Cromemco

Cromix-Plus User's

Reference Manual

November 1985

023-5013
Rev. C

CROMEMCO, Inc.
P.O. Box 7400
280 Bernardo Avenue
Mountain View, CA 94039

This manual was produced using a Cromemco System Three computer running under the Cromemco Cromix Operating System. The text was edited with the Cromemco Cromix Screen Editor. The edited text was proofread by the Cromemco SpellMaster Program and formatted by the Cromemco Word Processing System Formatter II. Camera-ready copy was printed on a Cromemco 3355B printer.

TABLE OF CONTENTS

LIST OF TABLES

Chapter 1

INTRODUCTION

This manual describes the Shell commands and utility programs supplied with the Cromix-Plus Operating System. For easy reference, commands are listed alphabetically. If you are unsure about the name of a particular command, consult chapter 2, a summary of available commands. Each command is discussed in detail in chapter 3.

To use this manual effectively, a basic understanding of the Cromix Operating System is required. Refer to the Introduction to Cromix-Plus, in particular, chapters 4 and 6.

Chapter 2

SUMMARY OF COMMANDS AND UTILITIES

The Cromix utility programs perform many necessary functions. They are similar to and used in conjunction with the Cromix Shell commands.

In contrast to the Shell commands, utility programs are not intrinsic to the Cromix-Plus Operating System but must be called from disk when needed.

Write and append access for all utilities is limited to privileged users.

The following list summarizes the commands and programs described in detail in this manual.

access	changes access privileges of a file
bak	deletes files with **.bak** extensions
blink	links Z80 programs
boot	loads an operating system into memory and begins execution
ccall	calls another Cromix system via a modem
cdoscopy	copies file to and from CDOS disks
cdosfix	strips CONTROL-Z's from the end of CDOS files
CE	enhanced version of the Screen editor, written in "C" language
check	runs the Dcheck and Icheck utilities
chowner	changes the owner or group owner of a file
clist	displays a file with page headings and line numbers
clock	executes a command line and reports time needed to execute that command line
cmpasc	compares two ASCII files
compare	compares two ASCII files
config	configures memory for a **.bin** program

convert	converts file format (ASCII to EBCDIC, EBCDIC to ASCII, and so on)
convobj	converts **.obj** files to .o68 format
copy	copies a file
cptree	makes a copy of a directory and its descendants
create	creates a file
crogen	generates an operating system
daemon	prints files in the **usr/spool** directory
day	executes a command on the specified day
dcheck	verifies the integrity of a file system
ddump	converts and copies a file from one device to another
delete	deletes a file
deltree	deletes a directory and all descendant files
diskinfo	prints hard disk information
directory	changes directories or displays the pathname of the current directory
ecc	manipulates the error-correcting memory controllers
echo	echoes arguments to the terminal
exit	exits from a Shell
find	locates files
fixsb	restores Superblock
flush	flushes system buffers
free	displays the amount of unused space on a device
ftar	creates and retrieves file archives
goto	transfer control within a command file
group	allows users to change their groups
help	displays the on-line manual
icheck	checks the integrity of a file system

idump	displays the contents of an inode
if	conditionally executes another command
initflop	initializes a floppy diskette
inithard	initializes an STDC or SMD hard disk
inittape	initializes a floppy tape
input	reads a line from STDIN and writes it to STDOUT
iopload	loads a file into an IOP
kill	sends a kill signal to a process
link68	links 68000 programs; used with Crogen utility
logerr	accumulates errors detected and reported by the Ecc utility
l	lists directory or file information
ls	lists directory or file information
mail	sends or displays mail
makdev	creates a device file
makdir	creates a directory file
make	constructs executable programs from separate modules
makfs	sets up the structure for a file system on disk
maklink	creates another name for an existing file
match	finds all occurrences of a string within a file
mode	displays or alters the device modes
mount	enables access to a file system
move	moves a file from one directory to another
msg	sends messages between users
ncheck	displays file information
newuser	displays information for new users
octload	loads a file into Octart
passwd	changes a user password, adds, or deletes a user

passwd	changes a user password, adds, or deletes a user
patch	patches a file
path	displays the pathname of executable files
priority	changes the priority of a process
priv	changes user status to that of a privileged user
prompt	changes the Shell prompt until the user next logs in
pstat	displays the status of a process
query	summarizes Shell commands and utility programs
ramdisk	creates, deletes, verifies, and checksums RAM disks
rcopy	copies a file or block device
rename	changes the name of a directory or a file
repeat	repeats a command
rewind	restores arguments within a command file to their original positions
rfile	allows binary files to be received over phone lines (used with Ccall)
root	displays the name of the device containing the root directory
sc	calls the Screen editor with the **-n** option, to create files compatible with the UNIX Operating System
scan	scans a directory tree
screen	calls the Screen editor
setpri	changes the priority of a process
sfile	allows binary files to be sent over phone lines (used with Ccall)
shell	creates a shell process
shift	shifts arguments in a command file
shutdown	shuts down the operating system
sim	allows CDOS programs to run under the Cromix Operating System
sleep	puts a process to sleep for a specified number of seconds

sort	sorts or merges files
spool	queues files and sends them to a printer
startup	contains commands executed every time the system is started up
stdload	loads a program into an STDC
sync	provides a one-time flush of system buffers
sysdef	generates a configuration file for the Crogen utility
tar	creates and retrieves file archives
tee	pipes output to a file as well as to STDOUT
term	displays or changes the terminal name
termcaps	defines terminal capabilities
testinp	tests for equality between the contents of a file and one or more strings
time	displays or alters the time and date
touch	changes the modification times of files to the current time (used with Make)
type	displays an ASCII file
uboot	standalone boot program for the UNIX Operating System
unmount	disconnects a mounted file system from the current file system
update	updates a Cromix system disk with a newer system disk
usage	displays directory-size information
vdt	controls special terminal functions
version	displays the version number of the operating system or a utility program
wait	waits until all detached processes have finished
wboot	writes the boot program to the boot area of a disk
who	lists the users presently logged in
>	redirects the standard output to a file
>>	appends the standard output

<table>
<tr><td><</td><td>redirects the standard input from a file</td></tr>
<tr><td>>*</td><td>redirects the standard output and standard error to a file</td></tr>
<tr><td>><</td><td>sequentially pipes the standard output only</td></tr>
<tr><td>><*</td><td>sequentially pipes the standard output and standard error</td></tr>
<tr><td>>>*</td><td>appends the standard output and standard error to a file</td></tr>
<tr><td>|</td><td>pipes the standard output only</td></tr>
<tr><td>|*</td><td>pipes both the standard output and standard error</td></tr>
</table>

Chapter 3

SHELL COMMANDS AND UTILITY PROGRAMS

An on-line version of this chapter is contained in the **/usr/help** directory, where it is accessible to all users via the Help command. To display information about a particular command, give the Help command, as in the following example:

```
% help match
```

This command displays detailed information about the Match command. You can display information about any command in the same way.

utility:	**ACCESS**	
purpose:	This program changes the access privileges associated with a file.	
user access:	all users	files owned by the user
	privileged user	any file
summary:	access [+rewa].[+rewa].[+rewa] file-list	
arguments:	access-privilege-specifier string	
	one or more pathnames	
options:	none	

Description

The Access utility allows a user to change file-access privileges.

The access-privilege-specifier string (first argument) contains three clusters of access flags separated by periods. The first cluster indicates owner permitted access, the second indicates group access, and the third indicates public access. Each cluster is composed of zero or more of the following flags, given in any order:

+	add the specified privileges
r	read access
e	execute access
w	write access
a	append access

Access privileges under the Cromix-Plus system are discussed in detail in the following paragraphs.

File Protection

The Cromemco Cromix-Plus Operating System offers protection for files on many levels.

All files may be opened for exclusive or nonexclusive access. A file opened for exclusive access may not be opened by another process until it is closed by the process that originally opened it. If a file is opened for nonexclusive access, it may be simultaneously opened and accessed by more than one process.

File access privileges are divided into three population segments and four types of file accesses.

The first population segment is the **owner** of the file. This is normally the creator of the file. The second population segment is the **group** to which the owner belongs. A user's group number can be verified in the **/etc/passwd** file. The third population segment is the general **public**. This segment includes all system users.

There are four types of file access for each population segment. The first is **read** access. Read access allows the designated user to read the file. If a user has read access for a directory, the user may list the contents of the directory.

The second is **execute** access. Execute access allows the user to execute the file. If the user has execute access for a directory, the user may use the directory in a pathname.

The two remaining types of access are **write** access and **append** access. Write access allows the user to write to the file, meaning the user may write over or change data in the file.

Append access allows data to be added to the end of the file. Data may then be written to the file at a point **past** the end of file, and the end-of-file indicator is moved to the end of the newly added data. If append access is the only access specified, data written to the file may not be read.

Append access does not imply write access, but write access implies append access.

One type of access privilege for a population segment does not imply any other access privilege for that population segment. The categories of access privileges are combined to provide meaningful data handling. For example, a user with write access to a file normally has read access.

One important point to consider when determining file access privileges is that the file's owner is a member of a group and a member of the public. Implicitly, the user has all access privileges granted to the public and to the group. Any member of the group enjoys all access privileges granted to the public.

All files are created with default access privileges as follows: read, execute, write, and append access privileges for the owner; read and execute access privileges for the group and public. The default owner is the user name of the user who executed the command creating the file. The system gathers its information on user name from the **/etc/passwd** file. (This default may be changed by generating a new operating system with the **Crogen** utility.)

When files are created by programs that a user is running (e.g., Screen, Debug, WriteMaster, etc.), those files take on default attributes as described. These same programs can also alter existing files. In this case, the owner name is unchanged, but file access attributes may change. For instance, the Screen editor does not change file attributes after an editing session, but using the **w** command within the Debug program changes all file attributes to the default values. Since this effect may vary from command to command within a single program, and from program to program, users should be aware that file attributes are not immutable.

Access privileges take on a different meaning when applied to a directory. Read access for a directory means the user can use the **Ls** utility to see the contents of the directory. Execute access means the directory may be used in a pathname or that the user has access **through** the directory. Write access means the user can alter the directory.

The **Ls** utility program with the **-l** option may be used to check the access privileges associated with a given file. For example, the following command will list the access privileges of file **xyz:**

```
% ls -l xyz
312 1 rewa re-- re-- joe Mar-09 18:25 xyz
```

Reading this display from left to right, two items precede the access information: the numbers of bytes in the file (312) and the number of links to the file (1). The access information is displayed as three clusters of four characters. The four characters are **r** (read), **e** (execute), **w** (write) and **a** (append). The presence of one of these characters indicates that the specified population segment is endowed with the specified access privilege. The population segments are, from left to right, **owner, group**, and **public**. Thus, in the above display, the owner has all four access privileges, while the group and public have only read and execute access privileges.

The last four items in the preceding display are the name of the owner of the file (joe), the date and time the file was created and, finally, the name of the file.

Users working within the Cromix file system must explicitly check the access attributes of files and directories they work in, use, or create for other users. Users must be aware of accessory files that may be required by programs they are running--help files, libraries, and so forth. Access and ownership of the accessory file--and access and ownership of parent directories all the way to the root--must be compatible with the operation of the program being executed.

For all errors implying access limitations, always check access privileges and the ownership of the directories, files, and ancestor directories involved.

Notes

The Access utility allows the user to change file access privileges in several different ways. The first of these is to re-enter each access privilege for each population segment, making the desired changes. For example:

```
% access rewa.rw.a xyz
% ls -l xyz
312     1 rewa r-w- ---a joe          Mar-09 18:25   xyz
```

The second method for specifying access privileges involves the use of the plus sign (+) in one or more of the access population clusters. When used in this manner, the plus sign means that the attributes for the specified population segment remain the same. The plus sign may also be followed by access privileges to be added for the given population segment.

```
% ls -l abc
517      1 rewa re--  re--    joe          Mar-09 18:26   abc
% access +.+.+a  abc
% ls -l abc
517      1 rewa.re.re-a       joe          Mar-09 18:26   abc
```

utility:	**BAK**
purpose:	Deletes **.bak** files from the current directory
user access:	~~all users~~ privileged user
summary:	bak
arguments:	none
options:	none

Description

The Bak utility deletes files in the current directory with the filename extension **.bak.**

utility:	**BLINK**
purpose:	This program links relocatable Z80 files.
user access:	all users
summary:	blink [-dinpqrxz] [-b outname] filename [-s libname] . . .
arguments:	one or more filenames optional **-s** followed by a library name

options:

-b	output filename
-d	data section address
-i	IOP starting address
-n	no map
-p	program address
-q	do not display map
-r	relocatable binary
-s	search library
-x	bit-mapped
-z	size (use with -r)

Description

The Blink utility is a two-pass virtual linker of Z80 programs. One or more input files can be specified. An executable binary file is generated. Blink can be used to generate relocatable binary files that can share a bank of memory with other programs.

Options

The **-b** option may be used to specify the output filename. If used, the **-b** option must be followed by a space and the name of the binary file to be created. If this option is not used, the output file adopts the name of the first relocatable file specified on the command line. The output file has the filename extension **.bin.** This option may be used to force the output file to have a filename extension of **.com.** These are programs compatible with the CDOS Operating System only if they were written using CDOS system calls. The format for linking these files is:

```
% blink -b filename.com modulenames
```

The **-d** option is followed by a space and the hex value of the data section starting address.

The **-i** option is followed by a space and the hex value of the starting address for an IOP program. It allows relocation of the program above the memory area occupied by the IOP Monitor. The IOP Monitor occupies memory between addresses 0000 and 0800 hex in ROM, and between 7F00 and 7FFF hex in RAM. This option creates an automatic header for the program to be run in the IOP using the Iopload utility program.

The **-n** option prevents creation of a link map. Otherwise, the link map is created and written to a file with the filename extension **.map**.

The **-p** option must be followed by a space and the hex value of the program starting address. If no starting address is specified, the program starts at 100 hex. A relocatable binary program is placed wherever there is space in a memory bank.

The **-q** option inhibits display of the link map on the terminal. Otherwise, the link map is displayed on the terminal.

The **-r** option causes generation of the output file in relocatable binary format. Programs in this format can be executed with another process in a single bank of memory. The **-r** option is used with the **-z** option discussed below.

The **-s** option precedes the filename of the library to be searched. The option applies only to the file immediately following it, and must be specified for each file to which it applies. Blink searches the **.rel** file for necessary functions. If no library is specified using the **-s** option, and there is no library in the current directory, the program looks into **/usr/lib**, which is the default system library directory.

The **-x** option makes the output file a bit-mapped self-relocating file. This option generates a self-relocating file which, when loaded into a user bank, loads in highest available memory and sets high memory to the byte just below itself. This option is used in linking the Cromemco Debug program.

The **-z** option allocates a specific size for the program segment. This switch is used only with the **-r** option, and only when free space (more than Blink normally allocates) is desired in the program area.

Notes

Blink manages memory so as to link programs up to the total amount of memory available. The memory area used by the linker during execution does not impose a restriction on the size of the program being linked. Thus, Cromix-Plus programs up to 64K, minus 1K of memory occupied by the Cromix-Plus Operating System in each user bank, can be linked by Blink.

Relocatable binary programs are treated as normal Z80 programs under the Cromix-Plus system.

CDOS programs running under the Cromix-Plus Operating System are limited to approximately 4K less memory than the 63K available to Cromix-Plus programs. This is because Sim, the CDOS simulator, must also be loaded.

COBOL programs using segmentation cannot be linked with Blink.

utility:	**BOOT**
purpose:	This utility loads an operating system into memory.
user access:	privileged user
summary:	boot [filename]
arguments:	filename (optional)
options:	none

Description

The Boot utility loads an operating system into memory.

If no argument is given, the file **/cromix.sys** is loaded, and execution begins. In this manner, the Boot utility can be used to warm boot the Cromix-Plus Operating System.

Example:

```
# boot

System shutdown in progress
System shutdown complete
```

The raw console then displays:

```
Floppy = 1, Hard disk = 2, STDC = 6
Enter major root device number: 6
Enter minor root device number: 0

System initialization complete
```

Here, the Boot utility is executed and the Cromix-Plus Operating System reloaded. The root device is specified as STDC disk (6) number 0.

If Boot is followed by a filename, the file is assumed to have a **.sys** extension.

Notes

Because this program loads an operating system, it interrupts any active processes. Be sure that no one else is executing a program and that there are no detached processes running on the system before executing the Boot utility. Otherwise, data may be lost.

One quick method to determine if there are users on the system is to execute the Ps (Process Status) command:

```
# ps -a

PID State Command
  1   S   -
112   R   Shell
105   R   screen letter
 18   S   login 1 19200 tty6
 94   S   shell
 16   S   shell
 15   S   shell
 14   S   shell
 89   S   login 1 9600 tty1
```

Here the command is executed with the **-a** (for all) option. The display shows one user running the Screen Editor program to edit a file named **letter.** If the Boot program were executed at this point, the user would lose all editing changes made during this session.

As long as all lines of the Ps display show a command of **shell** or **login,** no processes are running, and it is safe to load an operating system.

The Boot utility may be executed only by a privileged user.

utility:	**CCALL**
purpose:	This program calls another Cromix (or non-Cromix) system using a modem.
user access:	all users
summary:	ccall [-q] [-d dev-name] [-b baud]
arguments:	none
options:	-q quiet (default is verbose) -d qtty device-name (default is /dev/modem) -b baud rate (default is current rate)

ccall is always using the current directory it was called from

[illegible]

Description

The Ccall utility allows one Cromix-Plus system to act as a terminal to another Cromix-Plus system (or to another completely different system). For best performance, connect the modem to any serial port on the IOP/Quadart. With some restrictions, Ccall can also run on a modem connected to the OCTART or TU-ART. When connected to the IOP/Quadart, a 12-wire cable designed for this purpose should be used. The modems on each system must be compatible.

Using the Maklink utility, make a link from the appropriate qtty device in the **/dev** directory to **/dev/modem.**

Ccall provides no automatic calling or modem initialization. Once the baud rate on the **/dev/modem** device is set (if necessary), all characters typed on the terminal are sent to the modem, and any character received from the modem is echoed to the terminal. The following Ccall commands are also available (Ccall interprets lines beginning with a tilde (~) as escape sequences):

~.	Terminate Ccall
~~...	Send the line: "~..." to the remote system.
~< filename	Redirect the contents of a file to the remote system, as though the contents had been typed from the terminal.
~> filename	Redirect output from the remote system to the specified file. Output redirected in this way is written to an ordinary file and to the standard output (for display).

To write output to the specified file only, type a colon after the redirect symbol:

```
~>: filename
```

To append output to a file, use the redirect-and-append symbol (>>) with either form of the command.

~>
: The command ~> ends redirection.

~sh
: Invoke an interactive shell on the local system. To exit the Shell type EXit or CNTRL-Z.

~sh command
: Execute the command on the local system (via shell -c).

~put [-f] file-list
: Copy the specified files from the local system to the remote system. If ~put finds a file with the same name at the remote system, that file is not copied unless the -f option is used to overwrite the existing file.

~put uses the Sfile and Rfile utilities to perform error-free block transfers.

~take [-f] file-list
: Copy the specified files from the remote system to the current directory on the local system. The -f option overwrites files with the same name.

~take uses the Sfile and Rfile utilities to perform error-free block transfers.

~h
: Print a summary of the Ccall commands.

Options

The **-b** option sets the baud rate of the transmitting device. Possible settings are 110, 150, 300, 1200, 2400, 4800, 9600, and 19200 baud. Without this option, the baud rate set for the device **/dev/modem** is used; if a baud rate has not been defined (or the device was discarded), 1200 baud is used.

The **-d** option specifies a transmitting device (qtty) other than **/dev/modem**.

The **-q** (for quiet) option reduces the number of Ccall messages.

Notes

The remote system should have a compatible modem that can automatically answer the phone call. On a remote Cromix system, the mtty terminal connected to the modem must be enabled in the **/etc/ttys** file. If the mtty baud rate is set to automatic (a), the caller can establish the baud rate by repeatedly pressing RETURN until the remote system recognizes the baud rate and responds with a login prompt.

Ccall sets many of the modes for both the local and remote systems. The optional mode settings are described below.

Remote system, mtty, SIGHUPall

The mtty driver enables SIGHUPall on all mtty devices (and reenables it each time a device is DIScarded). When the communication link is broken, all processes on the remote system that use mtty as the controlling terminal are aborted (Shell processes are logged out) by the SIGHANGUP signal (unless this signal is trapped or otherwise ignored). If you disable SIGHUPall after logging in, a broken communication has no effect on the remote precesses, and you can reestablish the phone link and proceed from where you left off. However, disabling SIGHUPall is a serious security risk, as another user can call in and inherit the previous user's shell, password, and so on.

Remote system, mtty, DIScard

The mtty driver enables DIScard on all mtty devices. Thus, when an mtty device is finally closed (i.e., when the user EXits the Shell or when his Shell is killed by the SIGHUP signal), the device is reinitialized.

Remote system, mtty, HUPenable

The mtty driver disables HUPenable on all mtty devices. Thus, the driver does not try to hang up the phone when the mtty device is finally closed. Note that the modem may be configured to hang up if the data carrier is lost. If you enable HUPenable, exiting from the Shell will hang up the phone.

Local system, modem

Ccall does not change the DIScard and HUPenable modes of the local modem (SIGHUPall is ineffective). If you enable HUPenable before using Ccall, the phone will hang up when Ccall exits (you can hang up the phone at any time with the command: "Mode modem hup"). If DIScard is not enabled, the next caller can use Ccall at the previous baud rate without having to specify the **-b** option; if DIScard is enabled, the baud rate will default to 1200 baud unless the **-b** option is used.

utility: **CDOSCOPY**
purpose: This utility copies files to and from CDOS disks.

user access: all users

summary: cdoscopy [-belvw] devname file-list

arguments: Cromix device name

name(s) of the file(s) to be copied

options:

-b	binary file
-e	erase file
-l	list CDOS directory
-v	verbose
-w	write CDOS file

Description

The Cdoscopy utility copies files from a Cromemco Disk Operating System (CDOS) format disk to a Cromemco Cromix-Plus Operating System format disk, and vice versa. For example:

```
% cdoscopy fdb letter
% cdoscopy -w sfda notes
```

The first of these command lines copies a CDOS file named **letter** (located on a large floppy disk in drive B) into the user's current directory. The second command line copies the Cromix file named **notes** from the user's current directory to a small floppy disk in drive A. In the first case, the file is converted from a CDOS format to a Cromix format. A Cromix-format-to-CDOS-format conversion takes place in the second example.

The Cromix-Plus Operating System **cannot** read CDOS files. Programs to be executed and data to be read under the Cromix-Plus Operating System **must** be transferred from CDOS formatted disks to Cromix formatted disks before execution begins.

Where a file pathname is specified, CDOS considers the lowest level filename. This is the portion of the pathname to the right of the rightmost slash. For instance, the following command line puts the file named **memo** onto the CDOS format disk in drive B.

```
% cdoscopy -w fdb /usr/mary/memo
```

Options

The **-b** option copies binary files. When this option is used, the 1Ah (end-of-file mark) is not stripped from the end of the file.

The **-e** option erases the specified file(s) from the CDOS disk.

The **-l** option displays the contents of the CDOS directory.

The **-v** option displays files while they are copied to and from CDOS disks.

The **-w** option causes the file to be written to the CDOS disk.

Notes

When an ambiguous CDOS file reference is used, it must be enclosed in quotation marks.

CDOS filenames must also be legal Cromix filenames. If not, use the CDOS Operating System to rename the files, then use the Cdoscopy utility.

Examples:

```
% cdoscopy -v fda "*.txt"
% cdoscopy -l fdb
```

These examples assume that the disks in drive A (fda) and B (fdb) are CDOS disks. The first example copies all CDOS files on drive A having the filename extension **.txt** into the current directory. The ambiguous CDOS file reference was placed inside quotation marks.

The second example displays the directory of the CDOS disk in drive B (Cromix file designation **fdb**).

Refer to the Cromix-Plus System Administrators Guide for a list of device names.

utility:	**CDOSFIX**
purpose:	This program strips the ^Z's from the end of CDOS files.
user access:	all users
summary:	cdosfix filename [filename ...]
arguments:	one or more filenames
options:	none

Description

The Cdosfix utility strips the ^Z's from the end of files created using the CDOS Operating System.

utility: **CE**
purpose: This program is used to edit files.

user access: all users

summary: ce [-rucnteafms] filename [filename]

arguments: filename(s) to be edited

options:

-r	read-only mode
-u	update mode
-c	lines terminated by CR-LF pair
-n	lines terminated by LF only
-t	blanks in output replaced by TABs
-e	blanks in output not replaced by TABs
-a	display cursor address
-f	display name of file being edited
-m	multi-file session
-s	single file session

Description

The CE utility is an enhanced version of the Screen editor written in the "C" language. Unlike Screen, CE incorporates most text changes without redisplaying the entire screen. Also, text entered after the 80th character is not wrapped to the next line, but is "hidden" outside the edge of the screen. The horizontal scroll command (Yank) is provided to display hidden text.

CE uses the appropriate escape sequences for each terminal as defined in the **/etc/termcaps** file. The termcap file entry for the Cromemco 3101 describes the simplest terminal supported by CE.

```
# Cromemco 3101 Terminal
#
C1|3101|Cromemco 3101 Terminal:\
          :co#80:li#24:\
          :up=\EA:do=\EB:nd=\EC:nb=\ED:\
          :cm=2\EF%+ %+ :\
          :cl=2\EE:cd=2\EJ:ce=2\EK:\
          :dc=\EP:\
          :kl=^A:kr=^D:ku=^W:kd=^X:
          :am:
```

The following terminal properties must be defined:

1. Number of rows and columns ("li" and "co")
2. Cursor up, down, left, right ("up", "do","nb" and "nd")
3. Cursor addressing ("cm")
4. Clear functions ("ce" - clear to the end of line,
   "cd" - clear to the end of screen,
   "cl" - clear screen)
5. Delete character at cursor ("dc")

6. Arrow keys definition ("kl", "kr", "ku", "kd")

On terminals without arrow keys, control characters can serve the same function. On the 3101, use CONTROL-A for cursor left ("kl"), CONTROL-D for cursor right ("kr"), CONTROL-W for cursor up ("ku"), and CONTROL-X for cursor down ("kd"). The "am" entry means that text scrolls up when the cursor reaches the bottom right corner of the screen.

To minimize screen refreshing, your terminal also needs line insert/delete functions; to use horizontal scrolling, you need character insert/delete functions in page mode. The Cromemco C5 terminal, with it's special insert left/right column feature, is particularly efficient for horizontal scrolling (refer to the section on TERMCAPS).

Options

The **-r** option (read-only mode) does not accept commands to change the contents of the file; without this option, all CE commands are accepted (the **-u** option is assumed).

The **-c** option terminates all lines with a CR,LF sequence; without this option, lines are terminated with LF only (the **-n** option is assumed).

The **-e** option does not replace blanks in the output with TABs; without this option, leading blanks are compressed with the appropriate number of TABs (the **-t** option is assumed). On input, CE replaces all TAB characters with blanks.

The **-f** option displays the name of the file being edited; without this option, the row and column address of the cursor is displayed (the **-a** option is assumed).

The **-m** option accepts any number of filename arguments for sequential editing; without this option, CE takes the first filename as the input file and the second filename (if any) as the output file (the **-s** option is assumed). For example, the command

```
CE -m *.c
```

edits all files in the current directory with a **.c** filename extension (provided that there are not too many files).

Commands

Most of the new CE commands are selected by the upper case letter of the analogous Screen command. For example, a lower case "i" works the same as in Screen, but an upper case "I" inserts one blank line without pushing the rest of the line after the insertion point.

The following commands are either unique to CE or variants of those used by Screen:

a(gain) Repeats the last Find or Substitute command.

b(racket) Moves the cursor to the next matching bracket if the cursor is positioned on a "(", "[", or "{". If there is no matching bracket, the terminal beeps and the cursor does not move. If the character at the cursor is not a bracket, the **b** command is ignored.

d(elete) While in delete mode, pressing "w" deletes all blank and non-blank characters from the cursor position to the next word.

e(xit) CE prompts for confirmation if you press "Q" to exit from a file that has been modified. Pressing "t" in exit mode quits the current file and aborts the entire muti-file editing session.

f(ind) Pressing "e" before giving the search pattern of a Find command makes the search case sensitive. Pressing "r" and a marker number before the search pattern terminates the search at the given marker (the marker must exist before giving the command).

i(nsert) If a line is longer than 80 characters, pressing "i" for insert mode shifts the whole line left 10 characters, giving you the chance to terminate it. Pressing RETURN restores the line to its original "window dependent" position. An upper case "I" inserts one blank line without pushing the rest of the line after the insertion point.

K(ase) Pressing "K" toggles the case of all alphabetic characters from the cursor position to the next non-letter, non-digit character.

n(ext) Combines the current line with the next line. Pressing "N" joins the lines as they are; pressing "n" adds one space at the juncture.

Q After pressing "Q", you can change (toggle) any of the modes by pressing "m" (read-only/update), "n" (CR/CRLF), "d" (display cursor address/filename), or "i" (auto/manual indentation). When indentation is automatic (default), CE starts each new line in the same column as the previous line.

s(ubstitute) Same new features added as for the Find command.

w(rite) Pressing "a" before giving the marker numbers appends the appropriate text to the end of the named file.

y(ank) Changes the indentation of a portion of the file defined by two markers. After setting the markers, place the cursor at the right column position and press "y", followed by two marker numbers. The indentation is then changed by pressing either the left or right arrow keys (optionally preceded by a repeat count).

Y(ank) Pressing "Y" allows you to change the position of the editing window (horizontal scroll). Columns 1 - 80 are normally displayed.

">" In Edit mode, pressing ">" moves the cursor forward to the next word (the next string of non-blank characters).

"<" In Edit mode, pressing "<" moves the cursor backward to the previous word (the last string of non-blank characters).

"/" In Edit mode, pressing "/" moves the cursor to the end of the line. If the right end of the line is off the screen, the cursor stops at the screen edge.

"?" In Edit mode, pressing "?" moves the cursor to the start of the line. If the left end of the line is off the screen, the cursor stops at the screen edge.

"`" To clear existing markers, press "`" followed by either a marker number (to clear that marker) or an "*" (to clear all markers).

Notes

Pressing the left-arrow key moves the cursor one indentation level to the left; pressing the delete key moves the cursor one position back.

control V in insert mode allows control character

utility:	**CHECK**
purpose:	This program runs the Dcheck and Icheck utilities.
user access:	privileged user
summary:	check [-s] [devname]
arguments:	optional device name
options:	-s

Description

The Check command runs the programs Dcheck and Icheck on a file system. Check should be run after rebooting the system or any time the integrity of the file system is in doubt. The Startup command file program executed after every boot-up indicates when the Check program needs to be run. See the Startup command file description in this manual for more information on Check.

Options

The **-s** option is the salvage option used with Dcheck and Icheck to repair most file system problems. See the description of the Dcheck and Icheck utilities in this manual for more information. The system is rebooted after running Check with the **-s** (for salvage) option.

Notes

In general, it is safer to run Check with the **-s** option only on an unmounted device. When run on a mounted device, especially the root device, the file-structure problem you are attempting to correct may be immediately recreated.

utility: **CHOWNER**
purpose: This program changes the owner or group of a file.

user access: privileged user

summary: chowner [-gv] ownername file-list

arguments: name or number of the user to whom ownership is to be transferred

or

name or number of the group to which ownership is to be transferred

and

one or more filenames

options:
-g change group
-v verbose

Description

The Chowner utility changes the owner or group associated with any type of file. If the file **abc** is in the current directory and is owned by **mark**, the LS utility might display it as:

```
# ls -l abc
27     1 rewa re-- re-- mark          Mar-11 19:59   abc
```

Using the Chowner utility, ownership can be transferred to **cindy:**

```
# chowner cindy abc
# ls -l abc
27     1 rewa re-- re-- cindy         Mar-11 19:59   abc
```

Options

The **-g** option allows the Chowner utility to change the group associated with the file. This option is used in the manner previously described, substituting the group name for the owner name.

The **-v** option displays the name of each file as its ownership is changed.

Notes

When the ownership of a file is changed, the group with which the file is associated changes to that of the new owner.

utility:	**CLIST**
purpose:	This program displays files with page headings and line numbers
user access:	all users
summary:	clist [file-list]
arguments:	one or more file pathnames
options:	none

Description

The Clist program displays the files specified by its argument(s). When displaying a file, Clist numbers each line and adds a heading showing the filename and the time the file was last modified.

When called without an argument, Clist waits for input from the standard input.

utility:	**CLOCK**
purpose:	This program executes a command line and reports the time used to execute that command line.
user access:	all users
summary:	clock [-h] command-line
arguments:	command line to be executed
options:	-h report time in the form hh:m:ss

Description

The Clock utility executes the given command line and reports the time, in seconds, used to execute that command line. The times reported are CPU time and real time.

Options

The **-h** option displays time used in the form "hh:mm:ss" instead of in seconds alone (the default).

utility:	**CMPASC**
purpose:	This program compares two ASCII (text) files.
user access:	all users
summary:	cmpasc [-bm #] [-lrt] file1 file2
arguments:	2 filenames
options:	-b # memory size, in bytes -l print line numbers -m # match this many lines -r ignore RETURN preceding LINEFEED -t expand TAB characters

Description

The Cmpasc utility compares two ASCII (text) files and displays differences.

Cmpasc displays lines from the first file (file1) with corresponding lines from the second file (file2).

If there are too many unmatched lines in a file (more than can be stored in the allotted memory), the **-b** option can be used to increase allotted memory.

Options

The **-b** option defines the amount of memory set aside for storing unmatched lines. (The default value is 32,768 bytes per file.)

The **-l** option adds line numbers to the display.

The **-m** option, followed by a number, defines how many successive lines from one file must match the corresponding lines in the second file to be considered a match. (The default value is 3.)

The **-r** option ignores a RETURN character preceding a LINEFEED. Thus, a line ended by the LINEFEED character alone is equal to a line ended by a RETURN-LINEFEED pair.

The **-t** option expands TAB characters before comparing lines.

Example:

```
% cmpasc -l fileone filetwo
-----> fileone
26 This file is sample file one.
-----> filetwo
26 This file is sample file two.
```

Notes

The Cmpasc utility compares characters with the parity bit reset. As a result, the bytes 0x8D and 0x0D are considered equal.

utility:	**COMPARE**
purpose:	This program compares two files.
user access:	all users
summary:	compare [-t] file1 file2
arguments:	2 filenames
options:	-t terse

Description

The Compare program compares two files and reports differences in length and content.

Compare lists differences between the files on a byte-by-byte basis. It displays an address in hexadecimal, then the byte in the first file at that address, followed by the corresponding byte in the second file. Compare does not adjust for offset, should one file lack one or more bytes in the middle (e.g., if part of a file was deleted).

Options

The **-t** option suppresses the list of differences. When this option is used, only a message is displayed to indicate whether the files are the same or different.

utility:	**CONFIG**
purpose:	configures memory for any Cromix-Plus **.bin** program
user access:	all users
summary:	config filename [memory-size[kd]]
arguments:	Cromix-Plus **.bin** filename memory size, followed by:

k	memory size in kilobytes (the default)
d	memory size in decimal number of bytes

Description

The Config utility allows users to select the memory size that the Cromix-Plus Operating System allocates for any **.bin** program. The filename specified can be any Cromix-Plus **.bin** file (with or without the **.bin** filename extension). Memory size is assumed to be in kilobytes, unless followed by **d** to denote a decimal number of bytes. Although a valid memory size can range from 8K to 16000K, it will be rounded up to the nearest 4K.

If a filename is given, but no memory size, Config will report the number of kilobytes currently allocated to that file.

utility: **CONVERT**
purpose: Converts ASCII to EBCDIC and other transformations

user access: all users

summary:

```
convert [-faslutbprge] [-i isize] [-o osize] [[inputfile]
     outputfile]
```

arguments: optional input filename
optional output filename

options:

-f	input lines are fixed length
-a	convert to ASCII (from EBCDIC)
-s	strip trailing blanks
-l	convert to lower case
-u	convert to upper case
-t	expand TAB characters
-b	compress spaces in TAB characters
-p	pad output line with blanks
-r	insert CR in front of NL
-g	output lines are fixed length
-e	convert to EBCDIC (from ASCII)
-i #	input line size
-o #	output line size

Description

The Convert utility, reads lines from an input file, transforms those lines as specified, and writes the converted lines to an output file. The available options determine how the input lines are transformed; the arguments, if any, define the input and output file names.

With no arguments, the Convert utility reads lines from standard input and writes the converted lines to standard output. If one argument is given, it is taken as the input filename (transformed lines are again written to standard output); if two arguments are given, they are taken as input and output filenames, respectively. In most cases a device file can be substituted for the input and/or output files.

As usual, a dash "-" substituted for the input or output filenames represents the standard input or standard output, respectively.

The Convert utility has no provisions for such functions as tape positioning. Use the Mode utility to perform such functions before calling the Convert utility. Since the general Rcopy utility can be piped into the Convert utility (or the output of the Convert utility can be piped into Rcopy), the Convert utility needs no specialized knowledge of the I/O devices. It is best understood as a filter that transforms standard input to standard output.

Options

The **-f** option indicates that all input lines have a fixed length (determined by the **-i** option); without this option, lines are terminated by a \n character.

The **-a** option converts EBCDIC input to ASCII output. EBCDIC characters with no convenient ASCII counterpart are changed to spaces. Since end-of-line characters (\n) are not recognized, the **-f** option must be used with the **-a** option.

The **-s** option discards the trailing blanks in all input lines.

The **-l** option converts upper case characters to lower case characters.

The **-u** option converts lower case characters to upper case characters.

The **-t** option replaces each TAB character with enough spaces to reach the next TAB position. The TAB positions are 1, 9, 17, and so on.

The **-b** option compresses multiple spaces into TAB characters, based on TAB positions 1, 9, 17, and so on.

The **-p** option pads output lines with blanks to match the line size set by the **-o** option.

The **-r** option terminates each output line with \r\n rather than with \n alone. The **-r** option has no effect if the **-g** option is set.

The **-g** option indicates that all output lines are of fixed length (determined by the **-o** option), and have no line terminators (intended for use with the **-p** option).

The **-e** option converts ASCII input to EBCDIC output.

The **-i** option followed by a number (i_size) defines the size of the input lines. If the **-f** option is set, each input line will be exactly i_size characters long; if the **-f** option is not set, the **-i** option defines the maximum input line size (line terminators \n or \r\n are not counted). Without the **-i** option, input lines default to 80 characters (if the **-f** option is set) or to 512 characters (the **-f** option is not set). Input lines that are longer than the specified i_size are artificially broken into multiple lines, so a small **-i** value cannot be used to truncate long lines. To truncate long output lines, use an adequate i_size with the **-o** option.

The **-o** option followed by a number (o_size) defines the size of output lines. If the **-g** option is set, each output line will be exactly o_size characters long; if the **-g** option is not set, the **-o** option defines the maximum output line size (line terminators \n or \r\n are not counted). Without the **-o** option, output lines default to 80 characters (if the **-g** option is set) or to 512 characters (if the **-g** option is not set). Output lines that are longer than the specified o_size are truncated.

Notes

Lines are processed one at a time, in ten steps:

1. A line is read into the input buffer. If the **-f** option is set, then exactly i_size (defined by the **-i** option) characters are read; if the **-f** option is not set, reading of the line proceeds character by character until one of the following happens:

   - the \n character is read
   - the \0 character is read
   - the input buffer contains i_size characters.

   Any \r characters encountered are discarded. The terminating character is not stored in the buffer.

2. If the **-a** option is set, the characters in the input buffer are transformed from EBCDIC to ASCII.

3. If the **-s** option is set, the trailing blanks in the input buffer are discarded.

4. If the **-l** option is set, all upper case characters in the input buffer are changed to lower case characters.

5. If the **-u** option is set, all lower case characters in the input buffer are changed to upper case characters.

6. The characters in the input buffer are moved to the output buffer. If the **-t** option is set, all TAB characters are expanded during this process; if the **-b** option is set, multiple spaces (in the appropriate positions) are replaced by a TAB character. If neither **-t** not **-b** are set, the input characters are simply moved to the output buffer. In all cases, characters exceeding the o_size are discarded.

7. If the **-p** option is set, the output line is padded with blanks to o_size characters.

8. If the **-e** option is set, the characters in the output buffer are converted from ASCII to EBCDIC.

9. If the **-g** option is not set, a \n character (or a \r\n pair if the **-r** option is set) is added after every o_size characters in the output buffer.

10. All characters in the output buffer are written to the output file.

Examples

The first example creates a compressed version of the input file "text" (extra spaces are converted to TABs where possible) in the output file "text.new." Lines are less than 80 characters each.

```
convert -b text text.new
```

The next example creates an ASCII version of the EBCDIC input file "cobol.ibm" (containing card images, 80 characters each, of a cobol program) in the output file "cobol.asc."

```
convert -fas cobol.ibm cobol.asc
```

In the previous example, if the input is on 9-track tape:

```
mode tp rewind
convert -fas /dev/tp cobol.asc
```

In the previous example, if the input is on cartridge tape:

```
rcopy /dev/fted | convert -fas - cobol.asc
```

The last example creates a truncated (72 characters per line) version of the FORTRAN input file "prog.xxx" (120 characters per line terminated by \n) in the output file "prog.for." The first half of the command truncates the output lines at 72 characters; the second half strips all trailing blanks.

```
convert -o 72 prog.xxx | convert -s > prog.for
```

utility:	**CONVOBJ**
purpose:	This program converts a .obj file to a .o68 file
user access:	all users
summary:	convobj [-jms] filename
arguments:	pathname of a .obj file
options:	-j build jump table -m write map -s create a single module

Description

The Convobj utility converts the **.obj** files generated by the C compiler/Code generator to .o68 format to be used with the link68 linker. The conversion is intended for the cases when a user wants complete control over the linking process. The user must supply his own libraries and his own main module (probably written in assembler). For each module the Convobj utility creates up to three psects: CODE psect with REA and EXE attributes contains the code, IDATA psect with REA and WRI attributes contains initialized data, and UDATA psect with REA, WRI, and UNI attribute contains the uninitialized data. Global variables are entry points, not common psects. All percent signs (%) in the names are changed to dollar signs ($) because the Assembler does not allow the use of the percent sign in the names. Static functions are moved to the global module (with the name being the original file name). Static functions are not entry points. They are accessed through the global module entry point. The code of each module, including the jump table, is limited to 32 K.

Options

The -j option generates a jump table to access the external functions. This ensures that the Link68 utility will not generate the overflow errors. The -j option is not needed if the generated .bin program is small enough. Only if the Link68 utility reports word overflow errors the -j option should be used to generate long references to external functions.

The -m option causes an entry point map to be generated on a file with the extension .cmp. The entry point map might be useful in case static functions are used, to find their location in the global module.

The -s option squeezes all the modules in one file into a single module. Non-static functions still have entry points, but those entry points are all located in the same CODE psect--the only one.

Example

```
convobj -sm test
```

will convert **test.obj** file to **test.o68** file. The **test.o68** file will contain a single module with all functions concatenated together. The entry point map will be written to the file **test.cmp.**

utility:	**COPY**
purpose:	This utility copies a file.
user access:	all users
summary:	copy [-dftv] source-file destination-file [-dftv] file-list dirname
arguments:	two single file pathnames **or** one or more file pathnames **and** a directory pathname
options:	-d directory and device files ok -f force -t time -v verbose

Description

The Copy utility copies one or more files. It does not alter the source file(s).

In its simplest format, copy duplicates file **abc** as file **xyz,** with both files residing in the current directory:

```
% copy abc xyz
```

To copy to or from a directory other than the current directory is more complex:

```
% copy abc /usr/fred/xyz
```

Here the pathname of the destination file is specified. The file **abc** exists in the current directory. It is being copied to the directory **/usr/fred**, and its name is to be **xyz** in that directory.

In the command:

```
% copy abc /usr/fred
```

the pathname of the destination directory is specified. The file **abc** exists in the current directory and is being copied to the directory **/usr/fred** without having its name changed.

The following form of the command can be used to create copies of all C language programs in the directory **/usr/archives:**

```
% copy *.c /usr/archives
```

This Copy command copies all files in the current directory with filenames ending in **.c** to the directory **archives**. The files maintain their original names.

Options

The **-d** option allows directory and device files to be copied. If this option is not used, directory and device files are not copied. For example, a command such as:

```
copy -d /dev/tty2 data
```

can be used to transfer all characters typed at terminal 2 into the file named **data** until a terminating character is received. The terminating character for console devices is CONTROL-Z.

The **-f** option makes the copied file overwrite an existing file with the same pathname. If this option is not specified and another file exists with the destination pathname, an error is reported.

The **-t** option causes a file to be copied **only** if:

1. The file does not exist in the destination directory; or

2. The source file has been modified more recently than the destination file. This comparison is performed on a file-by-file basis.

The **-v** option displays the name of each file as it is copied.

Notes

With the exception of "time dumped," which is automatically zeroed, files are copied with ownership and times preserved. If the system clock is reasonably accurate, the **-t** option can be very useful.

utility:	**CPTREE**
purpose:	This program copies a tree.
user access:	all users
summary:	cptree [-ftv] source destination [file-list]
arguments:	source directory destination directory optional file list
options:	-f force -t time -v verbose

Description

The Cptree utility copies the source directory, and all its descendant directories and files, to the destination directory. Existing links within the source directory are preserved.

If a file list is specified, only files whose names match at least one of the names in the list are copied. Ambiguous filenames enclosed in quotation marks may be included in the file list.

Options

The **-f** option causes the copied files to overwrite any file with the same pathname. If this option is not invoked and another file exists with the destination pathname, an error is reported.

The **-t** option causes a file to be copied **only** if:

1. the file does not exist in the destination directory, or

2. the source file has been modified more recently than the destination file. This comparison is performed on a file-by-file basis.

The **-v** option causes display of the name of each file as it is copied.

Notes

With the exception of "time dumped," which is automatically zeroed, files are copied with ownership and times preserved. If the system clock is reasonably accurate, the **-t** option can be very useful.

Shell command:	**CREATE or CRE**
purpose:	This command creates a file.
user access:	all users
summary:	cre file-list
arguments:	one or more pathnames
options:	none

Description

The Create command is used to create one or more files.

The files are zero bytes in length and have default access privileges. They are owned by the user who created them and are in the domain of their creator's group.

If the specified pathname already exists, an error is reported.

Notes

This command makes a standard data file. Refer to the Makdir command or the Makdev utility program if you need to make a directory or device file.

utility: **CROGEN**
purpose: This program generates a Cromix-Plus Operating System.

user access: privileged user

summary: Crogen pathname [sysdef]

arguments: pathname of new system file
optional system definition file

options: -d include system debugger
-m include system map

Description

The Crogen utility generates a new operating system by creating a configuration file based on the data in a system-definition file.

The file **/gen/sysdef** is provided for this purpose. The **sysdef** file can be used as is or with possible modifications, or a user's own system-definition file may be substituted.

After Crogen creates the configuration file, Link68 generates the system file.

Shell command:	**DAEMON**
purpose:	This command prints the files spooled by the Spool utility.
user access:	all users
summary:	daemon device-name
arguments:	pathname of the device to be run by daemon
options:	none

Description

The Daemon command is usually run detached. Typically, it is started by the Spool utility, though a user can start it himself.

The daemon will examine the **/usr/spool** directory and select all files, one at a time, that are to be printed on the specified device. The files are selected in order of increasing priority. Files that were spooled with a forms number different than the current forms number are skipped. The daemon keeps running until it runs out of eligible files, or until killed by the Spool utility or by the user. Only one daemon can be running per device. If another daemon is started to run the same device while the first one is still active, the new daemon will die immediately.

Example:

```
# daemon /dev/typ &
```

This command starts a detached daemon to print the files spooled on **typ** printer.

utility:	**DAY**
purpose:	This program executes a command on the day specified.
user access:	all users
summary:	day [day-of-the-week command-line]
arguments:	day of the week command line
options:	none

Description

The Day utility executes a command on the day specified. Day checks the system clock for the specified day. This program is useful in applications that require certain tasks be done on certain days of the week.

Notes

When used without an argument, Day displays the name of the current day.

Example:

The following command line will remind you of a weekly Wednesday meeting.

```
% day wed echo "This is Wednesday, remember your meeting"
```

utility:	**DCHECK**
purpose:	This program verifies the integrity of a file system.
user access:	all users
summary:	dcheck [-s] [devname(s)]
arguments:	optional device name(s)
options:	-s salvage directory structure

Description

The Dcheck utility verifies the integrity of a file system's internal directory structure. If possible, Dcheck with the -s option should be run on an unmounted file system. If the file system that needs to be fixed is the root, Dcheck should be run by itself, with no other users or tasks running concurrently. If another task is writing to the disk, the results of Dcheck may be incorrect.

If the -s option is used while another task or user is using the disk, the directory on the disk may be irreparably damaged.

MESSAGES RETURNED BY DCHECK

Cannot read super block
The super block cannot be read.

Insufficient memory
There is not enough available memory to run Dcheck. Either free additional memory, or create a new disk with fewer inodes. (Use Cptree to transfer the contents of the disk to the new disk.)

Inode xxxxx, Cannot read inode
A disk I/O error occurred while trying to read the specified inode.

Inode xxxxx, error reading directory
A disk I/O error occurred while trying to read a directory.

Inode xxxxx, file "ffffff" has bad inode

Inode xxxxx is a directory inode that contains an active file with an impossible inode number. Use Ncheck to locate the directory. Then delete the offending file, and run Dcheck with the **-s** option.

Inode xxxxx, directory with more than 1 parent

A directory is linked to more than 1 parent directory. Use Ncheck to locate the names of the files, and delete all but one link. Then run Dcheck with the **-s** option.

Inode xxxxx, directory with wrong parent

This error indicates the inode is pointing to the wrong parent. Use the Dcheck utility with the **-s** option to correct this error.

Inode xxxxx, bad link count xxxxx, should be xxxxx

The number of names pointing to this inode from various directories is greater or less than expected. Use Dcheck with the **-s** option to correct this error.

Inode xxxxx, more than 255 links

There are more than 255 names for this inode. Use Ncheck to find all the names. Delete some names to bring the total number of names to 255 or less. Then run Dcheck with the **-s** option.

Inode xxxxx, bad inode number in inode

Each inode contains its own inode number. This error means the inode specified has the wrong number. Use Dcheck with the **-s** option to correct this error.

Inode xxxxx, unallocated inode with xxx links

Although this inode is unallocated, names point to it. Use Ncheck to find these names, then delete them.

Inode xxxxx, allocated inode with 0 links

This inode is still allocated, though there are no names for it. Use Dcheck with the **-s** option to correct this error.

Inode xxxxx, bad directory entry count

This inode is a directory. The number of directory entries in the inode differs from the actual number of directories. Use Dcheck with the **-s** option to correct this error.

End of Dcheck (This is the last message)

The program has finished executing.

Options

The **-s** option fixes problems reported by Dcheck. The program corrects an incorrect inode number when:

1. The inode is allocated;

2. The inode link is nonzero; and/or

3. The inode is being pointed to (i.e., is in use).

The program does not correct an incorrect inode number if the inode is unallocated.

Notes

Immediately after running Dcheck with the **-s** option, run Icheck with the **-s** option. After both programs are run, the system must be rebooted. Refer to the Boot utility for additional information.

It is not necessary to reboot if the **-s** option is not used or if Dcheck is run on an unmounted device.

Shell command:	**DELETE or DEL**
purpose:	This command deletes a file.
user access:	all users
summary:	del [-v] pathname(s)
arguments:	one or more pathnames
options:	-v verbose

Description

The Delete command removes a link to a file. If there is only one link to the file, the file is no longer accessible, and the space it occupied is made available.

Options

The **-v** option displays the name of each file as it is deleted.

Notes

To remove all links to a file, making it inaccessible, use the Ls command with the **-i** option to find the inode number of the file in question. Use that inode number as an argument to Ncheck, and find the names of all files linked to the file.

A directory may be deleted by specifying a directory pathname.

In order to delete a directory, it must not:

1. Contain any files;

2. Be the current directory for any user; or

3. Be the root directory of a device.

Examples:

In the following example, the file named **schedule** is deleted from the current directory.

```
% ls -m

     3,016      1 letter
       200      1 memo
     1,408      1 schedule

% del schedule
% ls -m

     3,016      1 letter
       200      1 memo
```

If there is more than one link to a file and one of the links is deleted, the file is no longer accessible through that link. The file remains on disk and is accessible through the remaining links.

The following example concerns itself with part of the **/dev** directory. As the Cromix-Plus Operating System is shipped, the dummy file **prt** is linked to the dot-matrix printer driver **lpt1**. In the first listing that follows, the link is shown by the 2 preceding each filename. When the file **prt** is deleted, the file **lpt1** remains intact, and the number of links is reduced to one.

```
# ls -m

5:5   C      2 lpt1
5:5   C      2 prt
6:5   C      1 typ1

# del prt
# ls -m

5:5   C      1 lpt1
6:5   C      1 typ1
```

utility:	**DELTREE**
purpose:	This program deletes a tree, including all files and subtrees.
user access:	all users
summary:	deltree [-a] pathname [file-list]
arguments:	pathname
options:	-a suppresses user verification

Description

The Deltree utility deletes all files and subtrees in the tree (directory) specified. Normally, Deltree prompts the user with the file or directory name, followed by a question mark. If the user types **y**, the file or directory is deleted; otherwise it is not. If the **-a** option is used, Deltree asks once whether the user really wants to delete the entire tree, instead of prompting for verification of each file. If the user types **y**, all files and subtrees are deleted. If **n** is typed, Deltree returns control to the Shell.

If Deltree is called from within the specified directory, the program will not allow the deletion of that directory. All the files must be deleted from a directory before the directory itself is deleted.

If the directory pathname is followed by a list of filenames, only the corresponding files will be deleted.

Options

After asking for verification, the **-a** option deletes all files and subtrees automatically.

Shell command:	**DIRECTORY** or **D**
purpose:	This command displays the name of or changes the current directory.
user access:	all users
summary:	d [dir name]
arguments:	optional directory pathname
options:	none

Description

When given without an argument, the Directory command displays the pathname of the current directory.

Given with a directory pathname, the Directory command makes the specified directory the current directory.

utility:	**DISKINFO**
purpose:	This program prints information from the hard disk label, the partition table, and the alternate track table.
user access:	privileged user
summary:	diskinfo devicename [id_line]
arguments:	device name optional identification line
options:	none

Description

The Diskinfo utility prints information from the label (on block 0) of the specified Cromemco hard disk, as well as the partition table and the alternate track table. The devicename argument should be the device name of the entire disk (e.g., **std31** on drive 0; or **std63** on drive 1). The optional id_line argument is printed on the first line of the printout for identification purposes.

"Location of alternate track table" and "Location of partition table" are the starting byte numbers of the respective tables. The "Location of alternate tracks" is the starting cylinder number of the alternate tracks. The "Start of write precompensation" is the cylinder number where write precompensation begins. On disks without write precompensation, this number equals the number of cylinders. The partition table (if any) is printed after the disk parameters, followed by the alternate track table. For example, the command

```
# diskinfo std31 "Main System Micropolis Disk Serial Number 12345"
```

prints the following:

```
Main System Micropolis Disk Serial Number 12345

Device /dev/std0 6:31 STDC Hard Disk   May-21-1985 10:25:06

Disk parameters
        Number of heads..................     =  6
        Number of cylinders..............     =  830
        Number of alternate tracks.......     =  60
        Start of write precompensation...     =  830
        Location of alternate tracks.....     =  407

        Number of sectors per track......     =  20
        Bytes per sector.................     =  512
        Location of alternate track table     =  9792
        Location of partition table......     =  9728
        Starting cylinder of disk........     =  1
        Disk label.......................     =  CSTD
```

```
   Partition table
     410

    Alternate track table
     # bad track alt track      # bad track alt track      # bad track alt track
         hd  cyl   hd   cyl          hd  cyl   hd   cyl          hd  cyl   hd   cyl
     0   0    53    2   410      1   0    65    3   407      2   0   143    4   407
     3   0   144    5   407      4   0   145    0   408      5   0   146    1   408
     6   0   147    2   408      7   0   241    3   410      8   0   300    3   408
     9   0   368    4   408     10   0   614    4   410     11   0   615    5   408
    12   0   616    5   410     13   1   183    0   411
```

utility:	**DUMP**
purpose:	This program displays a file in hexadecimal and ASCII.
user access:	all users
summary:	dump [-b #] [file-list] [-e #] [-k #] [-s #] [-o #]
arguments:	one or more optional file pathnames
options:	-b first byte -e last byte -k first block -s width -o offset address

Description

The Dump program displays the file(s) specified by the pathname(s). When called without an argument, Dump waits for input from the standard input.

Dump displays any type of file. The file is displayed in hexadecimal with an ASCII equivalent to one side. All numeric arguments to the Dump utility are assumed to be decimal numbers unless followed by an **h** (for hexadecimal).

Options

The **-b** option allows the user to specify the first byte of a file to be dumped.

The **-e** option allows the user to specify the last byte of a file to be dumped.

The **-k** option allows the user to specify the first block to be dumped.

The **-s** option allows the user to specify the swath width of the dump.

The **-o** option causes a specified offset to be added to all addresses displayed by Dump.

Example:

```
% dump -b 1000h -e 5000h filename
```

This command dumps the file **filename** starting with the 1000th (hex) byte and ending with the 5000th (hex) byte.

utility:	**ECC**
purpose:	This program manipulates the error-correcting memory controllers.
user access:	privileged user
summary:	ecc [-e] [-t #] [ON] [OFF]
arguments:	Optional keyword ON or OFF
options:	-e examine memory controller -t # reexamine every # seconds

Description

The Ecc program can enable, disable, or examine the error-correcting memory controllers. When called without options or arguments, Ecc reports the controller status (enabled or disabled).

If Ecc is called with the -e option, it will examine the memory controllers, report any recorded errors to the standard output, and clear the error status. If Ecc is called with the -t option, the memory controllers will be scanned periodically as defined by the numerical argument following **-t.**

Options

The -e option requests the Ecc program to scan the controllers and report errors.

The -t option, followed by a number, defines the number of seconds between two consecutive scans.

Example:

```
    # ecc on
    # ecc -t 30 &
```

This sequence of commands enables the error-detection hardware and then scans it every 30 second as a detached process. The errors will be reported on standard output.

utility:	**ECHO**
purpose:	This program echoes its arguments to the terminal.
user access:	all users
summary:	echo text
arguments:	any text
options:	-e send to STDERR -n do not print newline

Description

The Echo program echoes its arguments. Text may be enclosed within single or double quotation marks to insure correct interpretation by the Shell.

Options

The **-e** option echoes arguments to the standard error channel.

The **-n** option suppresses the echo of a newline.

Shell command:	**EXIT or EX**
purpose:	This command exits from a Shell.
user access:	all users
summary:	ex [return-value]
arguments:	Optional value returned to the parent process
options:	none

Description

The Exit command exits from a shell. If no higher level shell is active, the Cromix-Plus Operating System logs the user off the system. If there is a higher level shell active, it receives from the exiting shell a return value as defined in the following paragraph.

If the Exit command is used without an argument (or if a command file runs to end-of-file), the return status will be -1 if an error such as a syntax error was detected anywhere in the command file. If there was no error detected, the return value will be the value returned by the last executed command. An Exit command with an argument (a value) can be used to return that value.

Return values are used by the "if -err" command. Each shell keeps track of the value returned by the last command, and this value is used to determine the truth of the "if -err" condition. Any nonzero value indicates an error condition.

Example:

Here is the contents of a possible command file to compile (and, optionally, link) a number of C source files. A user-defined -l option requests linking. The command file will abort if any step returns an error.

```
 if "#1" = "-l" shift                % Get rid of -l
 if "#1" = "" goto huh               % If no arguments

 %compile                            % Compile all source files
 if "#1" = "" goto endcompile        % If none is left
 c #1                                % C compiler
 if -err goto err                    % If errors in compilation
 code #1.i #1.obj                    % Generate code
 if -err goto err                    % If errors in code generation
 del #1.i #1.scr >* /dev/null        % Delete scratch files
 shift                               % Next argument
 goto compile
%endcompile

rewind
if "#1" != "-l" exit                 % No linking requested
shift                                % Get rid of -l
crolinker #* /usr/lib/clib /usr/lib/paslib
exit

%err                                 % In case of errors
 del #1.i #1.scr >* /dev/null        % Delete scratch files
 exit 1                              % Return error

%huh                                 % If no arguments
 echo "Huh?"
 exit 1                              % Return error
```

If both "exit 1" commands were replaced by "exit" command, the return value would be the value returned by the last command (Delete or Echo).

utility:	**FIND**
purpose:	This program locates files.
user access:	all users
summary:	find pathname [!] expression
arguments:	pathname [!] expression(s)
options:	File specifiers: -name -type x -links n -user name or number -group name or number -size n -blocks n -mtime n Action Specifiers: -exec command-line -ok command-line -print Logical Operators: -a -o

Description

The Find utility locates a file. The pathname is the pathname of the tree, directory, or file to be searched, and the expression is the string to be found and what is to be done with it.

Expressions are combinations of file criteria and operations. Refer to the following list.

Parentheses may be used to change the order of evaluation of the items in the Find expression. Used with parentheses, the expression must be enclosed within quotation marks so that the Shell passes them to the Find utility.

When one of the action specifiers is used to execute a program, the return value of that program can be evaluated and used within the expression.

The **!** operator may precede the expressions to negate the sense of the tests.

Options

File Specifiers

-name file-list

The file-specifying keyword **name** is followed by a list of one or more unique or ambiguous filenames. If an ambiguous filename is used, it must be enclosed within quotation marks. The Find utility finds all files that match the file list.

-type b block device
c character device
f file
d directory

The file-specifying keyword **type** is followed by either **b**, **c**, **f**, or **d**, as shown. The Find utility finds all files of that type.

-links n

The file-specifying keyword **links** is followed by a number, **n**. The Find utility finds all files with that number of links. If the number is preceded by a plus sign, all files with more than that many links are found; if a minus sign is used, all files with fewer than **n** links are found.

-user name
number

The file-specifying keyword **user** is followed by a user name or number. The Find utility finds all files owned by the specified user.

-group name
number

The file-specifying keyword **group** is followed by a group name or number. The Find utility finds all files owned by the specified group.

-size n

The file-specifying keyword **size** is followed by a number, **n**. The Find utility finds all files of the specified size, in bytes. If the number is preceded by a plus sign, all files with more than that number of bytes are found; if a minus sign is used, all files with fewer than **n** bytes are found.

-blocks n

The file-specifying keyword **blocks** is followed by a number, **n**. The Find utility finds all files using that number of blocks (actual number of blocks occupied by the file). If the number is preceded by a plus sign, all files occupying more than the specified number of blocks are found; if a minus sign is used, all files with fewer than **n** blocks are found.

-mtime n — The file-specifying keyword **mtime** is followed by a number, **n**. The Find utility finds all files modified **n** days ago. If the number **n** is preceded by a plus sign, all files modified **n** or more days ago are found; if a minus sign is used, all files modified fewer than **n** days ago are found.

Action Specifiers

-exec command-line

The action-specifying keyword **exec** is followed by a command line. This may be any valid command line, that is, any line that can be entered in response to the Shell prompt. This command line is then executed each time the Find utility finds a file meeting the find criteria. A pair of braces ({}) may be placed within the command line. They will be replaced by the name of the file found.

-ok command-line

The action-specifying keyword **ok** is used in the same manner as **exec**. When **ok** is used, the Find utility prompts the user prior to executing each command line. The user may respond with a **y** to execute the command line or **n** to prevent its execution.

-print — The action-specifying keyword **print** is used to display the pathnames of files found.

Logical Operators

-a — The **-a** operator is used to logically **AND** two items in the Find expression.

-o — The **-o** operator is used to logically **OR** two items in the Find expression.

Notes

The expression used with the Find command is evaluated from left to right. Items to be found and actions to be performed may be combined logically by use of the **-a** and/or **-o** logical operators. Either operator combines the sum of the expression to its left with the subsequent item in the expression. For example:

```
find / -name ted -a -print

find / -name ted -o -name mary -a -print
```

The first example finds all files with the filename **ted** and prints the pathnames of these files. If the print instruction is omitted, all of the correct files are found, but no action is taken: their names are not displayed. The second example demonstrates the use of the logical OR. All files with the filename **ted** OR **mary** are found and their pathnames printed.

Examples:

The following example finds all subdirectories of the current directory, then executes an **ls** command with the **-d** and **-e** options.

```
% find . -type d -a -exec l -de {}
```

The next example finds all entries with a **.c** extension, then lists the entry with the -l option.

```
# find / -name "*.c" -a -exec ls -l {}
```

utility:	**FIXSB**
purpose:	This command file restores the Superblock.
user access:	privileged users
summary:	fixsb
arguments:	none
options:	none

Description

The Fixsb utility file restores the Superblock, should it be destroyed accidentally. This command file has the same function as the Makfs utility used with the **-r** option, but without the possible risks associated with running an older version of the Makfs utility.

After restoring the Superblock, the Fixsb command automatically runs Icheck, to check inodes in the file.

Notes

Fixsb is only to be used on disks whose file systems were created with the default number of inodes. Refer to the Makfs utility for additional information.

utility:	**FLUSH**
purpose:	Writes system I/O buffers to disk
user access:	all users
summary:	flush # &
arguments:	number of seconds
options:	none

Description

The Flush utility is a system-maintenance program that flushes (i.e., writes to disk) system buffers by executing the **_update** system call every specified number of seconds.

The **startup.cmd** file can invoke Flush, as a background process with an interval of, say, 30 seconds, every time the system is booted.

utility:	**FREE**
purpose:	This program displays the amount of unused space remaining on a device.
user access:	all users
summary:	free [devname1 ... devnameN]
arguments:	optional list of device names
options:	none

Description

The Free program displays the amount of unused space remaining on a specified device. If no device is specified, the free space is displayed for all mounted devices.

Example:

The following is a sample output of the Free utility. It shows the available free space in blocks, kilobytes, and bytes.

```
/dev/root      7,513 blocks      3,756K      3,846,656 bytes
```

To extract a file without overwriting the original
d [illegible] ftar
[illegible] bin
[illegible]
[illegible]
[illegible]

utility: **FTAR**
purpose: Creates and retrieves file archives.

user access: all users

summary: ftar options archive [file-list]
/dev/sfd0

arguments: archive name
optional filenames

options: exactly one of the following per command:

-c	create files on new archive
-t	list files on archive
-x	extract files from archive
-y	compare files from archive

other options

-d	update the time dumped for each processed file
-i	get list of files from stdin
-u	use Unix compatible Tar format
-v	list the names of files processed
-w	wait for confirmation before processing
-k #	archive size in kilobytes
-b #	buffer size in kilobytes

Description

The Ftar utility can create a file archive, retrieve files from an archive, or list the contents of an archive. An archive can be written to, or read from, any device or file. Before using Ftar to back up files onto a device, the device must be initialized.

The first argument is the archive file (or device) name. If the first argument is "-", the standard input (or standard output) can be used to "pipe" data to and from an archive. The remaining arguments are the names of files and directories to be read or written. If a directory name is specified, all the files in the directory are transferred (recursively).

Archives created by the Tar utility cannot be restored with Ftar.

Options

One of the following for each Ftar command:

-c Creates a new archive of the named files, device files, and/or directories. Any existing data in the archive is overwritten.

-t Lists the named files, device files, and/or directories in the archive. If no names are specified, the entire archive is listed.

-x Copies the named files, device files, and/or directories in the archive to the current directory (unless the extracted files are governed by absolute pathnames). The owner, modification time, and access privileges are restored. If no file arguments are given, the entire archive is extracted. If there are multiple files of the same name, the last one overwrites all of the previous.

-y Similar to the **-x** option, except that the named files, device files, and/or directories are compared with the identically named files in the current directory. Any differences are reported.

Other options

The **-d** option updates the time dumped for each file written by Ftar; the dump time is useful for selective Ftar operations (refer to the last example).

The **-i** option accepts filenames from the standard input (until a CONTROL-Z is encountered) and writes the files to an Ftar archive. The explicitly listed files are processed first.

The **-u** option prevents writing of device files and directories to an archive (to conform to the UNIX Tar utility). During restore, directories are generated as needed.

The **-v** option lists each file as it is processed. When used with the **-t** option, a fuller description of the file is given.

The **-w** option prompts for confirmation before processing each file. Enter "y" or "Y" to confirm the action; any other response cancels the action.

The **-k** option, followed by a number expressed in 1-kilobyte units, limits the archive to the specified size. The **-k** option works in all cases, but is necessary only for nine track tapes.

The **-b** option, followed by a number expressed in 1-kilobyte units, defines the size of the Ftar buffer. The **-b** option is ignored unless the buffer size specified is larger than the one calculated by Ftar.

Notes

With the **-t**, **-x**, or **-y** options, directories are extracted, listed, or compared with all their descendent directories. When extracting a subtree, all directories above the one being extracted must exist prior to calling Ftar.

The Ftar utility permits multiple-volume archives if the archive is written to (or read from) a block device. On Cflop devices (non-uniform floppy disks), archives start at block 1; on Stdc devices (hard disks), archives start at block 20. On all other devices, archives start at block 0. There is no Ftar label, so there is no need to first create an empty archive. The only requirement is that the device must be initialized.

Except for Cflop and Stdc devices, archives written directly to a device are the same as archives piped into the Rcopy utility. For Cflop and Stdc devices, Rcopy must be told where to start writing. However, pipes cannot be used with multiple-volume archives.

Some filenames in archives created by UNIX may be illegal under Cromix-Plus. Cromix-Plus truncates filenames over 24 characters, and may create duplicate filenames in the process. To prevent Ftar from overwriting such files, each file must be extracted separately (using its full name) and renamed. The Ftar utility also converts upper case characters to lower case, and all illegal characters (e.g. -, [,], etc.) to the "$" character.

Examples

To view the progress of directory backups to a new archive of one or more small floppy disks:

```
ftar -cv /dev/sfdc directory_names
```

To view the progress of file backup to a new archive on floppy tape:

```
ftar -cv /dev/ftcd file1 file2 ...
```

To extract all files from the archive on a floppy tape:

```
ftar -xv /dev/ftcd
```

To compare files in the archive on a floppy tape with existing files:

```
ftar -yv /dev/ftcd
```

To list the contents of a tape archive in long form:

```
ftar -tv /dev/tp1
```

To write a floppy tape with the file names entered from standard input (or from a pipe):

```
ftar -cvi /dev/ftcd
```

To create a new archive on cartridge tape of all the ordinary files in the "my" directory that have been modified since being transferred (refer to the Scan utility):

```
scan my 'type == is_ordin && tmodify >
  tdumped && print(path)' | ftar -divc /dev/ftcd
```

The example above will not work (all "my" directory files will be copied) unless the -d option was set when the files were first transferred by Ftar.

Shell command:	**GOTO or GO**
purpose:	This command causes transfer of control within a command file.
user access:	all users
summary:	go label
arguments:	line label
options:	none

Description

The Goto command transfers control within a command file. Control is transferred to the line specified by **label**. This command is used to execute the same commands within a command file repeatedly. When used in conjunction with the If and Shift commands, the Goto command becomes part of a conditional loop with varying parameters.

A percent sign (%) anywhere on a line means the rest of the line is a comment. A comment at the beginning of a line--with no space after the percent sign--is a label.

The Goto command given with a nonexistent line label causes termination of command file execution.

Example:

```
%sample_label
x
y
z % this is a comment
goto sample_label
```

This sample command file causes repeated execution of the commands **x**, **y**, and **z**. The first line of the command file is a line label, as indicated by the leading percent sign.

The percent sign indicates a comment on the fourth line of the file. The fifth (last) line of the file transfers control to the specified label (sample_label).

utility:	**GROUP**
purpose:	This program allows users to change their groups.
user access:	all users
summary:	group [group name]
arguments:	new group name
options:	none

Description

The Group utility examines the **/etc/group** file for a named group. If this group is not found, an error message is displayed, and execution of the utility is aborted.

If the named group is found and if there is a password associated with it, the Group utility prompts for the password. If the user responds with the correct password or if there is no password associated with the group, a new shell is formed in which the user belongs to the named group. Upon exiting from the newly created shell, the user's previous status is reinstated.

utility:	**H** or **HELP**
purpose:	This program displays pages from the user manual on Shell commands and utility programs.
user access:	all users
summary:	help [-d directory] [-e escape file] [topic ...]
arguments:	optional command or utility names
options:	-d name of directory with help files -e name of file with escape sequences

Description

The Help utility program provides on-line descriptions of the Shell commands and utility programs (for information regarding other aspects of the system refer to the appropriate Cromix-Plus manual).

If you enter "help" and press RETURN, the Help utility lists the available topics and prompts you to select one. Entering "help" and a program name will list the manual entry on that topic. Help displays the manual entry one page at a time. The percentage of the file yet to be viewed is displayed at the bottom of the screen.

The following functions will aid you in viewing the manual entry:

SPACE BAR	Display the next page.
u	Display the previous page.
RETURN KEY	Display the next line.
DOWN ARROW	Display the next line.
UP ARROW	Display the previous line.
b	Display the first page.
h	Display the list of available functions.
r	Return to list of available topics if the Help utility was called with no arguments.
q	Exit the program.

Trying to move outside the file (e.g., pressing SPACE BAR when the last page is displayed) has no effect.

Modifying the On-line Manual

The text files for the on-line manual are in the **/usr/help** directory (unless the **-d** option selects an alternate directory). Each topic is described in a file of the same name (plus a **.hlp** filename extension). To add more help files, all filenames must have the **.hlp** extension, otherwise the Help utility cannot access them.

Modifying the Format File

The **/usr/help** directory has three format files: **help.msg, help1.msg,** and **help2.msg.** The file **help1.msg** is a simpler version of **help2.msg,** which takes advantage of the attributes of the Cromemco 3102 terminal. As shipped, **help2.msg** is linked to the file **help.msg,** which is used by the Help utility to display the manual (unless the **-e** option selects an alternate file).

The format file must have 10 lines, each with a different format string:

Line #1	Title line of directory listing
Line #2	Topic prompt
Line #3	Percentage display of unviewed part of file
Line #4	Error message if incorrect command is given
Line #5	Help message
Line #6	Sequence to clear the line containing the cursor
Line #7	Sequence to clear screen
Line #8	Sequence to move cursor to home position
Line #9	Sequence to insert an empty line above the line containing the cursor
Line #10	Sequence to move the cursor to the last line on the screen

The substring %ld%% on line 3 generates a string with an associated value.

Besides the normal printing characters, a format string may have any of the following special characters:

\n	new line character
\r	return character
\t	tab character
\f	form feed character
\b	backspace character
\ddd	character defined by three octal digits ddd (e.g., \033 is the ESC character)

utility: **ICHECK**
purpose: This program verifies the integrity of a file system.

user access: all users

summary: icheck [-sm] [-b blk# ...] [devname ...]

arguments: optional list of device names

options:
-s salvage
-b blocks
-m missing blocks

Description

The Icheck utility verifies the integrity of the file system's inode structure. After a power failure or after the computer has been reset, run Icheck on all devices that were mounted at the time of failure.

If no device names are specified, Icheck checks the integrity of all mounted devices. The list of mounted devices is obtained from the file **/etc/mtab.**

If no options are specified, Icheck produces a report on the file system, but does not alter it. A sample report and explanation follow.

If the -s option is used while another task or user is using the disk, the directory on the disk may be irreparably damaged.

```
% icheck

Device:  /dev/fda

Blocks missing:                         0
Bad free blocks:                        0
Duplicate blocks in free list:          0
Blocks in files and in free list:       0
Bad blocks:                             0
Duplicate blocks:                       0

Device files:                          16
Ordinary files:                       117
Directories:                           14
Blocks used in files:               1,462
Free blocks:                          845
Free inodes:                          358
```

Blocks missing

All disks (also referred to as block devices) are divided into allocation units called **blocks.** A block is 512 bytes. Every block should appear either in a file or in the **free list.** Blocks appearing in files include those permanently assigned as either system or inode blocks. The free list is a list of all blocks available for use.

A block is **missing** if it appears neither in a file nor in the free list. Missing blocks do not compromise the integrity of the file system, and the problem does not need to be corrected immediately. If a block is missing, it is simply not available for use.

The problem may be corrected by executing Icheck with the **-s** option.

Bad free blocks

This message pertains to blocks located in the free list. The term **bad** indicates that the block number is out of range. A block number can be out of range if it is:

1. Past the end of the disk;
2. In the system area of the disk; or
3. In the inode area of the disk.

Bad free blocks **do** compromise the integrity of the file system, and the problem should be corrected immediately by executing Icheck with the **-s** option. No files are affected.

Duplicate Blocks in Free List

This message means the same block number appears twice in the free list.

Duplicate blocks in the free list **do** compromise the integrity of the file system, and the problem should be corrected immediately by executing Icheck with the **-s** option. No files are affected.

Blocks in Files and in Free List

This message means some blocks appear both in files and the free list. Blocks in files and in free list **do** compromise the integrity of the file system, and the problem should be corrected immediately by executing Icheck with the **-s** option.

Bad Blocks

This is similar to **Bad free blocks** except that the bad blocks appear in files.

Bad blocks **do** compromise the integrity of the file system, and the problem should be corrected immediately.

Icheck reports the inode number of the bad blocks. The Ncheck utility is then used to determine the names of the files containing bad blocks. These files must be deleted. The file may be copied to another file before it is deleted; the new file should be carefully checked because it will probably not be correct.

Duplicate Blocks

This is similar to **Duplicate blocks in free list** except that the duplicate blocks appear in files.

Duplicate blocks **do** compromise the integrity of the file system and the problem should be corrected immediately.

Icheck reports the inode number of the duplicate blocks. The Ncheck utility is then used to determine the names of the files containing duplicate blocks. At least one of these files must be deleted. The Icheck utility should then be run with the **-s** option.

The file may be copied to another file before it is deleted and should be carefully checked because it will probably not be correct.

MESSAGES RETURNED BY ICHECK

Cannot read super block
The super block cannot be read.

Insufficient memory
The disk contains too many inodes for Icheck to check. Free additional memory or create a new disk with fewer inodes and use Cptree to transfer the contents of the disk to the new disk.

Inode xxxxx, cannot read inode
A disk I/O error occurred while trying to read the specified inode.

Inode xxxxxx, bad usage count
This inode has an incorrect usage count. The usage count is used by the Usage utility program to calculate the amount of disk space used. This error can be corrected by running Icheck with the **-s** option.

Inode xxxxxx, cannot write to inode
This error message occurs when the Icheck utility is attempting to correct an inode and an error occurs.

Block xxxxxx, inode xxxxxx, block used in file
This is not an error message. This message is displayed when the **-b** option is used, indicating the number of the inode in which the specified block is used.

Block xxxxxx, inode xxxxxx, bad block number
Refer to the previous discussion of **Bad blocks.**

Block xxxxxx, inode xxxxxx, duplicate block in the system
Refer to the previous discussion of **Duplicate blocks.**

Block xxxxxx, inode xxxxxx, duplicate block in free list
Refer to the previous discussion of **Duplicate blocks.**

Block xxxxxx, inode xxxxxx, block in file system and free list
Refer to the previous discussion of **Duplicate blocks.**

Block xxxxxx, missing block
This message is printed when the **-b** option is used to find the status of a certain block and the block is missing. Refer to the previous discussion of **Blocks missing.**

Block xxxxxx, block in free list
This message is printed when the **-b** option is used to find the status of a certain block and the block is in the free list.

Block xxxxxx, bad free block
Refer to the previous discussion of **Bad free blocks.**

Block xxxxxx, cannot write free list block
When running Icheck with the **-s** option, the free list is recreated. This error message is printed when there is an error in writing the free list.

Block xxxxxx, cannot read block
This message is printed when a block cannot be read.

Options

The **-s** option salvages and recreates the free list.

The **-b** option displays information about blocks. The option must be followed by a list of block numbers. The numbers must be separated by commas, or they must be separated by spaces and the entire list enclosed in quotation marks.

The **-m** option prints the numbers of missing blocks.

Notes

When using the **-s** option, Icheck must be used in conjunction with the Dcheck utility. Icheck is run after Dcheck. Both utilities should be run using the **-s** option. After both programs are run, the system must be rebooted. It is not necessary to reboot if the **-s** option is not used or the device was not mounted. Refer to the Boot utility for additional information.

Do not execute the Icheck utility when other processes are being executed. This includes detached processes as well as other user processes.

utility:	**IDUMP**
purpose:	This program displays the contents of an inode.
user access:	all users
summary:	idump [-x] blockdev inode-list
arguments:	block device name list of one or more inode numbers
options:	-x dump inode in hexadecimal

Description

The Idump utility displays the contents of the specified inode(s). Unless the **-x** option is used, inodes are displayed in symbolic form.

Shell command:	**IF**
purpose:	This command is used to conditionally execute another command.
user access:	all users

summary:

```
if -err command
   -rewa filename command
   string-1 = string-2 command
   string-1 != string-2 command
```

arguments: error condition specifier

or

access method and a filename

or

two strings separated by the equal (=) or not equal (!=) relational operator

and

a command line

options: none

Description

The If command is used to place a condition on the execution of another command. It is frequently used in conjunction with the Goto command. Referring to the summary above, the If command has three basic forms.

The first form executes a command if the previous command returned an error. (Refer to the discussion of the Exit command.)

In its second form, the If command causes **commands** to be executed if a particular access method applies to the file specified.

The third and fourth forms of the If command cause **command** to be executed when the specified relational condition is true or false. Neither of these forms of the If command requires that the strings be enclosed in quotation marks. However, both forms **do** require a space on either side of the relational operator (= or !=).

utility: **INITFLOP**
purpose: This program initializes a floppy disk.

user access: privileged user

summary: initflop [[-dsvz] [-c #[,#]] [-h #[,#]
[-l name] [-u #] [-n #] devname]

arguments: Cromix device name

options:

-c CYLINDERS to initialize
-d Single DENSITY
-h HEADS to initialize
-l CDOS disk label name
-n Number of CDOS directory entries
-s Single SIDED
-u UNIFORM format disk sector size
-v Verbose
-z CDOS format disk

Description

The Initflop program is used to initialize floppy disks. Parameters may be passed from the command line or interactively from the terminal as the program executes.

Following is a sample script of a typical Initflop session to format a small (5-inch) Cromix floppy disk. The user's responses are printed in boldfaced type--everything else is displayed by Initflop. RETURN is pressed to select a default response (default values are displayed in angle brackets):

```
Initialize Floppy Disks        version                    xx.yy

Press:     RETURN to supply default answers
           CTRL-C to abort program
Warning:   INITFLOP can destroy all disk data

Disk to initialize (devname)?   sfdd

Testing:
     Rotational speed:  300 RPM

Formatting
     Disk type (C=CDOS, X=CROMIX, U=UNIFORM)?  <X>    RETURN
     Single or double sided (S/D)?  <D>    RETURN
     Single or double density (S/D)?  <D> RETURN
     Single or double tracked (S/D)?  <S> RETURN

     First cylinder (0-27H)?  <OH>    RETURN
     Last cylinder (0-27H)?  <27H>    RETURN
     Surfaces (0-1,All)?  <All>       RETURN
```

Surface,	Cylinder
0	00
1	00
0	01
1	01
:	:
:	:

The Initflop program first asks for the device name of the drive containing the disk to be formatted. Legal responses are the device names of the floppy disk drives connected to your system, such as **fda, fdb, sfda, sfdb,** and so on. Absolute pathnames may also be used, for example **/dev/fda** or **/dev/sfdb.** Be sure to specify the device name correctly, as the Initflop program destroys all data on a floppy disk.

Initflop briefly tests the specified drive. Drive speed, which is particularly important when formatting, is displayed by the program.

Next, Initflop prompts for the "type" of disk to be formatted. That is, will the disk be used with the CDOS Operating System or with the Cromix Operating System, or is it a uniform-density disk?

Initflop then prompts for information about disk sides and density. Double-sided, double-density is the default.

Next, Initflop asks for the numbers of the first and last cylinder to be formatted. On occasion, it may be necessary to format only a portion of a disk (for example, one track may be experiencing frequent errors). Selecting the default responses will format the entire disk.

Initflop then prompts for the surfaces to be initialized. To format only one surface, select the corresponding value from the prompt. Selecting the default by pressing RETURN initializes all surfaces.

At this point, Initflop proceeds to format the disk. As the disk is formatted, cylinder and surface numbers are displayed. For proper head positioning, the disk is formatted from the outermost cylinders inward.

When Initflop is called with a device name as an argument, the responses to the preceding prompts are taken from the command-line options. If no options are specified, double-density, double-sided Cromix disk, with all heads and cylinders formatted is selected by default.

The **-d** option specifies a single-density disk.

The **-s** option specifies a single sided disk.

The **-z** option specifies a CDOS-format disk. When the **-z** option is used, the disk label name (up to 8 characters) may be specified with the **-l** option and the number of directory entries (64 to 512) may be specified with the **-n** option. The disk label is only written if the track containing the label has been formatted.

The **-u** option, with a sector size as an argument, specifies a uniform-density disk with the corresponding sector size. Valid sector sizes are 128, 256, 512, and 1024 bytes.

The **-c** option can be used to specify the cylinders to be formatted. If only one argument is given, only that cylinder will be formatted. If two arguments are given, all cylinders from the first argument through the second argument will be formatted.

The **-h** option can be used to specify that only certain surfaces are to be formatted. If only one argument is given, only that surface will be formatted. If two arguments are given, all surfaces from the first argument through the second argument will be formatted.

When arguments are given on the command line, Initflop normally operates in the "quiet" mode. To display information about the type of disk being formatted, and Initflop's progress, use the **-v** (verbose) option.

Example

```
# initflop -zv -c 1,3 -l library -n 64 sfdc

Device name:  sfdc
Rotational speed:      300 RPM
CDOS disk      Double sided      Double density
First cylinder:  1.
Last cylinder:  3.
All surfaces

surface,cylinder
0          00
1          00
0          01
:          :
:          :

Disk name:  library
Date on disk:  10/10/84      (current date is printed)
Number of directory entries:  64
```

Had the **-v** option been omitted, no information would have been displayed.

utility:	**INITHARD**
purpose:	Initializes an STDC, SMD, or WDI-II hard disk
user access:	privileged user
summary:	inithard [[-v] [-c #[,#]] [-h #[,#]] devname]
arguments:	optional Cromix device name
options:	-v verbose -c cylinders to initialize -h heads (surfaces) to initialize

inithard -v -c 56 -h 1 /dev/hd0

Description

The Inithard utility initializes STDC, SMD, or WDI-II hard disks. No other processes, including Flush, should be running while initializing a hard disk. Inithard must be run interactively to initialize the entire hard disk, but a single partition may be initialized from the command line.

Options

The **-c** option, followed by one cylinder number, formats only that cylinder. Following **-c** with two numbers (separated by a comma), formats all cylinders from the first cylinder number through the second.

The **-h** option, followed by one surface number, formats only that surface. Following **-h** with two numbers (separated by a comma), formats all surfaces from the first surface number through the second.

The **-v** option provides status reports of the initialization when Inithard is run from the command line.

Example

A typical interactive session to format an entire STDC hard disk is shown below (user responses are boldfaced). Press RETURN to select the default response (displayed in angle brackets) of any prompt. To begin, enter **inithard** (as a privileged user) and press RETURN.

```
Inithard version xx.yy

Press:    RETURN to supply default answers
          CTRL-C to abort program
          To only do Alternate tracks and partitions answer
             "x" to "first cylinder" question.
Warning:  INITHARD can destroy all disk data

Device name?  std31
```

To initialize a single partition of the hard disk, enter the corresponding device name (e.g., **std0**); to initialize the entire disk, or to change the partition or alternate track tables, enter the device name for the whole disk (**std31** for STDC drive 0; **std63** for STDC drive 1).

If the disk has been initialized before, the following prompt is displayed:

```
Disk is already formatted.  Do you wish to continue (Y/N) <N> ?
```

Answer "Y" to continue. The following drive specification prompts appear only if you entered the device name for the entire disk (refer to the Cromix-Plus Administrator's Guide). If the disk has never been initialized, the default values are the possible maximums; otherwise the default values are the values entered during the last initialization. The following default values are for the Hitachi drive (STDC prompts only).

```
Number of surfaces (1 - 32) <7> ?
Number of cylinders (1 - 1024) <713> ?
```

The following prompts appear only for SMD drives:

```
Number of sectors (1 - 64) <36> ?
Interleave factor (1 - 35) <7> ?
```

The following prompt appears for all drives:

```
Max number of alternate tracks (0 - 112) <56> ?
```

The following prompt appears for STDC or SMD drives:

```
Starting cylinder of disk (0 - 713) <0> ?
```

The next prompt appears only for STDC drives:

```
Starting cylinder for write precomp <0 -713) <256> ?
```

The starting cylinder for write precompensation depends on the drive. On drives that do not need write precompensation, enter the number of cylinders (Refer to table 3-1). Additional information on write precompensation and other drive characteristics is provided in chapter 2 of the Cromix-Plus Administrator's Guide.

The next prompts ask for the area of the disk to be formatted (the numbers in parentheses depend on the number of heads and cylinders entered above):

```
First cylinder (0-713) <0> ?
Last cylinder (0-713) <713> ?
First surface (0-6) <0> ?
Last surface (0-6) <6> ?
```

If Inithard is called with a device name as an argument, the responses to the preceding prompts are taken from the command-line options. If no options are specified, all cylinders and all heads are selected by default. Caution: Formatting begins after you answer these prompts, and overwrites all data on the tracks specified. If you enter "x" and press RETURN for the first-cylinder prompt, Inithard will skip the formatting process and display the disk partition table.

As the tracks are formatted, the cylinder and surface numbers appear briefly on the screen. Note the cylinder and surface (head) numbers of any disk errors that appear, such as:

```
Disk error:  /dev/std31, cylinder 417., surface 0., status FF04
```

Next, if you are initializing the entire disk for an STDC or SMD drive, you are prompted for the disk partitions. Existing partitions are listed first, and you must decide whether to retain them:

```
Partition table
60     144    410    948
Do you wish to retain existing partitions (Y/N) <Y> ?
Do you wish to declare disk partitions (Y/N) <N> ?
```

Refer to the Cromix-Plus Administrator's Guide for the appropriate values for your particular drive.

After declaring disk partitions, the current alternate track table is displayed as shown below. Make a copy of the list of alternate tracks and keep it in a safe place. If no alternate tracks have been declared, a message to that effect is displayed.

```
Existing alternate tracks:

Alternate track table
  # bad track alt track      # bad track alt track      # bad track alt track
     hd   cyl  hd   cyl         hd   cyl  hd   cyl         hd   cyl  hd   cyl
  0  0     53  2    410      1  0     65  3    407      2  0    143  4    407
  3  0    144  5    407      4  0    145  0    408      5  0    146  1    408
  6  0    147  2    408      7  0    241  3    410      8  0    300  3    408
  9  0    368  4    408     10  0    614  4    410     11  0    615  5    408
 12  0    616  5    410     13  1    183  0    411     14  1    308  1    411
 15  1    395  2    411     16  1    612  3    411     17  1    619  4    411
 18  1    627  5    411     19  1    662  0    409     20  1    665  1    409
 21  1    681  2    409     22  1    705  0    412     23  1    707  1    412
 24  2    101  3    409     25  2    391  4    409     26  2    663  2    412
 27  2    774  3    412     28  2    786  5    409     29  3     73  4    412
 30  3    751  5    412     31  4    151  0    410     32  4    152  1    410
 33  4    406  0    413     34  5     35  1    413     35  5    269  2    407
 36  5    418  0    407     37  5    460  1    407

      Do you wish to retain existing alternate tracks (Y/N) <Y> ?

      Do you wish to declare alternate tracks (Y/N) <N> ?
```

Answer "Y" if any of the tracks reported as "disk errors" during formatting are not in the alternate track table. Enter the head and cylinder numbers of each bad track in response to the following prompts:

```
Enter bad track:
    Enter head - RETURN to stop (0-6):
    Enter cylinder (0-713):
```

After entering all bad tracks, press RETURN until you see the prompt:

```
Do you wish to edit alternate track table (Y/N) <N> ?
```

Enter "Y" to edit the alternate track table. When you are finished editing, you will see the following:

```
Alternate tracks require 10 cylinders
Alternate tracks located at cylinder 407
Do you wish to change this location (Y/N) <N> ?
```

Enter "Y" to specify a new location for the alternate tracks. CAUTION: Any data in the current alternate track locations will be lost if the location is changed.

Table 3-1 lists the important parameters for standard Cromemco STDC hard disk drives.

Table 3-1: STDC HARD DISK DRIVE PARAMETERS

Drive	Number of Heads	Number of Cylinders	Write Precomp Starting Cyl.
Vertex V150	5	987	987
Micropolis 1304	6	830	830
CDC	6	925	0
Hitachi	7	714	256
IMI (HD21)	6	306	214
Maxtor XT2140	11	1225	1225
Maxtor XT1140	15	918	918

utility:	**INITTAPE**
purpose:	This program initializes a floppy tape.
user access:	privileged user
summary:	inittape [[-cdfvx] [-p #] [-s #[,#] devname]]
arguments:	Cromix device name
options:	-c check mode (verify tape only) -d detached mode -f fast mode (no retension delay) -p number of verification passes -s STREAMS to initialize -v verbose -x examine VTOC only

Description

The Inittape program is used to initialize cartridge tapes. Parameters may be passed from the command line or entered interactively as the program executes.

Options

The **-d** option skips the prompt for editing the error map.

The **-c** option verifies the tape and displays the VTOC (bad sector map), but skips initialization.

The **-f** option defeats the tape retension delay, but using it while the tape is being retensioned results in a device open error.

The **-p** option, followed by a number, specifies the number of verification passes (the default is 3).

The **-s** option, followed by a single number (0 through 5), formats the stream number specified; using two numbers separated by a comma formats a range of streams from the first stream through the second.

The **-x** option skips initialization and displays only the VTOC.

The **-v** option displays progress reports of the initialization when Inittape is run from the command line.

Example

A typical interactive session to format a cartridge tape is shown below (user responses are boldfaced). Press RETURN to select the default response (displayed in angle brackets) of any prompt. To begin, enter **inittape** (as a privileged user) and press RETURN.

```
Inittape   version xx.xx

Press:     RETURN to supply default answers
           CTRL-C to abort program
Warning:   inittape will destroy all disk data

Do you wish retension delay of 180 seconds (Y/N) <N> ?  RETURN
Device name?   ftcd
Do you wish to examine VTOC only (Y/N) <N> ?  RETURN
Do you wish to VERIFY tape only (Y/N) <N> ?  RETURN
Number of VERIFY passes (0-10) <3> ?  3
```

All tape cartridges are retensioned (run from beginning to end and back to the beginning) when first placed in the drive. If the retension cycle is over (the drive light is out), press RETURN for the first prompt; otherwise enter "Y" and Inittape will sleep for 3 minutes.

The next prompt is for device name of the tape drive. Enter the device name of the drive connected to your system, such as **ftab** or **ftcd** (do not use subsection names such as **ftcd0**). Be sure to specify the device name correctly, as initialization destroys all data on the tape. Entering a device name while the tape is being retensioned results in a device open error and a repeated prompt for the device name.

Enter "Y" to the next prompt to skip initialization and display the VTOC (bad sector map). Enter "Y" to the "VERIFY tape only" prompt to verify the tape's current formatting (initialization is skipped).

The last prompt above selects the number of verification passes (the default is 3). To verify a tape, select at least one pass; to initialize a tape without generating a VTOC, select 0 passes.

```
Formatting
First stream (0-5.)?  <O>   RETURN
Last stream (0-5.)?   <5.>  RETURN
```

Inittape prompts for the numbers of the first and last stream to be formatted. On occasion, you may want to format a portion of a tape (e.g., when a single stream has frequent errors). Press RETURN for both prompts to format the entire tape. When inittape is called with a device name as an argument, the responses to the preceding prompts are taken from the command-line options. If no options are specified, all streams are selected by default.

The stream numbers are displayed as the tape is formatted (the entire stream is formatted in streaming mode, with interrupts disabled).

```
Initializing stream 0
        .
        .
        .
Initializing stream 5
Verify pass 1 on stream 0
        .
        .
        .
Verify pass 3 on stream 5

********************  ERROR MAP  ********************

Stream Segment Sectors

     0        6     1#   2*   3
Do you wish to edit error map (Y/N) <N> ?  y
Command (Add, Delete, End) ?  d
     Enter side,segment,sector ?  0,6,3
Command (Add, Delete, End) ?  e

********************  VTOC  ********************

Stream  Segment  Sectors

     0        6     1    2
```

After the tape is formatted and/or verified, the list of bad sectors is printed. Each sector number is followed by a '#' (sector bad on all passes), a '*' (sector bad on more than one, but not all, passes), or a space (sector bad on only one pass).

Inittape allows you to add or delete entries in the error map. To delete or add a sector, enter the side (stream), segment, and sector numbers separated by commas. Any field (side, segment, or sector) entered as a negative number means all valid entries for that field.

utility:	**IOPLOAD**
purpose:	This program loads a program into an IOP
user access:	privileged user
summary:	iopload filename device
arguments:	filename of program to be loaded
	IO device name (io1 .. io4)
options:	none

Description

The Iopload utility loads a file into an IOP. This utility is normally used to load the IOP/Quadart driver **/etc/quadart.iop** into an IOP. Iopload is located in the **/etc/** directory.

utility:	**INPUT**
purpose:	This program reads a line from standard input and writes it to standard output.
user access:	all users
summary:	input
arguments:	none
options:	none

Description

The Input utility reads a string from the standard input and, upon reading a newline character, writes that string to the standard output. This utility can be used to write interactive command language programs by redirecting its output to a file, and then testing the contents of the file with the Testinp utility.

Input stops reading when it reads a zero byte, a carriage return, or a line feed character. The terminating character is not written to STDOUT.

Example:

```
% input > temp
```

The above command reads one line from the standard input and writes it to the file "temp."

Shell command:	**KILL**
purpose:	This command sends a signal to a process.
user access:	all users
summary:	kill [-12345678] [PID #s]
arguments:	process id
options:	-1 abort
	-2 user
	-3 kill
	-4 terminate (default)
	-5 alarm
	-6 broken pipe
	-7 modem hang up
	-8 reserved

Description

The Kill command sends a signal to the process specified. If the signal type is unspecified, Kill sends a terminate signal. When a signal is sent to process 0, the signal is also sent to all processes initiated from the user's terminal.

If the user is a privileged user, and a user signal is sent to process 1 (**kill -2 1**), system shutdown is initiated.

Kill 0 aborts all background jobs attached to the user's terminal.

Kill -1 1 consults the **/etc/ttys** file and allows any terminals that have been added to be logged on. It also logs off any terminals that have been deleted from the file.

Options

The **-1** option causes an abort signal to be sent to the process. This option has the same effect as CONTROL-C from the keyboard, and aborts only interactive programs. Detached processes continue unaffected.

The **-2** option sends a user signal to the process. It is generated by a character typed at the terminal. The character that generates the signal is determined by the mode.

The **-3** option sends a kill signal to the process. This kill signal cannot be ignored or trapped. It is typically used to abort a program caught in an infinite loop.

The **-4** option sends a terminate signal to the process. The terminate signal kills both interactive and background processes. This is the default type of signal sent by the Kill command.

The **-5** option sends an alarm signal to the process.

The **-6** option is sent by the operating system when a pipe is used improperly.

The **-7** option is sent by the characters drivers if a modem detects a hangup condition.

The **-8** option is reserved for future use.

utility:	**LINK68**
purpose:	This program links **.o68** files.
user access:	all users
summary:	link68 [options] [-o outname] filename . . . filename [-s libname] . . .
arguments:	one or more filenames optional library name, preceded by **-s**

options:

-a0	value of A0 register
-a1	value of A1 register
-a2	value of A2 register
-a3	value of A3 register
-a4	value of A4 register
-a5	value of A5 register
-a6	value of A6 register
-a7	value of A7 register
-n	no map
-o	output file name
-q	do not display map
-s	search library
-x	lsect alignment size
-y	unformatted output
-z	memory size

Description

The Link68 utility is a two-pass virtual linker. From one or more input files, Link68 generates an executable binary file with a **.bin** filename extension. Link68 can be used to generate absolute images of standalone programs, such as the Cromix-Plus Operating System.

Options

The **-an** option, where 0 <= n <= 7, specifies the address to be loaded into the An register. Link68 uses this address to generate the A register with offset types of effective addresses. This option should be used only if the **.o68** files being linked were generated by nonstandard programs.

The **-n** option prevents creation of a link map. If **-n** is not specified, a link map is created and written to a file with the filename extension **.map.**

The **-o** option, with a filename as an argument, specifies the output filename. If **-o** is not specified, the output file has the name of the first relocatable file specified on the command line, with the filename extension **.bin.**

The **-q** option inhibits display of the link map on the terminal. If **-q** is not specified, the link map is displayed.

The **-s** option, with a library filename as an argument, specifies a library to be searched. To search more than one library, use multiple **-s** options. Link68 searches the **.o68** file for necessary functions.

The **-x** option, with a hex number as an argument, defines the padding size of each linked lsect. The size of each lsect will be a multiple of this padding size. If **-x** is not specified, the value 4096 is used.

The **-y** option, with a hex number as an argument, specifies the starting address of an absolute image. When **-y** is specified, the generated output file has not internal structure. **cromix.sys** is an example of a file linked in this way.

The **-z** option defines the amount of free memory to be added to the end of the binary file. This memory can be used for stack or any other purpose.

utility:	**LOGERR**
purpose:	This program accumulates errors detected and reported by the Ecc program.
user access:	privileged user
summary:	logerr [#]
arguments:	Optional time period in seconds
options:	none

Description

If called without an argument, Logerr displays the contents of the file that accumulates errors reported by the Ecc program. If Logerr is called with an argument, the argument defines how often the Ecc program should scan the error-correction hardware and append any errors to a file for later display by Logerr.

Example:

```
# logerr 30
```

The Ecc program will scan the error-correcting hardware every 30 seconds.

The following example is a typical display when Logerr is called without argument.

```
# logerr
*ECC started Jul-21-1984 16:45:22
 Single bit error - MCU 1, Card 0c, Row 3, Column 18
 Single bit error - MCU 1, Card 0d, Row 1, Column 18
```

utility:	**L**
purpose:	This program lists directory or file information in the old Cromix-20 style.
user access:	all users
summary:	l [-adrst] [file-list]
arguments:	optional file or directory pathname(s)
options:	-a all -d directory information -r reverse order -s summary -t time modified

Description

The L utility lists directory or file information in alphabetical order. If no pathname is specified, L lists the contents of the current directory. If a directory pathname is given, the contents of that directory are listed. If a file pathname is given, information about that file is listed.

Options

The **-a** option lists the names of all files, including invisible files (those files whose names begin with a period).

The **-d** option lists information about the directory, rather than the contents of the directory.

The **-r** option performs the sort specified in reverse order. Thus, an alphabetical listing is given in reverse alphabetical order, and a time-date listing is listed most recent file first.

The **-s** option generates a summary of listed files.

The **-t** option sorts the file list in order of time last modified. This order is from oldest to most recent unless the **-r** option is used.

Notes

This utility is a simple command file, which uses the Ls utility to produce the result. Refer to the description of the Ls utility.

utility: **LS** (List)
purpose: This program lists directory or file information.

user access: all users

summary: ls [-abdeilmrst] [file-list]

arguments: optional file or directory pathname(s)

options:

-a	all
-b	brief
-d	directory information
-e	everything
-i	inode number
-l	long list
-m	medium list
-r	reverse order
-s	summary
-t	time modified

Description

The Ls program lists directory or file information in alphabetical order. If no pathname is specified, it lists the contents of the current directory. If a directory pathname is given, the contents of that directory are listed. If a file pathname is given, information about that file is listed.

Options

The **-a** option lists the names of all files, including invisible files (those files whose names begin with a period).

The **-b** option makes a brief list, which contains only filenames.

The **-d** option lists information about the directory, rather than the contents of the directory.

The **-e** option lists everything about a file.

The **-i** option lists an inode number, rather than the file size.

The **-l** (long) option makes a long list of information. This option does not display as much information as the **-e** option.

The **-m** option makes a medium list (more information than **-b**, less than **-l**), containing file size, number of links, and name.

The **-r** option performs the sort specified in reverse order. Thus, an alphabetical listing is given in reverse alphabetical order, and a time-date listing is listed most recent file first.

The **-s** option generates a summary of disk space used.

The **-t** option sorts the file list in order of time-last-modified. This order is from oldest to most recent unless the **-r** option is used.

Notes

The meaning of the file-size information displayed by the Ls utility is as follows. If the file listed is a regular (data) file, the number associated with the file is its size in bytes (or number of characters). If the file is a directory, the number is the number of files stored in that directory. If the file is a device file, the numbers are the major and minor device numbers.

When options are combined on the command line, the most extensive option prevails.

Example:

Samples of the output from the Ls utility follow. Each is preceded by a note as to the option utilized. Without options, Ls displays a sorted, columnar list of filenames.

The following shows an output of Ls with the **-b** option, containing only filenames.

```
apa
apa1
apb
apc
apd
ape
```

The following shows an output of Ls using the **-m** option. For a filename, the field on the extreme left contains the number of bytes in the file. This is followed by the number of links to the file, and the filename. If the entry represents a directory, as in the first entry shown, the leftmost number shows the number of files in the directory. The D indicates it is a directory. The last two fields show the number of links and the directory name.

```
    3 D    1 cromix.doc
1,559      1 default.fm0
```

A sample of the output of **Ls** using the **-e** option is shown below. This is the most complete display. The name of each file in the directory is displayed on the extreme left. To the right, on the same line, is the number of bytes in the file. The first column of the remaining lines lists the operations performed on the file: created, modified, accessed, or dumped. To the right of each operation is the date and time the operation was last performed.

The rightmost column contains additional information. At the top the read, execute, write and access privileges for the owner, group, and all other users are shown. The second line is the login name of the file owner. The third entry lists the number of links to the file, and the final entry is the inode number.

To the extreme right of the owner's login name is an entry showing the group name of the user: in this case, **pubsl**.

```
Directory:  cromix.doc
locktest                                9   directory
          created:   Oct-21-1984 13:56:57   rewa re-- re--
          modified:  Oct-21-1984 13:56:57   karen           pubsl
          accessed:  Nov-19-1984 12:49:41   links:  1
          dumped:    000-00-1900 00:00:00   inode:  734

pipetest                               10   directory
          created:   Oct-21-1984 13:56:13   rewa re-- re--
          modified:  Oct-21-1984 13:56:13   karen           pubsl
          accessed:  Nov-19-1984 12:49:33   links:  1
          dumped:    000-00-1900 00:00:00   inode:  781

system.c                            1,641
          created:   Oct-21-1984 13:56:10   rewa re-- re--
          modified:  Oct-21-1984 13:56:11   karen           pubsl
          accessed:  Nov-31-1984 12:17:05   links:  2
          dumped:    000-00-1900 00:00:00   inode:  782
```

The following example shows the Ls program output using the **-i**, or inode number, option. This display first shows the directory name. The names of all files and directories within the subject directory are listed on the right. The inode number associated with the file is shown to the left.

```
Directory: cromix.doc
           734  locktest
           781  pipetest
           782  system.c
```

The following example shows the Ls program output using the **-l** option. If the second field in the entry is a **D**, for directory, the leftmost field indicates the number of files in that directory. If the second field is blank, the entry is a file, and the leftmost field shows the number of bytes in the file. Moving to the right, the third field indicates the number of links to the file or directory.

The next field shows the read, execute, write, and append access of the directory or file for the owner, group, and all other users, in that order. Immediately to the right of the access privileges is the login name of the owner. The three rightmost fields in this format are the most recent date and time of file access, and the file or directory name.

```
Directory: cromix.doc
        9 D  1 rewa re-- re-- karen           Oct-21 13:56   locktest
       10 D  1 rewa re-- re-- karen           Oct-21 13:56   pipetest
    1,641    2 rewa re-- re-- karen           Oct-21 13:56   system.c
```

The following is a sample of Ls program output using the **-s** and **-m** options. This display is the same as that obtained using the **-m** option, except that the last line of the display is a summary showing, from left to right, the number of files, number of blocks, and total bytes in the directory.

```
Directory: cromix.doc
        9 D  1 locktest
       10 D  1 pipetest
    1,641    2 system.c
3 files      6 blocks         2,313 bytes
```

What follows is a sample of Ls program output using the **-t** and **-m** options. These files are listed in order of the time last modified.

```
Directory: cromix.doc
    1,641    2 system.c
       10 D  1 pipetest
        9 D  1 locktest
```

utility:	**MAIL**
purpose:	This program sends or displays mail.
user access:	all users
summary:	mail [-agnvy] [user-name]
arguments:	optional list of user names **or** optional list of group names
options:	-a all -g group -n do not save mail -v verbose -y save mail

Description

Given without arguments, Mail displays mail sent to the user. After the mail is displayed, the Mail utility asks whether the user wants to save the mail. Saved mail is appended to the file **mbox** in the current directory.

Given with one or more user names as arguments, the Mail utility sends mail to one or more users. To send mail, enter the message after pressing RETURN at the end of the command line. A **CONTROL-Z** terminates the message and returns control to the Shell. In order to send mail, a user must have write and append access to the current directory, since mail creates a temporary file **mail.temp**.

Options

The **-a** option sends mail to all users. The list of users for this option is obtained from the **/etc/passwd** file.

The **-g** option sends mail to members of a specified group(s). Group members are defined in the **/etc/group** file.

The **-n** option causes mail <u>not</u> to be saved.

The **-v** option displays the list of users who received mail.

The **-y** option saves mail.

Notes

Upon logging in to the system, a user is informed if there is mail.

utility:	**MAKDEV**
purpose:	This program creates a device file.
user access:	all users
summary:	makdev [-c] devname b/c majornum minornum
arguments:	device name
	block or character device specification
	major device number
	minor device number
options:	-c conditional

Description

The Makdev utility associates a device drive number with a name. After the program is executed, references to the device name refer to the device indicated by the device number.

Options

The **-c** option displays an error message if no device driver corresponds to the specified device number.

Notes

Makdev calls for two numbers in its arguments: a major device number, which is the driver number, and a minor device number, which is the device number.

Some utilities demand that certain devices be owned by **bin.** For example, Spool expects the print devices to be owned by **bin.** Use the Chowner utility to change device ownership as needed.

Shell command:	**MAKDIR or MAKD**
purpose:	This command creates a directory file.
user access:	all users
summary:	makdir dir1 [... dirN]
arguments:	directory pathname(s)
options:	none

Description

The Makdir command creates directory files.

utility: **MAKE**
purpose: This utility automates the construction of executable programs from separate modules.

user access: all users

summary: make [-vdf] makefile [arg1 arg2 ...]

arguments: instruction file with possible arguments

options:
-v verbose
-d debug
-f force

Description

Most complex programs are constructed from a number of separate modules. Make provides the means for automatically executing the necessary steps (i.e., compilations, assemblies, linkages) to construct a finished program. It also provides selective execution of just those steps necessitated by the modification of any of the constituent files.

Since the actions of Make are predicated on date and time, it is very important to keep the system time and date reasonably accurate.

Make takes its instructions from a text file, which must have the suffix **.mak**. The **.mak** file consists of two kinds of lines:

> conditional statements--those with a colon somewhere in the line.
> commands--those without a colon.

Make scans each conditional statement line. If any of the files following the colon have been modified later than any of the files preceding the colon, the commands on the command lines following the conditional statement are executed.

The following is an example based on the Cromemco 68000 FORTRAN-77 environment:

```
prog1.obj:        prog1.for progdata.for
        fortran prog
        code prog1.i prog1.obj
prog2.obj:        prog2.for progdata.for
        fortran prog2
        code prog2.i prog2.obj
prog.bin:         prog1.obj prog2.obj /usr/lib/ftnlib.obj
        crolinker -oprog prog1 prog2 /usr/lib/ftnlib
```

The preceding instructions tell Make which parts of the program construction should be executed if any of the constituent files have been modified.

Multiple files on either side of the conditional colon are permitted.

```
prog2.obj:          prog2.for progdata.for
```

is equivalent to:

```
prog2.obj:          prog2.for
prog2.obj:          progdata.for
```

and

```
foo.obj: foo.for /usr/lib/ftnlib.obj
bar.obj: foo.for /usr/lib/ftnlib.obj
```

is equivalent to:

```
foo.obj bar.obj:    foo.for /usr/lib/ftnlib.obj
```

If the files to the right of the colon have not been modified more recently than those to the left of the colon, none of the commands are executed, and the Make utility scans for the next conditional statement.

The commands before the first condition line are executed unconditionally. Also, a line with only a colon forces the following commands to be executed unconditionally.

The Make utility allows the use of the DIRectory command to change the current directory. Also, arguments to the Make utility can be referenced anywhere in the **.mak** file in a shell-like fashion. The name of the make file is #0.

Options

The **-v** option will display the program's progress.

The **-d** option will display the times that Make used for its conditional comparisons.

The **-f** option will cause all actions to be taken regardless of time consideration.

utility:	**MAKFS**
purpose:	This program sets up the structure for a file system on disk.
user access:	privileged user
summary:	makfs [-b #] [-i #] [-r #] [-s #] devname(s)
arguments:	device name

options:	-b #	first inode block number
	-i #	number of inodes
	-r	restore Superblock
	-s #	number of blocks used

Description

The Makfs utility sets up a structure for a file system on block devices. It establishes the number of inodes, the blocks dedicated to those inodes, blocks dedicated to the system, and blocks dedicated to the user.

Makfs is run on all floppy disks and on some hard disks before the disk is mounted for the first time.

The Makfs utility destroys any existing data on the device. It warns and prompts the user before destroying data.

The Makfs utility stores the inode number in all of the inodes created.

Options

The **-r** option restores the Superblock should it be accidentally destroyed. Use this option with caution. If you previously ran Makfs with the **-i**, **-s**, or **-b** option, give the **-r** option with the same argument given **-i**, **-s**, or **-b**. **Failure to do so will destroy the file structure.**

After running Makfs with the **-r** option, run Icheck to complete the restoration process.

The **-b** option specifies which block should be the first to contain inodes. Except for block 1 (Super block), preceding blocks are not used. The argument given **-b** cannot be less than 2.

If **-b** is not used, block 20 is selected by default, with blocks 0 and 2 through 19 allocated as follows: boot program (0, 2-17), partition table (18), alternate track table (19).

The **-i** option establishes a file system with a nonstandard number of inodes. This option is used only if you need more files than the default allows. Otherwise, Makfs decides how many inodes are needed and uses that number.

Makfs rounds the number of inodes specified **down** to the nearest multiple of four (the number of inodes in a block).

The **-s** option specifies the number of blocks, from the beginning, to be used for the file structure. Blocks beyond the point specified by the argument to **-s** are not used.

If **-s** is not used, the maximum number of blocks is selected by default.

Notes

In lieu of running Makfs with the **-r** option, a more prudent method of restoring the Superblock is to use the Fixsb utility. Fixsb restores the Superblock and then runs Icheck automatically.

utility:	**MAKLINK**
purpose:	This program creates another name for an existing file.
user access:	all users
summary:	maklink [-fv] file-list dirname [-fv] source-file destination-file
arguments:	filenames followed by a directory name **or** source file followed by destination file
options:	-f force -v verbose

Description

Every file you create has one link from its pathname to an inode. Thus, the Cromix-Plus system can access that file when you specify its pathname. The Maklink program creates additional links. In effect, Maklink creates another name (or pathname) for an existing file.

Options

The **-f** option causes the new link to overwrite another file with the same pathname if one exists. If the **-f** option is not used, and another file exists with the same name, an error results and Maklink is aborted.

The **-v** option displays the names of files as they are being linked.

Notes

No link is possible between two different file systems. That is, links cannot extend between two different devices (disks).

utility:	**MATCH**
purpose:	This program finds all occurrences of a string within a file.
user access:	all users
summary:	match [-bcelqr] string file-list
arguments:	string file list
options:	

-b	block numbers
-c	count
-e	exact match
-l	line number
-q	quiet
-r	reverse match

Description

The Match utility searches through the specified files for all occurrences of the string and displays each line containing a match. Unless the **-e** option is used, Match is not case sensitive. If no file is specified, input is accepted from the standard input device.

Options

The **-b** option displays the block number with the matching line.

The **-c** option prints a count of the matching lines. The lines themselves are not displayed.

The **-e** option displays only lines that match the given string exactly - a case sensitive match.

The **-l** option displays the line number together with the matching line.

The **-q** option does not display filenames where no match occurred.

The **-r** option reverses the sense of the match, displaying only lines that do **not** contain a match to the given string.

Notes

Strings of more than one word and ambiguous strings may be specified on the command line, surrounded by quotation marks. The same characters represent ambiguous strings as are used by the Cromix Shell (*, ?, [ ]).

In addition, \n may be specified at the beginning or end of a string to force the match of that string at the beginning or end of a line of text, respectively. The search for the string is case insensitive unless the **-e** option is used. If the ambiguous characters * or ? are used, the string should be enclosed in quotation marks (").

If Match is used to search a file that is not a text file, control characters may be sent to the terminal. This may lock up the terminal; if you are using a Cromemco 3102 terminal, press CONTROL-Reset, or turn the terminal OFF and then ON again to restore terminal operation.

Example:

```
% who
  john tty1
  roger tty2

% who|match roger

  roger tty2
```

utility:	**MODE**
purpose:	This program displays or alters the device modes.
user access:	all users
summary:	mode [devname] [characteristic(s)]
arguments:	optional device name optional characteristic(s)
options:	-v verify

Description

The Mode utility displays or alters the operational characteristics (or modes) of a device. If Mode is run without arguments, the modes of the device that issued the Mode command are displayed (e.g., your terminal).

To display the characteristics of another device, specify the device name as the first argument (you must have read access to the device, else a "File not accessible" error occurs). If no modes are specified, Mode displays the characteristics of the selected device without changing them.

Device modes can be altered by specifying the desired settings as arguments (you must have execute access to the device, else a "Channel access error" occurs). For example:

```
# mode lpt1 width 132 -tabexpand
```

Some characteristics can be turned on by simply stating the appropriate name, and turned off by preceding the name with a dash (e.g., **-tabexpand**). Other characteristics must be followed by a number (e.g., **width 132**). All numbers are assumed to be decimal unless followed by an "h" for hexadecimal (e.g., **delaycode 7fh**).

Some characteristics use ASCII characters as values. A character value may be entered by pressing the ASCII key, or by typing its hexadecimal value. Control characters may also be entered by pressing the caret key (^) followed by the character. For example, the default line kill character on terminal devices (CONTROL-U) can be changed to CONTROL-A in any one of three ways:

1. Type **mode lkill**, then hold down the CONTROL key and press **A**
2. Type **mode lkill 01h**
3. Type **mode lkill ^A**

All commands are entered by pressing the RETURN key. Using method 2 or 3 you can make RETURN the user-signal key (**mode sigchar 0dh** or **mode sigchar ^M**).

When displayed, the first part of each mode name is capitalized (e.g., PAuse). The capitalized part of the name must be used when changing that characteristic (e.g., both **mode tty1 -pa** and **mode tty1 -pause** turn off the pause mode of tty1).

Option

The -v option displays the new mode settings after you change them.

Notes

In CBREAK, RAW, and BINARY modes, calls that read characters (.rdbyte, .rdline, nor .rdseq) do not wait for a line terminator; they return after a single byte is entered. The Shell sets the mode to nonCBREAK, nonRAW, and nonBINARY each time it prompts for a new command line. A program, PROG, can be run in BINARY mode, by typing

```
% mode binary; prog
```

The function keys on a 3102 terminal are disabled when the Cromix-Plus Operating System is booted up. The mode command **mode fn** enables the function keys (**mode -fn** disables them again). Each function key sends a 2-byte sequence (CONTROL-B (^B) and some other character) to the system. For example, CONTROL-B and "p" are sent when function key 1 is pressed. These 2-byte sequences can be used by applications programs to perform special functions.

The following table shows the available modes for character devices. The modes are then described in detail, followed by a similar discussion for block devices.

CHARACTER DEVICES

	tty	qtty	sysdev	timer	lpt	qslpt	ffp	cent	typ	slpt	tape	sec
ABortenable	+	+	-	-	-	-	-	-	-	-	-	+
Baud	+	+	-	-	-	+	-	-	-	+	-	+
BINary	+	+	-	-	-	-	-	-	-	-	-	+
BLKsize	-	-	-	-	-	-	-	-	-	+	-	-
BLKSwrt	-	-	-	-	-	-	-	-	-	-	+	-
	1	2	3	4	5	6	7	8	9	10	11	12

	1	2	3	4							11	12
Block	-	-	-	-	-	-	-	-	-	-	+	-
BMargin	+	+	-	-	+	+	-	-	+	+	-	-
CBreak	+	+	-	-	-	-	-	-	-	-	-	+
CHECK	-	-	-	-	-	-	-	-	+	-	-	-
Correction	-	-	-	+	-	-	-	-	-	-	-	-
Corr error	-	-	-	-	-	-	-	-	-	-	+	-
CRDEVice	+	+	-	-	+	+	-	-	+	+	-	+
CRIGNore	-	-	-	-	-	-	-	-	-	+	-	-
CWidth	-	-	-	-	-	-	-	-	+	-	-	-
DELAY	+	+	-	-	+	+	-	-	-	+	-	+
DELECho	+	+	-	-	-	-	-	-	-	-	-	+
DIScard	+	+	-	-	+	+	-	-	+	+	-	+
ECho	+	+	-	-	-	-	-	-	-	-	-	+
End of tape	-	-	-	-	-	-	-	-	-	-	+	-
EOFclose	-	-	-	-	-	-	-	-	-	-	+	-
Erase	+	+	-	-	-	-	-	-	-	-	-	+
ESCreturn	+	+	-	-	-	-	-	-	-	-	-	+
EVENparity	+	+	-	-	-	+	-	-	-	+	-	+
FFexpand	+	+	-	-	+	+	-	-	-	+	-	+
File	-	-	-	-	-	-	-	-	-	-	+	-
File mark	-	-	-	-	-	-	-	-	-	-	+	-
FMark	-	-	-	-	-	-	-	-	-	-	+	-
FNkeys	+	+	-	-	-	-	-	-	-	-	-	+
FORMs	-	-	-	-	+	-	-	-	+	+	-	-
Hard error	-	-	-	-	-	-	-	-	-	-	+	-
High speed	-	-	-	-	-	-	-	-	-	-	+	-
HUPenable	-	+	-	-	-	-	-	-	-	-	-	+
IBsize	-	-	-	-	-	-	-	-	-	-	+	-
IMmediateecho	+	+	-	-	-	-	-	-	-	-	-	+
LCase	+	+	-	-	-	-	-	-	-	-	-	+
Length	+	+	-	-	+	+	-	-	+	+	-	+
LHeight	-	-	-	-	-	-	-	-	+	-	-	-
LKill	+	+	-	-	-	-	-	-	-	-	-	+
LMargin	-	-	-	-	-	-	-	-	+	-	-	-
Load point	-	-	-	-	-	-	-	-	-	-	+	-
NLECho	+	+	-	-	-	-	-	-	-	-	-	+
OBsize	-	-	-	-	-	-	-	-	-	-	+	-
ODDparity	+	+	-	-	-	+	-	-	-	+	-	+
ON LINE	-	-	-	-	-	-	-	-	+	-	+	-
PAPER OUT	-	-	-	-	-	-	-	-	+	-	-	-
PAuse	+	+	-	-	-	-	-	-	-	-	-	+
PSthimble	-	-	-	-	-	-	-	-	+	-	-	-
RAW	+	+	-	-	-	-	-	-	-	-	-	+
READY	-	-	-	-	-	-	-	-	-	-	+	-
Rewind	-	-	-	-	-	-	-	-	-	-	+	-
Rewinding	-	-	-	-	-	-	-	-	-	-	+	-
RIBBON OUT	-	-	-	-	-	-	-	-	+	-	-	-
Secure	-	-	-	-	-	-	-	-	-	-	+	-
SIGAllchars	+	+	-	-	-	-	-	-	-	-	-	+
SIGChar	+	+	-	-	-	-	-	-	-	-	-	+
SIGenable	+	+	-	-	-	-	-	-	-	-	-	+
SIGHUPall	-	+	-	-	-	-	-	-	-	-	-	+
TABexpand	+	+	-	-	+	+	-	-	-	+	-	+

	1	2	3	4	5	6	7	8	9	10	11	12
TANdem	+	+	-	-	-	-	-	-	-	-	-	+
Unload	-	-	-	-	-	-	-	-	-	-	+	-
Width	+	+	-	-	+	+	-	-	+	+	-	+
WRAParound	+	+	-	-	+	+	-	-	+	+	-	+
Wrt protect	-	-	-	-	-	-	-	-	-	-	+	-

ABortenable

When the ABortenable switch is on (**mode ab**), pressing CONTROL-C sends a SIGABORT signal to all processes controlled by the terminal; when off (**mode -ab**), CONTROL-C is treated like any other character.

Baud

The Baud parameter, followed by a number, sets the baud rate of a serial device (e.g., **mode b 9600** changes the baud rate of your terminal to 9600).

The baud rates designated Auto, Nochg, and Ctswait are special cases. If Auto mode is used with a terminal, the driver tries different baud rates until it reads a RETURN from the input. If Nochg mode is used, the baud rate has been previously established (e.g., by RDOS). If Ctswait is used, the driver waits for a CTS signal from the modem.

BINary, CBreak, and RAW

If any of these parameters is enabled, any read from the device returns after each input character. These parameters disable the action of various other parameters, as shown in the following table (+ means that the parameter causes the given effect, a space means it does not.)

Effect	CBreak	RAW	BINary
Return after each character input	+	+	+
No erase, linekill, or EOF (CONTROL-Z) characters	+	+	+
No output PAuse or output Width truncation		+	+
Treat XOFF (CONTROL-S), XON (CONTROL-Q) as regular input		+	+
No tandem mode - no input buffer flow control			+
Treat CONTROL-C and SIGChar key as regular input			+
No checking or changing of input parity bit			+

No delays after any output control characters such as tabs			+
No echoing of input			+
No function key decoding			+
No character transformations - ignore the LCase, CREDEVice, and TABexpand settings			+

BLKsize

A serial printer can accept only a limited number of characters without some kind of handshaking. With the ETX/ACK protocol, the driver sends one block of characters followed by an ETX character and waits until the printer returns an ACK character before sending the next block. The number of characters in a block never exceeds BLKsize characters. If an escape sequence is detected, the current block is immediately terminated by an ETX character, thereby handling escape sequences correctly (provided no escape sequence exceeds BLKsize characters). Refer to your printer documentation for the value of BLKsize.

BLKSwrt

BLKSwrt is the number of blocks written to a tape device during the last write operation. It cannot be changed with the Mode utility.

Block

Block is the command for tape positioning. It should be followed by a space and the block number. Tape blocks are numbered 1, 2, 3, ..., up to the number of blocks in the file. If the specified block is larger than the total number of blocks in a file, the tape moves to the beginning of the next file.

BMargin

If a printer device is within BMargin lines of the bottom of the page, a formfeed moves the paper to the top of the next page. The length of a page is determined by the parameter Length (see below).

CBreak

See BINary.

CHECK

CHECK indicates a malfunction that can be resolved only by DIScard or a power OFF/ON sequence.

Correction

This is the number of seconds per 100 days to be added to or subtracted from the system timer.

Corr error

Corr error is a flag that is set when the tape formatter detects a correctable error. The flag cannot be set by the Mode utility.

CRDEVice

The CRDEVice switch must be on for a carriage return device and off for a newline device. If CRDEVice is on, a RETURN character read from the device is translated into a newline character before being passed to the calling program (and a RETURN, linefeed sequence is echoed to the device). On output, newlines are translated into RETURN, linefeed sequences. If CRDEVice is off, no translations are made.

The single newline character (0Ah) performs a RETURN, linefeed sequence.

CRIGNore

Some printers, such as the CLQ, need only a RETURN character (0Dh) to move to the start of a new line. If CRIGNore is on, the driver ignores all RETURN characters and translates newline characters into RETURN characters.

CWidth

The value of CWidth defines the width of each character in units of 1/120th of an inch. The default is 12, or 10 characters per inch.

DELAY

The DELAY is the decimal equivalent of a byte determining the amount of delay inserted after certain characters are sent to the output. For TTYs, the interrupt process is suspended for the time required to send a number of characters. For QTTYs, the interrupt process is suspended for some multiple of one-tenth of a second. The bit assignments for DELAY are:

Character	**DELAY Bits**	**QTTY Values (seconds)**	**TTY Values (chars)**
newline	0 and 1	0, .1, .2, .3	0, 4, 8, 12
tab	2 and 3	0, .1, .2, .3	0, 4, 8, 12
carriage return	4 and 5	0, .1, .2, .3	0, 4, 8, 12
formfeed	6	0, .8	0, 128
backspace	7	0, .1	0, 4

For example, **mode qtty1 delay a3h** sets the QTTY1 newline delay to 0.3 seconds, the RETURN delay to 0.2 seconds, the backspace delay to 0.1 seconds, and the TAB and formfeed delays to zero.

DELECho

On TTY, QTTY, and MTTY devices, the character following DELECho is echoed in response to any one of the delete characters. If the letter **R** follows DELECho, the echoed response to a delete character is backspace,space, backspace.

DIScard

When a driver is first used, mode settings are established in a data area for each device. Values are changed or not changed as the Mode calls dictate, and these values remain even if the device is closed. If DIScard is set, closing a device discards all information, and all tables are reinitialized when the device is next opened.

TTY and LPT drivers define the actual devices during system generation (refer to Crogen). To attach a parallel printer to a connector that was used for a TTY terminal, you must generate (and reboot) a properly configured Cromix-Plus Operating System.

ECho

If the ECho switch is on (no preceding dash), characters entered on the terminal are echoed to the screen. ECho cannot be changed by the Mode utility.

End of tape

End of tape is a flag describing the tape device status. It cannot be set by the Mode utility.

EOFclose

If EOFclose is set, the tape controller writes a double file mark when the device is closed.

Erase

The character following Erase serves as an extra erase character, along with DEL (7Fh) and CONTROL-H (08h, also referred to as backspace). For example, the command

```
% mode erase _
```

selects the underscore as an auxiliary delete character. Note that DEL and backspace still function as delete characters.

ESCreturn

If this switch is enabled, the ESCAPE character terminates the **.rdline** and **.rdseq** functions as if RETURN had been pressed.

EVenparity and ODDparity

ODDparity and EVenparity controls the handling of the parity bit, as follows (+ means enabled, - means disabled).

EVenparity	ODDparity	Function for Input Characters
-	-	does not check parity but strips parity bit
+	-	checks for even parity before stripping parity bit
-	+	checks for odd parity before stripping parity bit
+	+	leaves parity unchecked and unchanged

EVenparity	ODDparity	Function for Output Characters
-	-	strips parity bit
+	-	makes character even parity
-	+	makes character odd parity
+	+	leaves parity bit unchanged

FFexpand

If FFexpand is on, each formfeed character (0bh) sent to a printer device is converted to enough newlines to move the paper to the top of the next page. The length of a page is determined by the Length mode. If FFexpand is off, the formfeed character is sent without conversion.

File

On a tape device, a file number following the File mode moves the tape to the start of that file. Tape files are numbered from 1 through the number of files on the tape. The following command moves the tape to the sixth file on TP1:

```
mode tp1 file 6
```

If the specified file number is larger than the total number of files recorded on the tape, the tape moves to the end of the tape reel. This motion may be aborted by taking the tape drive off-line and entering CONTROL-C at the terminal keyboard.

File mark

File mark is a flag describing the tape device status. It cannot be set by the Mode utility.

FMark

FMark is a command to write the file mark on a tape device (e.g., **mode tp1 fm**).

FNkeys

If FNkeys is on, the driver echoes a CONTROL-B for each of the two bytes sent when a function key on the Cromemco 3102 terminal is pressed. This allows the 2-byte function key sequences to be passed to a program.

FORMs

A printer driver that understands the concept of paper forms can be told what paper form is loaded into the printer (or made to recognize some other physically distinct change). The printer daemon prints only those files that were spooled with the same forms number (refer to the **-z** option of the Spool utility.)

When the operator needs to change paper forms (print thimble, etc.), the FORMs number can be set to any unused value. Then, after the current file is printed, the operator can make the change, set any necessary modes, and select a new FORMs value. After the printer daemon is restarted, all files with the new FORMs value will be printed.

The actual FORMs numbers are left to the System Administrator.

Hard error

The Hard error flag is set on a tape device when the tape formatter finds an error that it cannot correct. This flag cannot be set by the Mode utility.

High speed

High speed is a flag describing the tape device status. It cannot be set by the Mode utility.

HUPenable

If this switch is on and an IOP/OCTART terminal device closes, the modem on the IOP device is hung up.

IBsize

IBsize defines the length in bytes of the first block of the last file read from a tape device. It cannot be changed by the Mode utility.

IMmediateecho

If IMmediateecho is on, characters typed ahead are echoed immediately, and echoed again when they are read. If IMmediateecho is off, characters are echoed only at the time they are read.

Length

This is the page length in lines of the designated device. When the Mode utility displays the page length, the word length is followed by the specified page length. To change the page length, use the argument length followed by a space and the desired page length.

LCase

If LCase is on, terminal devices TTY, QTTY, and MTTY convert uppercase alphabetic input characters to lowercase.

LHeight

The value of LHeight defines the height of a printed line in units of 1/48th of an inch. The default value is 8 (6 lines per inch).

LKill

The LKill character deletes the current input line for terminal drivers. This performs multiple deletes back to the last prompt character.

LMargin

The value of LMargin defines the first printable position, expressed in units of .1 inches (the default is 0).

Load point

Load point is a flag describing the tape device status. It cannot be set by the Mode utility.

NLECho

Determines if a newline character will be echoed.

OBsize

OBsize is a command to set the block length of files written to the tape device. Keyword OBsize should be followed by a blank and the desired size. The following command sets the output block length of TP3 to 4096 bytes:

```
mode tp3 ob 4096
```

ODDparity

See EVENparity.

ON LINE

Signifies the printer is on-line (i.e., powered on).

PAPER OUT

Signals the printer is out of paper.

PAuse

If PAuse is on, terminal devices pause after displaying the number of lines specified by Length. The output resumes only after an XON (CONTROL-Q) is entered on the keyboard.

PSthimble

When using the Typ driver to operate a "spinwriter" printer, the setting of PSthimble must agree with the print thimble in use. The default setting (-PSthimble) supports the normal (non-proportional) print thimble. If printed copy is unreadable (or readable, but erratically spaced), check the setting of PSthimble.

RAW

See BINary.

READY

READY is a flag describing the tape device status. It cannot be set by the Mode utility.

Rewind

Rewind is a command to rewind the tape device (e.g., **mode tp1 r**).

Rewinding

Rewinding is a flag describing the tape device status. It cannot be set by the Mode utility.

RIBBON OUT

Signals the printer is out of ribbon.

Secure

Secure is a command to erase the tape at high speed (e.g., **mode tp1 s**).

SIGenable, SIGChar, and SIGALLchars

If SIGenable is on and SIGALLchars is off, pressing the SIGChar key causes terminal devices to send a SIGUSER signal to all processes controlled by the terminal. The SIGChar key character is not put into the input stream. If SIGenable is off, then the SIGChar key is treated in the same manner as any other key.

The terminal that controls a process is the terminal on which the owner of the process logged in to the system.

If SIGenable and SIGALLchars are both on, pressing the SIGChar key causes the SIGUSER signal to be sent to all processes controlled by the terminal, but the SIGChar key character is also put into the input stream.

If SIGALLchars is on but SIGenable is off, every terminal key pressed before a system call to read input sends the SIGUSER signal to all controlled processes. (Only characters typed-ahead send signals.) The characters are also put into the input stream.

Note that shells are set up to ignore SIGUSER signals, so that a user is not logged off by them. Any program running in a nondetached mode that does not either ignore or trap SIGUSER signals is aborted by them. The **.signal** system call provides a means for ignoring or trapping signals.

SIGHUPall

If this switch is on and the modem of an IOP terminal device hangs up, the signal SIGHANGUP is sent to all processes controlled by the device. A process is controlled by the terminal from which the user who initiated the process logged in. For example, a user who has logged in on QTTY1 and hangs up without logging out is logged off by the resulting SIGHANGUP signal, provided SIGHUPall is enabled.

TABexpand

If TABexpand is on, every TAB character (09H) is converted to enough spaces to bring the output to the next standard TAB stop. Standard tab stops are multiples of 8 at columns 1, 9, 12, etc., on the terminal.

TANdem

Tandem mode allows a receiving Cromix system to control the rate of input data using the DC1/DC3 handshaking protocol. The device sending data may be a Cromix system or another computer. When used to communicate between two Cromix systems, the ttys on the sending and receiving systems should not be selected in the ttys files. Both drivers should be set to the same Baud rates, have RAW mode enabled, and ECho and CRDevice disabled.

The receiving system should have TANdem mode enabled, and the receiving program or command file should already be executing before sending begins. In TANdem mode, the receiving system sends a DC3 (XOFF) character when its tty driver buffer is full, and sends a DC1 (XON) character when the driver is ready to accept more characters.

Unload

Unload is a command to unload the tape device (e.g., **mode tp1 u**)>

Width

The Width function specifies the number of columns displayed before truncation or wrap-around. If Width = 0, no truncation or wrap-around occurs.

WRAParound

If WRAParound is on, and the device output column reaches the page Width, an extra newline is sent to the device. This allows the remainder of the output line to be printed on the next line. If WRAParound is off, the remainder of the line is truncated. If Width = 0, no truncation or wrap-around occurs.

Wrt protect

Wrt protect is a flag describing the tape device status. It cannot be set by the Mode utility.

BLOCK DEVICES

	cflop	uflop	allmem	stdc	ramdsk	tflop	smd	hd	amem
ADDress	-	-	-	-	+	-	-	-	-
BSTEP	-	-	-	+	-	-	-	-	-
CDOS	+	+	-	+	-	-	-	+	-
CROMIX	+	+	-	+	-	-	+	+	-
CYLinders	+	+	-	+	-	+	+	+	-
DDensity	+	+	-	-	-	-	-	-	-
DSide	+	+	-	-	-	-	-	-	-
Free	-	-	-	-	-	-	-	-	+
HARDerr	+	+	-	+	-	+	+	+	-
ICACHE	-	-	-	-	-	-	-	-	+
MFree	-	-	-	-	-	-	-	-	+
MOUNTed	+	+	-	+	+	-	+	+	-
MSys	-	-	-	-	-	-	-	-	+
MTot	-	-	-	-	-	-	-	-	+
READonly	+	+	-	+	+	-	+	+	-
RETENsion	-	-	-	-	-	+	-	-	-
RETRY	+	+	-	+	-	+	+	+	-
RPM	+	+	-	+	-	-	-	+	-
SECSIZ	+	+	-	+	-	+	+	+	-
SECtors	+	+	-	+	-	+	+	+	-
SIZE	+	+	-	+	+	+	+	+	-
SOFTerr	+	+	-	+	-	+	+	+	-
SURfaces	-	-	-	+	-	+	+	+	-
VERIFY	+	+	-	+	+	-	+	+	-

VERsion	+	+	-	+	+	+	+	+	-
VOICEcoil	+	+	-	-	-	-	-	-	-
WRITEprotect	+	+	-	+	-	+	+	+	-

ADDress

This value indicates, in hexadecimal notation, the starting address of the RAM disk device.

BSTEP

This switch indicates whether the driver is using buffered step mode.

CDOS

This switch is set if the device is a CDOS device.

CROMIX

This switch is set if the device is a Cromix device.

CYLinder

This value indicates, in decimal notation, how many cylinders are on the device.

DDensity

This switch is set if the device is a double-density device.

DSide

This switch is set if the device is a double-sided device.

Free

The Free value is the total amount of unused memory (not necessarily continuous).

HARDerr

This is the total number of hard errors reported on the raw console. To change the accumulated total number, use the argument **harderr** followed by a space and the desired total.

ICACHE

This flag describes the state of the 68020 on-chip cache memory. It can be turned on or off. Setting this switch is ineffective on the 68000 or 68010 processor.

MFree

The Mfree value is the remaining amount of continuous unused memory.

MOUNTed

This switch is set after the device has been mounted (**_mount** system call).

MSys

The amount of memory occupied by the operating system proper.

MTot

The total amount of memory the system is running on.

READonly

This switch is set if the device is mounted for read only.

RETENsion

This value indicates the number of tape repositions before a retension is performed on the floppy tape device. Setting this value to 0 (e.g., **mode ftcd reten 0**) disables the retension feature. Setting this value to a negative number causes an immediate retension, but no change to the old value.

RETRY

This value indicates the number of times a read operation will be retried.

RPM

RPM value defines the rotational speed of the device (in revolutions per minute).

SECSIZ

This field defines the number of bytes per sector.

SECtors

This field defines the number of sectors per track.

SIZE

SIZE defines the size of the device, in kilobytes.

SOFTerr

This is the total number of soft errors (retries). To change the accumulated total number, use the argument **softerr** followed by a space and the desired total.

SURfaces

This field defines the number of surfaces on the devices.

VERIFY

This switch is set if the device reads back the written data to check for errors.

VERsion

VERsion indicates the version number of the firmware (if applicable) or the version number of the iolib library where the driver was introduced. The iolib library itself may have a higher version number if other components of the library have been changed.

VOICEcoil

This switch is set if the device uses a voice coil positioning mechanism.

WRITEprotect

This switch is set if the device is physically write protected.

utility:	**MOUNT**
purpose:	This program enables access to a file system.
user access:	privileged users
summary:	mount [-r] [devname dummyname]
arguments:	device name file pathname
options:	-r read only

Description

The Mount utility enables access to a file system.

When called without arguments, Mount lists the currently mounted devices.

The Mount utility looks on the disk to be mounted for the file **/etc/passwd**. Finding that file, it looks for the special user name **mount.** If this name is present and has a password associated with it, Mount prompts the user for the password before mounting the disk. Thus, it is possible to protect disks from being mounted by an unauthorized user.

User Access

By editing the file **/etc/mutab**, a privileged user can allow selected, non-privileged users to mount specified devices. The Mount utility consults this file whenever a non-privileged user gives the Mount command.

Each line in the **mutab** has up to four fields, separated by colons:

```
flag : device : username : groupname
```

Flags can be zero (entry disabled) or one (entry enabled). The device field must be the name of a block device in the **/dev** directory. Username is the user allowed to mount that device. If a groupname is specified (field 4), all members of that group may mount the device. If username or groupname is not specified, all users may mount the device.

A typical **mutab** file is shown here:

```
0:fda
1:fdb  :  tom
1:fdb  :          :  users
```

Line 1: Because a username or groupname is not specified, any user can mount device **fda.** In the example, this entry is disabled by a zero flag in field 1.

Line 2: User **tom** may mount device **fdb**

Line 3: All members of the **users group** may mount device **fdb.**

Options

The **-r** option causes the file system to be mounted for read-only access.

Notes

A file system that has been mounted **must be unmounted** by use of the Unmount utility **before the mounted disk is removed from the system.** If this is not done, the integrity of the data on the mounted system cannot be assured.

Do not attempt to mount a file system on a nonexistent device. Devices which do not exist may be deleted from the **/dev** directory.

Example:

```
% create newfilesys
% mount fdb newfilesys
% ls -m
    .
    .
    .
145 D newfilesys

%
```

In the example above, the user first creates a dummy file. After mounting, the name of this dummy file becomes the root directory name of the file system to be mounted. After unmounting, this name becomes a dummy filename once again.

The Mount command is given with the device name where the file system is located. Refer to the Cromix-Plus System Administrator's Guide for a complete list of device names.

The Ls utility shows that the new file system has been mounted and gives the name of the root directory.

utility:	**MOVE**
purpose:	This program moves file(s) from one directory into another.
user access:	all users
summary:	move [-ftv] file-list dirname [-ftv] srcfile destfile
arguments:	two single file pathnames **or** one or more file pathnames **and** a directory pathname
options:	-f force -t time -v verbose

Description

The Move program moves one or more files from one directory to another directory. This program destroys the source file(s). The Move program does not change the access privileges of the moved files. If files are transported from directory A to directory B, the owner of directory B may not have full access privileges for the files. The program Chowner must be run to change the owner of these files.

Options

The **-f** option causes the moved file to overwrite another file with the same pathname if one exists. If this option is not used and another file exists with the destination pathname, an error is generated and the Move program aborted.

The **-t** option causes a file to be moved **only** if:

1. The file does not exist in the destination directory; or

2. The source file was modified more recently than the destination file. This comparison is performed on a file-by-file basis.

The **-v** option displays the names of the files being moved.

Notes

With the exception of "time dumped", which is automatically zeroed, files are moved with ownership and times preserved.

utility:	**MSG**
purpose:	This program sends messages between users.
user access:	all users
summary:	msg [-any2] [user-name or devname]
arguments:	text terminated by CONTROL-Z
options:	-a all -n disable -y enable -2 Cromemco 3102 terminal

Description

The Msg utility sends messages between users or from a user to a device. Sending a message to a device is useful when a device is online but no user is in attendance.

If **msg** is typed and immediately followed by a RETURN, then a message is displayed to inform the user of the status of incoming messages. Incoming messages may be disabled or enabled by using the **-n** and **-y** options. Terminating a message with CONTROL-Z automatically sends the message **End of message** to the receiving user.

The Msg command followed by (optionally the **-2** option and) a user or device name and RETURN allows a message to be entered. The message is transmitted to the destination user after each RETURN is pressed. A CONTROL-Z terminates the message and returns the originating user to the Shell.

Options

The **-a** option broadcasts a message to all users currently logged on to the system. This can be used by the privileged user to warn other users of interruptions to system usage such as rebooting. This message is sent to all users whether or not they have message receiving enabled. The message is preceded by the warning **Broadcast message.** Only privileged users are permitted to use this option. A message sent with the **-a** option is not transmitted until the entire message is given. Hence, when the **-a** option is specified, it may be followed on the command line by the name of a file that contains a broadcast message.

The **-n** option causes incoming messages to be disabled.

The **-y** option allows incoming messages to be received.

The **-2** option sends messages to the status line of a Cromemco 3102 terminal.

Notes

To clear the status line of a Cromemco 3102 terminal after receiving a message transmitted using the -2 option, type **CONTROL-shift** followed by **CONTROL-1.**

If two-way communication is desired, a protocol should be established to prevent the confusion that arises when two messages are transmitted simultaneously. A suggested protocol follows: One user transmits at a time. A single **o** (short for over) is transmitted on a line by itself to indicate the end of the message. Upon seeing the **o**, the other user responds, terminating the message with an **o**. When the entire communication is finished, one user transmits **oo** (short for over and out) followed by a **CONTROL-Z**. The other user should type a **CONTROL-Z** also.

Two-way communication can be established by the Msg utility. When a user receives a message:

```
Message from xxxx
```

the receiving user should type:

```
msg xxxx
```

This allows users to send each other messages. In the example above, **xxxx** represents a user name.

utility:	**NCHECK**
purpose:	This program displays file information.
user access:	all users
summary:	ncheck [-i inode-#'s] [dirname or filename]
arguments:	directory or file pathname
options:	-i list of inode numbers

Description

The Ncheck program displays the inode number, link count, and pathname of all files contained in the specified directory and all subdirectories. If no arguments are supplied, Ncheck uses the root directory.

Options

The **-i** option, which displays information about specified inodes only, requires an argument. This argument can be one or more inode numbers separated by commas or it can be one or more inode numbers separated by spaces with the entire list enclosed in quotation marks.

```
ncheck -i 637, 922

    922  Ø  /usr/[illegible]/99986e02.[illegible]

    637  Ø  /usr/[illegible]/99986e02.bak
```

utility:	**NEWUSER**
purpose:	This program displays information for new users.
user access:	all users
summary:	newuser
arguments:	none
options:	none

Description

The Newuser utility displays the file **newuser.msg**, which contains important information about the present version of the Cromix-Plus Operating System.

utility: **OCTLOAD**
purpose: This program loads a program into an Octart

user access: privileged user

summary: octload filename device

arguments: filename of program to be loaded

IO device name (io1 .. io4)

options: none

Description

The Octload utility loads a file into an Octart. This utility is normally used to load the Octart driver **/etc/octart.iop** into an Octart. Octload is located in the **/etc/** directory.

utility:	**PASSWD**	
purpose:	This program changes the passwd and group files.	
user access:	all users	
summary:	passwd [-dgn] [user1 user2...]	
arguments:	user1 user2...	
options:	-d	delete
	-g	group
	-n	new user

Description

The Passwd utility has three functions. It may be used to change a user's own password. A privileged user may use it to add and delete from the list of users permitted to log on to the system. By using the delete function followed by the add function, the privileged user may change the login status of any user.

In any one of these three modes of operation, user name(s) are specified either on the command line or during the execution of the Passwd program.

To change the password only, enter the command **passwd** followed by a RETURN. The Passwd program prompts for a user name and a new password.

Options

The **-d** option deletes a specified user or group.

The **-g** option alters the **/etc/group** file (instead of the **/etc/passwd** file).

The **-n** option adds new user(s) or group(s).

Establishing a New User

A new user may be added using the Passwd program. In the following example, the user logs on as the privileged user **system** and creates a new user **fred** with the password **mountain:**

```
Login: system

Logged in system Jun-24-1980 17:12:15 on console
# passwd -n

Name: fred
Password: xxx
User number: 5
Group number: 0
Directory: /usr/fred
Starting Program:

Name:
#
```

The Passwd program prompts for a user name. The response to this prompt is the user name typed in response to the **Login:** prompt. Press RETURN after entering the name.

Next, the program prompts for a user password. If no password is desired, press RETURN in response to the prompt. If you do enter a password, it is encrypted, and the encrypted password displayed on the screen. When a user logs on, this password must be entered after the password prompt.

The program prompts for the user and group identification numbers. Each of these is an unsigned integer between 0 and 65535. A zero in the user field indicates a privileged user. A zero in the group field indicates the user is not a member of any group. Any other number has significance only within a given system.

The **Directory:** prompt allows specification of an initial directory, which is the user's home directory. If this directory does not exist, the system creates one. The user is the owner of this directory. If the home directory already exists, the Passwd utility prints this information.

Finally, the Passwd program prompts for a **Starting program.** If RETURN is pressed in response to the prompt, the user has full use of the Shell program. If the name of a program is entered here, the user is brought up running the program specified and is logged off upon exiting from the program. Any valid Shell command line may be entered in response to this prompt.

Deleting a User

A user is deleted from the list of users (**/etc/passwd** file) by running the Passwd program with the **-d** option. In the following example, the user **fred** is deleted:

```
# passwd -d

Name: fred

Name: RETURN
#
```

Note that only a privileged user may delete a user.

Changing a Password

When called without any options, the Passwd program allows the privileged user to change any user's password and any user to change his or her own password. To change a password, call the Passwd program as follows:

```
% passwd
Name: fred
Password: xxx

Name:  RETURN
%
```

Notice that the password encryption is displayed only after the password and a RETURN have been entered.

Changing User Characteristics

If the privileged user has occasion to change user characteristics other than the password, the user must be deleted and added again with the new characteristics.

utility:	**PATCH**
purpose:	This program patches a file.
user access:	all users
summary:	patch [filename]
arguments:	optional filename
options:	none

Description

The Patch utility displays and alters specified bytes within a file. Enter the command name plus a filename and press RETURN. When you see a greater-than sign (>), enter any of the available commands: "d" for display, "e" for exit, "h" for calculate, "q" for query, "s" for substitute, "t" for truncate, or "z" for zap. If Patch is called without a filename, only the "h" and "e" commands are valid.

Generally, if you give a command without arguments Patch uses the arguments from the previous command. Start addresses and swath values can be arbitrary expressions (refer to the "h" command).

Display

The "d" command displays part of a file. The possible forms are:

d start S swath	Display swath bytes from start address.
d start	Display the same number of bytes from start
d S swath	Display swath bytes from where last d command ended.
d	Display the same number of bytes from where last d command ended.

If no start address is given, the d command wraps to the beginning of the file when the end of the file is reached.

Exit

The "e" command (no arguments) incorporates the changes from the current Patch session and exits the program. To exit from Patch without saving the changes, press CNTRL-C.

Calculate

The "h" command serves as a calculator, and has the form:

```
h expression
```

The syntax of expression can be described in BNF form as follows (::= means "defined as" and | means "or"):

```
<expression> ::= <term> | +<term> | -<term> |
        <expression>+<term> | <expression>-<term>
<term> ::= <factor> | <term>*<factor> | <term>/<factor>
<factor> ::= <number> | (<expression>)
<number> ::= <hex string> | <dec string>.
<hex string> ::= <hex digit> | <hex string><hex digit>
<dec string> ::= <dec digit> | <dec string><dec digit>
<hex digit> ::= <dec digit> | a | b | c | d | e | f |
                               A | B | C | D | E | F
<dec digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

The expression result is printed in decimal and hexadecimal form.

Query

The "q" command searches for a pattern of bytes. The possible forms are:

q start S swath pattern	Look for pattern through swath bytes from start address.
q start pattern	Look for pattern through the same number of bytes from start.
q S swath pattern	Look for pattern through swath bytes from where last q command ended.
q start S swath	Look for same pattern through swath bytes from start.
q start	Look for same pattern through the same number of bytes from start.
q S swath	Look for same pattern through swath bytes from where last q command ended.
q	Look for same pattern through the same number of bytes from where last q command ended.

If no start address is given, the q command wraps to the start of the file when the end of the file is reached.

The pattern, which is always a non-empty sequence of bytes, can be described in BNF form as follows:

```
<pattern> ::= <item> | <space><item> | <pattern><space><item>
<space> ::= SPACE | TAB | <space>SPACE | <space>TAB
<item> ::= <number> | <string>
<string> ::= '<simple string>'
<simple string> ::= <empty> | <single string><character>
<empty ::=
<character> ::= <ASCII character other than backslash> |
                <escape character>
<escape character> ::= \n | \r | \t | \f | \b | \0 | | \ddd | \x
```

The escape characters are:

\n	0A
\r	0D
\t	09
\f	0C
\b	08
\0	00
\ddd	ddd should be three octal digits
\x	if x is a non-escape character, it is taken verbatim.

If the q command finds the pattern, it displays 16 bytes of the file, starting with the pattern sought.

Substitute

The "s" command displays a file word by word and allows you to change selected bytes. The possible forms are:

s start	Display word at start address.
s	Display word from where the last s command ended.

The s command displays one word, and waits for one of the following responses:

.	Exit substitute command
RETURN	Display next byte
-	Display last byte
pattern	Replace this byte (and the following bytes, as needed) with the specified pattern. Pattern has the same form as in the q command.

Truncate

The "t" command truncates a file to the specified length, and has the form:

```
t size
```

The file is truncated (or extended) to the "size" number of bytes.

Zap

The "z" command overwrites the specified number of bytes with a given pattern. The available forms are:

z start S swath pattern	Fill swath bytes with pattern beginning at start.
z start pattern	Fill the same number of bytes with pattern beginning at start.
z S swath pattern	Fill swath bytes with pattern beginning where the last z command ended.
z start S swath	Fill swath bytes with zeros beginning at start.
z start	Fill the same number of bytes with zeros beginning at start.
z S swath	Fill swath bytes with zeros from where the last z command ended.

Shell command:	**PATH or PA**
purpose:	This command finds the full pathname of an executable file.
user access:	all users
summary:	path file-list
arguments:	filename
options:	none

Description

The Path command searches the current directory for the specified file with an extension of **.bin, .com,** or **.cmd.** It then searches the **/bin** directory for a **.bin** or **.com** file and the **/cmd** directory for a **.cmd** file. If the specified command is a Shell command, Path notifies the user of that fact. Path locates only executable files.

Path lets you make sure you are running the correct version of your program, rather than a copy that may have been altered.

Shell command: **PRIORITY or PRI**
purpose: This command changes the priority of a process.

user access:
all users (priorities 0 through +40)
privileged user (priorities +40 through -40)

summary: pri [±priority-number][command-line]

arguments: priority number (optional)

command line (optional)

options: none

Description

The Priority command establishes the priority of a process. Priority numbers range from -40 (highest) to +40 (lowest). The highest priority a nonprivileged user may specify is 0, the lowest is +40. A privileged user may specify any priority.

If the Priority command is executed without a priority number, the default value is +10. All processes run without using the Priority command are assigned a priority of 0.

If a command line is given as an argument, the priority specified applies to the process(es) initiated by the command line. If no argument is given, the priority applies to the current shell and all children of the current shell created after execution of the Priority command.

All ready processes are handled in a "round-robin" fashion. The higher the priority, the longer the process is allowed to run before the next process is executed. A -40 priority gets a "time-slice" of 576 milliseconds, a +40 priority gets 16 ms, and a 0 priority gets 96 ms (distribution varies exponentially).

utility:	**PRIV**
purpose:	This program allows any user the status of a privileged user.
user access:	all users
summary:	priv
arguments:	none
options:	none

Description

The Priv utility examines the **/etc/passwd** file for a user named **system.** If this user is not found, an error message is displayed and execution of the utility is aborted.

If the user named **system** is found and there is a password associated with the user, the Priv utility prompts for the password. If the user responds with the correct password or if no password is associated with the user **system,** a new Shell is formed in which the user has the status of a privileged user. Upon exiting from the new shell, the user's previous status is reinstated.

Shell command:	**PROMPT**
purpose:	This command changes the prompt.
user access:	all users
summary:	prompt [char]
arguments:	any character
options:	none

Description

The Prompt command changes the prompt. **Char** is the new character that the Shell is to use as a prompt. It must be a single character. If no character is specified, the prompt is changed to the pound sign (#) for the privileged user and to the percent sign (%) for any other user.

Notes

Changing the prompt from a percent sign to a pound sign does not make a user a privileged user.

Shell command:	**PSTAT or PS**
purpose:	This command displays the status of a process.
user access:	all users
summary:	pstat [-abl]
arguments:	none
options:	-a all -b brief display -l long display

Description

The Pstat command displays the following information on the status of a process:

PID	process identification number
state	state of process: Sleeping Ready Terminated
user id #	
group id #	
Ctty	controlling tty, the tty from which the process was started
Priority	priority of the process
Base	page number of the first memory allocated to the process
Seconds	number of seconds the process has been executing
command line	command line which invoked the process

Options

The **-a** option lists the status of all processes. If the **-a** option is not selected, only those processes with the ID of the user giving the Pstat command are displayed.

The **-b** option displays a brief list of processes and their status.

The **-l** option displays a long list of processes and their status.

utility:	**Q or QUERY**	
purpose:	This program displays a short description of a specified utility program or Shell command.	
user access:	all users	
summary:	query [-s] [name]	
arguments:	the names of one or more utility programs or Shell commands	
options:	-s	system function lists system call data as well as commands and utilities.

Description

The Query program searches a file containing one line descriptions of all of the utility programs and Shell commands for the name given as an argument.

When using Query without an argument, a listing of all one line descriptions of utilities and Shell commands is displayed.

The Query program considers names that are part of other keywords. When the name **fil** is given, Query finds all occurrences of the name **file** as well. This is helpful when the correct spelling of a name is unknown.

After using Query to find the name of the desired command, additional information is obtained by entering **help**, followed by the name of the command. For further details, refer to the Help utility.

The Query program uses the file **/usr/query/query_data** as a database. This file may be edited using the Screen Editor.

Options

The **-s** option searches the file **/usr/query/sys_data, /usr/query/jsys_data,** and **/usr/query/mode_data** before searching the default file, which gives information on the programs only.

The **/usr/query/sys_data** file gives a list of system calls associated with the command. The **/usr/query/jsys_data** and **/usr/query/mode_data** are linked to the files **/equ/jsysequ.z80** and **/equ/modeequ.z80**, respectively.

Example:

The following example demonstrates the use of the Query program.

```
% query delete

query_data
delete    -  removes a file or directory from a file system
deltree   -  deletes a directory and its descendents
passwd    -  change a user password, add or delete a user
```

In the above example, the Query program has displayed all descriptions of Shell commands and utility programs that contain the word **delete** in their descriptions.

utility:	**RAMDISK**
purpose:	This program creates, deletes, verifies, and checksums RAM disks.
user access:	privileged user
summary:	ramdisk [-c #] [-d] [-r] [-s] device-name(s)
arguments:	RAM disk device names
options:	-c # create RAM disk of given size -d delete RAM disk (return memory) -r read RAM disk to show checksum errors -s salvage RAM disk (recompute checksums)

Description

The Ramdisk utility should be called with one option and one or more RAM-disk device names as arguments. Ramdisk performs different actions, depending on the specified option.

Options

The **-c** option must be followed by the size of RAM disk to be created. The size is expressed in K (1024 bytes). The requested amount of memory will be allocated by the system and used as RAM disk. The contents of RAM disk will be all zeros. The Mode utility used on such a RAM disk will show a smaller number of available blocks, as one or more blocks, depending on size, will be used for checksums on individual blocks.

The **-d** option deallocates the RAM disk created by the **-c** option. The memory occupied by the RAM disk is returned to the system's memory pool.

The **-r** option reads all of the RAM disk. Any checksum errors are reported on the raw console.

The **-s** option recomputes all checksums. After running Ramdisk with the **-s** option, the RAM disk driver will report no checksum error. Because checksum errors should not be ignored, use the **-s** option with discretion.

Notes

The Mode utility, when applied to a RAM-disk device, shows various characteristics, including the size, in blocks, and the verify flag. Although Mode can be used to turn off the verify flag, resulting in increased speed, this is not recommended. Turning the verify flag off defeats a valuable checking mechanism. While the flag is off, write operations do not compute a checksum. Thus, when the flag is turned on again, the RAM disk driver may indicate numerous errors. (The checksums are no longer up to date.)

When using Makfs to create a file system on a RAM disk, **makfs -b** 2 forces the beginning of the inode area to the block following the Superblock. This is acceptable because there is no need for a boot area on a RAM disk.

If a RAM disk is mounted to the file **/ram**, the Shell will search for programs in the following directories (in the order shown):

current directory, **/ram, /bin, /cmd**

Example

The following example is a typical command file that creates a RAM disk, loads it with some often-used programs, and mounts it to **/ram**.

```
ramdisk -c 128 rd0                  % Create 128K RAM disk
makfs -b 2 rd0                      % Create file structure without
                                    % boot area
mount rd0 /ram                      % Mount RAM disk
dir /bin                            % Copy useful files from /bin
copy copy.bin version.bin mode.bin ls.bin /ram
dir /cmd                            % Copy useful commands from /cmd
copy bak.cmd /ram
```

Execution of a similar command file might be specified in the file **startup.cmd**.

utility:	**RCOPY**
purpose:	This utility copies a file or block device
user access:	privileged user
summary:	rcopy [options] arguments
arguments:	two file pathnames or one or two block device pathnames or a file pathname and a block device pathname

options:	-b #	first block on input device
	-e #	last block on input device
	-d #	first block on output device
	-s #	block count (swath)
	-l #	I/O buffer size
	-c	compare mode
	-f	forced write mode
	-p	physical mode (Cflop only)
	-r	read-only mode
	-t	terse mode
	-v	verify mode

Description

Rcopy (for privileged users) copies a file or block device to another file or block device. Rcopy is primarily intended for copying a hard disk onto cartridge tape or another hard disk. Files saved with Ftback (68000 Cromix version 20.65) cannot be restored with Rcopy. Rcopy can be used with version 20.08 or higher of the Tar utility. A "-" in place of a file or device name indicates the standard input or standard output.

Options

-b The number following **-b** is the first block of the input device to be read; without this option, reading begins at block 0. This option is useful for copying specific data, such as a hard disk label (block 0) or alternate track table (block 19).

-e The number following **-e** is the last block of the input device to be read; without this option, reading stops at the last block (unless the end of the output device is reached first).

-d The number following **-d** is the first block of the output device to be written; without this option, writing begins at block 0.

-s The number following **-s** is the total number of blocks to be read.

-l The number following **-l** is the length of the I/O buffer in units. If the CTD is used for input or output, a unit is 17K; otherwise a unit is 1K. Without this option, the buffer is 10 units (170K for the CTD and 10K for all other files/devices). A buffer of 510K (**-l 30**) yields an optimum speed of 2 Mbytes/minute when copying between a CTD and a hard disk. This option is ignored when copying between uniform (UNIX) floppy disks.

-c After copying to or from a cartridge tape, repeat the Rcopy command with the **-c** option to verify that the input and output are identical. For all other files or devices, use the **-v** option to copy and verify on the same pass.

-f Overwrites the existing output file (use only when output is to a file).

-p Reduces copying time when both input and output devices are Cromix-Plus floppy disks (large or small, single- or double-sided, single- or double-density). In this case the buffer size equals the cylinder size, regardless of the **-l** setting.

-r Reads the input file or device for errors, but creates no output (only one argument is required).

-t Provides a terse progress report as copying proceeds.

-v Compares the copied output with the input and sends an error message if they are not the same. Do not use the **-v** option for cartridge tapes; verification of cartridge tapes requires a second execution of Rcopy with the **-c** option.

Examples

When copying to an entire hard disk (**std31** or **std63**), be sure not to overwrite the disk label (block 0) or alternate track table (block 19). To save blocks 0 and 19 on floppy disk files **label** and **alttable**, enter (for an 8" diskette mounted in drive A as /a):

```
rcopy -t -s 1 /dev/std0 /a/label
rcopy -t -b 19 -s 1 /dev/std0 /a/alttable
```

To restore these blocks in the event of a hard disk crash, enter:

```
rcopy -t -s 1 /a/label /dev/std0
rcopy -t -d 19 -s 1 /a/alttable /dev/std0
```

If the CTD is the input or output device, copying and verifying requires separate Rcopy commands. To copy partition **std3** to floppy tape, and then verify the copy, enter:

```
rcopy -t /dev/std3 /dev/fted
rcopy -tc /dev/std3 /dev/fted
```

Since the tape is rewound after each Rcopy command, it is correctly positioned for the second pass.
To create and verify a Tar archive of directory **/usr/data** on a tape cartridge, enter:

```
tar -cvf - /usr/data | rcopy - /dev/fted
tar -cvf - /usr/data | rcopy -c - /dev/fted
```

Tar writes directory **/usr/data** to the standard output (the dash after the f option). Rcopy takes the data from standard input (the dash for the input file), and writes it to the tape. In this case, Tar issues the progress reports (**v** option selected).

To restore a specific file or directory from a tape created with the Tar utility, enter:

```
rcopy /dev/fted - | tar -xvf - filename
```

To list the contents of a tape created with the Tar utility, enter:

```
rcopy /dev/fted - | tar -tvf -
```

To dump the contents of a tape created by Rcopy alone, enter:

```
rcopy /dev/fted - | dump
```

utility:	**READALL**
purpose:	This program reads a device to check for errors.
user access:	privileged user
summary:	readall [-at] [-c #[,#]] [-h #[,#]] [-l #] devicename
arguments:	device name
options:	-a do not read declared bad tracks -t do not write progress report -c cylinder numbers to read -h surfaces (heads) to read -l number of passes

Description

Readall checks for bad tracks on an STDC hard disk or any other block device that supports the primitive read setmode. The device name must be in the **/dev** directory, and the device must already be initialized. For an STDC drive, use the device name for the entire disk (**std31** for drive 0; **std63** for drive 1).

Error messages (reports of bad tracks) are sent to the standard error device; status messages are sent to the standard input device. Numbers for the **-c** and **-h** options are assumed to be decimal unless followed immediately by an "h" for hexadecimal.

Options

The **-a** option skips reading of tracks declared bad in the alternate track table (valid only for hard disk drives).

The **-t** option suppresses progress messages and displays only error messages (useful when redirecting output).

The **-c** option, followed by a number, selects a single cylinder to be read; if **-c** is followed by two numbers separated by a comma, a range of cylinders is read (from the first number through the second). Without this option, all cylinders are read.

The **-h** option, followed by a number, selects a single surface (head) to be read; if **-h** is followed by two numbers separated by a comma, a range of surfaces is read (from the first number through the second). Without this option, all surfaces are read.

The **-l** option selects the number of passes (each track is read once on each pass). Without this option, only one pass is made.

Examples:

To make one pass through cylinders 5 through 16 (10h) on device **/dev/std31**, enter:

```
# readall -c 5,10h std31
```

To make 5 passes through the entire device **/dev/std31** (except for tracks declared bad in the alternate track table), enter:

```
# readall -a -l 5 std31
```

To send error messages to both a file and the terminal, use a command such as:

```
# (readall -a -l 5 std0 > /dev/tty) |* tee error_file
```

Shell command:	**RENAME or REN**
purpose:	This command changes the name and/or directory of a file.
user access:	all users
summary:	ren oldfile1 newfile1 [... oldfileN newfileN]
arguments:	one or more pairs of file pathnames (existing pathname first, followed by new pathname)
options:	none

Description

The Rename command changes a filename and/or the directory where it is located.

This command does not move a file from one device to another.

Shell command:	**REPEAT** or **REP**
purpose:	This command repeats a command.
user access:	all users
summary:	rep count command
arguments:	a count of the number of repetitions command
options:	none

Description

The Repeat command is used to repeat a command a specified number of times.

Example:

```
% repeat 5 echo "this line is displayed five times"
this line is displayed five times
this line is displayed five times
this line is displayed five times
this line is displayed five times
this line is displayed five times
%
```

Notes

The Repeat command may be terminated by a semicolon and in this case any command(s) following a semicolon are executed only once. This means that the following command displays the date three times and then displays the time once:

```
% repeat 3 date; time

Wednesday, November 12, 1980
Wednesday, November 12, 1980
Wednesday, November 12, 1980
Wednesday, November 12, 1980 18:54:04
```

Shell command:	**REWIND or REW**
purpose:	This command restores the arguments used to call a command file.
user access:	all users
summary:	rew
arguments:	none
options:	none

Description

The Rewind command restores the arguments used to call a command file. It nullifies the effect of any Shift commands given within the command file. After execution of the Rewind command, **#1** represents the first argument of the original command file, **#2** the second, and so on.

utility:	**RFILE**
purpose:	This program allows binary files to be received from users over the phone lines with error-free results.
user access:	all users
summary:	rfile [-q] [-f] [-d device-name] [-b baud-rate] destination-directory
arguments:	destination directory pathname (must already exist)
options:	-q quiet (default is verbose) -f force -d qtty device name (default is STDIN/STDOUT) -b baud rate (default is current baud rate)

Description

The Rfile utility allows binary disk files to be received by a user on one Cromix system from a user on another Cromix system that is using the Sfile utility to transmit files.

Rfile may operate at either 300 or 1200 baud. It must use an asynchronous modem such as the Cromemco MDM-1200, a Bell 212A, or a Bell 103 type. The modem used must be compatible with the modem the Sfile utility is using to transmit the data. The modem can be connected to any serial port on the Quadart using a 12-wire cable constructed for this purpose.

The qtty driver is used to connect Rfile with the IOP/Quadart and modem. The Cromix-Plus system being used must include the IOP/Quadart drivers (refer to Crogen), and a device file for the qtty should be set up in the **/dev** directory using Makdev. The corresponding entry in the **/etc/ttys** file should have a 0 in the first column.

Example

```
% rfile -d qtty2 -b 300 recvtemp
```

This command line sets the reception rate at 300 baud and stores all of the data received from qtty2 into the existing directory named **recvtemp.**

Messages returned by Rfile

Waiting for phone call

Rfile is in an idle state waiting for a connection from the Sfile utility. Rfile will remain in this mode indefinitely if Sfile fails to make a valid connection.

Receiving [filename] from [devname] into [directory-name]

A valid connection to the Sfile utility has been established, and data is being transferred from the qtty device to the specified disk file in the specified directory.

Connection lost -- ABORT RFILE

The line to Sfile was prematurely disconnected.

ABORT RFILE

A CONTROL-C character was received from the user at the keyboard; Rfile does an orderly exit back to the Cromix Shell.

Options

The **-q** option specifies a different set of Rfile console messages. (Used by Ccall and when Rfile is running on a remote machine.)

The **-f** option causes Rfile to overwrite any existing file with the same pathname as the file sent by Sfile. If this option is not specified and another file exists with the destination pathname, an error is reported.

The **-d** option specifies which qtty device is to be the receiver.

The **-b** option sets the baud rate of the receiving device.

Notes

Rfile is used in conjunction with Sfile. Refer to the Sfile utility for additional information.

When used without an argument, Rfile displays a summary of the command line syntax.

utility: **ROOT**

purpose: This program displays the name of the device containing the root directory.

user access: all users

summary: root

arguments: none

options: none

Description

The Root program displays the root directory's device pathname.

Example:

```
# root

/dev/std0
```

utility: **SC**
purpose: This program allows the user to edit files.

user access: all users

summary: sc filename [filename]

arguments: name(s) of file to be edited

Description

When called without an option or with the **-c** option, the Screen program (**/bin/screen.bin**) creates output files with a CR-LF (carriage RETURN-linefeed) pair at the end of lines. When called with the **-n** option, only a LF character is used as a line terminator. Such files can then be used with the UNIX Operating System.

The Sc program (**/cmd/sc.cmd**) calls the Screen editor with the **-n** option.

Note

If you prefer to alter the Screen program so **-n** instead of **-c** is the default, patch the file **screen.bin** with a zero at location 0FAH. This patched version will operate slightly faster than **sc.cmd.** You may wish to call the patched version **sc.bin,** in which case, the file **sc.cmd** may be discarded.

utility:	**SCAN**
purpose:	This program scans a directory tree
user access:	all users
summary:	scan pathname ['expression']
arguments:	directory pathname optional expression in single quotes
options:	none

Description

The Scan utility is a modernized version of the Find utility. It scans all the files in the specified directory structure for the expression. The expression may be given from the command line or from STDIN (the standard input). If the expression is given from the command line, it should be enclosed in single quotation marks. If the expression is given from STDIN, quotation marks are not required. Scan reads the expression until CONTROL-Z (end-of-file) is typed from the terminal.

The expression is similar to an expression in the C programming language.

Types

Each value is either an integer value or a string value.

Constants

Integer constants can be decimal, hexadecimal, or octal, as in C. String constants are (doubly) quoted strings. In addition, there are a few predefined constants:

is_ordin	type int	value 0
is_direct	type int	value 1
is_char	type int	value 2
is_block	type int	value 3
is_pipe	type int	value 4
ac_read	type int	value 0x01
ac_exec	type int	value 0x02
ac_writ	type int	value 0x04
ac_apnd	type int	value 0x08
days	type int	value 60*60*24

Variables

The following variables are defined at the beginning of the program. There is no assignment, so they cannot be changed by the user.

getuser	type int	current user number
getgroup	type int	current group number
now	type int	current time (number of seconds from 00:00:00 on March 1, 1960)

The following variables are defined after each file is scanned. They provide information from the inode. Again, there is no assignment, so they cannot be changed by the user.

path	string	Full pathname
dir	string	Directory pathname
name	string	File name without directory and extension

ext	string	Extension
		Example: If
		path = "/usr/user1/test.c"
		then
		dir = "/usr/user1/"
		name = "test"
		ext = ".c"

owner	int	File owner
group	int	File group
type	int	File type
aowner	int	Owner access
agroup	int	Group access
aother	int	Other access
nlinks	int	Number of links
size	int	File size
inode	int	Inode number
parent	int	Parent inode
dcount	int	Directory count
usage	int	Usage (blocks)
tcreate	int	Time created (see now)
tmodify	int	Time modified (see now)
taccess	int	Time accessed (see now)
tdumped	int	Time dumped (see now)

Functions

There are a number of predefined functions. These functions yield a value of predefined type, and the arguments are of predefined type.

int shell(s) string s;	Execute command **s**. Return exit value from the command.
int ok	Get a single character from **/dev/tty**. If it is Y or y, return 1, else zero.
int strequ(s,t) string s, t;	Compare string **s** to string **t**. Return 1 if they match, else zero. (The string **t** can contain ambiguous shell characters, such as "*" or "?".
int print(s) string s;	Print string **s** to a line by itself. Always return 1.
int printn(s) string s;	Print string **s** without newline character. Always return 1.
string username(user) int user;	Return string corresponding to given user number, empty string if not found.
string groupname(group) int group;	Return string corresponding to given group number, empty string if not found.
int usernum(name) string name;	Return user number corresponding to given name, -1 if not found.
int groupnum(name) string name;	Return group number corresponding to given name, -1 if not found.

Operators

Operators are listed in order of decreasing priority. In general, operators can be used only on a pair of integers unless explicitly allowed on a pair of strings. Their meaning is the same as in C.

Operator	**Associativity**	**Notes**
( )	left to right	
! ~ -	right to left	Integers only.
* / %	left to right	Integers only.
+ -	left to right	Integers only.
<< >>	left to right	Also on strings. Shifting a string means shifting out.
< <= > >=	left to right	Also on strings. On strings, a lexicographic comparison.
== !=	left to right	Also on strings. On strings, a comparison for strict equality or inequality.
&	left to right	Integers only.
^	left to right	Integers only.
|	left to right	Also on strings. The resulting string is the concatenation of the left and right argument, (in this order).
&&	left to right	Integers only.
||	left to right	Integers only.
? :	left to right	Integers and strings.
,	left to right	Integers and strings.

Examples

For brevity, each sample expression is written as though entered from STDIN, without quotes.

Print all filenames that include **ted:**

```
name == "ted" && print(path)
```

Print all filenames that include **ted** or **mary:**

```
(name == "ted" || name == "mary") && print(path)
```

List **-e** all directories:

```
type == is_direct && shell("ls -e "|path)
```

List **-l** all files with extension **.c:**

```
ext == ".c" && shell("ls -l "|path)
```

Create forced links of all **jsysequ.asm** files that are not already linked to the correct file. If files do not compare, just print the differences:

```
(name|ext) == "jsysequ.asm" && inode != 234 &&
shell("cmpasc -rt /equ/jsysequ.asm "|path) == 0 &&
shell("maklink -vf /equ/jsysequ.asm "|path)
```

Delete, after verification, all files that have not been accessed for more than a year:

```
type == is_ordin && taccess < now - 365*days &&
printn("Delete "|path|"?  ") && ok && shell("del "|path)
```

Delete all files that belong to the user **temp**:

```
owner == usernum("temp") && shell("delete -v "|path)
```

or

```
username(owner) == "temp" && shell("dele -v "|path)
```

Compare all files in directory **foo** to files in the directory **bar:**

```
type == is_ordin && shell("compare -t foo/"|name|ext|" bar/"|name|ext)
```

To create a command file where directories are given as arguments, replace **foo** with #1 and **bar** with #2.

utility:	**SCREEN**
purpose:	This program is used to edit files.
user access:	all users
summary:	screen [-cn] filename [filename]
arguments:	name of file to be edited
options:	-c CR-LF pair used as line terminator -n LF only used as line terminator

Description

The Screen utility program enables the user to edit files. Please refer to appendix B of the Introduction to Cromix-Plus for a complete discussion of the Screen editor.

When called without an option or with the **-c** option, the Screen program (**/bin/screen.bin**) creates output files with a CR-LF (carriage RETURN-linefeed) pair at the end of lines. When called with the **-n** option, only a LF character is used as a line terminator. Such files can then be used with the UNIX Operating System.

Cromix-Plus utilities understand lines terminated by either the LF character or CR-LF pair. A few utilities, however, were originally written for the CDOS Operating System. These programs (notably, those using older versions of Z80 assembler) might require the CR-LF pair as the line terminator.

The Sc program (**/cmd/sc.cmd**) can be used to automatically call Screen with the **-n** option.

If you prefer to alter the Screen program so **-n** instead of **-c** is the default, patch the file **screen.bin** with a zero at location 0FAH. This patched version will operate slightly faster than **sc.cmd.** You may wish to call the patched version **sc.bin**, in which case, the file **sc.cmd** may be discarded.

utility:	**SETPRI**
purpose:	This command changes the priority of a process.
user access:	all users (priorities 0 through +40) privileged user (priorities +40 through -40)
summary:	setpri process-id priority
arguments:	process id number (PID) priority number
options:	none

Description

The Setpri utility assigns a priority number to a given process. Priority numbers range from -40 (highest priority) to +40 (lowest priority). A non-privileged user can assign only non-negative priorities to his own processes; a privileged user can assign any priority to any process.

Example

To assign a priority of 20 to the process whose PID number is 456, enter:

```
setpri 456 20
```

utility: **SFILE**

purpose: This program allows binary files to be sent between users over the phone lines with error free results.

user access: all users

summary: sfile [-q] [-d device-name] [-b baud-rate]
[-n phone-number] [-l login-name]
[-p password] file-list

arguments: one or more filenames

options:

-q	quiet (default is verbose)
-d	qtty device name (default is STDOUT)
-b	baud rate (default is current baud rate)
-n	phone number (dashes may be used)
-l	login name
-p	password

Description

The Sfile utility allows binary disk files to be transmitted from a user on one Cromix system to a user on another Cromix system using the Rfile utility to receive files.

Sfile operates at either 300 or 1200 baud. It must use an asynchronous modem such as the Cromemco MDM-1200, a Bell 212A, or a Bell 103 type. The modem used must be compatible with the modem the Rfile utility is using to receive the data. The modem can be connected to any serial port on the Quadart using a 12-wire cable constructed for this purpose.

The qtty driver is used to connect Sfile with the IOP/Quadart and modem. The Cromix-Plus system being used must include the IOP/Quadart drivers (refer to Crogen), and a device file for the qtty should be set up in the **/dev** directory using Makdev. The corresponding entry in the **/etc/ttys** file should have a 0 in the first column.

If a Cromemco MDM-1200 modem is being used, a telephone number can be added to the command line for automatic dialing. Otherwise, the call must be manually originated using standard modem procedures.

Example

```
% sfile -d qtty1 -b 300 -n 555-1212 letter.txt
```

This command line sets the transmission rate at 300 baud and transmits the file **letter.txt** using device qtty1 with a connected MDM-1200 that auto-dials the number 555-1212.

Messages returned by Sfile

Now waiting for call to complete ...

Sfile is waiting for a connection to be made with another modem over the phone lines. This call can either be dialed manually using Bell equipment, or the MDM-1200 can dial the number and establish the connection automatically.

No answer -- Call aborted

If a connection is not established within 60 seconds, Sfile exits to the Cromix Shell.

Transmitting [filename] to [devname]

When a valid connection is established with the Rfile utility at the other end, the specified file is transmitted through the specified qtty device.

Rfile not responding -- Sfile aborted

If a connection is established with another modem, Sfile determines if the Rfile utility is ready at that end. If Rfile is not running at the other end, or if Rfile is running at an incompatible baud rate, Sfile disconnects the line and exits to the Cromix Shell.

Connection lost -- ABORT SFILE

The line was disconnected in the middle of a transmission; Sfile exits to the Cromix Shell.

ABORT SFILE

A CONTROL-C character was received from the user at the keyboard; Sfile does an orderly exit to the Cromix Shell.

Options

The **-q** option specifies a different set of Sfile console messages. (Used by CCall and when Sfile is running on a remote system.)

The **-d** option specifies which qtty device is to be the transmitter.

The **-b** option sets the baud rate of the transmitting device.

The **-n** option specifies a phone number to be used by the MDM-1200 modem.

The **-l** ("ell") option specifies a login name to be used on the remote Cromix system.

The **-p** option specifies a password to be used on the remote Cromix system.

Notes

Sfile is used in conjunction with Rfile. Refer to the Rfile utility for additional information.

When called without an argument, Sfile displays a summary of the command-line syntax.

Shell command:	**SHELL or SH**
purpose:	This command creates a Shell process.
user access:	all users
summary:	[shell] [cmd file]
arguments:	optional command file
option:	-c complete input line -p parsed input line -q quiet -z do not terminate on CONTROL-Z

Description

Given without an argument, the Shell command creates an interactive shell process; given with the name of a command file, Shell executes that file as it echoes each line of the file to the terminal. Entering only the name of a command file implies the use of the Shell command, but file is not echoed to the terminal.

If there are two filenames in the current directory that differ only by their **.bin** and **.cmd** filename extensions, entering the filename executes the **.bin file**, not the **.cmd** file. In such cases, enter "shell" plus the filename to execute the **.cmd** file (refer to chapter 4 of the Introduction to Cromix-Plus manual for a description of the search path for executable files).

On Cromix-Plus version 30.94 and higher, the Shell recognizes the variable

```
#err
```

as the decimal value of the last error number.

Options

These options are needed only when a program is calling a shell; the **-c** and **-p** options make no sense when the Shell is called from the terminal.

The **-c** option indicates that a full command line is being passed to the Shell (the line is not parsed into arguments).

The **-p** option indicates that the command being passed to the Shell has been parsed.

The **-q** option prevents the lines in a command file from being echoed to the terminal (STDOUT).

If the Shell is called with the **-z** option, CONTROL-Z will not terminate an interactive shell.

Example:

As the value of #err might be a negative number, the -- sign notifies the Echo utility that there are no more options.

```
bad_command
if #err != -1 echo -- #err error number
```

Notes

The Shell command line editor is enabled if the "kr" attribute is present in your terminal's Termcap file (the attributes "kI", "kE", "kl", "kr", "nb", and "nd" must also be present). If defined, the "pd" attribute and the "pi", "pe" pair will also be used. The terminal type can be set automatically by the **/etc/ttys** file or manually by the Term command.

Shell command:	**SHIFT**
purpose:	This command shifts the arguments in a command file.
user access:	all users
summary:	shift
arguments:	none
options:	none

Description

The Shift command is used to shift the arguments in a command file. After execution of the Shift command, **#1** represents the second argument from the original command line, **#2** represents the third, and so on. After another execution of the Shift command, **#1** represents the third argument, etc.

The Rewind command nullifies the effects of the Shift command.

For examples of Shift and Rewind commands, refer to the sample command file under the discussion of the Exit command.

utility:	**SHUTDOWN**
purpose:	This program shuts down the system.
user access:	privileged user
summary:	shutdown
arguments:	none
options:	none

Description

The Shutdown program (**/cmd/shutdown.cmd**) contains commands to shut down the operating system by killing all processes, flushing buffers, and logging off all users. It first warns users and provides a five-second countdown.

Shutdown also has a facility that works with the Startup command to detect inadvertent system terminations.

Run the Shutdown program whenever system operation is to be terminated.

utility:	**SIM**
purpose:	This utility allows CDOS programs to run under the Cromix-Plus Operating System.
user access:	all users
summary:	(sim) progname arg0,arg1,...,argn
arguments:	program name **and** arguments to the program to be run
options:	none

Description

The Sim program allows CDOS programs to run under the Cromix-Plus Operating System. The CDOS simulator is automatically loaded when a file with the extension **.com** is executed.

Notes

The Cdoscopy utility program is the only way to read files from or write files to CDOS format disks from the Cromix-Plus Operating System.

Drive/File Access From CDOS Programs

For CDOS programs to gain access to files on various drives, the CDOS Simulator converts disk specifiers to directory names. For example:

B:Filename becomes **/B/Filename**

If no disk specifier or the disk specifier A is used (as in **A:Filename**), the file is assumed to be in the current directory.

To take full advantage of this scheme, Cromemco recommends a file structure be constructed as follows:

1. Create files **B**, **C**, **D**, etc. in the root directory. Each file corresponds to one of the disk drives in the system.

2. Mount each disk on the appropriate drive using the Mount utility:

```
# mount fdb /b
```

   Note that these **must be** Cromix format disks.

3. The files on those disks may be read and written from CDOS programs. The CDOS simulator, running under the Cromix-Plus Operating System, automatically converts the CDOS drive specifiers to the appropriate directory names.

4. Each disk mounted must be unmounted before it is physically removed from the system. To do this, use the Unmount utility:

```
# unmount fdb
```

Disks created in this manner are in the Cromix Operating System format and not CDOS compatible.

The permanent mapping of CDOS directories to Cromix directories is defined by a table in **sim.bin**. This table contains a 16-byte entry for each CDOS directory. The entry is the pathname of the Cromix directory to which the CDOS directory maps. Using Dump, you can display the contents of the table. The entry for CDOS directory **A** starts at location 509H, the entry for CDOS directory **B**, at location 519H, and so on.

To change the mapping of CDOS directories, the Patch utility can be used to change the contents of the table. When changing an entry:

1. Each pathname must end with a slash mark ("/").

2. Each string must be terminated with a null character.

3. The pathname, including the final "/" cannot be longer than 15 characters.

"A" refers to the current directory, and should not be changed.

Users may easily define the mapping of CDOS directories to Cromix directories when calling Sim explicitly.

Example:

```
sim -c /usr/lib program.com arguments
```

refers all references to the CDOS **C** directory to the Cromix directory **/usr/lib.** Thus, one or more CDOS directories may be remapped to specific Cromix directories by giving the letter of the CDOS directory as a flag with the pathname of the Cromix directory as the next argument.

Shell command:	**SLEEP**
purpose:	This command suspends program execution.
user access:	all users
summary:	sleep time
arguments:	time in seconds
options:	none

Description

The Sleep command suspends execution of a process for the number of seconds specified. Sleep can be used to execute a command after a certain amount of time. For example:

```
sleep 60 ; command
```

This example executes the command after 60 seconds, or one minute.

utility:	**SORT**
purpose:	This utility sorts or merges files.
user access:	all users
summary:	sort [-mubdfirt?] [-o path] [-n lines] [-l size] [+x[.y[bdfir]] ...] [file_list]
arguments:	input filename(s)

options:

-m	merge sorted input files
-u	unique records only
-b	leading spaces and tabs ignored
-d	dictionary order
-f	consider uppercase as lowercase
-i	ignore all control characters and 7Fh
-r	reverse order
-t?	use ? as field separator
-o	output file
-n	number of lines in memory
-l	maximum line size
+x.y	sort on keys

Description

The Sort utility sorts the lines in one or more files in ASCII order. ASCII order is: nonprinting characters, blanks, punctuation, digits, uppercase alphabetic characters, and lowercase characters.

Each line (or record) in a file is a string of characters terminated by a newline (0Ah). The "+x.y" option allows you to begin sorting on any character in the line; without the +x.y option, sorting begins on the first valid character.

When called without arguments, Sort takes input from STDIN and sends output to STDOUT. An ASCII table is included in the Cromix-Plus Programmer's Reference Manual.

Options

The **-b** option ignores leading tabs and spaces, and sorts lines or fields according to their first nonblank character. If the first column shown below is the input file, a sort with no options produces column two, while a sort with the **-b** option produces column three.

```
INPUT FILE        SORT WITH NO OPTIONS        SORT WITH -b

maser                   McKinley              MacDowell
McCormack            MacDowell                MacLeish
MacDowell            McCormack                McCormack
   McKinley          MacLeish                    McKinley
mace                 mace                     mace
MacLeish             make                     make
make                 maser                    maser
```

In column two, the record with the most white space comes first because blank spaces precede characters and letters in an ASCII sort. The last column is closer to alphabetical order, but uppercase entries are first (blanks and tabs are retained, though they do not affect the order of the file).

The **-d** option sorts by letters, numbers, and blanks, but ignores special and nonprinting characters (dictionary order). If the first column shown below is the input file **db.in**, a sort with no options produces column two (standard ASCII sort), while a sort with the **-d** option produces column three. In a dictionary sort, lines with only special characters are sorted in standard ASCII order, but in lines with both special and alphabetic characters, special characters are treated as blanks.

```
INPUT db.in       SORT WITH NO OPTIONS        SORT WITH -d

a                     **a                     +++
+++                   +++                     . 444
aaa                   +C                      BBB
BBBBB                 . 444                   BBBB
C                     BBB                     +C
+C                    BBBB                    C
BBB                   C                       **a
. 444                 a                       a
**a                   aAa                     aAa
aAa                   aaa                     aaa
```

To create a standard ASCII sort in output file **db2.out**, enter:

```
% sort -o db2.out db.in
```

To create a dictionary sort in output file **db.out**, enter:

```
% sort -d -o db.out db.in
```

Note that the output of a dictionary sort is not in simple alphabetical order (uppercase letters precede lowercase letters).

The **-f** option treats upper- and lowercase letters equally. Special and nonprinting characters retain their order of precedence. If the first column shown below is the input file **db.in**, a sort with the **-f** option produces column two.

```
INPUT db.in         SORT WITH -f

a                   **a
+++                 +++
aaa                 +C
BBBBB               . 444
C                   a
+C                  aAa
BBB                 aaa
. 444               BBB
**a                 BBBBB
aAa                 C
```

The **-r** option reverses the ASCII order of precedence (e.g., "z" precedes "a"). If the first column shown below is the input file **db.in**, a sort with the **-r** option produces column two.

```
INPUT db.in         SORT WITH -r

cat                 scat
hat                 sat
scat                pat
sat                 hat
pat                 cat
   gnat             1
   splat            !  Wo Fat
   slat                gnat
1                      splat
!  Wo Fat              slat
```

To create a reversed sort in output file **db.out**, enter:

```
% sort -r -o db.out db.in
```

Note that the **-r** option completely reverses the ASCII ordering scheme for blank space, letters, and numbers.

The **-m** option merges previously sorted input files to create an ordered output (provided that both input files are sorted according to the same scheme). As shown below, the input file **db.in** (previously sorted without options) can be merged with itself using the **-m** option, as follows:

```
% sort -m -o db.out db.in db.in
```

INPUT db.in	SORT WITH -m
Bat	Bat
Fat	Bat
cat	Fat
rat	Fat
	cat
	cat
	rat
	rat

The merged output is in ASCII order, as are the input files. With adequate disk space, you can merge any number of files.

The **-u** option deletes the duplicate records in a file, leaving one copy of each record. For example, using the output file **db.out** from the previous merge operation as an input file, the command

```
% sort -u -o u.out db.out
```

creates an output file **u.out** that is identical to **db.in** of the previous example (no duplicate entries). For the **-u** option to identify two records as a match, they must be identical in all respects.

The **-i** option ignores all control characters and 7Fh (DEL) when sorting (e.g., "ab c" would be identical to "abc").

The **-l** option, followed by a number, defines the maximum line length; without this option, lines are limited to 80 characters. If an input line is longer than the maximum line length, the program will abort.

The **-n** option, followed by a number, indicates the number of lines that are kept in memory; without this option, 1000 lines are kept. If all input lines cannot be stored in memory, excess lines are stored on disk and merged later in a single result file. For faster operation when sorting large files, use the **-n** option to increase the number of memory-resident lines.

The **-t** option, followed immediately by a character, indicates that the specified character will be taken as the field separator (rather than space or tab) wherever it occurs in the input file. This option is used only with the **+x.y** option.

The **+x.y** option specifies the first character on the line where sorting begins; without this option, sorting begins on the first valid character on the line. The **x** indicates the number of fields to be skipped (counting from the left end of the line); **y** indicates the number of characters to be skipped within the selected field. If x=0, the first field on the line is selected.

Without the **-t** option, each field (except the first one) starts with a space or tab following the last nonblank character of the previous field. If the -t option is used, each field ends with the specified character. For example, without the **-t** option, a line is broken into fields as follows:

```
Four scores and seven years ago
000011111122223333334444445555
```

With the **-t** option such as "-t:", a line is broken up as follows:

```
system :        :0      :0       :/
000000000111111111111111111112222222333333334
```

If one or more of the **-b**, **-d**, **-i**, or **-r** options immediately follow the "y" value, they affect only the selected field; otherwise, they affect the entire sort. As shown below, the input file **alpha** has four records, each with eight fields of 3 bytes each. Note that all fields except the first one start with a blank.

```
abc def ghi jkl mno pqr stu vwx
Abc dEf ghI Jkl Mno pQr sTu Vwx
aBC DeF gHI Jkl mNo Pqr StU vWx
ABc dEF GHi jKL mnO pQR sTu VWx
```

To sort the file in dictionary order, starting with the first character of the second field ("def"), enter:

```
% sort -d +1.0 -o alpha.out alpha
```

The output file **alpha.out** would be:

```
aBC DeF gHI Jkl mNo Pqr StU vWx
ABc dEF GHi jKL mnO pQR sTu VWx
Abc dEf ghI Jkl Mno pQr sTu Vwx
abc def ghi jkl mno pqr stu vwx
```

To sort **alpha** using only the second and third characters of the second field (not counting the spaces), enter:

```
% sort +1.1db -o alpha.out alpha
```

The **d** and **b** options affect only the second field.

In a more complex example, the file **who** contains information on immigrants to California, as follows:

```
Lopez      Jack      Spain      Bower     Orange
McNiff     John      England    Rose      Sonoma
Rizzo      Jill      Italy      Bly       Kings
Ross       Jerry     Wales      Green     Placer
Mcniff     John      England    Greer     Placer
```

To sort the file by country of origin, last name, first name, and county (starting with the first character of each field and ignoring upper and lowercase letters in the last-name field), enter:

```
% sort -t^I +2.0 +0.0f +1.0 +4.0 -o who.out who
```

Since each field is separated by a tab rather than blanks, the **-t** option selects the tab (CONTROL-I) as the field separator. The output file **who.out** would be:

```
Mcniff     John      England    Greer     Placer
McNiff     John      England    Rose      Sonoma
Rizzo      Jill      Italy      Bly       Kings
Lopez      Jack      Spain      Bower     Orange
Ross       Jerry     Wales      Green     Placer
```

Because of the **-f** option, the difference in capitalization between Mcniff and McNiff is ignored, and the two records are sorted by county.

utility: **SPOOL**
purpose: This utility queues files and sends them to a printer.

user access: all users

summary: spool [[devname]pathname(s)]

arguments:

device name

If no device name is specified, output is directed to **/dev/prt.** The device name may be used to direct the output of the Spool program to any of the system's printers.

pathname

Filenames must be used to add files to the printing queue. Filenames or the sequence numbers assigned by the Spool program may be used to delete or change priority. If no pathname is given, input from STDIN (terminated by CONTROL-Z) will be spooled.

options:

-a	all files
-d	enter and delete
-h	header
-f	FORTRAN filter
-v	verbose
-w [ON/OFF]	wrap around
-p #	priority of spooled files
-m #	multiple copies
-b #	bottom margin
-s #	character width
-t #	line height
-u #	left margin
-z #	forms number

Commands

-l	list
-c #	change priority
-k	kill
-q	quit

Description

The Spool utility allows one or more users to send printing jobs to one or more printers in an orderly sequence that may be changed at any time.

If no file is specified, input is taken from STDIN. This means the Spool utility can be used with redirected input or pipes.

When the Spool utility is called to add files to the printing queue, the files are copied into a directory named **/usr/spool**.

After the execution of the Spool program with any of its options, the specified files are sent to the printer. This is accomplished by a function intrinsic to the Cromix Operating System.

Output from the Spool program may be directed to any character device located in the current directory or in the device table (**/dev**).

If no device is specified, **/dev/prt** is assumed. The Cromix Operating System, as shipped, assumes a dot-matrix printer is the system printer. If a different printer is to be used as the system printer, change the printer type (refer to the Cromix-Plus System Administrator's Guide).

As requests are made to print additional files, the Spool program forms a **print queue**. Each file entered into the queue is assigned a unique **sequence number**. Once in the printing queue, files may be referenced by their filename or sequence number.

If two or more files in the queue have the same filename, a reference to that filename refers to all files with the same name. For example, if the **-k** (kill) option is used with a filename that appears more than once in the queue, all files with that name are deleted from the queue. The sequence number can always be used to refer to a specific file.

Each file added to the printing queue is assigned a priority number ranging from 0 to 9. Zero is the highest priority and is reserved for a privileged user. If no priority is specified, a value of 5 is assigned automatically. A priority number must be specified when using the **-c** (change priority) option.

If two users request a print job with the same priority, the requests are serviced on a first-come, first-served basis.

A user other than a privileged user has access only to files that the user placed in the printing queue. The priority of a file in the printing queue can be changed by the user who initiated the printing request or by a privileged user. In a similar manner, only the privileged user or the user who added a file to the printing queue can delete the file from the queue by using the **-k** (kill) option. Any user can list **all** of the files in the printing queue by using the **-la** (list all) options.

Ambiguous file references must be used with caution. When an ambiguous file reference is expanded, it generates a list of filenames matching files in the current directory. An ambiguous file reference can be used when giving the Spool program files to add to the printing queue.

Ambiguous filenames are expanded from a directory, and not from a spool queue. An ambiguous file reference does not work properly when killing or changing the priority of files in the printing queue if files with the same name do not exist in the current directory. This is the case when the **-d** (delete) option is used to add files to the printing queue, or if the current directory is changed by the user.

If Spool is interrupted for any reason, such as a power failure, jobs are left in the queue. There are three ways to restart Spool. (Before restarting Spool, the printer should be manually brought to top-of-form.)

The first method is to spool another job. This restarts printing at the beginning of the first job in the queue (the job that was interrupted).

The second method, used when there are no more jobs to be spooled, is to enter the command line:

```
# daemon /dev/yyy
```

where yyy is the device name of the printer being spooled to (usually **prt**). This also restarts printing at the beginning of the interrupted print job.

The third method is to delete all spool jobs using the command line:

```
# spool -qa
```

and then respool all unprinted jobs.

Options for Spooling Files

The **-d** option adds all specified files to the spool queue and deletes them from the directory in which they reside. This option may include a device name, and must include a list of one or more filenames.

The **-f** option interprets the first character of each line according to FORTRAN conventions:

0	double vertical space before printing
1	formfeed before printing
+	no vertical movement
other	single vertical space before printing

The **-h** option causes all specified files to be preceded by a one-page header. The first line of the header page contains the name of the user who spooled the file, the date and time, and the name of the file. This is followed by the same information displayed in large characters. The large character portion of the header page truncates the user and filenames to eight characters. Note that the header uses the full width of standard 132-column paper.

The **-m** option prints files a specified number of times. The maximum number of copies is 255.

The **-p** option assigns a priority number to a printing job at the time it is initiated. The option must be followed by the desired priority number and may include a device name.

The **-v** option displays the list of files being processed. It may be used with all options except **-l**.

Mode Options

Mode options are stored with the spooled file. The printer daemon sets the mode options, as specified by the Spool command, before printing each file and restores them after printing each file.

The **-w** option, followed by ON or OFF, turns wraparound on or off. If wraparound is turned off, printed lines will be truncated. If the **-w** option is not selected, the printer daemon will not set the mode.

The **-b** option, followed by a number, defines the bottom margin, i.e., the number of lines not used at the end of a page. If the **-b** option is not selected, the printer daemon will not set the mode.

The **-s** option specifies the character width on a TYP printer. The **-s** option must be followed by a number defining the character width. If the **-s** option is not selected, the printer daemon will not set the mode.

The **-t** option defines the character height on a TYP printer. The **-t** option must be followed by a number defining the character height. If the **-t** option is not selected, the printer daemon will not set the mode.

The **-u** option, followed by a number, defines the left margin. If the **-u** option is not selected, the printer daemon will not set the mode.

The **-z** option, followed by a forms number, specifies which form of paper should be used for printing. Printing will not start until the printer driver is told that the printer contains the required paper form. If the **-z** option is not selected, the value zero is used. For details, refer to the Mode utility.

Options for Listing Files

The **-l** option lists all jobs in the printing queue that the user initiated. They are listed in a table with the following information:

1. **Filename** of print file

2. Name of **user** requesting printing job

3. **Se**quence number of printing job

4. Destination **device** of printing job

5. **Priority** of printing job

6. **Pages** in printing job

7. **Lines** in printing job

8. **Copies** to be printed

9. **Forms** number to be used

A privileged user always gets a list of all jobs in the printing queue.

The **-la** option lists **all** printing jobs in a table. Refer to the list option.

Options for Changing Priority

The **-c** option sets the priority of all specified files in the spool queue to the specified value. This option is followed by a priority number, and must include a list of one or more filenames or sequence numbers.

Options for Removing Files from the Spool Queue

The **-k** option deletes all specified files from the spool queue. If a specified file is printing, the printing is aborted. This option must include a list of one or more filenames or sequence numbers.

The **-q** option deletes all files that have been directed to the specified device from the spool queue.

The **-qa** option may be used only by a privileged user. It deletes **all** files that have been directed to the specified device from the spool queue.

Notes

Where no option is specified, the files specified by the pathname are added to the printing queue. A device name may be specified.

If more than one option is used, and one or more of the options requires an argument, the following syntax should be followed.

```
% spool -hv -m 3 -p 1 filename
```

The options that do not require arguments (**h** and **v** above) are grouped, preceded by a hyphen (-), and followed by a space. This group is followed by the option(s) that require arguments. Each option is preceded by a hyphen and followed by a space, a number, and another space. Additional option and argument pairs may follow. Finally, the filename(s) of the file(s) to be spooled are entered.

In the following examples, assume the print files **t**, **u**, **w**, **x**, **y**, and **z** exist in the current directory. First, place each of these files in the printing queue:

```
% spool -v t u w x y z
t
u
w
x
y
z
%
```

Because the verbose option is used, the Spool program listed each file as it was copied to the **spool** directory. The list option is then used to display the printing queue:

```
% spool -l

    Filename   User   Seq   Device     Pri   Pages   Lines   Copies   Form
 -> t          fred   36    5:5 prt    5     2        95     1        0
    u          fred   37    5:5 prt    5     2       107     1        0
    w          fred   38    5:5 prt    5     1        42     1        0
    x          fred   39    5:5 prt    5     2       115     1        0
    y          fred   40    5:5 prt    5     2       115     1        0
    z          fred   41    5:5 prt    5     3       160     1        0
```

The arrow at the upper left of the listing indicates the file currently being printed. All jobs have a priority of 5 because no priority was indicated when the jobs were put in the queue.

Next, change the priority of file **y** to 2 and change the priority of the file with the sequence number 39 (file **x**) to 3. Then, delete the file **u** from the queue using the **-k** option. Finally, add a "message" to the printing queue, and display the revised printing queue. A "message" is input from the terminal (STDIN), which the user sends to the printing queue by typing CONTROL-Z (end-of-file).

```
% spool -c 2 y
% spool -c 3 39
% spool -k u
% spool
this is a message
^Z% spool -l

    Filename   User   Seq   Device     Pri   Pages   Lines   Copies
 -> t          fred   36    5:5 prt    5     2        95     1
    y          fred   40    5:5 prt    2     2       115     1
    x          fred   39    5:5 prt    3     2       115     1
    w          fred   38    5:5 prt    5     1        42     1
    z          fred   41    5:5 prt    5     3       160     1
    ----       fred   42    5:5 prt    5     1         2     1
```

To spool multiple copies of a job, use the **-m** option.

Example:

The command

```
% spool -m 3 pay7000
```

prints 3 copies of the report **pay7000**.

The command

```
% spool -hm 3 pay7000
```

prints 3 copies of **pay7000** with one header page at the beginning of each copy.

A pipe can be used to redirect output from a program to the printer. The following command line prints a list of the files in the current directory.

```
% ls | spool
```

utility:	**STARTUP**
purpose:	This file contains commands that are executed whenever the system is started up.
user access:	all users
summary:	startup
arguments:	none
options:	none

Description

As shipped, the Startup program (**/etc/startup.cmd**) executes the Time program to set the system clock and date. Additional commands may be added to **/etc/startup.cmd** at the System Administrator's discretion.

After the system is booted, Startup notices whether the system was last shutdown properly. If it was not, Startup informs the user the check program should be run to verify file system integrity.

utility:	**STDLOAD**
purpose:	This program loads a program into an STDC.
user access:	privileged user
summary:	stdload filename device
arguments:	filename of program to be loaded
	std device to be loaded
options:	none

Description

The Stdload utility loads a file into an Stdc. This utility is normally used to load the Stdc firmware **/etc/stdcfirm** into an Stdc. Stdload is found in the /etc/ directory.

utility:	**STRCMP**
purpose:	This program test for equality between the contents of stdin and a set of strings.
user access:	all users
summary:	strcmp [-efr] string(s)
arguments:	one or more strings
options:	-e exact -f compare first characters -r reverse sense of test

Description

This utility compares the string read from stdin to the set of strings on the command line and sets an error return code if none of the strings matches the contents of the input string.

The test made by Strcmp is case-insensitive unless the **-e** option is used. Strcmp tests for equality between the strings. To locate text strings embedded in the text of a file, use the Match utility.

Options

The **-e** option forces the match to be case sensitive.

The **-r** option reverses the sense of Strcmp by setting the error code if a match is not made.

The **-f** option checks only the first character of the input string against the first character of each of the control strings.

Example

```
% echo -n "Do you want to shut down the system?"
input | strcmp YES OUI SI
if -err goto noshutdown
kill -2 1
%noshutdown
```

The example above is a typical command file that uses Strcmp and Input. The first line sends the string within quotation marks to the standard output. The second line uses the Input utility to send the user's response into the pipe to be analyzed by the Strcmp utility. The Strcmp then tests for occurrences of the strings YES, OUI, or SI. If any of the control strings was entered, the system is shut down using the Kill command. If the user did not enter any of the control strings, Strcmp sets an error code. The command that follows passes control to the label **noshutdown**. If the user answers no to the question, the system is not shut down.

Shell: command:	**SYNCHRONIZE or SYNC** This program provides a one-time flush of system buffers.
user access:	all users
summary:	sync
arguments:	none
options:	none

Description

System buffers should be occasionally written to the disk:

- at shutdown.
- at unmount.
- whenever the flush utility (if it is running) says so.
- when the Sync command is issued.

utility:	**SYSDEF**
purpose:	This program generates a configuration file for the Crogen utility.
user access:	all users
summary:	Sysdef output_file input_file
arguments:	Pathname of system configuration file followed by the pathname of configuration description file.
options:	none

Description

The Sysdef utility is a specialized compiler that generates the configuration module used by the Link68 utility to generate a new **cromix.sys** file (refer to the Crogen utility). The input file, usually called **sysdef**, contains a Shell-like description of the system parameters. The following are general rules:

Each line has a number of fields separated by white space. The first field is the name of the parameter being defined, the remaining fields are the arguments. Their precise form depends on the parameter being defined. A percent sign (%) marks the end of useful information. The text in a line following the percent sign is a comment. All numbers are entered in decimal form. The order of definition of individual system parameters is arbitrary.

The following sections describe the precise meaning of the system parameters.

Maxmem

The Maxmem command defines the size of the system memory in 256K units. This unit size is chosen because the smallest memory card is 256K bytes. The number of such units supported must follow the command name. One memory unit is the smallest reasonable value, and defines a minimal Cromix-Plus system which would be quite limited. During the initialization process, Cromix-Plus will verify all maxmem memory units in 16K units. Each 16K unit will be reported as present (+), absent (-), or erroneous (E). This means that a Cromix-Plus system can be built with an arbitrarily large maxmem (62 being the largest reasonable value). This would cause Cromix-Plus to check a large amount of non-existing memory. Memory units above maxmem will not be referenced, memory units below maxmem will be used if they exist and are usable. For example:

```
maxmem 2
```

means 1/2 Megabyte of memory, a reasonable sized Cromix unit.

CDEV

Cromix supports 12 character drivers, numbered from 1 to 12. These numbers are also used as (character device) major device numbers. The CDEV command must be followed by the major device number. If a device number is not used, further data is not required. If the device number is to be used, the device number must be followed by the driver name and, if necessary, by the arguments the driver requires. If a driver requires arguments, they must be integers separated by white space. The list of driver names supplied in the **iolib.o68** file, together with the explanation of the meaning of the arguments, if applicable, can be found at the end of the **sysdef** file supplied with Cromix-Plus in the **/gen** directory. For example:

```
CDEV  01    tty      0 2 5
CDEV  02
CDEV  03    sysdev
.......................................
```

says that major device number 1 is the tty driver that will support the minor device numbers 0, 2, and 5, i.e., the FDC terminal and two TUART terminals hooked the first TUART. Next, major device number 2 will not be supported, major device number 3 will be the system device (required), and so on.

BDEV

Cromix-Plus supports 8 block device drivers, numbered from 1 to 8. These numbers are used as (block device) major device numbers. The BDEV command must be followed by the major device number. If a device number is not to be used, further data is not required. If the device number is to be used, the device number must be followed by the driver name and by any arguments the driver might require. If a driver requires arguments they must be integers separated by white space. The list of driver names, supplied in the **iolib.o68** file, together with the explanation of the meaning of the arguments, can be found at the end of the sysdef file supplied with Cromix-Plus in the **/gen** directory. For example:

```
BDEV  01   cflop
BDEV  02   uflop
BDEV  03
........................
```

says that major device number 1 is the (Cromix) floppy driver which can support up to four 8-inch or 5-inch floppies. Major device number 2 will support up to four uniform style floppies, major device number 3 will not be used and so on.

RAW

Cromix-Plus uses the concept of a raw character driver. This is a very simple driver which is not accessible to user programs. It is used by the system only for:

- memory test display during startup.
- definition of root device during bootstrap.
- display of error detailed error messages such as disk I/O errors.
- display of catastrophic error messages when the normal system functions are considered to be too dangerous to use (e.g., unexpected interrupt).

Example:

```
RAW      raw-fdc
```

defines the FDC terminal to be (also) a RAW terminal.

The raw driver is used as an input device only to determine the root device if the root device was not defined in the **sysdef** file when the system was generated. This means that Cromix-Plus can run with the RAW terminal disconnected; however, this practice is not recommended. The usual solution is to use the FDC terminal both as a RAW terminal and as a standard terminal (tty1). Reasons which in the past led to generation of earlier versions of Cromix without the FDC terminal, have little validity with Cromix-Plus:

> Cromix 11.XX and Cromix 20.XX did not have a sufficient number of process tables and/or Shell buffers since both were bound to be allocated in the single 64K bank that was occupied by the kernel and the drivers. A few K of memory was an important requirement. Cromix-Plus does not have this limitation: All of the available memory can be used for process tables if a user so decides.
>
> Tty terminals were inferior to qtty terminals because the tty terminals could lose characters when interrupts were disabled for substantial periods of time. This problem has been greatly reduced and a tty terminal is nearly as reliable in this respect as a qtty terminal.

ROOT

The file **Cromix.sys** which is to be booted may be read from any device (automatically from Reset or explicitly using the Boot command). When the system file begins execution it must determine which device is going to be the root device. The ROOT command offers three possibilities.

```
ROOT    none
```

means that Cromix will ask the operator (using the RAW device) to determine the root device.

```
ROOT    boot
```

means that the root device should be the same as the device from which the file **Cromix.sys** was read.

```
ROOT    6 0
```

for example, means that root device is to be the device with the major device number 6 and the minor device number 0 (presumably std0).

LOGIN

After the system is booted, the terminal identified as the **/dev/console** can be automatically logged on to any predefined user name. For example,

```
LOGIN   system
```

If the keyword LOGIN is not followed by a user name an automatic log on will not occur.

LOGMSG

During the login procedure, the system identifies itself. Identification consists of three lines

Version of the operating system
Copyright message
LOGMSG message

The third line is determined by whatever follows the LOGMSG command. For example,

```
LOGMSG  My Customized Version from Nov 3, 1984
```

It is recommended that the user always modify the LOGMSG message whenever a **sysdef** file different from the distributed version is used.

ACCESS

The ACCESS command defines the access permissions that newly created files will be given by Cromix-Plus (until the access permission is changed). For example,

```
ACCESS    rewa.r.r
```

The argument following the ACCESS keyword has the same form as for the access utility.

Bufcnt

Cromix-Plus keeps Bufcnt disk blocks in the memory. This implements the idea of a virtual disk: Blocks are read from the main memory and written to the main memory. In the case where a requested block is not residing in main memory, it will actually be read from the disk (or written to the disk to make room for other blocks). Bigger Bufcnt might substantially increase the throughput. The trade-off is that more memory is used (each block takes 544 bytes) and more widespread damage may occur in the case of a power failure. Note that Shell contains the Sync command which will flush buffers at request. Also, the Flush utility should be run in the background.

Inocnt

Cromix-Plus keeps Inocnt inodes in memory. This implements the idea of a virtual disk: Inodes are read from main memory and are written to main memory. In the case where a requested inode is not residing in main memory it will actually be read from the disk (or written to the disk to make room for other inodes). Bigger Inocnt might substantially increase the throughput. The trade-off is that more memory is used (each inode take 144 bytes) and more widespread damage may occur in the case of a power failure.

Filcnt

Filcnt defines the total number of files that can be open simultaneously. Each file structure takes 18 bytes.

Usrcnt

Usrcnt is the number of process tables. Usrcnt defines the maximum number of processes can exist at any time. Each process table takes 1,554 bytes.

Logcnt

Logcnt defines the number of users that can log on the system at the same time. Each user provided for requires 30 bytes.

Mntcnt

Mntcnt defines the number of devices that can be mounted at the same time. Each entry requires 22 bytes.

Lckcnt

Lckcnt defines the number of locks (See _lock system call) that can be installed in the system. 24 bytes are required for each lock.

Freecnt

The system has a small alloc-free mechanism used to obtain small chunks of memory. Freecnt is its size. Do not change it unless you understand the ramifications completely.

END

The END command is a delimiter indicating that the rest of the file is considered to be comment. In this section you will find the names of drivers together with descriptions of arguments (if applicable).

utility: **TAR**
purpose: Creates and retrieves file archives.

user access: privileged users

/dev/sfd2

summary: tar -rxtc[vwfblmkoi] [archive block kilobytes] [filename(s)]

arguments: one per command, from the following list:

-r	add files to end of archive
-x	extract files from archive
-t	list files on archive
-c	create files on new archive

options: options, grouped (as applicable) after one of the preceding arguments.

v	list the name of the file being processed
w	wait for user confirmation before processing
f	next argument is the name of the archive
b	next argument is blocking factor
l	unresolved links reported
m	modification times not restored
k	next argument is archive volume size (in kilobytes)
o	do not check for overwriting of existing archive
i	cancels use of file system identifier on floppy disks

Description

The Tar utility can be used to create file archives on tape, floppy disk, or an ordinary file. Before using Tar to back up files onto floppy disks, the disks must be initialized for the Cromix Operating System. (Refer to the Init utility.) Single- or double-sided, single- or double-density, disks can be used.

Once an archive is created, its contents can also be retrieved using Tar.

Arguments

One of the following for each Tar command:

-r The named files and/or directories are added to the end of an existing archive.

-x The named files and/or directories are extracted from the archive and transferred to the current directory (unless the extracted files are governed by absolute pathnames). The owner, modification time, and access are restored for ordinary files and directories--unless a directory already exists, in which case, the existing characteristics are retained. If no file argument is given, the entire archive will be extracted. For tapes, if there are multiple entries for the same file, the last will overwrite all previous entries.

-t The named files and/or directories in the archive are listed.

-c Creates files on a new archive, but prompts for continuation if existing data will be overwritten.

Options

v Lists each file as it is processed. When used in conjunction with the **-t** argument command, a fuller description of the file is given.

w Prompts for confirmation of each file to be processed. If the user responds by typing **y**, the action is performed; otherwise, it is cancelled.

f Takes the next argument as the archive name. The default archive is **/dev/tp1.** If the name of the file is '-', Tar writes to the standard output or reads from the standard input, as appropriate.

b Takes the next argument as the blocking factor (the number of Tar blocks per tape block) on a new tape archive. Tar blocks are 512 bytes each. The maximum (and default) blocking factor is 16 blocks. Since tape archives are limited to 65,535 tape blocks, the maximum size of a tape archive can vary from 32.7 megabytes to 524.3 megabytes. The blocking factor is determined automatically for existing tape archives.

l Reports when links to the files are not resolved (for use with the **-c** and **-r** arguments). If Tar runs out of memory for the link table, the message:

"No room to check links for file : <file name>"

is displayed, and this file and the files linked to it are written to the archive. (Normally, only one copy of the file is saved).

m Modification times for extracted files will be changed to the time of extraction.

k Takes the next argument as the size of the archive, in kilobytes. This option is useful for splitting large files into separate "volumes" on fixed-size devices such as floppy disks. When creating a multi-volume archive, Tar will prompt for the next volume. When extracting from a multi-volume archive, Tar only prompts for a new volume if a split file has been partially restored. Without the **k** option, Tar will not check for exceeding the disk size. (For use with floppy disks and ordinary files.)

o When used with the **-c** argument, omits the check for overwriting an existing archive. The **o** option must be used when creating a new tape archive over an existing archive of less than 512 bytes.

i When creating an archive on floppy disk, Tar normally puts an 8-byte file system identifier into the first 8 bytes of block 1 (2 bytes for the Cromix version, 3-byte string "tar," 1 zero byte, and 2 bytes for the Tar version). Tar checks for this identifier when adding, extracting, or listing, and prompts for continuation if it is incorrect. The **i** option cancels use of the Tar identifier.

Examples

To view the progress of directory backups to a new archive of one or more small floppy disks, each limited to 390 kilobytes:

```
tar -cvfk /dev/sfdc 390 directory_names
```

To view the progress of a file backup to an existing tape archive:

```
tar -rfv /dev/tp1 file1 file2 ...
```

To list the contents of a tape archive in long form:

```
tar -tvf /dev/tp1
```

Notes

If disk I/O errors occur while reading or writing a floppy disk archive, Tar will attempt to recover. On read errors, Tar will write the block to the Cromix file and display the location where the questionable block was written. On write errors, Tar will stop writing the file, back up to where the file began, and write an end-of-archive at that point. Thus, the volume's prior contents will be intact. Tar will then prompt for a new disk and try rewriting the file.

If the message "Cheksum error" appears, the integrity of the file just processed is questionable.

Tar cannot write to an uninitialized tape. To create an archive on an uninitialized tape, write a dummy file of size 512 bytes or greater to the tape and use the Ddump utility as follows:

```
ddump if=[filename] of=/dev/tp1
```

After the tape is reloaded, Tar can write to the tape.

Shell command:	**TEE**
user access:	all users
summary:	tee [-a] pathname
argument:	pathname
options:	-a append

Description

Tee takes input from the standard input file and sends it to the standard output, as well as to the file specified by the pathname provided in the argument.

Options

The **-a** option appends output to the specified file.

Example:

/dev/prt will tee to printer

```
% sort short | tee sort0
```

This command sorts the file **short**, and sends the sorted output to the terminal (standard output) and to the file named **sort0** in the current directory.

Shell command:	**TERM**
purpose:	This command displays or changes the terminal name.
user access:	all users
summary:	term [terminal_name]
arguments:	optional terminal name
options:	none

Description

Each process is associated with a terminal name, and the name of each terminal is initially defined in the **/etc/ttys** file. To display the name of the current terminal, enter the Term command with no arguments; to change the terminal name, enter the Term command with the name of the new terminal.

Terminal names are case sensitive, and should be four characters long. The terminal name must be defined by a set of terminal attributes in the **/etc/termcaps** file. If your terminal is not defined in the **/etc/termcaps** file, you must update the file (refer to the section on TERMCAPS).

Data base:	**TERMCAPS**
purpose:	The file /etc/termcaps is a data base describing various terminal capabilities.

Description

The **/etc/termcaps** file lists the terminal capabilities of various terminal names. Refer to the Term command to display or change the name of your current terminal (a user program can read the terminal name with the **ustat** system call). The terminal name is initially obtained from the **/etc/ttys** file, and is passed from one process to another. If necessary, a user program may read the **/etc/termcaps** file and adjust its output for a particular terminal.

Termcap File Structure

Each terminal description in the **/etc/termcaps** file consists of one or more lines, each composed of fields separated by colons. Except for the last line, all lines in a description are terminated by a backslash. Blank lines and lines with a pound sign (#) in column 1 are ignored as comment lines. For readability, empty fields can be inserted (blank space ending in a colon) anywhere in the description.

The first field of each description is the terminal name, and consists of three subfields separated by vertical bars. The first 2-character subfield can be used to identify the terminal. The second, 4-character subfield is the official terminal name (used with the Term command or entered in the **/etc/ttys** file). The third subfield is arbitrary and may be any name that fully describes the terminal.

The remaining fields start with a case sensitive, two-character capability name. Capability names described here or in Section 5 of the UNIPLUS System V User's manual are reserved; all others are arbitrary.

A capability field takes one of three forms: boolean, numeric, or string. In the boolean form, a capability name (followed by a colon) indicates that the given terminal has the specified capability. In the numeric form, a capability name is followed by a pound sign (#) and a decimal number, while in the string form the name is followed by an equal sign and a character string. If the first character after the equal sign is a decimal digit, it indicates the number of NULL characters to be added at the end of the string.

Special characters may be imbedded in the string:

\n	newline character
\r	RETURN character
\t	TAB character
\b	backspace character
\f	formfeed character
\E	ESC character
\xxx	the character with octal value xxx
\<char>	character <char>, allows using a backslash or up arrow character in a string
^<char>	character <char> anded with 0x1f

Escape sequences, such as those used for cursor movement, are first encoded by a function similar to printf. Thus, arguments can be inserted in the string at appropriate places. The following forms are recognized:

%d	same effect as printf
%2	same effect as %2d in printf
%3	same effect as %3d in printf
%.	same effect as %c
%+x	add x to value then %
%>xy	if value > x add y; takes effect at next %
%r	reverse next two arguments
%i	increment all arguments by 1
%%	single %

The currently defined capability names are as follows:

al str Insert line at cursor.

cd str Clear to the end of screen.

ce str Clear screen.

cl str Clear to the end of line.

cm str Cursor addressing sequence.

co num Number of columns on the screen.

ct str Toggle cursor on-off.

dc str Delete character at cursor position.

dl str Delete line at cursor.

do str Move cursor down one line.

ei str Exit insert line mode.

fc str Turn cursor off (make it invisible).

Ic str Insert a space (page mode) at cursor position.

ic str Insert a space (line mode) at cursor position.

il str A string specific to the Cromemco C5 terminal. It must be followed by character count and that number of characters. These characters overwrite the leftmost column of the screen.

im str Enter insert line mode.

ir str A string specific to the Cromemco C5 terminal. It must be followed by character count and that number of characters. These characters overwrite the rightmost column of the screen.

kE str Character used to turn off insert mode for Shell retype capability.

kI str Character used to turn on insert mode for Shell retype capability.

kR str Character used for Shell retype capability, and serves as a flag to indicate if retype exists. If "kR" is defined, the following capabilities must also be defined: kI, kE, kl, kr, nb, nd, pd, pi, pe.

kl str Character sent by left arrow key.

kr str Character sent by right arrow key.

li num Number of lines on the screen.

nb str Nondestructive backspace (move cursor left).

nd str Nondestructive space (move cursor right).

oc str Turn cursor on.

pe str String to turn off the "pi" insert mode.

pi str String to enter page insert mode (subsequent typing pushes down the remaining characters on the screen).

pd str String to delete character at cursor position in page mode (the remaining characters on the screen are moved up).

rs str Remove standout.

sd str Select a screen for display.

se str Exit standout.

so str Enter standout. Sequence to turn on some highlighting.

sq str Enquiry - which screen is currently active.

sw str Select a screen for write.

up str Move cursor up one line.

wi str Write with invisible cursor at given position ("we" must also be set).

we str Terminating string for write with invisible cursor. An invisible write consists of the "wi" string, the string to be written, and the "we" string.

wf str Turn wrap off. Insert in page mode will not shift the last character of each line to the first position of next line.

wo str Turn wrap on. Insert in page mode will shift last character of each line to the first position of the next line ("wf" must also be set).

Example

The termcap file entry for the Cromemco C5 terminal is shown below:

```
# Cromemco C-05 Terminal
#
C5|C-05|Cromemco C-05 Terminal:\
          :co#80:li#24:\
          :up=\EA:do=\EB:nd=\EC:nb=\ED:\
          :dl=\EM:al=\EL:\
          :cm=\EF%+ %+ :\
          :cl=\EE:cd=\EJ:ce=\EK:\
          :dc=\EP:ic=\EQ \E@\ED:\
          :am:\
          :Ic=\Ea \E@\ED:im=\EQ:ei=\Eo@:\
          :pd=\E`:pi=\Ea:pe=\E@:\
          :wi=\E\^%+ %+ :we=^]:\
          :wo=\E.L:wf=\E.J:\
          :il=\E.O:ir=\E.P:\
          :sq=\E.o:sw=\E.W%+0:sd=\E.U%+0:\
          :oc=\Er:fc=\Eq:ct=\EZ:\
          :so=\EdP:se=\Ed@:rs=\Ee:\
          :kR=^R:kI=^I:kE=\E:\
          :kl=^H:kr=^L:ku=^K:kd=^J:
```

The "ic" string is a "program" that inserts a space in line mode by using the strings for line insert ("im"), exit line insert ("ei"), and backspace ("nb"). Similarly, the "Ic" string inserts a space in page mode using the "pi", "pe", and "pd" strings.

The "wi" string (write with invisible cursor) is similar to cursor addressing ("cm"), and includes an exit string ("we"). The "wo" and "wf" strings (C5 and C10 only) turn word wrap on and off for insert/delete in page mode. The "il" and "ir" strings (C5 only) insert left and right columns for horizontal scroll (used by the CE editor).

The "sq" string queries the C5 for the currently active screen, and "sw" and "sd" select the nth screen for write and display, respectively. The "oc", "fc", and "tc" strings define the cursor on, cursor off,and cursor toggle sequences. The "so" "and "se" strings turn "standout"(highlighting) on and off (inverse video used here); unlike "se", "rs" removes standout without inserting the normal attribute.

utility:	**TESTINP**
purpose:	This program tests for equality between the contents of a file and a particular string.
user access:	all users
summary:	testinp [-dfr] file string(s)]
arguments:	file pathname one or more strings
options:	-d delete -f compare file after test first characters -r reverse sense of test

Description

This utility compares the contents of a file to a string or strings and sets an error return code if one of the strings does not match the contents of the file specified.

The test made by Testinp is case insensitive. Testinp tests for equality between the file and the string. To locate text strings embedded in the text of a file, use the Match utility.

Options

The **-r** option reverses the sense of Testinp by setting the error code if a match **does** occur.

The **-f** option checks only the first character of the file passed as an argument against the first character of each of the control strings.

The **-d** option deletes the file passed as an argument after the test. This option is useful in many command files using a temporary file created during the command file execution.

Example:

```
echo "Do you want to shut down the system?"
input > temp
testinp -d temp YES OUI SI
if -err goto noshutdown
kill -2 1
%noshutdown
```

The example above is a typical command file that uses Testinp and Input. The first line sends the string within quotation marks to the standard output. The second line uses the Input utility to send the user's response to the file **temp**. On the third line, Testinp tests the contents of the file **temp** for occurrences of the strings YES, OUI, or SI. Testinp then deletes **temp**. If the file contains one of the control strings, the system is shut down using the Kill command. If the file **temp** does not contain one of the control strings, Testinp sets an error code. The command that follows passes control to the label noshutdown. If the user answers **no** to the question, the system is not shut down.

utility:	**TIME**
purpose:	This program displays or alters the time and date.
user access:	all users for display privileged user for changes
summary:	time [-se2]
arguments:	none
options:	-s set system values -2 see 3102 clock -e European style display (dd/mm/yy)

Description

The Time program displays or changes the time and date. If the **-s** option is not specified, the current date and time are displayed. If the **-s** option is used, the user is prompted for the date and then the time. Although the date is displayed with the **/** separator, and time is displayed using the **:** separator, any convenient separator character (such as a space or a period) can be used when entering the date and time. The Time utility displays a new prompt until a valid date or time is entered.

Options

The **-s** option causes the user to be prompted for a new date and time. Only a privileged user can specify the **-s** option.

The **-e** option causes the time to be displayed in the European style, with the day and month reversed.

The **-2** option allows the user to set the clock in the user's terminal. The terminal must be a Cromemco 3102, C-5, or C-10 terminal.

Notes

The Time utility allows shortcuts when the date and time are entered.

To enter date:

No value given	Keep old values.
One value	Used for day. Month and year unchanged.
Two values	Used for month and day (day and month in the European style). Year unchanged.
Three values	Month, day, and year (day, month, and year in the European style).

To enter time:

No value given	Keep old values.
One value	Used for minutes. Keep old hours, set seconds to zero.
Two values	Used for hours and minutes, set seconds to zero.
Three values	Hours, minutes, seconds.

utility:	**TOUCH**
purpose:	This program changes the modification times of files to the current time.
user access:	all users
summary:	touch [-c] filename(s)
arguments:	pathnames of one or more files
options:	-c do not create files if they do not exist

Description

The Touch program changes the modification time of a file to the current time. If the file does not exist, Touch creates the file unless called with the **-c** option.

The Touch program is for use with the Make program.

Options

The **-c** option prevents Touch from creating files if they do not exist.

Shell command:	**TYPE or TY**
purpose:	This command displays an ASCII (text) file.
user access:	all users
summary:	ty [file-list]
arguments:	optional file pathnames
options:	none

Description

The Type command displays the file(s) specified by the pathname(s). Type can be used only to display ASCII (text) files. To display other kinds of files, use the Dump utility.

Type uses STDIN when called without an argument, and sends output to STDOUT.

Example:

```
# ty /dev/qtty5 > diskfile
```

This command line accepts data from **/dev/qtty5** and sends it to **diskfile**.

file:	**UBOOT.SYS**
purpose:	Standalone boot program for the UNIX Operating System
user access:	privileged user
summary:	boot uboot
arguments:	uboot
options:	none

Description

The file **uboot.sys** is a standalone boot program for the UNIX Operating System. Uboot reads the UNIX kernel from the default device (std(2,0)unix). If you press the ESC key immediately after giving the "boot uboot" command, you will be prompted for the device name, the major and minor device numbers, and the filename. Currently supported devices are:

std	STDC disk
sfd	small floppy
fd	large floppy

Uboot.sys may be patched to use another default device. For example to change the default device to std(33,0) use the following sequence of commands: (User input is in boldface).

```
# patch uboot.sys
> q Osfffff 'std('
00001de0: 73 74 64 28   20 32 2c 30 - 29 75 6e 69   78 00 73 74 std( 2,0)unix.st
> s 1de4
00001de4:  20 '33'
00001de6:  2c .
> e
#
```

The value entered in the substitute command (s 1de4) is the value displayed by the query command plus 4.

utility:	**UNMOUNT**
purpose:	This program disconnects a mounted file system from the current file system.
user access:	privileged user
summary:	unmount devname [-x]
arguments:	device name
options:	-x do not eject disk

Description

The Unmount utility program disables access to a file system. A file system that has been mounted **must be unmounted** by use of the Unmount utility **before the mounted disk is removed from the system or the system is powered-down.** If this is not done, the integrity of the data on the mounted system cannot be assured.

Options

The **-x** option causes a floppy disk not to be ejected when it is unmounted.

utility: **UPDATE(1...2)**
purpose: These command files update an existing Cromix file system from new release diskettes.

user access: privileged user

summary: device name

options: none

Description

The Update programs update an existing Cromix file system from new system diskettes.

After booting the system from disk 1 and rooting on disk 1, run Update1, using the following command syntax:

```
# update1 drive
```

where **drive** is the destination drive.

Update1 will reboot the system. Boot your newly updated drive and run update2 for all additional new system disks using the following command syntax:

```
# update2 drive
```

where **drive** is the source drive.

Because the Update programs are command files, you can refer to the files themselves for information about how they work. Command files are ordinary text files. Thus, you can display them with Type or even change them with the Screen editor.

utility:	**USAGE**
purpose:	This program displays directory size information.
user access:	all users
summary:	usage [file-list]
arguments:	directory or file pathname(s)
options:	none

Description

The Usage utility displays the physical disk space (in blocks) and the logical file space (in bytes) occupied by a directory and all of its descendants. If only a single file is specified, the size of that file is reported. If no pathname is given, the current directory is assumed.

Knowing the number of blocks occupied by a directory is useful before using the Cptree utility.

utility:	**VDT**
purpose:	Special terminal functions
user access:	all users
summary:	vdt function
arguments:	keyword which identifies function
options:	none

Description

The argument to the Vdt utility is a keyword which identifies the actual command. The commands available are:

clear	Clear screen.
cursor	Turn cursor on.
cursoff	Turn cursor off.
wrapon	Turn wrap around on.
wrapoff	Turn wrap around off.
clock	Set hard clock.

If the keyword is followed by a message, the message is loaded into status line of the terminal. if there is no message, the status line is cleared.

Note

The Vdt command is intended to be used only on a C-5 or a C-10 terminal.

utility:	**VERSION**
purpose:	This program displays the version number of the Cromix Operating System or a utility program.
user access:	all users
summary:	version [file and/or directory list]
arguments:	optional file and/or directory list
options:	-c calculate CRC -v verbose

Description

When called without an argument, the Version utility displays the version of the Cromix Operating System being run. A simple check for internal consistency is also performed. If the consistency check reveals problems, an appropriate message is displayed instead of the version number. Reboot the system as soon as possible.

When called with the name of a utility program, Version displays the version number of that utility. When called with a directory name, Version displays the version number of each of the programs in the directory. The following command displays the version numbers of all of the programs in the **/bin** directory:

```
% version /bin
```

The characters **RB** appearing in an entry indicate that the file is a Relocatable Binary file. Programs written for Z80 Cromix (including relocatable binary files) are allocated 16 pages (64k) of memory by the Cromix-Plus Operating System.

The characters **68** appearing in an entry indicate the file is a 68000 binary file. For these programs, the operating system allocates as many 4k pages of memory as required.

The Version utility (version 0.10 and higher) searches for the following string of bytes in a file: 0FDH 0EDH 0FDH 0EDH. The bytes immediately following are assumed to contain CRC information, version number, release number, and an optional login message.

To illustrate, consider the following portion of a file:

```
dc.b     0FDH 0EDH 0FDH 0EDH
dc.b     0, 0, 0, 0
dc.b     version, revision
dc.b     'login message \n\0'
```

When called with the **-c** option, Version computes a CRC value for the file. The first two bytes immediately following the FDEDFDED pattern contain the version number of the Version program; the next two bytes contain the CRC.

When called without the **-c** option, Version reports good or bad file integrity by comparing a newly calculated CRC with the previously calculated CRC value.

Options

The **-c** option causes the CRC value calculated for the file to be placed in the file for future comparison. The **-v** option causes the pathnames of files to be printed.

Notes

In checking system consistency, Version considers the system corrupted if someone, while using Debug68, aborted the program instead of using the Exit command. The system is unharmed. The situation may be bothersome, however, because the Restrp program cannot remove the error.

Shell command:	**WAIT**
purpose:	This command suspends execution and waits for the PID-specified process to terminate.
user access:	all users
summary:	wait [PID]
arguments:	optional PID number
options:	none

Description

The Wait command causes the Cromix Operating System to suspend operation until the process specified by the process id number (PID) has terminated. If no process is specified, Wait suspends execution of the current process until all detached processes belonging to that user have terminated.

utility:	**WBOOT**
purpose:	Writes the boot program to the boot area of a disk.
user access:	privileged user
summary:	wboot devname [pathname]
arguments:	device name where the boot program is to be written.
	optional pathname of the boot program to be written.
options:	none

Description

The Wboot utility writes the boot program into the boot area of a device. This is necessary for a device to be bootable. Devices available are floppy disks, WDI-II disks, and STDC hard disks (though RDOS prior to version 03.12 cannot access them). The boot program to be written is selected by the Wboot utility. The programs required must reside in the **/etc** directory (fdboot, sfdboot, hdboot, stdboot). As an alternative, a user can write his own boot program if he specifies the appropriate filename as the second argument.

Wboot uses the system drivers to access the device, so verify that there is an entry in **/dev** for the device name, and that you have run **crogen** to include the system drivers.

utility: **WHO**

purpose: This program displays a list of users who are currently logged in.

user access: all users

summary:

```
who [/etc/account]
    [am i]
```

arguments: optional /etc/account

or

optional am i

options: none

Description

When the Who utility is called without an argument, the **/etc/who** file is consulted and a report is displayed showing the users currently logged on, together with the time each one logged on.

When followed by **am i**, the name of the user calling the Who utility is displayed.

If the Who utility is called followed by **/etc/account**, the information contained in the account file is displayed.

Reader Responses To This Documentation

Dear Reader,

We have made a sincere effort to provide you with the information you need in this manual. If you should find the documentation deficient or in error, let us know so we can correct it. We appreciate and value your response; it will be useful in improving the documentation. Please detach and use the Reader Response Card below to send us your comments.

Thank you for your time and interest in Cromemco products.

Sincerely,
Winthrop A. Stiles III
Technical Publications Manager

(Detach Here)

(Detach Here)

Cromemco®

Reader Response Card

To: Winthrop A. Stiles III,
Technical Publications Manager

Re (Manual title):______________________________

My System is (Specify configuration):______________________________

The following information is incorrect (Please specify page number):______________________________

____

____

____

____

____

____

(Fold Here)

The following additional information would be helpful:______________________________

____

____

What general suggestions do you have for improving this manual?______________________________

____

____

If you need a response from Cromemco, please print your name, mailing address, and telephone number:

Name:______________________________

Address:______________________________

____

____

Telephone: (   )______________________________

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 599 MOUNTAIN VIEW, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Cromemco®

Attn: Winthrop A. Stiles III
Technical Publications Manager
280 Bernardo Avenue
P.O. Box 7400
Mountain View, CA 94039

(Fold as indicated and tape the edge)

Cromemco®

280 Bernardo Ave.
P.O. Box 7400
Mountain View, CA 94039