COHERENT Update Release Notes

Release 3.1.0

Mark Williams Company
60 Revere Drive
Northbrook, Illinois 60062
Telephone: (708) 291-6700

Printed in U.S.A.

Revision 3.1.0 Printing 5 4 3 2 1

Published by Mark Williams Company, 60 Revere Drive, Northbrook, Illinois 60062.

E-mail:	uunet!mwc!support	(Technical Support)
	uunet!mwc!sales	(General Information)
BIX:	join mwc	
CompuServ:	76256,427	

Printed in the U.S.A.

Contents

Section 1:
Release Notes

This document contains the release notes for COHERENT update version 3.1.0. Be sure to read this document carefully before you begin the update process.

Section 2 below is a compatibility list. It describes hardware reported to work with this release of COHERENT, and gives a list of incompatible hardware. Please check these lists prior to installing your COHERENT update.

Section 3 below contains Lexicon articles that have changed since the initial 3.0.0 release of COHERENT. These include changes to the Lexicon articles themselves, programs which have been enhanced as well as new additions to the COHERENT system.

Enhancements

The 3.1.0 update package includes:

- Information on accessing the MWC UUCP Bulletin Board System
- Configurable AT hard disk driver for unsupported disks
- Adaptec AHA-154x series SCSI device driver
- COM3 and COM4 serial line support
- RAM-disk device driver
- A generic multi-port serial card device driver
- New **cpio** and **tar** file archiving utilities
- A video arcade style game called **chase**
- Elvis v1.4 — a **vi** and **ex** clone
- New "miscellaneous" function library with source code
- Enhanced **mail**, **scat**, **man** and **msgs** utilities
- 16-bit versions of **compress**, **uncompress** and **zcat**
- Enhanced **fsck** supporting filesytems larger than 35MB
- New and updated on-line manual pages
- Numerous bug fixes and minor enhancements

Installing the COHERENT Update

Before attempting to install the COHERENT OS Update, be sure that you thoroughly read the release notes.

In order to perform the installation, you must first log in as **root** (the superuser).

To install the COHERENT Update from a high density 5.25 inch distribution in dri enter the following command:

```
/etc/install CohUpd310 /dev/fha0 1
```

Please note that the last three characters of the first argument are **numeric** represent the version number of the update you are about to install.

To install the COHERENT Update from a high density 3.5 inch distribution in dr enter the following command:

```
/etc/install CohUpd310 /dev/fva0 1
```

The installation program will prompt you to insert the write protected floppy disk drive 0. After the installation completes, place your distribution disk in a safe away from heat or magnetic fields.

The MWC Bulletin Board System

Mark Williams Company has set up a COHERENT UUCP node for dial-up use b customers. We expect it to serve three major purposes:

1. As a remote site for customers to test their UUCP configuration by calling the node.
2. As a source of news, bug fixes, and public domain software for download customers. In addition, the system provides a listing of "third party" so and hardware which works with COHERENT.
3. As a mail drop for customers who request an account, where correspon (electronic mail) from MWC and other customers can be picked up.

Accessing the MWC BBS

You can use any conventional 1200- or 2400-baud modem or a 9600-baud Trail modem. Examples below will assume a 2400-baud Hayes-compatible modem con to COM1 unless stated otherwise. See your modem instruction manual for more mation if your modem is not Hayes compatible.

You should be able to communicate locally with the modem via the COHERENT k command. The following example asks for current settings of all modem registe suming that the modem is not enabled for logins and that you are running as ro superuser). Note that the following session uses the "local" or "non modem c version of the serial port device. User commands are shown in bold.

```
$ kermit cbl 2400 /dev/com1l
kermit: connected...   <== this line is printed by kermit
at&v                   <== use atn? for the Trailblazer
... list of modem register contents follows ...
(type ^c - caret followed by letter "c" - to exit kermit)
```

The modem should be configured for "no echo" and "terse" mode. Usually, these are dicated by settings "E0" and "V0" for Hayes-compatible modems.

In order to properly handle operation of the modem, you must use a cable wh provides "full modem control", ie: connects all the modem control signals from y modem to your serial port. A simple "3-wire" connection will not work correctly! In dition, the status of the "carrier detect" line from the modem should reflect true mod carrier detect (CD) status (ie: be sure the modem is not programmed to assert CD all time, and that the modem cable does not have CD strapped to some other signal at end or the other). Similarly, the modem must force a "disconnect" when the "data minal ready" (DTR) signal is deasserted by the computer. Failure to observe th pratices may cause the modem to not properly disconnect in certain situations. See UUCP tutorial "*UUCP: Remote Communications Utility*" in the COHERENT refere manual for further details regarding cabling and setting up UUCP.

There must be a valid entry in **/etc/ttys** for the modem. Continuing with our exam the following would be typical:

```
$ ls -l /etc/ttys
-rw-r--r--  1 root     root      128 Wed Oct 10 11:06 /etc/t
$ cat /etc/ttys
11Pconsole
1rLcom1r
11Pcom21
```

During the initial set-up of UUCP, you should insure that no stale UUCP control or d files exist in directory **/usr/spool/uucp**. For example, to remove any stale files, en the command:

```
$ rm -f /usr/spool/uucp/LCK* /usr/spool/uucp/TM*
```

In addition, you must insure that the UUCP subsystem has access to your modem devi For example, entering the following command changes the permissions on all variants our sample modem port:

```
$ chmod 666 /dev/com1*
```

File **/usr/lib/uucp/L-devices** should contain an entry for your modem. Newer d tributions of COHERENT are shipped with sample entries for low-speed (1200 or 24 baud) and high-speed (9600 baud Trailblazer) modems commented out. E **/usr/lib/uucp/L-devices** to match your equipment configuration. The following sho typical entries for the sample configuration:

```
$ cd /usr/lib/uucp
$ ls -l L-devices
-rw-r--r--  1 uucp    uucp     197 Wed July 18 20:54 L-device
$ cat L-devices
#type    line     disable baud   brand
#----    -----    ------- ----   ------
ACU      com1l    com1r   2400   hayes
#ACU     com1l    com1r   9600   tbfast
DIR      com2l    com2l   9600   direct
```

File **/usr/lib/uucp/L.sys** should have an entry for the MWC BBS node whose sitenam is **mwcbbs**. Newer distributions of COHERENT are shipped with sample lines co mented out. Edit the **mwcbbs** entry to suit your equipment; be sure to replace the te **SERIALNUM** with your 9-digit COHERENT serial number, or you will not be able access the BBS. The following access information will be used to generate the **mwcb** entry in your **/usr/lib/uucp/L.sys** file:

```
Modem Speed and Type      BBS Phone Number
--------------------      ----------------
2400 baud generic          1-708-559-0412
9600 baud Trailblazer      1-708-559-0445

Expect String       Send String
-------------       -----------
""                  \r\d\r
in:--in:            nuucp
word:               public
word:               (your COHERENT serial number)
```

Please note that in the example below, entries are continued over multiple lines; in actual file, each entry must be on a single line, but the line may exceed 80 character length.

```
$ ls -l L.sys
-rw-r--r--  1 root   root     269 Tue Oct  9 16:18 L.sys
$ cat L.sys
...
mwcbbs       Any ACU 2400 17085590412 (continued on next line)
 "" \r\d\r in:--in: nuucp word: public word: SERIALNUM
#mwcbbs      Any ACU 9600 17085590445 (continued on next line)
 FAST \r\d\r in:--in: nuucp word: public word: SERIALNUM
```

Finally, file **/usr/lib/uucp/Permissions** should have a **MACHINE** entry for **mwc** Normal use of the BBS is in anonymous UUCP mode, in which all callers appear to a sitename of "bbsuser". If you have version 3.1.0 or later of COHERENT, **MYNAME** keyword in **/usr/lib/uucp/Permissions** allows you to appear to the c site as "bbsuser". If you are still running version 3.0.0, you will have to change the tents of **/etc/uucpname** to be **bbsuser**. Newer distributions of COHERENT incl

sample entry for the Mark Williams BBS node in file **/usr/lib/uucp/Permissions**.

```
$ ls -l Permissions
-rw-r--r--  1 root    root      359 Tue Oct  9 01:38 Permissions
$ cat Permissions
...
MACHINE=mwcbbs MYNAME=bbsuser \
        COMMANDS=rmail:uucp \
        READ=/usr/spool/uucppublic:/tmp \
        WRITE=/usr/spool/uucppublic:/tmp \
        REQUEST=yes SENDFILES=yes
...
```

Downloading Files

For your initial contact with **mwcbbs**, we recommend that you download the introductory message **howto.start** from directory **/usr/spool/uucppublic/mwcnews**. For example, the command:

```
$ uucp mwcbbs!/usr/spool/uucppublic/mwcnews/howto.start /tmp
```

will queue up a request to copy file **howto.start** to the **/tmp** directory on your system. This file contains information on available downloads, new COHERENT developments, and how to obtain an individual account on the system. If the time at which you type the **uucp** command is within the range of times allowed in the **/usr/lib/uucp/L.sys** entry, your modem will call **mwcbbs** immediately — you can check this with the **ps** or **uulog** commands. If you wish to force polling of mwcbbs at a time not specified in **/usr/lib/uucp/L.sys**, you can do so with the command:

```
$ su uucp /usr/lib/uucp/uucico -Smwcbbs &
```

You can then monitor the progress of the transfer via the command:

```
$ uulog -f mwcbbs
```

To stop monitoring the call, enter a **<ctrl-C>**.

Electronic Mail Access

The following electronic mail addresses are available to customers. For issues relating to administration of the BBS node, you can send mail to **mwcbbs!admin**. For matters relating to support of COHERENT and related products, send mail to **mwcbbs!support**. Additional mail aliases may appear from time to time. Please remember that mail sent to either of the aforementioned addresses will only be sent to **mwcbbs** when **uucico** is invoked. This can be accomplished from the command line, as shown in the example above, or at regular intervals as a cron task in **/usr/lib/crontab**. A sample **/usr/lib/crontab** line is:

```
15 20 * * * su uucp /usr/lib/uucp/uucico -smwcbbs
```

This line will check every day at 8:15 PM for **uucp** and **mail** requests which have bee queued for **mwcbbs**, and, if any are found, will call **mwcbbs**. See the **cron** Lexicon a ticle for further information about **/usr/lib/crontab**.

Please note that while the BBS supports inbound electronic mail from your system t **mwcbbs** via anonymous UUCP, we have no mechanism for replying to your electroni mail messages unless you have an account on **mwcbbs**. To request an account o **mwcbbs**, request the introductory article on the BBS called **howto.start** from directory **/usr/spool/uucppublic/mwcnews** on machine **mwcbbs**.

Related Reading

For further discussions on UUCP, networking and electronic mail, see the **UUCP** Lexicon article and the UUCP tutorial in the COHERENT reference manual.

Section 2:
Compatibility Information

This section details machines, add-on cards and BIOS ROMs reported to work with th COHERENT operating system, as well as those reported not to work.

Before you continue, please note the following caveats:

First, this is only a partial list of the hardware on which COHERENT runs. We receiv confirmation of new machine configurations on an almost daily basis. If you believe tha you have a machine, BIOS or add-on board that is **not** compatible with COHERENT bu is listed below, please call our technical support department.

Second, manufacturers make changes to their hardware as part of redesigns or produc improvements. These can include logic, timing, firmware or functionality changes While efforts have been made to support tested products, Mark Williams Company can not guarantee compatibility with products not under its control.

Lastly, if you believe that your computer cannot run COHERENT, please contact th Mark Williams Company technical support department.

Systems

The following systems have been tested with COHERENT. Note that configuration vary, especially with respect to disk controllers, so not all possible configurations hav been tested.

```
ABM AT
Acer 910, 1100, 1116
AGI 1800A, 3000D, 3000G
AGL 286-12
ALR PowerFlex, 386SX, 386/220
American Semiconductor 286 PC
AMI 386SX, 386
Arche 386/25
AST Premium 286, 386/33
```

AT&T 6386
Austin 386SX, 386/33
Bentley 286
Bitwise 33-386 Portable
Bondwell 286 Laptop
Cheetah International i486/25
Club AT, 1800
Commodore 286
Compaq 286, 386, 386 Portable
Compaq SLT 286, LTE/286
CompuAdd 286-10, 286-12
CompuAdd 216, 220, 320, 325
Computer Directions 386SX
Comtex 386/20
Dataworld 386/33
Dell System 210, 220, 300, 310, 325
DTK PEM-2000 386
Dyna 386/20
EDP 386SX
Emerson 8286ECV
EPS 386
Epson Equity II+, III+
Executive AT-286
Five Star 386/20
Gateway 2000 (RLL and ESDI)
GCH 386 AT
Hauppauge 386
HP Vectra RS/20 (ESDI), ES/12, QS/20
Hyundai LT3/286
IBM PC/AT (286)
Jameco 3550
JDR M386
Leading Edge 386, D3, 6000
Logix 386-25
MAXAR 386
Micro-1 386
Micro-Designs 386, 25MHz
Micro Express 386
Micronics 386
Mitsubishi 286L, 386
MultiTech 900
MYLEX MWS386, 25 MHz
NCR 386, PC-810
NEC 386/25, Powermate 386/20, 386SX

Northgate 286/20, 386/16, 486
Olivetti M280, H28, M380
Omega 386/20
Optima 386
Packard Bell Axcel 386SX, PB900
Packard Bell Pack-Mate, Legend V
PC Brand 386/20, 386/25
PC Designs ET 286
PC's Limited AT
PC Pros 486
PC Systems 386-20
PeaCock 286 AT
Pulse 386-SX
Samsung 5550, 5800
Schneider Euro AT
SEFCO 16 MHz 386SX
Sharp 5541
Siemens 750
Smart Micro 286, 386
Standard Brands 386-25, 386/SX
Sunnytech 386-20
Sys Technologies 386
Tandon 386/20, 386/33
Tandy 3000HL, 3000HD, 4000
Televideo AT 8MHz
Telex 1280
Tera-Tek 386
Touche' 5550T
Tri-Star 386
Unisys 2850, 286 PW
UTI 386
Victor 386
Wang PC 240 AT, PC 350, PC 381
Wells American AT, 14 MHz
Wyse 2108, 2112, 2200, 3216
Zenith 248, SuperSport 286
Zenith TurboSport 386, 386/33
ZEOS 286, 386, 386SX, 386 Portable

Add-On Boards

The following add-on boards have been tested with COHERENT. Note that board and firmware revisions may vary. Not all possible configurations have been tested.

```
Adaptec AHA-1540A, AHA-1542A SCSI H/A
Adaptec AHA-1540B, AHA-1542B SCSI H/A
Arnet Multi-8 8 port serial
Arnet COM4 QUAD RS-232, PLUS4 QUAD RS-232
BTC 1505 Monochrome Graphic Printer Card
Data Technology DTC7287 RLL 1:1
DTK PTI-217 IDE HD/FD
DTK Graphicsmith
DTK PEI-301 32-bit memory expansion
Emulex DCP/MUX
Geesee Trading PC-COM 4 port serial
IBM monochrome printer card
National Computer Ltd NDC545 MFM
SEFCO serial adapter
SEFCO monochrome adapter
Ultrastore Ultra 12 ESDI
Western Digital WD1006V-MM2 1:1 MFM
Western Digital WD1006V-SR2 1:1 RLL
Western Digital WD1007 ESDI
```

BIOS ROMs

The following BIOS ROMs have been tested with COHERENT.

```
AMI 286
DTK 386
IBM AT (286)
PHOENIX 386
PHOENIX 386SX
```

The following BIOS ROMs are known **not** to work correctly with COHERENT.

```
AMI 386
```

When running protected mode software, the AMI 386 BIOS fails to reset the system correctly when rebooting via a <CTRL-ALT-DEL> key sequence. Use the RESET button in order to correctly reset your system.

Incompatible Hardware

The following hardware is known **not** to work with this release of COHERENT.

```
AT&T 6300
IBM MicroChannel PS/2 models
Perstore RLL hard disk controllers
Western Digital WD1004-27X (XT)
XT disk controllers
```

Section 3:
The Lexicon

The following pages contain Lexicon articles that have changed or been added sin
release 3.0.0 of COHERENT.

aha154x — Device Driver

Adaptec AHA-154x device driver

The **aha154x** device driver allows the user to use SCSI interface devices attached t
Adaptec AHA-154x series host adapter.

The COHERENT **aha154x** device driver has major number 13. It can be acce
either as a block-special device or as a character-special device. The minor nun
specifies the device and partition number for disk-type devices, allowing the use of u
8 SCSI-IDs with up to 4 Logical Unit Numbers (LUNs) per SCSI-ID and up to four
titions per LUN.

The first **open** call on a SCSI disk device allocates memory for the partition table
reads it into memory.

Minor Device Numbers

The minor device number is decoded as follows:

```
Bit number:     7 6 5 4 3 2 1 0
Meaning:        S I I I L L P P
```

where *S* indicates the "special" bit, *III* indicates a three-bit field containing the SCSI-
in the range of 0 through 7, *LL* indicates a two-bit field containing a LUN in the range
0 through 3, and *PP* indicates a two-bit field containing either a partition number for d
k-type devices or a set of special modes for devices other than disks.

The "special" bit and the partition number interact as follows:

Description	S Bit	PP	Device	Type
partition a	0	00	/dev/sd?a	disk
partition b	0	01	/dev/sd?b	disk
partition c	0	10	/dev/sd?c	disk
partition d	0	11	/dev/sd?d	disk
partition table	1	00	/dev/sd?x	disk
no rewind	1	01	/dev/sd?n	tape
RESERVED	1	10	---	---
rewind on close	1	11	/dev/sd?	tape

Loading The Driver

The **aha154x** loadable device driver must be loaded a on system which does not have
SCSI hard disk as the root device via the **/etc/drvld** command, as follows:

```
/etc/drvld -r /drv/aha154x
```

Files

/dev/sd* — block-special devices
/dev/rsd* — character-special devices

See Also

device drivers, drvld, scsi

Notes

This release of the **aha154x** device driver only supports disk-type devices. A future version of the driver will add support for tape-type and other devices.

array — Definition

An **array** is a concatenation of data elements, all of which are of the same type. All the elements of an array are stored consecutively in memory, and each element within the array can be addressed by the array name plus a subscript.

For example, the array **int foo[3]** has three elements, each of which is an **int**. The three **int**s are stored consecutively in memory, and each can be addressed by the array name **foo** plus a subscript that indicates its place within the array, as follows: **foo[0]**, **foo[1]**, and **foo[2]**. Note that the numbering of elements within an array always begins with '0'.

Arrays, like other data elements, may be automatic (**auto**), **static**, or external (**extern**).

Arrays can be multi-dimensional; that is to say, each element in an array can itself be an array. To declare a multi-dimensional array, use more than one set of square brackets. For example, the multi-dimensional array **foo[3][10]** is a two-dimensional array that has three elements, each of which is an array of ten elements. The second sub-script is always necessary in a multi-dimensional array, whereas the first is not. For example, **foo[][10]** is acceptable, whereas **foo[10][]** is not. The first form is an indefinite number of ten-element arrays, which is correct C, whereas the second form is ten copies of an indefinite number of elements, which is illegal.

You can initialize automatic arrays and structures, provided that you know the size of the array, or of any array contained within a structure. An automatic array is initialized in the same manner as aggregate, but initialization is performed on entry to the routine at run time, instead of at compile time.

Flexible Arrays

A **flexible array** is one whose length is not declared explicitly. Each has exactly one empty '[ ]' array-bound declaration. If the array is multidimensional, the flexible dimension of the array must be the *first* array bound in the declaration; for example:

```
int example1[][20];/* RIGHT */
int example2[20][];/* WRONG */
```

The C language allows you to declare an indefinite number of array elements of a set length, but not a set number of array elements of an indefinite length.

Flexible arrays occur in only a few contexts; for example, as parameters:

```
char *argv[];
char p[][8];
```

as **extern** declarations:

```
extern int end[];
```

or as a member of a structure — usually, though not necessarily, the last:

```
struct nlist {
        struct nlist *next;
        char name[];
};
```

Example

The following program initializes an automatic array, and prints its contents.

```
main()
{
        int foo[3] = { 1, 2, 3 };

        printf("Here's foo's contents: %d %d %d\n",
                foo[0], foo[1], foo[2]);
}
```

See Also

definitions, struct

The C Programming Language, ed. 2, pages 25, 83, 210

ASKCC — Environmental Variable

Force prompting for CC names

ASKCC=*YES/NO*

The environmental variable **ASKCC** directs the program **mail** to prompt for carbon-co names. A carbon-copy (or CC) name gives another person to whom a mail messa should be sent. To turn on prompting, use the command:

```
export ASKCC=YES
```

See Also

environmental variables, mail

assert.h — Header File

Define **assert()**

#include <assert.h>

assert.h is the header file that defines the macro **assert**.

See Also

assert(), header files

at — Device Driver

Drivers for hard-disk partitions

/dev/at* are the COHERENT system's AT devices for the hard-disk's partitions. Ea device is assigned major-device number 11, and may be accessed as a block- or characte special device.

The **at** hard-disk driver handles two drives with up to four partitions each. Min devices 0 through 3 identify the partitions on drive 0. Minor devices 4 through 7 identi the partitions on drive 1. Minor device 128 allows access to all of drive 0. Minor devi

129 allows access to all of drive 1. To modify the offsets and sizes of the partitions, t
the command **fdisk** on the special device for each drive (minor devices 128 and 129).

To access a disk partition through COHERENT, directory **/dev** must contain a de
file that has the appropriate type, major and minor device numbers, and permissions.
create a special file for this device, invoke the command **mknod** as follows:

```
/etc/mknod /dev/at0a b 11   0 ; : drive 0, partition 0
/etc/mknod /dev/at0b b 11   1 ; : drive 0, partition 1
/etc/mknod /dev/at0c b 11   2 ; : drive 0, partition 2
/etc/mknod /dev/at0d b 11   3 ; : drive 0, partition 3
/etc/mknod /dev/at0x b 11 128 ; : drive 0, partition table
```

Drive Characteristics

To read the characteristics of a hard disk, use the call to **ioctl** of the following form:

```
#include <sys/hdioctl.h>
hdparm_t hdparms;
    . . .
ioctl(fd, HDGETA, (char *)&hdparms);
```

where *fd* is a file descriptor for the hard disk device and *hdparms* receives the d
characteristics.

Non-Standard and Unsupported Types of Drives

Prior releases of the the COHERENT **at** hard-disk driver would not support disk dri
whose geometry was not supported by the BIOS disk parameter tables. COHERE
release 3.1 adds support for these drives during installation by "patching" the d
parameters into the bootstrap and the **/coherent** image on the hard disk. Please n
that if you intend to move your COHERENT root file system from an unsupported dr
to another disk drive, you will need to patch these parameters to match the dr
geometry of the new disk drive.

Files

/dev/at* — Block-special files
/dev/rat* — Character-special files

See Also

device drivers, fdisk

boot.fha — Device Driver

Boot block for floppy disk

To be bootable, a COHERENT file system must contain a boot block (either **boot**
boot.fha). In addition, all hard disks must contain the master boot block **mboot** or
equivalent.

boot.fha is a boot block for a hard disk partition or a 15-sector floppy. It must be
stalled as the first sector of the partition or diskette, as follows:

```
/etc/fdformat -a /dev/fha0
/etc/badscan -v -o proto1 /dev/fha0 2400
/etc/mkfs /dev/fha0 proto1
rm proto1
cp /conf/boot.fha /dev/fha0
```

boot.fha searches its root directory '/' for file **autoboot**. If it finds this ker **boot.fha** loads and runs it. Otherwise, it gives the prompt ?, to which the user m type the name of the operating-system kernel to load (typically, **coherent**). If **boot.** cannot find the requested kernel or if an error occurs, **boot.fha** repeats the prompt the user must type another name.

Files

/conf/boot.fha — Partition or 15-sector 96tpi floppy boot block

See Also

badscan, boot, device drivers, fdisk, mboot, mkfs

brc — System Maintenance

Perform maintenance chores, single-user mode
/etc/brc

The shell script **/etc/brc** is executed by the **init** process when the COHERENT syst enters single-user mode. The commands in **brc** do such things as set system clock, the local time zone, and call **fsck** to scan and (if necessary) fix all file systems on y machine.

See Also

init, rc, system maintenance

calendar — Command

Reminder service
calendar [-a]

calendar is the COHERENT system's "reminder service". It reads a user's **$HO** directory and looks for a file called **.calendar**. This file contains information organi by date. If **calendar** finds **.calendar**, it reads it and checks the date of each entry; if event is scheduled to happen today or tomorrow, it prints it. Thus, you can use **cal dar** to remind you of both one-time events (such as appointments) and yearly eve (such as anniversaries).

The following gives an example of a **.calendar** file. Note that **calendar** understa different formats of dates:

```
Apr 16   Dave's birthday
7/6      Dad's birthday
Sep 26   Mom's birthday
Jun 30   Barry's birthday
10/4     Marianne's birthday
Jul 31   Anniversary!
Mar 16   Pot luck luncheon
```

Each user can run **calendar** by embedding the command

```
calendar
```

in his **.profile**.

If you wish, you can run **calendar** automatically for all users on your system, by inse ting it into file **/usr/lib/crontab**. In this case, **calendar** should be used with its **-a** o tion; this forces **calendar** to search every user's **$HOME** directory for a **.calendar** fil and mail the appointments it finds to that user.

Note that **calendar**'s definition of tomorrow understands weekends but not holiday thus, if invoked on a Friday, it will return the events for that day and the followin Saturday, Sunday, and Monday. If Monday is a holiday, however, you will not receiv appointments for Tuesday.

See Also

commands

cc — Command

Compiler controller
cc [*compiler options*] *file* .... [*linker options*]

cc is the program that controls compilation of C programs. It guides files of source an object code through each phase of compilation and linking. **cc** has many options to assis in the compilation of C programs; in essence, however, all you need to do to produce a executable file from your C program is type **cc** followed by the name of the file or file that hold your program. It checks whether the file names you give it are reasonable selects the right phase for each file, and performs other tasks that ease the compilation o your programs.

File Names

cc assumes that each *file* name that ends in **.c** or **.h** is a C program and passes it to th C compiler for compilation.

cc assumes that each *file* argument that ends in **.s** is in Mark Williams assembly lan guage and processes it with the assembler **as**.

cc also passes all files with the suffixes **.o** or **.a** unchanged to the linker **ld**.

How cc Works

cc normally works as follows: First, it compiles or assembles the source files, naming th resulting object files by replacing the **.c** or **.s** suffixes with the suffix **.o**. Then, it link the object files with the C runtime startup routine and the standard C library, and leave the result in file *file*. If only one object file is created during compilation, it is delete after linking; however, if more than one object file is created, or if an object file of th same name existed before you began to compile, then the object file or files are n deleted.

Options

The following lists all of **cc**'s command-line options. **cc** passes some options through t the linker **ld** unchanged, and correctly interprets to it the options **-o** and **-u**.

A number of the options are esoteric and normally are not used when compiling a C program. The following are the most commonly used options:

-c	Compile only; do not link
-f	Include floating-point **printf**
-l*name*	Pass library **lib***name***.a** to linker
-o *name*	Call output file *name*
-V	Print verbose listing of **cc**'s action

-A MicroEMACS option. If an error occurs during compilation, **cc** automatically invokes the MicroEMACS screen editor. The error or errors are displayed in one window and the source code file in the other, with the cursor set to the line number indicated by the first error message. Typing **<ctrl-X>>** moves to the next error, **<ctrl-X><** moves to the previous error. To recompile, close the edited file with **<ctrl-Z>**. Compilation will continue either until the program compiles without error, or until you exit from the editor by typing **<ctrl-U>** followed by **<ctrl-X><ctrl-C>**.

-B[*string*]
Backup option. Use alternate versions of the compiler for **cc0**, **cc1**, **cc2**, and **cc3**. If *string* is supplied, **cc** prepends it to the names of the phases of the compiler to form the pathnames where these are found. Otherwise, **cc** prepends the name of the current directory. If a **-t** option was previously given, only the parts of the compiler specified by it are affected. Any number of **-B** and **-t** options may be used, with each **-t** option specifying the passes affected by the subsequent **-B** option. For example, the command

```
cc -tp2 -Bnew hello.c
```

compiles **hello.c** using **newcc2** in place of the ordinarily used **/lib/cc2**, and using **newcpp** in place of the ordinarily used **/lib/cpp**.

-c Compile option. Suppress linking and the removal of the object files.

-D*name*[=*value*]
Define *name* to the preprocessor, as if set by a **#define** directive. If *value* is present, it is used to initialize the definition.

-E Expand option. Run the C preprocessor **cpp** and write its output onto the standard output.

-f Floating point option. Include library routines that perform floating-point arithmetic. Because the floating-point routines require approximately five kilobytes of memory, the standard C library does not include them; the **-f** option tells the compiler to include them. If a program is compiled without the **-f** option but attempts to print a floating point number during execution by using the **e**, **f**, or **g** format specifications to **printf**, the message

```
You must compile with -f option for floating point
```

will be printed and the program will exit.

-I *name*
Include option. Specify a directory the preprocessor should search for files given

in **#include** directives, using the following criteria: If the **#include** statement reads

```
#include "file.h"
```

cc searches for **file.h** first in the source directory, then in the directory named in the **-I***name* option, and finally in the system's default directories. If the **#include** statement reads

```
#include <file.h>
```

cc searches for **file.h** first in the directories named in the **-I***name* option, and then in the system's default directories. Multiple **-I***name* options are executed in the order of their of appearance.

-K Keep option. Do not erase the intermediate files generated during compilation. Temporary files will be written into the current directory.

-l *name*
library option. Pass the name of a library to the linker. **cc** expands **-l***name* into **/lib/lib***name***.a**. If an alternative library prefix has been specified by the **-tl** and **-B***string* options, then **-l***name* expands to *string***lib***name***.a**. **Note that this is a *linker option*, and so must appear at the end of the cc command line, or it will not be processed correctly.**

-M *string*
Machine option. Use an alternate version of **cc0**, **cc1**, **cc1a**, **cc1b**, **cc2**, **cc3**, **as**, **lib*.a**, and **crts0.o**, named by fixing *string* between the directory name and the pass and file names.

-n Instruct the linker **ld** to bind the output with separate shared and private segments, and which each starting on a separate hardware-segment boundary. This allows several processes to simultaneously use one copy of the shared segment. Note that programs linked with this option will run a little more slowly than if they were not so linked; however, if a program forks (e.g., **kermit**) or will be used by more than one user at a time (e.g., MicroEMACS), this slightly slower time will be more than offset by the program's being spared having to read an entire copy of itself from the disk.

-N[p0123sdlrt]*string*
Name option. Rename a specified pass to *string*. The letters **p0123sdlrt** refer, respectively, to **cpp**, **cc0**, **cc1**, **cc2**, **cc3**, the assembler, the linker, the libraries, the run-time start-up, and the temporary files.

-o *name*
Output option. Rename the executable file from the default to *name*. If this option is not used, the executable will be named after the first **.c** or **.o** file on the command line.

-O Optimize option. Run the code generated by the C compiler through the peephole optimizer. The optimizer pass is mandatory for the i8086, Z8000, and M6800 compilers, and need not be requested. It is optional for the PDP11 compiler, but is recommended for all files except those that consist entirely of initialized tables of data.

-q[p0123s]
Quit option. Terminate compilation after running the specified pass. The letters **p0123s** refer, respectively, to **cpp**, **cc0**, **cc1**, **cc2**, **cc3**, and the assembler. For example, to terminate compilation after running the parser **cc0**, type **-q0**.

-Q Quiet option. Suppress all messages.

-S Suppress the object-writing and link phases, and invoke the disassembler **cc3**. This option produces an assembly-language version of a C program for examination, for example if a compiler problem is suspected. The assembly-language output file name replaces the **.c** suffix with **.s**. This is equivalent to the **-VASM** option.

-t[p01ab23sdlrt]
Take option. Use alternate versions of the compiler phases and other files specified in the following string. If no following string is given, the **cc** uses alternate version of every phase of the compiler, except the preprocessor. If the **-t** option is followed by a **-B** option, **cc** prepends the prefix string named in the **-B** option to the phases and files named in the **-t** option; otherwise, the it looks for the alternate forms in the current directory.

-U *name*
Undefine symbol *name*. Use this option to undefine symbols that the preprocessor defines implicitly, such as the name of the native system or machine.

-V Verbose option. **cc** prints onto the standard output a step-by-step description of each action it takes.

V*string*
Variant option. Toggle (i.e., turn on or off) the variant *string* during the compilation. Variants that are marked **on** are turned on by default. Options marked **Strict:** generate messages that warn of the conditions in question. **cc** recognizes the following variants:

-VASM
Output assembly-language code. Identical to **-S** option, above. It can be used with the **-VLINES** option, described below, to generate a line-numbered file of assembly language. Default is **off**.

-VCOMM
Permit **.com**-style data items. Default is **on**.

-VFLOAT
Include floating point **printf** routines. Same as **-f** option, above.

-VLINES
Generate line number information. Can be used with the option **-VASM**, described above, to generate assembly language output that uses line numbers. Default is **off**.

-VQUIET
Suppress all messages. Identical to **-Q** option. Default is **off**.

-VSBOOK
Strict: note deviations from *The C Programming Language*, ed. 1. Defa
off.

-VSCCON
Strict: note constant conditional. Default is **off**.

-VSINU
Implement struct-in-union rules instead of Berkeley-member resolution r
Default is **off**, i.e., Berkeley rules are the default.

-VSLCON
Strict: **int** constant promoted to **long** because value is too big. Default is

-VSMEMB
Strict: check use of structure/union members for adherence to standard
of C. Default is **on**.

-VSNREG
Strict: register declaration reduced to auto. Default is **on**.

-VSPVAL
Strict: pointer value truncated. Default is **off**.

-VSRTVC
Strict: risky types in truth contexts. Default is **off**.

-VSTAT
Give statistics on optimization.

-VS
Turn on all strict checking. Default is **on**.

-VSUREG
Strict: note unused registers. Default is **off**.

-VSUVAR
Strict: note unused variables. Default is **on**.

-V3GRAPH
Translate ANSI trigraphs. Default is **off**.

See Also
as, C language, cc0, cc1, cc2, cc3, commands, cpp, ld
The C Language, tutorial

chase — Command

Highly amusing video game
/usr/games/chase [**-c**] [*speed*]

chase is a COHERENT version of a popular video game. It runs on the console
IBM AT COHERENT system with input from the console keyboard. **chase** ass
that the system console is a monochrome display adapter unless you select the **-c**
display option.

To accomodate different computer system speeds and different levels of skill, **chas** prompts the user to type a speed when the game begins. Press **<return>** to try out th game with the default speed of ten; typing a higher number makes the game slower, lower number makes it faster. If you can play at speed zero on a fast computer system you play too many video games. If you know the speed you want, you can enter it as command-line argument. If you see the boss coming, you can quit by pressing **<ctrl C>**.

The Rules

The player (represented by a blinking shaded rectangle) attempts to evade four "ghosts' (represented by shaded rectangles with arrows) while erasing dots from the playing board maze.

At the beginning of a game, the four ghosts are in the *ghost box* above the center of the maze and the player is below it. The maze is filled with dots, including four blinking diamonds called *power pellets*. The ghosts emerge from the ghost box and chase the player. The console arrow keys move the player left, right, up, or down through the maze. Typing '0' stops the player. The player continues to move in the same direction until a wall of the maze stops him, you type a '0', or you type another arrow key.

When the player eats a power pellet, he acquires super power and can chase the ghosts briefly; the ghosts change color while the player has super power. If the player catches a ghost, he scores a bonus and the ghost returns to the ghost box temporarily. Once a player eats all the dots on the board, the game continues at the next level.

The upper left corner of the screen displays a score and the current board level. Each dot the player eats scores ten points. The first ghost a player eats while he has super power scores 200 points, the second 400, the third 800, and the fourth 1,600. At certain times during the game, a bonus letter appears below the ghost box; the player scores 100 points for eating the bonus letter on level 'A', 300 on level 'B', 500 on level 'C', and so on.

The lower left corner of the screen displays the number of extra players remaining in the current game (initally two). Another bonus player appears every 10,000 points, to a maximum of three extra players. The game ends when the ghosts eat the last player.

See Also

commands

check — Command

Check file system
check [-s] *filesystem ...*

check uses the commands **icheck** and **dcheck** to check the consistency of a file system. It acts on each argument *filesystem* in turn; it calls first **icheck** and then **dcheck** on each to detect problems.

If **-s** is specified, **check** attempts to repair any errors automatically. You should first unmount the file system, if possible. If the root device is involved, you should be in single-user mode and then reboot the system immediately (without typing **sync**).

See Also

clri, commands, icheck, ncheck, sync, umount

Notes

Certain errors, such as duplicated blocks, cannot be fixed automatically. Decisio be made by a human.

In earlier releases of COHERENT, **check** acted upon a default file system if no specified.

This command has largely been superceded by **fsck**.

com — Device Driver

Device drivers for asynchronous serial lines

The COHERENT system has drivers for four asynchronous serial lines, **com1** t **com4**.

A serial line can be opened into any of four different "flavors", as follows:

com?l	Interrupt driven, local mode (no modem control)
com?r	Interrupt driven, remote mode (modem control)
com?pl	Polled, local mode (no modem control)
com?pr	Polled, remote mode (modem control)

"Local mode" means that the line will have a terminal plugged into it, to directly the computer. "Modem control" means that the line will have a modem plugged Modem control is enabled on a serial line by resetting the modem control bit (bit the minor number for the device. This allows the system to generate a hangup when then modem indicates loss of carrier by dropping DCD (Data Carrier Detec modem line should always have its DSR, DCD and CTS pins connected. If left ha spurious transitions can cause severe system thrashing. To disable modem contro given serial line, use the minor device which has the modem control bit set (bit 7) **open** to a modem-control line will block until a carrier is detected (DCD goes true).

"Interrupt mode" means that the port can generate an interrupt to attract the att of the COHERENT system; "polled mode" means that the port cannot generate an rupt, but must be checked (or "polled") constantly by the COHERENT system to activity has occurred on it.

The COHERENT system uses two device drivers to manage serial lines: one manages COM1 and COM3, and the other manages COM2 and COM4. D limitations in the design of the ports, you can enable interrupts on either CO COM3 (or on COM2 or COM4), but not both. If you wish to use both ports taneously, one must be run in polled mode. For example, if you wish to open all serial lines, you can open two of the lines in interrupt mode: you can open either C or COM3 in interrupt mode, and you can open either COM2 or COM4 in inte mode. The other two lines must be opened in polled mode.

Opening a device in polled mode consumes many CPU cycles, based upon the spe the highest baud rate requested. For example, on a 20 MHz 80386-based mac polling at 9600-baud was found to consume about 15% of the CPU time. As only device can use the interrupt line at any given time, the best approach is to mak high-speed line of the pair interrupt driven and open the low-speed or less-frequ

used line in polled mode. However, if you enable a polled line for logins, the port is open and will be polled as long as the port remains open (enabled). Thus, even if a port is not in use, the fact that it has a **getty** on it consumes CPU cycles. As a rule of thumb, try and open a port in interrupt mode. If you cannot, use the polled version. Also note that use of any of the four serial ports in polled mode prevents other polled serial device drivers, such as the **hs** generic multi-port polled serial driver, from being used at the same time.

If you intend to use a modem on your serial port, you must insure that the DCD signal from the modem actually *follows* the state of carrier detect. Some modems allow the user to "strap" or set the DCD signal so that it is always asserted (true). This incorrect setup will cause COHERENT to think that the modem is "connected" to a remote modem, even when there is no such connection.

In addition, if you wish to allow remote logins to your COHERENT system via your modem, you must insure that the modem does **not** echo any commands or status information. Failure to do so will result in severe system thrashing due to the **getty** or **login** processes endlessly "talking" to your modem.

Changing Default Port Speeds

Serial lines **com1** through **com4** default to 9600 baud when opened. This default speed can be permanently changed on a "per port" basis by changing the value of driver variables **C1BAUD_**, **C2BAUD_**, **C3BAUD_** or **C4BAUD_**. The list of acceptible values can be found in header file **<sgtty.h>** and range from 1, corresponding to 50 baud, up to 17, which corresponds to 19,200 baud.

To change the default value for a port, you must use the **/conf/patch** command. For example, to change the default speed for port **com2** to 2400 baud, enter the following command while running as the superuser:

```
/conf/patch /coherent C2BAUD_=12
```

The change will not take effect until the next time that you boot your system.

See Also

com1, com2, com3, com4, device drivers

Diagnostics

An attempt to open a non-existent device will generate error messages. This can occur if hardware is absent or not turned on.

com1 — Device Driver

Device driver for asynchronous serial line COM1

/dev/com1 is the COHERENT system's standard interface to asynchronous serial line COM1. The interface is assigned major device 5, and is accessed as a character-special device. The I/O address for the corresponding 8250 SIO is 0x3F8 (COM1). **com1** generates interrupt IRQ4.

Four versions of device **com1** are in directory **/dev**, as follows:

Device Name	*Major*	*Minor*	*I/O Type*	*Modem Control?*
/dev/com1l	5	128	Interrupts	No
/dev/com1r	5	0	Interrupts	Yes
/dev/com1pl	5	192	Polled	No
/dev/com1pr	5	64	Polled	Yes

For details on how these versions differ, see the entry for **com**.

Files

/dev/com1l — Interrupt-driven, non-modem (local) line
/dev/com1r — Interrupt-driven, modem (non-local) line
/dev/com1pl — Polled, non-modem (local) line
/dev/com1pr — Polled, modem (non-local) line

See Also

com, com3, stty

com2 — Device Driver

Device driver for asynchronous serial line COM2

/dev/com2 is the COHERENT system's standard interface to asynchronous serial l COM2. The interface is assigned major device 6, and is accessed as a character-sp device. The I/O address for the corresponding 8250 SIO is 0x2F8 (COM2). c generates interrupt IRQ3.

Four versions of device **com2** are in directory **/dev**, as follows:

Device Name	*Major*	*Minor*	*I/O Type*	*Modem Control?*
/dev/com2l	6	128	Interrupts	No
/dev/com2r	6	0	Interrupts	Yes
/dev/com2pl	6	192	Polled	No
/dev/com2pr	6	64	Polled	Yes

For details on how these differ, see the entry for **com**.

Files

/dev/com2l — Interrupt-driven, non-modem (local) line
/dev/com2r — Interrupt-driven, modem (non-local) line
/dev/com2pl — Polled, non-modem (local) line
/dev/com2pr — Polled, modem (non-local) line

See Also

com, com4, stty

com3 — Device Driver

Device driver for asynchronous serial line COM3

/dev/com3 is the COHERENT system's standard interface to asynchronous serial l COM3. The interface is assigned major device 5, and is accessed as a character-sp device. The I/O address for the corresponding 8250 SIO is 0x3E8 (COM3). c generates interrupt IRQ4.

Four versions of device **com3** are in directory **/dev**, as follows:

Device Name	*Major*	*Minor*	*I/O Type*	*Modem Control?*
/dev/com3l	5	129	Interrupts	No
/dev/com3r	5	1	Interrupts	Yes
/dev/com3pl	5	193	Polled	No
/dev/com3pr	5	65	Polled	Yes

For details on how these differ, see the entry for **com**.

Files

/dev/com3l — Interrupt-driven, non-modem (local) line
/dev/com3r — Interrupt-driven, modem (non-local) line
/dev/com3pl — Polled, non-modem (local) line
/dev/com3pr — Polled, modem (non-local) line

See Also

com, com1, stty

com4 — Device Driver

Device driver for asynchronous serial line COM4

/dev/com4 is the COHERENT system's standard interface to asynchronous serial lin COM4. The interface is assigned major device 6, and is accessed as a character-speci device. The I/O address for the corresponding 8250 SIO is 0x2E8 (COM4). **com** generates interrupt IRQ3.

Four versions of device **com4** are in directory **/dev**, as follows:

Device Name	*Major*	*Minor*	*I/O Type*	*Modem Control?*
/dev/com4l	6	129	Interrupts	No
/dev/com4r	6	1	Interrupts	Yes
/dev/com4pl	6	193	Polled	No
/dev/com4pr	6	65	Polled	Yes

For details on how these differ, see the entry for **com**.

Files

/dev/com4l — Interrupt-driven, non-modem (local) line
/dev/com4r — Interrupt-driven, modem (non-local) line
/dev/com4pl — Polled, non-modem (local) line
/dev/com4pr — Polled, modem (non-local) line

See Also

com, com2, stty

compress – Command

Compress a file

compress [**-dfvc**] [**-b***num*] [**-w** *tempfile*] [*file* ...]

compress compresses a file using the Lempel-Ziv algorithm. With text files and chives, it often can achieve 50% rate of compression.

If one or more *files* are specified on the command, **compress** compresses them and pends the suffix **.Z** onto the end of each compressed file's name. If no *file* is specified the command line, **compress** compresses text from the standard input and writes compressed text to the standard output.

compress recognizes the following options:

-d Decompress rather than compress.

-f Force an output file to be generated even if no space is saved by compression.

-v Verbose mode: force **compress** to write statistics about its performance.

-c Send output to stdout.

-b The "bits" option. **compress** uses the compression level set via the *num* ment. Previous releases of **compress** would only allow values of *num* up to with 12 being the default value if the **-b** option was not specified. The versio **compress** introduced with COHERENT version 3.1 handles values up to with 16 being the default.

-w The "workfile" option. **compress** uses *workfile* to write its temporary file. default **compress** uses RAM device **/dev/ram1** for temporary storage. For reason, it is strongly advised that you not use **/dev/ram1** as a RAM disk. option may be necessary on machines with limited amounts of RAM.

If you wish to ensure backwards compatibility with previous releases of COHERENT, not use **compress** with a *num* value greater than 12.

See Also

commands, ram, uncompress, zcat

cpio – Command

Archive utility

cpio is an archiving utility that reads and writes files in the format specified by the **Archive/Interchange File Format** specified in IEEE standard 1003.1-1988.

See the compressed **tar** archive **/usr/src/alien/pax.tar.Z** for full documentation **cpio**.

Copyright Information

cpio was developed by Mark H. Colburn and sponsored by The USENIX Associat **cpio** is provided in binary form per the licensing terms set forth by the author. See **/usr/src/alien/pax.tar.Z** for licensing terms.

See Also

commands, pax, ustar

ctags — Command

Generate tags and refs files for elvis editor
ctags [**-r**] *files...*

ctags generates the files **tags** and **refs** from a group of C-source files. **tags** is used by the **elvis** editor's **:tag** command, **<ctrl-]>** command, and **-t** option. **refs** is used by the command **ref**.

Each C-source file is scanned for **#define** statements and global function definitions. The name of the macro or function becomes the name of a tag. For each tag, a line is added to **tags**, which contains the following:

- the name of the tag
- a tab character
- the name of the file containing the tag
- a tab character
- a way to find the particular line within the file

refs is used by the command **ref**, which can be invoked via **elvis**'s **K** command. When **ctags** finds a global function definition, it copies the function header into **refs**. The first line is flush against the right margin, but the argument definitions are indented. The command **ref** can search **refs** much faster than it could search all C-source files. The file-names list will typically include the names of all C-source files in the current directory, in the following format:

```
$ ctags -r *.[ch]
```

The **-r** to **ctags** tells it to generate both **tags** and **refs**. Without **-r**, it generates only **tags**.

See Also

commands, elvis, ref

Notes

This version of **ctags** does not parse ANSI source code very well. It has trouble recognizing the ANSI function definitions.

This program was written by Steve Kirkendall (kirkenda@cs.pdx.edu or ...uunet!tektronix!psueea!eecs!kirkenda). Source code for this program is available via the Mark Williams bulletin board, USENET and other sources. It is offered as a service to COHERENT users, but is not supported by Mark Williams Company. *Caveat utilitor.*

curses — Overview

Library of screen-handling functions

curses is a set of routines that allow you to manipulate the screen in a sophisticated manner. These routines use the **termcap** functions to read information about the user's terminal. This allows you to write programs that can perform rudimentary graphics on a wide variety of terminals.

curses contains routines that do the following:

- Move the cursor about the screen.
- Insert text onto the screen, either in normal or reverse video (if supported by th display device).
- Read what is typed by the user and display it properly.
- Organize the screen into one or more rectangular regions, or *windows*, optionall draw a border around each, and manage each independently.

curses organizes the screen into a two-dimensional array of cells, one cell for ever character that the device can display. It maintains in memory an image of the scree called the *curscr*. A second image, called the *stdcur*, is manipulated by the user; whe the user has finished a given manipulation, **curses** copies the changes from the *stdcur* t the *curscr*, which results in their being displayed on the physical screen. This act copying from the *stdscr* to the *curscr* is called *refreshing* the screen. **curses** keeps trac of where all changes have begun and ended between one refresh and the next; this lets i rewrite only the portions of the *curscr* that the user has changed, and so speed u rewriting of the screen.

curses records the position of a "logical cursor", which points to the position in th *stdscr* that is being manipulated by the user, and also records the position of the physic cursor. Note that the two are not necessarily identical: it is possible to manipulate th logical cursor without repositioning the physical cursor, and vice versa, depending on th task you wish to perform.

Most **curses** routines work by manipulating **WINDOW** object. **WINDOW** is defined i the header **curses.h** as follows:

```
#define WINDOW _win_st
struct _win_st {
      short                 _cury, _curx;
      short                 _maxy, _maxx;
      short                 _begy, _begx;
      short                 _flags;
      short                 _ch_off;
      bool                  _clear;
      bool                  _leave;
      bool                  _scroll;
      char                  **_y;
      short                 *_firstch;
      short                 *_lastch;
      struct _win_st        *_nextp, *_orig;
};
```

Type **bool** is defined in **curses.h**; an object of this type can hold the value of true (no zero) or false (zero).

The following describes each **WINDOW** field in detail.

_cury, _curx

Give the Y and X positions of the logical cursor. The upper left corner o the window is, by definition, position 0,0. Note that **curses** by conven tion gives positions as Y/X (column/row) rather than X/Y, as is usu

elsewhere.

_maxy, _maxx
Width and height of the window.

_begy, _begx
Position of the upper left corner of the window relative to the upper le corner of the physical screen. For example, if the window's upper le corner is five rows from the top of the screen and ten columns from th left, then **_begy** and **_begx** will be set to ten and five, respectively.

_flags One or more of the following flags, logically OR'd together:

_SUBWIN — Window is a sub-window
_ENDLINE — Right edge of window touches edge of the screen
_FULLWIN — Window fills the physical screen
_SCROLLWIN — Window touches lower right corner of physical screen
_FULLINE — Window extends across entire physical screen
_STANDOUT — Write text in reverse video
_INSL — Line has been inserted into window
_DELL — Line has been deleted from window

_ch_off Character offset.

_clear Clear the physical screen before next refresh of the screen.

_leave Do not move the physical cursor after refreshing the screen.

_scroll Enable scrolling for this window.

_y Pointer to an array of pointers to the character arrays that hold the window's text.

_firstch Pointer to an array of integers, one for each line in the window, whose value is the first character in the line to have been altered by the user. If a line has not been changed, then its corresponding entry in the array is set to **_NOCHANGE**.

_lastch Same as **_firstch**, except that it indicates the last character to have been changed on the line.

_nextp Point to next window.

_orig Point to parent window.

When **curses** is first invoked, it defines the entire screen as being one large window. The programmer has the choice of subdividing an existing window or creating new windows; when a window is subdivided, it shares the same *curscr* as its parent window, whereas a new window has its own *stdscr*.

Mark Williams Company will document its **curses** library in full in a later release of this manual. The following table, however, summarizes the functions and macros that that compose the **curses** library.

addch(*ch*) **char** *ch*;
Insert a character into *stdscr*.

addstr(*str*) **char** **str*;
Insert a string into *stdscr*.

box(*win, vert, hor*) **WINDOW** **win*; **char** *vert, hor*;
Draw a box. *vert* is the character used to draw the vertical lines, and *hor* is used to draw the horizontal lines. For example

```
box(win, '|', '-');
```

draws a box around window **win**, using '|' to draw the vertical lines and '-' to draw the horizontal lines.

clear()
Clear the *stdscr*.

clearok(*win,bf*) **WINDOW** **win*; **bool** *bf*;
Set the clear flag for window *win*. This will clear the screen at the next refresh but not reset the window.

clrtobot()
Clear from the position of the logical cursor to the bottom of the window.

clrtoeol()
Clear from the logical cursor to the end of the line.

crmode()
Turn on control-character mode; i.e., force terminal to receive cooked input.

delch()
Delete a character from *stdscr*; shift the rest of the characters on the line one position to the left.

deleteln()
Delete all of the current line; shift up the rest of the lines in the window.

delwin(*win*) **WINDOW** **win*;
Delete window *win*.

echo()
Turn on both physical and logical echoing; i.e., character are automatically inserted into the current window and onto the physical screen.

endwin()
Terminate text processing with **curses**.

erase()
Erase a window; do not clear the screen.

getch()
Read a character from the terminal.

getstr(*str*) **char** **str*;
Read a string from the terminal.

getyx(*win,y,x*) **WINDOW** **win*; **short** *y,x*;
Read the position of the logical cursor in *win* and store it in *y,x*. Note that this a macro, and due to its construction the variables *y* and *x* must be integers, n pointers to integers.

inch() Read the character pointed to by the *stdscr*'s logical cursor.

WINDOW ***initscr**()
Initialize **curses**.

insch(*ch*) **char** *ch*;
Insert character *ch* into the *stdscr*.

insertln()
Insert a blank line into *stdscr*, above the current line.

leaveok(*win,bf*) **WINDOW** **win*; **bool** *bf*;
Set _**leave** in *win* to *bf*.

char ***longname**(*termbuf, name*) **char** **termbuf, *name*;
Copy the long name for the terminal from *termbuf* into *name*.

move(*y,x*) **short** *y,x*;
Move logical cursor to position *y,x* in *stdscr*.

mvaddbytes(*y,x,da,count*) **int** *y,x*; **char** **da*; **int** *count*;
Move to position *y,x* and print *count* bytes from the string pointed to by *da*.

mvaddch(*y,x,ch*) **short** *y,x*; **char** *ch*;
Move the logical cursor to position *y,x* and insert character *ch*.

mvaddstr(*y,x,str*) **short** *y,x*; **char** **str*;
Move the logical cursor to position *y,x* and insert string *str*.

mvcur(*y_cur,x_cur,y_new,x_new*) **int** *y_cur, x_cur, y_new, x_new*;
Move cursor from position *y_cur,x_cur* to position *y_new,x_new*.

mvdelch(*y,x*) **short** *y,x*;
Move to position *y,x* and delete the character found there.

mvgetch(*y,x*) **short** *y,x*;
Move to position *y,x* and get a character through *stdscr*.

mvgetstr(*y,x,str*) **short** *y,x*; **char** **str*;
Move to position *y,x*, get a string through *stdscr*, and copy it into *string*.

mvinch(*y,x*) **short** *y,x*;
Move to position *y,x* and get the character found there.

mvinsch(*y,x,ch*) **short** *y,x*; **char** *ch*;
Move to position *y,x* and insert a character into *stdscr*.

mvwaddbytes(*win,y,x,da,count*) **WINDOW** **win*; **int** *y,x*; **char** **da*; **int** *count*;
Move to position *y,x* and print *count* bytes from the string pointed to by *da* int window *win*.

mvwaddch(*win,y,x,ch*) **WINDOW** **win*; **int** *y,x*; **char** *ch*;
Move to position *y,x* and insert character *ch* into window *win*.

mvwaddstr(*win,y,x,str*) **WINDOW** **win*; **short** *y,x*; **char** **str*;
Move to position *y,x* and insert character *ch*.

mvwdelch(*win,y,x*) **WINDOW** **win*; **int** *y,x*;
Move to position *y,x* and delete character *ch* from window *win*.

mvwgetch(*win,y,x*) **WINDOW** **win*; **short** *y,x*;
Move to position *y,x* and get a character.

mvwgetstr(*win,y,x,str*) **WINDOW** **win*; **short** *y,x*; **char** **str*;
Move to position *y,x*, get a string, and write it into *str*.

mvwin(*win,y,x*) **WINDOW** **win*; **int** *y,x*;
Move window *win* to position *y,x*.

mvwinch(*win,y,x*) **WINDOW** **win*; **short** *y,x*;
Move to position *y,x* and get character found there.

mvwinsch(*win,y,x,ch*) **WINDOW** **win*; **short** *y,x*; **char** *ch*;
Move to position *y,x* and insert character *ch* there.

WINDOW ***newwin**(**lines, cols, y1, x1**) **int** *lines, cols, y1, x1*;
Create a new window. The new window is *lines* lines high, *cols* columns wid
with the upper-left corner at position *y1,x1*.

nl() Turn on newline mode; i.e., force terminal to output **<newline>** aft
<linefeed>.

nocrmode()
Turn off control-character mode; i.e., force terminal to accept raw input.

noecho()
Turn off echo mode.

nonl()
Turn off newline mode.

noraw()
Turn off raw mode.

overlay(*win1,win2*) **WINDOW** **win1*, *win2*;
Copy all characters, except spaces, from their current positions in *win1* to ide
cal positions in *win2*.

overwrite(*win1,win2*) **WINDOW** **win1*, *win2*;
Copy all characters, including spaces, from *win1* to their identical positions
win2.

printw(*format[,arg1,...argN]*) **char** **format*; [*data type*] *arg1,..argN*;
Print formatted text on the standard screen.

raw() Turn on raw mode; i.e., kernel does not process what is typed at the keyboard,
passes it directly to **curses**. In normal (or *cooked*) mode, the kernel interce
and processes the control characters <*ctrl-C*>, <*ctrl-S*>, <*ctrl-Q*>, and <*ct*

Y>. See the entry for **stty** for more information.

refresh()
Copy the contents of *stdscr* to the physical screen.

resetty()
Reset the terminal flags to values stored by earlier call to **savetty**.

savetty()
Save the current terminal settings.

scanw(*format[,arg1,...argN]*) **char** **format*; [*data type*] *arg1,..argN*;
Read the standard input; translate what is read into the appropriate data type

scroll(*win*) **WINDOW** **win*;
Scroll *win* up by one line.

scrollok(*win,bf*) **WINDOW** **win*; **bool** *bf*;
Permit or forbid scrolling of window *win*, depending upon whether *bf* is set true or false.

standend()
Turn off standout mode.

standout()
Turn on standout mode for text. Usually, this means that text will be displa in reverse video.

WINDOW ***subwin**(*win,lines,cols,y1,x1*) **int** *win,lines,cols,y1,x1*;
Create a sub-window in window *win*. New sub-window is *lines* lines high, c columns wide, and is fixed at position *y1,x1*. Note that the position is relative the upper-left corner of the physical screen.

touchwin(*win*) **WINDOW** **win*;
Copy all characters in window *win* to the screen.

waddch(*win,ch*) **WINDOW** **win*; **char** *ch*;
Add character *ch* *win*.

waddstr(*win,str*) **WINDOW** **win*; **char** **str*;
Add the string pointed to by *str* to window *win*.

wclear(*win*) **WINDOW** **win*;
Clear window *win*. Move cursor to position 0,0 and set the screen's clear flag.

wclrtobot(*win*) **WINDOW** **win*;
Clear window *win* from current position to the bottom.

wclrtoeol(*win*) **WINDOW** **win*;
Clear window *win* from the current position to the end of the line.

wdelch(*win*) **WINDOW** **win*;
Delete the character at the current position in window *win*; shift all remainir characters to the right of the current position one position left.

wdeleteln(*win*) **WINDOW** **win*;
Delete the current line and shift all lines below it one line up.

werase(*win*) **WINDOW** **win*;
Clear window *win*. Move the cursor to position 0,0 but do not set the screen's clear flag.

wgetch(*win*) **WINDOW** **win*;
Read one character from the standard input.

wgetstr(*win,str*) **WINDOW** **win*; **char** **str*;
Read a string from the standard input; write it in the area pointed to by *str*.

winch(*win*) **WINDOW** **win*;
Force the next call to **refresh**() to rewrite the entire screen.

winsch(*win,ch*) **WINDOW** **win*; **char** *ch*;
Insert character *ch* into window *win* at the current position. Shift all existing characters one position to the right.

winsertln(*win*) **WINDOW** **win*;
Insert a blank line into window *win* at the current position. Move all lines down by one position.

wmove(*win,y,x*) **WINDOW** **win*; **int** *y, x*;
Move current position in the window *win* to position *y,x*.

wprintw(*win,format[,arg1,...argN]*) **WINDOW** **win*; **char** **format*; [*data type*] **arg1,..argN**;
Format text and print it to the current position in window *win*.

wrefresh(*win*) **WINDOW** **win*;
Refresh a window.

wscanw(*win,format[,arg1,...argN]*) **WINDOW** **win*; **char** **format*; [*data type*] *arg1,..argN*;
Read standard input from the current position in window *win*, format it, and store it in the indicated places.

wstandend(*win*) **WINDOW** **win*;
Turn off standout (reverse video) mode for window *win*.

wstandout(*win*) **WINDOW** **win*;
Turn on standout (reverse video) mode for window *win*.

These routines are declared and defined in the header file **curses.h**.

Structure of a curses Program

To use the **curses** routines, a program must included the header file **curses.h**, which declares and defines the functions and macros that comprise the **curses** library.

Before a program can perform any graphics operations, it must call the function **initscr**() to initialize the **curses** environment. Then, the program must call **cmdwind** to open the *curscr*.

As noted above, **curses** manipulates text in a copy of the screen that it maintains in memory. After a program has manipulated text, it must call **refresh**() to copy these

terations from memory to the physical screen. (This is done because writing to screen is slow; this scheme permits mass alterations to be made to copy in memory, written to the screen in a batch.)

Finally, when the program has finished working with **curses**, it must call the func **endwin()**. This frees memory allocated by **curses**, and generally closes down the cu environment gracefully.

Example

The following program, called **curexample.c**, gives a simple example of programm with **curses**. To compile this program, use the command line:

```
cc curexample.c -lcurses -lterm
```

Note that order in which the libraries are called is significant.

When this program is run, it clears the screen, then waits for you to type a Y coordin a space, and then an X coordinate. Note that these do not echo on the screen. It m the cursor to the requested coordinates, and there display any non-numeric string you type. If you type numerals, **curexample** will assume that you wish to move cursor to a new location. To exit, type **<ctrl-C>**.

```
#include <ascii.h>
#include <ctype.h>
#include <curses.h>

#define NORMAL 0
#define INY  1
#define INX  2

main()
{
     int c, y, x, state;

     initscr();     /* initialize curses */
     noecho();
     raw();

     clear();
     move(0, 0);

     for(state = NORMAL;;) {
          refresh();
          c = getch();
          if(isdigit(c)) {
```

```
                switch (state) {
                case NORMAL:
                        y = x = 0;
                        state = INY;
                case INY:
                        y *= 10;
                        y += c - '0';
                        break;
                case INX:
                        x *= 10;
                        x += c - '0';
                }
        } else {
                if (c == A_ETX) { /* ctl-c */
                        noraw();
                        echo();
                        endwin();
                        exit(0);
                }
                switch (state) {
                case INX:
                        state = NORMAL;
                        move(y, x);
                case NORMAL:
                        addch(c);
                        break;
                case INY:
                        state = INX;
                }
        }
    }
}
```

See Also

curses.h, libraries, termcap

Strang J: *Programming with curses.* Sebastopol, Calif, O'Reilly & Associates Inc., 1986.

Notes

curses is copyrighted by the Regents of the University of California.

dcheck — Command Maintenance

Directory consistency check

dcheck [-s] [-i *inumber* ...] *filesystem* ...

dcheck performs a consistency check on each specified *filesystem*. It scans all the directories in each *filesystem* and keeps counts of all i-nodes referenced. It compares these counts against the link counts maintained in the i-nodes. dcheck notes any discrepancies, and notes allocated i-nodes with a 0 link count.

If the **-i** switch is present, **dcheck** compares each *inumber* in the list against those each directory. It reports matches by printing the i-number, the i-number of the par directory, and the name of the entry.

The **-s** switch causes **dcheck** to correct the link count of errant i-nodes to the en count.

Since **dcheck** is two-pass, the file system should be unmounted. If **-s** is used on the r file system, the system should be rebooted immediately (without performing a syn The raw device should be used.

See Also

check, dir.h, icheck, ncheck, sync, umount

Diagnostics

If the link count is 0 and there are entries, the file system must be mounted and entries removed immediately. If the link count is nonzero and the entry count is *larg* the **-s** option must be used to make the counts agree. In all other cases there may wasted disk space but there is no danger of losing file data.

Notes

In earlier releases of COHERENT, **dcheck** acted upon a default file system if none w specified.

This command has largely been replaced by **fsck**.

device drivers — Overview

A *device driver* is a program that controls the action of one of the physical devices tached to your computer system.

The following table lists the device drivers included with this edition of the COHEREN system. The first field gives the device's major device number; the second gives its nam and the third describes it. When a major device number has no driver associated with that device is available for a driver yet to be written.

0:	***mem**	Interface to memory
1:	**tty**	Primitive tty driver
2:	**kb/mm**	Keyboard and video
3:	**lp**	Parallel line printer
4:	**fl**	Floppy drive
5:	**al0**	Serial line 0 (COM1 and COM3)
6:	**al1**	Serial line 1 (COM2 and COM4)
7:	**hs**	Generic polled multi-port serial card
8:	**rm**	RAM disk
9:		
10:		
11:	**at**	AT hard disk
12:		
13:	**scsi**	SCSI device driver
14:		
15:		

16:
17:
18:
19:
20:
21:
22:
23: **sem** System V compatible semaphores
24: **shm** System V subset shared memory
25: **msg** System V compatible messaging
26:
27:
28:
29:
30:
31:

Also included are drivers for the following devices:

console	Console driver
ct	Controlling terminal driver
null	The "bit bucket"

See Also

at, boot, com, console, ct, fl, Lexicon, lp, mboot, mem, msg, null, scsi, sem shm, tape, termio

Notes

See the Release Notes for your release of COHERENT for a list of supported devices an device drivers.

The devices **msg**, **sem**, and **shm** are loadable drivers that can be loaded into memor using the command **drvld**. See their respective entries for more information.

drvld — Command

Load a loadable driver into memory
/etc/drvld **-r** *driver*

drvld loads a loadable driver into memory. *driver* names a loadable driver. Only the su peruser **root** can run **drvld**.

A loadable driver is one that is not linked into the kernel when it was built. The curren suite of loadable drivers include multi-port serial cards, various SCSI host adaptors, an a variety of add-on cards. The COHERENT drivers for shared memory, semaphore and message passing are also implemented as loadable drivers, due to the efficient size o the COHERENT kernel.

The **-r** option to **drvld** specifies that any temporary files it creates in directory **/tm** should be removed.

Note that **drvld** expects to find its entry in directory **/drv**, not in **/dev**.

Files

/drv — directory containing loadable drivers

See Also

commands, device drivers, sload()

Notes

COHERENT supports user-written, loadable device drivers generated with COHERENT device-driver kit.

elvis — Command

Clone of UNIX-standard screen editor
elvis [*flags*] [*+cmd*] [*file1 ... file27*]

elvis is a clone of **vi** and **ex**, the standard UNIX screen editors.

Unlike MicroEMACS, the COHERENT system's other screen editor, **elvis** is a m editor whose command structure resembles the **ed** line editor. *Modal* means th keystroke assumes a different meaning, depending upon the mode that the editor i **elvis** uses three modes: *visual command mode, colon command mode,* and *input m* The following sections summarize the commands associated with each mode.

Visual Command Mode

Visual-command mode closely resembles text-input mode. One quick way to tell modes apart is to press the **<esc>** key. If **elvis** beeps, then you are in visual-comm mode. If it does not beep, then you were in input mode, but pressing **<esc>** swit you to visual-command mode.

Most visual-mode commands are one keystroke long. The commands are in two gro movement commands and edit commands. The former group moves the cursor thr the file being edited, and the latter group alters text.

The following sections summarize the command set for **elvis**'s visual command mode

Visual-Mode Movement Commands

The following summarizes the visual mode's movement commands. *count* indicates the command can be optionally prefaced by an argument that tells **elvis** how often t ecute the command. *move* indicates that the command can be followed by a move command, after which the command is executed on the text that lies between the p where the command was first typed and the point to which the cursor was mo Typing the command a second time executes the command for the entire line which the cursor is positioned. *key* means that the command must be followed by a gument. The following describes

<ctrl-B> Move up by one screenful.

[count] **<ctrl-D>**
Scroll down *count* lines (default, one-half screenful).

[count] **<ctrl-E>**
Scroll up *count* lines.

<ctrl-F>	Move down by one screenful.
<ctrl-G>	Show file status and the current line line.
[count] **<ctrl-H>**	Move one character to the left.
[count] **<ctrl-J>**	Move down *count* lines.
<ctrl-L>	Redraw the screen.
[count] **<ctrl-M>**	Move to the beginning of the next line.
[count] **<ctrl-N>**	Move down *count* lines (default, one).
[count] **<ctrl-P>**	Move up *count* lines (default, one).
<ctrl-R>	Redraw the screen.
[count] **<ctrl-U>**	Scroll up *count* lines (default, one-half screenful).
[count] **<ctrl-Y>**	Scroll down *count* lines.
<ctrl-]>	If the cursor is on a tag name, go to that tag.
<ctrl-^>	Switch to the previous file.
[count] **<space>**	Move right *count* spaces (default, one).
! *[move]*	Run the selected text through an external filter program.
" *key*	Select which cut buffer to use next.
$	Move to the end of the current line.
%	Move to the matching (){}[] character.
' *key*	Move to a marked line.
[count] **(**	Move backward *count* sentences (default, one).
[count] **)**	Move forward *count* sentences (default, one).
*****	Go to the next error in the error list.
[count] **+**	Move to the beginning of the next line.

[count] **,**
: Repeat the previous **f** or **t** command, but move in the opposite d
tion.

[count] **—**
: Move to the beginning of the preceding line.

[count] **.**
: Repeat the previous *edit* command.

/ text
: Search forward for *text*, which can be a regular expression.

0
: If not part of a count, move to the first character of this line.

:
: Switch to colon-command mode to execute one command.

[count]
: *Repeat the previous* **f** or **t** command.

? text
: Search backwards for *text*, which can be a regular expression.

@ key
: Execute the contents of a cut-buffer as **vi** commands.

[count] **B**
: Move backwards *count* words (default, one).

[count] **E**
: Move forwards *count* words (default, one).

[count] **F** *key*
: Move left to the *count*'th occurrence of the given character (defa
first).

[count] **G**
: Move to to the *count*'th line in the file (default, last).

[count] **H**
: Move to the top of the screen.

K
: Look up a keyword.

[count] **L**
: Move to the bottom of the screen.

M
: Move to the middle of the screen.

N
: Repeat the last search, but in the opposite direction.

P
: Paste text before the cursor.

Q
: Shift to colon-command mode.

[count] **T** *key*
: Move left *almost* to the given character.

U
: Undo all recent changes to the current line.

[count] **U**
: Move forward *count* words (default, one).

[count] **Y**
Copy (or "yank") *count* lines into a cut buffer (default, one).

Z Z Save the file and exit.

[[Move back one section.

]] Move forward one section.

^ Move to the beginning of the current line, but after indent.

' *key* Move to the *key* character.

[count] **b**
Move back *count* words.

[count] **e**
Move forward to the end of the *count*'th word.

[count] **f** *key*
Move rightward to the *count*'th occurrence of the given character.

[count] **h**
Move left *count* characters (default, one).

[count] **j**
Move down *count* characters (default, one).

[count] **k**
Move up *count* characters (default, one).

[count] **l**
Move right *count* characters (default, one).

m *key* Mark a line or character.

n Repeat the previous search.

p Paste text after the cursor.

[count] **t** *key*
Move rightward *almost* to the *count*'th occurrence of the give character (default, one).

u Undo the previous edit command.

[count] **w**
Move forward *count* words (default, one).

y *move*
Copy (or "yank") text into a cut buffer.

z *key* Scroll the screen, repositioning the current line as follows: + i dicates top of the screen, — indicates the bottom, . indicates the mi dle.

[count] **{**
Move back *count* paragraphs (default, one).

[count] |
: Move to the *count*'th column on the screen (leftmost, one).

[count] }
: Move forward *count* paragraphs (default, one).

Visual-Mode Edit Commands

The following describes the visual mode's editing commands.

[count] **#**
: Increment a number by *count* (default, one).

[count] **&**
: Repeat the previous **:s//** command.

< *move*
: Shift the enclosed text left.

> *move*
: Shift the enclosed text right.

[count] **A** *input*
: Append input at end of the line.

C *input*
: Change text from the cursor through the end of the line.

D
: Delete text from the cursor through the end of the line.

[count] **I** *input*
: Insert text at the beginning of the line (after indentations).

[count] **J**
: Join lines the current with the following line.

[count] **O** *input*
: Open a new line above the current line.

R *input*
: Overtype.

[count] **S** input
: Change lines, like **cc**.

[count] **X**
: Delete *count* characters from the left of the cursor (default, one).

[count] **a** *input*
: Insert text after the cursor.

c *move*
: Change text.

d *move*
: Delete text.

[count] **i** *input*
Insert text at the cursor.

[count] **o** *input*
Open a new line below the current line.

[count] **r** *key*
Replace *count* characters with text you type (default, one).

[count] **s** *input*
Replace *count* characters with text you type (default, one).

[count] **x**
Delete the character at which the cursor is positioned.

[count] **~**
Toggle a character between upper case and lower case.

Colon-Mode Commands

The following summarizes the set of colon-mode commands. It is no accident that these commands closely resemble those for the **ed** line editor: they come, in fact, from **ex**, the editor upon which both **vi** (the UNIX visual editor) and **ed** derive. For that reason colon-command mode is sometimes called **ex** mode.

line indicates whether the command can be executed on one or more lines. *line* can be a regular expression. Some commands can be used with an optional exclamation point; if done so, the editor assumes you know what you are doing and suppresses the warnings and prompts it would normally issue for these commands. Please note, finally, that most commands can be invoked simply by typing the first one or two letters of their names.

abbr *[word full_form]*
Define *word* as an abbreviation for *full_form*.

[line] **append**
Insert text after the current line.

args *[file1 ... fileN]*
With no arguments, print the files list on **elvis**'s command line. With one or more arguments, change the name of the current file.

cc *[files]*
Invoke the C compiler to compile *files*, and redirects all error messages into file **errlist**. After the compiler exits, scan the contents of **errlist** for error messages; if one is found, jump to the line and file indicated on the error line, and display the error message on the status line.

cd *[directory]*
Switch the current working directory. With no argument, switch to the **$HOME** directory.

[line][,line] **change** *["x]*
Replace the range of lines with the contents of cut-buffer *x*.

chdir *[directory]*
Same as the **cd** command.

[line][,line] **copy** *targetline*
Copy the range of lines to after the *targetline*.

[line][,line] **delete** *["x]*
Move the range of lines into cut buffer *x*.

digraph*[!] [XX [Y]]*
Set *XX* as a digraph for *Y*. With no arguments, display all currently defined digraphs. With one argument, undefine the argument as a digraph.

edit*[!] [file]*
Edit a file not named on the **elvis** command line.

errlist*[!] [errlist]*
Find the next error message in file **errlist**, as generated through **elvis**'s **cc** or **make** commands.

file *[file]*
With an argument, change the output file to *file*. Without an argument, print information about the current output file.

[line][,line] **global** */regexp/ command*
Search the range of lines for all lines that contain the regular expression *regexp*, and execute *command* upon each.

[line] **insert**
Insert text before the current line.

[line][,line] **join**
Concatenate the range of lines into one line.

[line][,line] **list**
Display the requested range of lines, making all embedded control characters explicit.

make *[target]*
Same as the **cc** command, except that **make** is executed.

map*[!] key mapped_to*
Remap *key* to *mapped_to*. Normally, remapping applies just to visual-command mode; '!' tells **elvis** to remap the key under all modes. With no arguments, show all current key mappings.

[line] **mark** *x*
Set a mark on *line*, and name it *x*.

mkexrc
Save current configuration into file **./.exrc**, which will be read next time you invoke **elvis**.

[line][,line] **move** *targetline*
Move the range of lines to after *targetline*.

next*[!] [files]*
Switch to the next file on the **elvis** command line.

Next*[!]*
Switch to the preceeding file on the **elvis** command line.

[line][,line] **number**
Display the range of lines, with line numbers.

previous*[!]*
Switch to the preceeding file on the **elvis** command line.

[line][,line] **print**
Display the specified range of lines.

[line] **put** *["x]*
Copy text from cut buffer *x* after the current line.

quit*[!]*
Quit **elvis**, and return to the shell.

[line] **read** *file*
Read the contents of *file* and insert them after **line** (default, the last line).

rewind*[!]*
Switch to the first file on the **elvis** command line.

set *[options]*
Set an **elvis** option. With no arguments, list current settings for all options.

shell Invoke a shell.

source *file*
Read a set of colon-mode commands from *file*, and execute them.

[line][,line] **substitute** */regexp/replacement/[p][g][c]*
For the range of lines, replace the first instance of *regexp* with *replacement*. tells **elvis** to print the *last* line upon which a substitution was performed. means perform a global substitution, i.e., replace all instances of *regexp* on ea line with *replacement*. *c* tells **elvis** to ask for confirmation before performing ea substitution.

tag*[!] tagname*
Find *tagname* in file **tags**, which records information about all tags. If foun jump to the file and line upon which the tag is set.

[line][,line] **to** *targetline*
Copy the range of lines to after the *targetline*.

unabbr *word*
Unabbreviate *word*.

undo
Undo the last editing command.

unmap*[!] key*
Unmap *key*.

version
Display the current version of **elvis**.

[line][,line] **vglobal** */regexp/ command*
Search the range of lines for all lines that do not contain the regular expression *regexp*, and execute *command* upon each.

visual
Enter visual-command mode.

wq
Save the changed file, and exit.

[line][,line] **write***[!] [[>>]file]*
Write the file being edited into *file*. With the >> argument, append the edited text onto the end of *file*.

xit*[!]*
Same as the **wq** command, described above, except that it does not write files that have not changed.

[line][,line] **yank** *["x]*
Copy the range of lines into cut buffer *x*.

[line][,line] **!** *command*
Execute *command* under a subshell, then return.

[line][,line] **<**
Shift the range of lines left by one tabwidth.

[line][,line] **=**
With no range of lines specified, print the number of the current line. With line arguments, print the endpoints of the lines in question, and the number of lines that lie between them. (Remember, *line* can be a regular expression as well as a number.)

[line][,line] **>**
Shift the range of lines right by one tabwidth.

[line][,line] **&**
Repeat the last substitution command.

@ x
Read the contents of cut-buffer *x* as a set of colon-mode commands, and execute them.

Input Mode Commands

Most keystrokes are interpreted as being text and inserted directly into the text; however, some keystrokes are still interpreted as commands. Thus, you can perform an entire session of simple editing directly within input mode without switching to either of the command modes.

The following summarizes the commands that can be executed directly within in mode:

<ctrl-A>	Insert a copy of the last input text.
<ctrl-D>	Delete one indent character.
<ctrl-H>	Erase the character before the cursor.
<ctrl-L>	Redraw the screen.
<ctrl-M>	Insert a newline.
<ctrl-P>	Insert the contents of the cut buffer.
<ctrl-R>	Redraw the screen, like **<ctrl-L>**.
<ctrl-T>	Insert an indent character.
<ctrl-U>	Move to the beginning of the line.
<ctrl-V>	Insert the following keystroke, even if special.
<ctrl-W>	Backspace to the beginning of the current word.
<ctrl-Z><ctrl-Z>	Write the file and exit **elvis**.
<esc>	Shift from input mode to visual-command mode.
<del>	Delete the current character.

Command-line Options

elvis lets you name up to 27 files on the command line, thus allowing you to edit up 27 files simultaneously. The "next file" and "previous file" commands described abo allow you to shift from one file to another during the same editing session; in this wa for example, you can cut text from one file and paste it into another.

elvis recognizes the following command-line options:

-r Recover a previous edit. Because **elvis** uses the program **virec** for file recover invoking it with this option simply prints a message that tells you to run **virec**.

-R Invoke **elvis** in "read-only" mode. This is equivalent to invoking **elvis** via th link **view**.

-t *tag*
Begin editing at *tag*.

-m *[file]*
Invoke **elvis** in error-handling mode. It searches through *file* for something tha looks like an error message from a compiler, then positions the cursor at tha point for editing.

-e Begin in colon-command mode.

-v Begin in visual-command mode.

-i Begin in input mode.

+command
Execute *command* immediately upon beginning editing. For example

```
elvis +237 foo
```

causes *elvis* to move directly to line 237 immediately upon beginning to edit file **foo**.

Files

/tmp/elv* — Temporary files

See Also

commands, ed, ex, me, vi, view

Notes

Full documentation for **elvis** is included with this release in compressed file **/usr/src/alien/Elvis.doc.Z**.

elvis is a public-domain program written by Steve Kirkendall (kirkenda@cs.pdx.edu or ...uunet!tektronix!psueea!eecs!kirkenda), assisted by numerous volunteers. Source code for it is available through the Mark Williams bulletin board, USENET and numerous other outlets.

elvis is distributed as a service to COHERENT customers, as is. It is not supported by Mark Williams Company. *Caveat utilitor.*

ex — Command

Line-oriented editor
ex [*flags*] [*+cmd*] [*file1 ... file27*]

ex is a link to **elvis**, which is a clone of the UNIX **vi/ex** set of editors. Invoking **elvis** through this link forces it to operate solely in colon-command mode, just as the UNIX **ex** editor operates.

For information on how to use this version of **ex**, see the Lexicon page for **elvis**.

See Also

commands, ed, elvis, me, vi, view

Notes

elvis is a public-domain program written by Steve Kirkendall (kirkenda@cs.pdx.edu or ...uunet!tektronix!psueea!eecs!kirkenda), assisted by numerous volunteers. Source code for it is available through the Mark Williams bulletin board, USENET and numerous other outlets.

elvis is distributed as a service to COHERENT customers, as is. It is not supported by Mark Williams Company. *Caveat utilitor.*

fdformat — Command

Format a floppy disk
/etc/fdformat [*option* ...] *special*

fdformat formats a floppy disk. The given *special* should be the name of the special f that correspond to the floppy disk drive.

fdformat recognizes the following options:

-a Print information on the standard output device during format. As it forma cylinder, it will print a line of the form

```
hd=0 cyl=25
```

on your screen.

-i *number*
Use *number* (0 through 7; default, 6) as the interleave factor in formatting.

-o *number*
Use *number* (default, 0) as the skew factor for sector numbering.

-v Verify formatting and verify data written with the **-w** option.

-w *file*
Format the floppy disk and then copy *file* to it track by track. The raw de should be used.

The command **mkfs** builds a COHERENT file system on a formatted floppy disk. T command **dos** builds a DOS file system on a formatted floppy disk and transfers files or from it. The command **mount** mounts a floppy disk containing a file system to all access to it through the COHERENT directory structure. The command **umount** mounts a floppy disk.

Example

The following command formats a 2400-block, 5.25-inch floppy disk in drive 0 (otherwi known known as drive A):

```
/etc/fdformat /dev/fha0
```

The following command formats a 1440-block, 3.5-inch floppy disk in drive 1 (otherwi known as drive B):

```
/etc/fdformat /dev/fqa1
```

See Also

commands, dos, fd, mkfs, mount, umount

Diagnostics

When errors occur on floppy-disk devices, the driver prints on the system console an e ror message that describes the error.

Notes

fdformat formats a track at a time. **fdformat** can be interrupted between tracks, which may result in a partially formatted floppy disk.

floppy disks — Technical Information

To use a floppy disk with COHERENT, you must:
(1) format it with **/etc/fdformat**,
(2) build an empty filesystem on it with **/etc/mkfs**,
(3) mount it with **/bin/mount** or **/etc/mount**,
(4) copy files to or from it, e.g. with **cp** or **cpdir**, and
(5) unmount it with **/bin/umount** or **/etc/umount**.

Some commonly used diskette device names and formats are:

Device name	Sectors/track	Heads	Sectors	Bytes	Format
/dev/f9a0	9	2	720	360 KB	5.25"
/dev/fqa0	9	2	1440	720 KB	3.5"
/dev/fha0	15	2	2400	1.2 MB	5.25"
/dev/fva0	18	2	2880	1.44 MB	3.5"

Device names ending in '0' indicate drive A:, names ending in '1' indicate drive B:.

For example, to copy directory **/dir** to a 5.25" high density diskette in drive 0 (A:):

```
/etc/fdformat -a /dev/fha0
/etc/badscan -v -o proto /dev/fha0 2400
/etc/mkfs /dev/fha0 proto
rm proto
/etc/mount /dev/fha0 /f0
cpdir -vd /dir /f0/dir
/etc/umount /dev/fha0
```

/bin/mount and **/bin/umount** provide handy abbreviations for mount and umount commands. For example,

```
mount f0
cpdir -vd /dir /f0/dir
umount f0
```

is a more convenient way to perform the last three commands in the above example.

See Also

fd, fdformat, technical information

Notes

Because COHERENT does not write cached disk data to the disk until a **sync** occurs or the disk device is unmounted, removing a disk from the disk drive without unmounting it can produce incorrect data or an invalid filesystem on the disk. Another disk inserted into the drive after a disk has removed without unmounting may be clobbered by data intended for the first disk. Always be sure to unmount disks before removing them from the disk drive.

fsck — Command

Check and repair file systems interactively
/etc/fsck [**-fnqy**] [**-t** *tempfile*] [*filesystem ...*]

fsck checks and interactively repairs file systems. If all is well, **fsck** merely prints t number of files used, the number of blocks used, and the number of blocks that are fr If the file system is found to be inconsistent in one of the aspects outlined below, **fs** asks whether it should fix the inconsistency and waits for you to reply **yes** or **no**.

The following file system aspects are checked for consistency by **fsck**:

- If a block is claimed by more than one i-node, by an i-node and the free list, or mo than once in the free list.
- Whether an i-node or the free list claims blocks beyond the file system's range.
- Link counts that are incorrect.
- Whether the directory size is not aligned for 16 bytes.
- Whether the i-node format is correct.
- Whether any blocks are not accounted for.
- Whether a file points to an unallocated i-node.
- Whether a file's i-node number is out of range.
- Whether the super block refers to more than 65,536 i-nodes.
- Whether the super block assigned more blocks to the i-nodes than the system co tains.
- Whether the format of the free block list is correct.
- Whether the counts of the total free blocks and the free i-nodes are correct.

fsck prints a warning message when a file name is null, has an embedded slash '/', not null-padded, or if '.' or '..' files do not have the correct i-node numbers.

When **fsck** repairs a file system, any file that is orphaned (that is, allocated but n referenced) is deleted if it is empty, or copied to a directory called **lost+found**, with its node number as its name. The directory **lost+found** must exist in the root of the f system being checked before **fsck** is executed, and it must have room for new entri without requiring that new blocks be allocated.

fsck accepts the following options:

-f Fast check. **fsck** only checks whether a block has been claimed by more than one node, by an i-node and the free list, or more than once in the free list. If necessa **fsck** will reconstruct the free list.

-n No option: a default reply of **no** is given to all of **fsck**'s questions.

-q Quiet option: run quietly. **fsck** automatically removes all unreferenced pipes, a automatically fixes list counts in the super block and the free list. File-name wa ning messages are suppressed, but **fsck** still prints the number of files used, t number of blocks used, and the number of blocks that remain free.

-t Specify temporary file option: **fsck** uses RAM device **/dev/rram1** for temporary storage when checking filesystems larger than approximately 35 megabytes. This option allows the user to specify temporary storage other than the RAM device.

-y Yes option: a default reply of **yes** is given to all of **fsck**'s questions.

If you do not name a file system in **fsck**'s command line, **fsck** checks the file systems named in the file **/etc/checklist**.

If **fsck** is invoked for a file system larger than approximately 35 megabytes, it uses the RAM device **/dev/rram1** for temporary storage. For this reason, it is strongly advised that you not use **/dev/rram1** as a RAM disk.

Files

/etc/checklist

See Also

clri, commands, icheck, ncheck, ram, sync, umount

Notes

The correction of file systems almost always involves the destruction of data.

You can run **fsck** only when the COHERENT system is in single-user mode.

Previous editions of **fsck** could check no partition larger than 35 megabytes. This restriction has been lifted.

hp — Command

Prepare files for HP LaserJet-compatible printer
hp [-acfl] [-i*marg*] [-t*top*] [-p*lines*] *file* ...

The **hp** command translates **nroff** font specifications into the correct escape sequences for an HP LaserJet compatible printer. It also allows the user to set indentation, page length, landscape mode, and so on. Because some LaserJet printers stack pages in reverse order as they are printed, **hp** can put pages out in reverse order.

Option **-f** prints pages in the normal order; without this option, **hp** prints pages in reverse order.

Option **-i***marg* sets the indent to the given *marg*.

Option **-l** prints pages in landscape mode.

Option **-p***lines* sets the page length to *lines*.

Option **-t***top* sets the top margin to *top*.

See Also

commands, hpd

hs — Device Driver

Device driver for polled serial ports

The COHERENT **hs** driver adds support for up to eight serial lines, **/dev/hs00** through **/dev/hs07**.

Serial lines controlled via the **hs** driver can be opened in one of two ways, as follows:

/dev/hs?? — Polled, local mode (no modem control).

/dev/hs??r — Polled, remote mode (modem control).

Any port used with the **hs** device driver will be polled, i.e. interrupt operation is r used. Please refer to the **com** Lexicon article for explanations of "local" vs "remot and "polled" vs "interrupt-driven".

To use the **hs** driver, first configure it to match your equipment (see below), then lo the driver using the following command while running as **root** (the superuser):

```
/etc/drvld -r /drv/hs
```

To unload the driver without rebooting Coherent, first use the **ps** command with the option to get the PID number for the **hs** driver process, then unload the driver proce by using the **kill** command. Note that the **hs** driver process will not unload until **opened** ports have been closed. For example (user input shown in bold):

```
$ ps -d
TTY        PID
-------      0  <idle>
-------     38  <hs>
...
$ kill kill 38
```

The present version of COHERENT limits "polled" operation to one device driver at time. Therefore, if any of the **com** family of devices is used in polled mode, **hs** devic cannot be used. Conversely, **/dev/com1pl** through **/dev/com4pl** and **/dev/com1** through **/dev/com4pr** cannot be used if the **hs** driver is in use. Both drivers can present at the same time, but polled devices may not be **open** under both drivers at t same time. Note that enabling a port via **/etc/enable** keeps it open continuously.

Port Configuration

The default configuration for the **hs** driver is for four ports, at hexadecimal address 0x3F8, 0x2F8, 0x3E8, and 0x2E8, at a speed of 9600 baud. The driver is configured setting the following parameters.

1. the number of ports
2. the I/O address for each port
3. the default speed of each port

All steps in the configuration should be done as **root** (the superuser). Patch the numb of ports into driver variable **HSNUM_**. For example, if you wish to support three por enter:

```
/conf/patch /drv/hs HSNUM_=3
```

Address and speed information are stored sequentially starting at variable **HS_PORTS** The speed for each port is indicated by the corresponding value found in **<sgtty.h**

from 1, corresponding to 50 baud, to 16, corresponding to 9600 baud. If the three ports in the example above are at hexadecimal adresses of 0x2A0, 0x2B0, and 0x2C0, with speeds of 2400, 2400, and 9600 baud, respectively, then the following three patches should be performed:

```
/conf/patch /drv/hs HS_PORTS_=0x2A0 HS_PORTS_+2=12
/conf/patch /drv/hs HS_PORTS_+4=0x2B0 HS_PORTS_+6=12
/conf/patch /drv/hs HS_PORTS_+8=0x2C0 HS_PORTS_+10=16
```

Finally, nodes must be created for each port using the **mknod** command. The major device number is 7; the minor number will range from 0 through 7 for ports **/dev/hs00** through **/dev/hs07**, respectively, with 128 added to the device minor number if modem control is desired. The following commands will make nodes in /dev for local and remote versions of the three ports in the example:

```
/etc/mknod -f /dev/hs00  c  7  0
/etc/mknod -f /dev/hs01  c  7  1
/etc/mknod -f /dev/hs02  c  7  2
/etc/mknod -f /dev/hs00r c  7  128
/etc/mknod -f /dev/hs01r c  7  129
/etc/mknod -f /dev/hs02r c  7  130
```

See Also

com, device drivers, drvld

Diagnostics

An attempt to open a non-existent device will generate error messages. This can occur if hardware is absent or not turned on.

install — Technical Information

Building distribution kits for use by install

/etc/install provides a standardized mechanism to install upgrades and add-on software to the COHERENT system.

install is invoked with a command line of the form:

/etc/install *id device ndisks*

where *id* specifies the update or add-on package to be installed, *device* is the "mountable" block device from which the update or add-on package will be installed, and *ndisks* specifies the number of diskettes in the distribution.

The *id* field should be formed from the set of upper- and lower-case letters, as well as digits, the period (.), and the underscore (_) character. Its length cannot exceed nine characters. Field *device* should be selected from the standard block-special floppy-disk devices.

Distribution Details

Distributions usable by **install** consist of a set of mountable floppy disks, each containing a file system created by **mkfs**. This allows the diskettes to be independent of each other and also allows the user to insert the diskettes in any order. **install** records the fact that it has read a given diskette from the distribution, thus preventing the user from at-

tempting to read a given diskette more than once during an installation session.

Diskettes should be built using **mkfs**, with possible input being generated by **unmki** Each diskette in the distribution must contain a file of the form

/id.sequence

in the root directory. Here, *id* must match the aforementioned field of the same nam The *sequence* part of the file name indicates which diskette in the distribution this di kette is, from one through the total number of diskettes.

install uses the command **cpdir** to copy each of the distribution diskettes to directory on the current system. Therefore, all diskettes should be "root based" (i.e., full pa names should be used). Because **install** is run by the superuser, **cpdir** preserves t date and time for each file, and preserves ownership and modes. To keep file ownersh consistent with COHERENT conventions, make files that are neither *setuid* nor *setg* owned by user **bin** and group **bin**. Directories found on the distribution diskettes will created on the target file system, as needed. Be careful when choosing the ownersh and mode of directories because your system's security could inadvertently be cor promised.

Postprocessing

After all diskettes in a distribution have been successfully copied, **install** checks for t existence of a file of the form

/conf/*id*.post

where *id* matches the *id* field found on the **install** command line. If found, **insta** executes this file to allow special "postprocessing," such as installing additional manu pages or performing installation-specific commands.

Before you complete your postprocessing, you should remove any *id* files of the followi form from the target system:

/conf/*id*.post
/id.sequence

Adding Manual-Page Entries

As part of building a distribution, you will usually need to generate pre-processed "cooked" manual-page entries for distribution with your upgrade or add-on packag These should reside in subdirectories of **/usr/man**, with the name of the subdirecto being specific to your product. This naming convention avoids name-space collision should multiple applications use the same name for a manual-page entry.

If you install new or additional manual pages, you must update the index file used by t **man** command to locate manual entries. File **/usr/man/man.index** on the target fi system contains index entries for all manual pages on the system. As part of postproce sing, you will generally need to append index information for your manual pages to t end of the existing index file. In addition, file **/usr/man/man.help** contains the **ma** command's help message. This includes a list of valid topics and some explanatory tex You should also append to this file a brief list of the manual page entries that you ha added. For further information on manual pages, see the Lexicon entry for the co mand **man**.

Logging

install logs all partial as well as completed installations in file **/etc/install.log**. This information includes date/time stamps and the command line arguments to **install**.

Files

/etc/install.log

See Also

install (command), man, mkfs, technical information, unmkfs

kill() – COHERENT System Call (libc)

Kill a system process
#include <signal.h>
kill(*pid, sig***)**
int *pid, sig*;

kill() is the COHERENT system call that sends a signal to a process. *pid* is the process identifier of the process to be signalled, and *sig* identifies the signal to be sent, as set in the header file **signal.h**. This system call is most often used to kill processes, hence its name.

See Also

COHERENT system calls, signal(), signal.h

libraries – Overview

A **library** is an archive file of commonly used functions that have been compiled, tested, and stored for inclusion in a program at link time.

The COHERENT system stores its libraries in two directories, **/usr/lib** and **/lib**. The following libraries are kept in **/usr/lib**:

libcurses.a	**curses** library
libl.a	**lex** library
libmp.a	Multi-precision arithmetic library
libterm.a	**termcap** library
liby.a	**yacc** library
lib.b	**bc**'s function library (in **bc** source)

The following libraries are kept in **/lib**:

libc.a	General functions and system calls
libm.a	Mathematics routines

Library Functions

The following overview articles introduce the library functions included with the COHERENT system:

COHERENT system calls
ctype macros
curses
general functions
mathematics library
multiple-precision mathematics
STDIO
string functions
terminal-independent operations
time

See Also

ar, C language

logmsg – System Maintenance

Hold COHERENT Login Message
/etc/logmsg

The file **/etc/logmsg** holds the message that COHERENT displays to prompt the use to log in. The superuser **bin** can use **ed** or MicroEMACS to change the message t whatever she prefers.

See Also

system maintenance

Notes

The default message consists of the bell character **<ctrl-G>** followed by the te **Coherent login:**. If the bell annoys you, simply delete the **<ctrl-G>** fro **/etc/logmsg**.

major number – Definition

Device numbering

A *major number* specifies the device driver associated with a given device name found i the directory **/dev**. COHERENT uses a device's the major number as an index into a internal table of device-driver pointers.

Every COHERENT device has a device number associated with it. This device numb is of type **dev_t**, as defined in **<sys/types.h>**. The macro **major()** in **<sys/stat.h** extracts the major number from a given device number.

See Also

device drivers, minor number, stat.h

mail – Command

Computer mail
mail [-mpqrv] [-f *file***] [***user* ...**]**

mail allows you to exchange electronic mail with other COHERENT system user either on your own system or on other systems via UUCP. If one or more *users* a specified, **mail** reads a message from the standard input, appends the date and the se

der's name, and sends the result to each *user*. **mail** prints the prompt

```
Subject:
```

on the screen, requesting that you give the message a title. A message can be terminated with an end-of-file character (**<ctrl-D>**), a line that contains only the character '.', or a line that contains only the character '?'. If a message is ended with a question mark, **mail** feeds the message into an editor for further editing. The editor used is the one named in the user's **.profile** with the command line **export EDITOR=***editor*; if no editor is named in **.profile**, it uses **ed**. After a message is ended, if you have defined environmental variable **ASKCC** to **YES**, you will be asked for a list of users to carbon copy the message to.

mail looks up each *user* in file **/usr/lib/mail/aliases**. If there is a match, the new name is used in place of *user*. If *user* is of the form

sys!*user*

or

sys! ... !*user*

it is treated as a UUCP destination. **mail** then invokes **uucp** command to pass the message to *sys*, whose responsibility it becomes to pass the message to *user*.

For local users, **mail** writes its messages into the file **/usr/spool/mail/***user*. This file is called the user's "mailbox"; Each *user* who has received mail is greeted by the message "You have mail." when she logs in. **mail** normally changes the contents of the mailbox as the user works with them; however, **mail** has options that allow the contents of the mailbox to remain unchanged if the user desires.

If no *user* is given, **mail** reads and displays the user's mail message by message. If environmental variable **PAGER** is defined, **mail** will "pipe" each message through the command specified in **PAGER**. For example, the **.profile** command line:

```
export PAGER="exec /bin/scat -1"
```

would invoke **/bin/scat** for each mail message with the command line argument **-1** (the digit one).

The following commands allow the user to save, delete, or send each message to another user interactively.

d Delete the current message and print the next message.

m [*user* ...]
Mail the current message to each *user* given (default: yourself).

p Print the current message again.

q Quit, and update mailbox file to reflect changes.

r Reverse the direction in which the mailbox is being scanned.

s [*file* ...]
Save the current mail message with the usual header in each *file* (default: **$HOME/mbox**).

t [*user* ...]
Send a message read from the standard input, terminated by an end-of-f character or by a line containing only '.' or '?', to each *user* (default: yourself).

w [*file* ...]
Write the current message without the usual header in each *file* (defau **$HOME/mbox**).

x Exit without updating the mailbox file.

<newline>
Print the next message.

- Print the previous message.

EOF Quit, updating mailbox; same as **q**.

? Print a summary of available commands.

!*command*
Pass *command* to the shell for execution.

The following command line options control the sending and reading of mail.

-f *file* Read mail from *file* instead of from the default, **/usr/spool/mail/***user*.

-m Send a message to the terminal of *user* if she is logged into the system wh mail is sent.

-p Print all mail without interaction.

-q Quit without changing the mailbox if an interrupt character is typed. Normal an interrupt character stops printing of the current message.

-r Reverse the order of printing messages. Normally, **mail** prints messages in t order in which they were received.

-v Verbose mode. Show the version number of the **mail** program, and display e panded aliases.

If you wish, you can create a signature file, **.sig.mail**, in your home directory. **mail** a pends the contents of the signature file to the end of every mail message, as a signatu A signature can be your system's path name (for **uucp** messages), your telephone nu ber, an amusing *bon mot*, or what you will.

Files

$HOME/dead.letter — Message that **mail** could not send
$HOME/mbox — Default saved mail
$HOME/.aliases — Personal mail alias file
$HOME/.sig.mail — Signature file
/etc/passwd — User identities
/etc/utmp — Logged in users
/tmp/mail* — Temporary and lock files
/usr/lib/mail/aliases — Aliases of users
/usr/spool/mail — Mailbox directory, filed by user name

See Also

ASKCC, commands, msg, PAGER, write

Notes

mail stores mail for a given *user* in file **/usr/spool/mail/***user*. *user* owns this file, and can therefore permit or deny access to the mail by other users.

man — Command

Print online manual sections
man [**-w**] [*topic* ...]

man prints the COHERENT lexicon entries for each specified *topic* on the standard output. It uses **scat** to display text (with the **-s** option to suppress blank lines). With no arguments, **man** prints a list of each available *topic*.

When used with the **-w** option, it prints the path name of the file instead of printing the document itself.

If environmental variable **PAGER** is defined, **man** will "pipe" its output through the command specified in **PAGER**. For example, the **.profile** command line:

```
export PAGER="exec /bin/scat -l"
```

would invoke **/bin/scat** with the command line argument **-l** (the digit one).

Manual-Page Control Files

The **man** command uses two control files when processing manual-page requests. File **/usr/man/man.help** contains the **man**'s help message. This includes a list of valid topics and some explanatory text. The second control file, **/usr/man/man.index**, contains index entries for all manual pages on the system. Lines in this text file are of the form:

relative-path-name topic

where *relative-path-name* gives the subdirectory and file in **/usr/man** that hold the manual-page entry, and *topic* gives a manual-page topic associated with this file. For example, entries

```
COHERENT/ascii  ascii
COHERENT/ascii  ASCII
local/chess  chess
```

associate system manual-page **/usr/man/COHERENT/ascii** with either upper- or lower-case spellings of topic **ascii**. Similarly, rules for a user-written chess game are found in file **/usr/man/local/chess** and are retrieved using topic **chess**.

Adding Manual-Page Entries

When writing new manual-page entries for COHERENT, we recommend that you place them in subdirectories of **/usr/man**. These subdirectories should be uniquely named to avoid possible name-space collisions. A good rule-of-thumb is to name the subdirectory after the application with which it is associated. This also allows them to be updated easily, as all manual-pages associated with a given application reside in a specific subdirectory.

When you add manual-page entries to the system, you should also append a list of
to **/usr/man/man.help**. In addition, you must append a line to the end c
/usr/man/man.index for each newly added topic.

Files

/usr/man/* — Directories that hold manual pages

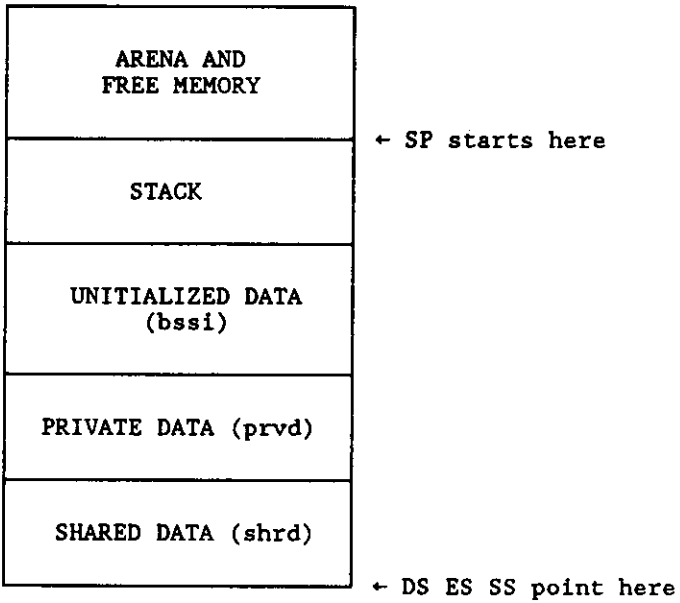
See Also

commands, help, install, PAGER, scat

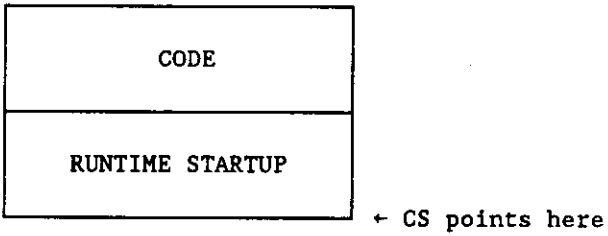
memory allocation — Technical Information

The following diagram shows how COHERENT allocates memory.

Data Segment (maximum size 64 kilobytes)



Code Segment (maximum size 64 kilobytes)



Note that COHERENT can relocate the code and data segments at its own convience and merely repoint the required segment registers.

The stack *descends* from the highest address in its space toward the static data area; new arguments are placed on the stack in its *lowest* address. Everything from the top of the stack space to the end of the data segment is free to accept dynamically allocated data.

The size of the stack cannot be altered while a program is running. By default, the run-time startup sets the stack size to two kilobytes (2,048 bytes) Note, however, that a highly recursive function may cause the stack to grow larger than two kilobytes so that it overwrites other data areas. This will cause your program to work incorrectly. To reset the amount of stack allocated to a program, use the command **fixstack**.

See Also

data formats, fixstack, technical information

minor number – Definition

Device numbering

A *minor number* specifies the device or type of device to use. COHERENT uses
minor number of a given device in a driver-specific manner. For example, a hard-d
driver may use the minor number to select a disk drive and partition.

Every COHERENT device has a device number associated with it. It is of type **dev_t**,
defined in **<sys/types.h>**. The macro **minor()** in **<sys/stat.h>** extracts the mi
number from a given device number.

See Also

device drivers, major number, stat.h

misc – Technical Information

Library of miscellaneous functions

misc is a collection of library routines. These routines are useful for handling s
programming tasks as allocation of memory, copying of strings, displaying variables fr
C with COBOL-like "picture" descriptions, and supporting virtual arrays via second
storage.

Source code for the library is kept in the compressed **tar** archive
/usr/src/alien/misc.tar.Z. To extract the files into a new subdirectory called **misc**,
the command:

```
zcat /usr/src/alien/misc.tar.Z | tar xvf -
```

To build the library, type the following:

```
cd misc
make
```

For a full description of each function, consult the included **Read_me** file.

Files

/usr/src/alien/misc.tar.Z — Compressed **tar** archive of sources

See Also

tar, technical information, zcat

Notes

The **misc** library is provided on an *as-is* basis only. *Caveat utilitor!*

mknod() – COHERENT System Call (libc)

Create a special file

#include <sys/ino.h>
#include <sys/stat.h>
mknod(*name, mode, addr*)
char **name*; **int** *mode, addr*;

mknod is the COHERENT system call that creates a special file. A *special file* is one through which a device is accessed, or a named pipe.

mode gives the type of special file to be created. It can be set to **IFBLK**, for a block-special device, such as a disk driver; to **IFCHR**, for a character-special device, such as a serial-port driver; or to **IFPIPE**, for a named pipe.

address is a parameter interpreted by the driver; it might specify the channel of a multiplexor or the unit number of a drive. Note that this is not used with named pipes.

See Also

COHERENT system calls, pipe

modemcap — Technical Information

Modem description language

modemcap is a language for describing modems to your system. It resembles the **termcap** language in its syntax, although the two are by no means identical. With **modemcap**, you can describe your modem to any program that automatically dials out on your modem; this should spare you the tedium of continually describing your modem to one program after another.

The copy of **/etc/modemcap** included with your release of COHERENT already contains descriptions of many popular modems; the chances are good that yours has already been described for you.

Each **modemcap** command is one of three types: *flag*, *string*, or *number*. A *flag* command signals that your modem is performs a particular action or has a particular feature. A *string* command gives the command that your modem recognizes to perform a particular action. For example, many modems recognize that the string **at** means that you want to gain its attention. Finally, a *number* command sets a value or parameter for your modem, such as the highest baud rate it recognizes.

The following table describes each **modemcap** command:

Name	*Type*	*Meaning*
ad	number	Delay after **as**
as	flag	Numbers are in ASCII, not binary
at	string	Attention string, forces model into command mode from online mode
bd	number	Highest online baud rate
bl	number	Alternate lower baud rate
ce	string	Command end string (required if CS is present)
cl	string	String from modem on remote connection at BL baud rate
co	string	String from modem on remote connection at BD baud rate
cs	string	Command start string
de	string	End dial command string (required if DS is present)
di	flag	Modem has a dialer
ds	string	Start dial command string
id	number	Delay after IS
is	string	Initialization string, resets modem to offline, ready to dial

hc	**flag**	Modem hangs up when DTR drops
hu	**string**	Hangup command
tt	**flag**	Modem dials touchtone by default (or DS is set that way)

All commands, such as **ds** (dial command) and **hu** (hang up) will be prefixed by **cs** a ended with **ce**. If there is a common prefix and suffix, use this feature. Otherwise, e command will have to have the entire string built in.

Example Entry

The following gives the entry in **/etc/modemcap** for the Hayes Smartmodem 1200:

```
hy|hayes|Hayes Smartmodem 1200:\
      :as:at=+++:ad#3:bd#1200:bl#300:cs=AT:ce=\r:co=CONNECT:\
      :cl=CONNECT:di:ds=DT :de=:is=ATQ0 V1 E1\r:id#2:\
      :hc:hu=H0 V0 E0 Q1:tt:
```

Each field is separated by a colon. A backslash '\' character at the end of each line the last lets the description extend over more than one line.

The three fields gives three versions of the modem's name, separated by vertical bars The first version of the name is a two-character mnemonic; this must be unique. T other two versions give fuller versions of the name; these are optional.

The following explains each field in detail:

as	Numbers are in binary mode.
at=+++	To gain the attention of the modem, type +++.
ad#3	Delay three milliseconds after a number.
bd#1200	Maximum baud rate is 1200.
bl#300	Minimum baud rate is 300.
cs=AT	To initiate a command string, type **AT**.
ce=\r	A command string is ended by a carriage-return character.
co=CONNECT	Modem returns the string **CONNECT** when it makes a conn tion at 1200 baud.
cl=CONNECT	Modem returns the string **CONNECT** when it makes a conn tion at 300 baud.
di	The modem can dial a telephone number.
ds=DT	Begin dialing, touch-tone mode.
de=	No special string is needed to end the dial string.
is=ATQ0	To initialize the modem, type **ATQ0 V1 E1<return>**.
id#2	Delay two seconds after entering the initialization string.

hc The modem hangs up when DTR drops (i.e., it hangs up when the program requests a hangup).

hu=H0 To hang up, type **H0 V0 E0 Q1**.

tt The modem dials touch-tone by default.

Currently Recognized Modems

The file **/etc/modemcap** includes descriptions of the following modems:

Trailblazer, 9600 baud
Trailblazer, 2400 baud
Hayes Smartmodem 1200
Avatex 2400 (clone of Hayes Smartmodem 2400)
Prometheus Promodem 1200
Signalman Mark XII
Radio Shack Direct-Connect 300 Modem

See Also

technical information, termcap

motd – Technical Information

File that holds message of the day
/etc/motd

The file **motd** holds the message of the day. Its contents are displayed on every user's screen whenever he logs in.

Only the superuser can alter the contents of this file.

See Also

technical information

mount() – COHERENT System Call (libc)

Mount a file system
#include <sys/mount.h>
#include <sys/filsys.h>
mount (*special, name,flag*)
char **special*, **name*; **int** *flag*;

mount() is the COHERENT system call that mounts a file system. *special* names the physical device that through which the file system is accessed. *name* names the root directory of the newly mounted file system. *flag* controls the manner in which the file system is mounted, as set in header file **sys/mount.h**.

See Also

COHERENT system calls, fd

msg – Command

Send a one-line message to another user
msg *user*
message

The command **msg** prints the one-line *message* on the screen of *user*.

The message is send as soon as you type **<return>** on the *message* line. If *user* is logged in or is not known to the system, **msg** prints an error message on your screen.

See Also
commands

msgs — Command

Read messages intended for all COHERENT users
msgs [*-q*] [*number*]

msgs selects and displays messages that are intended to be read by all COHERE users. Messages are mailed to the login **msgs**. They should contain information me to be read once by most users of the system.

The command **msgs** normally is in a user's **.profile**, so that it is executed every time logs in. When invoked, it prompts the user with the identifier of the user who sent message and the message's size. **msgs** then asks the user if he wishes to see the res the message. The user should reply with one of the following:

y	Display the message.
<return>	Display the message.
n	Skip this message and go to the next one.
.	Redisplay the last message.
q	Quit **msgs**.
number	Display message *number*; then continue.

If environmental variable **PAGER** is defined, **msgs** will "pipe" each message thro the command specified in **PAGER**. For example, the **.profile** command line:

```
export PAGER="exec /bin/scat -l"
```

would invoke **/bin/scat** for each message with the command line argument **-l** (the d one).

msgs writes into the file **$(HOME)/.msgsrc** the number of the next message the u will see when he invokes **msgs**. **msgs** keeps all messages in the directory **/usr/ms** each message is named with a sequential number, which indicates its message numb The file **/usr/msgs/bounds** contains the low and high numbers of the messages in directory; **msgs** determines whether a user has not read a message by comparing the formation in **$(HOME)/.msgsrc** with that in **/usr/msgs/bounds**. If the contents **/usr/msgs/bounds** are incorrect, the problem can be fixed by removing that file; **m** will create a new **bounds** file the next time it is run.

When the contents of a message are no longer needed, simply remove that messa Avoid removing the **bounds** file and the highest numbered message at the same time.

msgs accepts the following command-line options:

-q Query whether there are messages; print "There are new messages" if there a and "No new messages" if not. The command **msgs -q** is often used in pro scripts.

number Start at message *number* rather than at the message recorded in **$(HOME)/.msgsrc**. If *number* is greater than zero, then start with that message; if *number* is less than zero, then begin *number* messages before the one recorded in **$(HOME)/.msgsrc**.

Files

/usr/spool/mail/msgs — Mail messages file
/usr/msgs/[1-9]* — Data base
/usr/msgs/bounds — File that contains message number bounds
$(HOME)/.msgsrc — Number of next message to be presented

See Also

commands, mail, PAGER, scat

PAGER — Environmental Variable

Specify Output Filter
PAGER="*command options*"

The environmental variable **PAGER** directs programs such as **msgs**, **mail** and others to "pipe" their output into the *command* specified as the value of **PAGER**. For example, the following sets up **/bin/scat** as the desired output filter and passes a command line option to it to specify that the output screen has 20 lines.

```
export PAGER="exec /bin/scat -120"
```

See Also

scat, environmental variables, mail, msgs

path() — General Function

Path name for a file
#include <path.h>
#include <stdio.h>
char **path**(*path*, *filename*, *mode*);
char **path*, **filename*;
int *mode*;

The function **path** builds a path name for a file.

path points to the list of directories to be searched for the file. You can use the function **getenv** to obtain the current definition of the environmental variable **PATH**, or use the default setting of **PATH** found in the header file **path.h**, or, you can define *path* by hand.

filename is the name of the file for which **path** is to search. *mode* is the mode in which you wish to access the file, as follows:

1	Execute the file
2	Write to the file
4	Read the file

path calls the function **access** to check the access status of *filename*. If **path** finds the file you requested and the file is available in the mode that you requested, it returns a pointer to a static area in which it has built the appropriate path name. It returns

NULL if either *path* or *filename* are NULL, if the search failed, or if the requested f not available in the correct mode.

Example

This example accepts a file name and a search mode. It then tries to find the file in of the directories named in the **PATH** environmental variable.

```
#include <path.h>
#include <stdio.h>
#include <stdlib.h>

void
fatal(message)
char *message;
{
     fprintf(stderr, "%s\n", message);
     exit(1);
}

main(argc, argv)
int argc; char *argv[];
{
     char *env, *pathname;
     int mode;

     if (argc != 3)
          fatal("Usage: findpath filename mode");

     if(((mode=atoi(argv[2]))>4) || (mode==3) || (mode<1))
          fatal("modes: 1=execute, 2=write, 3=read");

     env = getenv("PATH");
     if ((pathname = path(env, argv[1], mode)) != NULL) {
          printf("PATH = %s\n", env);
          printf("pathname = %s\n", pathname);
          return;
     } else
          fatal("search failed");
}
```

See Also

access(), access.h, general functions, PATH, path.h

pax — Command

Portable archive interchange

pax is an archiving utility that reads and writes **tar** and **cpio** formats, both the tr tional ones and the extended formats specified in IEEE document 1003.1. It han multi-volume archives and automatically determines the format of an archive w reading it.

pax supports three user interfaces: **tar**, **cpio**, and **pax**. The **pax** interface was desig by IEEE 1003.2 as a compromise in the chronic controversy over which of **tar** or **c** interfaces is superior.

See Also

commands, cpio, tar, ustar

Notes

To avoid confusion with the traditional COHERENT **tar** command, the **tar** command distributed with **pax** is renamed **ustar**.

See the compressed **tar** archive **/usr/src/alien/pax.tar.Z** for full documentation on **pax**, **cpio**, and **ustar**.

Copyright Information

ram — Device Driver

RAM device driver

The COHERENT **ram** devices allow the user to allocate and use the random access memory (RAM) of the computer system directly. A typical use is for a RAM disk, which is a COHERENT file system kept in memory rather than on a diskette or hard disk.

The COHERENT RAM device driver has major number 8. It can be accessed either as a block-special device or as a character-special device. The high-order bit of the minor number gives a RAM device number (0 or 1), allowing the use of up to two RAM devices simultaneously. The low-order seven bits specify the device size in 64 KB (128 block) increments. The first **open** call on a RAM device with nonzero size (1 to 127) allocates memory for the device; the **open** call fails if sufficient memory is not available. Accessing a RAM device with a minor number specifying size 0 frees the allocated memory, provided all earlier **open** calls have been closed.

Initially, COHERENT includes two RAM block devices, 512KB device **/dev/ram0** (8, 8) and 192KB device **/dev/ram1** (8, 131). It also includes **/dev/ram0close** (8, 0) and **/dev/ram1close** (8, 128). The system administrator should change the RAM devices to sizes appropriate for available system memory.

Note

The COHERENT installation program **/etc/build** uses RAM device **/dev/ram1** as a RAM disk during installation. Programs **compress**, **uncompress**, **zcat** and **fsck** sometimes use **/dev/ram1** as a temporary storage device. Users should avoid using **/dev/ram1** as a RAM disk because of these programs.

Examples

The following example formats and mounts a 512-kilobyte RAM disk on directory **/fast**.

```
mkdir /fast
/etc/mkfs /dev/ram0 1024
/etc/mount /dev/ram0 /fast
```

When the RAM disk is no longer needed, its allocated memory can be freed as follows:

```
/etc/umount /dev/ram0
cat /dev/null >/dev/ram0close
```

The next example replaces the default **/dev/ram0** with a one-megabyte device taining a COHERENT file system. The new minor number 16 specifies RAM dev and size 16 times 64 kilobytes (i.e., one megabyte). The new RAM device contains blocks of 512 bytes each.

```
rm /dev/ram0
/etc/mknod /dev/ram0 b 8 16
/etc/mkfs /dev/ram0 2048
```

Files

/dev/ram*

See Also

compress, device drivers, fsck, mkfs, mount, umount, uncompress, zcat

Notes

Moving frequently used commands or files to a RAM disk can improve system p mance substantially. However, the contents of a RAM device are lost if the system power, reboots, or crashes, so changes to files kept on a RAM disk should be store quently to the hard disk or to diskette.

If a RAM device uses most but not all of available system memory, its **open** cal succeed but subsequent commands may fail because insufficient memory remains fo system.

ref – Command

Display a C function header
ref *function_name*

ref looks up the function header of a function in any of a series of reference files bu the command **ctags**. It is used by the **elvis** editor's **<shift-K>** command. This mand checks the file **refs** in the current directory.

See Also

commands, ctags, elvis

Notes

ref is a public-domain program written by Steve Kirkendall (kirkenda@cs.pdx.ed ...uunet!tektronix!psueea!eecs!kirkenda). Source code for this program is available vi Mark Williams bulletin board, USENET and other sources. It is included as a servi COHERENT users, but is not supported by Mark Williams Company. *Caveat utilito*

scat – Command

Print text files one screenful at a time
scat [[*option* ...] [*file* ...]] ...

scat prints each *file* on the standard output, one screenful (24 lines) at a time if the put is a screen. **scat** reads and prints the standard inputif no *file* is named.

The text is processed to allow convenient viewing on a screen; the options desc

below select the nature of the processing. Options begin with '-' and may be interspersed with file names.

scat scans two argument lists. The first is in the environmental **SCAT**. It should consist of arguments separated by white space (space, tab, or newline characters), with no quoting or shell metacharacters. This string is a useful place to set terminal-dependent parameters (such as page width and length) and to place invocation lists (see below). The second argument list is supplied on the command line.

scat recognizes the following options:

-l Do not stop at EOF if exactly one file was specified on the command line.

-b*n* Begin output at input line *n*.

-c Represent all control characters unambiguously. With this option, **scat** prints control characters in the range 0-037 as a character in the range 0100-0137 prefixed by a carat '^'; for example, SOH appears as "^A" and DEL as "^?" It prints mark-parity characters (in the range of 0200-0377) with '~'; for example, mark-parity 'A' and SOH appear as "~A" and "~^A", respectively. It also prefixes the characters '^', '~', and '\' with a '\'. This option overrides the option **-t**.

-cs Like **-c**, but map space ' ' to underscore '_' and prefix underscore '_' with '\'.

-ct Like **-c**, but map tabs to spaces, not "^I".

-i*n* Shift the display window right *n* columns into the text field. This is useful for viewing long lines.

-l*n* Set the display window length to *n* lines. The default is 24 normally, 34 for the Tek 4012.

-n Number input lines; wrapped lines are not numbered.

-r Remote; the output is not paged.

-s Skip empty lines.

-S*n* Seek *n* bytes into input before processing.

-t Truncate long lines. Normally, **scat** wraps each long line, with the interrupted portion delimited by a '\'.

-w*n* Set the display window width to *n* columns. The default is 80 normally, 72 for the Tek 4012.

-x Expand tabs.

-. *suffix* Invoke options by file-name suffix. If a file name ends with *.suffix*, then **scat** scans the argument sublist starting immediately after the invocation flag. New options will apply to the invoking file only. A sublist is terminated by the end of the argument list, by a file name, by the "--" flag, or by another "-." (invocation lists do not nest).

-- Terminate a sublist (see previous option).

Numbers may begin with **0** to indicate octal, and may end in **b** or **k** to be scaled by 5
or 1,024, respectively.

If the output is being paged, **scat** waits for a user response, which may be one of t
following:

newline	Display next page
/	Display next half-page
space	Display next line
f	Print current file name and line number
n	**scat** next file
q	Quit

Example

The following shows how to use the environment argument list, invocation lists, and st
lists:

```
SCAT="-124 -.c -n -.s -b5"
export SCAT
scat *.c *.s
```

After processing the **SCAT** argument list, **scat** processes the command line argume
list "***.c *.s**" with the page length at 24 lines. If a file is a C source ("***.c**") the invo
option in the **SCAT** argument list numbers the output lines. If a file is an assem
source ("***.s**") **scat** skips the first four lines.

See Also

cat, commands, pr

sched.h – Header File

Define constants used with scheduling
#define <sys/sched.h>

sched.h defines constants and structures that are used by routines that perfo
scheduling.

See Also

header files

SCSI – Device Driver

SCSI device drivers

The COHERENT SCSI series of device drivers lets you use SCSI-interface devices
tached to host adapters from several vendors.

All COHERENT SCSI device drivers use major number 13, thus allowing all SC
devices to be accessed via standard device-naming conventions. Peripherals can be
cessed as either block- or character-special devices. The minor number specifies t
device and partition number for disk-type devices; this allows the use of up to eight SC
identifiers (SCSI-ID's), with up to four logical unit numbers (LUNs) per SCSI-ID and
to four partitions per LUN. Tape and other special devices decode the minor number
perform special operations such as "rewind on close" or "no rewind on close".

The first **open** call on a SCSI disk device allocates memory for the partition table and reads it into memory.

See the release notes for further information regarding supported host adapters and peripherals.

Files

/dev/sd* — block-special devices
/dev/rsd* — character-special devices

See Also

aha154x, device drivers, drvld

Notes

The Mark Williams Company's bulletin board makes available loadable device drivers for various SCSI host adapters, as well as device driver updates. See the release notes for further information.

security — Technical Information

Because COHERENT is a multi-user, multi-tasking operating system which can support users from remote terminals, steps must be taken to ensure that the system is secure. Sensitive information that is stored on the system must be protected from being read or copied by unauthorized persons; files must be protected against vandalization by intruders. Unless a reasonable degree can be guaranteed, no multi-user operating system can be trusted to archive important information.

In one sense, it is easy to achieve perfect security in a computer system. As Grampp and Morris have noted, "It is easy to run a secure computer system. You merely disconnect all dial-up connections, put the machine and its terminals in a shielded room, and post a guard at the door." For practical uses, however, security means balancing ease of access against restrictiveness: users should have easy access to what is properly theirs, and should be barred from system facilities that do not belong to them.

The COHERENT system has the following tools to assist with security.

Passwords Every user account can be "locked" with a password. Each user can assign her own password, and the system administrator can set passwords for the superusers **root** and **bin**.

Passwords should be changed frequently. A password should have at least six characters, should *not* be a common name or word, and preferably should include a mixture of upper- and lower-case letters, to prevent decryption by brute-force methods.

Passwords should be guarded jealously. In particular, the password for the superuser **root** should be kept secret, as she can read every file and execute every program throughout the system.

Permissions Execution of system-level programs, such as **mount**, is restricted to the superuser **root**. This prevents intruders from seizing superuser permissions through unauthorized manipulation of system services. Ordinary users are also restricted from directly access system devices, for the same

reason.

Encryption The command **crypt** performs rotary encryption, similar to that used the German Enigma machine. Files of sensitive information should encrypted, to protect them against being read by unauthorized pers Note that encryption is the only true defense against unauthori reading: not even the superuser can read an encrypted file unless has the encryption key.

Many COHERENT systems have only one user and are not networked; for such stallations, the normal level of security may be an annoyance. Passwords can be tur off by using the command **passwd** to set the password to **<return>**. The comm **chmod** can be used to widen access to devices and system-level utilities; see the Lexi entry for **chmod** for more information on file access.

Security ultimately is a system-wide responsibility. To quote Grampp and Morris, far, the greatest security hazard for a system ... is the set of people who use it. If people who use a machine are naive about security issues, the machine will be vul able regardless of what is done by the local management. This applies particularly to system's administrators, but ordinary users should also take heed."

See Also

chmod, crypt, passwd, technical information

Grampp FT, Morris RH: UNIX operating system security. *AT&T Bell Lab Tec* 1984;8:1649-1672.

semctl() — COHERENT System Call

Control semaphore operations

```
#include <sys/sem.h>
semctl(semid, semnum, cmd, arg)
int semid, cmd, semnum;
union semun {
        int val;
        struct semid_ds *buf;
        unsigned short array[];
} arg;
```

semctl controls a variety of semaphore operations. *cmd* sets the operation to be formed; the following *cmds* are executed with respect to the semaphore specified *semid* and *semnum*:

GETVAL Return the value of **semval** (READ).

SETVAL Set the value of **semval** to *arg.val* (ALTER).

GETPID Return the value of **sempid** (READ).

GETNCNT Return the value of **semncnt** (READ).

GETZCNT Return the value of **semzcnt** (READ).

The following *cmds* return and set, respectively, every semval in the set of semaphore

GETALL Place semvals into array pointed to by *arg.array* (READ).

SETALL Set *semvals* according to the array pointed to by *arg.array* (ALTER).

The following *cmds* are also available:

IPC_STAT Place the current value of each member of the data structure associated with *semid* into the structure pointed to by *arg.buf* (READ).

IPC_SET Set the value of the following members of the data structure associated with *semid* to the corresponding value found in the structure pointed to by *arg.buf*:

```
sem_perm.uid
sem_perm.gid
sem_perm.mode /* only low 9 bits */
```

This command can only be executed by a process that has an effective user identifier equal to either that of superuser or to the value of **sem_perm.uid** in the data structure associated with *semid*.

IPC_RMID Remove the system identifier specified by *semid* from the system and destroy the set of semaphores and data structure associated with it. This *cmd* can only be executed by a process that has an effective user identifier equal to either that of super user or to the value of **sem_perm.uid** in the data structure associated with *semid*.

semctl will fail if one or more of the following are true:

- *semid* is not a valid semaphore identifier [EINVAL].
- *semnum* is less than zero or greater than **sem_nsems** [EINVAL].
- *cmd* is not a valid command [EINVAL].
- Operation permission is denied to the calling process. [EACCES]
- *cmd* is **SETVAL** or **SETALL** and the value to which semval is to be set is greater than the system imposed maximum [ERANGE].
- *cmd* is equal to **IPC_RMID** or **IPC_SET** and the effective user identifier of the calling process is not equal to that of superuser and it is not equal to the value of **sem_perm.uid** in the data structure associated with *semid* [EPERM].
- *arg.buf* points to an illegal address [EFAULT].

Return Value

Upon successful completion, the value returned depends on *cmd* as follows:

GETVAL	The value of **semval**.
GETPID	The value of **sempid**.
GETNCNT	The value of **semncnt**.
GETZCNT	The value of **semzcnt**.
All others	Zero

Otherwise, **semctl** returns -1 and sets **errno** to an appropriate value.

Files

/usr/include/sys/ipc.h
/usr/include/sys/sem.h
/dev/sem
/drv/sem

See Also

COHERENT system calls, sem, semget(), semop()

Notes

To improve portability, the COHERENT system implements the semaphore function
a device driver rather than as an actual system call.

sleep() – General Function

Suspend execution for interval
sleep(*seconds***)**
unsigned *seconds*;

sleep suspends execution for *seconds*.

Example

The following example, called **godot.c**, demonstrates how to use **sleep**.

```
main()
{
     printf("Waiting for Godot ...\n");

     for ( ; ; ) {
          /* sleep for five seconds */
          sleep(5);
          printf("... still waiting ...\n");
     }
}
```

See Also

general functions

strcat() – String Function (libc)

Concatenate strings
#include <string.h>
char *strcat(*string1***,** *string2***) char *****string1***, ****string2***;

strcat appends all characters in *string2* onto the end of *string1*. It returns the modi
string1.

Example

For an example of this function, see the entry for **string functions**.

See Also

string functions, string.h, strncat()

Notes

string1 must point to enough space to hold itself and *string2*; otherwise, another portion of the program may be overwritten.

sync – COHERENT System Call (libc)

Flush system buffers
sync()

sync() is the COHERENT system call that copies the contents of all memory buffers to disk.

See Also

COHERENT system calls

tgoto() – Terminal-Independent Operation

Read/interpret termcap cursor-addressing string
char *tgoto(*cm, destcol, destline***)**
char ****cm*; int** ***destcol, destline*;

tgoto is one of a set of funtions that permit COHERENT to perform terminal-independent operations. It decodes a cursor-addressing string from the *cm* **termcap** feature, and writes it into *destcolumn* in *destline*. **tgoto** uses the external variables *UP* (from the **up** feature) and *BC* (if **bc** is given rather than **bs**) if it is necessary to avoid placing **\n**, **<ctrl-D>**, or **<ctrl-@>** into the returned string. Programs calling **tgoto** should turn off the **XTABS** bits, as **tgoto** may write a tab. If a '%' sequence is given that is not understood, **tgoto** returns "OOPS".

Files

/etc/termcap — Terminal capabilities data base
/usr/lib/libterm.a — Function library

See Also

termcap, terminal-independent operation

umount() – COHERENT System Call (libc)

Unmount a file system
umount(*filesystem***)**
char ****filesystem*;**

umount is the COHERENT system call that unmounts a file system. *filesystem* names the block-special file through which the file system is accessed. Note that this must have been previously mounted by a call to **mount**, or the call will fail.

See Also

COHERENT system calls, mount()

ustar – Command

Tape archive utility

ustar is an archiving utility that reads and writes files in the format specified by the **Archive/Interchange File Format** specified in IEEE document 1003.1-1988.

See the compressed **tar** archive **/usr/src/alien/pax.tar.Z** for full documentation on **tar**.

See Also

commands, cpio, pax, tar

Copyright Information

ustar was developed by Mark H. Colburn and sponsored by The USENIX Associa
ustar is provided in binary form per the licensing terms set forth by the author. See **/usr/src/alien/pax.tar.Z** for licensing terms.

uncompress – Command

Uncompress a compressed file
uncompress [**-w** *tempfile*] [*file* ...]

uncompress uncompressses one or more *files* that had been compressed by the c mand **compress**.

Each *file*'s name must have the suffix **.Z**, which was appended onto it by **compr** otherwise, **uncompress** prints an error message and exits. When **uncompress** has compressed a *file*, it removes the **.Z** suffix from that file's name.

If no *file* is specified on the command line, **uncompress** uncompresses matter read f the standard input, and writes its output to the standard output.

Older versions of **uncompress** could only uncompress files that had been compre with option **-b12** or lower, with **-b12** being the default. The edition of **uncompr** released with COHERENT version 3.1 now handles values up to 16 by using R device **/dev/ram1** for temporary storage. For this reason, it is strongly advised that not use **/dev/ram1** as a RAM disk.

The **-w** option allows the user to specify an alternate temporary storage file to **compress**. The default value for *tempfile* when the **-w** option is omitted is **/dev/ram**

See Also

commands, compress, ram, zcat

uucico – Command

Transmit data to or from a remote site
/usr/lib/uucp/uucico [**-r1**] [**-s***site*] [**-S***site*]

uucico is the UUCP command that actually transfers files to or from a remote *site* syntax is as follows:

-r1 Poll *site* unconditionally.

-s*site*
The name of the *site* to be polled. *site* must name one of the entrie **/usr/lib/uucp/L.sys.**

-S*site*

The name of the *site* to be polled. *site* must name one of the entries in **/usr/lib/uucp/L.sys**. Unlike the **-s** option, force execution even if not the correct time.

The messages sent by **uucico** are differentiated by the first letter of the message.

Example

To poll the site *sys* at five minutes after the hour, each hour, put the following entry into **/usr/lib/crontab**:

```
05 * * * * /usr/lib/uucp/uucico -ssys -rl
```

Files

/usr/lib/uucp/L.sys — List of reachable systems
/usr/spool/uucp/.Log/uucico/*sitename* — **uucico** activities log file for *sitename*
/usr/spool/uucp/*sitename* — Spool directory for work

See Also

commands, cron, uucp, UUCP, uulog, uutouch, uuxqt

UUCP – Overview

Unattended communication with remote systems

UUCP stands for "UNIX to UNIX copy". It is a system of commands that allows you to exchange files with other COHERENT or UNIX systems, in an unattended manner. With UUCP, you can send mail to other systems, upload files, and execute commands. When configured correctly, UUCP also lets other users upload files to your system, copy files from it, and execute commands. All this can be done without your having to sit at your console and type commands; thus, files can be transferred in the small hours, when telephone rates are lower and computers are relative free.

UUCP gives you access to the Usenet, a nation-wide network of UNIX and COHERENT users. Access to the Usenet will let you exchange mail with any of the thousands of Usenet users, receive mail from them, download source code for many useful programs, and read the latest news on a host of subject.

See Also

commands, uucico, uucp, uudecode, uuencode, uuinstall, uulog, uumvlog, uuname, uutouch, uuxqt
UUCP, Remote Communications Utility, tutorial

Notes

The Lexicon entry for **sh** contains a sample shell script that logs UUCP information into a file of your choice.

uucp – Command

Ready files for transmission to other systems
uucp [-bcCdm] *source1 ... sourceN dest*

uucp copies files *source1* through *sourceN* to the destination system *dest*. Either source or destination files can contain specifications for the remote system.

uucp recognizes the following options:

-c Instead of copying the *source* file to the spool directory, use the file itself. This i default.

-C Copy the source file to the spool directory.

-d Make directories on *dest* if they are necessary for copying the files.

-f Do not make intermediate directories for the file copy.

-g*grade*
grade is a single ASCII character indicating the importance of the files being tra mitted: the lower the value of *grade*, the more important the files.

-m Send mail to the requester when the file is sent.

-n*user*
Notify *user* on destination system that file was sent. Note that *user* may contai path:

```
-nuser!site
```

-x*debug debug* is a single-digit number, 0 to 9. The higher the level, the more inforr tion yielded.

Examples

The first example copies file **foo** to directory **/bar** on system **george**:

```
uucp foo george!/bar
```

The next example copies file **/foo** from system **george** into directory **/tmp** on your s tem:

```
uucp george!/foo /tmp
```

The next example copies file **/foo** from system **george** into file or directory **/bar** on s tem **ivan**:

```
uucp george!/foo ivan!/bar
```

Note that this assumes your system can talk to both **george** and **ivan** and that your s tem has permission to read file **/foo** on system **george** as well as to write file **/bar** system **ivan**.

The next example downloads files **/foo** and **/bar** from remote systems **ivan** and **geo** into directory **/tmp** on your system:

```
uucp ivan!/foo george!/bar /tmp
```

Files

/usr/lib/uucp/L.sys — List of reachable systems
/usr/lib/uucp/Permissions — List of system permissions
/usr/spool/uucp/.Log/*/*sitename* — **uucp** activities log files for *sitename*
/usr/spool/uucp/*sitename* — Spool directory for work

See Also

commands, mail, uucico, UUCP, uudecode, uuencode, uutouch, uuwatch, uuxqt

uudecode – Command

Decode a binary file sent from a remote system
uudecode [*file*]

uudecode takes a **file** encoded by **uuencode** and translates it back to binary. Any leading and trailing lines added by **uucp** are discarded.

If the *file* is not specified, standard input is read.

Example

Consider the file **tmp** consisting of:

```
begin 644 sys
M5&AE('%U:6-K(&)R;W=N(&9O>"!J=6UP<R!O=F5R('1H92!L87IY(&1O9RX*

end
```

Note that the third line is a space followed by a newline. To decode it, type:

```
uudecode tmp
```

The output contained in file **sys** will be:

The quick brown fox jumps over the lazy dog.

See Also

commands, uucp, UUCP, uuencode

Notes

The user on the remote system must be able to write the file.

uuencode – Command

Encode a binary file for transmission to a remote system
uuencode [*source*] *outputfile*

uuencode prepares a binary file for transmission to a remote destination via **uucp. uuencode** takes binary input and produces an encoded version, consisting of printable ASCII characters, on standard output, which may be redirected or piped to **uucp**. If *source* is not specified, the standard input is read.

The format of the encoded file is as follows:

1. A *header* line starting with the characters **begin** followed by a space. This is followed by the mode of the file in octal (see **chmod** for details) and the name of the output file specified on the command line. These last two fields are also separated by a space. The mode and the system name can be changed by directing the output into a file and editing it.

2. The *body* of the file, consisting of a number of lines, each no more than 62 characters long, including a newline character. Each line starts with a character count written as a single ASCII character, representing an integer value from 0 (octal 40) to 63 (octal 135) giving the number of characters in the rest of the line. This is

followed by the encoded characters and a newline. The last line of the body is consisting of an ASCII space (octal 40).

3. The trailer line has just the characters **end** on a line by itself.

The encoding is done by taking three bytes and storing them in four characters, si per character.

Example

To encode the file **tmp** consisting of the line

```
The quick brown fox jumps over the lazy dog.
```

to be sent to the remote system **george**, enter:

```
uuencode tmp sys
```

The output will be:

```
begin 644 sys
M5&AE('%U:6-K(&)R;W=N(&90>"!J=6UP<R!O=F5R('1H92!L87IY(&109

end
```

Note that the third line consists of a space followed by a newline.

See Also

commands, uucp, UUCP, uudecode

Notes

The file is expanded by more than one third, causing increased transmission time. can be a factor when sending large files.

uuinstall – Command

Install UUCP

uuinstall

uuinstall assists with the installation of UUCP. It uses screen templates, help lines prompts to help walk you through the installation of devices, remote systems, site n domains, and permissions. For a detailed description of its use, see the tutori UUCP in the front of this manual.

See Also

commands, UUCP

Notes

Only the superuser **root** can execute **uuinstall**.

uulog – Command

Examine UUCP operations

uulog [-fx] [*system* **]**

uulog copies the last part of the file **/usr/spool/uucp/.Log/uucico/***system* to see **uucico** has done recently. *system* names the remote system whose logfile will b amined. If it is not specified, logfiles for all systems are displayed.

uulog recognizes the following options:

-f Similar to the command **tail -f**: this forces **uulog** to display UUCP activity as it i written into the log file, until you interrupt it by typing **<ctrl-C>**.

-x Display the log files for the command **uuxqt** rather than **uucico**.

Files

/usr/spool/uucp/.Log/uucico/*system* — **uucico** log file for *system*
/usr/spool/uucp/.Log/uuxqt/*system* — **uuxqt** log file for *system*

See Also

commands, uucico, uucp, UUCP, uuxqt

uumvlog – Command

Examine UUCP operations
uumvlog *days*

uumvlog copies all UUCP log files into backup files, named for their respective com mands and the date upon which the backup was performed. *days* gives the number c days for which backup files should be kept: if a backup file is more than *days* days old then **uumvlog** will delete it.

This command should be run by **cron**, because the UUCP log files can threaten to ex haust available file space on a small system unless they are chopped back daily. Fo directions on how to do this, see the tutorial for UUCP or the Lexicon entry for **cron**.

Files

/usr/spool/uucp/.Log/*command/system* — UUCP log files

See Also

commands, crontab, uucico, uucp, UUCP, uuxqt

uuname – Command

List uucp names of known systems
uuname [-l]

uuname lists the names of all systems reachable directly by **uucp**. When used with th **-l** option, it prints the name of the local system.

Files

/usr/lib/uucp/L.sys — Site and remote login data list

See Also

commands, uucico, uucp, UUCP, uulog

uutouch – Command

Touch a file to trigger uucico poll
uutouch *system*

uutouch creates an empty control file for *system* in the directory **/usr/spool/uucp/***sys tem*. This forces UUCP to poll *system* when **uucico** is called with the option **-sany**.

the empty file for *system* aready exists, it is left alone.

There are three types of files in the spool directory **/usr/spool/uucp/***system*:

C. Command file.

D. Data file.

X. Execute file.

Example

A typical usage is to put the following line into **/usr/lib/crontab:**

```
0 7 * * * /usr/lib/uucp/uutouch george
```

This forces UUCP to schedule a poll to the remote system **george** at 7 AM local tin
The actual poll take place when **uucico** is started.

Files

/usr/spool/uucp/*sitename* — Directory for uucp work files

See Also

commands, cron, uucico, uucp, UUCP, uuxqt

uuxqt – Command

Execute commands requested by a remote system
uuxqt

uuxqt takes the execute files, those marked with the prefix **X.** in the directo
/usr/spool/uucp/*sitename*, and executes them. It will only execute programs for whi
the remote system has permission.

uuxqt may be called by either **uucp** or **uucico.** It is not generally considered a user-ca
able program.

Files

/usr/spool/uucp/*sitename* — Directory for execute files

See Also

commands, uucico, uucp, UUCP

vi – Command

Clone of UNIX-standard screen editor
vi [*flags* **] [** *+cmd* **] [** *file1 ... file27* **]**

vi is a link to the editor **elvis**, which is a clone of the UNIX editors **ex** and **vi.** F
details on how to run **vi**, see the entry for **elvis** in the Lexicon.

See Also

commands, ed, ex, elvis, me, view

Notes

elvis is a public-domain program written by Steve Kirkendall (kirkenda@cs.pdx.edu
...uunet!tektronix!psueea!eecs!kirkenda), assisted by numerous volunteers. Source co
for it is available through the Mark Williams bulletin board, USENET and numero

other outlets.

elvis is distributed as a service to COHERENT customers, as is. It is not supported by Mark Williams Company. *Caveat utilitor.*

view – Command

Screen-oriented viewing utility
view *file1 ... file27*

view is a link to **elvis**, which is a clone of the UNIX **vi/ex** set of editors. Invoking **elvis** through this link forces it to operate solely in read-only mode, justas the UNIX **view** utility operates.

For information on how to use this version of **view**, see the Lexicon page for **elvis**.

See Also

commands, ed, elvis, ex, me, vi

Notes

elvis is a public-domain program written by Steve Kirkendall (kirkenda@cs.pdx.edu or ...uunet!tektronix!psueea!eecs!kirkenda), assisted by numerous volunteers. Source code for it is available through the Mark Williams bulletin board, USENET and numerous other outlets.

elvis is distributed as a service to COHERENT customers, as is. It is not supported by Mark Williams Company. *Caveat utilitor.*

virec – Command

Recover the modified version of a file after a crash
virec [**-d** *tmpdir*] *textfilename...*
virec </tmp/elv*XXX*

virec extracts the most recent version of a text file from a temporary file in **/tmp**.

When you edit a file with **elvis**, only about five kilobytes of the file are stored in RAM; the rest is stored in a file in **/tmp**. **virec** extracts the "undo" version from the file stored in **/tmp**. This is most useful when the system (or **elvis**) crashes in the middle of a long editing session, since the "undo" version of the file contains everything except your last change.

There are two ways to use **virec**. The first, and most common, way to invoke **virec** is to give it the name of the file you were editing; it finds the matching file in **/tmp** and writes the newest available version of the file over the existing version. It then deletes the **/tmp** file.

The second way is to use the '<' to let **virec** read a particular **/tmp** file via **stdin**. Use this method when you either have forgotten which file you were editing and want to see its contents, or when you wish to recover a file without losing either the **/tmp** file or the current version of the text file.

The **-d** option tells **virec** to look for a temporary file in *directory* rather than in **/tmp**.

Files

/tmp/elv* — Temporary file created by **elvis**

See Also

commands, elvis

Notes

virec is a public-domain program written by Steve Kirkendall (kirkenda@cs.pdx.ed ...uunet!tektronix!psueea!eecs!kirkenda). Source code for this program is available vi Mark Williams bulletin board, USENET and other sources. Please note that program is distributed as a service to COHERENT users, but it is not supporte Mark Williams Company. *Caveat utilitor.*

zcat — Command

Concatenate a compressed file

zcat [*file* ...]

zcat concatenates one or more *files* that had been compressed with the comm **compress**. It uncompresses each *file* "on the fly," and prints the uncompressed text the standard output.

If no *file* is specified on the command, **zcat** uncompresses matter read from the stan input.

Older versions of **zcat** could only uncompress files that had been compressed with o **-b12** or lower, with **-b12** being the default if the option was omitted. This relea **zcat** now handles values up to **-b16** by using RAM device **/dev/ram1** for tempo storage. For this reason, it is strongly advised that you not use **/dev/ram1** as a disk.

See Also

commands, compress, ram, uncompress

Section 4:
Errata

Known Bugs and Limitations

1. On some systems, when using DOS 4.01, any operations from DOS that write to th file allocation table (FAT) of the DOS partition cause the COHERENT master boot t stop working.

2. In shell scripts, the start of a *here document* must be the last thing on a line, e.g.

```
cat > outfile << SHAR_EOF
```

works, but

```
cat << SHAR_EOF > outfile
```

does not.

3. In shell scripts, a newline may not occur inside a double-quoted string, although i may occur inside a single-quoted string. For example,

```
FOO='x
yz'
```

works, but

```
BAZ="x
yz"
```

does not.

4. Wildcards are *not* expanded in redirection arguments. If you enter

```
echo hello > junk
```

then

```
cat < junk
```

works, but

```
cat < ju*
```

does not.