qfind — Command

Quickly find all files with a given name
qfind [-adpv] *name* **...**
qfind [-bv] [-s*directory***]**

The command **qfind** prints the full path name of each file with a given *name*. It reads a prebuilt data base, for the sake of speed. This makes **qfind** much faster than **find** for locating a file; but it does mean that changes to the file system since the data base was last updated will not be reflected in what **qfind** prints.

The option **-b** tells **qfind** to build its data base in file **/usr/adm/qffiles**. By default, this data base names every file in your system. If you wish to suppress a directory, name it with the **-s** option. For example, to build the data base but suppress the directory **/usr/spool**, use the command:

```
qfind -b -s/usr/spool
```

This command excludes the contents of directory **/usr/spool** and all of its children from the **qfind** data base.

When invoked without the **-b** option, **qfind** reads its data base to find file names.

Normally, **qfind** prints the full path name of each file in the COHERENT system that ends with the given *name* (as it was when you last executed **qfind -b**.) With the **-d** option, **qfind** prints matching directories instead of files. With the **-a** option, **qfind** prints both matching files and matching directories.

Option **-p** specifies partial name matching. For example, **qfind -p foo** matches files **/src/foo.c** and **/doc/foo.r** as well as file **/usr/bin/foo.**

Finally, option **-v** tells **qfind** to print verbose output.

Files

/usr/adm/qffiles

See Also

commands, cron, find, whereis, which

Notes

Building the **qfind** data base with the **-b** option is slow, but it speeds finding files. You may find it convenient to use **cron** to execute **qfind -b** to rebuild the data base at night, or some other time when the machine is otherwise idle.

If you want to include all files in the data base, the superuser **root** must run **qfind -b**.

qpac — Command

Map the file system
qpac *raw_device*

Command **qpac** builds a map of the file system *raw_device*. It quits before it writes to or changes the file system. You can use this to examine how your file system is laid out.

See Also

commands, dpac, fmap, fsck, spac, upac

Notes

qpac is a link to the command **dpac**.

qpac was written by Randy Wright (rw@rwsys.wimsey.bc.ca).

qsort() — General Function (libc)

Sort arrays in memory
#include <stdlib.h>
void qsort(*data***,** *n***,** *size***,** *comp***)**
char ******data***; int** *n***,** *size***; int (****comp***)();**

qsort() is a generalized algorithm for sorting arrays of data in memory, using C. A. R. Hoare's "quicksort" algorithm. **qsort()** works with a sequential array of memory called *data*, which is divided into *n* parts of *size* bytes each. In practice, *data* is usually an array of pointers or structures, and *size* is the **sizeof** the pointer or structure. Each routine compares pairs of items and exchanges them as required. The user-supplied routine to which *comp* points performs the comparison. It is called repeatedly, as follows:

```
(*comp)(p1, p2)
char *p1, *p2;
```

Here, *p1* and *p2* each point to a block of *size* bytes in the *data* array. In practice, they are usually pointers to pointers or pointers to structures. The comparison routine must return a negative, zero, or positive result, depending on whether *p1* is logically less than, equal to, or greater than *p2*, respectively.

Example

For an example of this function, see the entry for **malloc()**.

See Also

libc, shellsort(), strcmp(), stdlib.h, strncmp()
The Art of Computer Programming, vol. 3
ANSI Standard, §7.10.5.2
POSIX Standard, §8.1

Notes

The COHERENT library also includes the sorting function **shellsort()**. These functions use different algorithms for sorting items; each algorithm has its strengths and weaknesses. In general, the quicksort algorithm is faster than the shellsort algorithm for large arrays, whereas the shellsort algorithm is faster for small arrays (say, 50 items or fewer). The quicksort algorithm also performs poorly on arrays with a small number of keys, e.g., an array of 1,000 items whose keys are all '7' and '8'.

To get around these limitations, the COHERENT implementation of **qsort()** has an adaptive algorithm that recognizes when the quicksort algorithm is performing badly, and calls **shellsort()** in its place.

quot — Command

Summarize file-system usage
quot [-c] [-f] [-n] [-t] *filesystem*

quot produces several different summaries about the ownership of files for each *filesystem* argument given. When no options are specified, **quot** produces a two-column listing that gives the amount of space used by each user, sorted in decreasing order of file space used; the first column gives the number of blocks used and the second gives the use name. Space is always given in blocks.

Options are available to modify the normal output or specify a completely different action.

quot recognizes the following options:

-c Give a three-column breakdown of files by size. The first column contains all file sizes, in increasing order. The second column gives the number of files of the size indicated in the first. The third gives a cumulative sum of the sizes of all files less than or equal to the current size.

-f Add an initial column that contains the number of files to the front of the normal output.

-n Takes as input a list of i-numbers and file names, one per line and sorted in ascending order by i-number; ignore all lines not in this form. The output is in two columns: the first gives the owner and the second contains the file name for each entry in the output. This conforms to usage with the following pipeline:

```
ncheck filesystem | sort +0n | quot -n filesystem
```

-t To the normal output, add a line that contains totals.

quot runs much faster with a raw device for *filesystem*.

Only the superuser **root** can run **quot**.

Files

/etc/passwd

See Also

ac, commands, ncheck, sort

Notes

Sparse files are recorded as if they had all of their blocks allocated.