

   a) charge summary, by user
   b) interactive usage, by user
   c) absentee usage, by user
   d) I/O daemon usage, by user

   In addition, the charge summary shows the project's disk and miscellaneous charges, lists the current prices, and may have a footnote intended as a message to all project supervisors.

2. short_bill - this report consists of just the charge summaries from the long bill. The prices and footnotes are suppressed. An elaborate grand total page is printed. This report is intended for the use of system administrators and facility directors, who find the long bill too bulky.

3. bill - this report is a listing, by account number, of the charges made to each account. One line is printed for each project, showing the charges this month and the charges to date, face amount, and requisition balance.

4. msum - this report is the monthly summary. It has one line per requisition or purchase order, and gives the same information as in "bill".

5. cards - at the MIT installation, this deck is punched for

transmittal to the MIT General Accounting Office. The program "punch_MIT_deck" produces this segment. The peculiar format of the cards and the many ad hoc decisions in the code are the result of the accounting office's requests.

It is expected that other installations will remove all mention of this program, if they don't need any external record of charges, or that they will insert some local program which may or may not produce cards in place of "punch_MIT_deck".

6. miscs.print - this is a listing of all miscellaneous charges and credits for the month.

8. dback - this report is the result of an absentee job fabricated by the billing process and run later. It lists every daemon session during the month in chronological order, and shows, for each session, who used the daemon, how much he used, and what shift and queue he used.

9. system_month.report - this segment gives a summary of Multics usage for the billing period.

10. diskreport - this report shows each project's disk usage for the month. It also has a map of every directory in the hierarchy which has a disk quota, giving its current usage, its charge for the month, and its quota.

11. console.report - this report shows the usage of each terminal, sorted by terminal ID. For each user who used a given terminal, the total connedt time during the month and the number of logins is shown.

4.2 Inouts to billing

The inout segments to the billing process and what of their contents is used are listed below.

1. PDT's - the copied pdt segments in >udd>SysAdmin>admin>safe_pdts contain the complete usage data for each user for the billing period.

2. projfile - this segment is used mostly to get the disk usage figures, which are stored there when a disk report is run. The project title and the name and address of the project supervisor are also used to create headings on

the long and short bills.

3. reqfile - this segment has the charges which are actually billed. Daily processing has updated "reqfile" from the figures in the PDT's, so the two should agree. (An error comment is printed if they do not, and the "reqfile" values are used.) The name and address of the person in charge of the account as well as requisition numbers, amounts, and cutoff dates are also extracted from this segment.

4. miscfile - this segment is the journal for miscellaneous charges and credits associated with a project. On both the short and long bills, all entries for a particular project are located in this segment and re-printed. (If the total of charges and credits in "projfile" does not match the total from adding the individual entries in "miscfile", an error comment will be printed and the total from "miscfile" actually billed.)

5. billing_footnote - this segment is optional. If present, all the text in it will be printed at the bottom of each project's usage summary. This segment can be used to announce forthcoming price changes, or make other announcements to the administrators for each project. It is printed on the long bill only.

6. disk_stat - this segment is used in the preparation of the disk report.

7. today.use_totals - this segment is used in the preparation of system_month.report.

8. The accounting segments saved in the subdirectory "h" are also used in the preparation of "dback".

4.3 Steps in billing

The procedure for preparing a bill consists of three parts: preparation, actual running-off of the output, and cleanup operations once the bill is judged correct.

The preparation phase consists of checking for any un-processed dormant accounting segments, and checking to see that a disk report has been run recently.

```
bill prepare
```

does most of this. The administrator should also make sure that the segment "billing_footnote" is up-to-date, and that all miscellaneous charges are input. The administrator may run a set of mailing labels at this time.

The actual running of the bill is initiated by typing

```
bill run MM DD YY BXXXXX
```

where MM/DD/YY is the date of the last day of the month. The argument BXXXXX is used by "punch_MIT_deck" - other installations will probably replace this argument with others more appropriate to their needs.

The "run" processing prepares all of the above reports and bills. It prints one copy of the bills in the highest queue before starting on the usage summary report.

As soon as the administrator has examined his copy of the bill and found no gross errors, he may type

```
bill accept MONTH
```

which causes the following:

1. printing of many copies of the billing output and reports. The addresses for these reports are obtained from "value_seg". This printing is done by the "rqbill" function.

2. Submission of absentee jobs to create the output reports "console.report" and "dback".

3. Copying of the segments used to create the bill into the directory HF. The names of these segments are prefixed by the MONTH specified as an argument -- for example, "reqfile" is copied into "HF>MONTH.reqfile".

4. Resetting of the various data bases for the next month. The program "reset_usage" is run to subtract the user usage figures in the copied pdt's in >udd>SysAdmin>admin>safe_pdts from the usage figures in the pdt's in >system_control_dir>pdt, so that the month-to-date usage totals will start over for a new month. The program "uc_ctr" is also run, to add the "charges this month" field in "reqfile" into the "charges this requisition", and reset the "charges this month". Any entries in "reqfile" or "projfile" which are marked as having been deleted during the month are then deleted.

5. Resetting of the disk meters in the directory branches. The program "reset_disk_meters" is used to subtract the value of the time-page-product integrator used for charging (stored in "disk_stat") from the value in the branch.

Once the bills have been mailed out, and everybody who should have a copy has got one, the administrator should delete the bills from the disk by typing

```
bill delete
```

The amount of storage required by the bills will vary, of course, depending on the number of users registered on the system, and the number of sessions used during the month. It may come to several thousand disk records. Be sure to have enough quota before starting the billing run.

Chapter 5: Daily Processing

The daily billing and statistics are produced through the agency of two absentee jobs which run every night, and which re-schedule themselves to run again the next night. The first job is the disk storage accounting. The second is the actual accounting-segment update run. Both jobs produce reports which are dorinted for the system administrator.

5.1 Use of Absentee

The disk accounting job is controlled by the command segment

dodrp.absin

This job executes only one command, namely "disk_auto" and then logs out.

The accounting-segment update run is controlled by the command segment

crank.absin

which also executes only one command, namely "crank", and then logs out.

Each job reschedules itself in queue 1 for the next night, by executing the "enter_abs_request" command to re-submit the job. The time at which the job is scheduled is kept in value_seg, under the variable names "crank_time" and "disk_time". The disk report should run before the crank, in order to charge the most up-to-date disk figure to the users.

The disk report job is restartable, so that if the system crashes during its run, it will be rerun.

The crank job cannot be restartable. If the system crashes and segments are reloaded, users might be double-charged or not charged at all for some usage. (See chapter 10, "Crash Recovery", for more information.) Therefore, the crank job is marked not restartable, and furthermore the variable "abort_crank" in value_seg is used to stop the crank from running again, if it crashes. When the crank is about to enter its critical region, it sets "abort_crank" to "true". When it passes the critical region, it sets "abort_crank" to "false". (Some parts of the job, those dealing with summaries and so forth, need not be protected.) When the crank first starts up, it checks

"abort_crank", and if it is "true" it immediately logs out.

5.2 Disk reporting run

See the writeup of "disk_auto" for details on what is done by the disk reporting run. The important result of a disk reporting job, from the point of view of charging, is the storing of current disk usage figures in the segment "projfile". These figures will be picked up by the accounting update job later and used to charge the project for disk usage.

5.3 Daily accounting run (crank)

The "crank" job executes a command of the same name in master.ec. This command performs the follwoing steps:

1. reschedule the crank for tomorrow

2. process all IO daemon records in >system_control_dir>IO_accounting which have not yet been processed and which have been completed. Call charge_daemon_usage to add this usage the the appropriage PDT entries in >system_control_dir>pdt.

3. make a backup copy of the answering service version of the pnt.

4. call pdt_copy to copy all system pdt's into safe_pdts

5. run "compute_bill" to charge user usage, from the pdt's, to the appropriate entries in reqfile.

6. run "daily_summary" to write a report giving account status and to modify the sat to cut off projects which are out of funds.

7. install the new sat.

8. run "daily_log_process" to prepare log extracts for system programmers and administrators.

9. run "copy_as_meters", "print_meters", "system_total", "usage_total", and "system_daily_report" to prepare a system statistical report.

10. dprint copies of the ouput reports.

11. check for the existence in >system_control_dir of any

data segments which need processing. IO_accounting files and logs are processed automatically. Answering service dumps are automatically printed.

The job sends a message to the system administrator reporting successful completion via the "send_message" command when it is all done.

5.4 Daily operation

Every day, first thing in the morning, the system administrator should read the output from the crank. If it has not been printed because of a system failure, or because the output is lost, the administrator should log in and type "day" (see the write-up of "day" in chapter 3, page 32) to see the output. If everything ran correctly, the administrator merely replies "yes" to the "Delete?" question and logs out. If the output has been printed successfully, the administrator can avoid having it typed again on the console by executing "dayok" instead.

Chapter 6: Adding a new Project

New_proj is the tool for adding a new project. You will want to know the following:

Supervisor

User list (must have at least one user)

account, requisition numbers

funds

termination date.

billing address

delegated? if so, what user id and what directory

initial quota

The project ID can be the same as a person ID if you want. Projects which have been deleted and then return are a problem: try to avoid this.

The sequence for adding a new project is:

1. new_proj (see page 41)

2. new_smf (see page 44)

3. register any new persons (see page 48)

4. use "upmf" to add each person to the project (see page 54)

5. delegate the project if requested (see page 33)

You may batch several projects in step 1, of course, and do many registrations in step 3.

Chapter 7: Adding and Deleting Users

To add a user to an existing project, you use the tool "upmf". (see page 54.) If the user is not already registered, upmf will type a warning message. If you get this message, register the user with "register". (see page 48.)

To delete a user from a project, use the tool "dpmf". (see page 35.) The user will remain registered as a person, and his address and password will remain in the system. If the user is leaving the installation, and is not expected to return, you should not remove him from the urf and pnt immediately, since his name may be on some access control lists. For instance, if the person named "Jones" quits the company, it would be unwise to remove the name "Jones" from the user registration file right away: suppose someone else named Jones arrived the next day. When he was added to the system, he would have access to some segments which he should not have. The unrestricted system administrator's tool "remove_user" can be used to completely remove a user from the PNT and user registration file.

Chapter 8: System Security

This section deals with making and keeping the system secure. A system may be run loosely or tightly, but some security is necessary even if the installation does not care if resources are mis-appropriated. Security measures not only frustrate immature individuals, they provide insurance that accidents will not destroy the system, and provide information about what happened when something went wrong.

8.1 passwords

Each registered Multics user has a personal password. This password is his personal property, and is associated with the person, not the user identity (combination of person and project). Project administrators need not know the passwords associated with the persons working on their project, although since the project administrator usually assists in getting a new user registered, he may know it.

Passwords are stored in scrambled form internally within the system. Once a user's password is entered (via the "register" or "change" commands) it is not stored in readable form anywhere in the system-administration data bases. When the user attempts to log in, the password he types is subjected to the same scrambling transformation, and the scrambled values are compared. This procedure makes it fairly difficult for a user to obtain another user's password, even if he is given access to the pnt by mistake.

The password mechanism is the strongest protection feature in Multics: if users cannot be uniquely identified, then different access control rights cannot be granted. Users should be encouraged to keep their passwords private, and to change them often. Sensitive passwords, in particular, should be changed regularly.

Users may now change their own passwords dynamically, by use of the "-change_password" argument to login. They should be encouraged to do so often. Once a user has changed his password by use of this login option, subsequent installations of the PNT will not reset the password. The "chpass" administrator command will continue to work though, since it uses the program "pass_util" to modify >system_control_dir>pnt directly.

The answering service keeps various counters in the PNT regarding the use of a password. In particular, the system counts

incorrect passwords, and can type a message to a user at login informing him how many times his password has been given incorrectly since its last correct use. This feature may alert a user that someone has tried to guess his password.

8.2 Access control

Access control is the system's second major line of defense. Consult MPM sections I.2.6 and II.3 for an explanation of the mechanism. Properly set, the access control mechanism can make the system very secure, at the cost of some inconvenience.

Access control works only if user identities are held secure. This is the reason that person identifiers are assigned only by the system administrator: otherwise, a clever project administrator could register an identity which corresponded to some name already on an access control list somewhere in the system, and so be able to log in and use this access control list entry for a purpose not foreseen by the creator of the ACL entry.

The general principle to apply is that no user should have write access to any system data base unless there is a good reason. Those few, trustworthy individuals who may modify system data bases should be impressed with the fact that they should not use the privilege casually. It is a good idea to run an absentee job occasionally which spot-checks the access control lists on system segments to make sure that something has not been left unprotected inadvertently.

The question of read access is less clear. It is possible to prevent users from reading anything except their own directories and their contents, or one can leave most of the system readable. Two specific cases are interesting: the choice between "e *.*.*" and "re *.*.*" on the CACL of >user_dir_dir makes the difference between users being able to tell who is registered on another project or not. The access control list on >system_control_dir>answer_table makes the difference between one logged-in user being able to determine another user's process id without permission and not.

Currently, the system is slightly deficient in that the SysDaemon and SysAdmin users have far too much power. Someday, we will change the system to lock the daemons into subsystems which cannot execute any but carefully-checked commands, and we will remove most global access rights, either getting along without them or accomplishing them in lower rings with lots of checking. As the system stands now, disclosure of a SysDaemon or SysAdmin password opens the system up completely. In the future, it would be nice to have the system secured to the point where no major

breach of the system's segment security could be accomplished without entering the machine room.

8.3 Telephone line security

Tapping a phone is very easy. If sensitive data, such as passwords, are transmitted by phone, the system is no more secure than the telephone connection box. An installation which thinks it is worth the trouble to protect against this sort of potential problem could consider dedicating GIOC channels to special-purpose uses such as daemons and administrative operations. The privileged users could then have start_up.ec segments which logged out if they were not invoked from the proper channel. Consoles on these dedicated channels could be hard-wired if close enough to the GIOC (there are other good reasons to do this) or could be connected over dedicated telephone lines, perhaps even with a pair of the "scrambler" data sets which are available commercially.

8.4 Terminal ID codes

If the terminals used on the system identify themselves with an ID code, it is often easy to track down what terminal was involved in a security breach. We are considering a method of restricting use of the system by using terminal ID codes, as an optional feature. Since these ID codes can always be faked, one cannot rely on this feature alone. But if terminal ID is free on a terminal you are considering, definitely get it. The program "console_report" is used to produce a listing of all terminals by ID code, with the list of users who used the terminal during the month.

8.5 Use of the log segment

The program "daily_log_process" can be used to monitor system operation by setting up its log_select_file to watch for conditions of interest. The version provided with the system will create a full log of all events, a log of all "interesting" events for system programming, and special logs for absentee problems and network problems. This segment can be edited as necessary to monitor specific individuals, or specific terminals.

8.6 Cross-checking in the billing process

It is very hard for any user to get away with using Multics resources without leaving some traces. Even if someone gets

logged in on a system administrator identity, and attempts to fudge the accounting records to conceal the fact that he logged in, most usage is recorded in more than one place. Unless the thief figures out how to patch all of the usage data, some accounting program will produce a "discrepancy" comment.

8.7 Machine-room security

It is not always necessary to run the facility like Fort Knox in order to decrease the chances for security breach substantially. If the general principle of having only operators or accompanied visitors is established, if output is distributed outside the computer room and users are not allowed to hang over the operations consoles, and if users are not allowed to mess with the machine when it is down, most possibilities for problems will be eliminated.

8.8 Tape security

If users can walk off with dump tapes, or mount them for themselves, the storage system is not at all secure. Work is currently going on concerning the design of high-quality detachable media security measures.

8.9 What to to after a breach.

Once the administrator finds out that someone has used a password he shouldn't have been able to, what does he do? Partly, the answer depends on what kind of problem has occurred. Clearly, all sensitive passwords should be changed simultaneously. Date and time modified of all segments should be checked, to see what has been changed. Prints of the logs and accounting segments should be made and cross-checked. Consider a complete reload.

8.10 Password trap

The answering service maintains a "trap" flag in the PNT, in each user entry. By using the program "pass_util", a system administrator can cause this flag to be set on. Whenever the person logs in, a message will be typed on the initializer and filed in the log, remarking that the password has been used. No indication that this has happened is given to the user. This feature can be used when you suspect that a certain password has been compromised and you wish to catch the fellow red-handed. Since the terminal ID and channel number are typed in the message, it is often easy to locate the user. There is another

flag which may be set, which "locks" the password, without changing it. (As a matter of fact, the user may even change his password while it is locked.) The user will be unable to log in if his password has been locked.

The incorrect password counter mentioned in section 8.1, and the identification of the terminal from which the last bad password was issued, are also stored in the PNT.

The system administrator may also choose to set an entry in the system message table to cause a user to receive a "blast" message whenever he is logged in and someone attempts to log in with the same name, project, and password. This feature may alert the user to the fact that his password has been compromised; it is also useful in some cases of network use, since a user may log in from a remote host which then goes down without informing Multics -- if the user then attempts to log in via some other network route, the blast message sent to the first instance of the user will cause him to be logged out.

Chapter 9: Storage Quota Management

Disk storage accounting is done by a self-rescheduling absentee job, as described in chapter 5. A byproduct of this process is a list of all projects which are low on disk. The administrator responsible for disk should attempt to give enough disk to all projects to allow them to keep operating. If prices are set correctly, this will always be profitable until the time comes to expand the building housing the computer.

If sufficient quota is avalable on >udd to satisfy a project's need, all the administrator needs to do is to type

```
movequota >udd>Proj INCR
```

to increase the storage available to Proj by INCR records.

To generate more quota on the system, do a setquota on the root only either from the initializer console or via "sac", and then move the quota down to >udd, as follows:

```
sac "sq > 3000: mq >udd 1000"
```

supposing that the root quota is normally 2000.

It is permissible to have more quota allocated than is physically available. The acutal amount of overallocation depends on how fast the community uses up quota and how closely the disk quota is watched: this can be determined only by experience. Caution in this process is recommended, because the system crashes if it runs out of space.

For more discussion on this point, see section 13.1 (page 86).

Chapter 10: Crash Recovery

If the system crashes while running the crank, a flag will have been set which prevents the crank from running again until the state of the accounting segments is determined. (This flag is the variable "abort_crank" in value_seg.) The various programs in the crank have been planned so that they can be re-run without ill effect, whenever possible. The first thing to do is to examine the absout file produced by the crank, to see what the crank was doing when it was interrupted. Next, check the date and time modified on the segments in >system_control_dir and the segments in >udd>SysAdmin>admin, and confer with operations to make sure that no files were backed up due to a reload.

Usually, at this point, it is possible to use the editors to create an exec_com file which consists of the remainder of the steps in the crank, and then to execute it on-line.

In some cases, data which was lost may be regenerated: for instance, if "projfile" is lost, you should retrieve a copy, use "eppo" to add any projects which were added since the copy you have, and to edit any other entries which changed, and then re-run the disk report to recalculate the project usage. It is difficult to provide a fool-proof prescription for recovery, since what must be done depends on what segments were lost or damaged. If your investigations seem to show that all is well, type

```
value$set abort_crank false
```

to allow the crank to run again.

Do an "lar" to make sure the crank and disk report jobs are scheduled. If not, do a "setcrank" and/or "setdisk".

Chapter 11: Setting up Accounting

There are two situations of interest concerning accounting setup. The first concerns the startup of accounting at an installation which has never had a Multics before -- this is called an "accounting cold start". The second concerns the modifications necessary for an installation which receives a new distribution of Multics, with significant changes to the accounting package which may require the reformatting of segments, or other modifications, in order to continue to operate.

11.1 Accounting cold start

Accounting cold starts are supported by the exec_com segment

acct_start_up.ec

which is distributed as part of the system libraries. In general, all that the local installation need do is, on its first successful bootload, enter admin mode on the initializer and type

ec acct_start_up cold

and then follow instructions.

11.2 Accounting warm start

Although some work has been done on designing an accounting warmstart, the supplied procedures may have defects if you are upgrading more than one level of accounting system in a jump, or if you have made modifications to the supplied accounting procedures for support of local installation features. It is imperative that you study and understand the contents of acct_start_up.ec before executing them. Read both the cold and warm start sections. The warm start procedure is designed to upgrade the accounting system from the previously-distributed level to the new level: if it looks like it will work, you may type

ec acct_start_up warm

from the initializer console. As of this date, no general accounting-distribution procedure has been developed. This deficiency in the system will be corrected, module by module, as time permits. In the meantime, the best thing to do is to read the exec_com and skip those steps which look like they would

destroy data.

Pay careful note to the header of the SAT which is created by the coldstart. If your SAT header does not look like this, you may need to change the default attributes.

It is a good idea to at the very least reconvert the SAT, via "new_smf", and reinstall it. It cannot hurt to recompile all PDT's and reinstall them too. Delegated projects may be a problem here. If the format or contents of the PDT changes in such a way that this step is absolutely necessary, a conversion program will be supplied.

Examine the segments created by the coldstart. If you don't have any of them, execute that section of the coldstart which creates them.

The segment bound_admin_old_ contains programs which used to be part of the system administration subsystem, but which have been obsoleted. The programs in this bound segment are sometimes used for conversion from one level of the accounting system to the next.

11.3 Local Installation parameters

Most installation parameters used in the operation of the administrative system are kept in the segment "installation_parms". The standard accounting cold-start exec_com segment sets these parameters to the MIT values, in most cases. A discussion of each parameter follows.

11.3.1 Shifts

System usage may be divided into up to eight shifts, numbered from 0 to 7. These shifts may begin at any half-hour during a week: the current shift is the same for everyone, and the shift is the same at the same hour during every week. There is currently no provision for over-riding the regular shift mechanism to account for holidays. Shifts are set by filling in a table which contains 336 entries, one for each half-hour in the week, with a digit from 0 to 7. At MIT, we use only shifts 1, 2, 3 and 4, and the hours are:

```
shift 1   0800-1800 weekdays
shift 2   1800-2400 weekdays
shift 3   0000-0800 every day
shift 4   0800-2400 weekends
```

11.3.2 Shift prices

Prices may be set for CPU time, connect time, process time, core usage, and device usage on a per-shift basis. The current mechanism does not actually generate any charges for any but CPU and connect time. Later versions will distinguish between connect time and process hours, to allow multiple processes per user and multiple terminals per user; we will also implement core charging and device channel usage charging. Any installation may, of course, set the prices for some resources to zero and use other administrative means to control resource usage.

The calculation of the proper price for the use of a resource has been the subject of endless argument. Every local installation will have its own special way of computing these rates. The method we have evolved at MIT is the following: first, we list all costs involved in running the service. This includes programmer and operator salaries, cards and paper, hardware rental, and modems. (Desired profit or planned loss could be included -- we attempt to break even.) These costs are then attributed to cost pools, one for each resource which is chargeable. Resources which cannot be attributed to the cost of providing a particular service, such as operator salaries or the motor generator, are lumped in the category "overhead," and the cost of this pool is spread proportionally over the others. Each pool's cost is then divided by estimated paying-customer usage, to obtain a "break-even price." These figures are then fudged to make reasonable prices. If an estimate is uncertain, we round up strongly. For shift differentials, we use the following table:

```
shift 1   115% of break-even
shift 2   90%
shift 3   50%
shift 4   80%
```

We also usually round the shift prices up to the nearest multiple of $.50 per minute, just to make the prices easy to remember.

The program "price" is available to interested system administrators from T. H. Van Vleck; it mechanizes the pricing philosophy discussed above.

11.3.3 Installation ID and titles

Installation_parms contains the 32-character installation ID field which is typed out when a user dials up. This should contain both the company and department abbreviation, and the city and state. For example,

MIT, Cambridge, Mass.

The installation titles are longer character strings, returned by the program system_info_$titles, which are used at the top of each page of the monthly bills and in many other reports. There are two strings, the company name and the department ID. Both strings are entered twice, once single-spaced and once double-spaced. For example,

Massachusetts Institute of Technology
Information Processing Center

and

M a s s a c h u s e t t s ....

Locally-written administrative tools should, of course, use these titles so that they can be exported to other installations.

11.3.4 Miscellaneous parameters

The maximum time allowed for a login time, the cycle time between initializer accounting updates, and the maximum inactive time before automatic logout for a normal user are all given in seconds in installation_parms. The number of login tries a user may have before the system hangs up on him is also specified there.

The usual values for these parameters are:

inactive_time	15-90 mins.
warning_time	3-5 mins.
login_time	3-5 mins. (more if noisy lines)
acct_update	10-60 mins.
login_tries	1-4

These values may be modified according to the needs of the installation. The ranges given are reasonable limits, though. For example, the accounting update cycle should not be too small, or initializer time will go way up, and it should not be too large unless the system hardly ever crashes, or revenue will fall off. Consult the SPS writeup of the ed_installation_parms command for more information.

11.3.5 Messages

Installation_parms also contains the messages typed out on the user's console if his account is out of funds, nearly out of

funds, past its termination date, or nearly past its termination date. We put the telephone number of the User Accounts Office in these messages, to make sure the user knows who to call.

11.3.6 Daemon and absentee rates

These rates are kept in installation_parms, in an item named "queue". Absentee CPU and real-time prices are stored by queue, as are IO daemon rates per 1000 lines. Queues one through four may have rates set.

11.3.7 Absentee "timax" parameter

The scheduling priority for a Multics process is calculated by the scheduler by a complex algorithm which takes several factors into account. Consult the MPM SPS for the details of this calculation. For system-administration purposes, what is important is that each process, when it is created, is assigned a parameter called "timax", which limits the depth to which the process may sink in the scheduling queues. Interactive processes are assigned the default timax, which is currently 8000000 microseconds. The timax value for absentee processes, however, is obtained from installation_parms, and may be different for different queues. A larger number will allow a lower queue: at MIT, we set all queues to have a timax of 16000000. This means that absentee jobs sink rapidly to the bottom of the queues, where they are given relatively long CPU time limits, and that interactive users will always get better response than absentees. By setting the values associated with the item "timax", an installation may experiment with other values for the absentee timax.

11.3.8 Configuration Table

A table is kept in "installation_parms" which describes each of the system configurations which an installation will use, by the number of CPU's, number of 64K memories, and shift. For each such entry, the table contains the value of the maximum number of load units allowed on the system, the maximum number of absentee users allowed, the highest-numbered (lowest-priority) absentee queue from which jobs will be run, and two response control parameters. Unless automatic adjustment of maxunits has been disabled (see the writeup of "admin" in the SPS, and the operator command writeup of "maxu"), this table will be referenced to set maxunits whenever the system is brought up, whenever the shift changes, and whenver the system configuration changes. The configuration is looked up in the table, and maxunits and the

absentee maxima are changed.

If response control is enabled, the system will attempt to adjust maxunits on every login and logout, so that the average queue length falls between the "low" and "high" values. Maxunits will be increased whenever the average queue length is below "low"; it will be decreased whenever the queue length is above "high"; and it will be set to the current number of users whenever the queue length is between "low" and "high".

The order of the elements in the configuration table is significant. The elements should be in order by number of CPU's, within that by number of memories, and within that by shift; smaller numbers first. The lookup will continue as long as any table element is less than the value being looked up, so that an element sepcifying 9 CPU's, 100 memories, and shift 8 will always stop the lookup. This feature can be used if, for instance, your installation does not wish to have different parameters depending on shift; instead of having eight table entries for each combination of CPU and memory for shifts 0-7, you may supply only one entry, which has a shift number of 8.

11.4 Runtime constants in value_seg

The symbol-table segment value_seg is used, with the active function value, to insert certain installation-variable parameters into the operations conducted by the master exec_com. The most important of these values are

crank_time	the time of day that the crank runs
disk_time	the time of day that the disk accounting runs (this should be before the crank)
admin_addr	address, in dprint header, of system administrator
accts_addr	address, in dprint header, of system accountant (user accounts office)
directorX_addr	addresses, in dprint header, of directors (currently, up to 7 may be named)
abort_crank	"true" if crank is to abort, else "false" (see section 5)
sysa_addr	address, in dprint header, of system assurance

sysprg_addr	address, in dprint header, of system programmer
admin_online	person and project id of user to send "crank ran" message to
log_number	current number of log segment being processed

The critical values are set to reasonable values by the accounting coldstart. they may be changed with the value$set command. The command "value$dump" may be given to type out all values in value_seg. If the various addresses in the dprint header are set to "skip", the "rqm" and "rqbill" commands will skip printing of the corresponding copy of the file.

11.5 Modifications to the exec_com segments

Some installations may wish to build additional administrative tools and insert calls to them into the copies of master.ec and biller.ec which are used by the local system administrator. In particular, most installations will wish to remove the call to punch_MIT_deck from the monthly billing and insert something more appropriate to their own needs. This is easy to do, since the system-library copies of the exec-coms are copied into >udd>SysAdmin>lib. It is a good idea, though, to keep careful notes on what modifications were made to the segments and why, so that the changes can be made again after a new version of the accounting package is distributed with a new system version.

11.6 Load control group table

Consult chapter 14 (page 89) for information on the system parameters which control the load control facility. Note that setting up load control requires determining what groups are required and which projects belong in each group, and then deciding how much of the system maxunits each group should be able to occupy. These decisions require some picture of the normal load pattern of the system, as well as knowledge of the priorities attached by management to individual projects. Installations which are just starting up should probably being with just two groups, "System" for the system users such as daemon processes and other uses which are "part of the system", and "Other" for everybody else. If significant system programming work will be done on the system, a group for the system programmers -- those on the project "Multics" -- may also be necessary.

11.7 System Message Table

Answering services numbered 6.12 and higher make use of a special system facility to allow local installation replacement of all answering-service produced messages to users, without modification to the code in bound_user_control_. The unbound segment as_error_table_ may be replaced by an installation to tailor the messages typed out by Multics to local installation requirements.

The procedure for modifying a message is quite simple: a system programmer edits as_error_table_.et to modify the text of the message, compiles the segment via error_table_compiler and alm, and installs the new system message table.

Several of the entries in the message table are actually format strings for ioa_. These items may be edited, but the data-transmission items in the format must, of course, remain in the same order.

If the installation does not wish to have some of the messages typed by the system, it may choose to make the text of the messages null in the table. For example, to suppress the "Last login was..." message, the following line should replace the statement for "last_login_msg" in as_error_table_.et:

```
last_login_msg:     !!!was,:
```

Chapter 12: Special User Identities

Several special user identities are built into the organization of the Multics system, or are referred to by the documentation by name. This section describes the various special identities and their peculiarities.

12.1 Full system administrators

Full system administrators are users on the project "SysAdmin" who have the normal initial procedure, "process_overseer_". Since they have potential write access on every segment in the system, and since they may access the highly-privileged gate "hphcs_" and the privileged gate "phcs_", they can cause great damage to the system by inappropriate execution of a command.

12.2 Restricted system administrators

Restricted system administrators are users on the project "SysAdmin" who have the special initial procedure, "accounts_overseer_". This program allows them only a small set of innocuous commands, and the ability to execute any function contained in the segment "master.ec". A great deal of self-checking is built into the command sequences contained in "master", so that a non-programmer can be trained to operate as a restricted system administrator.

12.3 IO.SysDaemon

The IO daemon has control of the printers, readers, and punches on the system. Its initial procedure is "daemon_exec_". The daemon is special in that it has write access on the daemon queues contained in

>daemon_dir_dir>io_daemon_dir

so that it may remove processed requests from the queues. The IO daemon does its accounting in the segment "IO__accounting" in >system_control_dir. Except when it is in the procedure "act_ctl_" it has no access to segments in >system_control_dir, to prevent users from requesting prints of the password segment. The project "SysDaemon" and the user "IO.SysDaemon" have the "multip" attribute flag, so that the IO daemon can be logged in more than once, in order to run more than one printer.

12.4 Backup.SysDaemon

The Backup daemon operates in ring 1, instead of ring 4, so that it may dump all segments which are part of the hierarchy. It comes up in ring 1 because the statement

```
ring:    1, 1:
```

has been added to its pmf entry in "SysDaemon.pmf", and because the smf gives permission to SysDaemon to have users log in in lower rings because it has the statement

```
ring:    1, 7:
```

in the pmf entry for the project "SysDaemon". Its initial procedure is "process_overseer_". This user has special access rights to all directories on the system, through the medium of the SPACL.

12.5 Dumper.SysDaemon

The Dumper daemon is used to perform complete dumps of the hierarchy. It is otherwise just like Backup.SysDaemon.

12.6 Retriever.SysDaemon

The Retriever SysDaemon is used to retrieve user segments from complete or incremental dump tapes. It operates in ring 1, and its initial procedure is "process_overseer_". It is on the SPACL.

12.7 Translator.SysDaemon (obsolete)

The Translator daemon has been eliminated, now that absentee is a regular facility of Multics. It represented a potential security hole in the system, since it attempted to write segments in the user area, and could have ended up writing segments which were accessable to *.SysDaemon but not to the user.

12.8 Ring_1_Repair.SysDaemon

This user comes up in "process_overseer_", but in ring 1. This is the only user who can execute any command from the console in ring 1. This identity is used to fix problems with access control in ring 1. The password for this user is kept in a sealed envelope in the operations area.

12.9 Repair.SysDaemon

This user identity is provided for evergency fixes by system programmers who need SysDaemon access privileges. The password for this user is kept in a sealed envelope in the operations area.

12.10 Repair.SysAdmin

This user identity is provided for emergency fixes by system programmers who need SysAdmin access privileges. The password for this user is kept in a sealed envelope in the operations area.

12.11 Network Daemon

This user is logged in to provide the functions associated with the Network Control Process. Its "start_up.ec" segment brings up the network, and then monitors the operation of the network. The primary functions of this user are to assist in the initial connection between foreign hosts and the system, and to perform segment-transfer operations.

12.12 anonymous users

Anonymous users are all given the access control ID "anonymous" instead of a person ID. Their usage is aggregated on the monthly bill and the statistics under the heading "Student Users". The "who" command lists them as "anonymous", and the "as_who" command lists them with a star in front of the name. There is no reason why an anonymous user cannot be a project administrator.

12.13 Fictitious persons

Sometimes, a project wants to register a full user identity which corresponds to no real person. This practice will work, but should be discouraged unless a good reason can be advanced, since it tends to circumvent the access control mechanism. To add a fictitious person, type in the name of the real person in charge of the identity to the register command, preceded by an asterisk. "register" will then know that it should not attempt to generate a userid from the name, and will ask for the userid. The names beginning with asterisk will be skipped when mailing labels are generated from the urf.

12.14 Project Administrators

Project administrators are registered Multics users who are in charge of some project. They need not be registered on the project they administer. A single user may be project administrator for many projects. Each project administrator appears in the smf entry for the project he administrates. He has a copy of the project's pmf, and may install a pdt which he creates from this pmf with cv_pmf, by use of the install command. The project administrator must be able to read "proj_admin_seg" in order to use the install command, and he must have "ea" access on the directory >system_control_dir>update in order to put the copy to be installed where the system can discover it.

Project administrators should be trustworthy people. They should be told to appoint an alternate if they go on vacation, since it is difficult for a system administrator to run a project which has been delegated to someone else.

12.15 Project "Multics"

Users on the "Multics" project are able to access the privileged gate "phcs_" to investigate the contents of the supervisor. Since they can, therefore, steal passwords from the input buffers, only responsible system programmers should be registered on this project.

12.16 Terminal repair

The project "Terminals" should be set up with one anonymous user whose initial procedure is "terminals_overseer_" so that remote-terminal service personnel can login and check out a terminal. Bumping should be on, and a grace of only 5 minutes set.

12.17 System operators

All system operators should be registered on the project "Operator". This provides them access to the system for the purpose of sending mail, document runoff, and harmless play. It is far better to provide them with a small amount of disk space and some CPU time of their own, than to discover them using the daemons for experimenting with the system.

Chapter 13: Full System Administrator Operations

There are several operations necessary to keep the system running smoothly which are not yet made into commands for a restricted administrator.

13.1 Disk quota moving

The daily disk report lists all those projects which are low on disk quota. Some of these projects deserve more disk, and others do not -- they will use up all that is available. An automatic program could be written to give more disk to those in need, and to take quota from those projects which had extra; but the algorithms involved would be complicated and ad hoc, and manual adjustment and intervention would still be required.

A restricted system administrator can do some of this quota moving, since he has the movequota command available, but it takes some experience, and is hard to explain.

The general strategy is to keep some extra quota on >user_dir_dir, available for giving to new projects or to old projects which are running out of disk. Issuing the command

```
movequota >udd>Alpha 50
```

will give the project named "Alpha" 50 more records of quota. A negative amount will move quota up from a project to >udd.

The difficulty arises when the free quota on >udd runs low. There are two possibilities: if the installation has a total quota which is already as much in excess of physical capacity as should be allowed (for instance, if the disk is over 90% physically full), no more quota should be generated. The administrator must either take quota from some other project, or from some system directory, or deny the request. Knowledge of the various projects' activities and moral suasion may have to be employed. A "hold" should be put on new project registrations, too, since new_proj will gripe if it cannot christen a new project with enough disk.

If, on the other hand, the administrator wishes to raise the total quota on the system, say by 1000 records, he should issue the command

```
sac "setquota > 3000; mq >udd 1000"
```

(assuming the quota on the root is normally 2000). The "send_admin_command" (sac) command sends a message to the initializer process which will be performed as soon as the initializer gets a chance. The command line executes a "setquota" to increase the root quota, and then executes a "movequota" to move the quota down to >udd, where it can be distributed as usual.

The sequence described above is the only time that the setquota command should be used. This is because the disk-metering code in the storage system and the disk-accounting program require that disk quota form a "connected tree" with no directory which has a quota inferior to one that does not.

See Chapter 9 (page 72) for more information on disk quota management.

13.2 Cleanup of segments

Giving the restricted system administrator the "delete" command would allow him to destroy any data segment on the system. Therefore, it is up to the full system administrator to clean up segments, especially in >udd>SysAdmin>admin and its subdirectories, which are no longer needed.

13.3 Load control group management

New projects will automatically be assigned to the default load control group, "Other", by the new_proj function. Placing a project in a group requires modification of the segment "smf.new", followed by a new_smf. All that is required is to add the line

```
group:          Blah:
```

after the "projectid" statement which defines a project. Generating a new group is done by running the program "ed_mgt", which is documented in the MPM SPS.

13.4 Special Project requirements

Creating privileged projects, or under-privileged projects, or editing the pmf of a project to give it special treatment, must be done by the full system administrator. Consult the MPM SPS writeups of cv_smf and cv_pmf for more information.

13.5 Access to project directories

Access control is the particular province of the system administrator. One thing he has to do that cannot be entrusted to the restricted system administrator is to provide the various project administrators with access to project directories. In some cases, the system administrator may be called upon to give access to one user's segments to a user from some other project. It is important to have someone responsible performing this function, so that ill-advised or malicious requests can be refused.

13.6 Crash Recovery

Unrestricted system administrators may be called in to rescue the crank or the system administration subsystem if the system crashes while some accounting operation is in progress. The specific steps to take depend on the nature of the crash and on the operations being performed when the system went down.

The worst crashes are those which are followed by a reload in which some accounting files are lost. Sometimes, this may keep the system from starting up; or worse, the system may start up, but be using old or inconsistent data. Operations should have standing orders to contact a system administrator if segments in the administrative directories are destroyed.

More often, the difficulty is less catastrophic: for instance, the system may crash during a "new_proj", and if the new_proj is repeated it will turn out that a project is in smf.new twice. Restricted system administrators should be told which functions cannot be restarted; or told never to do anything more, if the system crashed while they were doing something, until the state of the accounting files can be checked.

Chapter 14: Load Control Groups

Load control groups are groupings of projects which share a guaranteed access to the system. Each load control group has a quota of "primary" load units which represents a guaranteed number of users from the group who will always be able to log in. Users in excess of the primary quota will be assigned "secondary" status and will be allowed to log in if the system is not full. Secondary users will be preempted if a primary user wants to log in and the system is full. Secondary users may be preempted to primary, and their grace recalculated, if a primary user from the group logs out.

If a group's primary quota is full and the system is full, a user from the group will not be permitted to log in unless he is permitted to preempt some primary user in his group. This preemption is under control of the project administrator, who has two parameters he may adjust. In order to preempt another user, a new user must have the "preempting" attribute in the PDT, and his project must have the attribute in the SAT. In order to be preempted, a primary user must have his "grace" expire. The "grace" is set by the project administrator in the PDT, subject to a maximum grace given in the SAT.

The attribute flags "no_primary" and "no_secondary" may also be set in the PDT or the SAT. If a project has the "no_primary" attribute, none of its users may be primary. Similarly, if a project has the "no_secondary" attribute, none of its users may be secondary. These flags may also be set by the project administrator for individual users.

The "guaranteed_login" attribute is used only for system projects. If a user has this attribute in the PDT, and if his project has the attribute in the SAT, then the user will be logged in if at all possible if he specified the "-force" argument to login. He may overload the system in the process of logging in, or may even have to bump a primary user if the system has attained its maximum number of users.

The accounting system will be set up with two groups by the accounting cold start. One group is called "System", and is for the use of the daemons and system administration -- those users vital to the operation of the system. The rest of the users on the system will be placed in the "Other" group. To define additional load control groups, use the "ed_mgt" command to edit the master group table. For example, to add a group called "SysProg" and to give it to the project called "Multics", perform the following operations from an unrestricted system

administrator:

```
ed_mgt >system_control_dir>master_group_table
a SysProg 5
w
q

edm smf.new
i projectid:      Multics:
i group:       SysProg:
w
q

ec master new_smf
```

The control parameters for a group are its primary quota, expressed as

```
constant + max_units * numerator / denominator
```

so that the number of units allocated to a group has a guarantee and a portion which varies with the number of units allowed on the system; and, optionally, an absolute maximum number of units for the group, primary or secondary. This maximum is also expressed as

```
constant1 + max_units * numerator1 / denominator1
```

The absolute maximum can be used if you prefer to see the system at less than maxunits rather than having many secondary users in some group; this strategy may provide better response for those users who do get logged in. A primary quota of "-1" for a group indicates that the group should take "all the rest" of the primary quota on the system, after those groups which have fixed quotas have been deducted from maxunits. No more than one group should have a quota of "-1", or havoc will result. The total of all primary quotas must not add up to more than maxunits.

Bibliography

The following is a list of the writeups which contain additional information about the system administration facility, its programs and data bases.

Multics System Programmer's Manual

The published documents on system and user control in section BQ are almost completely worthless. They describe programs which were never used, and in most cases never written. In some cases, one can discover design principles mentioned in these old documents which have been followed in the construction of today's facilities.

A completely new section BQ is being prepared, which will provide an overview of the system and user control facilities actually part of the system. This project is not high-priority.

Multics Programmer's Manual

See section 1.2, "Logging in", and the command writeups for "login" and "logout".

Multics System Administrator's Manual

This document.

Multics Project Administrator's Manual

Describes the duties and powers of the project administrator. A copy of this document should be given to each project administrator.

On-line help segments

See the segments "charges.info", "load_ctl.info", "login.info", and "user_control.info".

Bibliography

Multics Programmer's Manual - System Programmer's Supplement

The currently installed programs which support administration and user control are documented in this manual. The writeups to read will be listed by category.

System Control

system_startup_ - system ring 1 overseer
system_control_ - system user-ring overseer
sys_log_ - initializer error messages

User Control and Answering Service

absentee_user_manager_ - absentee job control
absentee_utility_ - absentee job scheduler & utility
act_ctl_ - accounting
admin - operator commands
as_ - answering service startup
as_meter_ - system metering
asu_ - typewriter attachment
aswa_ - typewriter and network i/o
cpg_ - process creation
datebin_ - date conversion utility
dhodh_ - access checking utility
dial_ctl_ - slave console control
dialup_ - line control, login line parse, logout
dog_ - process destruction
hash_ - hash table utility
hash_index - hashing function
lg_ctl_ - login decision, user identification
load_ctl_ - load control decision
not_ascii_ - string-checking utility
rehash_ - hash table utility
scramble_ - password scrambling
test_dialup - testing tool
up_pdt_ - pdt validation and updating
up_pnt_ - pnt validation and updating
up_sat_ - sat validation and updating
up_sysctl_ - system table update control

Administration and billing

access_name - add or delete access control names in UCT
accounts_overseer_ - initial procedure for administrator
admin_util - lock and unlock sys_admin_data
as_who - privileged who, for operator and admin.
charge_daemon_usage - IO daemon charges from deck to PDT
charge_disk - charge disk to projects
charge_projects_to_accts - monthly billing

Bibliography

MPM SPS Sections

charge_user_ - PDT modification utility
charge_user_registration - monthly billing
charge_users_to_accts - monthly billing
clear_projfile - monthly billing, cleanup
clear_reqfile - monthly billing, cleanup
compute_bill - daily billing
console_edit - utility to update termseg
console_report - generate console report by id code
copy_as_meters - copy stat_seg for statistics report
copy_pnt - make pnt copy for testing with blank passwords
create_lines_file - installation startup tool
cv_persmf - compile persmf into pnt (obsolete)
cv_pmf - compile pmf into pdt
cv_smf - compile smf into sat
daemon_acct_print - print daemon accounting deck
daily_log_process - generate reports from log
daily_summary - daily billing report
disk_stat_print - print disk charge figures
disk_usage_stat - sweep disk and print statistics
disklow - report on projects low on disk
dump_anstbl - dump answer table for debugging
ed_installation_parms - edit installation parameters
ed_mgt - edit master group table
edurf - edit user registration segment
epro - edit project segment
erf - edit requisition segment
fill_urf - pass through urf, get missing data
format_attributes_ - output format utility
get_password_ - password reading utility
get_uid_with_lastname - search urf for last name
get_user_ - monthly billing subroutine
hash_table - hash table maintenance
idsort_ - fast sorting utility
install - install system tables
is_he_user - active function
is_legal_proj - active function
labl1 - print billing labels
mailing_labels - print user labels
merg - monthly billing
merge_urf_pnt - obsolete
misc - input miscellaneous charges
new_user - register or change user
pdt_copy - copy all PDT's to safe directory
ppro - print projfile
print_disk - print project disk charges
print_log - print log segment
print_meters - create statistics report
print_pdt - print pdt

Bibliography

MPM SPS Sections

print_pnt - print pnt
print_sat - print sat
process_daemon_cards - daily billing
process_session_cards - daily billing
proj_mtd - month-to-date project report
proj_usage_report - month-to-date project report
punch_MIT_deck - monthly billing
qfp - print reqfile
remove_user - unregister user
rename_pnts_ - utility for new_user
reset_disk_meters - monthly reset of disk meters
reset_processed_flag - patch accounting deck flags
reset_usage - reset usage data in PDT
reset_use_totals - reset usage totals for reports
send_admin_command - send command to initializer
set_admin_data_ - locking subroutine for sys_admin_data
sort_hist_file - sort billing data base
sort_projfile - sort project database
sort_reqfile - sort requisition data base
sweep - disk quota and charge accounting
sweep_disk_ - disk hierarchy sweep
sys_full_report - report on refused logins
system_daily_report - write daily report
system_monthly_report - write monthly report
system_total - compute system availability for reports
terminals_overseer_ - terminal repair init proc
up_ctr - update charges this requisition, monthly
urfp - print urf
urfsort - sort urf
usage_total - compute system usage for reports
value - active function and command
write_acct_bill - monthly billing
write_billing_summary - monthly billing
write_user_usage_report - monthly billing

Data Bases

answer_table - answering service user database
communications - system control password
disk_stat - disk charge figures
installation_parms - installation parameters
IO__accounting - daemon accounting deck
log - answering service log
miscfile - miscellaneous charges journal
mgt - master group table
pdt - project definition table
pit - process initialization table
pnt - person name table

Bibliography

MPM SPS Sections

proj_admin_seg - system table installation
projfile - project segment
reqfile - requisition segment
sat - system administrator's table
stat_seg - answering service statistics
sys_admin_data - system administration lock
urf - user registration segment
use_totals - report data base
whotab - public list of users logged in

master.ec - system administrator commands
biller.ec - billing operations
err.ec - administration errors
util.ec - administration utility
acct_start_up.ec - installation accounting startup

Appendix 1: Sample forms

Generating a few workflow forms for the use of the user accounting office can simplify the operation noticeably. The forms will enable you to make sure that the necessary information is captured. Each installation will have its own peculiar requirements for these forms, so no general scheme is provided. Samples of the forms in use at MIT are attached.

You will probably want the following:

person registration form

project registration form

project deletion form

tape registration form

retrieval request form

system problem/complaint form

credit request form

card processing request form

configuration change log

Roger says

user accounts all-purpose letter

user tape log

miscellaneous charge input

crash form

backup tape log

Massachusetts Institute of Technology

Information Processing Center

Person Registration for Multics System

This form is only needed for people who have never been registered on Multics before.

Name _____
Last First Middle

Mailing address _____

Telephone _____

Programmer number _____ (if any)

Each person registered on the Multics system is assigned a 1-24 character "Person ID", which is unique at this installation. The person ID is usually the last name (beginning with a capital letter) if possible: if someone else with the same name is already registered, the person ID will be the last name with initials prefixed (e.g. Smith, JSmith, JRSmith).

Default project ID _____

Registered persons must also be authorized to use a particular project by the project administrator for the project.

Please attach a slip of paper with a personal password. Your password may be 1-8 characters (letters and digits only).

You may change your default project ID or your password on-line any time you wish: consult the Multics Programmer's Manual.

Please return this form to:

Information Processing Center
User Accounts Office, room 39-513
M.I.T., 77 Massachusetts Avenue
Cambridge, Massachusetts 02139

If you have any problems with your registration, please contact the User Accounts Office (253-4118).

Name Assigned _____ Date _____

by _____

form revised 9/72

Massachusetts Institute of Technology

Information Processing Center

Project Registration for Multics System

Project Title _____

_____

Principal Investigator _____

Address _____

Project Supervisor _____

Address _____

Phone _____

Each project is assigned a "Project ID" (1-9 characters beginning with a capital letter) for identification and access control purposes. Please suggest a project ID for your project:

_____

Initial disk quota _____ records (default 25)

MIT acct _____ Requisition or P.O. _____

If you wish to administer your project on-line, please supply the following information:

Directory for PMF _____

Administrator (Person.Project) _____

Project ID Assigned _____ Date _____

by _____

form revised 9/72

Massachusetts Institute of Technology

Information Processing Center

Multics Project Administration Request

TO: User Accounts Office, MIT 39-513

FROM: _____

DATE: _____

SUBJECT: Project ID _____

[ ] Please delete the project and all its disk storage. Scratch all tapes and release all lockers associated with the project.

[ ] Please add the following persons to the project. Person registration forms are attached for any persons not previously registered.

_____ _____ _____

_____ _____ _____

_____ _____ _____

[ ] Please delete the following persons from the project. Do not delete their directories.

_____ _____ _____

_____ _____ _____

_____ _____ _____

[ ] Please give project administrator status to the following user (i.e. delegate the project).

Directory for PMF _____

Administrator (Person.Project) _____

Date done _____ by _____

form revised 9/72

Massachusetts Institute of Technology

Information Processing Center

Multics Project Administration Request Confirmation

TO: (Project supervisor) _____

FROM: User Accounts Office, MIT 39-513: 253-4118

DATE: _____

SUBJECT: Project ID _____

[ ] Your project has been set up on Multics as requested.

[ ] Your project has been deleted as requested.

[ ] The following users have been added to your project.

_____ _____ _____

_____ _____ _____

_____ _____ _____

[ ] The following users have been deleted from your project. You should delete their directories once you have copied any segments which you wish to preserve.

_____ _____ _____

_____ _____ _____

_____ _____ _____

[ ] Your project has been delegated to a project administrator. He may now add and delete users and control resource limits.

Directory for PMF _____

Administrator (Person.Project) _____

Date done _____ by _____

form revised 9/72

HONEYWELL INFORMATION SYSTEMS

Publications Remarks Form*

TITLE: MULTICS SYSTEM ADMINISTRATOR'S MANUAL PRELIMINARY EDITION

ORDER No.: AK50, REV. 0

DATED: FEBRUARY 1973

ERRORS IN PUBLICATION:

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION:

(Please Print)

FROM: NAME _____ DATE: _____

COMPANY _____

TITLE _____

_____

_____

CUT ALONG LINE

*Your comments will be promptly investigated by appropriate technical personnel, action will be taken as required, and you will receive a written reply. If you do not require a written reply, please check here. ☐

CUT ALONG

FOLD ALONG LINE

FIRST CLASS
PERMIT NO. 39531
WELLESLEY HILLS,
MASS. 02181

Business Reply Mail
Postage Stamp Not Necessary if Mailed in the United States

POSTAGE WILL BE PAID BY:

HONEYWELL INFORMATION SYSTEMS
60 WALNUT STREET
WELLESLEY HILLS, MASS. 02181

ATTN: PUBLICATIONS, MS 050

FOLD ALONG LINE

Honeywell

The Other Computer Company:

Honeywell

HONEYWELL INFORMATION SYSTEMS

6811
5C273
Printed in U.S.A.

In the U.S.A.: 200 Smith Street, MS 061, Waltham, Massachusetts 02154
In Canada: 2025 Sheppard Avenue East, Willowdale, Ontario

AK50, Rev. 0