hai — Device Driver

Host adapter-independent SCSI driver

hai is a host adapter-independent device driver that supports various SCSI devices. It supports the Adaptec 154x host adapter, and compatibles; and all host adapters built around the Future Domain TMC-950/9C50 chip set. With a supported host adapter, **hai** can support any mix of up to seven SCSI hard disks (either fixed or removable media), CD-ROM drives, and tape drives.

hai has major-device number 13. It can access devices either in block mode or character mode. The minor number specifies the device and partition number for disk-type devices; this allows the use of up to eight SCSI identifiers (SCSI-ID's), with up to four logical unit numbers (LUNs) per SCSI-ID and up to four partitions per LUN. Tape and other special devices decode the minor number to 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.

hai is a modular driver that you can configure to suit your system's suite of SCSI hardware. To build the driver, you must link the main **hai** module with the appropriate module for your system's SCSI host-adapter card, and a module for each type of SCSI device you have (hard disk, CD-ROM, or tape). Each of **hai** modules is described below. Usually, you will configure **hai** when you install COHERENT onto your system, but you can reconfigure **hai** at any time should you wish to add or modify your system's suite of SCSI devices. The script **/etc/conf/hai/mkdev** walks you through this process. Once you have reconfigured **hai**, run the program **/etc/conf/bin/idmkcoh** to build a new kernel; then reboot your system to invoke the newly built kernel and you're back in business.

Extending hai

hai is designed to be extendable to other host adapters and other SCSI devices. It is easy to extend **hai** to work with new hardware. It is possible to extend **hai** either to support a new host adapter, or to support new peripheral device, or both.

To adapt **hait** to a new a host adapter, you must write a handful of routines to initialize and access the host adapter. A host-adapter module must be able to do the following:

- Initialize the host adapter and ready it for future requests.
- Start a SCSI command and call a notification function when that command completes or times out.
- Abort a SCSI command in progress.
- Reset a device on the SCSI bus.
- Reset the SCSI bus.

It is easier to write a module for a peripheral device: you only need to send the appropriate SCSI commands to access the device as required by the COHERENT device-driver interface — i.e., **open()**, **close()**, **read()**, **write()**, and **ioctl**(). To do this, use the routines provided by the host-adapter module, when neccessary, to access the SCSI bus and the device.

The following sections of this article discuss each of **hai**'s constituent modules.

hai154x — Adaptec Host-Adapter Module

hai154x is the **hai** host-adapter module for the Adaptec 154x and compatible host adapters. This module lets you run any combination of SCSI hard disks, tape drives, or CD-ROM drives through any Adaptec AHA-154x host adapter.

The Adaptec AHA-154x is an intelligent ISA bus mastering SCSI host adapter. Its on-board processor and DMA controllers handle the SCSI bus protocol and the DMA transfer of SCSI data into the PC's main memory. **hai154x** uses port I/O, a DMA channel, and an interrupt line, which are configured through the following tunable variables:

HAI154X_BASE	Base port
HAI154X_INTR	Interrupt level
HAI154X_DMA	DMA channel

The following tunable parameters let you set the DMA transfer speed, the bus-on time, and the bus-off time on the SCSI bus:

HAI154X_XFERSPEED	DMA transfer speed, from the table below
HAI154X_BUSOFFTIME	Host-adapter bus-on time for DMA transfers
HAI154X_BUSONTIME	Host-adapter bus-off time for DMA transfers

Variable **HAI154X_XFERSPEED** must be set to one of the values given in the following table.

Setting	*Speed, megabytes/second*
0	5.0
1	6.7
2	8.0
3	10.0
4	5.7

The default setting is '4'.

You can use these parameters to tune the performance of the SCSI bus for your system. For most installations, the default settings should be work well.

haiss — Seagate Host-Adapter Module

haiss is the host-adapter module for host adapters built around the Future Domain TMC-950 or TMC-9C50 chip sets. It works with the following controllers:

Seagate ST01 or ST02
Future Domain TMC-845, 850, 860, 875, or 885
Future Domain TMC-840, 841, 880, or 881

Through this host-adapter module, you can run any combination of SCSI hard disks, tape drives, or CD-ROM drives through any of the above host adapters.

These host adapters map the SCSI bus data and signal lines onto memory addresses on the PC bus. **haiss** then uses standard memory-read and -write operations to access the state of the SCSI bus and the data on it. By default, this controller uses the Intel block-move instruction to transfer data between the device's buffer and the SCSI data-address range. This mode of transfer may be too fast for certain SCSI devices, in which case data can be transferred byte by byte. You can set how **haiss** transfers data; this is described below.

haiss can be used through the following tunable kernel variables:

HAISS_TYPE
The type of the card, as follows:

Type	*Controller*
0	Seagate ST01/02
1	Future Domain TMC-845/850/860/875/885
2	Future Domain TMC-840/841/880/881

HAISS_INTR
The interrupt vector to which the card is set. Although MS-DOS permits you to use this card without interrupts, COHERENT requires that you use interrupts.

HAISS_BASE
The real-mode segment address for the start of the card's RAM area. On all Future Domain and Seagate host adapters with an eight-kilobyte ROM, this is also the base address that is jumpered onto the card. On Seagate host adapters with a 16-kilobyte ROM, this is the base address jumpered on the card plus 0x0200.

HAISS_SLOWDEV
A bitmask of the target identifiers of all SCSI devices whose rate of data transfer is slower than the default transfer mode that the host adapter supports.

These variables are set automatically by the script **/etc/conf/hai/mkdev**, when you use it to configure **hai** for your system; or you can use the command **/etc/conf/bin/idtune** to tune them individually.

haict — Tape Device Module

haict is the device module that controls SCSI tape drives. It works a number of popular quarter-inch and DAT SCSI tape drives

SCSI tape-drive configuration is controlled by the tunable variables **HAI_TAPE_SPEC** and **HAICT_CACHE**.

HAI_TAPE_SPEC is a bitmap of the SCSI identifiers that identify tape drives on your system. For example, if a system has only one SCSI tape drive, and it is assigned SCSI identifier two, then you would set **HAI_TAPE_SPEC** to 0x04, which flips on bit two of that mask. (If you are versed in converting binary values into bit masks, note that the script **/etc/conf/hai/mkdev** handles that conversion for you — all you have to do is tell it what SCSI identifiers are set to which devices, and it does the rest.)

Variable **HAICT_CACHE** sets the size of block of memory that **hai** uses to buffer data that it writes to or reads from the tape drive. You can set this anywhere from zero to 256 kilobytes. The default is 16 kilobytes, which should works well with most tape drives. To tune this variable, use either the command **/etc/conf/bin/idtune** or the script **/etc/conf/hai/mkdev**. Please note that larger tape caches may not necessarily improve tape performance. For example, the program **cpio** for example uses a 5,120-byte buffer that limits the effectiveness of any tape-buffering scheme.

haicd — CD-ROM Device Module

haicd is the device module that controls SCSI CD-ROM drives. It permits you to read data from both audio CDs and CD-ROM that hold an ISO 9660 file system.

Configuration of **haicd** is controlled by the variable **HAI_CDROM_SPEC**, which is a bitmap of the SCSI identifiers that identify CD-ROM drives on your system. For example, if a system has only one SCSI CD-ROM drive, and it is assigned SCSI identifier three, then you would set **HAI_TAPE_SPEC** to 0x08, which flips on bit three of that mask. (If you are versed in converting binary values into bit masks, note that the script **/etc/conf/hai/mkdev** handles that conversion for you — all you have to do is tell it what SCSI identifiers are set to which devices, and it does the rest.)

As of this writing (September 1994), **haicd** has been tested with SCSI CD-ROM drives from Toshiba and NEC. The CD-ROM functions work with both makes of CD-ROM. Please note, however, that the audio functions of the NEC CDR-74 and CDR-84 CD-ROM drives deviate from the SCSI-2 specification considerably; therefore, the audio functions of **haicd** do not work on these drives.

haisd — Hard Disk Device Module

haisd is the **hai** device module that controls SCSI disk drives. Because **hai** allows multiple, overlapping, simultaneous access to the system's SCSI host adapter, the disk drives that **hai** controls operate independently of each other. **haisd** also chains "like" requests for multiple contiguous sectors, which reduces the overhead of starting SCSI commands and thereby inproves performance.

haisd is configured through the tunable kernel variable **HAI_DISK_SPEC**, which is a bitmap of the SCSI identifiers that identify hard-disk drives on your system. For example, if a system has two SCSI disk drives, one with SCSI identifier zero and the other with SCSI identifier one, **HAI_DISK_SPEC** to 0x03, which flips on bits 0 and 1 of that mask. (If you are versed in converting binary values into bit masks, note that the script **/etc/conf/hai/mkdev** handles that conversion for you — all you have to do is tell it what SCSI identifiers are set to which devices, and it does the rest.)

haisd determines partitioning information from the device's minor number as follows:

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

The 'S' field is the "special" bit: it distinguishes SCSI disk drives from tape drives. The 'P' fields are a binary value of the partition-table entry for this device, from 0 through 3. If the special bit is set and the partition fields are not 0, then **haisd** assumes that this device is *not* a disk drive and will not allow access to the device. The 'I' fields give the binary value of the SCSI identifier for this device, from zero through seven. This convention is used for all SCSI devices. Finally, the 'L' fields set the logical-unit number field, from 0 through 3. (If you are not skilled at setting bit maps by hand, do not despair: the configuration script **/etc/conf/hai/mkdev** automatically builds an appropriate device node for each SCSI disk.)

Files

/dev/sd* — Block-special SCSI-disk devices
/dev/Stp* — Block-special SCSI-tape devices
/dev/Scdrom* — Block-special SCSI CD-ROM devices
/dev/rsd* — Character-special SCSI-disk devices
/dev/rStp* — Character-special SCSI-tape devices
/dev/rScdrom* — Character-special SCSI CD-ROM devices

See Also

CD-ROM, device drivers, hai154x, haiss, haicd, haict, haisd, hard disk, tape

Notes

For a list of the block-special files via which you can access the devices that **hai** supports, see the Lexicon entries for **hard disk** and **tape**.

If you are using an Adaptec AHA-1540, AHA-1542C, or AHA-1542CF SCSI host adapter with a drive larger than one gigabyte and extended BIOS support turned on, then you must override the default number of heads to 255 and the number of sectors per track to *63*. Note that when you run the script **/etc/conf/hai/mkdev** (or install COHERENT onto your system), "255" appears as the default choice for the number of heads; however, the default choice for number of sectors is 32. Therefore, when you run **/etc/conf/hai/mkdev** or install COHERENT for a system that has one of the above-named SCSI controllers, you must select the default setting for the number of heads, but you must type "63" when asked for the number of sectors per track.

hai supercedes the older COHERENT device drivers **aha** and **ss**, which were specific to the Adaptec and Future-Domain controllers, and which controlled only SCSI disk drives.

hai was written by Chris Hilton (hilton@mwc.com).

hard disk — Technical Information

The hard disk is the primary means of storing and accessing data under the COHERENT system. This article introduces some aspects of the COHERENT system that affect the care and feeding of your hard disk.

Device Drivers

The COHERENT system comes with two drivers for hard disks: the **at** drivers, for AT-style hard disks (i.e., IDE, ESDI, MFM, or RLL disks); and **hai**, for the SCSI family of hard disks. **hai** is a host adapter-independent SCSI driver and also supports SCSI devices other than hard disks, e.g., SCSI tape. which is the old-style driver for Adaptec SCSI devices. For details on each driver, see its entry in the Lexicon.

The following describes how to enable or disable a given hard-disk driver in your kernel. To disable a hard-drive controller, log in as the superuser **root** and then execute the following commands:

```
cd /etc/conf
bin/idenable -d disk_driver
bin/idmkcoh -o /kernel_name
```

where *kernel_name* is the name you wish to give to the new kernel, and *disk_driver* is one of **at**, **aha**, **ss**, or **hai**.

To enable a hard disk, again log in as **root**; then type the following commands:

```
cd /etc/conf
bin/idenable -e disk_driver
# if you are installing the hai driver:
# hai/mkdev
bin/idmkcoh -o /kernel_name
```

where *disk_driver* is one of **at**, **aha**, **ss**, or **hai**.

Partitioning

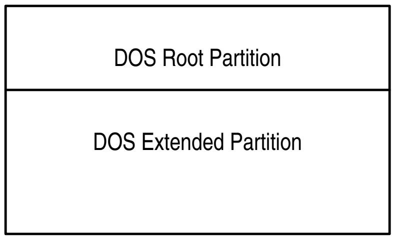
The COHERENT command **fdisk** displays information about how your hard disk is currently configured. You can also use it to repartition your hard disk and reassign partitions from MS-DOS to COHERENT, or vice versa.

This is an extremely powerful command, with which you can create much mayhem on your system. Like any powerful tool, it should be treated carefully and with respect. See the article on **fdisk** in the Lexicon for details on how to use this command.

Partitioning your hard drive can be an uncomplicated procedure. We offer these guidelines in an effort to make it as simple as possible. Before attempting any partitioning you should first back-up all the data currently on your hard drive. If you do not do this you risk losing data permanently. You should also know the correct physical parameters of your hard drive. This information can be obtained from your machine documentation or from the drive manufacturer. It is best not to rely on the parameters given in the BIOS: these may be translation parameters.

If your drive is formatted for MS-DOS, it is advisable to run MS-DOS **fdisk** before you start to install COHERENT. If the whole drive is taken up by DOS partitions, you must use MS-DOS **fdisk** to create a non-DOS area on the drive. It is not sufficient to have an empty MS-DOS logical drive set aside for COHERENT. COHERENT does not recognize MS-DOS logical drives, it only sees the whole partition. The following diagram shows the way the MS-DOS **fdisk** sees your drive:

And the following diagram shows the way the COHERENT **fdisk** sees your drive:



If you use COHERENT **fdisk** to repartition MS-DOS space, you risk causing MS-DOS **fdisk** to hang. One further word of warning. If you have an automated disk formatting and partitioning utility on your MS-DOS partition such as Disk Manager or Speedstor, you should operate it in “manual” mode, not in “automatic”.

Some hard drives have more than 1,024 cylinders. COHERENT can only recognize a drive up to this limit. You may have a utility such as Speedstor that allows you to place MS-DOS partitions beyond that boundary. COHERENT will not see those partitions, but you can still access them as usual through MS-DOS.

When partitioning a drive with more than 1,024 cylinders, be sure to run the partitioning utility before you start to install COHERENT. You should create a non-DOS partition that falls completely within the 1,022-cylinder boundary. Your next MS-DOS partition should start no sooner than the 1,026th cylinder.

Adding a COHERENT Partition

The following describes how to add a new COHERENT partition on your hard disk.

During your initial installation of COHERENT, the installation program handled the details of preparing your hard disk for COHERENT. Adding a partition after the system is installed is not difficult, but it requires that you understand the operation of the following commands: **badscan**, **chmod**, **chown**, **fdisk**, **fsck**, **mkfs**, and **mount**. See the Lexicon articles for each of these commands for further information before you attempt to add a partition.

In general, the following steps are required when creating a partition for use by COHERENT. Please note that you must not change the size of your existing root partition, or you may no longer be able to boot COHERENT from the hard disk.

1. Completely back up all partitions on your hard disk. Be sure to back up the COHERENT partitions, as well as any non-COHERENT partitions (e.g., those for MS-DOS or OS/2). Verify that your backups are *readable* and *correct.*

2. Log in as the superuser **root**. Make sure all other users are off the system; then invoke the command **/etc/shutdown**. This shuts down COHERENT and returns the system to single-user mode. Type the command **sync** to flush all buffers.

3. Invoke the COHERENT command **fdisk** and add the COHERENT partition to your disk, as described above. Be sure to write down the device name associated with your new partition (e.g., **/dev/at0c**) and its size.

4. The command **badscan** checks the device for bad blocks. If your partition resides on a non-SCSI device, run the command **badscan** as follows:

```
/etc/badscan -v -o /conf/proto.device raw_device xdevice
```

where *device* specifies the four-character block-special device name for the partition (e.g., **at0c**), *raw_device* is the full device path name for the character-special device associated with the partition (e.g., **/dev/rat0c**), and *xdevice* names the partition-table device for the disk drive (e.g., **/dev/at0x**).

5. Invoke the command **mkfs** to create a COHERENT file system on the new partition, as follows:

```
/etc/mkfs /dev/device /conf/proto.device
```

This invocation forces **mkfs** to use the contents of the "proto" file that **badscan** created when it built the *bad_block* list for the new partition.

6. If need be, use the command **mkdir** to create a directory to use as a *mount point* for the newly created file system. The mount point is the directory onto which this directory's file system will be appended. Usually, this directory is located under '/', also called the *root directory*. You can, however, mount a file system onto any directory that already exists. If you create a new directory (e.g., **/w** or **/mydir**), use the commands **chown** and **chmod** to set an appropriate ownership and mode for for the directory.

7. Edit the file **/etc/mount.all** and add a line of the following form:

```
/etc/mount device /mount_point
```

where *device* is the full path name of the device that specifies your new partition (e.g., **/dev/at0c**), and *mount_point* is the name of the directory that you created in the earlier step.

8. Finally, edit the file **/etc/checklist** and add the character special device name (e.g., **/dev/rat0c**) of the new COHERENT partition to it. This will ensure that COHERENT will automatically run **fsck** on that partition's file system whenever you boot the system. This can be vital in recovering from a system crash.

Adding Another Hard Disk

If you wish to add another hard disk to your system, you may have to run some low-level routines that are hardware specific. See the documentation that accompanies your hardware for details.

In brief, when you install the hard disk, you must partition it, as you did your original hard disk when you first installed COHERENT. If you wish to add non-COHERENT operating systems to one or more partitions, do so first; then add COHERENT to the remaining partitions, as described above.

Changing the Size of the Root Partition

Changing the size of your **root** file system requires that you reinstall COHERENT. It is strongly advised that you back up *all* partitions of your system before you attempt to do this. In addition, to reduce the time involved in restoring your data files, make an additional backup of all directories and files that have changed form your original COHERENT installation. The command **find** will help you locate all such files; see its Lexicon entry for details.

You should then follow the directions given in the release notes for installing COHERENT. Note that when you attempt to install COHERENT over an existing COHERENT partition, COHERENT will ask you if you are sure you know what you're doing before the installation procedure creates a new file system on the partition. Be sure to request that a new file system be created, or the installation will fail.

After installing the COHERENT distribution onto your new root partition, restore any data files and directories from the second set of backups that you performed.

See Also

Administering COHERENT, at, badscan, chmod, chown, fdisk, fsck, hai, ideinfo, mkfs, mount

Notes

For information on how an IDE drive is configured, use the command **ideinfo**. For details on how to use this command, see its entry in the Lexicon.

Some users have attempted to use Norton Utilities or similar tools to rearrange the partition table, only to find that COHERENT no longer boots. That is because the kernel has embedded within it the name of the partition on which it and its root file system live. By using Norton Utilities to shuffle the partition table, the kernel will no longer be able to find any of the files or utilities it needs to boot your system. If you still wish to shuffle your disk's partition table, be sure to change the name of the root device within the kernel *before* you change the partition table.

hash — Command

Add a command to the shell's hash table
hash [-r] [*command ...*]

The command **hash** lets you manipulate the Korn shell's hashing facility. A hashed command can be accessed instantly by the shell, without the delay of searching the directories in the **PATH** environmental variable.

When called with an argument, **hash** prints all hashed commands. When called with one or more *command* arguments, it adds *command* to its hash table. The option **-r** removes *command* from the hash table.

Note that before you can use hashing, you must use the **set** command to turn it on. For more information on the Korn shell's hashing feature, see the Lexicon entry for **ksh**.

See Also

commands, ksh

hdioctl.h — Header File

Control hard-disk I/O
#include <sys/hdioctl.h>

Header file **<sys/hdioctl.h>** declares constants and structures used to control hard-disk I/O.

Structure **ide_info** is used by the command **ideinfo** to hold information about IDE drives. It is defined as follows:

```
typedef struct ide_info {
       unsigned short ii_config;           /* Configuration */
       unsigned short ii_cyl;              /* Cylinders (default translation mode) */
       unsigned short ii_reserved;         /* reserved */
       unsigned short ii_heads;            /* heads (default translation mode */
       unsigned short ii_bpt;              /* bytes per track (unformatted) */
       unsigned short ii_bps;              /* bytes per sector (unformatted) */
       unsigned short ii_spt;              /* sectors per track (default translation mode) */
       unsigned short ii_vendor1[3];       /* vendor's unique data */
       unsigned short ii_serialnum[10];    /* serial number in ASCII */
       unsigned short ii_buffertype;       /* buffer type */
       unsigned short ii_buffersize;       /* buffer size in 512-byte sectors */
       unsigned short ii_eccbyteslong;     /* ecc bytes for r/w long */
       unsigned short ii_firmrev[4];       /* firmware revision in ASCII */
       unsigned short ii_modelnum[20];     /* model number in ASCII */
       unsigned short ii_doublewordio;     /* double word transfer flag */
       unsigned short ii_capabilities;     /* capabilities */
       unsigned short ii_reserved2;        /* reserved */
       unsigned short ii_piomode;          /* PIO data transfer timing mode */
       unsigned short ii_dmamode;          /* DMA data transfer timing mode */
       unsigned short ii_reserved3[75];    /* reserved */
       unsigned short ii_vendor2[32];      /* vendor unique data */
       unsigned short ii_reserved4[96]; /* reserved */
} ide_info_t;
```

Field **ii_config** is a set of flags that describes how the drive is configured, as follows:

bit 0	Not used.
bit 1	Disk is hard sectored.
bit 2	Disk is soft sectored.
bit 3	Disk is not MFM encoded.
bit 4	Disk's head switch time is less than 15 microseconds.
bit 5	Spindled motor control option is implemented.
bit 6	Fixed drive.
bit 7	Not used.
bit 8	Disk's transfer rate is less than five megabytes per second.
bit 9	Disk's transfer rate exceeds five megabytes per second but less than or equal to 10 megabytes per second.
bit 10	The disk's transfer rate exceeds ten megabytes per second.
bit 11	The rotational's speed tolerance is greater than 0.5%.
bit 12	The data strobe offset option is available.

bit 13	The track offset option is available.
bit 14	The format speed-tolerance gap required.
bit 15	Not used.

Structure **hdparm_s** holds the drive's attributes. It is configured for binary compatibility with ROM data.

```
typedef struct hdparm_s {
	unsigned char ncyl[2];		/* number of cylinders */
	unsigned char nhead;		/* number heads */
	unsigned char rwccp[2];		/* reduced write curr cyl */
	unsigned char wpcc[2];		/* write pre-compensation cyl */
	unsigned char eccl;		/* max ecc data length */
	unsigned char ctrl;		/* control byte */
	unsigned char fill2[3];
	unsigned char landc[2];		/* landing zone cylinder */
	unsigned char nspt;		/* number of sectors per track */
	unsigned char hdfill3;
} hdparm_t;
```

See Also

hard disk, header files, ideinfo

head — Command

Print the beginning of a file
head [+*n***[bcl]] [***file***]**
head [-*n***[bcl]] [***file***]**

head copies the first part of *file*, or of the standard input if none is named, to the standard output.

The given *number* tells **head** where to begin to copy the data. Numbers of the form +*number* measure the starting point from the beginning of the file; those of the form -*number* measure from the end of the file.

A specifier of blocks, characters, or lines (*b*, **c**, or **l**, respectively) may follow the number; the default is lines. If no *number* is specified, a default of +4 is assumed.

See Also

commands, dd, egrep, sed, tail

Notes

Because **head** buffers data measured from the end of the file, large counts may not work.

header files — Overview

A *header file* is a file of C code that contains definitions, declarations, and structures commonly used in a given situation. By tradition, a header file always has the suffix ".h". Header files are invoked within a C program by the command **#include**, which is read by **cpp**, the C preprocessor; for this reason, they are also called "include files".

Header files are one of the most useful tools available to a C programmer. They allow you to put into one place all of the information that the different modules of your program share. Proper use of header files will make your programs easier to maintain and to port to other environments.

COHERENT includes the following header files:

a.out.h	Include all COFF header files
acct.h	Format for process-accounting file
ar.h	Format for archive files
assert.h	Define **assert()**
sys/buf.h	Buffer header
sys/cdrom.h	Definitions for CD-ROM drives
coff.h	Format for COHERENT objects
sys/con.h	Configure device drivers
sys/core.h	Declare structure of a **core** file
ctype.h	Header file for data tests
curses.h	Declare/define **curses** routines
dbm.h	Header file for DBM routines
sys/deftty.h	Default tty settings
dirent.h	Define constant **dirent**

errno.h	Error numbers used by **errno()**
fcntl.h	Manifest constants for file-handling functions
sys/fd.h	Declare file-descriptor structure
sys/fdioctl.h	Control floppy-disk I/O
sys/fdisk.h	Fixed-disk constants and structures
sys/filsys.h	Structures and constants for super block
float.h	Define constants for floating-point numbers
fnmatch.h	Constants used with function **fnmatch()**
fperr.h	Constants used with floating-point exception codes
gdbm.h	Header file for GDBM routines
gdbmerrno.h	Define error messages used by GDBM routines
grp.h	Declare group structure
sys/hdioctl.h	Control hard-disk I/O
sys/ino.h	Constants and structures for i-nodes
sys/inode.h	Constants and structures for memory-resident i-nodes
sys/io.h	Constants and structures used by I/O
sys/ipc.h	Declarations for interprocess communication
sys/kb.h	Define keys for loadable keyboard driver
l.out.h	Format for COHERENT-286 objects
limits.h	Define numerical limits
sys/lpioctl.h	Definitions for line-printer I/O control
math.h	Declare mathematics functions
mnttab.h	Structure for mount table
mon.h	Read profile output files
sys/mount.h	Define the mount table
mprec.h	Multiple-precision arithmetic
sys/msg.h	Definitions for message facility
mtab.h	Currently mounted file systems
sys/mtioctl.h	Magnetic-tape I/O control
mtype.h	List processor code numbers
n.out.h	Define **n.out** file structure
ndbm.h	Header file for NDBM routines
netdb.h	Define structures used to describe networks
path.h	Define/declare constants and functions used with **path**
poll.h	Define structures/constants used with polling devices
sys/proc.h	Define structures/constants used with processes
sys/ptrace.h	Perform process tracing
pwd.h	Define password structure
regexp.h	Header file for regular-expression functions
sys/sched.h	Define constants used with scheduling
sys/seg.h	Definitions used with segmentation
sys/sem.h	Definitions used by semaphore facility
setjmp.h	Define **setjmp()** and **longjmp()**
sgtty.h	Definitions used to control terminal I/O
shadow.h	Definitions used with shadow passwords
sys/shm.h	Definitions used with shared memory
signal.h	Define signals
socket.h	Define constants and structures with **sockets**
sys/stat.h	Definitions and declarations used to obtain file status
stdarg.h	Declare/define routines for variable arguments
stddef.h	Declare/define standard definitions
stdio.h	Declarations and definitions for I/O
stdlib.h	Declare/define general functions
sys/stream.h	Definitions for message facility
string.h	Declare string functions
stropts.h	User-level STREAMS routines
termio.h	Definitions used with terminal input and output
termios.h	Definitions used with POSIX extended terminal interface
time.h	Give time-description structure
sys/timeb.h	Define **timeb** structure
sys/times.h	Definitions used with **times()** system call
sys/tty.h	Define flags used with tty processing

sys/types.h	Define system-specific data types
ulimit.h	Define manifest constants used by system call **ulimit()**
unctrl.h	Define macro **unctrl()**
unistd.h	Define constants for file-handling routines
sys/uproc.h	Definitions used with user processes
utime.h	Declare system call **utime()**
utmp.h	Login accounting information
sys/utsname.h	Define **utsname** structure
varargs.h.	Declare/define routines for variable arguments
sys/wait.h.	Define wait routines

Compilation Environments and Feature Tests

The COHERENT header files are designed to let you invoke any of several “compilation environments”. Each environment offers its own features; in this way, you can easily import code that conforms to the POSIX or ANSI standards, compile device drivers, or otherwise fine tune how your programs are compiled. To invoke a given compilation environment, you must set a *feature test*.

As discussed in the Lexicon article **name space**, the ISO Standard reserves for the implementation every identifier that begins with a single underscore followed by an upper-case letter. The POSIX Standards define several symbols in this name space that the implementation can use as “feature tests” — that is, as symbols that you can use in your source code to determine the presence or absence of a particular feature or combination of features. Note that a feature test applies to an *implementation* of C, rather than to an operating system. A feature test combines aspects of the host system and the language translator: some tests apply to the operating system, some purely to the C translator.

The operating system’s header files can define them (for example, **_POSIX_SAVED_IDS**) to control compilation of user code or to deal with optional features, or you can define them (e.g., **_POSIX_C_SOURCE**) to control how the system’s header files declare or define constants, types, structures, and macros.

In general, a feature test must either be undefined or have an integer value. It must not be defined as having no expansion text, or expand into a string. For example,

```
# CORRECT
cc -D_POSIX_C_SOURCE=1 foo.c
```

is correct, as is:

```
# CORRECT
cc -U_POSIX_C_SOURCE foo.c
```

However,

```
# WRONG
cc -D_POSIX_C_SOURCE foo.c
```

is incorrect, as is:

```
# WRONG
cc -D_POSIX_C_SOURCE="yes" foo.c
```

This is to permit the constants to be tested with expressions like

```
#if _POSIX_C_SOURCE > 1
```

where an integer value is required. (If the symbol is used in a **#if** test and is undefined, **cpp** replaces it with zero, which is still an integer value). This permits the implementation to use different values of the feature test to invoke different feature sets; and it simplifies testing for complex combinations of feature tests.

Although nearly all feature tests behave as shown above, there are a few exceptions, namely **_POSIX_SOURCE** and **_KERNEL**. These symbols are not defined as having a specific value, so many users do not supply a value. To deal with this, the COHERENT header files check whether these constants have expansion text. If they do not, the header files redefine these constants with value 1, so that they can be used like the other feature tests that the COHERENT header files define.

The following describes the feature tests used in the COHERENT header files, and briefly describes the compilation environment each invokes. Because we are continually adding new features to the kernel, this list is not guaranteed to be complete.

_DDI_DKI
Invoke the environment for compiling device drivers. This environment makes visible all DDI/DKI function prototypes and data definitions, and defines all fundamental data types and structures as mandated by UNIX System V, Release 4.

Please note that this feature test is an COHERENT extension, and is not portable to other operating systems.

_KERNEL
Invoke the environment for compiling the kernel or a device driver. This environment gives code full access to system's private header files. Under COHERENT, this option is equivalent to defining **_DDI_DKI** to value 1, because COHERENT only supports compiling DDI/DKI driver source code from System V, Release 4. This means that the definitions of many fundamental data types such as **pid_t** are changed to the System V, Release 4 definitions rather than the System V, Release 3 definitions used by user code. (This is a System V convention.)

_POSIX_SOURCE
_POSIX_C_SOURCE
Select a "clean" compilation environment, in which the headers defined in the **POSIX.1** or **POSIX.2** standards define no symbols other than the ones that those environments require. Defining **_POSIX_C_SOURCE** with value 1 selects the **POSIX.1** environment, as defined in the POSIX.1 standard. Defining **_POSIX_C_SOURCE** with value 2 selects the **POSIX.2** environment, as defined in the POSIX.2 standard. Defining **_POSIX_SOURCE** has the same effect as defining **_POSIX_C_SOURCE** with value 1.

_STDC_SOURCE
Select a "clean" compilation environment. In this environment, the headers that the ANSI C standard defines define no symbols other than those that the standard requires. This feature test is designed to let you compile conforming Standard C programs that themselves define functions or macros that the COHERENT header files defined in addition to those described in the ANSI standard.

Please note that this feature test is an COHERENT extension, and is not portable to other operating systems.

_SUPPRESS_BSD_DEFINITIONS
This feature test invokes a compilation environment that excludes all definitions that are included for compatibility with Berkeley UNIX. As of this writing, this feature test affects only the header file **<string.h>**, and prevents it from defining the macros **bcopy()**, **bzero()**, **index()**, and **rindex()**. Note that selecting a POSIX or Standard C environment also suppresses these definitions.

Please note that this feature test is an COHERENT extension, and is not portable to other operating systems.

_SYSV3
This feature test invokes a compilation environment in which all fundamental types and data structures have the definitions mandated by UNIX System V, Release 3.

_SYSV4
This feature test invokes a compilation environment in which all fundamental types and data structures have the definitions mandated by UNIX System V, Release 4. As of this writing, this facility is incomplete and used mainly to develop device drivers and extensions to the kernel.

Please note that this feature test is an COHERENT extension, and is not portable to other operating systems.

See Also

#include, C language, cpp, portability

help — Command

Print concise description of command
help [-d*c***] [-f***file***] [-i***file***] [-r] [***command***]...**

help prints a concise description of the options available for each specified *command*. If *command* is omitted, **help** prints a simple description of itself, followed by information about the command given by **$LASTERROR**, which is the last command returning a nonzero exit status.

help provides more information than the usage message printed by a command, but less than the detailed description given by the **man** command. The primary purpose of **help** is to refresh your memory if you have forgotten an option to *command*.

help looks in **/usr/lib/helpfile** for system information and the file named in environmental variable **$HELP** for user-specific information. Information about a *command* begins with a line

> #*command*

and ends with the next line beginning with '#' in **/usr/lib/helpfile** or **$HELP.**

help recognizes the following options:

-d*c* Use *c* as the delimiting character within the helpfile, instead of the default **#**.

-f*file* Read the help entries from *file* instead from the default, **/usr/lib/helpfile**.

-i*file* Read the helpfile's index from *file* instead of from the default, **/usr/lib/helpindex**. **help** uses the index to speed its retrieval of an entry, and does not work without it.

-r Rebuild the index. If you modify a helpfile, you must rebuild its index, or **help** will no longer retrieve items correctly.

Example

The following shows how to rebuild the index for helpfile **myhelp**, using **@** as the delimiting character:

```
help -d@ -fmyhelp -imyindex -r
```

Files

/usr/lib/helpfile — Additional system information
/usr/lib/helpindex — Index for helpfile
$HELP — User information
$LASTERROR — Default command help

See Also

apropos, commands, man, Using COHERENT

hmon — Command

Monitor the COHERENT System
hmon

The command **hmon** continually displays a summary of your system's activity. It uses an interactive display with which you can easily send a signal to a selected process.

When you invoke **hmon**, it displays a display that resembles the following:

```
     Last PID=91     Total Mem=15684K        Free Mem=7844K (50.01%)
     Total=20  Running=1   Zombies=0   Locked=0   Waiting=5   Sleeping=14
     PID=91  Idle=75.68%     User= 8.11%     Sys=16.22%
     Load= 1.60      Load Averages:  1:3.38  5:1.01  20:0.27

     PID   PPID  Username Ksize User  Sys   %User %Sys  Flag tty     S Command
        91    89 fred       148 00:04 00:01  5.41  1.80 4001 ttyp1   R hmon
        89    88 fred       129 00:00 00:00  0.00  0.00 6001 ttyp1   W ksh
        88     1 root       735 00:04 00:19  0.00  1.80 4001 null    S xterm
        86    80 fred       208 00:00 00:00  0.00  0.00 6001 ttyp0   S me
        80    78 fred       129 00:00 00:00  0.00  0.00 6001 ttyp0   W ksh
        79    76 fred       284 00:00 00:07  0.90  9.01 4001 null    S fvwm
        78    76 root       727 00:00 00:01  0.00  0.00 4001 null    S xterm
        76    64 fred        79 00:00 00:00  0.00  0.00 6001 null    S sh
        70    64 root      2423 00:15 00:11  1.80  3.60 6001 console S X
        64    54 fred       105 00:00 00:00  0.00  0.00 6001 color0  W xinit
        56     1 root        28 00:00 00:00  0.00  0.00 4001 com2l   S getty
        55     1 root        28 00:00 00:00  0.00  0.00 4001 com3l   S getty
        54     1 fred       129 00:00 00:00  0.00  0.00 6001 color0  W ksh
        53     1 root        28 00:00 00:00  0.00  0.00 4001 color1  S getty
        52     1 root        28 00:00 00:00  0.00  0.00 4001 color2  S getty
        51     1 root        28 00:00 00:00  0.00  0.00 4001 color3  S getty
        47     1 daemon      55 00:00 00:00  0.00  0.00    1 null    S lpsched
        45     1 root        36 00:00 00:00  0.00  0.00    1 null    S cron
```

The first four lines

```
     Last PID=91     Total Mem=15684K        Free Mem=7844K (50.01%)
     Total=20  Running=1   Zombies=0   Locked=0   Waiting=5   Sleeping=14
     PID=91  Idle=75.68%     User= 8.11%     Sys=16.22%
     Load= 1.60      Load Averages:  1:3.38  5:1.01  20:0.27
```

summarize your system's status. The lines that follow summarize each process. Each line contains the following information:

PID The identifier of the process.

PPID The process identifier its parent process. Note that process 1, **init**, has no parent process. For more details on **init**, see its entry in the Lexicon

Username
The login identifier of the user who owns this process.

Ksize The process's size, in kilobytes. Note that this does *not* include memory that the process allocates for itself.

User The amount of user time that this process has consumed.

Sys The amount of system time that this process has consumed.

%User The percent of user time this process has consumed.

%Sys The percent of system time this process has consumed.

Flag The process's flag bits OR'd together, as follows:

PFCORE	00001	Process is in core
PFLOCK	00002	Process is locked in core
PFSWIO	00004	Swap I/O in progress
PFSWAP	00010	Process is swapped out
PFWAIT	00020	Process is stopped (not waited)
PFSTOP	00040	Process is stopped (waited on)
PFTRAC	00100	Process is being traced
PFKERN	00200	Kernel process
PFAUXM	00400	Auxiliary segments in memory
PFDISP	01000	Dispatch at earliest convenience
PFNDMP	02000	Command mode forbids dump
PFWAKE	04000	Wakeup requested

For example, process 8460 has flag “4001”. This means that the process is swapped out and and that a wakeup has been requested. This is consistent with the ‘S’ status, which means that it is sleeping. Note that the flags for swapping do not contain useful information as COHERENT does not yet support demand paging.

tty The port from which the process was launched. This can be the console, a pseudo-tty, or a serial port.

S The process’s status, as follows:

R	Ready to run (waiting for CPU time)
S	Stopped for other reasons (I/O completion, pause, etc.)
T	Process is being traced by another process
W	Waiting for an existent child
Z	Zombie (dead, but parent not waiting)

Command
The name of the program that this process represents.

One of the process lines will be highlighted. You can shift the line of highlighting by pressing the keys `(ª)` and `(°)`. When a process line is highlighted, you can send that process a signal simply by pressing a key, as follows:

1 Send signal **HUP**. Equivalent to typing **kill -1**.

2 Send signal **INTR**. Equivalent to typing **kill -2**.

3 Send signal **QUIT**. Equivalent to typing **kill -3**.

9 Send signal **KILL**. Equivalent to typing **kill -9**.

Whether the signal has any effect will, of course, depend upon the degree of control you have over that process.

To refresh the **hmon** screen, type **L**. To quit, type **Q**.

See Also

commands, ps

Notes

hmon reads the free memory from **/dev/freemem**. If this device does not exist on your system, create it as follows:

```
mknod /dev/freemem c 0 12
chmog 444 sys sys /dev/freemem
```

hmon uses **curses** to manage its display. Your screen will not appear properly if the environmental variable **TERM** is not set correctly for the display device you are using, or if its **terminfo** entry is not correct.

hmon was written by Harry C. Pulley, IV (hpulley@uoguelph.ca).

HOME — Environmental Variable

User's home directory
HOME=*home directory*

The environmental variable **HOME** name's the user's home directory. Some commands use this name by default if they require the name of a directory and none is supplied. For example, if you type the change directory command **cd** without an argument, it will change the current directory to the one named by the **HOME**.

See Also

environmental variables

hosts — System Administration

Names and addresses of hosts on the local network
/etc/hosts

The file **/etc/hosts** gives the name and Internet-protocol (IP) address of remote hosts with which your system can communicate via a network.

Each line within **hosts** describes one host on the network. A description of a host begins with that host's IP address, in normal "dot" notation. This is followed by its name and any aliases it has — that is, other names that also refer to that host. For example, consider the following:

```
666.16.16.27      accounting  acct  beancounters
666.16.16.2 president   boss
666.16.3.5  engineering
```

As you can see, a given host can have more than one alias. Aliases need not be terse; however, you should not use an alias name that you would not want the users of that host to see.

An IP address can appear on more than one line. For example, entry

```
137.229.10.39     raven raven.alaska raven.alaska.edu
```

can also be rendered as:

```
137.229.10.39     raven
137.229.10.39     raven.alaska
137.229.10.39     raven.alaska.edu
```

You may find this to be more legible. However, if you need to change this host's IP address, you must be careful to change every entry, or trouble will result.

/etc/hosts must include the following standard entries:

```
127.1          localhost
127.0.0.1      loopback
```

When you specify only two parts of an Internet address, the second part represents the final three bytes of that address. Thus, the addresses **127.1** and **127.0.0.1** are, in fact, the same address.

The address **127.1** by convention names the local host. Packets sent to this address return to the local host: they do not go onto the Ethernet. This feature is useful in debugging software. The host names **localhost** and **loopback** are also conventional names for your local host.

/etc/hosts should also contain a separate entry for your local host's Internet address and name. You set the name for your system when you installed COHERENT. To change your system's name, edit the file **/etc/uucpname**.

See Also

Adminstering COHERENT, hosts.equiv, inetd.conf, networks, protocols, services, uucpname

hosts.equiv — System Maintenance

Name equivalent hosts
/etc/hosts.equiv

File **/etc/hosts.equiv** names every host on your network whose users are equivalent those on your system.

For example, if system **mwc** names system **lepanto** in its copy of **/etc/hosts.equiv**, then **mwc** assumes that user **fred** on **lepanto** is the same person as user **fred** on **mwc**.

See Also

Administering COHERENT, hosts inetd.conf, networks, protocols, services

hosts.lpd — System Administration

Local system name and domain
/etc/hosts.lpd

File **hosts.lpd** gives your system's name and domain, using dot notation. For example:

```
lepanto.mwc.com
```

Your system's name should be the same as that set in file **/etc/uucpname**, and its domain should be the same as that set in file **/etc/domain**.

See Also

Adminstering COHERENT, domain, hosts, hosts.equiv, inetd.conf, networks, protocols, services, uucpname

hp — Command

Prepare files for Hewlett-Packard LaserJet printer
hp [-acflr] [-i*marg* **] [-t***top* **] [-p***lines* **] [** *file ...* **]**

The command **hp** 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.

hp recognizes the following options:

-f	Print pages in the normal order. This is the default.
-i*marg*	Set the page indentation to *marg*.
-l	Print pages in landscape mode.
-p*lines*	Set the page length to *lines*.
-r	Print pages in reverse order (for LaserJet I).
-t*top*	Set the top margin to *top*.

Example

To generate listings of all C programs in the current directory, enter the command

```
pr *.c | hp | hpr -B
```

See Also

commands, hpd, printer

hpd — System Administration

Spooler daemon for laser printer
/usr/lib/hpd

hpd is the daemon that prints jobs spooled by the command **hpr**. All jobs are printed on the printer that is accessed through device **/dev/hp**. For information on this device, and on printer management in general, see the Lexicon entry **printer**.

The command **hpr** invokes **hpd** automatically. If there is no printing to do, or if another daemon is already running (as indicated by the file **dpid**), **hpd** exits immediately. Otherwise, it searches the spool directory for control files of listings to print. A control file contains the names of files to print, the user name, banner pages, and files to be removed upon completion.

hpd does not print listings in any particular order. There is no prioritization of printing, either by size or by requester.

The command **hpskip** aborts or restarts printing of the job currently being printed by **hpd**.

Files

/dev/rhp — Raw device for LaserJet printer
/usr/spool/hpd — Spool directory
/usr/spool/hpd/cf* — Control files
/usr/spool/hpd/df* — Data files
/usr/spool/hpd/dpid — Lock and process id

See Also

Administering COHERENT, despooler, hpr, hpskip, init, lpd, printer

Notes

Beginning with release 4.2, COHERENT also includes the printer daemon **despooler**, which prints files spooled with the command **lp**. For details on how COHERENT manages printing, see the Lexicon entry for **printer**.

hpr — Command

Spool a job for printing on the laser printer
hpr [-Bcemnr] [-b *banner***] [-f** *fontnum***] [***file ...***]**

The command **hpr** spools each *file* for printing on the Hewlett-Packard LaserJet printer. If no *file* is named on the command line, **hpr** spools what it reads from the standard input.

hpr recognizes the following options:

-B Suppress printing of a banner page. Note that **hpr** outputs its banner in plain text; therefore, if you have a PostScript printer, you *must* use this option. If you do not, your printer will hang.

-b *banner* Print *banner* on the banner page. The default banner is the user's login identifier.

-c Copy each *file* into the spooling directory, instead of reading the file from its home directory. This option lets you edit a *file* before it has finished printing.

-e Erase all "soft fonts" from the printer's memory.

-f *fontnum file1 ... fileN*
Load the Hewlett-Packard "soft fonts" stored in files *file1* through *fileN* into the printer's memory; set the font identifiers to begin at *fontnum*.

-m Write a message on the user's terminal when printing completes.

-n Do not send a message (default).

-r Remove the files when they have been spooled.

The command **hpskip** aborts or restarts printing of the file that is currently being printed. The command **hp** converts **nroff** output into a form usable by the LaserJet.

Examples

To print the file **foo** on the LaserJet, type:

```
hpr -B foo
```

The following example loads the soft fonts in files **foo**, **bar**, and **baz** into the printer's memory, and sets their font identifiers to begin at 15:

```
hpr -f 15 foo bar baz
```

Files

/dev/rhp — Raw device for LaserJet printer
/usr/lib/hpd — Line-printer daemon for LaserJet printer
/usr/spool/hpd — Spool directory for LaserJet printer
/usr/spool/hpd/dpid — Daemon lockfile

See Also

commands, hp, hpd, hpskip, printer

Notes

Beginning with release 4.2, COHERENT also includes the **lp** print spooler. **lp** offers a more sophisticated way to manage printers, especially on machines that support multiple printers of the same type. For details, see the Lexicon entries for **printer** and **lp**.

hpskip — Command

Abort/restart current job on Hewlett-Packard LaserJet
hpskip [-r]

The command **hpskip** aborts or restarts the job being printed on the printer plugged into device **/dev/hp**. The job must have been spooled with the command **hpr**.

By default, **hpskip** aborts the job and prints a message on the user's terminal. When invoked with the **-r** option, it restarts the printing of the current job. This is useful when a printing is spoiled due to, say, a paper jam.

Files

/usr/lib/hpd — LaserJet printer daemon
/usr/spool/hpd — Spool directory
/usr/spool/hpd/dpid — Daemon lockfile

See Also

commands, hpd, hpr, lpskip, printer

Notes

To cancel jobs spooled with the command **lpr**, use the command **lpskip**. To cancel or reprint jobs spooled with the command **lp**, use the commands **cancel** and **reprint**. See the Lexicon entry **printer** for details.

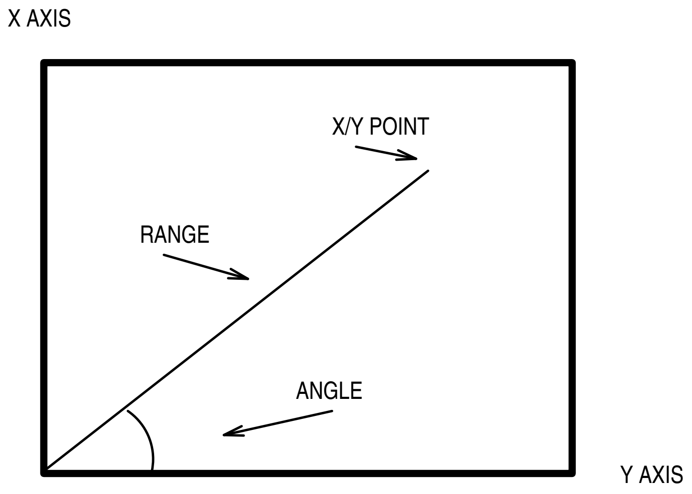
hypot() — Mathematics Function (libm)

Compute hypotenuse of right triangle
#include <math.h>
double hypot(*x*, *y***) double** *x*, *y***;**

hypot() computes the hypotenuse, or distance from the origin, of its arguments *x* and *y*. The result is the square root of the sum of the squares of *x* and *y*.

Example

The following example demonstrates the functions **hypot()** and **atan2()**. It converts an X/Y pair of rectangular coordinates into polar coordinates. Thus, an X/Y pair of 1,1 produces a range of 1.41 and 45°; and an X/Y pair of 3,4 would produce a range of five and 36.87°. The following sketch illustrates this:



This example was written by Brent Seidel (brent_seidel@chthone.stat.com):

```
#include <stdio.h>
#include <math.h>

main()
{
      double x, y, angle, range;
      char  buffer[100];

      printf("Enter the X/Y pair: ");
      fflush(stdout);
      gets(buffer);
      sscanf(buffer, "%lf,%lf", &x, &y);

      range = hypot(x, y);
      angle = atan2(x, y);
      printf("The range is %f\n", range);
      printf("The angle is %f radians or %f degrees.\n",
            angle, angle * 180.0/PI);
}
```

See Also

cabs(), libm