68xxx UniFLEX®
Utility Commands

MANUAL REVISION HISTORY

Revision	Date	Change
A	12/84	Original Release, 68000 UniFLEX Utility Commands
B	02/86	Manual Update for 68xxx. Additional commands and miscellaneous corrections.
C	08/86	Manual Update for Version 2.0 of 68xxx UniFLEX. New commands: kermit, newuser, su Revised software: cc, devcheck, dir, headset, load68k, login, move, rel20, rel68k, remove, set_termcap, shell Revised text only: at, badblocks, diskrepair, dperm, echo, env, int, makdev, password, perms, ttyset, tune

DISCLAIMER

The supplied software is intended for use only as described in this manual. Use of undocumented features or parameters may cause unpredictable results for which Technical Systems Consultants, Inc. cannot assume responsibility. Although every effort has been made to make the supplied software and its documentation as accurate and functional as possible, Technical Systems Consultants, Inc. will not assume responsibility for any damages incurred or generated by such material. Technical Systems Consultants, Inc. reserves the right to make changes in such material at any time without notice.

Preface

The UniFLEX® Operating System supports a wide variety of commands. Some commands are for the general manipulation of files--for example, creating files, deleting them, moving them about the system, and listing their contents. Other commands serve to create and control tasks. Still others perform general system maintenance.

This document provides a detailed description of each of the commands available with the operating system.

Preface

Syntax Conventions

The following conventions are used in syntax statements throughout this manual.

1. Items that are not enclosed in angle brackets, '<' and '>', or square brackets, '[' and ']', are "keywords" and should be typed as shown.

2. Angle brackets, '<' and '>', enclose descriptions which the user must replace with a specific argument. Expressions enclosed only in angle brackets are essential parts of the command line. For example, in the command

   addusr <user_name>

   the name of a user must be specified in the place indicated by <user_name>.

3. Square brackets, '[' and ']', indicate optional items. These items may be omitted if their effect is not desired.

4. The underscore character, '_', is used to link separate words that describe one term, such as "user" and "name".

5. Characters other than spaces that are not enclosed in angle brackets or square brackets must appear in the command line as they appear in the syntax statement.

6. If the word "list" appears as part of a term in a syntax statement, that term consists of one or more of the elements described by the rest of the term, separated by spaces. For example, the term

   <user_name_list>

   represents a list of user names.

7. Some utilities support optional features, known as options, which alter the effect of the command. An option usually consists of either a single character or a single character, followed by an equals sign, '=', followed by an argument. An "option string" is a plus sign followed by one or more options. An option string may contain any number of single-character options but only one option which takes an argument. An option requiring an argument must be the last option in an option string. Thus, the command line must contain a separate option string for each option requiring an argument. It may or may not contain a separate option string for each single-character option.

(continued)

The following are valid option strings:

```
+abcdefg
+abc=<arg>
```

The following are not valid option strings:

```
+abc=<arg>de
+a=<arg>b=<arg>
```

Unless specifically stated in the documentation about a particular command, option strings may appear anywhere on the command line after the command name.

Many common terms appear (often as abbreviations) in more than one syntax statement. The manual does not explain these terms each time they appear. However, the following table describes each one.

Table 1. Common Terms Used in Syntax Statements.

Term	Meaning
arg	Argument
char	Character
dev	Device
dir	Directory
file_name	A valid file name
perms	Permissions
splr	Spooler
str	String

addpath	Add a name to the list of directories the shell program searches when looking for an executable file.
addusr	Add a new user to the system.
alterpage	Alter the size of the paging space on a disk.
at	Submit commands for execution at a later date and time.
atexecute	Prepare the "at" subsystem for execution and initiate the "at daemon" that handles the lists of commands submitted by the "at" command.
atinfo	Display the current status of the "at" subsystem.
backup	Backup or restore files.
badblocks	Add the specified block to the file of bad blocks.
bcompare	Compare two files byte by byte.
blockcheck	Check the integrity of the allocation of all blocks used in files and of the free list on the specified device.
chd	Change the user's working directory.
compare	Compare two text files line by line and report the differences.
copy	Copy a directory or a file to the specified destination.
crdir	Create a directory.
create	Create an empty file for each file name on the command line.
crt_termcap	Create a file defining the capabilities of each terminal on the system.
date	Display or set the time and date.
delusr	Remove a user from the system.
devcheck	Check a device for I/O errors.
dir	List either the contents of a directory or information about a file.
diskrepair	Check and, optionally, repair inconsistencies in the logical structure of a disk.
dperm	Set the default permissions for the creation of files by the current shell program or by tasks generated by the current shell program.
dump	Send both a hexadecimal and an ASCII listing of a file to standard output.
echo	Write the arguments on the command line to standard output.
edit	Invoke the text editor in order to create a new text file or edit an existing one.
end	Stop the job currently being printed by the specified printer program.
env	Display or change environment variables.
fdncheck	Check the integrity of the structure of the file descriptor nodes (fdns) on the specified disk.
filetype	Attempt to identify the type of the specified file.
find	Search for a string in a file or in standard input.
format	Format a disk for system use.
free	Report the amount of free and used space on the specified device.

(continued)

hangup	Specify the action that the shell program is to take when it receives a hangup interrupt.
headset	Change information in the binary header of an executable file.
help	Display a brief description of the use and syntax of the specified command.
history	Display the details of recent activity on the system.
idle	Idle the specified printer program when it completes the current print job.
info	Display the contents of the information field associated with the specified binary file.
insp	Activate a printer spooler for the specified device.
int	Send a program interrupt to another task.
jobs	Report the task IDs and starting times of all background tasks originated by the user from the current shell program.
kermit	Transfer a file from one machine to another.
kill	Delete the specified file name from the file system.
lib-gen68k	Create a new library or update an existing one.
libinfo	Display information about a library.
link	Establish a new link to an existing file.
list	Write the contents of the specified file to standard output.
load68k	The "load68k" command is the 68xxx UniFLEX linking-loader.
log	Terminate the current shell program.
login	Give a new user access to the operating system and establish the standard environment for the current shell program.
ls	List either the contents of a directory or information about a file.
mail	Send mail to someone else or display any mail belonging to the user.
makdev	Create a special type of file, representing a device.
more	Display ASCII data with user control.
mount	Insert a block device at a node of the directory tree structure or display the mount table.
move	Move a file from one place to another.
newuser	Temporarily log in as a new user.
next	Restart an idled printer program.
nice	Lower the priority of the specified command.
owner	Change the owner of a file.
page	Format a file in pages.
password	Set or change a user's password.
path	Write the path name of the working directory to standard output.
perms	Change the permissions associated with a file.
print	Send a file to the specified printer spooler.
prompt	Define the prompt and reprompt strings issued by the shell program.
pstop	Deactivate the specified printer spooler.

(continued)

purge	Delete a file from the specified printer-spooler queue.
qdb	The "qdb" command is a machine-language debugging system.
relinfo	Display information about an executable or relocatable file.
rel20	The "rel20" command is the 68020 relocating assembler.
rel68k	The "rel68k" command is the 68000/68010 relocating assembler.
remove	Remove the specified file name from the file system.
rename	Change the name of the specified file.
rerun	End the current print job, return the file to the print queue, and idle the specified printer program.
resume	Resume execution of a suspended task.
setpath	Display or redefine the names of the directories that the shell program searches when looking for an executable file.
set_termcap	Interactively specify the capabilities of each terminal on the system.
shell	The shell program is a command interpreter which is the primary interface between the user and the operating system.
shutup	Take the system from multi-user mode to single-user mode.
status	Write to standard output a report on the status of all tasks belonging to the user.
stop	Bring the system to a halt.
strip	Remove the symbol table from an executable binary file.
su	Temporarily log in as a new user without changing the working directory.
suspend	Suspend the execution of a task.
tail	Write the end of the specified file.
time	Execute the specified command and report to standard error the amount of time required.
touch	Set the time of the last modification of a file to the current date and time.
ttyset	Examine or adjust the parameters associated with the user's terminal.
tune	Change the specified parameters in a file containing a copy of the UniFLEX Operating System.
unmount	Unmount a previously mounted device from the file system.
update_all	Process a set of files, performing the specified operation on each file if it is newer than the file it is compared to.
wait	Wait for a background task to complete before accepting any more input.

```
addpath <dir_name_list>

/etc/addusr <user_name>

alterpage <dev_name> [+pqv]

at <when> [+bdmrwx]

/etc/atexecute [+krs]

atinfo [+hp]

backup <dev_name> [<file_name_list>] [+aAbBCdDelLnNpqrRtTVzZ]

/etc/badblocks <dev_name> <address_list> [+dmpqsv]

bcompare <file_name_1> <file_name_2> [+es]

/etc/blockcheck <dev_name>

chd [<dir_name>]

compare <file_name_1> <file_name_2> [+<window_size>]

copy [<source_name_list>] <dest_file_name> [+bBcdDFlLnopPtTzZ]

crdir <dir_name_list>

create <file_name_list>

/etc/crt_termcap <ttycap_file> <ttyassoc_file> <termcap_file>
/etc/crt_termcap +d

date [ [<mm>-<dd>-<yy>] <hr>:<min>[:<sec>] ]

/etc/delusr <user_name> [+x]

/etc/devcheck <dev_name_list> [+bcdDfrsvV]

dir [<file_name_list>] [+abdflrsSt]

/etc/diskrepair <dev_name_list> [+abfmnpqruvz]

dperm [<perms_list>]

dump <file_name> [+i]
dump <file_name_list>

echo [<arg_list>] [+l]
```

```
edit [<file_name_1> [<file_name_2>]] [+bny]

end <splr_name>

env [<param_list>]

/etc/fdncheck <dev_name>

filetype <file_name_list>

find [+bcnsu] <str_1>[&<str_2>] [<file_name_list>]

/etc/format<xx> [+BCdfF1LmMnPqrsv]

free <dev_name_list> [+d]

hangup <on_or_off>

headset <file_name_list> [+aAbBcCdfIStXZ]

help [<command_name_list>]

history [<file_name>]

idle <splr_name>

info <file_name_list>

/etc/insp <splr_name> [+f]

int <task_ID_list> [+<int_num>] [+s]

jobs

kermit cl[te] <dev_name> <esc_char>
kermit r[l][dit] <dev_name>
kermit s[l][dit] <dev_name>

kill <file_name_list> [+dlpqs]

lib-gen68k o=<old_lib> n=<new_lib> [u=<update>] [<del_list>] [+al]

libinfo <library_name_list> [+emM]

link <file_name_1> <file_name_2>

list [<file_name_list>] [+l<num>]

load68k <file_name_list> [+aAbBcCdDefFiIlLmMnNoPqrRsStTuUwWxXyYZ]
```

```
log

login <user_name>

ls [<file_name_list>] [+abdflrsSt]

mail [<file_name_list>]

/etc/makdev <file_name> <dev_type> <maj_dev_num> <min_dev_num>

more [<file_name_list>]

/etc/mount [<dev_name> <dir_name> [r]]

move <file_name_1> <file_name_2> [+klps]
move <file_name_list> <dir_name> [+klps]

newuser [<user_name>]

next <splr_name>

nice <command_name>

owner <new_owner> <file_name_list>

page [<file_name_list>] [+fl<num>p]

password [<user_name>]

path

perms <perms_list> <file_name_list>

<print> [<file_name_list>] [+m]

prompt <prompt_str> [<reprompt_str>]

pstop <splr_name>

purge <splr_name> <file_name_list>

qdb [<image_file_name>]

relinfo <file_name_list> [+ehrs]

rel20 <file_name_list> [<param_list>] [+abefFiIJlLnosSu]

rel68k <file_name_list> [<param_list>] [+befFiIJlLnosStu]

remove <file_name_list> [+dklpqw]
```

```
rename <file_name_1> <file_name_2>

rerun <splr_name>

resume <task_ID>

setpath [<dir_name_list>]

/etc/set_termcap <ttycap_file> <ttyassoc_file> <termcap_file)
/etc/set_termcap +d

shell [+abclnvx] [<argument_list>]

/etc/shutup [[-]<minutes>]

status [+alswx]

/etc/stop [[-]<minutes>]

strip <file_name_list>

su [<user_name>]

suspend <task_ID>

tail <file_name> [<count>]

time <command_name>

touch <file_name_list>

ttyset [<param_list>] [+]

/etc/tune <file_name> [<param_list>]
/etc/tune <file_name> [+r]

/etc/unmount <dev_name>

update_all [<make_file_name>] [+q]
update_all <make_file_name> [<arg_list>] [+q]

wait [<task_ID>]
wait any
```

addpath

Add a name to the list of directories the shell program searches when looking for an executable file.

SYNTAX

addpath <dir_name_list>

DESCRIPTION

The "addpath" command, which is part of the shell program, adds a name to the end of the list of directories the shell program searches when looking for an executable file. This list, which is known as the search path, is searched sequentially. By default, the search path consists of the following directories: the user's working directory, "<home_dir>/bin", "/bin", and "/usr/bin". (The home directory is the user's login directory, as specified in the password file.) If the user is the system manager, the search path also includes the file "/etc", which is searched after "<home_dir>/bin" and before "/bin".

Arguments

<dir_name_list>	A list of the names to add to the search path. The names are added to the end of the list in the order the user specifies them on the command line.

EXAMPLES

1. addpath /usr/games
2. addpath .. bin

The first example adds the name "/usr/games" to the end of the list of directories to search.

The second example adds the parent directory of the user's working directory and the directory "bin" in the user's working directory to the end of the list of directories to search.

NOTES

- The "addpath" command is only effective while the shell program under which it is invoked is running. The list of directories searched by the login shell can be permanently altered by placing the appropriate command in the file ".startup" in the user's home directory. The shell program automatically executes this file each time the user logs in.

SEE ALSO

setpath
shell

addusr

Add a new user to the system.

SYNTAX

/etc/addusr <user_name>

DESCRIPTION

The "addusr" command is used to add a new user to the system. The specified user name must be unique to the system. It must be between one and eight letters long. All letters must be lowercase. Only the system manager may use this command.

The "addusr" command performs the following tasks:

1. Adds the new name to the bottom of the password file, "/etc/log/password".

2. Assigns a user ID to the user.

3. Creates a home directory owned by the new user with "rwxr-x" permissions. If the directory "/usr" exists, the name of the home directory is

   /usr/<user_name>

   Otherwise, it is simply

   /<user_name>

4. Puts a file named ".mail" in the user's home directory.

The system manager or the new user should use the "password" command to ensure protection of the new user's personal files.

Arguments

<user_name> A unique name assigned to the new user for use in response to the login prompt.

EXAMPLES

1. /etc/addusr chris

This example adds the user name "chris" to the bottom of the file "/etc/log/password", assigns a user ID, and creates the directory "/usr/chris" or "/chris"--which is owned by "chris" and has permissions "rwxr-x".

ERROR MESSAGES

Error assigning owner to "<dir_name>": <reason>
The operating system returned an error when "addusr" tried to make the specified user the owner of "<dir_name>". This message is followed by an interpretation of the error returned by the operating system.

Error assigning owner to ".mail": <reason>
The operating system returned an error when "addusr" tried to make the specified user the owner of the file ".mail". This message is followed by an interpretation of the error returned by the operating system.

Error creating "<dir_name>": <reason>
The operating system returned an error when "addusr" tried to create a home directory for the new user. This message is followed by an interpretation of the error returned by the operating system.

Error creating ".mail": <reason>
The operating system returned an error when "addusr" tried to create the file ".mail". This message is followed by an interpretation of the error returned by the operating system.

Error creating "." file: <reason>
The operating system returned an error when "addusr" tried to create the file ".". This message is followed by an interpretation of the error returned by the operating system.

Error creating ".." file: <reason>
The operating system returned an error when "addusr" tried to create the file "..". This message is followed by an interpretation of the error returned by the operating system.

Error locking password file: <reason>
The operating system returned an error when "addusr" tried to lock the password file. This message is followed by an interpretation of the error returned by the operating system.

Error opening "/etc/log/password": <reason>
The operating system returned an error when "addusr" tried open the password file. This message is followed by an interpretation of the error returned by the operating system.

Name must be 1 to 8 lowercase letters.
The specified user name must be between one and eight letters long. All letters must be lowercase.

Password file is locked. Try again later.
The commands "addusr", "delusr", and "password" all lock the password file so that two people cannot try to alter it at the same time. This message indicates that one of these commands currently has the password file locked.

Syntax: /etc/addusr <user_name>
The "addusr" command expects exactly one argument. This message indicates that the argument count is wrong.

The name "<user_name>" is already in use.
The specified user name must be unique to the system.

You must be system manager to run "addusr".
Only the system manager may execute the "addusr" command.

SEE ALSO

delusr
mail
password
perms

alterpage

Alter the size of the paging space on a disk.

SYNTAX

alterpage <dev_name> [+pqv]

DESCRIPTION

The "alterpage" command alters the size of the paging space on the disk associated with the specified device. The command prompts the user to specify the new size as an increase or decrease compared to the current size or as the total amount of paging space. The user may specify the size as either blocks or kilobytes. The program prompts for all necessary information as it runs (see MESSAGES).

When "alterpage" completes its task, it invokes the "diskrepair" command in order to correct the logical inconsistencies introduced by changing the size of the paging space. If the disk being altered is associated with the root device, "alterpage" suspends all tasks while it is running and, after altering the disk and running "diskrepair", intentionally stops the system. The user must then reboot the operating system.

Arguments

<dev_name> The name of the block device associated with the disk to alter.

Options Available

p Tell "diskrepair" to run with prompts.
q Tell "diskrepair" to run in quiet mode.
v Tell "diskrepair" to run in verbose mode.

EXAMPLES

1. alterpage /dev/fd0 +p

This example causes "alterpage" to alter the paging size on the disk associated with device "/dev/fd0". When "alterpage" finishes, it calls "diskrepair", which, because the 'p' option was specified, prompts the user for permission before making changes which require the deletion of either files or directories.

NOTES

- If the user invokes "alterpage" on a disk in a mounted device, the command unmounts the disk before altering the size of its paging space. The device remains unmounted after the command terminates.

- Paging space must be contiguous. Therefore, the extent to which "alterpage" can extend the paging space depends on the way in which the file system uses the volume space, which immediately precedes the paging space. The "alterpage" command cannot extend the paging space into volume space that is used by the file system. A user who needs more paging space than "alterpage" can provide must reformat the disk.

- The "alterpage" command always alters the size of the paging space by or sets the size of the paging space to a multiple of 8 blocks. If the user specifies an amount that is not a multiple of 8, "alterpage" uses the largest multiple of 8 that is less than the number actually specified.

MESSAGES

Altering size of paging space on device "<dev_name>".
The "alterpage" command prints this message before prompting the user for the change to make.

Altering size of paging space on root device.
System will stop when complete. Continue?
This message warns the user that the device being altered is the root device and that the system will intentionally shut down after the alteration is complete. The user should respond with a 'y' for "yes" to continue; with an 'n' for "no" to prevent the change.

Can increase size by <num_1> blocks (<num_2>K) to a maximum of <num_3> blocks (<num_4>K).
The "alterpage" command calculates and reports both the maximum possible increase in the size of the paging space and the maximum possible size.

Checking and repairing logical structure of the disk.
The program "diskrepair" is now rebuilding the disk.

Current size of paging space is <num_1> blocks (<num_2>K).
This message shows the size of the paging space in both blocks and kilobytes.

Decrease size by <num_1> blocks (<num_2>K)?
The "alterpage" command issues this prompt before decreasing the size of the paging space in order to give the user a chance to verify the choice. The user should respond with a 'y' for "yes"; with an 'n' for "no".

Enter amount to decrease size (blocks or kilobytesK):
If the user opts to decrease the size of the paging space, the "alterpage" command prompts for the amount by which to decrease it. If the value specified is a pure number, "alterpage" assumes that it represents a number of blocks. If the value is a number with the letter 'K' or 'k' appended to it, "alterpage" assumes that it represents a number of kilobytes.

Enter amount to increase size (blocks or kilobytesK):
If the user opts to increase the size of the paging space, the "alterpage" command prompts for the amount by which to increase it. If the value specified is a pure number, "alterpage" assumes that it represents a number of blocks. If the value is a number with the letter 'K' or 'k' appended to it, "alterpage" assumes that it represents a number of kilobytes.

Enter desired size of paging space (blocks or kilobytesK):
If the user opts to specify the size of the paging space, the "alterpage" command prompts for the desired size. If the value specified is a pure number, "alterpage" assumes that it represents a number of blocks. If the value is a number with the letter 'K' or 'k' appended to it, "alterpage" assumes that it represents a number of kilobytes.

Increase size by <num_1> blocks (<num_2>K)?
The "alterpage" command issues this prompt before increasing the size of the paging space in order to give the user a chance to verify the choice. The user should respond with a 'y' for "yes"; with an 'n' for "no".

Lowest unused block is block number <num>.
This message reports the number of the first unused block in the contiguous stretch of unused blocks adjacent to the paging space. The "alterpage" command cannot extend the paging space past this block.

Maximum size is limited to <num_1> blocks (<num_2>K).
The "alterpage" command calculates and reports the maximum possible size of the paging space.

New size is <num_1> blocks (<num_2>K).
The "alterpage" command calculates and reports the size of the paging space after making the specified change.

No ".badblocks" file on "<dev_name>".
Normally the "alterpage" command reads the bad-blocks file so that it can avoid placing blocks containing I/O errors in the paging space. This message is a warning to the user that the disk being altered does not contain a bad-blocks file.

Paging space starts at block <num>.
The "alterpage" command determines and reports the address of the first block of the paging space.

Reading disk.
The program is reading the disk to determine its structure. This process may be time-consuming, depending on the size of the disk.

Select 'i', 'd', 's', or 'e' for increase, decrease, specify, or exit:
This prompt is the first in the series of interactive prompts that allows the user to change the paging space.

Set size of paging space to <num_1> blocks (<num_2>K)?
The "alterpage" command issues this prompt before setting the size of the paging space in order to give the user a chance to verify the choice. The user should respond with a 'y' for "yes"; with an 'n' for "no".

Suspending all tasks.
The program is altering the paging space of the root device and must suspend all tasks before proceeding.

ERROR MESSAGES

Cannot alter paging space on a "backup" disk.
The disk in the specified device was created by the "backup" command and cannot be altered by "alterpage".

Cannot call "diskrepair".
The operating system returned an error when "alterpage" tried to invoke the "diskrepair" command. Most likely "diskrepair" is missing from the system or the user does not have permission to run it.

Cannot decrease a size of 0.
The disk does not contain any paging space.

Cannot increase size beyond <num_1> blocks (<num_2>K).
The user tried to increase the size of the paging space beyond the maximum possible size.

Cannot increase the size of the paging space.
The paging space already occupies as much room on the disk as it possibly can. A user who needs more paging space must reformat the disk.

Cannot suspend running tasks: <reason>
The "alterpage" command must suspend all running tasks when it runs on the root device of a system, but it could not do so. This message is followed by an interpretation of the error returned by the operating system.

"<dev_name>" has contiguous-file space below the paging space.
You must reformat the disk.
On a structurally sound disk the paging space always precedes the contiguous file space. The "alterpage" command cannot function if the contiguous file space precedes the paging space. The only way to correct this problem is to reformat the disk.

"<dev_name>" is not a block device.
The device specified must be a block device.

"diskrepair" terminated abnormally (status = <num>).
The "diskrepair" command received a program interrupt from the operating system. The termination status indicates what kind of interrupt it was. The user should try to invoke "diskrepair" on its own.

Error getting status of the root directory: <reason>
The operating system returned an error when "alterpage" tried to get the status of the root directory. This message is followed by an interpretation of the error returned by the operating system.

Error opening "<dev_name>": <reason>
The operating system returned an error when "alterpage" tried to open the specified device. This message is followed by an interpretation of the error returned by the operating system.

Error reading free list: <reason>
The operating system returned an error when "alterpage" tried to read the free list of the specified device. This message is followed by an interpretation of the error returned by the operating system.

Error reading SIR on device: <reason>
The operating system returned an error when "alterpage" tried to read the system information record (SIR) of the specified device. This message is followed by an interpretation of the error returned by the operating system.

Error seeking on "<dev_name>": <reason>
The operating system returned an error when "alterpage" tried to seek on the specified device. This message is followed by an interpretation of the error returned by the operating system.

Error unmounting device: <reason>
The operating system returned an error when "alterpage" tried to unmount the specified device. This message is followed by an interpretation of the error returned by the operating system.

Error writing new SIR to device: <reason>
The operating system returned an error when "alterpage" tried to write the new system information record (SIR) to the specified device. This message is followed by an interpretation of the error returned by the operating system.

Invalid option: '<char>'.
The option specified by <char> is not a valid option to the "alterpage" command.

No device specified.
The user invoked the "alterpage" command with one or more options but did not specify a device.

Out-of-range block found. Run "diskrepair".
The "alterpage" command encountered an out-of-range block while working on the disk. The user should invoke the "diskrepair" command to correct this problem.

Syntax: alterpage <dev_name> [+pqv]
The "alterpage" command expects exactly one argument, which may be accompanied by a list of options. This message indicates that the user specified neither a device nor any options.

The size can only be decreased by <num_1> blocks (<num_2>K).
The "alterpage" command cannot decrease the size of the paging space by the amount requested.

The size can only be increased by <num_1> blocks (<num_2>K).
The "alterpage" command cannot increase the size of the paging space by the amount requested.

You may specify only one device.
The "alterpage" command expects exactly one argument. This message indicates that the user specified more than one.

You must be system manager to run "alterpage".
Only the system manager may execute the "alterpage" command.

SEE ALSO

diskrepair

at

Submit commands for execution at a later date and time.

SYNTAX

at <when> [+bdmrwx]

DESCRIPTION

The "at" command submits commands read from standard input for execution at a later date and time, which are specified by the <when> argument. If standard input is a terminal, the prompt "at>> " is issued to standard error, requesting another line. To stop entering lines, the user must type the end-of-file character (control-D) as the first character of a line.

Arguments

<when> The date and time to execute the commands.

Format for Arguments

```
<when>   [<time>]
         [<time>] <date>
         [<time>] <day>
         now
```

The <time> parameter may be of the form <hh>:[<mm>] where <hh> is the hour number (0 through 23 inclusive), and <mm> is the minute number (0 through 59 inclusive, 0 by default). A twenty-four hour clock is assumed unless the user appends "am" or "pm" (or "AM" or "PM") to the <time> parameter (in which case the hour number <hh> must be between 1 and 12 inclusive). The <time> parameter may also be a keyword which describes the time of day (see Keywords). If the <time> parameter is omitted, 00:00 (midnight) is assumed.

The <date> parameter may be of the form [<mm>/]<dd> or <dd>[.<mm>] or [<mm>-]<dd> where <mm> is a number representing the month of the year (1 through 12 inclusive) and <dd> is a number representing the day of the month (1 through 31 inclusive, see NOTES). If the month <mm> is omitted, it defaults to the next month if both the day of the month <dd> and time <time> have passed in the current month or to the current month if they have not. The <date> parameter may also be of the form <month> <dd> or <dd> <month> where <month> is a keyword describing a month of the year (see Keywords), and <dd> is a number between 1 and 31 inclusive.

The <day> parameter is a keyword describing a day of the week (see Keywords). The keyword "now" requests execution as soon as possible.

Keywords

A keyword is recognized by any sequence of adjacent characters, beginning with the first character, that is unique to that keyword. For example, "su", "sun", "sund", "sunda", and "sunday" are all recognized as the keyword "sunday". However, "s" is not, since two other keywords, "september" and "saturday", also start with that sequence.

Keywords known to "at" are

Months	Weekdays	Time of Day	Miscellaneous
january	sunday	midnight	now
february	monday	noon	
march	tuesday		
april	wednesday		
may	thursday		
june	friday		
july	saturday		
august			
september			
october			
november			
december			

Options Available

b	When the specified time arrives and execution of the list of commands begins, do not wait for its completion before starting to execute commands queued by other "at" calls.
d	If the list of commands has expired (i.e., the "atexecute" command discovers that over an hour has passed since the requested execution time), tell "atexecute" to execute the commands at the first opportunity instead of deleting them.
m	Mail messages to the user telling at what times execution of the list of commands began and ended.
r	Resubmit the commands on successful completion with the same <time> parameter.
w	Execute on working days only.
x=<cmd>	Execute the command supplied as an argument (<cmd>) instead of obtaining a list of commands from standard input (see NOTES).

(continued)

EXAMPLES

1. at 5:00pm wednesday <wed_cmnds
2. at
3. at 8:am "+rwmx=shell startday"

The first example submits commands for execution at 5:00 P.M. on a Wednesday. The commands are read from the file "wed_cmnds".

The second example submits commands for execution at midnight. These commands are read from standard input. If standard input is a terminal, the prompt "at>> " requests the next command in the list being submitted. The user must type an end-of-file character as the first character on the line in order to end the list.

The third example submits the command "shell startday" for execution at 8:00 A.M. on working days only and requests that the command be resubmitted upon successful completion (the quotation marks are necessary because of the space character embedded in the command). This example sends time stamps to the user through the system mail when the command begins and when the command successfully ends.

NOTES

- The time parameter <when> is the next occurrence of that time. For example, the command "at 14:00 1" should be read as "at the next 14:00 hours on the first of a month" instead of as "on the next first of the month at 14:00 hours." Notice the difference in meaning if it is currently noon on the first of the month.

- If the user types a keyboard interrupt (control-C), the commands being submitted by "at" are discarded, and control returns to the calling procedure (usually the shell program).

- If the argument to the 'x' option contains a space or any other character which has special meaning to the shell program (such as the matching characters, the pipe symbol, or the symbols for I/O redirection), the user must enclose the option string which contains the 'x' option in quotation marks (see the third example). command.

- If the user represents a day of the month with a number that is greater than the largest day of that month, "at" interprets it as the last day of the month. For example, "2/31" always refers to the last day of February.

- If a list of submitted commands expires (the requested execution time passes by more than an hour and the 'd' option has not been requested) and the 'r' option has been requested, the list of commands is not executed, but is resubmitted with the same <time> parameter as though it had been executed.

ERROR MESSAGES

at error: Unknown option: <char>
The option <char> is not known to "at" (see SYNTAX and Options Available).

at error: Unrecognizable string: <str>
The characters <str> could not be deciphered by "at" (see Format of Arguments).

at error: Ambiguous string: <str>
The characters <str> are not unique to one keyword (see Keywords).

at error: Invalid construction of date and time.
The date and time specified contain conflicting information (see Format of Arguments).

at error: Month number is out of range.
The month number specified is less than 1 or greater than 12 (see Format of Arguments).

at error: Day of month is out of range.
The day of the month specified is less than 1 or greater than 31 (see Format of Arguments).

at error: Hour number is out of range.
The hour number specified is greater than 23 (see Format of Arguments).

at error: Minute number is out of range.
The minute number specified is greater than 59 (see Format of Arguments).

at error: "AM" or "PM" with twelve hour clock only.
Either "AM" or "PM" was used with an hour number which was not between 1 and 12 inclusive (see Format of Arguments).

at warning: ´r´ option ignored with ´now´ keyword.
The repeat option ´r´ is ignored if the <time> parameter is "now". The commands are not resubmitted upon successful completion.

at warning: ´w´ option ignored with ´now´ keyword.
The option that requests execution on working days only, ´w´, is ignored if the time specification is "now".

at warning: restart /etc/atexecute
The commands have been queued, but they cannot be successfully executed unless the "atexecute" command is run before the execution time arrives.

SEE ALSO

atexecute
atinfo
shell

atexecute

Prepare the "at" subsystem for execution and initiate the "at daemon" that handles the lists of commands submitted by the "at" command.

SYNTAX

```
/etc/atexecute [+krs]
```

DESCRIPTION

The "atexecute" command initiates the handling of lists of commands that have been submitted through the "at" command. It first prepares the directory containing the lists submitted ("/usr/gen/at"), then spawns the daemon, or continuous background task, that executes these lists.

The command fails if the system is in single-user mode. All expired lists of commands (those whose execution time has passed by more than one hour) that were submitted through "at" without the 'd' option are removed.

Options Available

- k Keep all lists of commands, including those which have expired.
- r Remove all lists of commands, regardless of their execution times.
- s Permit execution in single-user mode.

The "at daemon"

The "at daemon" (from now on referred to as the daemon) handles the execution of lists of commands submitted through the "at" command. It normally sleeps until an event occurs to indicate that there is something for it to do. These events are "alarm" interrupts, "hangup" interrupts, and interrupts from the "at" command.

The "alarm" and "at" interrupts indicate to "atexecute" that it must search the list of files containing lists of commands (called submitted files) to see if one is ready for execution. Any submitted file whose requested execution time has passed is ready for execution. If no file is ready for execution, the daemon computes the time until something will be ready to execute and sleeps for that amount of time. If at least one file is ready for execution, the daemon selects a file for execution.

(continued)

If more than one file is ready to run, the daemon selects the files in order of ascending execution time. If two or more files have the same requested execution time, the files submitted with the 'b' option are selected before those submitted without it. Otherwise, the order of selection is indeterminable.

Submitted files have names of the form "##########A?" where each pound sign, '#', is a digit (0-9), 'A' is the letter 'A', and '?' is any upper- or lowercase letter. Encoded in that file name are the requested execution time and the execution flags. (The "atinfo" command decodes that information.)

After a file has been selected for execution, the first number in its name is changed to its corresponding letter of the alphabet (e.g., 0 becomes 'A', 1 becomes 'B', etc.). A shell program is invoked to process the selected command list. The daemon waits for the shell program to complete before selecting another file unless the file was submitted with the 'b' option. When the shell ends, "atexecute" removes the file and searches to see if any remaining files are ready for execution.

If the daemon receives a "hangup" interrupt, it terminates gracefully as soon as possible. It terminates immediately if it is sleeping, but if it is currently executing a file, it waits until that file completes before terminating. In all cases, the daemon immediately breaks the communication link between "at" and itself.

Format of the "holidays" File

The "at" daemon handles the "at" option 'w' (execute on working days only). The daemon looks for a file called "/usr/gen/at/holidays". If it does not find the file, the daemon assumes that all days of the week are working days and that there are no holidays. If it finds the file "holidays", it expects it to contain a list of days of the week which are not working days and a list of dates which are holidays.

The first line of the file "holidays" is a list of keywords, separated by spaces, naming the days of the week which are not working days. A maximum of five nonworking weekdays is accepted. (See "at" for information on keywords.)

On the second and subsequent lines, the dates of the year that are holidays (nonworking days) are listed, one per line. The format for the date is the same as that for the <date> parameter described for the "at" command. A maximum of thirty holiday dates is accepted.

NOTES

- The 'k' and 'r' options are mutually exclusive and may not be specified together.

- If a file submitted through "at" with the 'r' option expires, it is resubmitted before it is removed.

- Communications between "at" and the daemon are made through a file called "/tmp/atxctrpid", which contains the task number of the daemon and other information. If the daemon discovers that this file has been deleted or corrupted, it terminates.

- The system manager can avoid having to execute the "atexecute" command every time the system is booted by putting the following command in the file "/etc/startup":

```
/etc/atexecute +ks
```

ERROR MESSAGES

atexecute error: 'k' incompatible with 'r'
The 'k' option (keep all expired files) is incompatible with the 'r' option (remove all submitted files).

atexecute error: 'r' incompatible with 'k'
The 'r' option (remove all submitted files) is incompatible with the 'k' option (keep all expired files).

atexecute error: System is in single-user mode.
The "atexecute" command does not allow itself to run if the system is in single-user mode unless the user specifies the 's' option.

atexecute error: /tmp/atxctrpid already exists
The file that allows communication between the "at" command and the "at" daemon already exists. Probably, the "atexecute" command is not necessary as a daemon is already running on the system. However, if "at" commands are warning that they are unable to awaken the daemon, the system manager should remove this file and try the "atexecute" command again.

atexecute warning: no holidays
The file "/usr/gen/at/holidays" could not be found or could not be deciphered. No holidays (nonworking days) are recognized.

SEE ALSO

at
atinfo

atinfo

Display the current status of the "at" subsystem.

SYNTAX

```
atinfo [+hp]
```

DESCRIPTION

The "atinfo" command examines the "at" subsystem and writes its current status, consisting of the status of the "at" daemon and a table of submitted lists of commands, to standard output. If the user specifies the ´h´ option, the file containing the information on nonworking days and holidays is also displayed.

The ´p´ option allows a user to examine and remove files that the same user previously submitted. The system manager may examine or remove any file. In response to the prompt "? ", the user may type an ´l´ to examine (list) the list of commands, an ´r´ to remove the list of commands, or an ´n´ to go to the next list of commands. A carriage return must follow the character. Typing only a carriage return is the same as typing an ´n´ followed by a carriage return. The ´p´ option causes "atinfo" to write all information to standard error. Otherwise, it writes all information to standard output.

Options Available

- h List the file containing information about holidays.
- p Prompt to list and remove each list of commands submitted.

EXAMPLES

1. atinfo +p

This example displays the current status of the "at" subsystem. After displaying an entry for each file in the queue submitted to "at" by the user, the "atinfo" command prompts for instructions on listing or removing the commands in that file.

MESSAGES

"at" daemon is available, process id is <num>
The "at" subsystem is currently active and is available for the execution of lists of commands. The process ID of the "at" daemon is <num>.

"at" daemon is not available
The "at" subsystem is not currently active. Lists of commands submitted by the "at" command cannot be executed until the subsystem is activated by the "atexecute" command.

ERROR MESSAGES

atinfo warning: Cannot be examined
The file has either been selected for execution or removed. Therefore, it may not be examined.

atinfo warning: Cannot be removed
The file has either been selected for execution or removed. Therefore, it may not be removed.

´<char>´ unknown. Use ´l´, ´n´, or ´r´
The "at" command cannot recognize the character typed in response to the question-mark prompt. The recognizable responses are ´l´, ´n´, and ´r´.

syntax: atinfo [+hp]
The "atinfo" command contains either an argument which is not an option string or an unknown option.

SEE ALSO

at
atexecute

backup

Backup or restore files.

SYNTAX

backup <dev_name> [<file_name_list>] [+aAbBCdDelLnNpqrRtTVzZ]

DESCRIPTION

The "backup" command is used to create and maintain archival backups of files or directories on the system. Although the program is named "backup", it can operate in four distinct modes, selected by options: create mode, append mode, catalog mode, and restore mode. In create mode "backup" copies the specified files or directories to the backup device. It destroys any data that are already on the backup device. In append mode, "backup" adds the specified files or directories to the backup device beyond all existing files. Thus, it is possible to append to a backup device a file with path and file names identical to those of an existing backup file. In catalog mode "backup" lists the contents of the backup device in much the same format as that used by the "dir" and "ls" commands. In restore mode it retrieves files or directories from a backup device.

The "backup" command stores files and directories on, and retrieves them from, block devices only. In most cases the backup device is some sort of disk, probably a floppy disk, but streaming tape devices may also be used. The "backup" command uses a unique file structure on backup devices, which is completely different from the standard UniFLEX file structure. Therefore, backup devices must not be mounted onto the UniFLEX file system using the "mount" command. The only way to read devices written by "backup" is to use "backup" in restore mode. The only other UniFLEX command which the user should use on a backup device is "devcheck".

If the backup device is a disk, it should generally be formatted before the backup operation begins. Although the UniFLEX file structure created by the format command is destroyed by "backup", the raw media-formatting is essential. During the backup process, the user is given the opportunity to request that "backup" format disks before writing to them.

Backups may extend over more than one volume of the backup medium. There are no restrictions on the sizes of files copied. If necessary, "backup" breaks files into segments and stores each segment on a different volume.

The "backup" command writes all prompts and error messages to standard error. It writes everything else to standard output.

Arguments

<dev_name>	Name of the backup device.
<file_name_list>	List of the names of files and directories to process. Default is the working directory.

If the user specifies a directory name as an argument in restore, create, or append mode, the program processes only the files within that directory. If the user also specifies the 'd' option, the program restores all files within the given directory and its subdirectories.

Options Available

a=<days>	Copy only those files which are no older than the specified number of days. A value of 0 specifies files created since midnight on the current day; a value of 1 specifies files created since midnight of the previous day, and so forth.
A	Append to a previous backup.
b	Print sizes of files in bytes.
B	Do not backup or restore files which end in ".bak".
C	Print a catalog of the files on an existing backup. If the user specifies the 'C' option, "backup" ignores all the names in <file_name_list>.
d	Backup or restore entire directory structures.
D	Do double-buffered writes to the tape if the hardware supports it. When the "backup" command performs double-buffered writes, it splits the buffer it would normally use in half. When the first half is filled, "backup" writes it to the tape while it is filling the second half. When the second half is filled and the first half has been written to the tape, "backup" writes the second half to the tape while refilling the first half. Writing continues in this manner until the operation is complete.
e	Erase the tape prior to writing. This option may only be used in create mode. The process of erasing a tape automatically retensions it.
l	List file names as they are copied or restored.

L	Do not unlink files before restoring.
n	Only restore a file if the copy on the backup device is newer than the copy at the destination. If the destination file does not exist, the program restores the file (unless prohibited by another option, such as the 'B' option).
N	Do not prompt for the initial volume.
p	Prompt the user with each file name to determine whether or not the specified procedure (backup or restore) should be performed on that particular file.
q	Suppress all printing except for prompts and error messages. This option is useful when "backup" is running in the background.
r	Retension the tape before starting the specified procedure. This option is used to eliminate any slack in the tape. The drive winds the tape forwards until it reaches the physical end of the tape, then rewinds it. If the backup procedure spans more than one tape, "backup" retensions each tape.
R	Restore files from an archive.
t[=<file_name>]	Backup only those files which have been created or modified since the date in the specified file. When the backup is finished, update the date in the file (see NOTES). If the user does not specify a file, the default is ".backup.time".
T[=<length>]	The backup device is a streaming tape device with a tape of the specified length (in feet). The default length is 450 feet. Currently, the "backup" command supports tapes 300, 450, or 600 feet long.
V=<volume_name>	Each set of backup volumes has a name. By default, the "backup" command prompts for the volume name. The user may avoid the prompt by specifying the name with the 'v' option. The name may contain as may as forty characters.
z	When it is operating in create or append mode, the "backup" command sets a flag in the header for each file indicating whether or not it is a contiguous file. By default, when operating in restore mode, it preserves the file type. The user may, however, use the 'z' option to specify that all files should be restored as contiguous files.
Z	When it is operating in create or append mode, the "backup" command sets a flag in the header for each file indicating whether

(continued)

or not it is a contiguous file. By default, when operating in restore mode, it preserves the file type. The user may, however, use the 'Z' option to specify that no files should be restored as contiguous files.

All modes except catalog mode are quiet. The 'l' option allows the user to see what the program is actually doing.

The 'n' option is only used in restore mode.

The 't' option can be used only in create and append modes. If the user specifies the 't' option, but the "backup time" file specified as its argument does not yet exist, "backup" copies all the files and directories listed on the command line. Thus, a user may obtain a full backup (either without the 't' option or with a nonexistent "backup time" file) or a partial backup, which includes only those files created since the last backup.

EXAMPLES

1. backup /dev/fd0 +l
2. backup /dev/fd0 +C
3. backup /dev/fd0 +lR
4. backup /dev/fd0 +ld file1 file2 dir1 dir2
5. backup /dev/fd0 +ld file1 file2 dir1 dir2 +a=5
6. backup /dev/fd0 +ltT=300
7. backup /dev/fd0 +lAt=backup_time
8. backup /dev/fd0 +lRn file1 dir2

The first example backs up all files in the working directory to the device "/dev/fd0". The file names are listed as they are copied to the device.

The second example lists the contents of the backup on "/dev/fd0". If this command is executed just after the command in the first example, a detailed listing of the files copied is printed. The format of this listing is very similar to that of the commands "dir" and "ls".

The third example restores all of the files, excluding subdirectories and their contents, from the backup on "/dev/fd0". If this command is executed just after the command in the first example, the files backed up in that example are restored.

The fourth example copies (in order) the files "file1" and "file2", then all files and directories contained in the directories "dir1" and "dir2".

The fifth example performs the same function as the fourth example except that it copies only those files which are five days old or less.

The sixth example creates the same backup as the first example, but only copies the files created or modified after the time contained in the file ".backup.time". If this file does not exist, all the files are copied. This example specifies that the backup device is a streaming tape device with a 300-foot tape.

The seventh example adds a set of files to a previously created backup. In particular, it adds exactly the files which were created or modified since the creation of the file "backup_time".

The eighth example restores the file "file1" from the backup. It then restores the files contained in "dir2" on the backup, creating the directory "dir2" if necessary. This example does not restore any subdirectories in "dir2" or any files or directories contained in subdirectories in "dir2".

NOTES

- When using append mode, the user must place the final volume of the backup medium in the backup device. Because the "backup" command always expects to receive the volumes in order, it issues a message saying that the user has inserted the wrong volume and prompts for permission to continue. In this case the user does want the last volume in the drive and should respond with a 'y' to the prompt. The program then appends files to that volume, requesting new volumes as necessary.

- In restore mode, file names or directory names on the command line are used to select the files or directories to be restored. The program searches the entire backup for each argument specified. If multiple files satisfy the restoration criteria, the program restores them all, destroying the older version as the new one is restored. Thus, the user must provide all backup volumes (in order) for each argument to ensure proper restoration.

- When files are restored, they are generally restored to the same directory location as the user specified when they were backed up. As files are backed up, "backup" makes an indication of the path name for each file. When files are restored, the program uses the path name to place the file in its proper directory location. If the path name is relative (i.e., does not begin with '/'), the path name of the restored directory is also relative. Thus, files backed up with a relative path name may be restored to a directory location different from the one in which they were created. An example should make this clear. If the working directory is backed up, either by specifying no source files or by using the directory name '.', the files are backed up with a relative path of '.'. When

these files are restored, they are placed in the directory '.', which might not be the same directory they originally came from. This feature allows the manipulation of entire file systems in a general fashion. To specify a unique directory location for a file, the user should specify its entire path name, starting with '/'.

. It is possible to restore backed up data onto the device currently being used as the root device or system disk. Two possible problems arise, however. First of all, if the UniFLEX operating system is restored from a backup, the result is not bootable. In such a case, the UniFLEX file must be copied from the original master disk and installed in order to allow booting. The second problem occurs if the shell program or the device "tty00" is restored over the current shell or "tty00". This operation leaves unreferenced files in the file system. Unreferenced files must be corrected with the "diskrepair" command. In general, it is always a good idea to run "diskrepair" on the root device after restoring backed-up data to it.

MESSAGES

Several of the following messages prompt the user for a positive or negative response. The program interprets any response that does not begin with an upper- or lowercase 'n' as a positive response.

Backup to "<file_name>"
Catalog of backup on "<file_name>"
Restore backup from "<file_name>"
Update backup on "<file_name>"
These messages are printed when "backup" begins. They notify the user of the function about to be performed.

Copy "<file_name>" (y/n)?
Restore "<file_name>" (y/n)?
If the user specifies the 'p' option, the program prints one of these prompts before it takes any action. A response of 'n' or 'N' indicates that the operation should not be performed for the given file. Any other response is interpreted as "yes".

Device model name?
The user should respond to this prompt with the model name which corresponds to the volume being formatted. Refer to the documentation for the format program for the available models.

Do you wish to abort "append" function and create a new backup?
This message is printed at the initiation of the "append" operating mode if an invalid header (indicating a bad backup format) is detected. The user has the option of aborting from "append" mode and switching to "create" mode.

... Erasing tape
If the user specifies the 'e' option, this message appears while the drive erases the tape. Because the drive winds the tape forwards until it reaches the physical end of the tape, then rewinds it, this procedure may take several minutes.

Format program name?
This prompt is issued in response to a "format" request for the next volume. The user should respond with the name of the appropriate formatting program for the given device.

Insert next volume - Hit C/R to continue:
This prompt is issued when the program needs a new backup volume. The user should type a carriage return only when the next volume has been placed in the device. When creating new backups or when appending to an old one, the user may enter the character 'f' or 'F', followed by a carriage return. If the program is in append mode, it automatically switches to create mode and starts a new backup. The 'f' and 'F' both indicate that the volume has been inserted in the drive, but that it must be formatted before continuing. If the user specifies an 'f', "backup" first tries to read the file "/etc/format.control" (the format-control file).

The format-control file may contain instructions for formatting all media in all drives the same way or different instructions for formatting media in different drives. In the first case, the first line of the file should contain the file specification of the formatting program (e.g., "/etc/formatfd"); the second, the model name. In the second case, the first character of the first line of the file must be an asterisk, '*'. The file specification for a device must immediately follow the asterisk. The second and third lines of the file should then contain the file specification of the formatting program and the model name, respectively, for formatting media in the specified device. This pattern of three lines may be repeated to specify formatting instructions for other devices. The first character of any line specifying a device must be an asterisk.

If the format-control file does not exist, "backup" prompts the user for the information necessary to format the volume. If the format-control file contains formatting instructions for individual devices and the user specifies a device that is not described in the file, "backup" prompts the user as if the format-control file did not exist.

If the user specifies an 'F' in response to the prompt to continue, "backup" immediately prompts for the information necessary to format the volume. Thus, the 'F' can be used to change the type of formatting being used during the backup operation.

If for any reason the formatting operation fails, "backup" again prompts the user to insert the next volume, and if the user tries to

format a disk, prompts for formatting instructions. These instructions become the default instructions that "backup" uses if the user subsequently specifies an 'f' in response to the prompt to continue.

link "<file_name_1>" to "<file_name_2>"
copy "<file_name>"
==== Copying from "<dir_name>"
The program prints these messages as it takes the corresponding action during a creation operation.

... Retensioning tape
If the user specifies the 'r' option, this message appears while the drive retensions the tape. Because the drive winds the tape forwards until it reaches the physical end of the tape, then rewinds it, this procedure may take several minutes.

... Spacing to EOT
When the "backup" command begins an append operation, it must first locate the logical end of the tape. To do so, it reads the tape track by track until it reaches the logical end. Once "backup" finds the end, the drive must rewind the tape to the beginning and wind it to the logical end of the tape before it will allow "backup" to write to the tape. This procedure may take several minutes.

This is Volume <number_1> -- Expected Volume <number_2> --
Continue?
The program expects the user to insert volumes in sequential order. If a volume appears out of order, "backup" prints this message. If the user types anything except an 'n' or an 'N' as the first character of the response to the message, "backup" ignores the fact that the volumes are out of order and continues with the backup. Otherwise, it prompts the user for another volume. If the program is in restore mode, it is important to insert volumes sequentially because "backup" cannot correctly restore files that are broken across volumes if the volumes are inserted out of order.

... Using <number> tape buffers
The "backup" command tries to obtain as much memory as possible to use as a buffer when it is working with a tape device so that it can minimize the number of times it needs to access the tape. This message tells the user the number of 512-byte blocks that it was able to obtain.

Volume name?
Each set of backup volumes has a name. The user should enter the name in response to this prompt. The name may contain as many as forty characters.

Volume <number> of "<vol_name>"
Whenever a new volume is inserted and properly validated, the program prints this message, which indicates the name of the backup volume and its sequence number.

ERROR MESSAGES

"backup" must run with system manager privileges!
Currently running as: <user_number>
Some features of "backup" require privileges only available to the system manager. In most installations the program will be installed so that these privileges are given to the program. If this message is printed, the user should check with the system manager.

'd' option only allowed for streaming tape (+T).
The 'T' option must be used with the 'd' option.

"<dev_name>" is not a block device
"<dev_name>" is not a tape device
The destination device for the backup must be a block device or, if the user specifies the 'T' option, a streaming tape device. This message indicates that the specified device (which is always the first argument) is not such a device.

'e' option allowed only in create mode.
The 'e' option is incompatible with the 'A', 'C', or 'R' option.

'e' option only allowed for streaming tape (+T).
The 'T' option must be used with the 'e' option.

***Error: <problem> "<file_name>": <reason>
The operating system returned an error when "backup" tried to perform the action described by <problem> on <file_name>. This error message is followed by an interpretation of the error returned by the operating system. The program tries to continue for all errors except "device full" during restore mode.

"<file_name>" not located - try again?
When using the program in restore mode, the user may specify which files or directories to restore. If the program cannot find a specified file or directory after searching the entire backup, it prints this message. If the response is not 'n' or 'N', the program searches the entire archive again. This option is allowed because volumes need not be inserted in order of their creation when the program is in restore mode. If one volume is left out or if the final volume is inserted before the entire archive has been processed, some files might not be processed. Note that if the user specifies more than one file name or directory name, the program processes the entire archive for each file before proceeding to the next one.

(continued)

-- Formatting not allowed during Catalog/Restore or on tape
The user may not format a disk if the program is in either catalog or restore mode or if the backup device is a streaming tape device.

*** Invalid Volume Header -- Not a "backup" disk ***
The program validates each backup volume before using it. If this validation fails, the program prints this message to indicate that something is wrong. The user then has another chance to insert the proper volume and continue. If validation fails while the program is in append mode, the user may abort from append mode and create a totally new backup instead.

'r' option only allowed for streaming tape (+T).
The 'T' option must be used with the 'r' option.

('t' or 'a') and ('C' or 'R') are incompatible options
'a' and 't' are incompatible options
'A' and 'C' are incompatible options
'A' and 'R' are incompatible options
'R' and 'C' are incompatible options
The program can run in only one of its four operating modes at a time. Specifying certain combinations of options implies the execution of more than one operating mode and is therefore illegal.

Invalid option: <char>
The option specified by <char> is not a valid option to the "backup" command.

** Warning: directory "<dir_name>" is too large!
** Some directories were ignored
** Warning: directory "<dir_name>" is too large!
** Some files were ignored
The program uses some internal tables during the backup process (not during restore or catalog). If the limits of these tables are exceeded (highly unlikely), these messages appear.

SEE ALSO

format

badblocks

Add the specified block to the file of bad blocks.

SYNTAX

/etc/badblocks <dev_name> <address_list> [+dmpqsv]

DESCRIPTION

The "badblocks" command adds the specified blocks to the file "/.badblocks" (also known as the bad-blocks file) on the specified device. The blocks in this file are inaccessible to the operating system. Thus, the user can remove a bad block from circulation by putting it in the bad-blocks file. When it has added all the specified blocks to the bad-blocks file, "badblocks" calls "diskrepair".

Arguments

<dev_name> — The name of the device which contains the bad blocks.

<address_list> — A list of the addresses of the blocks to place in the bad-blocks file. The user may specify the addresses in hexadecimal, decimal, or octal. The "badblocks" command interprets an address that begins with "0X" or "0x" (leading character is a zero) as a hexadecimal address; one beginning with a "0" (zero) that contains no digits greater than 7, as octal; any other address, as decimal. Thus, the following addresses are equivalent:

0X40 = 0x40 = 0100 = 64

The user should be careful not to use a leading 0 with a decimal address because "badblocks" will interpret it as an octal address if none of the digits is greater than 7.

Options Available

d — Do not run "diskrepair".

m=<address> — If a block must be used for mapping, use the one specified by this option. The address is specified in the same way as an address used as an argument to the "badblocks" command (see Arguments). By default, "badblocks" takes a

(continued)

	block for mapping from the free list. However, if structural problems or media errors make it impossible to do so, the ´m´ option can be used to tell "badblocks" which blocks to use for any necessary mapping. The ´m´ option may be used up to twelve times in one command, but it should never be necessary to use it more than twice.
p	Prompt for permission to make repairs while running "diskrepair".
q	Use quiet mode when running "diskrepair".
s	Ignore the validity of the system information record (SIR) when determining the size of the disk, the size of the fdn space, and the size of the paging space. This option should not be used unless the "badblocks" command fails due to problems in the SIR.
v	Run "diskrepair" in verbose mode and report the number of bad blocks in the bad-blocks file.

EXAMPLES

1. /etc/badblocks /dev/w0 596 10321
2. /etc/badblocks /dev/w0 596 10321 +pv

The first example places block numbers 596 and 10321 into the file "/.badblocks" on the disk in drive w0. It then calls "diskrepair".

The second example places block numbers 596 and 10321 into the file "/.badblocks" on the disk in drive w0. It then calls "diskrepair" with the ´p´ and ´v´ options.

MESSAGES

Total bad blocks = <num_of_bad_blocks>
If the user specified the ´v´ option, the "badblocks" command sends this message to standard output when it successfully completes its job.

ERROR MESSAGES

The error messages listed in this section are those messages which are returned by the "badblocks" command. A user who is running "badblocks" without the ´d´ option may receive other error messages, which come from the "diskrepair" command and are explained in the documentation for "diskrepair".

Cannot allocate memory for bad-block map.
The operating system could not allocate enough memory for the bad-block map. The user should use the "headset" command to increase the size of the "badblocks" program. If the problem persists, it may mean that the volume size in the SIR is too large. In such a case the user should execute "badblocks" with the 's' option.

Cannot call "/etc/diskrepair".
The "badblocks" command cannot read or execute the file "/etc/diskrepair".

Cannot open device.
The operating system returned an error when "badblocks" tried to open the specified device or read its fdn.

Cannot read indirection block.
This error message indicates that either the root directory or the file "/.badblocks" is damaged. The user should salvage as much of the data on the disk as possible, then reformat the disk.

Cannot read System Information Record.
The SIR is so badly damaged physically that "diskrepair" cannot read it. The user may be able to salvage information from the disk, but must eventually reformat it.

Cannot status the root directory.
The "badblocks" command cannot read the fdn which describes the root directory. The user may be able to salvage some information from the disk, but must eventually reformat it.

Data space overlaps contiguous-file space.
The information in the SIR indicates that the data space overlaps the contiguous-file space. The user should execute "diskrepair" to fix this problem. If and only if "diskrepair" fails to fix the problem, the user should execute "badblocks" with the 's' option.

Data space overlaps paging space.
The information in the SIR indicates that the data space overlaps the paging space. The user should execute "diskrepair" to fix this problem. If and only if "diskrepair" fails to fix the problem, the user should execute "badblocks" with the 's' option.

Disk too large or bad size in SIR.
The "badblocks" command checks to see that the size of the disk is within the range that it can handle. The current limit is approximately 400 Megabytes. If the data in the SIR indicate that the disk is larger than this limit, "badblocks" issues this error message. This error is also fatal to "diskrepair". The user should salvage as much information as possible and reformat the disk.

(continued)

Error reading ".badblocks" file.
The "badblocks" command must read the file "/.badblocks" before it adds any blocks to the file. This message indicates that it encountered an I/O error when it tried to read the file.

Error reading block <block_num>.
The specified block contains an I/O error.

Error reading fdn <fdn_num> in block <block_num>.
The specified fdn contains an I/O error.

Error writing block <block_num>.
The operating system returned an I/O error when "badblocks" tried to write to the specified block.

Error writing fdn <fdn_number> in block <block_num>.
The operating system returned an I/O error when "badblocks" tried to write to the specified fdn.

"/etc/diskrepair" terminated abnormally, status = <status>
The "diskrepair" command received a program interrupt from the operating system and terminated with the termination status indicated. The user cannot determine the source of such an error; however, it is not indicative of a problem with either "diskrepair" or the device. The "badblocks" command should be rerun, for the problem may not recur.

Invalid argument to ´m´ option
The argument to the ´m´ option must be a decimal number.

Invalid option: ´<char>´.
The option specified by <char> is not a valid option to the "badblocks" command.

No ".badblocks" file on "<dev_name>".
In order for the "badblocks" command to succeed, the bad-blocks file must already exist. The user should salvage as much information as possible, reformat the disk, and verify that the "format" program created a bad-blocks file.

No device specified.
The user did not specify a device on the command line.

No free blocks available for mapping.
The "badblocks" command needed a block to use for mapping in extending the file "/.badblocks", but no blocks were available. The user can reexecute "badblocks" with the ´m´ option. It is wise to specify as the argument to the ´m´ option a block which is in the paging space.

Not a block device.
The "badblocks" command can only operate on a block device.

Only one device may be specified.
The user must specify exactly one device on the command line.

Output directed to specified device.
When putting blocks in the bad-blocks file, it is impractical to try to redirect the output to the device that is being repaired. The user should reexecute "badblocks" without redirecting the output or redirecting it to a different, mounted device.

Paging space overlaps contiguous-file space.
The information in the SIR indicates that the paging space overlaps the contiguous-file space. The user should execute "diskrepair" to fix this problem. If and only if "diskrepair" fails to fix the problem, the user should execute "badblocks" with the ´s´ option.

SIR fdn block count error: <size>
The number of blocks reserved for fdns exceeds either 65,535 (the maximum allowed by UniFLEX) or the size of the disk. This error is also fatal to "diskrepair". The user should salvage as much information as possible and reformat the disk.

SIR pointer to free space is invalid.
The "badblocks" command needs a block from the free space to use for mapping in extending the bad-blocks file; however, because an internal pointer to the free space is invalid, it cannot obtain the necessary block. The user should should reexecute "badblocks" using the ´m´ option. It is wise to specify as the argument to the ´m´ option a block which is in the paging space.

Too many occurrences of the ´m´ option.
The user may not specify the ´m´ option more than twelve times.

SEE ALSO

diskrepair
format

bcompare

Compare two files byte by byte.

SYNTAX

bcompare <file_name_1> <file_name_2> [+es]

DESCRIPTION

The "bcompare" command compares two files byte by byte and reports the differences to standard output. The output is in the following form:

<address> --> <byte_in_file_name_1> : <byte_in_file_name_2>

The address that is reported is the offset from the beginning of the file, which is not necessarily the beginning of the data. Some UniFLEX files have headers which precede the data.

Arguments

<file_name_1>	The name of the first file to use.
<file_name_2>	The name of the file to compare to <file_name_1>.

Options Available

e=<end_byte>	Perform the comparison up to and including the byte specified by the hexadecimal number <end_byte>. The number specified must be greater than the number specifying the starting byte.
s=<start_byte>	Begin the comparison at the byte specified by the hexadecimal number <start_byte>. The number specified must be greater than or equal to 0.

EXAMPLES

1. bcompare test_1 test_2
2. bcompare test_1 test_2 +s=1000
3. bcompare test_1 test_2 +e=1000

The first example does a byte-by-byte comparison of the files "test_1" and "test_2". Any differences are reported to standard output, which defaults to the user's terminal.

The second example does a byte-by-byte comparison of the files "test_1" and "test_2" beginning with the 4,097th byte. Any differences are reported to standard output, which defaults to the user's terminal.

The second example does a byte-by-byte comparison of the files "test_1" and "test_2" from the beginning of the files up to and including the 4,097th byte. Any differences are reported to standard output, which defaults to the user's terminal.

NOTES

- Although the "bcompare" command is used mainly for comparing binary files, it can also compare text files. However, the "compare" command, which compares text files line by line, is also available.

MESSAGES

"<one_file>" is longer than "<the_other_file>".
  At the end of the comparison, "bcompare" reports any difference in the length of the two files.

ERROR MESSAGES

Error opening "<file_name>": <reason>
  The operating system returned an error when "bcompare" tried open the file <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Error seeking in "<file_name>": <reason>
  The operating system returned an error when "bcompare" tried to seek in the file <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Invalid end byte specified: "<str>".
  The string used to specify the end byte is not a valid hexadecimal number.

Invalid option: '<char>'.
  The option specified by <char> is not a valid option to the "bcompare" command.

Invalid start byte specified: "<str>".
  The string used to specify the start byte is not a valid hexadecimal number.

Multiple end bytes specified!
  The user may not specify more than one end byte.

Multiple start bytes specified!
The user may not specify more than one start byte.

Start byte beyond end byte.
The value of the argument to the 's' option must be less than or equal to the value of the argument to the 'e' option.

Syntax: bcompare <file_name_1> <file_name_2> [+es]
The "bcompare" command expects at least two and no more than four arguments and options on the command line. This message indicates that the sum of the arguments and options is less than 2 or greater than 4.

SEE ALSO

compare

blockcheck

Check the integrity of the allocation of all blocks used in files and of the free list on the specified device.

SYNTAX

/etc/blockcheck <dev_name>

DESCRIPTION

The "blockcheck" command checks the integrity of the allocation of all blocks used in files and of the free list on the specified device. It locates problems such as duplicate blocks, missing blocks, and invalid block addresses.

This command is primarily intended for use by the "diskrepair" utility, which calls it. It may also be used on its own. However, "blockcheck" can only check the disk; it cannot repair it. If the output from the command suggests that the disk is damaged, the user should execute "diskrepair" on the disk.

The "blockcheck" command should be executed only when no other tasks are active on the system. Otherwise, the results are unpredictable.

Arguments

<dev_name> The name of the device to check. It must be a block device.

EXAMPLES

1. /etc/blockcheck /dev/fd0

This example checks the integrity of the the allocation of blocks on the disk in floppy drive 0.

ERROR MESSAGES

Cannot check a "backup" disk.
The disk in the specified device was created by the "backup" command and cannot be checked by the "blockcheck" command.

Cannot read boot block.
When "blockcheck" tried to read the boot sector, it detected an I/O error.

Invalid option: '<char>'.
The option specified by '<char>' is not a valid option to the "blockcheck" command.

SEE ALSO

devcheck
diskrepair
fdncheck

chd

Change the user's working directory.

SYNTAX

chd [<dir_name>]

DESCRIPTION

The "chd" command, which is part of the shell program, changes the user's working directory to the directory specified on the command line. If no directory is specified, the default is the user's home directory (the directory entered on logging in). The user must have execute permission in the directory specified.

Arguments

<dir_name> The name of the directory to use as the working directory. Default is the user's home directory.

EXAMPLES

1. chd /usr/mark
2. chd book
3. chd

The first example changes the working directory to the directory "/usr/mark".

The second example changes the working directory to the directory "book", which resides in the current working directory.

The third example changes the working directory to the user's home directory.

ERROR MESSAGES

Home directory is not known.
The shell program to which the user issued the "chd" command is not the user's login shell. It does not know what the user's home directory is and, therefore, cannot execute the command.

Cannot change directories.
The operating system returned an error when the shell program tried to change directories. This message is preceded by an interpretation of the error produced by the operating system.

SEE ALSO

shell

compare

Compare two text files line by line and report the differences.

SYNTAX

compare <file_name_1> <file_name_2> [+<window_size>]

DESCRIPTION

The "compare" command compares two text files and indicates how they differ. The information provided is usually sufficient to allow the user to change one file into the other. By default, the "compare" command considers that it is in the same place in each of the files if three lines match.

The output from the command reports sets of lines which have been deleted from, added to, or changed in either file. These messages are written from the point of view of how to change the first file into the second file. For instance, the message

File "<file_name>" lines deleted

means that if the lines following the message are deleted from <file_name>, the two files will be the same.

The program also reports the presence of additional lines in a file with the following message:

File "<file_name>" has additional lines

This message is not from the point of view of changing one file into the other. Rather, it means that the file mentioned in the message is the file that contains additional lines.

If a set of lines is deleted from one file and the following line is changed as well, "compare" reports all those lines as lines that have been changed rather than inserted or deleted.

The "compare" command can handle files of any size, but can only process 250 lines at a time. If the files differ in any spot by 250 lines, the program reports 250 lines changed in each file and continues comparing them.

Arguments

<file_name_1>	The name of the first file to use.
<file_name_2>	The name of the file to compare to <file_name_1>.

Options Available

<window_size>	Use the integer <window_size> as the number of matching lines required before considering the files synchronized. The number specified must be between 1 and 250. The default is 3.

EXAMPLES

1. compare /usr/michael/test /usr/cathy/test
2. compare test test.bak +5

The first example compares the file "test" in the directory "/usr/michael" to the file "test" in the directory "/usr/cathy".

The second example compares the two files "test" and "test.bak" in the working directory. The window size for the comparison is five lines.

ERROR MESSAGES

Invalid option: '<char>'.
The option specified by '<char>' is not a valid option to the "compare" command.

Syntax: compare <file_name_1> <file_name_2> [+<window_size>]
The "compare" command expects exactly two arguments. This message indicates that the argument count is wrong.

SEE ALSO

bcompare

copy

Copy a directory or a file to the specified destination.

SYNTAX

copy [<source_name_list>] <dest_file_name> [+bBcdDFlLnopPtTzZ]

DESCRIPTION

The "copy" command allows the user to copy files or directories to the specified destination file. By default, if the destination file does not exist, "copy" creates it as a regular file. If the user does not specify a source file, the working directory is copied.

If the destination file specified is not a directory, the "copy" command can be used simply to make a copy of a regular file or a special file (a block or character device).

If both the source file and the destination file are directories, the "copy" command copies the source directory and all files within that directory that are not directories themselves to the destination directory. By default, it does not copy subdirectories or their contents. If the user specifies more than one source file, "copy" copies them to the destination directory in the order they are listed on the command line. If the corresponding file does not exist in the destination directory, "copy" creates it. If the corresponding file already exists, it is deleted and recreated before copying takes place. Thus, the contents of the file are lost and replaced by the contents of the file being copied. In addition, any links to that file are broken.

The "copy" command always makes a copy with the same permissions as the original file. The user must have execute permission in the directory in which copies are to be made. It is also necessary to have write permission in the file being copied to and, if the destination file specified on the command line is a directory, in that directory as well (unless the 'L' option is specified).

Arguments

<source_name>	The name of the source file or directory to copy. Default is the working directory.
<dest_file_name>	The name of the file in which to place the copy of the source file. The destination file may be any type of file.

(continued)

Options Available

- b Do not copy a file unless it already exists in the destination directory.
- B Do not copy any files with names ending in ".bak".
- c Do not copy a file if it already exists in the destination directory.
- d Copy the entire directory structure (all subdirectories and all their contents to the bottom of the directory tree) of each directory specified on the command line.
- D When copying a directory to the destination directory, append the name of the source file to the name of the destination directory. If a directory with the resulting name already exists, the copying proceeds as usual. If the parent directory for the resulting name already exists, the "copy" command creates the new directory and proceeds as usual. If the parent directory does not exist, the copy command reports an error.
- F If the destination file does not exist, create it as a regular file, not a directory. This option is always implied unless the 'T' option is in effect.
- l List the name of each file as it is copied and the name of the new copy.
- L If a file that already exists in the destination directory is to be updated, do not delete and recreate it before copying to it. Rather, overwrite the old contents with the new. This option preserves any links to the copy.
- n Copy a file only if it is newer than the corresponding file in the destination directory or if no corresponding file exists.
- o Retain original ownership of the file. The system manager must explicitly invoke this option in order to retain the original ownership of the file. For all other users, the use of the 'd', 'D', 'P', or 'T' option implies the 'o' option. They may also invoke the option explicitly.
- p Prompt the user for permission to copy each file.
- P Preserve the modification time of the source file in the copy.
- t If a specified source file is a directory, copy it only if that directory already exists in the destination directory.
- T If the destination file does not exist, create it as a directory.
- z By default, the "copy" command preserves contiguity: a file that is contiguous is copied to a contiguous file; a file that is not, is copied to a noncontiguous file. The user may, however, use the 'z' option to specify that all files should be copied to contiguous files.
- Z By default, the "copy" command preserves contiguity: a file that is contiguous is copied to a contiguous file; a file that is not, is copied to a noncontiguous file. The

 (continued)

user may, however, use the 'Z' option to specify that all files should be copied to noncontiguous files.

EXAMPLES

1. copy /usr/bin
2. copy file_name_1 file_name_2 /usr/bin +ln
3. copy /usr/bin +BP
4. copy tests /usr/bin +bLp
5. copy /usr /usr2 +do
6. copy tests /usr2/usr/tests +t
7. copy gen /usr2 +D
8. copy gen /usr2/gen

The first example copies all regular files in the working directory to "/usr/bin".

The second example copies the specified files to "/usr/bin" only if the source copy is newer than the version in "/usr/bin" or if no copy by that name exists in "/usr/bin". The command lists the name of each file as it is copied and the name of the new copy.

The third example copies all regular files in the working directory, except those with names ending in ".bak", to the directory "/usr/bin". As it makes each copy, it preserves the last modification time of the source file.

The fourth example copies "tests" from the working directory to the directory "/usr/bin" if and only if the file "/usr/bin/tests" exists. If "tests" is a directory, all regular files in that directory which already exist in "/usr/bin/tests" are also copied.

The fifth example copies the entire structure of the directory "/usr" to the directory "/usr2". The 'o' options tells the "copy" command to preserve ownership as it copies the files. Only the system manager may execute this form of the command.

The sixth example copies "tests" from the working directory to "/usr2/usr/tests" if "tests" is a regular file. If "tests" is a directory, the "copy" command copies from "tests" those directories with corresponding directories in "usr2/usr/tests". It also copies all regular files in any directories it copies, whether or not those files have corresponding files in the destination directory.

The seventh example copies "gen" to the directory "/usr2. If "gen" is a directory, the "copy" command copies it to the directory "/usr2/gen".

The last example has the same effect as the seventh example.

NOTES

- The options for the "copy" command are not position-independent. They must appear at the end of the command line.

- Because the destination file can be a regular file, it is possible for the user to inadvertently specify that more than one file be copied to a regular file. In such a case "copy" copies the source files one at a time to the destination file. The result is a copy of the last file to be copied.

ERROR MESSAGES

"Copy" must run with System Manager privilege!
Currently running as: <user_ID>
In order to function properly the "copy" command must be owned by "system" and have its user ID bit set ("s+" permission).

Copying over myself!
This message is a warning that a loop exists somewhere in the specified copying procedure. The command continues, ignoring the loop. For instance, if the user asks to copy the entire directory structure of the root directory to the directory "/usr2", the command recognizes a loop when it prepares to copy the directory "/usr2" to the destination file. The copying procedure continues, but none of the subdirectories from "/usr2" is copied to the destination file.

"<dest_file_name>" is a directory - +F ignored!
The user specified the ´F´ option, but the destination file already exists and is a directory. The "copy" command continues, ignoring the ´F´ option.

***Error: <problem> "<file_name>": <reason>
The operating system returned an error when "copy" tried to perform the action described by <problem> on <file_name>. This error message is followed by an interpretation of the error returned by the operating system.

Inconsistent options - aborting
Certain options, such as ´b´ and ´c´, conflict with each other. If the user specifies conflicting options, the program aborts.

Invalid option: ´<char>´.
The option specified by ´<char>´ is not a valid option to the "copy" command.

Syntax: copy [<source_name_list>] <dest_file_name> [+bBcdDFlLnopPtT]
The "copy" command expects at least one argument. This message indicates that the argument count is wrong.

** Warning: directory "<dir_name>" is too large!
** Some files were ignored
The "copy" command cannot copy more than 500 files from a single directory. In order to make the command succeed, the user should split the offending directory into two or more directories.

** Warning: directory "<dir_name>" too large!
** Some directories were ignored.
The "copy" command cannot process a file if the total number of directories in every directory between that file and the directory specified on the command line exceeds 50. In order to make the command succeed, the user should start at a lower point in the directory tree.

crdir

Create a directory.

SYNTAX

crdir <dir_name_list>

DESCRIPTION

The "crdir" command creates a directory for each name listed as an argument to the command. The user must have write permission in the parent directory (the directory in which the new directory is created) of each directory created. Each new directory contains the entry ".", which represents the directory itself, and the entry "..", which represents its parent directory.

By default, "crdir" creates a directory with "rwxrwx" permissions. However, any default permissions set by the "dperm" command override these permissions. The owner may, of course, change the permissions at any time by using the "perms" command.

Arguments

<dir_name_list> A list of the names of directories to create. All directories used in the name, except the last component of the name, must already exist.

EXAMPLES

1. crdir book
2. crdir /usr/sarah/book

The first example creates the directory "book" in the working directory.

The second example creates the directory "book" in the directory "/usr/sarah". If the directory "/usr/sarah" does not already exist, the command fails.

ERROR MESSAGES

Error creating "<dir_name>": <reason>
The operating system returned an error when "crdir" tried to create the specified directory. This message is followed by an interpretation of the error returned by the operating system.

Error linking "<dir_name>" to its "." file": <reason>
The operating system returned an error when "crdir" tried to link the "." entry to the directory itself. This message is followed by an interpretation of the error returned by the operating system.

Error linking ".." to parent of "<dir_name>": <reason>
The operating system returned an error when "crdir" tried to link the newly created directory to its parent. This message is followed by an interpretation of the error returned by the operating system.

Error setting owner for "<dir_name>": <reason>
Initially, the "crdir" command creates the new directory with the owner "system". It then changes the owner to the user who executed the command. In this case, the operating system returned an error when "crdir" tried to change the owner of the directory. This message is followed by an interpretation of the error returned by the operating system.

Syntax: crdir <dir_name_list>
The "crdir" command expects at least one argument. This message indicates that the argument count is wrong.

SEE ALSO

dperm
kill
perms

create

Create an empty file for each file name on the command line.

SYNTAX

create <file_name_list>

DESCRIPTION

The "create" command creates an empty file for each name specified on the command line. If the file does not exist, it is created with "rw-rw-" permissions, and the owner is the user who executes the command. If the file already exists, the owner and permissions remain intact. However, the file is truncated to a length of 0.

Arguments

<file_name> The name of the file to create. The last component of a file name may not contain more than fourteen characters. The "create" command ignores any additional characters.

EXAMPLES

1. create test
2. create /usr/julie/test

The first example creates the file "test" in the user's working directory.

The second example creates the file "test" in the directory "/usr/julie".

ERROR MESSAGES

Error creating "<file_name>": <reason>
The operating system returned an error when "create" tried to create <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Syntax: create <file_name_list>
The "create" command requires at least one argument. This message indicates that the argument count is wrong.

SEE ALSO

edit

crt_termcap

Create a file defining the capabilities of each terminal on the system.

SYNTAX

```
/etc/crt_termcap <ttycap_file> <ttyassoc_file> <termcap_file>
/etc/crt_termcap +d
```

DESCRIPTION

The "crt_termcap" command creates a file ("termcap") which describes the capabilities of each terminal on the system. This file makes it possible for programs to operate on many different terminals regardless of the individual characteristics of the terminals. The "crt_termcap" command or the "set_termcap" command must be used to create the file "termcap" before any programs which need that file can operate successfully.

Arguments

<ttycap_file>	Input file describing functional capabilities of each type of terminal that may be used on the system.
<ttyassoc_file>	Input file associating each active port with a particular kind of terminal.
<termcap_file>	Output file combining information from the two input files.

Options Available

d Use the file "/etc/ttycap" for <ttycap_file>, "/etc/ttylist" for <ttyassoc_file>, and "/etc/termcap" for <termcap_file>.

The "crt_termcap" command uses two input files to produce one output file. The first input file describes the functional capabilities of each terminal on the system. The second file indicates what type of terminal is on each active port in the system. These files must each conform to a particular format. The following two sections describe these formats.

Format of the File "ttycap"

The file "ttycap" contains one logical entry for each terminal on the system. The format of an entry is

<terminal_name> : <capability_list> :

where <terminal_name> is a character string (it may contain as many as ten characters) which identifies the terminal, and <capability_list> is a list describing the capabilities of the terminal. Each item in this list has the following format:

<keyword> = <value_list>

where <keyword> is a two-character sequence representing a function such as clearing the screen and <value_list> is a list containing decimal values, hexadecimal values, or both. Each value in the list must be separated from the following one by a plus sign, '+'. No spaces may appear between the values and the plus signs. All hexadecimal values must consist of two digits preceded by a dollar sign, '$'. The following are valid value strings:

```
1+2+3
$01+$ff+$80
```

The keywords currently supported are

ho	Home cursor.
cu	Move cursor up without modifying display.
cd	Move cursor down.
cl	Move cursor left.
cr	Move cursor right.
cs	Clear entire screen.
nr	Number of rows on screen (first row is 1).
nc	Number of columns on screen (first column is 1).
wt	Number of seconds to wait between clearing the screen function and sending more information to the terminal.
is	A string of characters sent to the terminal when processing begins in order to initialize the terminal.
bl	Clear the current line from the present cursor position through the end of the line without moving the cursor.
bm	Place the terminal in "background" mode. In this mode characters are written to the terminal with a lower intensity (brightness) than usual.
fm	Place the terminal in foreground (normal) mode.
pc	Position the cursor to an absolute location.
ku	Sent by the terminal in response to the up-arrow key.
kd	Sent by the terminal in response to the down-arrow key.
kl	Sent by the terminal in response to the left-arrow key.
kr	Sent by the terminal in response to the right-arrow key.
kh	Sent by the terminal in response to the home key.
k0-k9	Sent by the terminal in response to other special keys.

(continued)

The following entry describes a ct82 terminal manufactured by Southwest Technical Products, Corporation:

```
ct82:ho=16 cu=01 cd=02 cr=09 cl=04 cs=30+07+12 nr=20 nc=82
     ku=01 kd=02 kr=09 kl=04 is=28+18+30+19+30+20+30+07 bl=06
     bm=28+05  fm=28+21+30+07  wt=1 pc=$0b+$ff+$01+$ff+$80:
```

Not all terminals can support all the functions described here. All the information required to create the list of values should be contained in the manual describing the particular terminal. As can be seen from the previous example, definitions for all keywords are not necessary for the correct functioning of the utilities that use the file "termcap". However, definitions for the following keywords are essential:

cs, ho, nr, nc, and either pc or cu, cd, cl, and cr

The keyword "pc" enables a utility to position the cursor to any absolute location on the screen. Some terminals do not support this feature, in which case the utilities must use relative positioning of the cursor (using ho, cu, cd, cl, and cr) instead. Since the absolute positioning of the cursor is different on almost all terminals, the value string associated with the keyword "pc" is rather complex. At a minimum, absolute cursor positioning requires the desired row and column numbers to be sent to the terminal as part of a control sequence. Different terminals require the row and column portions of the sequence to be in different forms. There must be a method of transforming the desired row and column numbers, supplied in the range of 0 to some maximum, into the form required by a specific terminal. Two special portions of the the "pc" value string, called escape sequences, accomplish this transformation. Each escape sequence is replaced by the row or column number in the proper form, as defined by the information in the escape sequence. The escape sequence has the following format:

$ff+<flag>[+<bias>]

where <flag> is a set of 8 bits which describes the particular operation. These bits are explained in the following table (0 is the rightmost bit):

(continued)

Bit	Value	Meaning
0	0	This is a row reference.
	1	This is a column reference.
1	0	No bias is necessary.
	1	A bias must be added to the value before use.
2	0	Use the value as is.
	1	Subtract the value from the maximum row or column before use.
3	0	Do not convert the value to BCD.
	1	Convert the value to BCD.
4	0	Do not convert the value to decimal ASCII.
	1	Convert the value to decimal ASCII.
5	***	Unused. Must be 0.
6	***	Unused. Must be 0.
7	0	Value required if bits 0-6 are not all 0.
	1	Value required if bits 0-6 are all 0.

A bias must be specified as part of the escape sequence if an only if bit 1 of the flag is set to 1. Some examples should make this mechanism clearer. Consider the following "pc" string for the ct82:

pc=$0b+$ff+$01+$ff+$80

The $0b is required by the ct82 to initiate cursor positioning. The $ff is the start of an escape sequence. Its flag of $01 means that bit 0 is 1, and all other bits are 0. Thus, the escape sequence is a row reference with no bias. The binary value is used as is. It is not converted to either binary coded decimal (BCD) or decimal ASCII. The second $ff starts the next escape sequence. Its flag of $80 means that all bits from bit 0 through bit 6 are 0. Bit 7 is 1, as it must be if all other bits are 0. This flag is a column reference (bit 0 is 0), but in all other respects it means the same thing as the previous flag.

In order to position the cursor on a ct82 terminal to row 18, column 10, a utility must send the following characters to the terminal:

Character	Meaning
$0b	Initiate cursor positioning.
$09	Column 10 (0 is the leftmost column).
$11	Row 18 (0 is the uppermost row).

As another example consider the following cursor-positioning string for an Ambassador terminal manufactured by Ann Arbor Terminals:

pc=$1b+$5b+$ff+$12+$01+$3b+$ff+$13+$01+$48

The $1b and $5b are required by this terminal to initiate cursor positioning. The $ff begins the first escape sequence. Its flag of $12 means that bits 1 and 4 are 1, and all other bits are 0. Thus, the

escape sequence is a row reference with a bias (the next number in the value list, $01) that must be added to the binary value before use. The binary value is used as is. It should be sent in decimal ASCII. The $3b following the bias is required by this terminal as a row/column separator. The second $ff starts the second escape sequence. Its flag of $13 means the same thing as a flag of $12 except that bit 0 is 1, indicating that this flag is a column reference. The $48 immediately following the bias of $01 is required by this terminal to terminate cursor positioning.

In order to position the cursor on this terminal to row 18, column 10, a utility must send the following characters to the terminal:

Character	Meaning
$1b $5b	Initiate cursor positioning.
$31 $38	Row 18 in decimal ASCII (0 relative, +1 bias).
$3b	Row/column separator.
$31 $30	Column 10 in decimal ASCII (0 relative, +1 bias).
$48	Terminate cursor positioning.

As a final example, consider the cursor-positioning string for an Infoton 100 terminal from Infoton:

pc=$1b+$66+$ff+$03+$20+$ff+$02+$20

The $1b and the $66 are required by the terminal to initiate cursor positioning. The $ff starts the first escape sequence. Its escape flag of $03 means that bits 0 and 1 are 1, and all other bits are 0. Thus, the escape sequence is a column reference with a bias of $20 that must be added to the binary value before use. The next escape sequence has the flag $02, which means that only bit 1 is 1. All other bits are 0. Thus, this escape sequence is a row reference with a bias of $20. In order to position the cursor on this terminal to row 18, column 10, a utility must send the following characters to the terminal:

Character	Meaning
$1b $66	Initiate cursor positioning.
$29	Column 10 (0 relative, +$20 bias).
$31	Row 18 (0 relative, +$20 bias).

Format of the File "ttyassoc"

The second file used by the "crt_termcap" command contains a list indicating what type of terminal is actually connected to each port on the system. This file, called "ttyassoc" for terminal association file, can be created very simply using the file "/etc/ttylist", which is

supplied on all systems. The file "ttylist" can, in fact, be used as the file "ttyassoc" if it is modified to appear exactly as described in this section. All that is required in "ttylist" is a plus or minus sign, '+' or '-', in column 1, followed by a space, followed by a two-digit number representing a terminal. The file "ttyassoc" requires two additional fields. Existing programs which use the file "ttylist" ignore anything beyond the terminal number, so the file can be modified to look like the file "ttyassoc" without affecting the rest of the system.

The file "ttyassoc" contains one line for each terminal on the system. Each line has the following format:

+ <nn> : <terminal_type> : [<name>] :

An explanation of this format follows:

<nn>	A two-digit number representing the terminal.
<terminal_type>	The type of terminal attached to the port. This name should be one of the terminals described in the file "ttycap". The name may contain as many as ten characters.
<name>	A descriptive name, used primarily for documentation. This name is normally the name of the person most commonly using the terminal. However, it has no functional meaning and need not be present.

If the file "/etc/ttylist" is used as the "ttyassoc" file, inactive ports require as an absolute minimum the entry

- nn:::

for the "crt_termcap" command to function properly. The colons used as field separators must appear even if the fields are empty.

To access the file "ttylist" as the file "ttyassoc" the user must specify "/etc/ttylist" as the second argument on the command line.

EXAMPLES

1. /etc/crt_termcap /etc/ttycap /etc/ttyassoc /etc/termcap
2. /etc/crt_termcap +d

The first example combines the information in the files "/etc/ttycap" and "etc/ttyassoc" to form the file "/etc/termcap".

The second example is equivalent to the first example.

NOTES

- Unless the user specifies the 'd' option, all three arguments must be supplied; there are no default file names. Arguments must appear in the order specified in the syntax statement.

- The user can specify any file names as the three arguments to "crt_termcap".

- All utilities which use the "termcap" functions expect the file "/etc/termcap" to exist. Although any name can be specified as the third argument to the "crt_termcap" command, if the output file is not named "/etc/termcap", it should be linked to a file that is.

ERROR MESSAGES

*** Can't access ttyassoc file "<file_name>"
The utility did not have read permissions in the file specified as the "ttyassoc" file.

*** Can't access ttycap file "<file_name>"
The utility did not have read permissions in the file specified as the "ttycap" file.

Can't find description of the terminal "<term_name>".
A terminal name specified in the "ttyassoc" file was not one of the terminal names contained in the "ttycap" file. The "termcap" file will not be created.

*** ERROR : <system_error_message> while <action>
This general class of error messages describes any system errors encountered while performing such functions as reading, writing, opening, or closing files.

Syntax: crt_termcap <ttycap_file> <ttyassoc_file> <termcap_file>
crt_termcap +d
Unless the user specifies the 'd' option, "crt_termcap" expects exactly three arguments. Otherwise, it expects no arguments. This message indicates that the argument count is wrong.

*** Unrecognized option "<char_1>", Terminal = <terminal_name>,
Last valid option "<char_2>".
The option shown is not one of the legal options allowed in the "ttycap" file. The file "termcap" will not be built.

SEE ALSO

set_termcap

date

Display or set the time and date.

SYNTAX

```
date [ [<mm>-<dd>-<yy>] <hr>:<min>[:<sec>] ]
```

DESCRIPTION

The "date" command has two forms: one with an argument and one without. Any user may execute the "date" command without an argument. In response, the system returns the current date and time. The system manager may also use the "date" command with an argument to set the date and time.

Arguments

<mm>	A number from 1 to 12 inclusive representing the month.
<dd>	A number from 1 to 31 inclusive representing the day.
<yy>	A two-digit number representing the last two digits of the year.
<hr>	A number from 0 to 23 inclusive representing the hour. (Time must be expressed as 24-hour-clock time.)
<min>	A number from 0 to 59 representing minutes past the hour.
<sec>	A number from 0 to 59 representing seconds past the minute. The default is 0.

EXAMPLES

1. date 7-13-84 15:47:28
2. date 11:53
3. date 7-13 17:5
4. date

The first example sets the date to July 13, 1984, and the time to 3:47:28 P.M.

The second example sets the time to 11:53 A.M. The date defaults to the date stored in memory.

The third example sets the date to July 13 and the time to 5:05 P.M. The value for the year defaults to the stored value, and the value for seconds defaults to 0.

The fourth example displays the date and time currently stored in memory.

ERROR MESSAGES

Error opening "/act/history": <reason>
  Whenever the system manager sets the date, the "date" command must make two entries in the file "/act/history". The operating system returned an error when "date" tried to open that file. This message is followed by an interpretation of the error returned by the operating system.

Error seeking to EOF in "/act/history": <reason>
  Whenever the system manager sets the date, the "date" command must make two entries in the file "/act/history". The operating system returned an error when "date" tried to seek to the end of that file. This message is followed by an interpretation of the error returned by the operating system.

Error writing to "/act/history": <reason>
  Whenever the system manager sets the date, the "date" command must make two entries in the file "/act/history". The operating system returned an error when "date" tried to write to that file. This message is followed by an interpretation of the error returned by the operating system.

Invalid <arg> specified.
  The value specified for the argument shown in the error message is not within the acceptable range.

Only the system manager may change the date!
  The user who tried to change the date is not the system manager.

Syntax: date [[<mm>-<dd>-<yy>] <hr>:<min>[:<sec>]]
  The syntax of the command line is incorrect. Most probably, the arguments specifying the time are missing.

SEE ALSO

history

delusr

Remove a user from the system.

SYNTAX

/etc/delusr <user_name> [+x]

DESCRIPTION

The "delusr" command removes the specified user from the system. It removes the corresponding entry from the file "/etc/log/password" and descends the user's directory tree, deleting all files and subdirectories within it except those for which the owner does not have write permission. The "delusr" command prompts for permission to delete files protected in this way. If permission to delete such a file is denied and the file is a subdirectory, "delusr" does not delete anything within that subdirectory. If the user's home directory is empty after all the deletions have been made, "delusr" also deletes that directory. Only the system manager may execute this command.

Arguments

<user_name> The name of the user to delete from the system.

Options Available

x Do not delete any files.

EXAMPLES

1. /etc/delusr chris

This example deletes the line containing the entry for the user name "chris" from the file "/etc/log/password". It also descends the directory tree beginning at the specified user's home directory and deletes all files and subdirectories for which the owner has write permission. If the owner does not have write permission, "delusr" prompts for permission to delete the file. If the home directory is empty after all the deletions are complete, "delusr" deletes that directory also.

ERROR MESSAGES

Cannot delete a user with an ID of 0 or 1.
The "delusr" command cannot delete a user whose ID is 0 or 1.

Cannot execute "remove".
"<user_name>" not removed from system.
The "remove" command, which is called by "delusr" is not in "/bin" or "/usr/bin". The command aborts without editing the password file.

Error creating "<file_name>": <reason>
The operating system returned an error when "delusr" tried to create <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Error locking password file: <reason>
The operating system returned an error when "delusr" tried to lock the password file. This message is followed by an interpretation of the error returned by the operating system.

Error opening "<file_name>": <reason>
The operating system returned an error when "delusr" tried to open <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Invalid option: '<char>'
The option specified by <char> is not a valid option to the "delusr" command.

Name must be 1 to 8 lowercase letters.
The specified user name must be between one and eight letters long. All letters must be lowercase.

Password file is locked. Try again later.
The commands "addusr", "delusr", and "password" all lock the password file so that two people cannot try to alter it at the same time. This message indicates that one of these commands currently has the password file locked.

Syntax: /etc/delusr <user_name> [+x]
The "delusr" command expects exactly one argument. This message indicates that the argument count is wrong.

"<user_name>" is not in the password file.
The file "/etc/log/password" does not contain an entry for the specified user name.

You must be system manager to run "delusr".
Only the system manager may execute the "delusr" command.

SEE ALSO

addusr
remove

devcheck

Check a device for I/O errors.

SYNTAX

/etc/devcheck <dev_name_list> [+bcdDfrsvV]

DESCRIPTION

The "devcheck" command checks the specified device for I/O errors. As it checks the device, it prints informative messages, which tell the user what part of the device is being checked. It always checks the boot sector and the system information record (SIR). By default, it also checks the fdn space, the paging space, the volume space, and, if the disk contains any, the contiguous-file space. Every time it finds a bad block, it prints a message giving the address of the block in both decimal and hexadecimal. When it is finished, "devcheck" prints a message reporting the total number of bad blocks on the disk.

The user may determine which block "devcheck" is currently testing by typing control-C. The "devcheck" command catches this interrupt and returns a message of the following format:

-- Current block: 0x<hex_address>

In order to interrupt "devcheck" the user must type control-\.

If a floppy disk contains one or more bad blocks, it should probably be discarded. If a hard disk contains one or more bad blocks, these should be removed by either running the "badblocks" program, rerunning "devcheck" with the 'b' option, or reformatting the disk with the addresses of all bad blocks placed in the file "/.badblocks" (also known as the bad-blocks file). It is wise to run this command immediately after formatting a disk.

Arguments

<dev_name> A list of the names of the devices to check. They must be block devices.

Options Available

b Put all bad blocks in the file "/.badblocks" and, unless the user also specifies the 'd' option, call the "diskrepair" command after doing so (see the 'D' option).

c Check only the contiguous-file space.

d — Do not call "diskrepair" after putting bad blocks in "/.badblocks". This option requires the ´b´ option and is incompatible with the ´D´ option.

D=<opt_list> — Specifies a list of options to use when calling "diskrepair" after putting bad blocks into the bad-blocks file. This option requires the ´b´ option and is incompatible with the ´d´ option.

f — Check only the fdn space.

r=<range> — Check only the blocks in the specified range (inclusive). The format for the range is one of the following:

[<address_1>]-<address_2>

or

<address_1>-[<address_2>]

where the user may specify the addresses in hexadecimal, decimal, or octal. The "devcheck" command interprets an address that begins with "0X" or "0x" (leading character is a zero) as a hexadecimal address; one beginning with a "0" (zero) that contains no digits greater than 7, as octal; any other address, as decimal. Thus, the following addresses are equivalent:

0X40 = 0x40 = 0100 = 64

The user should be careful not to use a leading 0 with a decimal address because "devcheck" will interpret it as an octal address if none of the digits is greater than 7.

If the user does not specify <num_1>, the default value is 0. If the user does not specify <num_2>, the default value is the address of the last block on the disk.

s — Check only the paging space.

v — Check only the volume space.

V — By default, the "devcheck" command simply tries to read each block on the disk. This procedure may fail to identify bad spots which are pattern-sensitive. If the user specifies the ´V´ option, "devcheck" performs a more rigorous test on each block. First of all, it reads the block and copies the contents to a buffer. Secondly, it writes a specific pattern (determined by the address of the block) to the block. Next, it reads and verifies that

(continued)

pattern. Finally, it copies the original data from the buffer back to the block. Obviously, using the 'V' option substantially increases the length of time needed to check the device.

EXAMPLES

1. /etc/devcheck /dev/fd0
2. /etc/devcheck /dev/fd0 +v
3. /etc/devcheck /dev/fd0 +bD=v

The first example checks the entire device in floppy drive 0 for I/O errors.

The second example checks the boot sector, the SIR, and the volume space of the device in floppy drive 0 for I/O errors.

The third example checks the entire disk in floppy drive 0 for I/O errors. All bad blocks detected are put into the bad-blocks file. If it puts any blocks into the bad-blocks file, "devcheck" calls "diskrepair" with the 'v' option.

MESSAGES

Can't read '.badblocks' file - continuing without list.
The "devcheck" command encountered an I/O error when it tried to read the file ".badblocks".

File '.badblocks' not found - continuing with check.
The device specified does not contain a file named ".badblocks", or due to damage in the logical structure of the disk, "devcheck" cannot locate the file.

ERROR MESSAGES

"<dev_name>" is not a block device.
The device specified is not a block device.

Cannot execute "badblocks".
The "devcheck" command attempted to call "badblocks" to put a bad block into the bad-blocks file. However, the "badblocks" program could not be executed. Most probably the "badblocks" program does not exist on the system disk or the user does not have permission to execute it.

Cannot execute "diskrepair".
The "devcheck" command attempted to call "diskrepair" after having put bad blocks into the bad-blocks file. However, the "diskrepair" program could not be executed. Most probably the "diskrepair" program does not exist on the system disk or the user does not have permission to execute it.

Error opening "<dev_name>" : <reason>
The operating system returned an error when "devcheck" tried to open the specified device. This message is followed by an interpretation of the error returned by the operating system.

Invalid option: '<char>'.
The option specified by '<char>' is not a valid option to the "devcheck" command.

No ".badblocks" file, 'b' option ignored.
A ".badblocks" file must exist on the disk being tested in order for the 'b' option to be valid.

Syntax: /etc/devcheck <dev_name_list> [+bcdDfrsvV]
The "devcheck" command expects at least one argument. This message indicates that the argument count is wrong.

The '<char>' option requires the 'b' option.
The option specified by <char> requires that the user also specify the 'b' option. The option specified by <char> is ignored.

SEE ALSO

badblocks
diskrepair
format

dir

List either the contents of a directory or information about a file.

SYNTAX

dir [<file_name_list>] [+abdflrsSt]

DESCRIPTION

The "dir" command is used to list either the names of the files in the specified directory or, if the argument is not a directory, information about the specified file. By default, the names of the files in a directory are listed in alphabetic order with several names per line.

Format of the Output

The information given about a file is presented on one line, which contains several fields. These fields are described in this section in the order in which they appear.

<fdn_num>	The number of the file descriptor node (fdn) which describes the file in question. This field is not present unless the user specifies the 'f' option.
<file_name>	The name of the file being described.
<size>	The size of the file in blocks. If the file is a device, "dir" places the major and minor device numbers in this field.
<file_type>	A single character specifying the type of file. The character 'b' represents a block device; 'c', a character device; 'd', a directory; 'p', a pseudoterminal; and 'z', a contiguous file. If the field is blank, the file is a regular file.
<perms>	This field, which is composed of six columns, indicates what permissions are associated with the file. The first three columns represent permissions for the user who owns the file; the last three for other users. Permissions are always presented in the order read, write, and execute. They are represented by the letters 'r', 'w', and 'x'. A hyphen in a column means that the corresponding permission is denied. For example, if the permission field contains the sequence "rwxr-x", the user who owns the file may read, write, and execute the file, whereas other users may only read and

(continued)

execute it.

<link_count> The link count is the number of directory entries which point to a file. The link count for a directory is always at least 2 because the "." entry within the directory itself points to the same fdn as the directory entry for that file in its parent directory.

<owner> The user name of the owner of the file.

<last_mod_time> The time and date at which the file was last modified.

Arguments

<file_name_list> A list of the names of files to process. The default is the working directory.

Options Available

a List all files in a directory, including those whose names begin with a period, '.'. This option has no effect if the specified file is not a directory.

b List the file size in bytes rather than blocks. This option implies the '1' option.

d If a file within the directory being processed is a directory, list the names of all the files it contains. Continue this process for all directories which are descendants of the specified directory. This option allows the user to see the entire directory structure. It has no effect if the specified file is not a directory.

f List the number of the file descriptor node for each file. This option implies the '1' option.

l If the specified file is a directory, give detailed information about each file in the directory. This option has no effect if the specified file is not a directory because in such a case the information is automatically given.

r If the specified file is a directory, reverse the order in which the files would otherwise be listed. This option has no effect if the specified file is not a directory.

s If the specified file is a directory, list one file name on each line. This option is useful for creating a file which contains the names of all the files in a directory. This option has no effect if the specified file is not a directory.

S Print a summary of the information after listing all files.

t Sort files by the time of their most recent modification. By default, the most recently modified file is listed first.

EXAMPLES

1. dir +l
2. dir /usr/jay +abdfS
3. dir memo +f
4. dir /usr/marcy +rt
5. dir /usr/marcy +s

The first example lists information about each file in the working directory whose name does not begin with a period.

The second example lists information about all files, including those whose names start with a period, in the directory "/usr/jay" (the 'f' and the 'b' option both imply the 'l' option). In addition, the command displays a list of the files in each subdirectory that is a descendant of "/usr/jay". The information includes the fdn number of each file. The size of each file is shown in bytes. At the end of the output is a summary showing the total number of directories processed, the total number of nondirectory files processed, and the total number of blocks used by all the files.

The third example displays information about the file "memo" in the working directory. The information includes the fdn number of the file.

The fourth example lists the names of those files in the directory "/usr/marcy" which do not begin with a period. The names are sorted by the time of the last modification with the sense of the sort reversed so that the most recently modified file is the last one in the list.

The fifth example lists the names of those files in the directory "/usr/marcy" which do not begin with a period. One name appears on each line.

ERROR MESSAGES

Invalid option: '<char>'.
The option specified by '<char>' is not a valid option to the "dir" command.

** Warning: directory "<dir_name>" is too large!
** Some directories were ignored
The "dir" command cannot process a file if the total number of directories in every directory between that file and the directory specified on the command line exceeds 50. In order to make the command succeed, the user should start at a lower point in the directory tree.

** Warning: directory "<dir_name>" is too large!
** Some files were ignored
The "dir" command cannot list more than 500 names of files from a single directory. In order to make the command succeed, the user should split the offending directory into two or more directories.

SEE ALSO

ls

diskrepair

Check and, optionally, repair inconsistencies in the logical structure of a disk.

SYNTAX

/etc/diskrepair <dev_name_list> [+abfmnpqruvz]

Arguments

<dev_name_list> A list of the names of the devices to check.

Options Available

- a Automatically place in the file "/.badblocks" any bad blocks encountered, and continue to run "diskrepair" until the disk is fixed.
- b Perform "blockcheck" only.
- f Perform "fdncheck" only.
- m Ignore missing blocks.
- n Do not attempt to fix errors.
- p Prompt for permission to repair.
- q Use quiet mode.
- r Rebuild the free list whether or not it is in error.
- u Report on the usage of disk blocks.
- v Use verbose mode.
- z Do not print messages about possible errors in the sizes of files.

DESCRIPTION

The "diskrepair" utility checks the structure of the disk or disks specified in <dev_name_list>. The structure of the disk refers to the layout of and the connections among files, directories, free space, paging space, and other information that makes up the file system. Any inconsistencies in the structure are reported and, optionally, repaired. Although "diskrepair" does not methodically search for and repair media (I/O) errors, it can take care of any bad blocks it encounters. If the 'a' option is in effect when "diskrepair" encounters an I/O error, it calls the utility "/etc/badblocks", which places the offending block in the file "/.badblocks".

While it is operating, "diskrepair" calls two other utilities--"blockcheck" and "fdncheck", which are both located in the directory "/etc". "Blockcheck" is concerned with the allocation of blocks on the disk. It locates problems such as duplicate blocks, missing blocks, and invalid block addresses. "Fdncheck" is concerned with the directories on the disk. It locates problems such as unreferenced files, file names

with invalid associated files, and so forth.

There are two major modes of operation, simple and verbose. The simple mode is selected by default; the verbose mode is selected by the 'v' option. In the verbose mode "diskrepair" reports all detected errors. In the simple mode it reports only those errors which require the deletion of files or of directory entries. If executed in simple mode, "diskrepair" issues a message upon completion which informs the user whether or not the disk is in need of repair.

By default, all detected errors are automatically repaired (if possible). Two options exist to alter the handling of errors. The 'n' option instructs "diskrepair" not to repair any errors. The 'p' option instructs "diskrepair" to prompt the user for permission to repair the errors it reports. In verbose mode this option causes "diskrepair" to prompt the user regarding all errors. In the simple mode, the user is prompted only for those errors which require the deletion of files or of directory entries; all other errors are automatically repaired without prompting. It should be noted that most repairs result in a loss of data. The user can generally infer which data have been lost from the messages displayed.

When using the command in simple mode (without the 'v' option), the user need not understand what types of checks are made by "diskrepair". The only decisions required are whether or not to delete the reported files. In verbose mode, much more information is given to the user. While this document is not intended to give full details of this information, the following list shows most of the inconsistencies in disk structure for which "diskrepair" checks. First, however, a few definitions are in order. A "file descriptor node" (or fdn) is an area on the disk which contains all the information the system needs about a file. There should always be one fdn per file on the disk. A UniFLEX directory entry is simply a file name and a pointer to the proper fdn. There may be multiple directory entries pointing to the same fdn (multiple names for the same file). Each pointer to an fdn is called a "link" to that file. If there is a file with no links, it is considered "unreferenced". "Out-of-range" refers to a pointer to a disk block or to an fdn which is beyond the valid number of blocks or fdns for the disk being tested. Here now, is a partial list of inconsistencies that "diskrepair" checks for.

1. Blocks duplicated in files or free list
2. Out-of-range blocks or fdns
3. Missing blocks
4. Bad free list
5. Unreferenced files
6. Inactive fdns
7. Unknown fdn type
8. Incorrect link counts
9. Incorrect free block or free fdn count
10. Invalid sizes in System Information Record

Unreferenced files are handled in one of two ways. First, an attempt is made to give the file a name by putting it into the directory "lost+found" in the root directory of the disk being tested. The name given to the file is of the form "file<fdn>", where <fdn> represents the fdn number of the file. In order for this procedure to work, the directory "lost+found" must already exist on the disk being checked, and it must have room for the entry. (This directory is created when the user makes a system disk.) Initially, the lost-and-found directory has room for thirty-two entries--including "." and "..". A user who wants to increase the capacity of the lost-and-found directory must first create enough files in it to add another block to its size, then delete the files. This procedure should not be attempted on a disk that is already damaged. If it is not possible to put the unreferenced file into the "lost+found" directory (because there is either no directory "lost+found" or no room in it), "diskrepair" deletes the file (or prompts for permission to delete it if the 'p' option was specified).

If an error is associated with an fdn, a display of pertinent data from that fdn is printed. The display includes the fdn number of the file, its size in bytes, its owner, the time of its last modification, and one of the following types:

block device
character device
contiguous file
directory
inactive
pipe
pseudoterminal
regular file
unknown

The "diskrepair" utility should generally be run only in single-user mode on an otherwise inactive system. It should never be run on an active disk. If the 'n' option is not specified (the disk may be written to), "diskrepair" attempts to unmount the disk being tested. If the device being tested is the system disk (or root device), all running tasks are suspended until "diskrepair" is complete. If the disk being tested is the system disk, and if a repair is made which requires writing to the System Information Record (block number 1), "diskrepair" stops the system upon completion and issues an appropriate message instructing the user to reboot the operating system. This procedure is necessary to prevent conflicts between the written data and similar data kept in memory.

Detailed descriptions of the options follow.

The ´a´ Option

The ´a´ option automatically places any bad blocks found by "diskrepair" in the file "/.badblocks". It also runs "diskrepair" continuously until either the disk is fixed or the program has executed ten times.

The ´b´ Option

The ´b´ option instructs "diskrepair" to run only the "blockcheck" portion of the utility. This procedure is often considerably faster, but still provides a fairly complete assessment of the validity of the disk structure.

The ´f´ Option

The ´f´ option instructs "diskrepair" to run only the "fdncheck" portion of the utility. This option is useful if a problem is suspected in the directory structure, but the result is by no means a thorough check of the structure of the disk.

The ´m´ Option

The operating system maintains a list of blocks available for use called the free list. A missing block is any block in the volume space which is not a part of any file and is not in the free list. The existence of such blocks is a harmless error in the structure of the disk. "Diskrepair" generally places these blocks in the free list. The ´m´ option, however, instructs "diskrepair" not to rebuild the free list solely on account of missing blocks. This option reduces the time required for "diskrepair" to run if missing blocks are the only problem in the free list.

The ´n´ Option

The ´n´ option tells "diskrepair" to report all errors but to make no attempt to fix them. Therefore, "diskrepair" opens the device for reading only. This option is useful for checking the structure of a disk without risking the loss of data during repairs.

The ´p´ Option

If the user specifies the ´p´ option, "diskrepair" reports each error, followed by a prompt requesting permission for the proposed repair. All prompts require an answer of either ´y´ ("yes") or ´n´ ("no").

Many repairs result in the loss of data. (The user can generally infer

what has been lost from the messages "diskrepair" displays.) Judicious use of the 'n' and 'p' options not only allows the user to assess the damage to the disk and decide which information may be sacrificed during the repair process but also provides the opportunity to try to salvage the data before repairing the disk.

The 'q' Option

This option inhibits certain warnings and messages from "diskrepair". Several conditions exist which, while not technically errors in disk structure, may cause problems. These conditions usually result in a warning message; the 'q' option inhibits them.

The 'r' Option

By default, if "diskrepair" finds that the free list is in error, it rebuilds it. The 'r' option instructs "diskrepair" to rebuild the free list whether or not it contains errors. This option is useful if the free list is known to be bad or if the user wants to reduce fragmentation within the list.

The 'u' Option

The 'u' option generates a report on the block usage of the specified device. This report is printed at the end of the "diskrepair" operation. It contains statistics on (1) the number of each type of file in the file system and the total number of files in the system, (2) the number of unused blocks and the number of used blocks, including a breakdown of how the used blocks are allocated, and (3) the number of free fdns and the number of fdns in use.

The 'v' Option

"Diskrepair" operates in one of two modes: simple or verbose. Simple mode is selected by default; verbose mode is selected by the 'v' option. In simple mode "diskrepair" reports only those errors which require the deletion of either files or directory entries. In verbose mode all errors are reported. In addition, informative messages are printed describing what phase "diskrepair" is performing.

In verbose mode the 'p' option causes "diskrepair" to prompt for permission regarding all errors. In simple mode the user is prompted only for those errors which require the deletion of either files or directory entries; all other errors are automatically repaired without prompting.

The ´z´ Option

Normally, "diskrepair" reports a possible error in the size of a file. The ´z´ option suppresses such messages. If the ´q´ option is in effect, the ´z´ option is redundant.

EXAMPLES

1. /etc/diskrepair /dev/w0
2. /etc/diskrepair /dev/w0 +n
3. /etc/diskrepair /dev/fd0 +pv
4. /etc/diskrepair /dev/fd0 +ru
5. /etc/diskrepair /dev/fd0 +mq

The first example checks the logical structure of the disk in drive w0. By default, "diskrepair" tries to fix every error it encounters. These repairs may result in the loss of data from the disk.

The second example checks the logical structure of the disk in drive w0, reports those errors which require the deletion of either files or directory entries, but performs no repairs.

The third example checks the logical structure of the disk in floppy drive 0. "Diskrepair" reports all errors it finds and prompts for permission before making any repairs.

The fourth example checks the logical structure of the disk in floppy drive 0. "Diskrepair" rebuilds the free list no matter what and prints a summary of block usage when it is finished.

The fifth example also checks the logical structure of the disk in floppy drive 0. It does not rebuild the free list solely on account of missing blocks. Neither does it print the warnings and messages that result from problems which are not technically errors in the structure of the disk but which may cause problems.

NOTES

- If a user invokes "diskrepair" on the system disk while the system is in multi-user mode, the operating system suspends all other tasks while "diskrepair" is running. In such a case any open pipes or any open files that do not have names may appear to "diskrepair" to be logical inconsistencies on the disk when, if fact, they are not. Invoking the command with the ´n´ option under these circumstances prevents "diskrepair" from altering a disk that is actually intact.

- "Diskrepair" cannot solve all the problems a disk may have. For example, it does not methodically search for and repair physical media problems (but see the ´a´ option). As for problems with the

logical structure of the disk, "diskrepair" can only repair an error if the damaged information is redundant--that is, if there is some way of determining what the information should be. It cannot, for instance, fix a badly damaged SIR; nor can it repair a disk if the root directory is severely damaged. It is therefore imperative that up-to-date backups of all important files be maintained.

ERROR MESSAGES

Blockcheck terminated abnormally.
"Blockcheck" received a program interrupt from the operating system. The user cannot determine the source of such an error; however, it is not indicative of a problem with either "diskrepair" or the device. "Diskrepair" should be rerun, for the problem may not recur.

Cannot check a "backup" disk.
The disk in the specified device was created by the "backup" command and cannot be checked by the "diskrepair" command.

Cannot read boot block.
When "diskrepair" tried to read the boot sector, it detected an I/O error.

Cannot call /etc/blockcheck.
"Diskrepair" cannot read or execute the file "/etc/blockcheck".

Cannot call /etc/fdncheck.
"Diskrepair" cannot read or execute the file "/etc/fdncheck".

Cannot read System Information Record.
The SIR is so badly damaged physically that "diskrepair" cannot read it. The user may be able to salvage some information from the disk, but must eventually reformat it.

Cannot stat root.
"Diskrepair" cannot read the fdn which describes the root directory. The user may be able to salvage some information from the disk, but must eventually reformat it.

Cannot stat std. output.
"Diskrepair" cannot read the fdn of whatever file is open as standard output. The user should rerun "diskrepair" with the terminal as standard output.

Conflicting options.
The options specified on the command line conflict with each other.

(continued)

Device is busy.
Any alterations that "diskrepair" makes must be made when the disk is not in use. Therefore, "diskrepair" determines whether or not the specified disk is mounted, and, unless the user specifies the 'n' option, it tries to unmount a mounted disk before proceeding. This error message means that either some user's working directory is on the specified disk or some user is accessing a file on that disk.

Disk needs repair!
The structure of the disk is not logically sound. The user should rerun "diskrepair" to correct the problems.

Error reading block <block_num>.
Error reading fdn <fdn_number> in block <block_num>.
Error writing block <block_num>.
Error writing fdn <fdn_num> in block <block_num>.
"Diskrepair" encountered a physical error on the disk. If the 'a' option is in effect, "diskrepair" calls the program "/etc/badblocks", which places the offending block in the file "/.badblocks". If the 'p' option is also in effect, "diskrepair" prompts the user for permission to call this utility. The user should grant permission.

If the 'a' option is not in effect, but either the 'p' or 'n' option is in effect, "diskrepair" prompts for permission to continue. If the user chooses to continue when the 'n' option is not in effect, the results are entirely unpredictable. They depend on precisely which block is damaged. Continuing with "diskrepair" may cause further damage to the disk. We suggest that the user respond negatively to the prompt to continue whenever "diskrepair" reports an I/O error and immediately rerun "diskrepair" with the 'a' option.

ERROR UPDATING SIR. DISK IS BAD!
"Diskrepair" encountered an I/O error when it tried to make the necessary changes in the SIR. The user should try again to execute "diskrepair". If the error persists, the user cannot salvage any of the data on the disk.

/etc/blockcheck is invalid.
The version of the "blockcheck" command is not the correct one.

/etc/fdncheck is invalid.
The version of the "fdncheck" command is not the correct one.

Fdncheck terminated abnormally.
"Fdncheck" received a program interrupt from the operating system. The user cannot determine the source of such an error; however, it is not indicative of a problem with either "diskrepair" or the device. "Diskrepair" should be rerun, for the problem may not recur.

Intentional system stop. Reboot UniFLEX.
If the SIR of the root device must be updated, "diskrepair" kills all tasks running on the system and locks up the system so that no new tasks can begin. It then modifies the SIR. This procedure is necessary to prevent conflicts between the written data and similar data kept in memory. After updating the SIR, "diskrepair" stops the system and prints this error message. The user must reboot the system before proceeding.

Invalid option: '<char>'.
The option specified by '<char>' is not a valid option to the "diskrepair" command.

No device specified.
The user did not specify a device on the command line.

No such device.
The user specified a nonexistent device on the command line.

Not a block device.
"Diskrepair" can only operate on block devices.

Output directed to device under test.
When testing the structure of a disk, it is impractical to try to redirect the output (the results of the test) to a file on the disk being tested. The user should reexecute "diskrepair" without redirecting the output or redirecting it to a different, mounted device.

Permission denied.
A user who executes "diskrepair" without the 'n' option must have both read and write permission on the specified device. A user who executes "diskrepair" with the 'n' option needs only read permission.

Problems encountered. Diskrepair should be rerun.
"Diskrepair" may encounter more problems than it can fix during one run. For example, it can only handle a certain number of duplicate or out-of-range blocks. If "diskrepair" cannot fix all the errors it encounters, or if it encounters an I/O error but continues operation, it prints this error message when it finishes.

Unmount error: <error_num>
"Diskrepair" encountered some problem other than a busy device when it tried to unmount the device. The accompanying error number is the number of the UniFLEX error that caused the failure. The user should consult the operating system manual for an explanation of the error.

SEE ALSO

badblocks
blockcheck
fdncheck

dperm

Set the default permissions for the creation of files by the current shell program or by tasks generated by the current shell program.

SYNTAX

dperm [<perms_list>]

DESCRIPTION

Every time a user creates a file, the operating system assigns it a set of permission bits which determines whether or not the user who owns the file and other users may read, write, or execute the file. The permissions assigned depend on the command used to create the file. The editor, for example, creates all files with "rw-rw-" permissions, which allow the user who owns the file, as well as other users, to read and write, but not execute, the file. The default permission for "crdir" are "rwxrwx"; for create, "rw-rw-"; for "makdev", "rw-r--".

The "dperm" command, which is part of the shell program, is used to set the default permissions for the creation of a file. It allows the user to instruct the system always to deny certain permissions, independent of how the file is created. It is possible to independently turn off any of the permission bits for the user who owns the file and other users. If the user specifies no arguments, the operating system removes the existing default permissions.

It is only possible to deny permissions with the "dperm" command. The "perms" command may be used to add permissions to individual files. This command overrides the defaults set by "dperm".

Arguments

<perms_list> A list defining the permission bits to be used as defaults.

Format for Arguments

<perms_list> The first character of an element in a permissions list specifies if the argument applies to the user who owns the file ('u') or to other users ('o'). The second character must be a minus sign, '-', which indicates that the following permissions are to be denied. The minus sign is followed by one, two, or three of the characters 'r', 'w', and 'x' (for read, write, and execute).

EXAMPLES

1. dperm o-rwx
2. dperm u-w o-wx
3. dperm

The first example sets the default permissions so that the operating system denies all permissions to other users whenever it creates a file.

The second example sets the default permissions so that the operating system denies write permission to the user who owns the file, and both write and execute permission to other users whenever it creates a file.

The third example removes all default permissions.

NOTES

- The "dperm" command is only effective while the shell program under which it is invoked is running. The default permissions for files created by the login shell can be permanently altered by placing the appropriate command in the file ".startup" in the user's home directory. This file is automatically executed each time the user logs in.

ERROR MESSAGES

Error in permissions specification.
The format of the permissions list is incorrect. Most likely, the user has specified a plus sign, '+', instead of a minus sign, or has used an invalid character (see Format for Arguments).

SEE ALSO

perms

dump

Send both a hexadecimal and an ASCII listing of a file to standard output.

SYNTAX

```
dump <file_name> [+i]
dump [<file_name_list>]
```

DESCRIPTION

The "dump" command sends a hexadecimal and an ASCII listing of a file to standard output. The two versions of the file appear side by side. A line of output consists of the address in the file at which that line starts, the hexadecimal contents of the byte at that address and of the following fifteen bytes, and the sequence of characters represented by these bytes. A nonprintable character appears as a period, '.', in the ASCII part of the dump.

The user may interrupt the "dump" command at any time by typing a control-C. Normally, a control-C returns the user to the shell program. However, if the "dump" command is in interactive mode and is actually dumping information when the user types a control-C, "dump" stops the output and prompts for another address.

Arguments

<file_name> The name of the file to dump. The default is standard input.

Options Available

i Enter interactive mode. The 'i' option may only be used if exactly one file name appears on the command line. If the user specifies the 'i' option, the "dump" command prompts for the address at which to begin. The address is relative to the first byte in the file, whose address is 0. An address preceded by a period is a decimal address; otherwise it is a hexadecimal address. The user may specify a single address, a range of addresses (two addresses separated by a hyphen, '-'), or an initial address and an offset (an address followed by either a comma or a space, followed by a number). In the first case, the "dump" command displays sixteen bytes of information, beginning with the specified address. In the second case, it displays all the bytes from the first to the second address inclusive. In the third case, it

begins displaying bytes at the address specified and continues for as many bytes as the following number dictates.

EXAMPLES

1. dump memo /usr/cynthia/letter
2. dump letter +i

The first example sends both a hexadecimal and an ASCII listing of the file "memo", which is the working directory, and the file "letter", which is in the directory "/usr/cynthia", to standard output.

The second example enters interactive mode and prompts the user for the address at which to begin the dumping the file "letter".

ERROR MESSAGES

Cannot interactively dump multiple files.
The 'i' option may not be used if more than one file name appears on the command line.

Cannot interactively dump standard input.
If the user specifies no file name on the command line, the default is standard input. The 'i' option may not be used in such a case.

Error opening "<file_name>": <reason>
The operating system returned an error when "dump" tried open <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Invalid option '<char>': ignored.
The option specified by <char> is not a valid option to the "dump" command. The command ignores it.

echo

Write the arguments on the command line to standard output.

SYNTAX

echo [<arg_list>] [+1]

DESCRIPTION

The "echo" command writes the arguments in <arg_list> to standard output. Every nonhexadecimal argument except the last one is followed by a space character; the last argument is followed by a carriage return.

Arguments

<arg_list> A list of arguments to write to standard output.

Format for Arguments

<arg_list> Each element in <arg_list> consists either of a string or of a hexadecimal number preceded by a plus sign, '+'. The "echo" command writes a space character after each string argument; no space appears after a hexadecimal argument.

Options Available

1 Do not write a carriage return after echoing the argument list.

EXAMPLES

1. echo This is a test!
2. echo This is a test! +7 +1 >/dev/tty03

The first example writes the string "This is a test!" to standard output, which defaults to the user's terminal.

The second example writes the string "This is a test!", followed by a bell character (hexadecimal 7), to standard output. Standard output is terminal 3. The output is not followed with a carriage return (the "+1" is the option "plus el", not the hexadecimal argument "plus one").

edit

Invoke the text editor in order to create a new text file or edit an existing one.

SYNTAX

edit [<file_name_1> [<file_name_2>]] [+bny]

DESCRIPTION

The "edit" command may be used with zero, one, or two arguments. Normally, the user specifies one argument. In such a case, "edit" opens the specified file for editing and reads as much of the file as possible into the edit buffer. At the end of an editing session of a preexisting file, the editor renames the original file by appending the letters ".bak" to its name. If this addition would result in a file name of more than fourteen characters (the maximum allowed by the operating system), the editor shortens the original name before adding the suffix. If a backup file already exists, the editor prompts for permission to delete it.

If the user specifies no arguments, the editor prompts for the name of the file at the end of the editing session, before returning control to the operating system. It does not accept the name of an existing file. If only one file name is specified, the operating system opens that file for editing, creating it first if necessary. In either case, the editor creates files with "rw-rw-" permissions.

If the user specifies two file names, the operating system makes a copy of the first file specified, gives it the name specified by the second argument, and opens it for editing. If a file with that name already exists, the editor prompts for permission to delete it before proceeding. In such a case, the editor creates the new file with the same permissions as the old file.

Arguments

<file_name_1>	The name of the file to open for editing, or, if two file names are specified, the name of the file to copy.
<file_name_2>	The name to give to the copy of the file specified by <file_name_1>. It is this copy that is opened for editing.

Options Available

b Do not save the original copy of the file as a backup file at the end of the editing session.

n Do not read any text into the edit buffer. This option allows the user to make large insertions at the beginning of a file.

y If only one argument appears on the command line, at end of the editing session automatically replace any existing backup file with the original copy of the file being edited. If two arguments appear on the command line and the second file specified already exists, delete that file at the beginning of the editing session.

EXAMPLES

1. edit /etc/log/motd +b
2. edit test +ny
3. edit test oldtest

The first example opens the message-of-the-day file for editing. At the end of the editing session the original file is deleted.

The second example opens the file "test" in the working directory for editing but does not read any of it into the edit buffer. If the file does not exist, the editor creates it. At the end of the session the "edit" command automatically replaces any existing backup file with the original copy of "test".

The third example makes a copy of the file "test", names it "oldtest", and opens it for editing. If a file named "oldtest" already exists, the editor asks for permission to delete it.

MESSAGES

Delete existing copy of new file?
The file specified by <file_name_2> already exists. If the user responds with a 'y', the editor deletes the existing copy of the file and opens the new file for editing. If the user responds with an 'n', the editor leaves the existing file intact and returns the user to the operating system.

File already exists
File name?
The "edit" command was executed with no arguments on the command line. At the end of the editing session, when the editor prompted for the name of the file, the user specified an existing file. Under these circumstances, the editor does not accept the name of an existing file.

ERROR MESSAGES

Cannot create new file
The editor cannot open the file specified by <file_name_2>. Most probably, either the user specified a path name that could not be followed or the user does not have the permissions necessary to open the file.

Cannot open edit file
The editor cannot open the file specified by <file_name_1>. Most probably, either the user specified a path name that could not be followed or the user does not have the permissions necessary to open the file.

Cannot read edit file
The editor encountered an I/O error trying to read the specified file.

Edit file does not exist
The user has specified two file names on the command line, but <file_name_1> does not exist.

New file is the same as the old file
Both <file_name_1> and <file_name_2> refer to the same file. (If their names are not the same, they are links to the same file.)

Too many file names specified.
The "edit" command requires zero, one, or two arguments. This message indicates that the argument count is wrong.

Unknown option specified
An option on the command line is not a valid option to the "edit" command. The command ignores the option and proceeds.

SEE ALSO

dperm
The UniFLEX Text Editor

end

Stop the job currently being printed by the specified printer program.

SYNTAX

end <splr_name>

DESCRIPTION

The "end" command stops the job currently being printed by the specified printer program. It discards the unprinted part of the file and starts printing the next job in the print queue, if one exists.

Arguments

<splr_name> The name of the printer program to stop.

EXAMPLES

1. end spr

This example tells the printer program associated with "spr" to stop printing the current job, to discard the rest of that file, and to start printing the next job in the print queue, if one exists.

NOTES

. The "end" command is one of five commands that are linked to the file "/etc/prcon", which controls the printing of files.

ERROR MESSAGES

Cannot find spooler directory for "<splr_name>".
The directory "/usr/gen" does not contain a directory for the specified spooler.

Error opening "<file_name>": <reason>
The operating system returned an error when "end" tried to open the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error reading "<file_name>": <reason>
The operating system returned an error when "end" tried to read the specified file. This message is followed by an interpretation of the error returned by the operating system.

"<splr_name>" is already idle.
The "end" command is effective only on an active spooler.

Syntax: end <splr_name>
The "end" command expects exactly one argument. This message indicates that the argument count is wrong.

You must be system manager to run "end".
Only the system manager may execute the "end" command.

SEE ALSO

idle
insp
next
print
pstop
purge
rerun

env

Display or change environment variables.

SYNTAX

env [<param_list>]

DESCRIPTION

The "env" command is part of the shell program. Depending on the form invoked, it displays the value of one or more environment variables, deletes the specified environment variable, changes the value of an existing environment variable, or creates a new one.

The standard environment for a shell program is established by the "login" command. Standard variables are HOME, the user's home directory; MAIL, the location of the user's mailbox; PATH, the list of directories searched by the shell program when it is looking for an executable file; SHELL, the name of the login program; TERM, the type of terminal in use; and USERNAME, the name used to log in. The user may define additional variables. A shell program passes its environment to each child task.

Arguments

<var_list> A list of the variables to manipulate.

Format for Arguments

Each item in the list of variables must have the following format:

<name>[=[<value>]]

where <name> is the name of the environment variable to manipulate and <value> is the value to assign to the specified environment variable. If the user does not specify a name, the "env" command displays the name and value of each environment variable. The value is a string. Therefore, if the string contains any space characters, it must be enclosed in single or double quotation marks. If the user specifies an equals sign, '=', but does not specify a value, the "env" command deletes from the environment the specified variable.

When setting the value of the variable PATH, the user must place a colon, ':', after the name of each directory except the last one. The user may specify the working directory either by placing an empty field in the string (by beginning the string with a colon, by ending the string with a colon, or by placing two colons side-by-side) or by using

a period, '.', in any field to represent the working directory.

EXAMPLES

1. env
2. env NC=100
3. env NC
4. env NC=
5. env PATH=:/usr/bob/bin:/usr/larry/bin:/bin:/usr/bin

The first example displays a list of each environment variable and its value.

The second example defines an environment variable "NC" and sets its value to the string "100".

The third example displays the name "NC" and its value.

The fourth example deletes the variable "NC" from the environment.

The fifth example defines the search path for the shell program as the user's working directory, followed by the directories "/usr/bob/bin", "/usr/larry/bin", "/bin", and "/usr/bin". The shell program searches the directories in the order the user specifies them.

SEE ALSO

login
UniFLEX System Calls: exece

fdncheck

Check the integrity of the structure of the file descriptor nodes (fdns) on the specified disk.

SYNTAX

```
/etc/fdncheck <dev_name>
```

DESCRIPTION

The "fdncheck" command checks the integrity of the structure of the file descriptor nodes (fdns) on the specified disk. An fdn contains all the information that the operating system needs to know about a file. This information includes, but is not limited to, the type of file, the owner of the file, the size of the file, and the addresses of all the blocks that are a part of the file. The "fdncheck" command locates problems such as unreferenced files, directory entries with invalid associated files, and so forth.

This command is primarily intended for use by the "diskrepair" utility, which calls it. It may also be used on its own. However, "fdncheck" can only check the structure of the disk; it cannot repair it. If the output from the command suggests that the structure of the fdns is damaged, the user should execute "diskrepair" on the disk.

The "fdncheck" command should be executed only when no other tasks are active on the system. Otherwise, the results are unpredictable.

Arguments

<dev_name>	The name of the device to check. It must be a block device.

EXAMPLES

1. /etc/fdncheck /dev/fd0

This example checks the structure of the fdns on the disk in floppy drive 0.

ERROR MESSAGES

Cannot check a "backup" disk.
The disk in the specified device was created by the "backup" command and cannot be checked by the "fdncheck" command.

Cannot read boot block.
When "fdncheck" tried to read the boot sector, it detected an I/O error.

Invalid option: '<char>'.
The option specified by '<char>' is not a valid option to the "fdncheck" command.

SEE ALSO

blockcheck
devcheck
diskrepair

filetype

Attempt to identify the type of the specified file.

SYNTAX

filetype <file_name_list>

DESCRIPTION

The "filetype" command attempts to identify the type of the specified file. It looks at the first few bytes of the file to determine whether or not it is a binary file. If it is, it tries to identify it as one of the types it can recognize. If it does not recognize the type, it identifies the file as an "unknown" file. If the file is not a binary file, the "filetype" command looks at the first character of each line in the first block of the file. It tallies this information and uses it to attempt to identify the file. If it does not recognize the type, it identifies the file as a "regular text" file.

The "filetype" command uses the following format to report its results to standard output:

"<file_name> is a[n] "<file_type>" file.

It sends one message for each name listed on the command line. Types of files recognized by the "filetype" command are as follows:

68000 Binary
common block binary
current BASIC compiled
nonshared, not separate I and D (instruction and data space)
nonshared, separate I and D
overlapped text and data
relative
relocatable binary
shared, not separate I and D
shared, separate I and D
standard Pascal
system Pascal
VSAM indexed

(continued)

6809 Binary
- 6809 absolute binary
- 6809 common block binary
- 6809 current BASIC compiled
- 6809 original BASIC compiled
- 6809 relative
- 6809 relocatable binary
- 6809 segmented, no text binary
- 6809 shared-text binary
- 6809 standard Pascal
- 6809 system Pascal
- 6809 VSAM indexed

Text (both 68000 and 6809)
- [assembler or] COBOL text
- BASIC precompiler text
- BASIC text
- C text
- empty
- packed
- Pascal text
- Sort/Merge parameter
- text-processor text

Special files (both 68000 and 6809)
- block device
- character device
- directory

Arguments

<file_name_list> A list of the names of files to process.

EXAMPLES

1. filetype chapter_1
2. filetype /etc/formatfd /etc/mount

The first example attempts to identify the type of the file "chapter_1" in the working directory.

The second example attempts to identify the types of the files "formatfd" and "mount" in the directory "/etc".

ERROR MESSAGES

Error opening "<file_name>": <reason>
The operating system returned an error when "filetype" tried to open the file <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Error processing "<file_name>": <reason>
The operating system returned an error when "filetype" tried to process the file <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Error reading "<file_name>": <reason>
The operating system returned an error when "filetype" tried to read the file <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Syntax: filetype <file_name_list>
The "filetype" command requires at least one argument. This message indicates that the argument count is wrong.

find

Search for a string in a file or in standard input.

SYNTAX

```
find [+bcnsu] <str_1>[&<str_2>] [<file_name_list>]
```

DESCRIPTION

The "find" command looks in the specified file and sends to standard output a list of all lines, accompanied by line numbers, that contain the specified string. By default, upper- and lowercase characters are distinct. The "find" command cannot search directories, devices, or any file it interprets as a binary file (a file which begins with a nonprintable ASCII character other than a horizontal tab character, a form-feed character, or a carriage return).

Arguments

<str_1>	The string to search for.
<str_2>	The second string to search for if the "and" operator is used.
<file_name_list>	A list of the names of files to search. The files specified should be regular text files--not directories, special files, or binary files. The default file is standard input.

Specifying a String

The user may completely specify a string or may take advantage of the matching characters recognized by the "find" command. Because some of these matching characters also have special meanings to the shell program, strings which use them must be enclosed in single or double quotation marks.

\ When used just before any matching character, including itself, the backslash character negates the matching ability of the character.

? The question mark matches any character except a new-line character.

< A left-hand angle bracket specifies that the following string must be found at the beginning of a line. It loses its matching ability if it is not the first character of the string.

> A right-hand angle bracket specifies that the preceding string must be found at the end of a line. It loses its matching ability if it is not the last character of the string.

& The "and" operator may be used between two strings (see the syntax statement). The "find" command reports only those lines on which both strings occur.

[] Square brackets enclose a list or a range of characters from which the "find" command can choose when looking for a string. A list of characters consists of adjacent characters. A range consists of two characters separated by a hyphen.

! The exclamation point may be used in conjunction with the square brackets. If it is the first character inside the brackets, the "find" command can choose from all characters not specified in the brackets when looking for a string.

Options Available

Any options used with the "find" command must appear immediately after the command name.

b Skip files whose names end in ".bak".
c Instead of writing the lines that contain the specified string to standard output, report the number of lines containing the string.
n Do not print line numbers.
s List the names of all files skipped.
u Do not distinguish between upper- and lowercase characters.

EXAMPLES

1. find +u syntax test
2. find +bu "<syntax" test trial
3. find +u ´syntax&statement´ test
4. find +c "\<" test
5. find +u ´[a-e]nd´ test
6. find +su "standard output" *

The first example writes to standard output all lines from the file "test" which contain the string "syntax". The command does not distinguish between upper- and lowercase characters.

The second example writes to standard output all lines from the files "test" and "trial" which contain the string "syntax" at the beginning of the line. The command does not search files whose names begin with ".bak", nor does it distinguish between upper- and lowercase characters. Because matching characters are used to specify the string, the string must be enclosed in either single or double quotation marks.

The third example writes to standard output all lines from the file "test" which contain both the string "syntax" and the string "statement". The command does not distinguish between upper- and lowercase characters.

The fourth example writes to standard output the number of lines in the file "test" which contain a left-hand angle bracket. The matching ability of the angle bracket is negated because of the backslash character which precedes it.

The fifth example writes to standard output all lines from the file "test" which contain any of the following strings: "and", "bnd", "cnd", "dnd", or "end". The command does not distinguish between upper- and lowercase characters.

The sixth example writes to standard output all lines from all files in the working directory which contain the string "standard output". It also writes to standard output the names of all the files in the directory that it does not search (directories, devices, and any file it interprets as a binary file). The command does not distinguish between upper- and lowercase characters.

NOTES

- The "find" command cannot search a file referenced simply as "core" although it can search one referenced as "../core".

ERROR MESSAGES

Error opening "<file_name>": <reason>
The operating system returned an error when "find" tried to open the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error processing "<file_name>": <reason>
The operating system returned an error when "find" tried to process the specified file. This message is followed by an interpretation of the error returned by the operating system.

File "<file_name>" is not a text file.
The "find" command cannot search directories, devices, or any file it interprets as a binary file (a file which begins with a nonprintable ASCII character other than a horizontal tab character, a form-feed character, or a carriage return).

Invalid option: '<char>'. Command aborted.
The option specified by <char> is not a valid option to the "find" command.

Syntax: find [+bcnsu] <str_1>[&<str_2>] [<file_name_list>]
The "find" command expects at least one argument. This message indicates that the argument count is wrong.

SEE ALSO

shell

format

Format a disk for system use.

SYNTAX

/etc/format<xx> [+BCdfFlLmMnPqrsv]

Options Available

B	Write boot sector only. Do not format disk.
C=<contig>	Reserve <contig> cylinders for contiguous files.
d=<dev_name>	Format the device <dev_name>.
f=<blocks>	Establish <blocks> blocks for file descriptor nodes (fdns).
F	Perform UniFLEX structural formatting only. Do not format the medium.
l=<file_name>	Take bad sector addresses from <file_name>.
L	Take bad sector addresses from terminal.
m=<model_code>	Use <model_code> for disk parameters.
M	Take disk parameters from terminal.
n	Do not prompt for information on disk volume.
P	Prompt for disk parameters.
q	Use quiet mode.
r=<page_space>	Reserve <page_space> cylinders for paging space.
s=<file_name>	When formatting is complete, write a list of bad blocks to the specified file.
v	Verify the disk after formatting.

DESCRIPTION

The family of "format" commands is used to prepare disks for system use. This document is a general description of all "format" commands; specific "format" commands are documented elsewhere. The specific commands generally have a suffix appended to the general name "/etc/format". They perform media formatting as well as UniFLEX structural formatting. Only the system manager may execute this command.

If the system is unable to collect the physically contiguous memory required by the hardware to format a disk, an appropriate message is issued, asking whether or not the user is willing to wait. If the user responds with an 'n', the "format" command aborts. If the user responds with a 'y', the program sleeps for a short time and then tries again. If the required memory is repeatedly unavailable, it may be necessary to go into single-user mode where the reduced number of tasks almost always yields enough memory for formatting.

Complete descriptions of the options follow.

The 'B' Option

The 'B' option instructs the "format" command to write the boot sector onto the specified disk without first formatting the disk. The system assumes that the specified disk has previously been formatted. It writes only the boot sector (sector number 0).

The 'C' Option

The 'C' option reserves space on the disk for contiguous files, which can be accessed more quickly than noncontiguous files. The formatting program tries to reserve space that is completely contiguous. However, if necessary, it allows some bad blocks to interrupt the contiguous-file space. If the disk contains a large number of bad blocks, the "format" command may not be able to honor a request for contiguous-file space. The syntax for this option is

C=<contig>

where <contig> is a decimal number which specifies the number of cylinders to reserve for contiguous files. For example, "C=20" reserves twenty cylinders for contiguous files. (For a definition of "cylinder" and other terms, see Appendix A.) The amount of contiguous-file space to reserve depends entirely on the way in which a particular site uses the operating system.

The 'd' Option

Each specific "format" command operates on a particular disk unit unless another device is explicitly specified. If this default disk has been mounted (using the "mount" command), it is automatically unmounted prior to the formatting operation. The 'd' option allows the user to specify the device to be formatted. Its syntax is

d=<file_name>

where <file_name> is the complete file specification of the desired block device.

The 'f' Option

Formatted UniFLEX disks use fdn blocks to hold information about files on the disk. Each fdn block contains eight fdns. By default, "format" uses 3% of the total disk space for fdn blocks. This default value may

be overridden by the 'f' option, which allows the user to specify the decimal number of fdn blocks to establish on the disk. At least one block must be allocated for fdns on every formatted disk.

The 'F' Option

The 'F' options instructs the "format" command to forego formatting the medium and proceed with only the UniFLEX structural formatting (the creation of both the boot sector and the system information record [SIR], the initialization of the file descriptor nodes [fdns], the creation of the root directory, and the reservation of any swap space and contiguous-file space).

The '1' and 'L' Options

If "format" detects media errors (see the 'v' option), it notifies the user of these errors and automatically omits the faulty sectors from the available space on the disk by placing them in a file called ".badblocks" in the root directory of the disk. As a result, any program which physically accesses blocks from the disk (by directly reading blocks from the device rather than by opening and accessing them from a file) receives an I/O error if it accesses any of these bad blocks. Programs like "devcheck", which do access the disk physically, are aware of the existence of the ".badblocks" file and ignore all the blocks contained in that file.

It is also possible to specify in advance a list of bad sectors which "format" should omit from the available space. "Format" places the specified sectors in the file ".badblocks", thus omitting them from the available space on the disk. This list is specified with the '1' or 'L' option. These options are most commonly used on hard disks. Hard disk manufacturers often provide a list of bad spots that have been detected on a particular drive. The '1' and 'L' options allow the user to be certain that such spots are considered bad by "format". The bad spots must be specified as logical, 512-byte sector addresses. If the manufacturer's list of bad spots is given in 40-byte ranges or in 256-byte sectors, the user should refer to Appendix B for conversion to physical sector numbers. The physical sector numbers must then be converted to logical sector numbers, which take into account the sector interleaving. Appendix C contains a table of conversions from physical to logical sector numbers.

Using the '1' or 'L' option, the user then specifies the bad sectors to "format" by means of a head address, a cylinder address, and a logical sector number. The syntax for identifying a bad sector is as follows:

<HD>/<CYL>/<SC>

where <HD> is decimal number representing the head number or surface

number (0 through (no. of heads - 1)); <CYL> is a decimal number representing the cylinder number (0 through (no. of cylinders - 1)); and <SC> is either a decimal number representing the logical, 512-byte sector number (1 through 16 or 17 for mini-Winchesters) or an asterisk, '*', which indicates all sectors on the specified surface. For example, head number 1, cylinder number 57, sector number 2, is specified as

1/57/2

For Winchester drives the user may also specify the sector as

.<num>

where <num> is decimal number representing the number of bytes from the "index position". Alternatively, the user may specify the head, cylinder, and sector as

$<hex_num>

where <hex_num> is a hexadecimal number representing the logical block-number.

These bad sector addresses may be specified from either the terminal or from an existing disk file. If they are to be specified from the terminal, the 'L' (uppercase) option should be used. "Format" then prompts the user for bad sector addresses, one per line, before beginning the formatting procedure. The list is terminated by typing a control-D as the first character of a line. If the bad sectors are to be specified from a disk file, the 'l' (lowercase) option should be used. The syntax for this option is

l=<file_name>

For example, if the file containing the bad sector addresses is called "bad-spots" and is located in the "/etc" directory, the option to use is

l=/etc/bad-spots

This file should contain one sector address per line. Each address should conform to one of the formats described previously.

The 'm' and 'M' Options

Specific parameters about disk type and size are supplied through the 'm' or 'M' (model) option. The syntax for the 'm' option is

m=<model_code>

where <model_code> is coded information about the particular drive type and size. A list of valid codes is provided with the documentation of each "format" command.

The 'M' option does not take parameters on the command line, but instead causes "format" to prompt the user for the information. The user's response should be in the same format as that described for the 'm' option. In addition, the 'M' option asks to display a list of the models it knows about and their associated codes. The user can then use this information with the current 'M' option or with 'm' options on subsequent "format" commands. If neither the 'm' nor the 'M' option is specified, the model defaults to the standard drive supplied with the hardware.

The 'n' Option

The "format" command prompts the user for a "file system name", "volume name", and "volume number" unless the 'n' option is specified to inhibit such prompts.

The 'P' Option

The 'P' option may be specified in place of the 'm' or 'M' option. It tells "format" to prompt the user for the disk parameters. These parameters include the number of cylinders, the number of heads, the cylinder number at which to begin reduced write, the cylinder number at which to begin write precompensation, and the stepping (seek) rate. All responses should be in decimal unless otherwise indicated. This option is automatically invoked if the user specifies the 'm' option with an unknown code or makes an invalid selection with the 'M' option.

The 'q' Option

Before actually starting to format the disk, "format" normally sends a prompt to the terminal to ask if the user is ready to continue. The 'q' (quiet) option suppresses this prompt. In addition, it inhibits all informative messages from "format" if no errors are encountered during formatting.

The 'r' Option

The 'r' option is used to reserve paging space on the disk. The syntax for this option is

r=<page_space>

where <page_space> is a decimal number which specifies the number of cylinders to reserve. For example, "r=20" reserves twenty cylinders for paging space. (For a definition of "cylinder" and other terms, see Appendix A.) The amount of paging space necessary varies from one system to another. The system manager must try to anticipate the maximum number of tasks that will run on the system at any one time and the size of each task. For example, thirty-two 256-Kbyte tasks require 8 Mbytes of memory. The amount of physical memory available should be subtracted from the amount of memory needed and enough paging space allocated for the difference. Appendix D, which lists the cylinder sizes of various disks, can be used to determine the number of cylinders to reserve. Note that paging space is not necessary on data disks and that by default, "format" does not reserve paging space.

The ´s´ Option

The ´s´ option instructs the "format" command to write a list of any bad blocks it encounters to the specified file after the completion of the formatting procedure. The syntax for this option is

s=<file_name>

The program specifies each bad sector as

<HD>/<CYL>/<SC>

where <HD> is a decimal number representing the head number or surface number (0 through (no. of heads - 1)); <CYL> is a decimal number representing the cylinder number (0 through (no. of cylinders - 1)); and <SC> is a decimal number representing the logical, 512-byte sector number (1 through 16 or 17 for mini-Winchesters).

The ´v´ Option

The ´v´ (verify) option instructs "format" to verify the media after formatting. If this option is specified, "format" individually verifies every sector on the disk. It first writes an arbitrary pattern to each sector; then reads and verifies each one. Because verification of a large disk may take a long time, the "format" command prints symbols to indicate its progress. It prints an asterisk, ´*´, each time it finishes writing fifty sectors; a dollar sign, ´$´, each time it finishes reading and verifying fifty sectors. It reports any sectors which fail this test to the user.

The need for the ´v´ option depends on the hardware involved and is discussed in the documentation of each specific "format" command. The option is often desirable when the user is formatting a floppy disk because floppies do not automatically verify all written data. Some Winchester disk units perform automatic verification of all written

data; others do not.

NOTES

. Examples and error messages are discussed in the documentation for each individual "format" command.

ERROR MESSAGES

*** Error writing contiguous space free map ***
The disk contains too many bad blocks for the "format" command to honor the request for contiguous-file space.

SEE ALSO

System calls: create_contiguous

To format a floppy w/ DSDD 80 cyl, 9 (512 byte) sectors
format fd0 -v -2 — adds slot 2 files

(continued)

Appendix A
Definitions of Disk Drive Terminology

The following definitions apply to this document.

Head: A device which transfers data from one disk surface. Normally there is one head per recording surface.

Track: All the sectors accessible by one head on one surface without moving the head assembly.

Cylinder: All tracks accessible by all heads without moving the head assembly. Cylinder address is the corresponding position of the head assembly.

Standard 8-inch floppy disks have 77 cylinders, numbered 0 through 76. The distinction between a track and a cylinder on a floppy is sometimes overlooked. The term track is often incorrectly used in place of cylinder.

The tables in Appendix B, C, and D may not apply to all hardware. The user should consult the documentation from the vendor.

Appendix B
Conversion to Physical Sector Numbers for Mini-Winchesters

The following tables can be used to convert a 40-byte range or 256-byte physical sector number into the 512-byte physical sector number required by "format" for mini-Winchesters. A single 40-byte range or 256-byte sector may cross a 512-byte sector boundary and therefore map onto two 512-byte sectors.

Table B-1. Conversion from 40-byte Range to 512-byte Sectors

40-byte Zone	512-byte Sector(s)
0-10	1
11-20	1-2
21-30	2-3
31-40	3
41-50	3-4
51-60	4-5
61-70	5
71-80	5-6
81-90	6-7
91-100	7
101-110	7-8
111-120	8-9
121-130	9
131-140	9-10
141-150	10-11
151-160	11
161-170	11-12
171-180	12-13
181-190	13-14
191-200	14
201-210	14-15
211-220	15-16
221-230	16
231-240	16-17
241-250	17

(continued)

Table B-2. Conversion from 256-byte Sectors to 512-byte Sectors

256-byte Sector	512-byte Sector(s)
1	1
2	1-2
3	2
4	2-3
5	3
6	3-4
7	4
8	4-5
9	5
10	5-6
11	6
12	6-7
13	7
14	7-8
15	8-9
16	9
17	9-10
18	10
19	10-11
20	11
21	11-12
22	12
23	12-13
24	13
25	13-14
26	14
27	14-15
28	15
29	15-16
30	16-17
31	17
32	17

(continued)

Appendix C
Conversion from Physical to Logical Sector Number

The following table gives the logical, 512-byte sector number that corresponds to a given physical, 512-byte sector for mini-Winchester disks. To use this table, the user must know the sector interleave used by the specific "format" command.

Table C-1. Conversion from Physical to Logical Sector Number

Physical Sector No.	Logical Sector No. for Interleave of 2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1
2	10	7	14	8	4	6	16	3	13
3	2	13	10	15	7	11	14	5	8
4	11	2	6	5	10	16	12	7	3
5	3	8	2	12	13	4	10	9	15
6	12	14	15	2	16	9	8	11	10
7	4	3	11	9	2	14	6	13	5
8	13	9	7	16	5	2	4	15	17
9	5	15	3	6	8	7	2	17	12
10	14	4	16	13	11	12	17	2	7
11	6	10	12	3	14	17	15	4	2
12	15	16	8	10	17	5	13	6	14
13	7	5	4	17	3	10	11	8	9
14	16	11	17	7	6	15	9	10	4
15	8	17	13	14	9	3	7	12	16
16	17	6	9	4	12	8	5	14	11
17	9	12	5	11	15	13	3	16	6

(continued)

Appendix D
Cylinder Sizes for Various Disks

The 'r' option requires the specification of the number of cylinders to reserve for paging space. To calculate this number the user must know how much paging space is desired and the size of each cylinder.

Mini-Winchester disks contain between 8 and 9 Kbytes of storage per track, depending on the controller. The number of tracks per cylinder is dependent on drive type and can be obtained from the hardware documentation or from the documentation of the specific "format" command. Note that the number of tracks per cylinder is the same as the number of heads in the drive. For example, consider a Computer Memories model 5619 mini-Winchester disk. This device has 306 cylinders and six heads. Twenty cylinders are required to reserve 1 Mbyte of swap space (20 cyl. * 6 tracks/cyl. * 8.5 Kbytes/track).

The following table provides the necessary cylinder sizes for floppy disks.

Table D-1. Cylinder Sizes for Various Formats of Floppy Disks

		8" Disk		5" Disk	
Sides	Density	Sectors per cyl	Kbytes per cyl	Sectors per cyl	Kbytes per cyl
Single	Single	8	4	5	2.5
Single	Double	16	8	8	4
Double	Single	16	8	8	4
Double	Double	32	16	16	8

free

Report the amount of free and used space on the specified device.

SYNTAX

free <dev_name_list> [+d]

DESCRIPTION

The "free" command reports the amount of space remaining and the on amount of space used on the specified device. It reports the total number of free blocks available for use in files, the number of blocks in use, the total number of file descriptor nodes (fdns) available, the number of fdns in use, and--if the disk was formatted with any--the amount of contiguous-file space available and the amount in use.

The number of fdns available tells the user how many more files can be created on the device (assuming that sufficient free blocks remain for use in the files). If the number of available fdns drops to 0, no more files can be created on the disk, no matter how many free blocks remain.

Arguments

<dev_name_list> A list of the names of the devices to report on. The devices may be either mounted or unmounted.

Options Available

d Provide more detailed information with the output. This extra information includes the names of the file system and the volume if they were specified when the disk was formatted, as well as the amount of swap space on the disk.

EXAMPLES

1. free /dev/fd0

This example reports both the number of fdns available and the number of free blocks on the disk in floppy drive 0.

ERROR MESSAGES

Cannot open <dev_name>
The "free" command returns this error message for any of the following reasons: the device specified does not exist; the device specified exists, but no hardware is connected to it; the device exists, hardware is connected to it, but no disk is in the device.

<dev_name> is not a block device.
The specified device must be a block device.

Invalid option: '<char>'.
The option specified by '<char>' is not a valid option to the "free" command.

hangup

Specify the action that the shell program is to take when it receives a hangup interrupt.

SYNTAX

hangup <on_or_off>

DESCRIPTION

The "hangup" command, which is part of the shell program, specifies whether or not the shell program should terminate if it receives a hangup interrupt. When hangup is "on", the shell program terminates on receipt of a hangup interrupt. When hangup is "off", the shell program ignores hangup interrupts.

Arguments

<on_or_off> Either the string "on" or the string "off".

EXAMPLES

1. hangup on
2. hangup off

The first example instructs the shell program to terminate if a hangup interrupt is received.

The second example instructs the shell program to ignore hangup interrupts.

NOTES

- The "hangup" command is only effective while the shell program under which it is invoked is running. The "on" or "off" condition is propagated to all child tasks of the shell program. That is, if hangup is "off", all child tasks of the shell program ignore the hangup interrupt unless they specifically take action to handle the interrupt. If hangup is "on", all child tasks of the shell program terminate on receiving a hangup interrupt unless they specifically take action to handle the interrupt.

SEE ALSO

shell

headset

Change information in the binary header of an executable file.

SYNTAX

headset <file_name_list> [+aAbBcCdfIStXZ]

DESCRIPTION

The "headset" command can alter certain portions of the binary header of an executable object module. Features such as whether or not the module is shared-text, whether or not the module can produce a core dump, and the initial stack size can be altered without reloading the module. The characters used for options are identical to those used when invoking the loader with the "load68k" command. Those options which do not take an argument can be disabled by preceding the character with a minus sign, '-', instead of the usual plus sign, '+'.

Arguments

<file_name_list>	A list of the names of the files to process.

Options Available

a=<num>	Specifies the minimum number of pages to allocate to this task at all times. The minimum value for the argument is 0; the maximum, 32767. The default is 0. The operating system tries to honor the specified number, but if it cannot, it uses as many pages as it needs.
A=<num>	Specifies the maximum number of pages to allocate to this task at all times. The minimum value for the argument is 0; the maximum, 32767. The default is 0. The operating system tries to honor the specified number, but if it cannot, it uses as many pages as it needs.
b=<task_size>	Specifies the maximum size to which the task may grow during execution. The argument <task_size> may be "128K", "256K", "512K", "1M", "2M", "4M", "8M", "16M", "32M", "64M", "128M", "256M", "512M", "1G", "2G", "4G", "S", "M", "L", "small", "medium", or "large". All letters may be in upper- or lowercase. The size of a task specified by 'S' (or "small"), 'M' (or

(continued)

	"medium"), or 'L' (or "large") is vendor-dependent. Typically, however, 'S' specifies 128K; 'M', the size of physical memory; 'L', the maximum size allowed to a task. The default is "128K". If the task size specified by the user (or the default) is not large enough to hold the code from all the modules being loaded, the loader automatically adjusts the size to the smallest value that can contain all the code.
B	Set a bit in the binary header of the output module which tells the operating system to zero neither the bss segment nor any memory allocated while the task is running.
c=<source_type>	Sets a flag in the binary header of the output module which indicates the type of source code from which the module was created. The argument <source_type> may be "ASSEMBLER", "C", "COBOL", "FORTRAN", or "PASCAL". The names can be specified in either upper- or lowercase.
C=<config_num>	By default, the loader uses the configuration number of the current hardware. The user may, however, use the 'C' option to specify a configuration number which overrides the default. This option is useful when loading a module for a machine other than the one on which it is running.
d	Set the "no core dump" bit in the binary header.
f	Produce a demand-load executable module (one whose text segment is loaded only as needed). In such a module the text and data segments each begin on a 512-byte boundary. Preceding the 'f' option with a minus sign instead of a plus sign removes the demand-load nature of the module. The alignment of the text and data segments on 512-byte boundaries, however, persists.
I	Enable processing of floating-point interrupts. This option is only useful on a system with an MC68881 floating-point coprocessor.
S=<hex_num>	Specifies the initial stack size, which is written into the binary header of the module produced by the loader. The hexadecimal number is the number of bytes to reserve. The default is 0, in which

	case the system assigns the default stack size of 4K.
t	Produce a shared-text executable module.
X=<add_mask>	Specifies a 32-bit hexadecimal number, of which only the high-order 7 bits are used, which is used to mask out any of the high-order 7 bits in all addresses in the file.
Z	Align the text and data segments on 512-byte boundaries, padding with null bytes as necessary. Preceding the 'Z' option with a minus sign instead of a plus sign makes the module a demand-load module if its text and data segments are already aligned on 512-byte boundaries.

EXAMPLES

1. headset mathtest +t -d +S=2000
2. headset run_1 run_2 +tB +a=10

The first example makes the executable object module "mathtest" a shared-text module. It turns off the "no core dump" bit, so that the program can produce core dumps, and sets the initial stack size to hexadecimal 2000.

The second example changes the headers in the files "run_1" and "run_2". Both modules become shared-text modules. The operating system will zero neither the bss segment nor any memory allocated while the task is running. The minimum page allocation is set to ten pages.

NOTES

- The user may make a change in a header which results in an inconsistent header. In such a case the "headset" command makes whatever adjustments are necessary in the fields which were not changed to remove the inconsistency. The user is notified of these adjustments. For example, if the user alters the initial stack size, the task size might have to be changed. If this change is necessary, "headset" notifies the user and adjusts the task size to the appropriate value. Adjustments may also be made when either the minimum or maximum page allocation is altered. If the task size specified by the user is not large enough to hold the code from all the modules being loaded, "headset" automatically adjusts the size to the smallest value that can contain all the code.

- If the user changes either the minimum or the maximum value for page allocation so that the minimum is greater than the maximum, "headset" automatically adjusts them according to the following

rules. The value for the maximum is always greater than or equal to the value for the minimum. The value for the maximum can be 0, but if it is greater than 0, it must be at least 4.

MESSAGES

File "<file_name>": changed max page allocation to <num>.
The user specified a minimum page allocation that was above the current maximum page allocation. The utility set the maximum equal to the minimum.

File "<file_name>": changed min page allocation to <num>.
The user specified a maximum page allocation that was below the current minimum page allocation. The utility set the minimum equal to the maximum.

File "<file_name>": task size set to <task_size>.
The "headset" command had to adjust the task size either because the user specified an initial stack size that made the module larger, or because the task size specified on the command was too small for the calculated size of the module.

ERROR MESSAGES

Error opening "<file_name>": <reason>
The operating system returned an error when "headset" tried to open the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error processing "<file_name>": <reason>
The operating system returned an error when "headset" tried to process the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error reading "<file_name>": <reason>
The operating system returned an error when "headset" tried to read the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error seeking in "<file_name>": <reason>
The operating system returned an error when "headset" tried to seek in the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error writing to "<file_name>": <reason>
The operating system returned an error when "headset" tried to write to the specified file. This message is followed by an interpretation of the error returned by the operating system.

File "<file_name>" is not a binary file.
The specified file does not contain a binary header.

File "<file_name>" is not a regular file.
The specified file is either a device or a directory.

File "<file_name>" is not executable.
The specified file is not an executable binary file.

Illegal configuration specified.
The configuration type must be between 0 and 255 inclusive.

Illegal hex number: <hex_num>.
The number specified is not a valid hexadecimal number.

Illegal maximum page allocation specified.
The maximum page allocation must be between 0 and 32767 inclusive.

Illegal minimum page allocation specified.
The minimum page allocation must be between 0 and 32767 inclusive.

Illegal task size specified.
The argument specified is not a valid argument to the 'b' option.

Incompatible options: 'f' and 'Z'
The 'f' and 'Z' options cannot be specified simultaneously.

Invalid option: '<char>'.
The option specified by <char> is not a valid option to the "headset" command.

Minimum page allocation greater than maximum.
Both the 'a' and 'A' options appeared on the command line, but the minimum page allocation specified was greater than the maximum.

Unknown source type specified.
The argument specified is not a valid argument to the 'c' option.

SEE ALSO

load68k
relinfo
rel68k

help

Display a brief description of the use and syntax of the specified command.

SYNTAX

help [<command_name_list>]

DESCRIPTION

The "help" command returns a brief description of the use and syntax of the specified command. To obtain this information it looks for a file in the directory "/gen/help" with the same name as the specified command. Descriptions of most UniFLEX commands are available. If the user does not specify a command name, or if the command specified is "help", the "help" command displays a list of all the commands it can help with and prompts for the name of a specific command. Typing a carriage return terminates the command.

Arguments

<command_name_list>	A list of the names of commands about which the user wants information.

EXAMPLES

1. help copy kill
2. help

The first example displays a brief description of the use and syntax of the "copy" command, followed by a brief description of the "kill" command.

The second example displays a list of all the commands that the "help" command can help with, followed by a prompt for the name of a specific command.

NOTES

. The user may add files to "/gen/help". When the "help" command is executed, it simply looks for the specified file in "/gen/help", reads the contents, and writes it to standard output.

. If the file specified is a directory, the "help" command lists the contents of the directory and asks what command the user would like help with. If the command specified is not in that directory, "help" prompts for permission to search "/gen/help".

ERROR MESSAGES

Cannot help with <command_name>.
No description of the specified command is available to the "help" command.

Error opening "<file_name>": <reason>
The operating system returned an error when "help" tried to open the file <file_name>, which describes the specified command. This message is followed by an interpretation of the error returned by the operating system.

Error reading "<file_name>": <reason>
The operating system returned an error when "help" tried to read the file <file_name>, which describes the specified command. This message is followed by an interpretation of the error returned by the operating system.

Too many files in directory.
The "help" command cannot function if the directory "/gen/help" contains more than 500 entries.

history

Display the details of recent activity on the system.

SYNTAX

history [<file_name>]

DESCRIPTION

The operating system is capable of maintaining a record of certain activities such as logging in and logging out. If a file named "/act/history" exists, this accounting procedure automatically makes an entry in it each time the system is booted or stopped, each time it goes from single-user to multi-user mode, each time the system manager sets the date, and each time any user logs in or out.

A listing of the history file is unintelligible. The "history" command must be used to extract the information from the file. The output from the "history" command has the following form:

<num_or_code> [<user_name>] <date_and_time>

If the activity being recorded is a user logging in or logging out, the first piece of information is a two-digit number corresponding to the terminal that was used. Otherwise, it is a two-letter code which indicates the type of activity involved. The codes are as follows:

bt	The system was booted.
su	The system entered single-user mode.
mu	The system entered multi-user mode.
st	The system was stopped.
bd	Time and date just before setting date.
ad	Time and date set with "date" command.

If the user was logging in, the appropriate user name appears in the second field. For all other activities, this field is empty.

The third field is the time and date at which the activity took place. The time and date are taken from the values stored in memory. When the system is first booted, these values are unlikely to be correct. The system manager should, therefore, set the date immediately after booting the system.

Arguments

<file_name>	The name of the file to use as the history file. Default is "/act/history". Any file used should have the same format as "/act/history".

EXAMPLES

1. history
2. history /act/history.584

The first example displays the details of recent activity on the system as they are recorded in the default file, "/act/history".

The second example displays the details of activity of the system as they are recorded in the file "/act/history.584". Such a file might contain the data accumulated in "/act/history" during the month of May, 1984.

ERROR MESSAGES

Error opening "<file_name>": <reason>
The operating system returned an error when "history" tried to open the file <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Error reading "<file_name>": reason
The operating system returned an error when "history" tried to read the file <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Syntax: history [<file_name>]
The "history" command expects exactly one argument. This message indicates that the argument count is wrong.

idle

Idle the specified printer program when it completes the current print job.

SYNTAX

idle <splr_name>

DESCRIPTION

The "idle" command idles the printer program associated with the specified spooler when it completes the current print job. If no job is being printed, it idles the printer program immediately. A user can still put print jobs in the print queue, but the printer program does not print them while it is idle. The idled state is useful for temporarily stopping the printer while the ribbon or paper is changed. The "next" command restarts an idled spooler.

Arguments

<splr_name> The name of the printer program to idle.

EXAMPLES

1. idle spr

This command idles the printer program associated with the spooler "spr". The printer spooler continues to accept files into the queue.

NOTES

- The "idle" command is one of five commands that are linked to the file "/etc/prcon", which controls the printing of files.

ERROR MESSAGES

Cannot find spooler directory for "<splr_name>".
The directory "/usr/gen" does not contain a directory for the specified spooler.

Error creating ".idl*splr?": <reason>
The operating system returned an error when "idle" tried to create the file ".idl*splr?". This message is followed by an interpretation of the error returned by the operating system.

Error opening "<file_name>": <reason>
The operating system returned an error when "idle" tried to open the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error reading "<file_name>": <reason>
The operating system returned an error when "idle" tried to read the specified file. This message is followed by an interpretation of the error returned by the operating system.

"<splr_name>" is already idle.
The "idle" command is effective only on an active spooler.

Syntax: idle <splr_name>
The "idle" command expects exactly one argument. This message indicates that the argument count is wrong.

You must be system manager to run "idle".
Only the system manager may execute the "idle" command.

SEE ALSO

end
insp
next
print
pstop
purge
rerun

info

Display the contents of the information field associated with the specified binary file.

SYNTAX

info <file_name_list>

DESCRIPTION

A binary file may have space, called the information field, for storing textual information associated with the file. This information includes things like the version number and release date of the file as well as other useful information pertaining to the file. The "info" command displays the contents of the information field.

Arguments

<file_name_list> A list of the names of the files for which to display the information field.

EXAMPLES

1. info /uniflex
2. info /bin/edit /usr/bin/info

The first example displays the version number, release date, and copyright information for the file "/uniflex", the operating system itself.

The second example displays version numbers, release dates, and copyright information for the UniFLEX Text Editor ("/bin/edit") and the "info" command ("/usr/bin/info").

ERROR MESSAGES

Error opening "<file_name>": <reason>
The operating system returned an error when "info" tried to open the file <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Error processing "<file_name>": <reason>
The operating system returned an error when "info" tried to process the file <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Error reading "<file_name>": <reason>
The operating system returned an error when "info" tried to read the file <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Error seeking in "<file_name>": <reason>
The operating system returned an error when "info" tried to seek to the appropriate location in <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Error writing to "standard output": <reason>
The operating system returned an error when "info" tried to write the output of the "info" command to standard output. This message is followed by an interpretation of the error returned by the operating system.

"<file_name>" has no information field.
The optional information field is not present in the specified file.

"<file_name>" is not a binary file.
The specified file lacks the header which identifies it as a binary file. The argument to the "info" command must be a binary file.

"<file_name>" is not a regular file.
The specified file is a directory or a special file (a block or character device). The argument to the "info" command must be a regular file.

Syntax: info <file_name_list>
The "info" command requires at least one argument. This message indicates that the argument count is wrong.

SEE ALSO

addinfo
UniFLEX 68000 Assembler Manual

insp

Activate a printer spooler for the specified device.

SYNTAX

/etc/insp <splr_name> [+f]

DESCRIPTION

The "insp" command activates a printer spooler. The name used as <splr_name> must be the same as the last component of a device in the directory "/dev". To activate the printer spooler, "insp" creates a background task, which runs continuously until a "pstop" command stops it. In addition, it creates a file in the spooler directory called ".mrk*splr?", which contains the task ID of the background task. The background task, which is also called the printer program, checks the contents of the spooler directory approximately every 20 seconds. It sends any files present to the printer and deletes them from the spooler directory. Of course, the original copies of the files are still intact; only the references to them in the spooler directory are deleted. Each print job is preceded by a banner page, which shows the name of the file (the name of the user followed by a number), and a form-feed character.

Only the system manager may execute this command. The command must be executed for each printer every time the system is booted. A simple way to ensure that all spoolers are activated each time the system is booted is to place the appropriate commands in the file "/etc/startup". This file is executed every time the system goes from single- to multi-user mode.

Arguments

<last_component_of_dev_name>	The last component of the name of the device for which to activate a printer spooler.

Options Available

f Suppress both the banner page and the form-feed character which are normally printed before each print job.

EXAMPLES

1. /etc/insp ppr
2. /etc/insp spr +f

The first example activates a printer spooler for the parallel printer associated with the device "/dev/ppr". By default, the printer precedes each print job with a banner page and a form-feed character.

The second example activates a printer spooler for the serial printer associated with the device "/dev/spr". The 'f' option suppresses both the banner page and the initial form-feed character.

ERROR MESSAGES

Cannot find "/usr/gen/<splr_name>".
  There is no spooler directory by the specified name.

Error creating "<file_name>": <reason>
  The operating system returned an error when "insp" tried to create the specified file. This message is followed by an interpretation of the error message returned by the operating system.

Error forking <task>: <reason>
  The operating system returned an error when "insp" tried to fork. This message is followed by an interpretation of the error message returned by the operating system.

Error opening "<file_name>": <reason>
  The operating system returned an error when "insp" tried to open the specified file. This message is followed by an interpretation of the error message returned by the operating system.

Error writing "<file_name>": <reason>
  The operating system returned an error when "insp" tried to write the specified file. This message is followed by an interpretation of the error message returned by the operating system.

File '.mrk*splr?' already exists - spooler not invoked.
  The system shut down before a "pstop" command was sent to the device in question. When this happens, whether due to an oversight in the shut-down procedure or to a system crash, the background task is interrupted, but the ".mrk*splr?" file remains in the spooler directory. The solution to this problem is to issue a "pstop" command to the appropriate device.

Syntax: /etc/insp <splr_name> [+f]
  The "insp" command expects exactly one argument. This message indicates that the argument count is wrong.

You must be system manager to run "insp".
  Only the system manager may execute the "insp" command.

SEE ALSO

end
idle
next
print
pstop
purge
rerun

int

Send a program interrupt to another task.

SYNTAX

int <task_ID_list> [+<int_num>] [+s]

DESCRIPTION

The "int" command sends the specified interrupt to the task identified by the task ID on the command line. If the user does not specify an interrupt, a termination interrupt (SIGTERM) is sent. A task ID is reported by the shell program whenever the user executes a task in the background. An ID can also be determined by the "jobs" command.

Arguments

<task_ID_list>	A list of the task IDs of the tasks to interrupt. A task ID of 0 specifies all tasks associated with the user's terminal and owned by the user.
+<int_num>	The number associated with the interrupt the user wishes to send. The plus sign, '+', is necessary to distinguish the number of the interrupt from the task ID.

Table 1 describes the possible interrupts.

(continued)

Table 1. Table of Interrupts

Name	Number	Description	A	C	D	I	R
SIGHUP	1	Hangup	+	+	-	+	+
SIGINT	2	Keyboard	+	+	-	+	+
SIGQUIT	3	Quit	+	+	+	+	+
SIGEMT	4	A-line (Axxx) emulation trap	+	+	+	+	+
SIGKILL	5	Task kill	+	-	-	-	+
SIGPIPE	6	Broken pipe	+	+	-	+	+
SIGSWAP	7	Swap error	+	+	-	-	+
SIGTRACE	8	Trace	+	+	-	+	-
SIGTIME	9	Time limit	+	+	+	-	+
SIGALRM	10	Alarm	+	+	-	+	+
SIGTERM	11	Task terminate	+	+	-	+	+
SIGTRAPV	12	TRAPV instruction	+	+	+	+	+
SIGCHK	13	CHK instruction	+	+	+	+	+
SIGEMT2	14	F-line (Fxxx) emulation trap	+	+	+	+	+
SIGTRAP1	15	TRAP #1 instruction	+	+	+	+	+
SIGTRAP2	16	TRAP #2 instruction	+	+	+	+	+
SIGTRAP3	17	TRAP #3 instruction	+	+	+	+	+
SIGTRAP4	18	TRAP #4 instruction	+	+	+	+	+
SIGTRAP5	19	TRAP #5 instruction	+	+	+	+	+
SIGTRAP6	20	TRAP #6-14 instruction	+	+	+	+	+
SIGPAR	21	Parity error	+	+	+	-	+
SIGILL	22	Illegal instruction	+	+	+	-	+
SIGDIV	23	Division by 0	+	+	+	+	+
SIGPRIV	24	Privileged instruction	+	+	+	-	+
SIGADDR	25	Address error	+	+	+	-	+
SIGDEAD	26	A child task terminated	-	+	-	+	+
SIGWRIT	27	Write to read-only memory	+	+	+	-	+
SIGEXEC	28	Data or stack space violation	+	+	+	-	+
SIGBND	29	Segmentation violation	+	+	+	-	+
SIGUSR1	30	User-defined interrupt #1	+	+	-	+	+
SIGUSR2	31	User-defined interrupt #2	+	+	-	+	+
SIGUSR3	32	User-defined interrupt #3	+	+	-	+	+
SIGABORT	33	Program abort	+	-	-	-	+
SIGSPLR	34	Spooler signal	+	+	-	+	+
SIGINPUT	35	Input is ready	+	+	-	+	+
SIGDUMP	36	Take memory dump	0	+	+	+	+
	37-41	System-defined interrupts					
SIGUNORDERED	42	MC68881 branch or set on unordered operand	+	+	-	+	+
SIGINEXACT	43	MC68881 inexact result	+	+	-	+	+
SIGFPDIVIDE	44	MC68881 division by 0	+	+	-	+	+
SIGUNDERFLOW	45	MC68881 underflow	+	+	-	+	+
SIGOPERAND	46	MC68881 invalid operand	+	+	-	+	+
SIGOVERFLOW	47	MC68881 overflow	+	+	-	+	+
SIGSNAN	48	MC68881 signaling not-a-number	+	+	-	+	+
	49-63	Vendor-defined interrupts					

(continued)

Notes: A = Default state is "abort" (otherwise, "ignore")
C = Interrupt can be caught
D = Produces a core dump
I = Interrupt can be ignored
R = Resets to default state when triggered
0 = See NOTES

Options Available

s Send a soft interrupt to the specified task. A soft interrupt tells the operating system not to wake a sleeping task unless the task was put to sleep by the system call, "sys stop". In such a case, the system wakes the task and delivers the specified interrupt. Otherwise, the interrupt remains pending until another event wakes the task.

EXAMPLES

1. int 263
2. int +5 149
3. int 149 +5
4. int 313 314 +30 +s

The first example sends a termination interrupt (SIGTERM) to task number 263.

The second example sends a SIGKILL interrupt to task number 149. No program can trap or ignore a SIGKILL interrupt.

The third example is identical to the second one. The order of the arguments is irrelevant.

The fourth example sends a soft user-defined interrupt to tasks 313 and 314. If either task is sleeping, the interrupt to that task remains pending until another event wakes the task (unless the task was put to sleep by the system call "sys stop", in which case the system wakes the task and delivers the interrupt).

MESSAGES

[Soft] interrupt <int_num> sent to task <task_ID>.
The "int" command sends this message to standard output each time it successfully sends an interrupt to a task.

ERROR MESSAGES

Error sending [soft] interrupt <int_num> to task <task_ID>: <reason>
The operating system returned an error when "int" tried to send the interrupt. This message is followed by an interpretation of the error returned by the operating system.

Illegal interrupt specified: <int_num>
The number specified must be an integer between 1 and the number of signals, inclusive. At the time of this writing the number of signals is 63.

Illegal task ID specified: <task_ID>
The task ID specified contains some characters that are not digits. A legal task ID contains only digits.

Syntax: int <task_ID_list> [+<int_num>] [+s]
The "int" command expects at least one task ID and no more than one interrupt number. This message indicates that the argument count is wrong.

SEE ALSO

jobs

jobs

Report the task IDs and starting times of all background tasks originated by the user from the current shell program.

SYNTAX

jobs

DESCRIPTION

The "jobs" command, which is part of the shell program, reports the task IDs and starting times of all background tasks originated by the user from the current shell program. The task IDs are preceded by the letter 'T', for "task". This letter is not part of the task ID.

EXAMPLES

1. jobs

This example is the only valid form of the "jobs" command. It reports the task ID and starting time of all active background tasks originated by the user from the current shell program..

ERROR MESSAGES

No tasks active.
The user has no active tasks in the background.

SEE ALSO

int

kermit

Transfer a file from one machine to another.

SYNTAX

```
kermit cl[te] <dev_name> <esc_char>
kermit r[l][dit] <dev_name>
kermit s[l][dit] <dev_name>
```

Arguments

c	Connect the two machines. The letters 'c', 'r', and 's' each specify the mode of operation for "kermit". The user may specify only one of these letters.
r	Receive a file from another machine. The letters 'c', 'r', and 's' each specify the mode of operation for "kermit". The user may specify only one of these letters.
s	Send a file to another machine. The letters 'c', 'r', and 's' each specify the mode of operation for "kermit". The user may specify only one of these letters.
<dev_name>	The name of the device to use to communicate with the other machine. By default, "kermit" uses "/dev/modem". If the user specifies the 't' option, the argument <dev_name> must appear on the command line. If the 'e' option is also present, <dev_name> and <esc_char> should appear in the same order as the 't' and 'e' options. If the user does not specify the 't' option, this argument should not appear on the command line.
<esc_char>	The ASCII character to use as the first character of the escape sequence for breaking a "kermit" connection. When the user types the escape character, "kermit" holds it until it receives the next character. If that character is a 'c' or a 'C', "kermit" closes the connection between machines. If it is another escape character, "kermit" passes a single escape character to the receiving machine. Any other character causes "kermit" to send a bell (control-G) to the sending terminal and to wait for another character. The default escape character is '^'.

(continued)

If the user specifies the 'e' option, the argument <esc_char> must appear on the command line. If the 't' option is also present, <esc_char> and <dev_name> should appear in the same order as the 't' and 'e' options. If the user does not specify the 't' option, this argument should not appear on the command line.

<file_name_list> A list of the names of the files to transfer.

Options Available

d Provide debugging information. The user may specify this option once, twice, or three times. Each repetition of the letter tells "kermit" to supply more detailed debugging information.

i By default, "kermit" maps a carriage return (hexadecimal 0D) to a carriage return followed by a line-feed character (hexadecimal 0A) on output, discards line-feed characters on input, and masks all characters to 7 bits. When the user specifies the 'i' option, "kermit" does not alter a carriage return on output, does not discard line-feed characters on input, and sends or receives all 8 bits of each character.

l Execute on the local machine--that is, on the machine to which the terminal in use is connected. When running locally "kermit" communicates with the other machine through "/dev/modem" or the device specified with the 't' argument. By default, "kermit" runs on the remote machine and communicates with the local machine through the standard I/O channels. The 'l' option is mandatory with the 'c' argument. Without it, "kermit" terminates.

DESCRIPTION

The "kermit" command provides a protocol for moving files from one UniFLEX system to another. It is provided with both the 6809 and 68xxx operating systems to facilitate the process of upgrading to a 68xxx system. Although "kermit" is not infallible, it does go to great lengths to ensure that data transfer is error-free.

In order to use "kermit", the user must have a physical connection between two machines. Normally, a terminal port from one machine is connected to a terminal port on the other machine. The connection can be a simple cable or a more complex arrangement including a modem on each end. Except for the case of the remote machine in a transfer in login mode, the terminal port should not be enabled for login.

The "kermit" command can operate in local or remote mode. When the program is in local mode, all communication with the other machine takes place through a port other than the user's terminal. When it is in remote mode, communication takes place through the user's terminal.

EXAMPLES

The "kermit" command supports two methods of transferring files: "direct" and "login". To use direct mode the user must be able to execute commands on two separate terminals--one connected to the sending machine; one, to the receiving machine. In direct mode, communication normally takes place through the device "/dev/modem". The user may override this choice of device with the 't' argument, but whatever device is to be used must be linked to the serial port that is connected to the other machine. Thus, if the connection runs from the port associated with "/dev/tty01" on machine A to the port associated with "/dev/tty02" on machine B, the user must execute the following command on machine A:

link /dev/tty01 <comm_dev>

and the following command on machine B:

link /dev/tty02 <comm_dev>

where <comm_dev> is the name of the device to use for communication, usually "/dev/modem".

The terminal configuration of the two ports linked to the communication device must be identical. If the default configurations or the hardware values are not the same, the user must use the "ttyset" command on one or both machines to make them identical.

Once the configurations are the same and the appropriate links have been made as just described, the user is ready to begin transferring files. The first step is to invoke the command

kermit rl

on the machine that is to receive the files. This command tells "kermit" to receive a file in local mode. If the file does not begin to arrive withing approximately 1 minute, this command "times out" and terminates. Thus, the user has a little less than a minute to execute the following command on the sending machine:

kermit sl <file_name_list>

When the transfer begins and when it ends, "kermit" sends an appropriate message to each machine. When the transfer is complete, "kermit" terminates at both ends.

A user who does not have direct physical access to a terminal on each machine must transfer files in "login" mode. In this mode the terminal port used for communication on the remote machine must be enabled for "login". A port is so enabled when the first character of the entry corresponding to that port in the file "/etc/ttylist" is a plus sign, '+'. This machine must be in multi-user mode. It is not necessary to create a link for communication as communication from the remote machine takes place over the line to the remote terminal. The port used for communication on the local machine must be disabled for "login". It may operate in either multi- or single-user mode. The link between the communication device and the terminal being used must be established just as in direct mode, and the configuration of the terminal must match the configuration of the terminal being used at the remote machine.

To establish the connection between machines, use the following command on the local machine:

```
kermit cl
```

This command tells the operating system to establish the connection between the two machines, so that the user can invoke commands on the remote machine from the terminal connected to the local machine. Once "kermit" establishes this connection, the terminal on the local machine behaves as if it were plugged in directly to the remote machine. The operating system sends everything typed on the terminal (with the exception of the escape character) to the remote machine; any output from the remote machine appears on the terminal from which the "kermit cl" command was invoked.

The first thing to do is to log in to the remote machine from the local terminal. It is not necessary to create any links or to alter the configuration of the local terminal. To transfer a file from the remote to the local machine, the user begins by invoking the following command:

```
kermit s <file_name>
```

which tells the operating system to send the specified file from the remote machine to the local machine. The absence of the '1' option specifies that communication is to take place through the terminal currently in use. The system waits approximately 30 seconds before beginning to send the file. If no "kermit" is invoked at the receiving end within approximately 60 seconds after transmission begins, the "kermit" on the remote machine "times out" and terminates.

In order to invoke "kermit" at the receiving end, the user must break the "kermit" connection by typing the escape character (´^´ by default), followed by a ´c´ or a ´C´. When the UniFLEX prompt returns, the command to invoke the receiving "kermit" can be executed:

```
kermit rl
```

NOTES

- The "kermit" command is supplied as is, with no support from Technical Systems Consultants. Problems with the program can be reported to us, and we will forward them to the party who supplied it to us.

- Following is a statement of the policy on commercial use and distribution of KERMIT from the Columbia University Center for Computing Activities:

  The KERMIT file transfer protocol has always been open, available, and free to all. The protocol was developed at the Columbia University Center for Computing Activities, as were the first several KERMIT programs. Columbia has shared these programs freely with the worldwide computing community since 1981, and as a result many individuals and institutions have contributed their own improvements or new implementations in the same spirit. In this manner, the number of different systems supporting KERMIT implementations has grown from three to about sixty in less than three years. If Columbia had elected to keep the protocol secret, to restrict access to source code, or to license the software, the protocol would never have spread to cover so many systems, nor would the programs be in use at so many sites, nor would the quality of many of the implementations be so high.

  Although KERMIT is free and available to anyone who requests it, it is not in the "public domain". The protocol, the manuals, the Columbia implementations, and many of the contributed implementations bear copyright notices dated 1981 or later, and include a legend like

  > Permission is granted to any individual or institution to copy or use this document and the programs described in it, except for explicitly commercial purposes.

  This copyright notice is to protect Kermit, Columbia University, and the various contributors from having their work usurped by others and sold as a product. In addition, the covering letter which we include with a KERMIT tape states that KERMIT can be passed along to others; "we ask only that profit not be your goal, credit be given

where it is due, and that new material be sent back to us so that we can maintain a definitive and comprehensive set of KERMIT implementations."

Within this framework, it is acceptable to charge a reproduction fee when supplying KERMIT to others. The reproduction fee may be designed to recover costs of media, packaging, printing, shipping, order processing, or any computer use required for reproduction. The fee should not reflect any program or documentation development effort, and it should be independent of how many implementations of KERMIT appear on the medium or where they came from. It should not be viewed as a license fee. For instance, when Columbia ships a KERMIT tape, there is a $100.00 reproduction fee which includes a 2400' reel of magnetic tape, two printed manuals, various flyers, a box, and postage; there is an additional $100.00 order processing charge if an invoice must be sent. The tape includes all known versions of KERMIT, including sources and documentation.

Commercial institutions may make unlimited internal use of KERMIT. However, a question raised with increasing frequency is whether a company may incorporate KERMIT into its products. A hardware vendor may wish to include KERMIT with its standard software. A software house may wish to incorporate KERMIT protocol into its communications package, or to distribute it along with some other product. A timesharing vendor or dialup database may wish to provide KERMIT for downloading. All these uses of KERMIT are permissible, with the following provisos:

> A KERMIT program may not be sold as a product in and of itself. In addition to violating the prevailing spirit of sharing and cooperation, commercial sale of a product called KERMIT would violate the trade mark which is held on that name by Henson Associates, Inc., creators of The Muppet Show.
>
> Existing KERMIT programs and documentation may be included with hardware or other software as part of a standard package, provided the price of the hardware or software product is not raised significantly beyond costs of reproduction of the KERMIT component.
>
> KERMIT protocol may be included in a multi-protocol communication package as one of the communication options, or as a communication feature of some other kind of software package, in order to enhance the attractiveness of the package. KERMIT protocol file transfer and management should not be the primary purpose of the package. The price of the package should not be raised significantly because KERMIT was included, and the vendor's literature should make a statement to this effect.

(continued)

> Credit for development of the KERMIT protocol should be given to the Columbia University Center for Computing Activities, and customers should be advised that KERMIT is available for many systems for only a nominal fee from Columbia and from various user group organizations, such as DECUS and SHARE.

Columbia University holds the copyright on the KERMIT protocol, and may grant permission to any person or institution to develop a KERMIT program for any particular system. A commercial institution that intends to distribute KERMIT under the conditions listed above should be aware that other implementations of KERMIT for the same system may appear in the standard KERMIT distribution at any time. Columbia University encourages all developers of KERMIT software and documentation to contribute their work back to Columbia for further distribution.

Finally, Columbia University does not warrant in any way the KERMIT software nor the accuracy of any related documentation, and neither the authors of any KERMIT programs or documentation nor Columbia University acknowledge any liability resulting from program or documentation errors.

These are general guidelines, not a legal document to be searched for loopholes. To date, KERMIT has been freely shared by all who have taken the time to do work on it, and no formal legalities have proven necessary. The guidelines are designed to allow commercial enterprises to participate in the promulgation of KERMIT without seriously violating the KERMIT user community's trust that KERMIT will continue to spread and improve at no significant cost to themselves. The guidelines are subject to change at any time, should more formal detail prove necessary.

Commercial organizations wishing to provide KERMIT to their customers should write a letter stating their plans and their agreement to comply with the guidelines listed above. The letter should be addressed to:

KERMIT Distribution
Columbia University Center for Computing Activities
812 West 115th Street
New York, NY 10025

kill

Delete the specified file name from the file system.

SYNTAX

kill <file_name_list> [+dlpqs]

DESCRIPTION

The "kill" command deletes the specified file name from the file system. If the file is one whose link count is 1 (i.e., the file has no other links), the file is deleted and all information is lost. If the file is one that has multiple links, only the named link is deleted.

If the file is an empty directory (containing only the special directories "." and "..") and the user specifies the 'd' option, the directory is deleted. The "kill" command cannot delete a nonempty directory. The user must have write permission in the directory containing the file being deleted.

Arguments

<file_name_list> A list of the names of the files to delete from the file system.

Options Available

- d Even if the specified file is a directory, delete it. The directory must be empty.
- l List the name of each file as it is deleted.
- p Prompt the user for permission before deleting each file.
- q Do not report any errors.
- s Do not proceed to the next file name in the list if an error is encountered.

EXAMPLES

1. kill test_file
2. kill test* +pd
3. kill test* +q
4. kill test* +s

The first example deletes the file "test_file" from the file system if it is not a directory.

The second example prompts for permission to delete each file in the working directory whose name begins with the characters "test". If the user responds to the prompt with anything but a 'y' or a 'Y', the file in question remains intact. If the user responds with a 'y' or a 'Y' ("yes"), the file is deleted from the file system--as long as it is not a nonempty directory. The command prints the name of each file as it is deleted.

The third example deletes all nondirectory files in the working directory whose names begin with the characters "test". No prompts are given; no names are listed; no errors are reported. To learn the results of the command the user must list the contents of the working directory and compare it with the contents before the command was issued.

The fourth example begins to delete all nondirectory files in the working directory whose names begin with the characters "test". The command aborts as soon as it encounters any error. Because error messages are not suppressed, the user can tell whether or not the command succeeded.

NOTES

- The "kill" command guards against deleting the user's working directory. However, it cannot guard against deleting another user's working directory.

ERROR MESSAGES

Entry does not exist: <file_name>
The user tried to delete a nonexistent file.

Entry is a directory: <file_name>
The "kill" command cannot delete a directory unless the user specifies the 'd' option. This message only occurs when exactly one argument is given to the "kill" command. If more than one argument is specified, the message is not printed, but directories remain intact.

May not delete a special directory: <dir_name>
The "kill" command cannot delete the directories "." and "..".

May not delete nonempty directory: <dir_name>
The specified file is a nonempty directory. The "kill" command cannot delete a directory unless it is empty (contains just "." and "..").

May not delete the working directory: <dir_name>
The specified file is the user's working directory, which may not be deleted.

Path cannot be followed: <file_name>
One or more of the directories which make up the name of the file do not exist.

Permissions deny deleting file: <file_name>
The directory containing <file_name> is write protected and, therefore, may not be modified. The specified file cannot be deleted.

Syntax: kill <file_name_list> [+dlpqs]
The "kill" command expects at least one argument. This message indicates that the argument count is wrong.

Unknown option: <char>
The option specified by <char> is not a valid argument to the "kill" command.

SEE ALSO

remove

lib-gen68k

Create a new library or update an existing one.

SYNTAX

lib-gen68k o=<old_lib> n=<new_lib> [u=<update>] [<del_list>] [+al]

DESCRIPTION

The "lib-gen68k" command either creates a new library of relocatable or executable modules or updates an existing library. Each module in a library must have a name. The name is assigned to a module by either the "name" directive in the relocating assembler or the 'N' option of the linking-loader. The "lib-gen68k" command does not accept a module without a name. As it runs, "lib-gen68k" produces a report describing the action that it takes for each module in the library. The report includes the name of the module and the file from which it was read (the old library or one of the update files).

Arguments

o=<old_lib> The name of an existing library file which was previously created by the "lib-gen68k" command. If "lib-gen68k" is being called to create a new library, rather than to update an existing one, this argument is inappropriate. The 'n' argument, the 'o' argument, or both must appear on the command line.

n=<new_lib> The name of a new library. If a file with this name already exists, "lib-gen68k" deletes it without warning before writing the new library. If the user does not specify a name for the new library, it defaults to the name of the old library. In such a case "lib-gen68k" puts the new library in a scratch file, deletes the old library, and renames the scratch file with the name of the old library. The 'n' argument, the 'o' argument, or both must appear on the command line.

u=<update> The name of a file containing modules to add to the library. Modules of the same name are replaced by modules from the update file. The user may specify up to 255 update files by repeating the "u=<update>" argument for each one. However, when updating a large number of modules, it may be simpler to concatenate the modules by listing all of them with the "list" command and redirecting the output to a file. The user can

then update all the modules with only one invocation of the 'u' option rather than having to invoke it for each module.

<del_list> A list of the names of modules to delete from the old library.

Options Available

a Produce an abbreviated report, which contains information only about modules that were replaced, added, or deleted.

l Suppress the production of a report.

EXAMPLES

1. lib-gen68k n=binlib u=one u=two u=three
2. lib-gen68k o=binlib u=new +a
3. lib-gen68k o=binlib u=newmods n=newlib transpose add +l

The first example creates a new library named "binlib" which contains all the modules from the files "one", "two", and "three".

The second example updates the library "binlib" by adding or replacing modules from the file "new". The command produces an abbreviated report.

The third example updates the library "binlib" by adding or replacing modules from the file "newmods" and by deleting the modules named "transpose" and "add". The updated library is written to the file "newlib". The old library is deleted.

ERROR MESSAGES

An old or new library name must be specified.
The 'n' argument, the 'o' argument, or both must appear on the command line.

Invalid option: '<char>'.
The option specified by '<char>' is not a valid option to the "lib-gen68k" command.

No index found in <lib_name>
The "lib-gen68k" command creates every library with an index. This message indicates either that the file specified is not a library or that it is a library, has been badly damaged, and can no longer be used.

Record not found in <module_name>
One of the files in the list of names of modules to delete from the old library was not found in that library. The command ignores that file name and continues.

Record with no name found in <module_name>
Every relocatable or executable module that goes into a library must have a name. The user should remake the specified module and give it a name.

Unknown argument: <str>
The argument specified by <str> is not a valid argument to the "lib-gen68k" command.

Unrecognizable record in <module_name>
All modules in a library must be either executable or relocatable.

SEE ALSO

68xxx UniFLEX Relocating Assembler and Linking Loader

libinfo

Display information about a library.

SYNTAX

libinfo <library_name_list> [+emM]

DESCRIPTION

The "libinfo" command produces a list of the entry points and module names contained in a library produced by the "lib-gen68k" command. The user can optionally display only the entry points or only the module names. Information about a particular module within a library can also be displayed.

Arguments

<library_name_list>	A list of the names of the libraries to report on.

Options Available

e	Display only entry points in the specified library.
m	Display only module names in the specified library.
M=<mod_name>	Display information about module <mod_name>.

EXAMPLES

1. libinfo testlib
2. libinfo runlib +m
3. libinfo /lib/mathlib +M=Arctan

The first example lists all entry points and module names in the library "testlib".

The second example lists all the module names contained in the library "runlib".

The third example displays the entry points and module names in the module "Arctan" in the library "/lib/mathlib".

(continued)

NOTES

. The ´M´ option is incompatible with both the ´e´ and ´m´ options. If the user specifies incompatible options, "libinfo" uses the ´M´ option and ignores any others.

ERROR MESSAGES

Error opening "<file_name>" : <reason>
The operating system returned an error when "libinfo" tried to open the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error reading "<file_name>" : <reason>
The operating system returned an error when "libinfo" tried to read the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error seeking to <location> in "<file_name>" : <reason>
The operating system returned an error when "libinfo" tried to seek to the specified location (in hexadecimal) in the specified file. This message is followed by an interpretation of the error returned by the operating system.

"<file_name>" is not a library!
The file specified does not have the correct format for a library created with the "lib-gen68k" command.

Invalid option: ´<char>´.
The option specified by ´<char>´ is not a valid option to the "libinfo" command.

*** ´M´ taken, others ignored ***
The ´m´ and ´e´ options are incompatible with the ´M´ option. If the user specifies incompatible options, "libinfo" uses the ´M´ option and ignores any others.

SEE ALSO

lib-gen68k
relinfo

link

Establish a new link to an existing file.

SYNTAX

link <file_name_1> <file_name_2>

DESCRIPTION

The "link" command establishes a new link to an existing file. If the command is successful, both <file_name_1> and <file_name_2> refer to the same file.

A link cannot cross a volume boundary. A user must have write permission in the parent directory in which the new link is created; however, write permission in the directory containing the original copy of the file is unnecessary. Only the system manager may make a link to a directory.

Arguments

<file_name_1>	The name of the existing file to which to establish a link.
<file_name_2>	The name to link to the existing file.

EXAMPLES

1. link /usr/susan/.editconfigure .editconfigure

This example creates a file named ".editconfigure" in the user's working directory and links it to the existing file ".editconfigure" in the directory "/usr/susan".

ERROR MESSAGES

Cannot link across devices
: The specified file names reside on different volumes and, therefore, cannot be linked.

Entry already exists: <file_name_2>
: The "link" command cannot link an existing file to another file. The file specified by <file_name_2> must be a nonexistent file.

Entry does not exist: <file_name_1>
: If the file to which the link is to be made does not exist, it is impossible to link the files.

Entry is a directory: <file_name_1>
The existing file specified is, in fact, a directory, not a regular file. Only the system manager can link to a directory.

Invalid option: '<char>'.
The option specified by '<char>' is not a valid option to the "link" command.

Path cannot be followed: <file_name>
One or more of the directories which make up the name of the file do not exist.

Permissions deny access: <file_name>
The user does not have permission to access the specified file. If the file is the existing file, <file_name_1>, the user does not have execute permission in the parent directory. If the file is <file_name_2>, the user does not have write permission in the parent directory.

Syntax: link <file_name_1> <file_name_2>
The "link" command expects exactly two arguments. This message indicates that the argument count is wrong.

SEE ALSO

copy
move

list

Write the contents of the specified file to standard output.

SYNTAX

list [<file_name_list>] [+1<num>]

DESCRIPTION

The "list" command writes the contents of the specified file to standard output. If the user specifies more than one file, the files are listed one after the other with no space between them.

The default file name is standard input. A plus sign, '+', may also be used as an argument to indicate standard input.

Arguments

<file_name_list>	A list of the names of the files to write to standard output. The default is standard input.

Options Available

1	Include line numbers in the listing.
<num>	The number of the line at which to begin listing the file.

EXAMPLES

1. list test
2. list test +120 >>test.out
3. list part_1 part_2 + part_3 >whole_thing

The first example writes the file "test" to standard output, which defaults to the user's terminal.

The second example also writes the file "test" to standard output. However, in this case standard output is redirected so that the listing is appended to the contents of the file "test.out". The listing is accompanied by line numbers and starts at line 20 of the file.

The third example writes the files "part_1" and "part_2", followed by the text entered from standard input, followed by "part_3", to the file "whole_thing".

ERROR MESSAGES

Error listing "<file_name>": <reason>
The operating system returned an error when "list" tried to write <file_name> to standard output. This message is followed by an interpretation of the error returned by the operating system.

Error opening "<file_name>": <reason>
The operating system returned an error when "list" tried to open the file <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Error reading "<file_name>": <reason>
The operating system returned an error when "list" tried to read the file <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Invalid option: '<char>'. Command aborted!
The option specified by <char> is not a valid option to the "list" command.

Invalid starting line number. Command aborted!
The string used to specify the number of the starting line of the listing either is not a string of digits or is too large.

SEE ALSO

more
page

load68k

The "load68k" command is the 68xxx UniFLEX linking-loader.

SYNTAX

load68k <file_name_list> [+aAbBcCdDefFiI1LmMnNoPqrRsStTuUwWxXyYZ]

DESCRIPTION

The "load68k" command accepts as input one or more relocatable binary modules and produces as output either a relocatable module or an executable module. The relocatable modules used as input must have been produced by the relocating assembler or the linking-loader. Options are available for producing load and module maps as well as a global symbol table. Starting addresses for text and data segments can be adjusted for the particular hardware being used. The page size can also be adjusted. The loader can search libraries produced by the UniFLEX "lib-gen68k" utility in order to resolve external references.

The user can place all desired options in a file rather than specifying them individually on the command line. The operating system comes with one such file, the file "/lib/std_env", which describes the environment of the hardware in use. The loader always reads this file before processing any other options. It then processes options in the order in which they appear on the command line. If an option is specified more than once (e.g., once in a file and once on the command line), the last specification overrides all others.

Arguments

<file_name_list> A list of the names of the files to load.

Options Available

Those options which do not take an argument can be disabled by preceding the character with a minus sign, '-', instead of the usual plus sign, '+'.

a=<num> Specifies the minimum number of pages to allocate to this task at all times. The minimum value for the argument is 0; the maximum, 32767. The default is 0. The operating system tries to honor the specified number, but if it cannot, it uses as many pages as it needs.

A=<num> Specifies the maximum number of pages to allocate to this task at all times. The minimum value for the argument is 0; the

	maximum, 32767. The default is 0. The operating system tries to honor the specified number, but if it cannot, it uses as many pages as it needs.
b=<task_size>	Specifies the maximum size to which the task may grow during execution. The argument <task_size> may be "128K", "256K", "512K", "1M", "2M", "4M", "8M", "16M", "32M", "64M", "128M", "256M", "512M", "1G", "2G", "4G", "S", "M", "L", "small", "medium", or "large". All letters may be in upper- or lowercase. The size of a task specified by 'S' (or "small"), 'M' (or "medium"), or 'L' (or "large") is vendor-dependent. Typically, however, 'S' specifies 128K; 'M', the size of physical memory; 'L', the maximum size allowed to a task. The default is "128K". If the task size specified by the user (or the default) is not large enough to hold the code from all the modules being loaded, the loader automatically adjusts the size to the smallest value that can contain all the code.
B	Set a bit in the binary header of the output module which tells the operating system to zero neither the bss segment nor any memory allocated while the task is running. This option may not be effective on all machines.
c=<source_type>	Set a flag in the binary header of the output module which indicates the type of source code from which the module was created. The argument <source_type> may be "ASSEMBLER", "C", "COBOL", "FORTRAN", or "PASCAL". The names may be specified in either upper- or lowercase letters.
C=<config_num>	By default, the loader uses the configuration number of the current hardware. The user may, however, use the 'C' option to specify a configuration number which overrides the default. This option is useful when loading a module for a machine other than the one on which it is running.
d	Set the "no core dump" bit in the binary header. This option may not be effective on all machines.
D[=<hex_num>]	Specifies the starting address of the data segment. If the user does not specify the option, the starting address defaults to

	the address specified in "/lib/std_env". If the user specifies the option without an argument, the data segment starts immediately after the text segment.
e	Print each occurrence of all unresolved external references. By default, the loader prints only the first occurrence.
f	Produce a demand-load executable module (one whose text segment is loaded only as needed). Preceding the ´f´ option with a minus sign instead of a plus sign removes the demand-load nature of the module. The alignment of the text and data segments on 512-byte boundaries, however, persists.
F[=<file_name>]	Specifies the name of a file of options to process. If the user does not specify an argument, the loader reads the file "ldr_opts" in the working directory. The ´F´ option may be used repeatedly but may not be nested.
i	Write all global symbols to the symbol table of the binary file. By default, the loader includes a symbol table in a relocatable module but not in an executable one.
I	Enable processing of floating-point interrupts. This option is only useful on a system with an MC68881 floating-point coprocessor.
l=<library_name>	Specifies the name of a library to search. The ´l´ option may be used up to twelve times in a single invocation of the loader. If the library name specified begins with a slash, ´/´, the loader first looks for the library as specified. If it is not found or if the name does not begin with a slash, the loader searches the working directory, then the directory "lib" in the working directory, and finally the directory "/lib" for the specified library. If the user specifies less than twelve libraries, the loader automatically searches the library "Syslib68k" in the working directory after it has finished searching the libraries specified by the user. The loader always processes the libraries in the order in which the user specifies them on the command line.
L	Do not search any libraries for unresolved external references.

(continued)

m	Produce load and module maps and write them to standard output (see the 'M' option).
M=<file_name>	Specifies the name of the file in which to put the output of the 'm' option (load and module maps) and the 's' option (a global symbol table). The information in this file is purely textual. The user may edit or list the file like any other text file. If the 'm' or 's' option is used without the 'M' option, the loader sends the information to standard output.
n	Produce an executable module with separate instruction and data space.
N=<module_name>	Specifies the internal name of the module produced by the loader.
o=<file_name>	Specifies the name to give to the binary file.
P=<hex_num>	Specifies the page size. The hexadecimal number should always be a power of 2; otherwise, the results are unpredictable. The "load68k" command uses the page size to determine the starting address of the data segment when it immediately follows the text segment (the data segment starts at the next page boundary). The default is 0 (i.e., the loader rounds the starting address to the next even location after the end of the text segment).
q	Suppress quad-word alignment of all segments.
r	Produce a relocatable module as output. Do not search any libraries for external references even if the user specifies them with the 'l' option.
R	Produce a relocatable module as output. Search "/lib/syslib68k" and any libraries specified by the user for external references.
s	Write the global symbol table to standard output (see the 'M' option).
S=<hex_num>	Specifies the initial stack size, which is written into the binary header of the module produced by the loader. The hexadecimal number is the number of bytes to reserve. The default is 0, in which case the system determines the initial size of the stack.
t	Produce a shared-text executable module.
T[=<hex_num>]	Specifies the starting address of the text segment. If the user does not specify the

(continued)

	option, the linking-loader uses the starting address specified in the file "/lib/std_env". If no address is specified there, the starting address defaults to 0. If the user specifies the 'T' option without an argument, the starting address also defaults to 0. If, however, the 'x' option is in effect, the starting address never defaults to 0, but instead defaults to the first page boundary following the bss segment of the file specified by the 'x' option.
u	Do not print any "unresolved" messages when producing a relocatable module.
U=<trap_num>	Specifies the trap number for system calls. The default is hardware-dependent. The user can specify the argument as either "TRAPn" where 'n' is a number between 0 and 15 inclusive, or as a string of four hexadecimal digits which represent a bit pattern to use as the system call.
w	Load modules containing instructions specific to the MC68020 microprocessor.
W	Do not load modules containing instructions specific to the MC68020 microprocessor.
x=<file_name>	Specifies the name of the file whose symbol table is to form the basis of the new symbol table. By default, the new module loads into memory immediately after the specified file. To be safe, in conjunction with the 'x' option the user should always specify the 'T' option without an argument, so that it overrides any starting address in the file "/lib/std_env".
X=<add_mask>	Specifies a 32-bit hexadecimal number, of which only the high-order 7 bits are used, which is used to mask out any of the high-order 7 bits in all addresses in the file.
y	Load modules containing instructions specific to the MC68881 coprocessor.
Y	Do not load modules containing instructions specific to the MC68881 coprocessor.

(continued)

Z	Align the text and data segments on 512-byte boundaries, padding with null bytes as necessary. Preceding the 'Z' option with a minus sign instead of a plus sign makes the module a demand-load module if its text and data segments are already aligned on 512-byte boundaries.

EXAMPLES

1. load68k *.r +F=/lib/ldr_environ +t +l=Clib +o=tester
2. load68k t1.r t2.r +T=20000 +iN=mod +P=2000 +c=C +o=test
3. load68k sqrt +msM=loadmap +l=mathlib +i
4. load68k temp?.r +reo=combined.r
5. load68k t1.r t2.r +a=10 +A=100 +b=2M +l=testlib +do=test

The first example loads all files in the working directory whose names end with ".r". The loader reads the file "/lib/ldr_environ" and processes the options therein. It uses the library "Clib" to resolve external references. The executable output module, which is a shared-text module, is named "tester".

The second example loads the the files specified and produces a binary file named "test". The internal module-name is "mod". The text segment begins at 20000 hexadecimal, and the data segment follows it at the next page boundary (page size is 2000 hexadecimal). The source code is "C". All global symbols are inserted in the symbol table of the binary file.

The third example loads the file "sqrt" and, by default, produces a binary file named "sqrt.o". The loader searches the library "mathlib" for unresolved external references. It produces load and module maps, as well as a symbol table, and writes them to the file "loadmap". All global symbols are added to the symbol table of the binary file.

The fourth example loads the files in the working directory whose names match the pattern "temp?.r" and produces a relocatable module named "combined.r". The loader prints each occurrence of all unresolved external references rather than only the first occurrence of each. Because the 'r' option is specified, the loader does not search any libraries.

The fifth example loads the files "t1.r" and "t2.r" and produces the binary file named "test". The minimum page allocation is set to 10; the maximum, to 100. The task size of the module is set to 2 Megabytes. The executable module cannot produce a core dump.

(continued)

NOTES

- If the file "/lib/std_env" contains information about the starting address of the text segment, the data segment, or both, and if the user wishes to override this standard configuration, starting addresses for both text and data segments should be specified.

- If the minimum and maximum values for page allocation provided by the user make no sense, the loader automatically adjusts them according to the following rules. The value for the maximum is always greater than or equal to the value for the minimum. The value for the maximum can be 0, but if it is greater than 0, it must be at least 4.

SEE ALSO

68xxx UniFLEX Relocating Assembler and Linking-Loader

log

Terminate the current shell program.

SYNTAX

log

DESCRIPTION

The "log" command, which is part of the shell program, terminates the current shell. If that shell program is a single-user shell, the command puts the system into multi-user mode. If the shell program is the user's login shell, the "log" command terminates the user's session with the operating system. Otherwise, the "log" command returns control to the program which called the current shell.

EXAMPLES

1. log

This example, which is the only valid form of the "log" command, terminates the current shell program. Depending on the nature of that shell, the command either puts the system in multi-user mode, returns control to the program which called the shell, or terminates the user's session with the operating system.

NOTES

. The operating system accepts the synonym "logout" for "log".

SEE ALSO

login
shell

login

Give a new user access to the operating system and establish the standard environment for the current shell program.

SYNTAX

login <user_name>

DESCRIPTION

The "login" command gives a new user access to the operating system and establishes the standard environment for the current shell program. If the user does not have a password, the system automatically honors the command. If the user does have a password, the system requests it. If it is entered correctly correctly, the new user is given access to the operating system. Otherwise, the system returns an error message, followed by a login prompt. The "login" command is part of the shell program.

Standard Environment

A list of the parameters which define the standard environment follows. The user can create, delete, or change these parameters with the "env" command.

HOME	The name of the user's home directory--that is, of the directory that the user enters by default upon logging in. The "login" command reads the password file ("/etc/log/password") to determine the appropriate name.
MAIL	The location of the user's mailbox. If the user's home directory contains a file named ".mail", the "login" command sets MAIL equal to the file specification of that file. Otherwise, MAIL is not defined.
PATH	The list of directories the shell program searches when looking for an executable file. This list, which is known as the search path, is searched sequentially. By default, the search path consists of the following directories: the user's working directory, "<home_dir>"/bin", "/bin", and "/usr/bin". If the user is the system manager, the search path also includes the file "/etc", which is searched after "<home_dir>/bin" and before "/bin". The user may alter this parameter with the "addpath" or the "setpath" command, as well as with the "env" command.

SHELL	The name of the user's login program--that is, of the program that begins execution when the user logs in. The "login" command reads the password file ("/etc/log/password") to determine the appropriate name. If the corresponding field in that file is empty, the "login" program sets SHELL to "/bin/shell".
TERM	The type of the terminal as listed in the file "/etc/ttylist". If the terminal type is "modem", the "login" command prompts the user for the type of terminal. To be useful the user's response should be a string that is defined in the file "/etc/termcap". Typing a carriage return in response to this prompt leaves the type as "modem". If the file "/etc/ttylist" does not contain an entry for the terminal, TERM is not defined. In such a case, or if the user selects a string that is not defined in "/etc/termcap", programs requiring special terminal capabilities cannot function properly.
USERNAME	The user name used to log in.

Arguments

<user_name>	The name of the user to put in contact with the operating system.

EXAMPLES

1. login leslie

This example tells the operating system to give the user whose user name is "leslie" access to the operating system. It also establishes values for the standard environment parameters for the current shell program.

NOTES

- The value of TERM for a pseudoterminal is "PTY".

ERROR MESSAGES

"login" allowed only for the "login shell".
The "login" command can only be executed from a shell program that was started by a previous "login" command.

Login incorrect!
The combination of the user name specified and the password entered is invalid. This message is followed by a login prompt.

No "login" name specified.
The user did not specify a user name on the command line.

Not allowed in single-user mode.
The "login" command can only be executed when the system is in multi-user mode.

SEE ALSO

addpath
env
log
setpath
shell

ls

List either the contents of a directory or information about a file.

SYNTAX

```
ls [<file_name_list>] [+abdflrsSt]
```

DESCRIPTION

The "ls" command is identical to the "dir" command.

SEE ALSO

dir

mail

Send mail to someone else or display any mail belonging to the user.

SYNTAX

mail [<file_name_list>]

DESCRIPTION

The "mail" command sends mail to the specified user. If no recipient is specified, the "mail" command displays any mail belonging to the user. In order for a user to be able to receive mail, the file "/usr/<user_name>/.mail" must exist. The permissions on this file should be set to "rw----" so that other users may not read or directly alter the contents.

The system automatically checks for mail each time a user logs in. If the user does have mail, the system displays the message, "You have mail". The user may request to see the mail by issuing the "mail" command without an argument. The user may also use this command at any time to find out if any mail is waiting to be read. If the user has no mail, the command reports, "No mail". If there is mail, the "mail" command displays it (complete with a header which shows the time and date the mail was received and the user name of the person who sent it) and asks whether or not it should be saved.

There are three possible responses to this question. If the user types an 'n', the system deletes the contents of the file ".mail". If the user responds with a 'y', the "mail" command transfers the mail to another file in the login directory called "mailbox" and deletes the contents of the file ".mail". If the file "mailbox" does not exist, "mail" creates it. If it does exist, "mail" appends the contents of ".mail" to "mailbox". If the user responds with a carriage return, the file ".mail" is left intact.

If the user specifies an argument with the "mail" command, the command reads standard input until it finds an end-of-file character (control-D) as the first character on a line, and sends it to the specified user.

Arguments

<user_name_list> A list of the names of users to whom to send the mail.

EXAMPLES

1. mail john
2. mail john mary <memo
3. mail

The first example accepts input from standard input, which defaults to the user's terminal, until it encounters a control-D as the first character on a line. The mail is sent to the user whose user name is "john".

The second example mails the contents of the file "memo" to users "john" and "mary".

The third example displays the user's mail.

NOTES

- Sometimes users alter things in the file system which make it impossible to execute the "mail" command. In order for the "mail" command to function properly, the file "/bin/mail" must be owned by "system" and must have the user ID bit set.

ERROR MESSAGES

Cannot find "<user_name>" in the password file.
The file "/etc/log/password" does not contain an entry for the user <user_name>. Any other users specified on the command line will receive the mail.

Cannot find your name in the password file.
The file "/etc/log/password" does not contain an entry for the user issuing the command. This situation is extremely unlikely to occur.

Error creating "<file_name>": <reason>
The operating system returned an error when "mail" tried to create <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Error opening "<file_name>": <reason>
The operating system returned an error when "mail" tried to open <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Error reading "<file_name>": <reason>
The operating system returned an error when "mail" tried to read <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Error writing "<file_name>": <reason>
The operating system returned an error when "mail" tried to write to <file_name>. This message is followed by an interpretation of the error returned by the operating system.

User "<user_name>" has no mail box.
The login directory of the specified user does not contain a file named ".mail". Therefore, that user cannot receive mail.

SEE ALSO

perms
shell

makdev

Create a special type of file, representing a device.

SYNTAX

/etc/makdev <file_name> <dev_type> <maj_dev_num> <min_dev_num>

DESCRIPTION

The "makdev" command creates a special type of file which represents a device. This type of file allows the user to access the device drivers for the corresponding physical device. Only the system manager may use this command.

Arguments

<file_name>	The name of the file to create. The last component of the name of a special file that represents a block device must consist of a string of letters followed by a string of digits. The last component of the name of the special file that represents the character device associated with that block device must consist of the same string of letters, followed by the letter 'c', followed by the same string of digits.
<dev_type>	A letter designating whether the device is a block device, 'b'; a character device, 'c'; or a pseudoterminal, 'p' (not available on all systems).
<maj_dev_num>	A number which tells the operating system which set of device drivers to use for the specified device. The correspondence between a major device number and a device driver is hardware-dependent.
<min_dev_num>	A number which tells the operating system which physical device to associate with <file_name>. The correspondence between a minor device number and a physical device is hardware-dependent.

(continued)

EXAMPLES

1. /etc/makdev /dev/fd1 b 0 1
2. /etc/makdev /dev/fdc1 c 3 1

The first example creates a special file named "/dev/fd1", which represents a block device. Its major device number is 0; its minor device number, 1.

The second example creates a special file named "/dev/fdc1", which represents the character device associated with the block device "/dev/fd1". Its major device number is 3; its minor device number, 1.

NOTES

- Every disk requires both a block device and a corresponding character device in order to function properly.

ERROR MESSAGES

'<char>' is not a valid type of device.
The argument <dev_type> must be a 'b', for a block device; a 'c', for a character device; or a 'p' for a pseudoterminal.

Error creating "<file_name>": <reason>
The operating system returned an error when "makdev" tried to create the special file <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Invalid major device number: <num>
The number specified as the major device number is invalid.

Invalid minor device number: <num>
The number specified as the minor device number is invalid.

Syntax: /etc/makdev <file_name> <dev_type> <maj_dev_num> <min_dev_num>
The "makdev" command expects exactly four arguments. The command line does not conform to the syntax.

You must be system manager to run "makdev".
Only the system manager may execute the "makdev" command.

more

Display ASCII data with user control.

SYNTAX

more [<file_name_list>]

DESCRIPTION

The "more" command displays data on the user's terminal. It lets the user both control the number of lines displayed at a time and skip lines.

If the list of file names is omitted, the "more" command accepts data from standard input. It displays enough lines to fill the terminal's screen, then prompts for a command. If the list of file names contains a single name, the "more" command displays enough lines to fill the terminal's screen, then prompts for a command. The prompt contains the percentage of the file that has been displayed. If the list of file names contains multiple names, the "more" command introduces each file with a prompt and indicates when it reaches the end of each file.

Arguments

<file_name_list> The list of files to display with user control.

User Control

The "more" command prompts the user for a command with the prompt "More? ". Unless "more" is reading from standard input, this prompt is preceded by either the percentage of the file listed, "(<n>%) ", or the message, "Beginning: <file_name> " where <file_name> is the name of the file whose contents are about to be to displayed. In response to the prompt, the user types a single-character command telling the "more" command what to do next. The single-character command should not be followed by a carriage-return. The "more" command sends a control-G (bell) to the terminal if the character typed is not a command. A list of commands follows.

The space command, ' ', starts at the next line (or at the first line of the next file) and displays lines until it either fills the screen or reaches the end of the file.

The period command, '.', starts at the next line (or with the first line of the next file) and displays lines until it either displays enough lines to scroll half of the screen or reaches the end of the file.

The carriage-return command displays the next line (or the first line of the next file) if there is one.

The 's' or '/' response requests a search for a character string. When the "more" command issues the prompt, "Search string? ", the user should type either the string to find, followed by a carriage return, or just a carriage return, which tells "more" to search for the most recently specified string. The "more" command starts at the next line (or with the first line of the next file) and searches for the specified string. If it finds the string, it displays lines, starting with the line in which the search string first appears, until it either fills the screen or reaches the end of the file. If it does not find the string, it does not alter the display unless the input is coming from a pipe, in which case "more" terminates.

If "more" cannot accept a character, it sends a control-G (bell) to the terminal. The command does not accept control characters. Nor does it accept any characters after it fills the search-string buffer. Typing a character-delete character (usually a control-H) as the first character in response to the prompt, "Search string? ", or typing a line-delete character (usually a control-X) any time while entering the search string returns the "more" command to the "More? " prompt.

The 'n' response stops processing the current file and begins processing the next file in the list of file names, if there is one. If the input is coming from a pipe, "more" terminates.

The 'p' response stops processing the current file and begins processing the preceding file in the list of file names, if there is one. If no previous file exists, "more" begins processing the current file again. If the input is coming from a pipe, "more" terminates.

The 'q' response ends the "more" command. An end-of-file character (control-D) performs the same function.

The 'r' response rewinds the current file and begins processing it. If the input is coming from a pipe, "more" terminates.

(continued)

EXAMPLES

1. more hello.c
2. more *.c
3. list hello.c | more

The first example displays the file "hello.c" at the terminal with user control. It first clears the screen, then lists enough lines from the file to fill the screen. It then requests a command from the user by issuing a prompt that indicates the percentage of the file that it has displayed.

The second example displays at the terminal with user control all of the files in the working directory whose names contain the suffix ".c" . It first clears the screen, then introduces the first file by issuing a prompt containing its name. This prompt is a request for a command. After executing the first command, "more" prompts the user for another command with a prompt that indicates the percentage of the file that it has already displayed. When "more" reaches the end of the first file, it introduces the next file. This process continues until "more" has processed all the files in the list.

The third example displays at the terminal with user control the output from the "list" command. It first clears the screen, then lists enough lines to fill the screen. After filling the screen, "more" prompts the user for a command.

NOTES

- The "more" command uses the UniFLEX terminal capabilities information ("termcap") if that information is available for the terminal being used. If the "more" command seems to be handling a terminal poorly, the system manager should verify that the terminal capabilities for that terminal are correctly set.

- If no terminal capabilities are available for the terminal in use, the "more" command assumes that there are eighty columns to a line and twenty lines on the screen. It also assumes that the backspace character (hexadecimal 08) moves the cursor one place to the left and that the space character (hexadecimal 20) moves the cursor one place to the right and clears the character at that place.

- The "more" command does not use the last column of a line. Some terminals automatically advance to the next line after writing to the last column of a line; others do not. The last column is not used to avoid having to differentiate between the two types of terminals. Lines longer than the width of the terminal are split and displayed as two lines.

- The "more" command displays all control characters as "^X" where 'X' is the key which, if struck while the "control" key is depressed, normally produces that control character. For example, it displays each embedded tab character (control-I) as "^I".

- The "more" command automatically clears the screen before displaying any data.

ERROR MESSAGES

Broken pipe
: The "more" command caught a broken-pipe interrupt. A broken-pipe interrupt immediately stops the "more" command.

File is not a regular file: <file_name>
: The file specified exists but is not a regular data file. The "more" command works only with regular data files.

Hang up
: The "more" command caught a hang-up interrupt. A hang-up interrupt immediately stops the "more" command.

Input must come from a file or a pipe
: Data from standard input must come from a pipe or a redirected data file. This message indicates that standard input is something other than a pipe or a redirected data file, such as a terminal.

INTERRUPT!
: The "more" command caught a keyboard interrupt. A keyboard interrupt immediately stops the "more" command.

Invalid option: '<char>'.
: The option specified by '<char>' is not a valid option to the "more" command.

Output must go to a terminal
: Standard output must be a character-special file (i.e., a terminal). This message indicates that it is not.

Quit
: The "more" command caught a quit interrupt. A quit interrupt immediately stops the "more" command.

SEE ALSO

list
page

mount

Insert a block device at a node of the directory tree structure or display the mount table.

SYNTAX

/etc/mount [<dev_name> <dir_name> [r]]

DESCRIPTION

When used with arguments the "mount" command temporarily inserts a block device at a node of the directory tree structure. As long as the device is mounted, any references to <dir_name> actually access the root directory of the device mounted there. Any files in the directory at which the device is mounted are inaccessible for the duration of the mount.

When used without arguments the "mount" command writes to standard output a table of information about all the devices currently mounted on the system. This "mount table" tells where, when, and by whom each device was mounted.

Arguments

<dev_name> The name of the device to mount. It must be a block device.

<dir_name> The name of the directory on which to mount the specified device.

Options Available

r Mount the device for reading only. This option must not be preceded by a plus sign. It is useful when trying to salvage data from a damaged disk because it prevents inadvertent writing to the disk, which could make matters worse.

EXAMPLES

1. /etc/mount /dev/fd0 /usr2
2. /etc/mount /dev/w0 /usr2 r

The first example mounts the disk in floppy drive 0 on the directory "/usr2". References to "/usr2" now access the root directory of that disk.

The second example mounts the hard disk in drive w0 as "/usr2". Because the 'r' option appears on the command line, no user may write to the disk.

NOTES

- When a user's working directory is the root directory of a mounted device, the command "chd .." does not change the working directory.

MESSAGES

"<dev_name>" mounted as "<dir_name>" at <time_stamp> by "<user_name>".

ERROR MESSAGES

Error mounting "<dev_name>" on "<dir_name>": <reason>
The operating system returned an error when "mount" tried to insert the specified device in the directory tree. This message is followed by an interpretation of the error returned by the operating system.

Error opening "/etc/mtab": <reason>
The operating system returned an error when "mount" tried to read the file containing the mount table. This message is followed by an interpretation of the error returned by the operating system.

Error processing "<file_name>": <reason>
The operating system returned an error when "mount" tried to process <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Syntax: /etc/mount [<dev_name> <dir_name> [r]]
The "mount" command expects exactly two arguments and, optionally, the single option 'r'. This command indicates that the command line does not conform to the syntax.

Warning: Directory "<dir_name>" is not empty.
The directory on which the user wants to mount the specified device is not empty. The "mount" command continues, but files in that directory are not accessible as long as the device is mounted.

SEE ALSO

unmount

move

Move a file from one place to another.

SYNTAX

```
move <file_name_1> <file_name_2> [+klps]
move <file_name_list> <dir_name> [+klps]
```

DESCRIPTION

The "move" command moves files from one place to another. If the first form of the "move" command is used, it moves <file_name_1> to <file_name_2> and deletes <file_name_1> from the file system. If the second form is used, it moves each file named in <file_name_list> to <dir_name>. In either case, if a file with the same name as the file created by the "move" command exists, it is deleted without warning.

Directories and special files (block devices and character devices) may not be moved. The user must have write and execute permissions in the parent directory of each file being moved and in the directory to which the files are moved. Each original file is removed, and the last component of the file name is unchanged.

A file may not be moved from one device to another unless the user has read permission on the file. A file may not be moved to itself.

Arguments

<file_name_1>	The name of the file to move.
<file_name_2>	The name of the file to which to move <file_name_1>.
<dir_name>	The name of the directory to which to move all the specified files.

Options Available

k	Do not remove the original file from the file system.
l	List the name of each file as it is moved.
p	Prompt for permission to replace existing files.
s	Stop as soon as an error is encountered.

EXAMPLES

1. move test oldtest +l
2. move test /usr/elaine
3. move test /usr/elaine/oldtest +kp
4. move * /usr/elaine +s

(continued)

The first example moves the file "test" in the working directory to the file "oldtest" in the working directory. The "move" command issues a message describing the move. In effect, this command renames the original file.

The second example moves the file "test" from the working directory to the directory "/usr/elaine". The last component of the file name is preserved, so the name of the new file is "/usr/elaine/test".

The third example moves the file "test" from the working directory to the file "oldtest" in the directory "/usr/elaine". If the file "/usr/elaine/oldtest" already exists, the user is prompted for permission to delete the file. If permission is denied, the move does not take place. Even if the move takes place, the original files remain intact.

The fourth example moves all the files in the working directory to the directory "/usr/elaine". The last component of each file name is preserved. The command aborts if it encounters an error.

NOTES

- Normally, the "move" command links the new file to the original file and deletes the original one. Because a link between files on different devices is not permitted, an attempt to "move" a file to a different device results in the copying of the original file to the new file, followed by the deletion of the original file.

MESSAGES

"<file_name_1>" copied to "<file_name_2>"
This message is produced only if both the ´l´ and ´k´ options are specified. It means that <file_name_1> has been copied to <file_name_2>, but that the original file remains intact. This message indicates that the two files are on different devices.

"<file_name_1>" linked to "<file_name_2>"
This message is produced only if both the ´l´ and ´k´ options are specified. It means that the two files have been linked but that the original file remains intact (the user specified the ´k´ option).

"<file_name_1>" moved to "<file_name_2>"
This is the normal message issued by the "move" command if the ´l´ option is in effect. It means that <file_name_1> has been either linked or copied to <file_name_2>, and that <file_name_1> has been deleted.

ERROR MESSAGES

Cannot move a block special file: <file_name>
The file <file_name> is a block special file (block device) and may not be moved.

Cannot move a character special file: <file_name>
The file <file_name> is a character special file (character device) and may not be moved.

Cannot move across devices: <file_name>
The file <file_name> is read protected and, therefore, cannot be moved across devices.

Directory is not accessible: <dir_name>
The user does not have the necessary permissions (write and execute) to move a file to <dir_name>.

"<file_name_1>" and "<file_name_2>" are the same file.
The user tried to move a file to itself. If <file_name_1> and <file_name_2> are different, they are links to the same file.

Invalid option: '<char>'.
The option specified by <char> is not a valid option to the "move" command.

Permissions deny access: <file_name>
The user does not have write permission in the parent of the specified directory.

SEE ALSO

copy
link

newuser

Temporarily log in as a new user.

SYNTAX

```
newuser [<user_name>]
```

DESCRIPTION

The "newuser" command allows the user to log in as another user without logging out. If a name is specified on the command line, that name becomes the new login name. If no name is specified, "system" is used. If a password exists for the login name specified, "newuser" prompts for the password. The advantage of this command is that when finished as this new user, the user does not need to log in again but simply logs out and returns to the state that existed prior to the execution of the "newuser" command.

Arguments

<user_name>	The name of the user as whom to temporarily log in. The default name is "system".

EXAMPLES

1. newuser mary

This example temporarily logs in the user as "mary" (assuming the user knows the password if one exists).

NOTES

- The "newuser" command creates an environment for the shell program just as the "login" command does, with the exception that the value of the parameter TERM is taken from the parent task. If TERM is not defined in the parent task's environment, "newuser" determines its value from the file "/etc/ttylist".

SEE ALSO

log
login
su

next

Restart an idled printer program.

SYNTAX

next <splr_name>

DESCRIPTION

The "next" command restarts a printer program which has been idled by the "idle" or "rerun" command.

Arguments

<splr_name> The name of the printer program to restart.

EXAMPLES

1. next spr

This example restarts the printer program associated with "spr".

NOTES

. The "next" command is one of five commands that are linked to the file "/etc/prcon", which controls the printing of files.

ERROR MESSAGES

Cannot find spooler directory for "<splr_name>".
The directory "/usr/gen" does not contain a directory for the specified spooler.

Error opening "<file_name>": <reason>
The operating system returned an error when "next" tried to open the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error reading "<file_name>": <reason>
The operating system returned an error when "next" tried to read the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error unlinking ".idl*splr?": <reason>
The operating system returned an error when "next" tried to unlink the file ".idl*splr?". This message is followed by an interpretation of the error returned by the operating system.

"<splr_name>" is not idle.
The "next" command is effective only on an idle spooler.

Syntax: next <splr_name>
The "next" command expects exactly one argument. This message indicates that the argument count is wrong.

You must be system manager to run "next".
Only the system manager may execute the "next" command.

SEE ALSO

end
idle
insp
print
pstop
purge
rerun

nice

Lower the priority of the specified command.

SYNTAX

nice <command_name>

DESCRIPTION

The "nice" command lowers the priority assigned to the specified task by subtracting 5 from the number that would normally be assigned (the lower the number, the lower the priority). For example, a user might send a long compilation, the results of which are not immediately needed, to the operating system with the "nice" command. The system executes more pressing tasks immediately and works on the compilation when nothing else is running.

Arguments

<command_name> The name of the command to execute.

EXAMPLES

1. nice ls
2. nice pascal test.p

The first example lowers the priority of the command "ls".

The second example lowers the priority of the compilation of the pascal program "test.p".

ERROR MESSAGES

No command or a built-in command specified.
The "nice" command expects exactly one argument, and that argument may not be one of the commands, such as "jobs" or "wait", that is a part of the shell program.

SEE ALSO

shell
status

owner

Change the owner of a file.

SYNTAX

owner <new_owner> <file_name_list>

DESCRIPTION

The "owner" command changes the owner of the specified file. Only the system manager may execute this command.

Arguments

<new_owner>	The user name or user ID of the new owner of the file.
<file_name_list>	A list of the names of the files on which to change the owner.

EXAMPLES

1. owner system /usr/john/*
2. owner 110 /usr/john/*

The first example changes the owner of all the files in the directory "/usr/john" to "system".

The second example changes the owner of all the files in the directory "/usr/john" to the user whose ID is 110.

ERROR MESSAGES

Error changing owner for "<file_name>": <reason>
The operating system returned an error when "owner" tried change the owner of the specified file. This message is followed by an interpretation of the error returned by the operating system.

"<name>" is not a valid user name.
The specified name is not in the password file and, therefore, is not a valid user name.

<num> is not a valid user identification number.
The specified number is not in the password file and, therefore, is not a valid user ID.

Syntax: owner <new_owner> <file_name_list>
The "owner" command expects at least two arguments. This message indicates that the argument count is wrong.

You must be system manager to run "owner".
Only the system manager may execute the "owner" command.

page

Format a file in pages.

SYNTAX

page [<file_name_list>] [+f1<num>p]

DESCRIPTION

The "page" command formats a file for printing on a line printer or for viewing on a crt terminal. The output is formatted with sixty-six lines per page. Nine of the e lines are used for a header, which includes the name of the file, the current date and time, and the page number. Each page is terminated with a form-feed character. The output is sent to standard output. Thus, I/O redirection and pipes may be used in conjunction with the command.

Arguments

<file_name_list>	A list of the names of the files to format. The default is standard input.

Options Available

f	Replace the form-feed character at the end of the page with the appropriate number of line-feed characters.
l	Precede each line with a line number.
<num>	Print <num> lines at a time to the terminal. After outputting <num> lines the "page" command puts the terminal in hold mode. In order to see the next page of output, the user must type whatever character releases the hold. Usually, this character is the escape character. However, it may change depending on the parameters that are set with the "ttyset" command. This option disables the header. It is not meant for use with a printer. The only option which may be used in conjunction with this option is the 'l' option.
p=<num>	Format the file with <num> lines per page, nine of which are used by the header. The minimum value for the argument is 10; the default is 66. This option is not meant for use with a terminal.

(continued)

EXAMPLES

1. page text +123
2. page chap_1 chap_2 +fp=49 ^ppr

The first example formats the file "text" with twenty-three lines per page and sends it to the user's terminal. The header is absent, but each line is preceded by a line number.

The second example formats the files "chap_1" and "chap_2" with forty-nine lines per page (nine used as header) and pipes the output to the printer "ppr". Each page is terminated with the appropriate number of line-feed characters rather than a form-feed character.

ERROR MESSAGES

Error opening "<file_name>": <reason>
The operating system returned an error when "page" tried to open <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Error reading "<file_name>": <reason>
The operating system returned an error when "page" tried to read <file_name>. This message is followed by an interpretation of the error returned by the operating system.

File: <file_name> - No EOL found after 512 characters - aborted!
The "page" command buffers input internally. If no carriage return is encountered after 512 characters, the command aborts. Most probably, the specified file is a binary file.

Invalid option: '<char>'
The option specified by <char> is not a valid option to the "page" command.

Too few lines per page for 'p' option.
The user specified a number less than 10 as the argument to the 'p' option. Since the header uses nine lines, the argument must be greater than or equal to 10.

SEE ALSO

list
more
ttyset

password

Set or change a user's password.

SYNTAX

password [<user_name>]

DESCRIPTION

The "password" command sets or changes a user's password. Only the system manager may change another user's password. When a user invokes the command, the operating system prompts for the existing password (if there is one). If the password is entered correctly, the system prompts for the new password. Generally, a password should contain five or six lowercase, random characters. After the new password is entered, the system prompts for it again to verify it. If the second entry agrees with the first, the password is entered in the password file. In order to maintain the secrecy of the password, the operating system does not echo the characters typed in response to the prompts for either the existing or the new password.

Arguments

<user_name> The name of user whose password is being changed. The default is the user invoking the command.

EXAMPLES

1. password
2. password greg

The first example changes the password of the user who invoked the command.

The second example uses the form of the command which is restricted to use by the system manager. It changes the password associated with the user name "greg".

ERROR MESSAGES

Cannot find "<user_name>" in the password file.
The file "/etc/log/password" does not contain an entry for the user <user_name>.

Cannot find your name in the password file.
The file "/etc/log/password" does not contain an entry for the user issuing the command. This situation is extremely unlikely to occur.

Error linking "/tmp/pswd" to "/etc/log/password": <reason>
The operating system returned an error when "password" tried to link the new version of the password file to the old password file. This message is followed by an interpretation of the error returned by the operating system.

Error locking password file: <reason>
The operating system returned an error when "password" tried to lock the password file. This message is followed by an interpretation of the error returned by the operating system.

Error opening "<file_name>": <reason>
The operating system returned an error when "password" tried to open the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error unlinking "<file_name>": <reason>
The operating system returned an error when "password" tried to unlink the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error writing "<file_name>": <reason>
The operating system returned an error when "password" tried to write to the specified file. This message is followed by an interpretation of the error returned by the operating system.

Only the system manager may change another's password.
Use of the form of the "password" command that takes an argument is limited to the system manager.

Password file is locked. Try again later.
The commands "addusr", "delusr", and "password" all lock the password file so that two people cannot try to alter it at the same time. This message indicates that one of these commands currently has the password file locked.

Password not correct. Permission denied!
The user did not enter the existing password correctly.

Retry different - password unchanged.
The first and second entries of the new password were not identical. The password command aborts, leaving the original password in place.

Syntax: password [<user_name>]
The "password" command expects no more than one argument. This message indicates that the argument count is wrong.

path

Write the path name of the working directory to standard output.

SYNTAX

path

DESCRIPTION

The "path" command writes the path name of the working directory, followed by a carriage return, to standard output. The path name, also called the file specification, is the unique path from the root directory through the directory tree to the file in question.

EXAMPLES

1. path

This example is the only valid form of the "path" command. It writes the name of the working directory, followed by a carriage return, to standard output, which defaults to the user's terminal. Of course, the user may redirect standard output.

ERROR MESSAGES

Directory structure is corrupt
The directory path from the root directory, '/', to the working directory is corrupt. Therefore, the "path" command cannot determine the path name of the working directory.

SEE ALSO

chd

perms

Change the permissions associated with a file.

SYNTAX

perms <perms_list> <file_name_list>

DESCRIPTION

Every time a user creates a file, the operating system assigns it a set of permission bits which determines whether or not the user who owns the file and other users may read, write, or execute the file. The permissions assigned depend on the command used to create the file. The editor, for example, creates all files with "rw-rw-" permissions, which allow the user who owns the file, as well as other users, to read and write, but not execute, the file. The default permission for "crdir" are "rwxrwx"; for create, "rw-rw-"; for "makdev", "rw-r--".

Read permission allows a regular file to be read. A user cannot execute commands such as "list" and "copy" without read permission on the file in question. Write permission allows a file to be modified. Execute permission allows the name of the file to be used as a command.

Permissions for directories are similar to those for normal files. Read permission allows the user to read the names and addresses that are actually in the directory. Write permission allows the creation or deletion of files in the directory. Execute permission allows the directory to be searched for a name used as part of a file specification or file name. The user must have execute permission to successfully use a directory as the argument to the "chd" command.

In addition to these permissions, each file has associated with it a user ID bit. If this bit is set for a given file, any user executing the file has the same privileges as the owner of the program for the duration of the task.

The "perms" command changes the permission bits associated with a file. Only the owner of a file or the system manager may change the permissions associated with it.

Arguments

<perms_list>	The list of permission bits to alter. Permission bits not mentioned are not changed.
<file_name_list>	A list of the names of the files for which to alter the permissions.

Format for Arguments

<perms_list> The first character of an element in the permissions list specifies whether the argument applies to the user who owns the file ('u') or to others ('o'). The second character specifies whether to add ('+') or remove ('-') the permissions in question. The second character is followed by one, two, or three of the characters 'r', 'w', and 'x' (for read, write, and execute). The user ID bit is set or cleared with one of the following arguments: "s+" or "s-".

EXAMPLES

1. perms o-wx inventory
2. perms o+x u+x script
3. perms o-rw o+x s+ inventory script

The first example removes write and execute permissions for other users from the file "inventory" in the working directory.

The second example gives execute permissions on the file "script" to both the user who owns it and to other users.

The third example removes read and write permissions for others from the files "inventory" and "script". It also sets execute permissions for others, as well as the user ID bit. Thus, although other users may neither read from nor write to the files, they may execute them. While they are executing them, they have the same permissions on all files as the owner of these files does.

ERROR MESSAGES

Error changing permissions for "<file_name>": <reason>
The operating system returned an error when "perms" tried change the permissions on the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error processing "<file_name>": <reason>
The operating system returned an error when "perms" tried to determine the original permissions on the file. This message is followed by an interpretation of the error returned by the operating system.

Syntax: perms <perms_list> <file_name_list>
The "perms" command expects at least two arguments. This message indicates that the argument count is wrong.

Unrecognizable character, '<char>', found in permissions list.
Command aborted!
A character following a plus or minus sign in an element in the permissions list was not an 'r', 'w', or 'x'. The command aborts without altering any permissions.

SEE ALSO

dir
dperm

print

Send a file to the specified printer spooler.

SYNTAX

<print> [<file_name_list>] [+m]

DESCRIPTION

The <print> command, which may have several different names on any system, sends the specified file to a printer spooler for printing. The <print> command usually has a name which represents the particular printer to use for output. For example, if two printers are available, one might be called "lprinter" for line printer and the other, "daisy" for daisy wheel printer. Thus, the user selects the output device by selecting the command name which represents the desired device. If the user does not specify a file, <print> takes its input from standard input. The command can, therefore, be used in a pipe.

Once the file has been sent to the printer spooler, the <print> command prints a message of the following form:

"<file_name>" printed on <dev_name> as "<user_name<num>>"

where <file_name> is the name of the file printed (if available); <dev_name> is the name of the printer device to which the file was routed; and the last item is the name which the system assigns to the print file. This name, as well as the device name, is needed if it becomes necessary to delete the file from the print queue (see "purge").

Arguments

<file_name_list> A list of the names of the files to route to the printer spooler. The default is standard input.

Options Available

m Suppress the message announcing that the file has been sent to the printer spooler.

EXAMPLES

1. spr instructions
2. ppr /usr/jennifer/manual/instructions +m

The first example sends the file "instructions" in the working directory to the printer spooler for the device "spr".

The second example sends the file "instructions" in the directory "/usr/jennifer/manual" to the printer spooler for the device "ppr". The message announcing that the file has been queued is suppressed.

ERROR MESSAGES

Cannot find "<file_name>".
The <print> command cannot locate the specified file.

Cannot find your name in the password file.
The file "/etc/log/password" does not contain an entry for the user issuing the command. This situation is extremely unlikely to occur.

Error creating "<file_name>": <reason>
The operating system returned an error when <print> tried to create the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error linking "<file_name>": <reason>
The operating system returned an error when <print> tried to link the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error opening "<file_name>": <reason>
The operating system returned an error when <print> tried to open the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error reading "<file_name>": <reason>
The operating system returned an error when <print> tried to read the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error writing "<file_name>": <reason>
The operating system returned an error when <print> tried to write the specified file. This message is followed by an interpretation of the error returned by the operating system.

"<file_name>" is not a regular file. File not queued.
The <print> command cannot print directories or special files (block devices and character devices).

Invalid option: '<char>'. File not spooled.
The option specified is not a valid option to the <print> command.

SEE ALSO

end
idle
insp
next
pstop
purge
rerun

prompt

Define the prompt and reprompt strings issued by the shell program.

SYNTAX

```
prompt <prompt_str> [<reprompt_str>]
```

DESCRIPTION

The "prompt" command, which is part of the shell program, defines the prompt and reprompt strings returned by the operating system. The prompt string is the string issued by the operating system when it is ready to accept a command. The reprompt string is the string issued by the operating system when the user has entered a line terminated by the combination of a backslash character and a carriage return. This combination of characters tells the shell program that the line is incomplete.

Both the prompt and the reprompt strings should be enclosed in single or double quotation marks. Neither string may be more than fourteen characters long. The first tilde, '~', in each string is replaced in the prompt by the current time expressed in the form <hr>:<min>, which represents the hours and minutes on a 24-hour clock.

Arguments

<prompt_str>	The string to use as the new prompt. By default, the shell program issues the following prompt: ++
<reprompt_str>	The string to use as the reprompt string, the prompt returned by the operating system when the user enters a line terminated by a backslash character followed by a carriage return. By default, the shell program issues the following two-character reprompt: +>

EXAMPLES

```
1. prompt '% '
2. prompt '~ -->' 'cont: '
```

The first example changes the prompt to a percent sign, followed by a space.

The second example sets the prompt to the current time, followed by a space, followed by an arrow, followed by another space. It changes the reprompt string to "cont: ".

NOTES

. The "prompt" command is only effective while the shell program under which it is invoked is running. The prompt and continuation prompt of the login shell can be permanently altered by placing the appropriate command in the file ".startup" in the user's home directory. This file is automatically executed each time the user logs in.

SEE ALSO

shell

pstop

Deactivate the specified printer spooler.

SYNTAX

pstop <splr_name>

DESCRIPTION

The "pstop" command deactivates the specified printer spooler by interrupting the background task created by the corresponding "insp" command. The background task finishes printing the current task (if one exists), then deletes the ".mrk*splr?" file in the appropriate spooler directory. A deactivated printer spooler neither prints files nor accepts them into the print queue. Any files that are in the print queue when the "pstop" command is issued remain in the queue. All active spoolers should be stopped before the system is shut down. Some data may be lost if this command is executed while a job is being printed.

Arguments

<sprl_name> The name of the printer spooler to deactivate.

EXAMPLES

1. pstop ppr

This example sends an interrupt to the background task associated with the printer spooler "ppr". The spooler finishes printing the current print job, if one exists, before it stops.

NOTES

- The "pstop" command is one of five commands that are linked to the file "/etc/prcon", which controls the printing of files.

ERROR MESSAGES

Cannot find spooler directory for "<splr_name>".
The directory "/usr/gen" does not contain a directory for the specified spooler.

Error opening "<file_name>": <reason>
The operating system returned an error when "pstop" tried to open the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error reading "<file_name>": <reason>
The operating system returned an error when "pstop" tried to read the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error writing "<file_name>": <reason>
The operating system returned an error when "pstop" tried to write the specified file. This message is followed by an interpretation of the error returned by the operating system.

Syntax: pstop <splr_name>
The "pstop" command expects exactly one argument. This message indicates that the argument count is wrong.

You must be system manager to run "pstop".
Only the system manager may execute the "pstop" command.

SEE ALSO

end
idle
insp
next
print
purge
rerun

purge

Delete a file from the specified printer-spooler queue.

SYNTAX

purge <splr_name> <file_name_list>

DESCRIPTION

The "purge" command deletes a file from the specified printer-spooler queue.

Arguments

<splr_name>	The name of the device on which the file is to be printed.
<file_name_list>	A list of the names of the files to delete from the printer-spooler queue. Each name must be the one assigned to the file by the <print> command. This name consists of the name of the user, followed immediately by a three-digit string.

EXAMPLES

1. purge lprinter john231

This example removes the file named "john231" from the printer-spooler queue for the device "lprinter".

NOTES

- If the <print> command was invoked with the 'm' option or if for some other reason the user does not know the name assigned to the file by the <print> command, it may be possible to obtain a list of all the files in the queue by issuing the following command:

dir /usr/gen/<splr_name>

ERROR MESSAGES

Cannot find file "<file_name>".
The file specified is not in the spooler directory.

Cannot find spooler directory "/usr/gen/<splr_name>".
There is no spooler directory by the specified name.

Cannot purge "<file_name>".
The purge command cannot purge a directory from the spooler directory, nor can it purge a file whose name begins with a period.

Error unlinking "<file_name>": <reason>
The operating system returned an error when "purge" tried to unlink the specified file. This message is followed by an interpretation of the error returned by the operating system.

Syntax: purge <splr_name> <file_name_list>
The "purge" command expects at least two arguments. This message indicates that the argument count is wrong.

You do not own "<file_name>". File not purged.
Only the system manager may purge a file that belongs to someone else.

SEE ALSO

print

qdb

The "qdb" command is a machine-language debugging system.

SYNTAX

qdb [<image_file_name>]

DESCRIPTION

The "qdb" command is used to aid in the testing and debugging of machine-language programs. Because all programs are ultimately translated into machine language, any program may be debugged using "qdb" (although a debugger for a high-level language may be simpler to use if one is available).

The "qdb" command is used to examine or modify the image of a machine-language program. This image can be (1) a post-mortem memory dump of a program which has been aborted by the operating system, (2) a program image file, or (3) a program which is currently executing under the control of "qdb". If no image file is specified on the command line, the default is the file "core" in the working directory. The "qdb" command examines the file to determine whether it is a "core" image or an executable image file. If it is neither, "qdb" issues the message "Invalid image type" and terminates. The third type of image may only be created by the specification of the name of an executable image on the command line, followed by the execution of of the 'x' command to create the controlled task.

The commands available with "qdb" allow the user to examine memory locations within the program image, to modify memory locations, to set breakpoints, to execute single instructions (to single step through the program), to examine and change registers, and more. Some commands, such as single step, are only applicable when "qdb" is being used to control the execution of a task. However, most commands are available for use with all image types.

Arguments

<image_file_name>	The name of the file to debug. The default is the file "core" in the working directory.

(continued)

Commands Available

The "qdb" command normally works in an interactive environment. The basic command structure is designed to be simple both to use and to remember. In general, commands are of the following form:

<char> [<expression_list>]

where <char> is a single character specifying the command. Depending on the particular command, one or more expressions may or must be provided with the command. Expressions may include the operators '+' and '-', which are evaluated from left to right unless parentheses are used. Expressions may also include any of the following terms:

$<num>	The hexadecimal value of <num>.
<num>	The hexadecimal value of <num>. If this form is used, the number must start with a digit. If it starts with a character, "qdb" interprets it as a symbol.
#<num>	The decimal value of <num>.
<symbol>	The value of the specified symbol. Symbol names must be completely specified--that is, all characters are significant.
<register>	The contents of the specified register. The register may be D0 through D7, A0 through A7, SR, or PC. The letters used in specifying a register may be either upper- or lowercase.

Brief descriptions of the commands follow.

- `+` Execute a shell command.
- `=` Display the value of an expression in multiple formats.
- `?` Display the "help" menu.
- `b` Set a breakpoint.
- `B` List the breakpoints that are currently set.
- `c` Clear one or all breakpoints.
- `d` Dump a section of memory.
- `g` Continue execution of a program.
- `G` Execute the program until reaching a branch or a breakpoint.
- `i` Disassemble instructions.
- `I` Initialize symbol table.
- `k` Terminate the currently executing task.
- `K` Remove any pending signals for the controlled task.
- `m` Modify bytes in memory.
- `M` Display the current memory map.

(continued)

n Display the command line for the task.
q Terminate "qdb".
r Display the contents of all registers.
R Set the contents of a register.
s Execute a single instruction.
S Set a temporary breakpoint at the instruction following the current instruction.
T Trace instructions until reaching a branch or a breakpoint.
x Create a task to be executed under the control of "qdb".

More detailed descriptions of the commands and explanations of their syntax follow:

+ <shell_command>
This command allows the user to execute a single shell command without exiting "qdb".

= <expression>
This command displays the value of the expression symbolically, in hexadecimal, and in decimal.

?
This command displays a menu of commands available from "qdb".

b <location> [<count>]
The ´b´ command sets a breakpoint at the given location. When the program is executed, the instruction at the given location is replaced by a special instruction which indicates to the operating system that the user wants to break the flow of the program. When this instruction is executed in the program, the operating system suspends the program and notifies "qdb", which prints the location of the breakpoint and returns to command mode. If the user specifies a count, the breakpoint is executed <count> times before execution is halted and "qdb" notified. Once the count is exceeded, execution is halted every time the breakpoint is encountered unless it is reset by another ´b´ command or cleared.

B
The ´B´ command lists each breakpoint which is currently set as well as the corresponding <count> if it is nonzero.

c [<address>]
If the user does not specify an address, the ´c´ command prompts for permission to clear all breakpoints that are currently set. If the user does specify an

address, it clears the breakpoint at that address.

d <address_1> [<address_2_or_count>]
The 'd' command dumps the hexadecimal contents and the ASCII equivalents of a range of memory locations. Memory is displayed sixteen addresses to a line. The hexadecimal contents are displayed first, followed by their ASCII equivalents. Nonprintable characters are represented in ASCII by a period, '.'.

If the user specifies only one argument, the command displays the contents of the specified address. If the user specifies two arguments and the second one is greater than the first, the command interprets the second argument as an address. It displays the contents of memory from the first specified address to the second, inclusive. If the user specifies two arguments and the second one is less than or equal to the first, the command interprets the second argument as a count. It displays the contents of memory beginning at the first address and continuing for the number of addresses specified by the second argument.

The dump may be aborted by typing the return key during the dump. Control-C does not abort the command.

g
The 'g' command continues the execution of a controlled task. Execution continues until the program terminates, receives a signal or encounters a breakpoint. The user may only use this command when executing a controlled task.

G
The 'G' command executes the program until it encounters any branch instruction, any call instruction, or any breakpoint.

i [<address_1> [,<address_2_or_count>]]
If the user specifies two arguments and the second one is greater than the first, the 'i' command interprets the second argument as an address. If the user specifies two arguments and the second one is less than or equal to than the first, the command interprets the second argument as a count.

The 'i' command displays the contents of memory from the first specified address to the second, inclusive. If the user specifies two arguments and the second one is less than or equal to the first, the command interprets the second argument as a count. The 'i'

command interprets the specified location or range of locations as machine-language instructions and advances the location counter to the start of the last complete instruction within the specified range. If the user specifies no second argument or if the range specified by the second argument is shorter than the complete instruction, the command displays the instruction which begins at the starting address but does not move the location counter. A carriage return by itself is equivalent to the command "i ." except that the location counter is advanced to the beginning of the next instruction.

I [<file_name>]
The 'I' command specifies the name of the file whose symbol table is to be used to initialize the internal symbol table for "qdb". The symbol table is used to interpret symbolic addresses and values. If the user does not specify a file, the 'I' command prompts for the name of the file containing the symbol table to use. The file must be a binary image file. This command is normally for use with a core image file because such files do not contain any symbolic information. Once the symbol table is initialized, however, a core image file can be interpreted symbolically.

k
The 'k' command terminates execution of the current controlled task. If no controlled task exists, the command is not allowed. This command need not be used because the 'x' command implicitly kills any controlled task before creating another.

K
When a task which is running under the control of "qdb" receives a signal, the operating system notifies "qdb" and suspends the task. The "qdb" program then enters command mode, allowing the user to execute any "qdb" command. A user who wishes to ignore the signal may do so by entering the 'K' command. A user who wishes the signal to take effect should simply continue the program with the 'g' (or a similar) command.

m <address>
The 'm' command modifies the contents of one or more memory locations in the image file. In response to this command "qdb" first displays the specified address and its contents. The user may change the contents by entering any expression, may leave the contents as is by entering a period, '.', or may terminate the command

by entering just a carriage return. Unless the user terminates the command, "qdb" modifies the contents if appropriate, displays the next address with its contents, and waits for input from the user.

If the image file is a core dump or an executable file, the file itself is modified. If the image file is a controlled task (i.e., an 'x' command has been executed), only the memory of that task is altered. The executable file from which "qdb" created the task is not changed. Therefore, when patching code the user should be aware that patches are applied only to the executing image file.

M

The 'M' command displays a map of the logical addresses available to the task image. If the image is either a core dump or a controlled task, the map contains the ranges of addresses being used by the program. These ranges may change whenever the program executes a "break" or a "stack" system call. If the image is an executable file, the 'M' command displays the ranges of the addresses of the TEXT and DATA/BSS segments.

n

The 'n' command displays the command line which was used to create the task. This is merely a display of the command arguments passed to the program when it was created. In most cases the command line consists of the shell command used to invoke the program. The command line for a controlled task looks just like the command line entered with the 'x' command that created it except that the 'x' is replaced by the program name.

r

The 'r' command displays the contents of the registers for the image file, as well as the address of the program counter and the instruction located at that address. For a core dump it displays the contents of the registers at the time the program was aborted by the system and the location of the program counter at that time. The instruction displayed is the instruction that was in progress when the program was aborted. For a controlled task it displays the contents of the registers as they will be when execution resumes, the address at which execution will resume, and the instruction at that address. The registers for an executable file are undefined. If the user executes the 'r' command on an executable file, it displays the contents of the registers as zeros and the address and contents of the entry point of the program.

R <register_name> <expression>

The ´R´ command, which may only be used if the image file is a controlled task, alters the contents of a register. The register may be D0 through D7, A0 through A7, SR, or PC. The letters used in specifying a register may be either upper- or lowercase. The supervisor portion (the upper byte) of the status register may not be altered.

s

The ´s´ command executes a single machine-language instruction. When the instruction is complete, "qdb" displays the state of the task, including the new program counter, and the next instruction to be executed. The ´s´ command uses system facilities provided by the operating system. Thus, the user may safely single step through macro operations such as system calls.

S

The ´S´ command sets a temporary breakpoint at the instruction following the current instruction. This breakpoint is removed as soon as it is encountered the first time. If another ´S´ command is executed before the breakpoint is encountered, it removes the original breakpoint. This command may be used with any instruction, but it is normally used with a call to a subroutine.

T

The ´T´ command executes the program until it encounters any branch instruction, any call instruction, or any breakpoint. After the execution of every instruction "qdb" displays the address of the next instruction and the instruction itself.

x [<arguments>] [<I/O_redirection>]

The ´x´ command creates a controlled task from an image file. In order to execute this command, the user must first invoke "qdb" with the name of an executable image file as the argument. The task is halted before execution of its first instruction, so that "qdb" can accept commands to control its execution.

I/O redirection may be accomplished using the character ´<´ to redirect standard input, ´>´ to redirect standard output, and ´%´ to redirect standard error. No provisions are made for using either append mode (">>") or implied mapping (">%").

(continued)

NOTES

. The more breakpoints a user sets, the more time is used to execute the program.

ERROR MESSAGES

Breakpoint table full!
The user has already set the maximum number of breakpoints allowed by "qdb".

Can't access core/image "<image_file_name>"
The operating system returned an error when "qdb" tried to access the specified file. Most probably, either the file does not exist or the user does not have read permission in the file.

Can't open "<file_name>"
The "qdb" command was unable to open the file which the user specified as the file containing the symbol table to use. Most probably, either the file does not exist or the user does not have read permission in the file.

Can't write "<image_file_name>"
The user tried to use the 'm' command to modify the contents of a memory location in the image file, but "qdb" was unable to write to the file. Most probably, the user does not have write permission in the file.

Command too complicated
The user tried to use the '+' command to execute a shell command from "qdb", but the command line was too long for "qdb" to interpret.

-- Error during EXEC - <error_num>
The operating system returned an error when the user tried to create a controlled subtask using the 'x' command. This message is followed by the UniFLEX error number returned by the operating system.

Error in expression
The expression used contains a syntax error.

Illegal address
The address specified is not in the user's address space.

Illegal command, <char>, - ignored
The command specified by <char> is not a valid command for "qdb". The character is ignored, and "qdb" prompts the user for another command.

Illegal file type
  The 'I' command cannot determine the file type of the image file and, consequently, ignores the file. All previously defined symbols are no longer defined.

Illegal register name
  The register name specified by the user is not a valid register name. The register name must be one of the following: D0 through D7, A0 through A7, SR, or PC. The letters used may be upper- or lowercase.

"<image_file_name>" is not executable
  The user does not have execute permission in the specified image file.

Invalid image file "<file_name>"
  The file specified to the "qdb" command must be either an executable file or a core dump.

-- No command line
  The file being debugged is not a core file, nor did the user invoke it with the 'x' command. Therefore, no command line exists for the file.

Not executing a task!
  The command specified can only execute if the user has previously executed the 'x' command.

Sorry, can't execute a "core" file
  The 'x' command cannot be executed on a core file.

** Syntax error
  The 'x' command cannot parse the specified command line.

Undefined symbol
  An expression contains a term which appears to be a symbol (starts with a letter or an underscore character, '_') but is not in the symbol table. Hexadecimal values used in expressions must begin with a digit (a leading 0 is accepted) or a dollar sign, '$'.

ramdisk

Format a RAM disk (a pseudodisk device).

SYNTAX

/etc/ramdisk <dev_name> [<size>] [+f]

DESCRIPTION

The "ramdisk" command formats a section of random-access memory (RAM) for use as a pseudodisk, known as a RAM disk. The memory need not be contiguous. A system may support a maximum of four RAM disks (numbered 0 through 3).

In order for the "ramdisk" command to succeed, two devices associated with the RAM disk--one a block device and the other a character device--must exist. If these devices do not exist, the system manager must create them with the "makdev" command. The command also requires the presence of a mount table, the file "/etc/mtab". If this file is nonexistent, the system manager should create it with the "create" command.

The "ramdisk" command cannot format a mounted RAM disk; however a RAM disk must be mounted before a user can read from or write to it. Data remain on the RAM disk when it is unmounted. To return memory allocated to a RAM disk to the operating system, a user must reformat the RAM disk specifying a size of 0.

Reading from or writing to a RAM disk is considerably faster than the comparable operation on a disk device. However, RAM disks should be used judiciously because they consume system memory. A lack of memory may lead to a great deal of paging and to a general degradation of system performance.

Only the system manager may execute the "ramdisk" command.

Arguments

<dev_name> The name of the RAM disk to format. If a system supports RAM disks, they are created by the "crdisk" command with the names "/dev/ram0", "/dev/raml", "/dev/ram2", and "/dev/ram3". If the user specifies a name that does not begin with a slash character, '/', "ramdisk" prefixes the name with the string "/dev/".

<size> The size of the RAM disk in 512-byte blocks. The size must be a whole number between 0 and 8192 inclusive. The default is 512. If the number

specified is not a multiple of 8, "ramdisk" rounds it up to the next multiple of 8. The number specified determines the maximum amount of memory that the RAM disk can use. The operating system does not remove this memory from system memory immediately, but as the RAM disk needs it.

The upper limit of 8192 is a theoretical limit. The practical limit for a given system may be lower because the operating system limits the total amount of memory that it will allocate to all RAM disks combined to 75% of the amount of user-accessible memory available at the time the system is booted.

Options Available

f=<blocks> Specifies the number of blocks to reserve for use as file descriptor nodes (fdns). The minimum is 0. The maximum depends on the size of the RAM disk. An argument to the 'f' option must be less than the size of the RAM disk (after any rounding) minus 3. If the user specifies 0, "ramdisk" uses the default, which is 3% of the space on the RAM disk.

EXAMPLES

1. /etc/ramdisk /dev/ram0 1000
2. /etc/ramdisk ram1 0

The first example formats "/dev/ram0" with a maximum capacity of 1000 blocks.

The second example releases the memory in "/dev/ram1" to the operating system.

NOTES

- Not all versions of the UniFLEX Operating System support RAM disks.
- A RAM disk must be formatted before a user can access it. If a user tries to access an unformatted RAM disk, the operating system returns an I/O error.

ERROR MESSAGES

Argument to the 'f' option is too large.
The argument to the 'f' option must be less than the size of the RAM disk (after any rounding) minus 3.

"<dev_name>" is mounted.
The "ramdisk" command cannot format a mounted device.

"<dev_name>" is not a block device.
The device specified must be a block device.

"<dev_name>" is not a character device.
The "ramdisk" command tried to access the device associated with the specified block device and found that it was not a character device.

"<dev_name>" is not a RAM disk.
The device specified must be a RAM disk.

Error opening "<dev_name>": <reason>
The operating system returned an error when "ramdisk" tried to open the specified device. This message is followed by an interpretation of the error message returned by the operating system.

Error opening the mount table, "/etc/mtab": <reason>
The operating system returned an error when "ramdisk" tried to open the mount table. This message is followed by an interpretation of the error message returned by the operating system.

Error reading fdn for "<dev_name>": <reason>
The operating system returned an error when "ramdisk" tried to read the fdn associated with the specified device. This message is followed by an interpretation of the error message returned by the operating system.

Error reading the mount table, "/etc/mtab": <reason>
The operating system returned an error when "ramdisk" tried to read the mount table. This message is followed by an interpretation of the error message returned by the operating system.

Error setting size for "<dev_name>": <reason>
The operating system returned an error when "ramdisk" tried to set the size of the specified device. This message is followed by an interpretation of the error message returned by the operating system.

Insufficient memory for RAM disk.
The operating system will only allocate no more than 75% of the user-accessible memory available at the time the system is booted to all formatted RAM disks combined.

Invalid argument to the 'f' option: <arg>.
The argument to the 'f' option must be a whole number. The specified argument contains an invalid character.

Invalid option: '<char>'.
The only valid option to the "ramdisk" command is 'f'.

Invalid size specification.
The argument specifying the size must be a whole number between 0 and 8192 inclusive. The argument specified contains an invalid character.

Specified size exceeds maximum of 8192 blocks.
The argument specifying the size of the RAM disk cannot be greater than 8192.

Syntax: /etc/ramdisk <dev_name> [<size>] [+f]
The "ramdisk" command expects at least one and no more than two arguments, the second of which is a whole number between 0 and 8192 inclusive. This message indicates that the argument count is wrong.

You must be system manager to run "ramdisk".
Only the system manager may execute the "ramdisk" command.

SEE ALSO

create
makdev
mount
unmount

relinfo

Display information about an executable or relocatable file.

SYNTAX

relinfo <file_name_list> [+ehrs]

DESCRIPTION

The "relinfo" command displays information about the binary header, the symbol table, and both the relocation and external records in either an object file or all modules of a library. Normally, "relinfo" displays all the information. The available options restrict the display to the specified information (see Options Available).

Arguments

<file_name_list> A list of the names of files to report on.

Options Available

e Display only information about external records.
h Display only information about the binary header.
r Display only information about relocation records.
s Display only information about the global symbol table.

EXAMPLES

1. relinfo tester
2. relinfo /lib/mathlib +h
3. relinfo reporter +se

The first example displays information about the binary header, the symbol table, and both the relocation and external records in the object file "tester" in the working directory.

The second example displays the information about the binary headers from all the modules in the library "/lib/mathlib".

The third example displays the information about both the relocation and external records in the file "reporter" in the working directory.

ERROR MESSAGES

Error opening "<file_name>" : <reason>
The operating system returned an error when "relinfo" tried to open the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error reading "<file_name>" : <reason>
The operating system returned an error when "relinfo" tried to read the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error seeking to <location> in "<file_name>" : <reason>
The operating system returned an error when "relinfo" tried to seek to the specified location (in hexadecimal) in the specified file. This message is followed by an interpretation of the error returned by the operating system.

"<file_name>" is not a binary file!
The specified file does not have a valid binary header.

Invalid option: '<char>'.
The option specified by '<char>' is not a valid option to the "relinfo" command.

SEE ALSO

lib-gen68k
libinfo
load68k
rel68k

rel20

The "rel20" command is the 68020 relocating assembler.

SYNTAX

rel20 <file_name_list> [<param_list>] [+abefFiIJlLnosSu]

DESCRIPTION

The "rel20" command accepts one or more source files from the command line and, depending on the options specified, a single relocatable object-code module (also called a relocatable module), a listing of the assembled source code, or both. The assembler accepts most of the standard Motorola 68020 mnemonics and fully supports the 68020 instruction set. Differences between the Motorola standard instructions and the ones accepted by the assembler are described in detail in the manual, 68xxx UniFLEX Relocating Assembler and Linking-Loader.

Arguments

<file_name_list> A list of the names of files to assemble.

<param_list> A list of up to three parameters to pass to the assembler for use in the source code. The syntax for passing a parameter is

+<char>=[<str>]

where valid values for <char> are 'a', 'b', and 'c'. Normally, the shell program interprets a space character as a character separating two elements on the command line. A user may, however, include a space character in a command-line parameter by enclosing either the string or the entire parameter in a pair of delimiter characters (either single or double quotation marks). The user may specify the null string by leaving the string out altogether. If the user specifies one or more command-line parameters, the assembler searches the source code for substitutable parameters--code consisting of the following sequence:

&<char>

where valid values of <char> are once again 'a', 'b', and 'c'. The assembler replaces each substitutable parameter with the corresponding command-line parameter. For more details on command-line parameters see the 68xxx UniFLEX Relocating Assembler and Linking-Loader.

Options Available

a	Produce an abbreviated listing of the assembled source code. Such a listing contains only one line of output for each instruction. The output consists of an indicator character, the program counter, the first two bytes of the code generated by the instruction, the label, the instruction, the operands, and as much of any comment as possible.
b	Suppress binary output.
e	Suppress summary information.
f	Disable formatting of the listing of the assembled source code.
F	Enable "fix" mode. In fix mode, comments which begin with a semicolon, ';', are assembled.
i	Ignore the suffix ":w", which forces an address to the size of a word.
I	Ignore the suffix ":w", which forces an address to the size of a word, unless it is part of a "jmp" or a "jsr" instruction.
J	Ignore the suffix ":w", which forces an address to the size of a word, when it is part of a "jmp" or "jsr" instruction.
l	Produce a listing of the assembled source.
L	Produce a listing of the input file or files during the first pass.
n	Produce line numbers with the listing.
o=<file_name>	Specifies the name of the file containing the relocatable module produced by the assembler.
s	Produce a listing of the symbol table.
S	Limit symbols internally to 8 characters.
u	Classify all unresolved symbols as external.

EXAMPLES

1. rel20 asmfile
2. rel20 test.a +euo=test.r +a=D0
3. rel20 test.a test2.a test3.a +blns

(continued)

The first example assembles the source file "asmfile" and produces the relocatable binary file "asmfile.r". The assembler sends summary information to standard output but produces no source listing. Any errors detected are sent to standard output. The assembler does not search the source for substitutable parameters because the user did not specify any command-line parameters.

The second example assembles the file "test.a" and produces the relocatable file "test.r". No summary information is produced, and all unresolved references are classified as external. If the assembler detects no errors during the assembly, the user sees no output from this command. The assembler searches the source code for all substitutable parameters and replaces the sequence "&a" with the string "D0". It replaces any occurrences of "&b" and "&c" with the null string.

The third example assembles the three files specified but produces no binary output. A listing with a symbol table is sent to standard output. The listing includes line numbers. The assembler does not search the source for substitutable parameters because the user did not specify any command-line parameters.

SEE ALSO

68xxx UniFLEX Relocating Assembler and Linking-Loader

rel68k

The "rel68k" command is the 68000/68010 relocating assembler.

SYNTAX

rel68k <file_name_list> [<param_list>] [+befFiIJlLnosStu]

DESCRIPTION

The "rel68k" command accepts one or more source files from the command line and, depending on the options specified, a single relocatable object-code module (also called a relocatable module), a listing of the assembled source code, or both. The assembler accepts most of the standard Motorola 68000/68010 mnemonics and fully supports the 68000/68010 instruction set. Differences between the Motorola standard instructions and the ones accepted by the assembler are described in detail in the manual, 68xxx UniFLEX Relocating Assembler and Linking-Loader.

Arguments

<file_name_list> A list of the names of files to assemble.

<param_list> A list of up to three parameters to pass to the assembler for use in the source code. The syntax for passing a parameter is

+<char>=[<str>]

where valid values for <char> are 'a', 'b', and 'c'. Normally, the shell program interprets a space character as a character separating two elements on the command line. A user may, however, include a space character in a command-line parameter by enclosing either the string or the entire parameter in a pair of delimiter characters (either single or double quotation marks). The user may specify the null string by leaving the string out altogether. If the user specifies one or more command-line parameters, the assembler searches the source code for substitutable parameters--code consisting of the following sequence:

&<char>

where valid values of <char> are once again 'a', 'b', and 'c'. The assembler replaces each substitutable parameter with the corresponding command-line parameter. For more details on command-line parameters see the 68xxx UniFLEX Relocating Assembler and Linking-Loader.

Options Available

b	Suppress binary output.
e	Suppress summary information.
f	Disable formatting of the listing of the assembled source code.
F	Enable "fix" mode. In fix mode, comments which begin with a semicolon, ';', are assembled.
i	Ignore the suffix ":w", which forces an address to the size of a word.
I	Ignore the suffix ":w", which forces an address to the size of a word, unless it is part of a "jmp" or a "jsr" instruction.
J	Ignore the suffix ":w", which forces an address to the size of a word, when it is part of a "jmp" or "jsr" instruction.
l	Produce a listing of the assembled source.
L	Produce a listing of the input file or files during the first pass.
n	Produce line numbers with the listing.
o=<file_name>	Specifies the name of the file containing the relocatable module produced by the assembler.
s	Produce a listing of the symbol table.
S	Limit symbols internally to 8 characters.
t	Assemble for 68000 rather than 68010. This option only affects the code generation of the "move from CCR/SR" instruction.
u	Classify all unresolved symbols as external.

EXAMPLES

1. rel68k asmfile
2. rel68k test.a +euo=test.r +a=D0
3. rel68k test.a test2.a test3.a +blns

(continued)

The first example assembles the source file "asmfile" and produces the relocatable binary file "asmfile.r". The assembler sends summary information to standard output but produces no source listing. Any errors detected are sent to standard output. The assembler does not search the source for substitutable parameters because the user did not specify any command-line parameters.

The second example assembles the file "test.a" and produces the relocatable file "test.r". No summary information is produced, and all unresolved references are classified as external. If the assembler detects no errors during the assembly, the user sees no output from this command. The assembler searches the source code for all substitutable parameters and replaces the sequence "&a" with the string "DO". It replaces any occurrences of "&b" and "&c" with the null string.

The third example assembles the three files specified but produces no binary output. A listing with a symbol table is sent to standard output. The listing includes line numbers. The assembler does not search the source for substitutable parameters because the user did not specify any command-line parameters.

SEE ALSO

68xxx UniFLEX Relocating Assembler and Linking-Loader

remove

Remove the specified file name from the file system.

SYNTAX

remove <file_name_list> [+dklpqw]

DESCRIPTION

The "remove" command removes the specified file, which may be any type of file, from the file system. A user who is not the system manager must have write permission in the parent directory of the file being removed and, by default, must also have write permission in the file itself. The system manager may remove any file on the system. Restrictions on the deletion of a directory are discussed with the options.

Arguments

<file_name_list> A list of the names of files to remove from the file system. The list may include regular files, special files, and directories.

Options Available

- d If the specified file is a directory and it is empty, delete it. By default, the "remove" command does not delete directories.
- k If the specified file is a directory, delete it and all the files it contains.
- l List the name of each file as it is removed.
- p Prompt for permission to remove each file. The file is removed if the user responds to the prompt with a 'y'.
- q Do not print an error message if the specified file does not exist.
- w Prompt for permission to remove files for which the owner does not have write permission. By default, the "remove" command does not delete such files unless the user is the system manager. The file is removed if the user responds to the prompt with a 'y'.

EXAMPLES

1. remove first_file dir_file second_file +w
2. remove first_file dir_file second_file +dp
3. remove first_file dir_file +kl

(continued)

The first example removes the files "first_file" and "second_file", prompting for permission to do so if the owner does not have write permissions in the file. The file "dir_file" is not removed because it is a directory.

The second example prompts for permission to remove "first_file" and "second_file" (assuming the user has the proper permissions). It also prompts for permission to remove "dir_file" if it is empty.

The third example removes "first_file" and "dir_file" from the file system. In addition, it descends the directory structure of "dir_file", deleting the directory itself as well as every file and the contents of every file in the directory. The command lists the name of each file as it is deleted.

NOTES

- The "remove" command, especially when executed with the 'k' option, is an extremely powerful and potentially destructive command. It should be used with caution.

ERROR MESSAGES

Cannot delete the root directory: "/"
  The user tried to delete the root directory.

Directory "<dir_name>" is not empty.
  The "remove" command cannot delete a nonempty directory unless the user specifies the 'k' option.

Error deleting "<file_name>": <reason>
  The operating system returned an error when "remove" tried to delete <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Error deleting "." in "<dir_name>": <reason>
  The operating system returned an error when "remove" tried to delete the "." entry in <dir_name>. This message is followed by an interpretation of the error returned by the operating system.

Error deleting ".." in "<dir_name>": <reason>
  The operating system returned an error when "remove" tried to delete the ".." entry in <dir_name>. This message is followed by an interpretation of the error returned by the operating system.

Error getting status for "<file_name>": <reason>
The operating system returned an error when "remove" tried to read the fdn for <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Error opening "<file_name>": <reason>
The operating system returned an error when "remove" tried to open <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Error removing "<file_name>": <reason>
The operating system returned an error when "remove" tried to remove <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Invalid option: '<char>'
The option specified by <char> is not a valid option to the "remove" command.

Syntax: remove <file_name_list> [+dklpqw]
The "remove" command expects at least one argument. This message indicates that the argument is wrong.

You may not delete ".".
A user who wishes to delete the working directory must explicitly name the directory rather than using the symbol for it.

You may not delete "..".
A user who wishes to delete the parent of the working directory must explicitly name the directory rather than using the symbol for it.

SEE ALSO

delusr
kill

rename

Change the name of the specified file.

SYNTAX

rename <file_name_1> <file_name_2>

DESCRIPTION

The "rename" command tries to change the name of the specified file from <file_name_1> to <file_name_2>. In the process it makes a distinction between directories and all other types of file (text and special). If <file_name_2> specifies an existing directory, the "rename" command fails. If <file_name_1> specifies a directory, the name chosen for <file_name_2> must be a nonexistent directory with the same parent. If neither file specified is a directory, "rename" changes the name of <file_name_1> to <file_name_2>, deleting any existing file with the same name. In all cases the specified files must be on the same device.

Arguments

<file_name_1>	The name of an existing file or directory.
<file_name_2>	The new name for <file_name_1>. It may not specify an existing directory.

EXAMPLES

1. rename test oldtest
2. rename test /usr/elaine/oldtest
3. rename /usr/dir_1 /usr/dir_2

The first example changes the name of the file "test" in the working directory to "oldtest". If a file named "oldtest" already exists, it is deleted without warning.

The second example changes the name of the file "test" in the working directory to "/usr/elaine/oldtest".

The third example changes the name of the directory "/usr/dir_1" to "/usr/dir_2" as long as "/usr/dir_2" does not already exist.

ERROR MESSAGES

Both directories must have the same parent!
The "rename" command cannot change the name of a directory unless the new name represents a directory with the same parent.

Cannot rename across devices.
Both <file_name_1> and <file_name_2> must be on the same device.

Cannot rename root of a mounted device.
The "rename" command cannot rename the node at which a device is mounted.

Cannot rename "." or ".."!
The files "." and ".." in each directory are essential to the integrity of the structure of the disk. They may not be renamed.

Error locating parent of "<file_name>": <reason>
The operating system returned an error when "rename" tried to locate the parent of the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error renaming "<file_name_1>": <reason>
The operating system returned an error when "rename" tried change the name of <file_name_1>. This message is followed by an interpretation of the error returned by the operating system.

Error renaming to "<file_name_2>": <reason>
The operating system returned an error when "rename" tried to assign the new file name. This message is followed by an interpretation of the error returned by the operating system.

Error unlinking "<file_name_1>": <reason>
The operating system returned an error when "rename" tried to unlink <file_name_1> from the new file. This message is followed by an interpretation of the error returned by the operating system.

File "<file_name_1>" does not exist!
The first name on the command line must be the name of an existing file.

File "<file_name_2>" is a directory!
The "rename" command cannot assign the name of an existing directory to a file or directory.

Illegal file name specified: "<file_name>"
The name specified ends in a slash character, '/', and is therefore illegal.

Source and destination are same file!
Both <file_name_1> and <file_name_2> refer to the same file. (If their names are not the same, they are links to the same file.)

Syntax: rename <file_name_1> <file_name_2>
The "rename" command expects exactly two arguments. This message indicates that the argument count is wrong.

SEE ALSO

move

rerun

End the current print job, return the file to the print queue, and idle the specified printer program.

SYNTAX

```
rerun <splr_name>
```

DESCRIPTION

The "rerun" command ends the current print job, returns the file to the print queue, and idles the printer program. When the printer program is restarted with the "next" command, it starts printing the first job in the print queue, which may or may not be the job that was interrupted by the "rerun" command. This command is useful if, for instance, the paper on the printer gets out of alignment while a file is being printed.

Arguments

<splr_name>	The name of the spooler printing the job to rerun.

EXAMPLES

1. rerun ppr

This example tells the printer program associated with "ppr" to stop printing the current job, to replace the file it is printing in the queue, and to idle the printer.

NOTES

- The "rerun" command is one of five commands that are linked to the file "/etc/prcon", which controls the printing of files.

ERROR MESSAGES

Cannot find spooler directory for "<splr_name>".
The directory "/usr/gen" does not contain a directory for the specified spooler.

Error creating ".idl*splr?": <reason>
The operating system returned an error when "rerun" tried to create the file ".idl*splr?". This message is followed by an interpretation of the error returned by the operating system.

Error opening "<file_name>": <reason>
The operating system returned an error when "rerun" tried to open the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error reading "<file_name>": <reason>
The operating system returned an error when "rerun" tried to read the specified file. This message is followed by an interpretation of the error returned by the operating system.

"<splr_name>" is already idle.
The "rerun" command is effective only on an active spooler.

Syntax: rerun <splr_name>
The "rerun" command expects exactly one argument. This message indicates that the argument count is wrong.

You must be system manager to run "rerun".
Only the system manager may execute the "rerun" command.

SEE ALSO

end
idle
insp
next
print
pstop
purge

resume

Resume execution of a suspended task.

SYNTAX

```
resume <task_ID>
```

DESCRIPTION

The "resume" command resumes the execution of a task suspended by the "suspend" command. Only the system manager may execute this command.

Arguments

<task_ID> The task ID of the task to resume.

EXAMPLES

1. resume 125

This example resumes execution of task 125.

MESSAGES

Task <task_ID> resumed.
The "resume" command prints this message if it successfully resumes execution of the specified task.

ERROR MESSAGES

Cannot resume task: <reason>
The operating system returned an error when the "resume" command tried to resume the specified task. This message is followed by an interpretation of the error returned by the operating system.

Syntax: resume <task_ID>
The "resume" command expects exactly one argument. This message indicates that the argument count is wrong.

SEE ALSO

suspend

setpath

Display or redefine the names of the directories that the shell program searches when looking for an executable file.

SYNTAX

setpath [<dir_name_list>]

DESCRIPTION

The "setpath" command, which is part of the shell program, is used either to display or to redefine the list of names of the directories that the shell program searches when looking for an executable file. This list is known as the search path. The shell program searches the directories in the order in which the user specifies them on the command line.

Arguments

<dir_name_list> A list of the names of directories to use as the search path. The default list consists of the following directories: the user's working directory, "<home_dir>/bin", "/bin", "/usr/bin", and, if the user is the system manager, "/etc". (The home directory is the user's login directory, as specified in the password file.)

EXAMPLES

1. setpath . bin /bin /usr/games

This example sets the search path so that it consists of the user's working directory, the directory "bin" in the user's working directory, and the directories "/bin" and "/usr/games".

NOTES

- The "setpath" command is only effective while the shell program under which it is invoked is running. The list of directories searched by the login shell can be permanently altered by placing the appropriate command in the file ".startup" in the user's home directory. This file is automatically executed each time the user logs in.

SEE ALSO

addpath
shell

set_termcap

Interactively specify the capabilities of each terminal on the system.

SYNTAX

```
/etc/set_termcap <ttycap_file> <ttyassoc_file> <termcap_file>
/etc/set_termcap +d
```

DESCRIPTION

The "set_termcap" command allows the user to interactively specify the following about each terminal on the system: whether or not the terminal port may be used as a "login" terminal; what the default baud rate is; and what type of terminal it is. On completion the "set_termcap" command updates <ttyassoc_file> to reflect any changes and creates <termcap_file>.

For most applications, <ttyassoc_file> is the file "/etc/ttylist", which is used by the system to determine which terminals may be used for "login" purposes, and <termcap_file> is the file "/etc/termcap", which is used by screen-oriented programs to determine how to perform certain terminal-specific functions, such as cursor manipulation.

Arguments

<ttycap_file>	Input file describing the functional capabilities of each type of terminal that may be used on the system.
<ttyassoc_file>	Input file associating each active port with a particular kind of terminal and indicating whether or not the system may use the terminal as a "login" device.
<termcap_file>	Output file combining information from the two input files.

Options Available

d Use the file "/etc/ttycap" for <ttycap_file>, "/etc/ttylist" for <ttyassoc_file>, and "/etc/termcap" for <termcap_file>.

EXAMPLES

1. /etc/set_termcap /etc/ttycap /etc/ttylist /etc/termcap
2. /etc/set_termcap +d

In the first example, "set_termcap" displays a description of the first terminal in the file "/etc/ttylist" and asks the user whether or not the entry is to be changed. If the user chooses not to change the entry, it is left as is, and "set_termcap" displays a description of the next entry from "/etc/ttylist". If the user elects to change the entry, "set_termcap" first asks whether or not the terminal port should be enabled for "login" purposes. It then asks for the default baud rate. Next, "set_termcap" displays the first type of terminal listed in "/etc/ttycap" and asks the user whether or not it is the type of terminal desired. If the user neither opts to leave the type as is nor selects the type shown, "set_termcap" displays the next type of terminal. Finally, the user has the option of changing any of the optional information associated with the port.

Unless the user terminates the program, "set_termcap" continues with the next entry from "/etc/ttylist". When all entries from "/etc/ttylist" have been updated, "set_termcap" rewrites the file and uses the information in "/etc/ttycap" and "/etc/ttyassoc" to create "/etc/termcap".

The prompts used in this interaction are discussed in MESSAGES.

The second example is equivalent to the first.

MESSAGES

This section discusses the messages displayed by "set_termcap" during its interaction with the user. Any response other than one of the documented ones is ignored by "set_termcap", which considers that response equivalent to a carriage return.

Enter ´c´ to change an entry, ´q´ to terminate, ´n´ for next entry.
The "set_termcap" command issues this message just before displaying an entry from <ttyassoc_file> and asking whether or not the user wishes to change the entry. If the user enters a ´c´, "set_termcap" begins to prompt for changes to the entry; if the user enters a ´q´, "set_termcap" terminates after making the changes requested in the current session; if the user enters an ´n´ (or a carriage return), "set_termcap" displays the next entry from the file.

<sign><baud_code><nn>:<terminal_type>:[<optional_info>]: -- (c,n,q)?
This prompt describes the current configuration of a terminal. The value of <sign> may be a plus sign, ´+´, which indicates that the terminal is enabled for "login", or a minus sign, ´-´, which indicates that it is not. The baud rate is specified by an entry from the following table:

Code	Speed	Code	Speed
space	Hardware default	8	1200
1	75	9	1800
2	110	a	2400
3	134.5	b	3600
4	150	c	4800
5	200	d	7200
6	300	e	9600
7	600	f	19200

The string <nn> is a two-digit number representing the terminal, and <terminal_type> is the name for the type of terminal attached to the port. The <optional_info> is normally the name of the person most commonly using the terminal. However, it has no functional meaning and need not be present. The user may select to change ('c') the entry or leave it as is ('n' or carriage return).

-- Enable (+, -)?
This is the first prompt given when the user elects to change an entry. A response of '+' indicates that "set_termcap" should enable the port for "login"; of '-', that it should not. If the user responds with a carriage return, "set_termcap" leaves the status of the port as is.

-- Speed ?
The user should enter the desired baud rate for the terminal port in response to this prompt. If the hardware default is desired, the user should enter a period, '.'. Responding with a carriage return leaves the baud rate unchanged.

-- <terminal_type> (y,n,q)?
The "set_termcap" command issues a prompt for each type of terminal described in <ttyassoc_file> until the user selects a new type ('y'), the user elects to leave the entry unchanged ('q'), or the list of available selections is exhausted (in which case the type of terminal remains unchanged).

-- Description ?
After displaying any current optional information, "set_termcap" issues this prompt. A user who wishes to replace the optional information with new information may type as many as 127 printable characters. Tying a period, '.', removes the old entry without replacing it. Typing a carriage return leaves the field unchanged.

(continued)

ERROR MESSAGES

*** Can't access ttyassoc file "<file_name>"
The utility did not have read permissions in the file specified as the "ttyassoc" file.

*** Can't access ttycap file "<file_name>"
The utility did not have read permissions in the file specified as the "ttycap" file.

Can't find description of the terminal "<term_name>".
A terminal name specified in the "ttyassoc" file was not one of the terminal names contained in the "ttycap" file. The "termcap" file will not be created.

*** ERROR : <system_error_message> while <action>
This general class of error messages describes any system errors encountered while performing such functions as reading, writing, opening, or closing files.

Syntax: /etc/set_termcap <ttycap_file> <ttyassoc_file> <termcap_file>
/etc/set_termcap +d
Unless the user specifies the 'd' option, "set_termcap" expects exactly three arguments. Otherwise, it expects no arguments. This message indicates that the argument count is wrong.

*** Unrecognized option "<char_1>", Terminal = <terminal_name>,
Last valid option "<char_2>".
The option shown is not one of the legal options allowed in the "ttycap" file. The file "termcap" will not be built.

SEE ALSO

crt_termcap

shell

DESCRIPTION

The shell program is a command interpreter which is the primary interface between the user and the operating system. It is the program which runs by default when a user logs into the system and which sends the system prompt to the screen. The shell program collects and interprets the user's commands. It executes some commands, known as built-in commands (including "addpath", "chd", "dperm", "jobs", "log", "login", "nice", "prompt", "setpath", "time", and "wait") itself. It passes others to the UniFLEX core which, in turn, performs the operations requested.

Generally, the shell program is used interactively. In this form the command line consists of a command name, which may be followed by arguments and options, as appropriate. All elements of the command line must be separated by spaces. The command may be one of the commands supplied with the operating system, the name of a binary file produced by either the assembler or a compiler, the name of a BASIC compiled file, or the name of a text file (with execute permission turned on) which contains a series of commands to execute. In all cases the shell program spawns a subshell which executes the specified command. If the command name is the name of a BASIC compiled file, the shell program spawns a subshell which loads and executes BASIC, which, in turn, executes the specified compiled file.

Search Path

Because most commands reside on disk, the shell program must locate the command before executing it. By default, the shell program sequentially searches the following directories: the user's working directory, "<home_dir>/bin", "/bin", and "/usr/bin". If the user is the system manager, the system also searches the directory "/etc" immediately after searching "<home_dir>/bin". (The home directory is the user's login directory, as specified in the password file.)

The list of directories searched by the shell program is known as the search path. The user may add to this list with the "addpath" command or redefine it with the "setpath" command.

Multiple Commands on a Line

The user may specify more than one command on a command line by separating them with any of several special symbols. Parentheses may be used to group commands on a command line containing more than one

command. The shell program spawns a subshell to run each set of commands in parentheses.

The shell program sequentially executes commands that are separated by a semicolon, ';'. If a task terminates abnormally, the shell program stops executing the command line.

If the user follows a command with an ampersand, '&', the shell program, as usual, spawns a subshell which executes the command. However, in this case the shell does not wait for the task to complete before returning a prompt. Thus, the user may start another command while the first one is executing. A single shell program can support a maximum of five of these "background tasks". Each time the user sends a task to the background, the shell program reports the task ID assigned to that task, preceding it with a 'T', which is not part of the task ID. The user may need the task ID to execute the "wait" or "int" command. The task ID may also be obtained by executing the "jobs" command, which returns the task ID and starting time of all background tasks originated by the user at the current terminal from the shell program. The ampersand may be used following a single command or separating one task from another on the command line.

Two additional command separators, the conjunction operator ("&&") and the disjunction operator ("||"), are available. These separators make execution of the command following the operator dependent on the outcome of the execution of the command preceding it. A command is "true" if it terminates with a termination status of zero, indicating successful completion, and "false" if it terminates with a nonzero termination status, indicating failure. When two commands are separated by the conjunction operator, the shell program executes the second one only if it completes the first one successfully (it is "true"). When two commands are separated by the disjunction operator, the shell program executes the second one only if the first one fails (it is "false"). Normally, the command line is evaluated from left to right; however, parentheses may be used to group commands. Commands in parentheses are treated as a single command. Commands separated by a pipe (see Redirected I/O) are also treated as one command.

The processing of the command separators may be summarized as follows:

&& If the command preceding the conjunction operator succeeds, the shell program tries to execute the next command. If the command preceding the conjunction operator fails, the shell program looks for a disjunction operator. If it finds one, it tries to execute the command which follows it. If it does not find one, processing of the command line ceases.

|| If the command preceding the disjunction operator succeeds, the shell program looks for a semicolon, ';'. If it finds one, it tries to execute the command which follows it. If it does not find one, processing of the

command line ceases. If the command preceding the disjunction operator fails, the shell program tries to execute the next command.

; If the command preceding a semicolon succeeds, the shell program tries to execute the next command. If the command preceding a semicolon fails, processing of the command line ceases.

& Whether the command preceding a single ampersand succeeds or fails, the shell program processes the next command on the command line.

Consider the following example:

<task_1> && <task_2> || <task_3> && <task_4>

The shell program first tries to execute <task_1>. If the task is unsuccessful, the shell skips <task_2> and proceeds to <task_3>. If <task_3> fails, the shell program skips <task_4>; if <task_3> succeeds, it tries to execute <task_4>. If, however, <task_1> succeeds, the shell program tries to execute <task_2>. If <task_2> also succeeds, the shell program skips the rest of the command line. If, after the successful execution of <task_1>, <task_2> fails, the shell tries to execute <task_3>. If and only if <task_3> succeeds, it goes on to <task_4>.

The use of parentheses can change the interpretation of the same set of commands separated by the same operators:

<task_1> && (<task_2> || <task_3>) && <task_4>

In this case, the shell once again begins by trying to execute <task_1>. If it fails, the shell program skips the remaining tasks. If, on the other hand, <task_1> is successful, the shell program spawns a subshell (because of the presence of the parentheses). This subshell tries to execute <task_2> and, if and only if it fails, it tries to execute <task_3>. If <task_2> succeeds, it returns a termination status of "true" to its parent shell. If <task_2> fails but <task_3> succeeds, it also returns a termination status of "true". If, however, both <task_2> and <task_3> fail, the termination status returned is "false". If the termination status returned by the subshell is "true", the parent shell tries to execute <task_4>.

Termination Status

Normally, the shell program does not report the termination status of a command it executes unless the task terminates abnormally (because of a program interrupt). A list of the possible program interrupts appears in the documentation of the "int" command. The shell program does, however, always report the termination status of a background task, even

if it terminates normally.

Redirected I/O

The shell program associates three files with every command it executes: standard input, standard output, and standard error. Standard input is the file from which a command takes its input. Standard output is the file to which a command sends its output. Standard error is the file to which many error messages are directed. By default, the system uses the user's keyboard as standard input and the user's display as both standard output and standard error. However, the user can direct the shell program to use another file for any of these standard files. This process is known as I/O redirection.

The symbol '<' tells the shell program to redirect standard input to the file whose name follows the symbol. Similarly, the symbols '>' and '%' are used to redirect standard output and standard error. The file to which standard input is redirected must already exist. However, if the file to which standard output or standard error is redirected does not exist, the system creates it. In fact, if the file does already exist, the system deletes the contents of the file before executing the command. To avoid this effect, the user may instead direct the shell program to append data to the file specified as standard error or standard output by duplicating the symbol used for redirection. For example, to execute the "time" command on "ls" redirecting standard output to the file "time_out" and standard error to the file "time_err", the user types

```
time ls >>time_out %%time_err
```

If either of the specified files already exists, its contents remain intact and the relevant output from the command is added to the end of the file.

It is also possible to redirect standard output, standard error, or both to another task. This form of redirection is accomplished by using a "pipe". A pipe is a function that connects programs so that the output from one program becomes the input for another. Standard output is piped from one task to another by using one of the symbols '|' or '^'. For instance, the user can get a listing of all the files in the working directory, format the listing with the "page" command, and print the listing on the printer "spr" with the following command:

```
ls . | page | spr
```

Similarly, the user can redirect standard error with either of the symbols "%|" or "%^".

Although the user can place many pipes on the command line, a single task can only support one pipe. Thus, the user cannot pipe standard error and standard output to separate tasks. It is possible, however, to duplicate standard error onto standard output and to redirect them both to the same task. The user has a choice of symbols for duplicating standard error onto standard output: ">%" or "%>". Neither of these symbols takes an argument. After duplicating standard error onto standard output, the user redirects standard output to a file or a task in the usual way. For instance, the user can get a listing of all the files in the working directory, redirect both standard error and standard output to the "page" command, and print the results on the printer "spr" with the following command:

```
ls . >% | page | spr
```

Whenever standard error and standard output are routed to the same destination, their contents may be intermingled.

Finally, the following constructions redirect I/O from or to the null device, "/dev/null": "<-" for standard input, ">-" for standard output, and "%-" for standard error. If either standard output or standard error is redirected to the null device, its contents are lost. If the null device is used as standard input, an end-of-file character is read.

Continuation of the Command Line

Command lines may be continued across more than one physical line by terminating each line, except the last, with a backslash character, '\', immediately followed by a carriage return. By default, the shell program uses the prompt "+>" to indicate that the line being entered is a continuation of the previous line (the user may change the prompt with the "prompt" command). When the shell program processes the line, it replaces the backslash and the carriage return with a space. Typing a line-delete character (control-X) only affects the physical line being typed. The user may delete previous lines of a continued command line by typing a keyboard interrupt (control-C), which deletes the entire command line.

Matching Characters

The operating system recognizes several characters, known as matching characters, which allow the user to specify files with similar names without typing each name individually. The special characters are the asterisk, '*'; the question mark, '?'; and a pair of square brackets, "[]". The shell program matches these special characters to characters in the names of the entries in the specified directory according to the rules described in this section. If the matching character appears in the last component of the file name, the shell tries to match it to the

names of all files in the specified directory (by default, the working directory). If the matching character appears in any other position in the file name, the shell tries to match it to the names of directories only.

When the shell program encounters an asterisk in the command line, it matches it to any character or characters, including the null string but not including a leading period. Thus, the command

```
list *.bak ^ spr
```

lists all files in the working directory whose names end in ".bak" and do not begin with a period. The output is printed on the device "spr".

The question mark matches any single character except the null character or a leading period. For example, the command

```
list chapter_?
```

lists all files whose names begin with the string "chapter_" and end with a single character other than the null character. It is possible to use more than one matching character at a time. For instance, in response to the command

```
list *.?
```

the shell program lists all files in the working directory whose names end with a period followed by a single character (except, of course, those whose names begin with a period).

The use of square brackets allows the user to specify a set of characters to use in the matching process. The set of characters is defined by listing individual characters or by specifying two characters separated by a hyphen. In the former case, the shell program looks for all file names which use any one of the enclosed characters in the appropriate place. In the latter, the two characters specify a class of characters containing the two characters themselves and any characters which lexically fall between them in the ASCII character set. For example, if the user's working directory contains nine files named "chapter1", "chapter2", "chapter3", and so forth, the following command may be used to list the first three chapters, the fifth chapter, and the last three chapters:

```
list chapter[1-357-9]
```

If the shell program cannot find a match for any of the arguments containing matching characters, it aborts the command. If it finds a match for at least one argument containing matching characters, it ignores any other arguments containing matching characters for which it cannot find a match.

If the name of a file does actually contain one of the matching characters or a space character, the user must enclose the name in single or double quotation marks. In such a case the shell program passes the arguments to the command without performing any character matching.

Shell Scripts

A shell script is a file that contains a list of commands for the shell program. Such a file might consist of a list of commands that are frequently executed in sequence or of a single, lengthy command that is often used. If the user sets execute permissions on such a file, the name of the file can be used as a command.

The user may add to the versatility of a shell script by using arguments within the script. The arguments are specified within the script as "$1", "$2", "$3", and so forth. The argument "$0" specifies the name of the calling program. These arguments may appear anywhere in a command argument. For example, the following one-line script may be used to format a floppy disk:

```
/etc/formatfd +qnd=/dev/fd$1 +m=FD-$2
```

If this script is stored in an executable file named "f", the command

```
f 0 DD
```

formats the disk in drive 0 as double-sided, double-density.

If an argument being passed to a command actually contains an ampersand, it must be enclosed in single quotation marks so that the shell program does not try to perform any substitution. Note that single quotation marks prevent both substitution of arguments and the expansion of matching characters whereas double quotation marks prevent the expansion of matching characters but allow the substitution of arguments.

The shell program supports several commands that are used exclusively with shell scripts. These commands--"verbose", "exit", "proceed", and "sabort"--are discussed in this section.

When the shell program executes a script file, it does not normally echo the commands being executed. The "verbose" command causes the shell program to echo commands from a script file as they are executed. Each line that is echoed is preceded by two hyphens and a space character.

The "verbose" command may be called without arguments or with one argument, which must be one of the strings "on" or "off". If called without an argument, the default is "on". The command may be executed by the login shell or may be part of a shell script. The verbose attribute is always passed from a parent shell program to a child shell, but never from a child to a parent.

The shell program permits the user a limited amount of control over the processing of script files. Normally, it sequentially processes commands in a script file until either one of the commands fails or it reaches the end of the file. If one of the commands fails, the shell program begins to search the remainder of the script file for a line that contains one of the commands "exit" or "proceed". If it encounters one of these commands, the shell program resumes processing the script after that command. The only difference between the commands "exit" and "proceed" is that during successful execution of a script file the shell program stops processing the file if it encounters an "exit" command, whereas it ignores a "proceed" command. The search for both these commands takes place before both the substitution of any arguments and the expansion of any matching characters. Thus, the shell program does not see an "exit" or "proceed" command that is created as the result of either of these processes.

An example of the use of the "proceed" command follows:

```
/etc/mount /dev/fd0 /usr2
/usr2 runjob
echo "Successful execution."
proceed
/etc/unmount /dev/fd0
```

In this example, the shell program mounts a disk and tries to execute the command "/usr2/runjob" on that disk. If the command succeeds, the shell program echoes the message, "Successful execution." and proceeds to unmount the disk. If, on the other hand, the command fails, the shell program skips all commands between the one that failed and the "proceed" command. It resumes execution with the "unmount" command. Thus, if "/usr2/runjob" fails, the user's disk is unmounted, but no message is sent to standard output.

This example can be modified to notify the user of either successful or unsuccessful execution by using the "exit" command:

```
/etc/mount /dev/fd0 /usr2
/usr2/runjob
/etc/unmount /dev/fd0
echo "Successful execution."
exit
/etc/unmount /dev/fd0
echo "Unsuccessful execution."
```

In this example, if "/usr2/runjob" succeeds, the shell program continues execution with the "unmount" command and echoes the string "Successful execution." to standard output. The "exit" command then causes the shell program to stop processing the script because it encounters the "exit" command during normal execution. If "/usr2/runjob" fails, the shell program skips all commands until it encounters the "exit" command. It then resumes execution with the "unmount" command and echoes the string "Unsuccessful execution." to standard output.

The user may at times wish to force the execution of every command in a shell script regardless of the failure of previous commands. The "sabort" command can be used to turn off the search for either an "exit" or "proceed" command, thus forcing execution of every command in the script.

The "sabort" command may be called without arguments or with one argument, which must be one of the strings "on" or "off". When "sabort" is "on", the shell program looks for an "exit" or "proceed" command whenever a command in the script fails. When "sabort" is off, the shell program processes every command in the script. If the user executes the "sabort" command without an argument, it both rescinds the effect of any previous "sabort on" and fails. Thus, if it is executing a shell script, the shell program immediately begins looking for an "exit" or "proceed" command.

The "sabort" command may be executed by the login shell or may be part of a shell script. The attribute is always passed from a parent shell program to a child shell, but never from a child to a parent.

System Script Files

Whenever the system goes from single- to multi-user mode, it automatically executes the file "/etc/startup" if it exists. This file may, therefore, be used to execute commands which the system manager would otherwise have to execute manually each time the system is booted. Such commands might include the killing of any stray temporary files and the activation of all printer spoolers on the system.

In addition to this system script file, the system also supports startup files for individual users. Whenever a user logs in, the shell program looks for a file named ".startup" in the user's home directory (as defined in the password file). If the file exists and the user has read

permissions in it, the shell executes the file before issuing the system prompt.

The shell program can also be used as a command in its own right. This form is used primarily to execute a shell script for which execute permissions are not set, to call the shell program from another program, or in the password file. The documentation for its use in this way follows.

SYNTAX

shell [+abclnvx] [<argument_list>]

DESCRIPTION OF THE "SHELL" COMMAND

If the "shell" command is executed without any options or arguments, the operating system simply spawns another shell for the user. This shell program functions as a normal shell, but because it is the child of the shell program from which the command was executed, it does not know what the user's home directory is. The "log" command returns control to the parent shell.

The "shell" command can also be executed with options only. This form of the command also spawns a shell program that interacts with the user. If used in the password file, the command should be executed with the 'l' option (see Options Available).

Finally, the "shell" command can be executed with arguments or with both options and arguments. This form may be used, for example, to execute a shell script for which the user does not have execute permissions. Either of the following commands executes the file "script":

```
shell script
shell <script
```

The shell program first checks to see that the file specified as an argument is actually a file containing commands. If it is not, the shell does not execute it unless the user specifies the 'c' option (see Options Available).

Arguments

<argument_list> A list of arguments to pass to the shell command. Each element in the argument list consists of a command name followed by the appropriate arguments and options. The elements in the list must be separated by a valid command separator (';', '&', "&&", or

"||"). If any separator characters are used, the entire argument list must be enclosed in single or double quotation marks.

Options Available

Options specified to the shell program must appear immediately after the name "shell" on the command line, so that they are not confused with options that pertain to the arguments passed to the shell.

- a Start execution with the "sabort" attribute off.
- b Ignore control-C and control-\.
- c Process the argument list as a command.
- l Run as a login shell. A login shell tries to find the name of the user's home directory by looking in the file ".home?". It also automatically executes the file ".startup" in the working directory.
- n Run all background tasks with lowered priority (as does the "nice" command).
- v Start execution with the verbose attribute on.
- x Execute the next command without forking unless necessary. This option is only used when calling a shell program from another program.

NOTES

. It is impossible to specify a null string as an argument to a command because the shell program removes null strings from the command line.

ERROR MESSAGES

Built-in commands may not use pipes.
Input to or output from the shell built-in commands may not be routed through a pipe.

Cannot execute "<cmd_name>".
The operating system was unable to execute the specified command. Either the command does not exist or the user does not have execute permission.

Cannot initialize tables.
This error, which should not occur, is usually indicative of a hardware failure. If it does occur, contact the vendor.

(continued)

Cannot open I/O redirection file.
The operating system returned an error when the shell program tried to open the file specified for I/O redirection. Most probably, the path specified cannot be followed (one of the directories does not exist) or the user does not have the permissions necessary for opening the file. This message is preceded by an interpretation of the error produced by the operating system.

Cannot open pipe.
The operating system returned an error when the shell program tried to open the specified pipe. This message is preceded by an interpretation of the error produced by the operating system.

Error opening a file.
The operating system returned an error when the shell program tried to open the specified file. This message is preceded by an interpretation of the error produced by the operating system.

Error reading a file.
The operating system returned an error when the shell program tried to read the specified file. This message is preceded by an interpretation of the error produced by the operating system.

Error writing a file.
The operating system returned an error when the shell program tried to write to the specified file. This message is preceded by an interpretation of the error produced by the operating system.

I/O redirection conflict.
The user tried to redirect standard input, standard output, or standard error to more than one place.

I/O redirection error.
The operating system returned an error when the shell program tried to perform the specified I/O redirection. This message is preceded by an interpretation of the error produced by the operating system.

Memory overflow.
There is not enough memory available to perform the specified command. Most probably, the expansion of the matching characters used on the command line, for which many matches were possible, caused the error.

Missing "]" or invalid character range.
Either the right-hand square bracket is missing from the specification of a range of matching characters, or the range specified is invalid.

(continued)

No matching file names found.
Matching characters appear on the command line, but no file names match the specified pattern.

Parenthesis usage error.
The parentheses used on the command line are unbalanced.

Too many tasks.
The shell program tried to execute a fork, but too many tasks were running at the time. The limit to the number of tasks allowed either to the individual user or to the operating system as a whole was reached.

Unknown error.
This error should not occur. If it does, contact the vendor.

Unrecognized argument to builtin command.
The argument specified is not a valid argument to the built-in command in question.

Unterminated string.
The quotation marks used on the command line are unbalanced.

SEE ALSO

addpath
chd
dperm
env
hangup
jobs
log
login
nice
prompt
setpath
time
wait

shutup

Take the system from multi-user mode to single-user mode.

SYNTAX

/etc/shutup [[-]<minutes>]

DESCRIPTION

The "shutup" command takes the system from multi-user mode to single-user mode. In doing so, it sends a message at various time intervals to all terminals that are logged in to warn users of the impending switch to single-user mode. By default, the "shutup" command waits fifteen minutes, issues a hang-up interrupt to each task, waits fifteen seconds, and puts the system into single-user mode. This procedure gives any tasks that are executing when "shutup" is executed the opportunity to terminate cleanly without the loss of data.

Only the system manager may execute this command. When "shutup" is invoked, the system reports its task ID, so that the system manager can interrupt the command if necessary.

Arguments

-	The hyphen, or minus sign, tells the "shutup" command to omit the hang-up interrupts and the 15-second delay.
<minutes>	The number of minutes to wait before switching modes. The minimum value which may be used is 0, which causes the system to switch modes immediately; the maximum is 60. The default is 15.

EXAMPLES

1. /etc/shutup 30
2. /etc/shutup 0
3. /etc/shutup -0

The first example causes the system to enter single-user mode thirty minutes after it is issued. When the thirty minutes have passed, the "shutup" command issues a hang-up interrupt to each task, waits fifteen seconds, and puts the system into single-user mode.

The second example issues a hang-up interrupt to each task, waits fifteen seconds, and puts the system into single-user mode.

The third example causes the system to enter single-user mode immediately, without issuing any interrupts and without delaying.

ERROR MESSAGES

You must be system manager to run "shutup".
  Only the system manager may execute the "shutup" command.

SEE ALSO

int
stop

status

Write to standard output a report on the status of all tasks belonging to the user.

SYNTAX

```
status [+alswx]
```

DESCRIPTION

The "status" command reports the status of tasks running on the system to standard output. By default, this report does not include shell or login programs and is restricted to tasks belonging to the user who executes the command. The command is not always completely accurate due to the dynamic nature of the operating system. By default, the "status" command reports on the following parameters:

Task-id	The number assigned to the task by the operating system.
Mode	Indicates whether the task is in memory ('c') or has been swapped to the disk ('s').
tty	The number of the terminal from which the task originated. An "xx" in the field indicates that no terminal is associated with the task.
Prio	If the entry in this field is a number, it indicates the priority of the task. A higher number indicates a higher priority. Other priorities are described in the Table 1.

Table 1. Possible Priorities for a Task.

Priority	Meaning
buf	Waiting for a system buffer.
disk	Waiting for some disk activity.
file	Waiting for some file activity.
halt	Halted by another task.
in	Waiting for input from the terminal.
out	Waiting for output to the terminal to end.
pipe	Waiting for pipe data (usually input).
upd	Updating an fdn.
slp	Sleeping (not executing).
swap	Being swapped to or from the disk.
sys	Highest possible priority.
wait	Waiting for another task to end.

(continued)

Time	If the command is "System", this parameter is the amount of unused CPU time since the system was booted. Otherwise, it is the total CPU time that a particular task has used.
Command	The command which originated the task. By default, the "status" command shows the first thirty-five characters of the command line; the rest are truncated. The command "System" is the operating system. The command "/etc/init" executes the login program. If the "status" command cannot determine what was on the command line, this field contains the entry "???".

Options Available

a	List all tasks on the system, not just those belonging to the user.
l	Produce a more detailed description of the status of each task.
s	Produce a statistical summary of the use of the operating system.
w[=<num>]	Wait <num> seconds after reporting the status; then produce another report. The command repeats 100 times. The default is thirty seconds.
x	List all tasks (a normal listing does not include shell programs, the "System" command, or the command "/etc/init").

If the user specifies the 'l' option, the following additional items are included in the report:

Status	The status of the task. Possible values include run (task is running), sleep (task is waiting for something to happen), and term (the task has terminated).
User	The user name of the person who owns the task. If two or more user names share the same user ID, "status" uses the name that first appears in the password file.
Parent	The task ID of the parent task. If the parent task in no longer active, the ID shown in this field is 1.
Size	The amount of memory that the task is using.
Res	A rough measure of the amount of time a task has been in memory or swapped out to the disk. Each unit represents four seconds. The largest number that is ever displayed is 255. This number is set to 0 whenever a task is swapped into or out of memory.

(continued)

If the user specifies the 's' option, those of the following statistics which apply to the particular system are included in the report. They represent activity on the system since the time the system was booted.

Total block I/O transfer attempts.
The number of times the system has tried to access a disk block in the cache.

Total disk I/O operations.
The number of times the system has had to access the disk. This statistic does not include swap operations.

Total blocks freed.
The number of blocks that have been released from a file to the free list. If the same block has been released more than once, each release is counted.

Total system calls.
The number of times the system has executed a system call.

Total PAGE IN operations.
The number of times the system has read a page from the swap device. This statistic applies only to virtual-memory systems.

Total swap operations.
The number of times the system has swapped a task to or from the disk. This statistic applies only to systems without virtual memory.

Total PAGE OUT operations.
The number of times the system has written a page to the swap device. This statistic applies only to virtual-memory systems.

Total memory copy operations.
The number of times the operating system succeeded in enlarging a task by copying it directly to a larger space in memory without having to swap the task. This statistic applies only to systems without virtual memory.

Total pages stolen.
The number of times the system had to take memory from one user to give to another. This statistic applies only to virtual-memory system.

Total pages copied.
The number of 4K pages the operating system had to move in the process of making memory-to-memory copies. This statistic applies only to systems without virtual memory.

(continued)

EXAMPLES

1. status +s
2. status +alxw=15

The first example writes to standard output the default information about the status of all tasks except shell programs that belong to the user. A summary of the use of the operating system is included in the output.

The second example writes to standard output detailed information about the status of all tasks on the system. It waits fifteen seconds, then issues another report. The command repeats 100 times unless the user interrupts it by typing a control-C.

ERROR MESSAGES

Invalid option: '<char>'.
The option specified by '<char>' is not a valid option to the "status" command.

stop

Bring the system to a halt.

SYNTAX

/etc/stop [[-]<minutes>]

DESCRIPTION

The "stop" command halts the system. In doing so, it sends a message at various time intervals to all terminals that are logged in to warn users of the impending system shutdown. By default, "stop" issues a hang-up interrupt to each task and waits for fifteen seconds before shutting down the system. This procedure gives any tasks that are executing when "stop" is executed the opportunity to terminate cleanly without the loss of data. Using the "stop" command is the only safe way to shut down the system.

Only the system manager may execute this command. When "stop" is invoked, the system reports its task ID, so that the system manager can interrupt the command if necessary.

Arguments

-	The hyphen, or minus sign, tells the "stop" command to omit the hang-up interrupts and the 15-second delay.
<minutes>	The number of minutes to wait before shutting down the system. The minimum value which may be used is 0; the maximum, 60. The default is 0.

EXAMPLES

1. /etc/stop 30
2. /etc/stop 0
3. /etc/stop -0

The first example causes the system to shut down thirty minutes after it is issued. When thirty minutes have passed, the "stop" command issues a hang-up interrupt to each task, waits fifteen seconds, and shuts down the system.

The second example issues a hang-up interrupt to each task, waits fifteen seconds, and shuts down the system.

The third example causes the system to shut down immediately, without issuing any interrupts and without delaying.

ERROR MESSAGES

You must be system manager to run "stop".
Only the system manager may execute the "stop" command.

SEE ALSO

int
shutup

strip

Remove the symbol table from an executable binary file.

SYNTAX

```
strip <file_name_list>
```

DESCRIPTION

The "strip" command removes the symbol table from an executable binary file.

Arguments

<file_name_list> A list of files to process.

EXAMPLES

1. strip testprog

This example removes the symbol table from the executable binary file "testprog".

ERROR MESSAGES

Error creating "<file_name>": <reason>
The operating system returned an error when "strip" tried to create the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error opening "<file_name>": <reason>
The operating system returned an error when "strip" tried to open the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error unlinking "<file_name>": <reason>
The operating system returned an error when "strip" tried to unlink the specified file. This message is followed by an interpretation of the error returned by the operating system.

File "<file_name>" cannot be located.
The specified file does not exist.

File "<file_name>" is a device or a directory.
The specified file is not a regular file.

SEE ALSO

load68k
relinfo
rel20
rel68k

su

Temporarily log in as a new user without changing the working directory.

SYNTAX

su [<user_name>]

DESCRIPTION

The "su" command allows the user to log in as another user without logging out and without changing the working directory. If a name is specified on the command line, that name becomes the new login name. If no name is specified, "system" is used. If a password exists for the login name specified, "su" prompts for the password.

Although the "su" command does prompt for a password, it essentially ignores the rest of the new user's entry in the password file ("/etc/log/password"). Rather than making the new user's home directory the working directory, "su" does not change the working directory. Therefore, it does not execute any startup file (".startup") that is normally executed when the new user logs it. In addition, "su" ignores any designation of a login program in the password file; instead, it always executes a shell program.

One advantage of this command is that when finished as this new user, the user does not need to log in again but simply logs out and returns to the state that existed prior to the execution of the "su" command.

Arguments

<user_name>	The name of the user as whom to temporarily log in. The default name is "system".

EXAMPLES

1. su mary

This example temporarily logs in the user as "mary" (assuming the user knows the password if one exists). The working directory does not change.

NOTES

. The "su" command creates an environment for the shell program just as the "login" command does, with the exception that the value of the parameter TERM is taken from the parent task. If TERM is not defined in the parent task's environment, "su" determines its value from the file "/etc/ttylist".

SEE ALSO

log
login
newuser

suspend

Suspend the execution of a task.

SYNTAX

```
suspend <task_ID>
```

DESCRIPTION

The "suspend" command temporarily halts the execution of the specified task. A task remains suspended until it is resumed or interrupted. Only the system manager may execute this command.

Arguments

<task_ID> The task ID of the task to suspend.

EXAMPLES

1. suspend 125

This example suspends task 125.

MESSAGES

Task <task_ID> suspended.
The "suspend" command prints this message if it succeeds in suspending the task.

ERROR MESSAGES

Cannot suspend task: <reason>
The operating system returned an error when the "suspend" command tried to suspend the specified task. This message is followed by an interpretation of the error returned by the operating system.

Syntax: suspend <task_ID>
The "suspend" command expects exactly one argument. This message indicates that the argument count is wrong.

SEE ALSO

resume
int

tail

Write the end of the specified file.

SYNTAX

```
tail <file_name> [<count>]
```

DESCRIPTION

By default, the "tail" command writes to standard output the last 250 characters in the specified file. If the first character written comes from the middle of a line, the "tail" command begins that line with an ellipsis, "...", to indicate the omission. It prints all other lines exactly as they appear in the file.

Arguments

<count>	The number of characters to print. The number specified must be a positive integer. If it is greater than the number of characters in the file, the "tail" command prints the entire file. The default is 250.
<file_name>	The name of the file from which to print the last <count> characters.

EXAMPLES

1. tail module_1
2. tail module_1 600

The first example writes the last 250 characters of the file "module_1" to standard output.

The second example writes the last 600 characters of the file "module_1" to standard output.

ERROR MESSAGES

Error opening "<file_name>": <reason>
The operating system returned an error when "tail" tried to open the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error reading "<file_name>": <reason>
The operating system returned an error when "tail" tried to read the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error seeking in "<file_name>": <reason>
The operating system returned an error when "tail" tried to seek in the specified file. This message is followed by an interpretation of the error returned by the operating system.

Error writing to standard output.
The operating system returned an error when "tail" tried to write to standard output. This message is followed by an interpretation of the error returned by the operating system.

Invalid count specified: <count>
The argument <count> must be a positive integer.

Syntax: tail <file_name> [<count>]
The "tail" command expects as arguments the name of exactly one file and, optionally, a positive integer. This message indicates that the user specified no arguments.

time

Execute the specified command and report to standard error the amount of time required.

SYNTAX

time <command_name>

DESCRIPTION

The "time" command, which is part of the shell program, executes the specified command and reports to standard error the amount of real time elapsed during execution of the specified task, the amount of CPU time used by the system, and the amount of CPU time used by the user.

Arguments

<command_name> The name of the command to execute and time.

EXAMPLES

1. time pascal test.p >test_time

This example compiles the Pascal program "test.p" and sends to standard error a report of the amount of time used.

ERROR MESSAGES

No command or a built-in command was specified.
The "time" command expects exactly one argument, and that argument may not be one of the commands, such as "jobs" or "wait", which is a part of the shell program.

touch

Set the time of the last modification of a file to the current date and time.

SYNTAX

touch <file_name_list>

DESCRIPTION

The "touch" command sets the time of last modification for the specified file to the current date and time. The user must have read and write permission in a file in order to "touch" it. This command is often used in conjunction with the "update_all" command. It is also useful for correcting the last modification time of a file which was created or modified when the system time was incorrect.

Arguments

<file_name_list> A list of the names of the files to modify.

EXAMPLES

1. touch letter memo

This example changes the modification time of the files "letter" and "memo" to the current date and time.

ERROR MESSAGES

Error seeking to beginning of file "<file_name>": <reason>
The operating system returned an error when "touch" tried to seek to the beginning of <file_name>. This message is followed by an interpretation of the error returned by the operating system.

Error touching "<file_name>": <reason>
The operating system returned an error when "touch" tried to change the last modification time of <file_name>. This message is followed by an interpretation of the error returned by the operating system.

File "<file_name>" does not exist!
The "touch" command could not find <file_name> in the file system.

SEE ALSO

date
update_all

ttyset

Examine or adjust the parameters associated with the user's terminal.

SYNTAX

ttyset [<param_list>] [+]

DESCRIPTION

The "ttyset" command displays or changes the configuration of various system parameters relating to the user's terminal. The default configuration for all terminals is as follows:

-raw	+echo	+tabs	-case
+alf	+becho	-schar	-cntrl
+esc	-xon	-any	
8 Data	1 Stop	none	B9600
crt	bs=08	dl=18	

Arguments

<param_list> The list of parameters to adjust. If the user specifies no parameters, the command writes the current configuration of the user's terminal to standard output.

Format for Arguments

+raw Set raw I/O mode. This mode is normally used only by special programs.

+echo Enable character echo.

+tabs Expand tabs on output. This parameter should be set for terminals which do not perform tab expansion, so that the system expands tabs to every eighth column.

+case Convert upper- to lowercase on input and lower- to uppercase on output.

+alf Output a line-feed character after each carriage return.

+becho Enable echo of backspace character. When enabled, this parameter outputs an extra backspace and space character to erase on-screen characters for those terminals which do not automatically do so.

+schar Accept input one character at a time. This mode is normally used only by special programs.

(continued)

+cntrl	Ignore meaningless control characters (all control characters except carriage return, horizontal tab, control-C, control-D, control-backslash, backspace, and the line-delete character). When enabled, this parameter keeps strange, unwanted characters out of the user's files.
+esc	Enable the escape character (hexadecimal 1B) for starting and stopping output.
+xon	Enable the XON/XOFF mechanism for starting and stopping output. Control-S stops output; control-Q starts it. When output is stopped, XOFF characters are ignored; when output is not stopped, XON characters are ignored.
+any	Accept any character to start output after it has been stopped by either an escape character or a control-S.
bs=<num>	Set the backspace character to hexadecimal <num>. Default is hexadecimal 08 (control-H).
dl=<num>	Set the line-delete character to hexadecimal <num>. Default is hexadecimal 18 (control-X).

All parameters beginning with a plus sign, '+', may instead begin with a minus sign, '-', to set the opposite condition. In addition, the user may set the following parameters:

DB=<7_or_8>	Set the number of data bits.
SB=<1_or_2>	Set the number of stop bits.
even	Even parity mode.
odd	Odd parity mode.
none	No parity.
B<num>	Set the baud rate to <num>. Legal values of <num> are 75, 110, 134.5, 150, 200, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, 9600, and 19200.

Only certain combinations of values for data bits, stop bits, and parity are legal. The legal configurations are shown in Table 1.

(continued)

Table 1. Legal Configurations for Data Bits, Stop Bits, and Parity

Data Bits	Stop Bits	Parity
7	2	Even
7	2	Odd
7	1	Even
7	1	Odd
8	2	None
8	1	None
8	1	Even
8	1	Odd

The following delays can be set for mechanical terminals which require time for long carriage movements, such as carriage returns. Experimenting with the various delays determines the correct setting for a particular terminal.

crt Set delays for crt (no delays).
hcs Set slow hard-copy delays.
hcm Set medium hard-copy delays.
hcf Set fast hard-copy delays.

Options Available

\+ Display the current values for all "ttyset" parameters after making the changes.

EXAMPLES

1. ttyset
2. ttyset +alf -becho
3. ttyset +tabs hcm +

The first example displays the current values of the "ttyset" parameters for the user's terminal.

The second example enables the automatic output of a line-feed character after a carriage return and disables the echo of the backspace character.

The third example enables the expansion of tab characters on output and sets a medium speed, hard-copy delay. The command displays the current "ttyset" parameters after making the changes.

NOTES

- The "ttyset" command is generally used to set the configuration of the terminal from which it is called. Assuming permissions are granted, it is possible to execute the "ttyset" command on another terminal or even a serial printer device by redirecting input from the desired device. For example, the following command enables XON/XOFF processing on the serial printer "spr":

```
ttyset +xon </dev/spr
```

  Although the "ttyset" command can be performed on a serial printer device, the only argument which can be set or which has any effect is the "xon" argument.

- The "ttyset" parameters "raw", "schar", and "tabs" are reset by the shell program each time it issues a prompt for a command. Thus, if the user types the command "ttyset +raw" in response to a prompt from the shell program, raw I/O mode is enabled, but when the command is complete, the shell program turns raw I/O mode off before issuing the next prompt. To the casual observer it appears that the "ttyset" command failed. If, however, the same command is performed by an "exec" statement in a BASIC program, raw I/O mode remains in effect for the duration of the BASIC program (unless specifically turned off by another "ttyset" operation) because the shell program is not reentered.

- Not all hardware supports all legal baud rates, and not all hardware allows the dynamic changing of baud rates.

- The baud rate can be set either by using the "ttyset" command or by editing the file "/etc/ttylist". Each entry for an active terminal consists of a plus sign, '+', followed by a space, followed by the two-digit number that represents that particular terminal. Replacing the space with a single hexadecimal digit changes the baud rate. The digits correspond to the legal baud rates with a '1' representing a baud rate of 75 and an 'f' representing a baud rate of 19200.

ERROR MESSAGES

Error getting terminal status: <reason>
The operating system returned an error when "ttyset" tried to get information on the current configuration of the terminal. This message is followed by an interpretation of the error returned by the operating system.

Error setting terminal characteristics: <reason>
The operating system returned an error when "ttyset" tried set the new parameters. This message is followed by an interpretation of the error returned by the operating system.

Illegal baud rate specified: <num>
Command aborted!
Legal values for the baud rate are 75, 110, 134.5, 200, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, 9600, and 19200.

Illegal configuration: '# Data bits, # Stop bits, <set> Parity'.
The combination of values used to set data bits, stop bits, and parity is illegal. See Table 1 for a list of the legal combinations.

Invalid parameter, "<param>", found.
Command aborted!
The parameter specified by <param> is not a valid parameter for the "ttyset" command.

tune

Change the specified parameters in a file containing a copy of the UniFLEX Operating System.

SYNTAX

```
/etc/tune <file_name> [<param_list>]
/etc/tune <file_name> [+r]
```

DESCRIPTION

The "tune" command is used to alter certain parameters which govern the behavior and performance of the UniFLEX Operating System. The command operates in three modes: read-only, interactive, and automatic. In read-only mode "tune" displays the current value, in the specified file, of each the of parameters it can alter. If the user does not have write permission in the specified file, "tune" automatically executes in read-only mode. A user who does have write permission in the file can execute "tune" in read-only mode by specifying the 'r' option.

If the user has write permission in the specified file and specifies neither a parameter list nor the 'r' option, "tune" executes in interactive mode. In this mode it displays current values one by one. To change the value of a parameter the user enters the new value and a carriage return following the display. To leave the value as is the user types just a carriage return. The "tune" command imposes certain restrictions on the values of the parameters it alters. If the user enters a value in interactive mode which violates one of these restrictions, "tune" does not alter that value. Rather, it responds with an error message and waits for the user to enter a new value.

Instead of going through the entire list of parameters interactively, the user can directly specify on the command line which values "tune" is to change. The value of a parameter not mentioned on the command line does not change. After it makes the changes, "tune" displays a list of all parameters and the values specified by the user. If any of the values violates the restrictions mentioned previously, "tune" displays a message to that effect and sends a bell (control-G) to the terminal. Although the display shows the values specified by the user, "tune" does not make changes in the file if the changes violate the restrictions.

Arguments

<file_name>	The name of a file that contains a copy of the operating system.
<param_list>	A list of the parameters to change and the values to assign to them.

Format for Arguments

```
<param_list>   <param_name>=<decimal_number>
               <param_name>=<maj_dev_num>/<min_dev_num>
```

Table 1 lists the parameters that "tune" can change.

Table 1. Parameter names and descriptions

<param_name>	Description
buffers	Number of system buffers.
iolists	Maximum number of lists of I/O characters.
DST	Flag for the observation of Daylight Savings Time (0 indicates it is not observed locally; 1, that it is).
files	Maximum number of files that can be open at one time.
locked_recs	Maximum number of entries allowed in the table of locked files.
mounts	Maximum number of devices that can be mounted.
page_space	Maximum number of pages to be used by paging device (up to the limit set by the "format" command). One page equals 8 blocks of disk.
pipe_dev	Major and minor device numbers of the pipe device.
root_dev	Major and minor device numbers of the root device.
seek_rate	Seek rate of the floppy disk drive.
page_dev	Major and minor device numbers of the paging device.
tasks	Maximum number of tasks the system can simultaneously execute.
text_segs	Maximum number of unique shared-text programs that the operating system can simultaneously execute.
time_limit	Maximum number of seconds a task can run.
time_zone	Time difference in minutes between local time and Universal Time. A positive value of "time_zone" indicates the number of minutes west of Greenwich; a negative value, the number of minutes east.
user_tasks	Maximum number of tasks allowed to each user.

The values for all these parameters are originally set when one of the "/uniflex" files is copied from the master disk to a system disk. These default values, as well as the limits imposed on each parameter, are shown in Table 2.

(continued)

Table 2. Default Values and Limits for "tune" Parameters

<param_name>	Default	Minimum	Maximum
buffers	sd	8	192
iolists	sd	8	255
DST	0	0	1
files	sd	16	255
locked_recs	32	0	Value of "files"
mounts	5	2	32
pipe_dev	sd	0/0	Last block device number
root_dev	sd	0/0	Last block device number
seek_rate	0	0	sd
page_dev	sd	0/0	Last block device number
page_space	sd	256	16 Megabytes
tasks	sd	8	128
text_segs	20	2	20
time_limit	0	0	32767 (0 = no limit)
time_zone	300	-1440	1440
user_tasks	10	5	Number of tasks

Notes: sd = system-dependent

Options Available

r Execute the command in read-only mode--that is, display the current value for each parameter, but do not make any changes.

EXAMPLES

1. /etc/tune /uniflex +r
2. /etc/tune /usr2/uniflex-WIN tasks=32 page_dev=00/01

The first example displays a list of the items that "tune" can adjust. The current value of each item in the file "uniflex" in the root directory appears in parentheses.

The second example changes the specified parameters in the file "uniflex-WIN" (the mini-Winchester version of the operating system) in the directory "/usr2". Presumably, a system-build disk is mounted on "/usr2". This command sets the maximum number of tasks allowed on the system to 32 and defines the device whose major device number is 0 and whose minor device number is 1 as the paging device. In order for this particular version of the operating system to be able to perform paging, a disk formatted with page space must be in the specified device.

NOTES

- The "tune" command changes the values of the parameters in the file specified on the disk only, not in memory. Therefore, the changes have no effect until the user boots the operating system from the modified version.

ERROR MESSAGES

'r' option incompatible with command-line parameters.
The 'r' option, which tells "tune" to operate in read-only mode, conflicts with the specification of parameters on the command line. The command is aborted.

***Value out of range [num_1, num_2]
The value specified for a parameter is not within the range of acceptable values. The limits of the range are shown inside the square brackets.

***Value must be a multiple of <num>.
The value for the number of buffers in the system must be a multiple of 8. The value for the time zone must be a multiple of 60.

unmount

Unmount a previously mounted device from the file system.

SYNTAX

/etc/unmount <dev_name>

DESCRIPTION

The "unmount" command unmounts the specified device from the file system. Once the device is unmounted, the files in the directory on which it was mounted become accessible. Only the system manager may execute this command.

Arguments

<dev_name> The name of the device to unmount.

EXAMPLES

1. /etc/unmount /dev/fd0

This example unmounts the device in floppy drive 0.

ERROR MESSAGES

Error processing "<dev_name>": <reason>
The operating system returned an error when "unmount" tried to process "<dev_name>". This message is followed by an interpretation of the error returned by the operating system.

Error unmounting "<dev_name>": <reason>
The operating system returned an error when "unmount" tried to unmount the specified device. This message is followed by an interpretation of the error returned by the operating system.

Syntax: /etc/unmount <dev_name>
The "unmount" command expects exactly one argument. This message indicates that the argument count is wrong.

SEE ALSO

mount

update_all

Process a set of files, performing the specified operation on each file if it is newer than the file it is compared to.

SYNTAX

```
update_all [<make_file_name>] [+q]
update_all <make_file_name> [<arg_list>] [+q]
```

DESCRIPTION

The "update_all" command reads the specified "makefile", which must conform to a special format, and conditionally performs the command or commands in that file. By default, "update_all" sends informative messages to standard output telling the user what it is doing. The command is most often used to recompile programs whose sources have been updated.

Arguments

<make_file_name>	The name of the file to read for instructions. This file must be in a special format (see Format of the "makefile"). If no other arguments are present, the default is the file "makefile" in the working directory. If other arguments are present, the user must specify the name of the "makefile".
<arg_list>	A list of strings to substitute for any string designators that appear in the "makefile" (see Format of the "makefile). If this argument is used, the user must specify the name of the "makefile".

Format of the "makefile"

The "makefile" is composed of modules, each of which is terminated with a percent sign, '%', in column 1. A module itself is composed of up to two parts. The first part specifies the process that "update_all" is to perform. The format for this first part is as follows:

```
[<item-one>::[$]<item_two>;]<command_sequence>
```

where <item_one> and <item_two> are the names of files; "::" is the "is newer than" operator; the dollar sign, '$', changes the interpretation of the "is newer than" operator if <item_one> exists but <item_two> does not; and the semicolon, ';', separates the names of the files from the

command sequence.

The command sequence is composed of one or more UniFLEX commands. The "update_all" command replaces any sequence of more than one space character with a single space. Multiple commands are separated by additional semicolons. If the commands do not fit on one line, the user must begin and end the sequence with an exclamation point, '!', which serves to delimit the entire command sequence. If the first portion of the module uses more than one line, the second exclamation point marks the boundary between the first and second portions of the module. The command sequence is executed if <item_one> is newer than <item_two>.

The user may substitute an ampersand, '&', for any character or sequence of characters in <item_one>, <item_two>, or the command sequence. In such a case the "update_all" command substitutes for all ampersands the strings specified in the second portion of the module. If the second portion of the file is absent, no command sequence is performed. This portion consists of one or more lines, each of which contains a single string to substitute for the ampersands. The "update_all" command replaces each occurrence of an ampersand with the string on the first line of the second portion of the module and performs the command sequence if <item_one> is newer than <item_two>. It then replaces all ampersands with the string from the second line, continuing in this fashion until it reaches the end of the second portion of the module (marked by a percent sign in column 1).

The user may substitute a string designator (a pound sign, '#', followed by a digit from '1' through '9' inclusive) for any character or sequence of characters in <item_one>, <item_two>, or the command sequence. In such a case, the "update_all" command substitutes for each pound sign and its digit the corresponding element of <arg_list>. If the number represented by a digit is greater than the number of elements in <arg_list>, the sequence of the pound sign and digit remains intact.

If the file represented by <item_one> exists but the file represented by <item_two> does not, and if the "is newer than" operator is not followed by a dollar sign, "update_all" considers <item_one> newer than <item_two>. Under the same circumstances, if the "is newer than" operator is followed by a dollar sign, "update_all" does not consider <item_one> newer than <item_two>. In any case, if the file represented by <item_one> does not exist, or if neither the file represented by <item_one> nor <item_two> exists, <item_one> is not considered newer than <item_two>.

For instance, consider the following command:

```
update_all makefile s 1 y
```

and the accompanying "makefile":

```
&::&.b;asmb & +#1#2#3
file_1
file_2
  .
  .
  .
file_n
%
```

This "update_all" command makes the following translation of the "makefile":

If "file_1" is newer than "file_1.b", execute the command "asmb file_1 +sly".

If "file_2" is newer than "file_2.b", execute the command "asmb file_2 +sly".

It continues in this fashion until "file_n" is processed. The percent sign in column 1 marks the end of the module, and because it is the only module in the file, the "update_all" command terminates.

More than one set of commands can be processed with a single "makefile" if the user includes more than one module in the file.

Note that the use of the pound signs allows the same makefile to be used for another version of the "asmb" command. However, if the user does not specify three arguments on the command line, "update_all" cannot perform all the substitutions and the operating system cannot recognize the resulting form of the "asmb" command because it contains one or more string designators.

Options Available

q Do not send informative messages to standard output.

NOTES

- The "chd" command has no effect in a "makefile".
- In order to remove the special meaning from either of the characters '#' or '&', the user must precede it with a backslash character, '\'. Similarly, to remove this special connotation from a backslash, the user must use two backslashes in a row.

ERROR MESSAGES

*** Can't access Makefile "<file_name>" - aborted!
The operating system returned an error when "update_all" tried to open <file_name> for reading. Most probably, the file specification is incorrect, the file does not exist, or the user does not have read permission for the file.

*** Error: Command too complicated.
<command_sequence>
After substitution for the ampersands has taken place, the command sequence is too long (the limit is 1,024 characters).

*** Error: Pattern too complicated.
<command_sequence>
The pattern for the command sequence (before substitution for ampersands takes place) is too long (the limit is 1,024 characters).

Invalid option: '<char>'.
The option specified by '<char>' is not a valid option to the "update_all" command.

Makefile syntax error - aborted
The "update_all" command was unable to interpret the "makefile".

Syntax: update_all [<make_file_name>] [+q]
update_all <make_file_name> [<arg_list>] [+q]
The "update_all" command requires exactly one argument. This message indicates that the argument count is wrong.

SEE ALSO

touch

wait

Wait for a background task to complete before accepting any more input.

SYNTAX

```
wait [<task_ID>]
wait any
```

DESCRIPTION

The "wait" command, which is part of the shell program, tells the shell program not to accept any more commands until the specified background task is complete. The termination status of the task is reported when it is complete. If the user does not specify a task ID, the shell program waits for all active background tasks that are children of the shell program that issued the "wait" command to finish before accepting any new commands. The user may interrupt the "wait" command with a control-C.

Arguments

<task_ID>	The ID of the task to wait for. The shell program reports the ID when it sends a task to the background. The ID may also be obtained by executing either the "jobs" or the "status" command.
any	If the user specifies the argument "any", the shell program waits for any one background task that is one of its children to finish before accepting a new command.

EXAMPLES

1. wait 495
2. wait
3. wait any

The first example tells the shell program to accept no further commands until task 495 is complete.

The second example tells the shell program to accept no further commands from the user until all background tasks belonging to that shell program are complete.

The third example tells the shell program to accept no further commands from the user until one background task belonging to that shell is complete.

(continued)

ERROR MESSAGES

No tasks running in the background.
The shell program has no tasks running in the background.

Specified task not running in the background.
The task specified either is not a child of the current shell program or does not exist.

SEE ALSO

jobs
shell
status