NAME

samefile − find identical files

SYNOPSIS

samefile [-g size] [-l | -r] [-s sep] [-0aiqVvx]

DESCRIPTION

samefile reads a list of filenames (one filename per line) from stdin. For each filename pair with identical contents, a line consisting of six fields is output: The size in bytes, two filenames, the character ‘‘=’’ if the two files are on the same device, ‘‘X’’ otherwise, and the link counts of the two files. The output is sorted in reverse order by size as the primary key and the filenames as the secondary key.

OPTIONS

-0 Indicates that the input list of file names is NUL terminated, for example as generated by implementations of **find**(1) that support the **-print0** option. Without this option, the file names are assumed to be newline terminated.

-a Do not sort files with same size alphabetically.

-g size Compare only files with size greater than *size* bytes. Default is 0.

-i Allow files with the same device/i-node pair to be added to the binary tree. This might be useful if output will be fed into some other program. If this option is used, the statistics displayed when using **-v** will not contain the ‘‘You have a total of x bytes in identical files’’ line because **-i** prohibits proper calculation of this value.

-l Do not check if files with identical contents are hard links created by **ln**(1). By default, **samefile** checks if files with identical contents are hard linked and, if they are, does not write a name pair to stdout. A slight speedup is gained when using this option. This option is incompatible with the **-r** option.

-q Do not issue warning messages when **open**(2) fails. When you encounter such a warning, **open** probably failed due to a ’permission denied’ error on files or directories for which you have no read permission. Useful if you are not root and want to compare your files against files in a system directory like */etc*

-r Report whether identical files are hard linked. The separator string followed by the [bracketed] link count is appended to each name pair if they are hard links created with **ln**. This option is incompatible with the **-l** option. Note that this kind of output has only four fields and will appear unsorted before the actual output of **samefile**.

-s sep Use string **sep** as the output field separator, defaults to a tab character. Useful if filenames contain tab characters and output must be processed by another program, say **awk**(1).

-V Print the version information and exit.

-v verbose mode. Write some statistical messages about memory usage and work reduction as well as the sum of the sizes of all identical files to stderr.

-x Switch off intelligence. This option prevents **samefile** from being smart. If files *file1, file2* and *file3* are identical, it will do 3 comparisons instead of just the two needed and write more output. See the discussion under INTERNALS why this could be useful. If this option is used, the statistics displayed when using **-v** will not contain the ‘‘You have a total of x bytes in identical files’’ line because **-x** prohibits proper calculation of this value.

INTERNALS

samefile uses two stages to give optimum performance.

In the first stage, all non-plain files are skipped (directories, devices, FIFOs, sockets, symbolic links) as well as files for which **stat**(2) fails and files that have a size less than or equal to *size.* Output of the first stage (the filenames) is written into a binary tree with one node for every file size. It is also at this early stage where checks for hard links are done. If hard links are found, and **-r** is requested, the name pairs are output immediately. The whole list of hard linked name pairs will therefore appear before any output of the second stage.

For any i-node only one filename will be added to the binary tree (unless **-i** was requested.)

In the second stage all files having the same size are compared against each other. The rules of mathematical logic are applied to reduce work and output noise (unless **-x** is requested): if files a, b, and c have the same size and **samefile** finds that a = b and a = c then it will not compare b against c (and will not output a line for b and c) but only for a = b and a = c. Note however, that because only the first filename per i-node gets into the second stage, the output for a group of identical files with different i-node numbers is also minimized. Suppose you have six identical files of size 100 in an i-node group consisting of the three i-nodes with numbers 10, 20 and 30 (the term 'i-node group' has nothing to do with the i-node group notion of some file systems - it merely refers to a set of i-nodes addressing files with identical contents):

```
$ ls -i
  10 file1    20 file4    30 file6
  10 file2    20 file5
  10 file3
$ ls | samefile
100    file1   file4   =    3    2
100    file1   file6   =    3    1
```

The sum of the sizes in the first column is the amount of disk space you could gain by making all 6 files links to only one file or remove all but one of the files. To be precise, disk space is allocated in blocks - you will probably gain two blocks here, rather than 200 bytes. Note that it is not enough to just remove file4 and file6 (you would gain only 100 bytes because file5 still exists.) The proper way is to use the -i option. The output will look like

```
100    file1   file2   =    3    3
100    file1   file3   =    3    3
100    file1   file4   =    3    2
100    file1   file5   =    3    2
100    file1   file6   =    3    1
```

Removing all files listed in the third field will leave only file1. Making all files hard links to file1 is easy. If the fourth field is a "=" do a forced hard link. If you need to know about *all* combinations of identical files, then you use both the **-i** and **-x** option. This produces

```
$ ls | samefile -ix
100    file1   file2   =    3    3
100    file1   file3   =    3    3
100    file1   file4   =    3    2
100    file1   file5   =    3    2
100    file1   file6   =    3    1
100    file2   file3   =    3    3
100    file2   file4   =    3    2
100    file2   file5   =    3    2
100    file2   file6   =    3    1
100    file3   file4   =    3    2
100    file3   file5   =    3    2
100    file3   file6   =    3    1
100    file4   file5   =    2    2
100    file4   file6   =    2    1
100    file5   file6   =    2    1
```

EXAMPLES

Find all identical files in the current working directory:

$ ls | samefile

Find all identical files in my HOME directory and subdirectories and also tell me if there are hard links:

$ find $HOME -type f | samefile -r

Find all identical files in the */usr* directory tree that are bigger than 10000 bytes and write the result to *usr.dups* (that one is for the sysadmin folks, you may want to ’amp’ - put it in the background with the ampersand & - this command because it takes a few minutes.)

$ find /usr -type f | samefile -g 10000 >usr.dups

DIAGNOSTICS

You will see a short usage message if you use an invalid option.

malloc - free = xxxx

I didn’t free the memory I’ve **malloc**(3)ed. You found a bug. Please report it to the author.

Allocation failed for ’expr’ ...

Oops! You ran out of virtual memory. You must have a real big filename list. Try to use a smaller one or increase resources available to your processes. For more information see **ulimit**(1) or your similar shell builtin.

SEE ALSO

ln(1), **find**(1), **rm**(1), **df**(1)

BUGS

There are no known bugs. The source has been **lint**(1)ed and all possible care has been taken while coding. If you find a bug (or miss a feature) please contact the author.

HOME

The official **samefile** home page www.schweikhardt.net/samefile/ is maintained by the author Jens Schweikhardt - schweikh at schweikhardt dot net