Homework 4: CS114 Unix Tools, Fall 2007

Due electronically (via CMS) on Wednesday 31 October 2007, 11:59PM

1 Packages

In this section, you will download and install a software package, and then run it. In general this process can be somewhat complicated, but here we will make it very simple.

- Download the `.tar.gz` source tarball of version 2.8.6 of the Lynx web browser. This file can be found at `http://lynx.isc.org/lynx2.8.6/lynx2.8.6.tar.gz`. You can either download this on your computer and transfer it to your directory on the CSUG linux machines, or download it directly on the CSUG machines (using, e.g. `wget`).
- Extract the tarball into a subdirectory of your home directory, and `cd` into the directory you've just created. Instructions for using `tar` can be found in lecture 10.
- Execute `./configure --prefix=$HOME` This will run for a bit and output a page or so of text.
- Execute `make`. This will build `lynx`, and so it will take a little while to run.
- Execute `make install`. This will copy the lynx program into the `bin` subdirectory of your home directory.
- Run `lynx`. You probably have to type `rehash` to force the shell to realize that `lynx` is now in a place it knows about. If for some reason the shell cannot find the program, you can type the absolute path, which should be `/home/`*`your-net-id`*`/bin/lynx`.
- Navigate to this page: `http://www.cs.cornell.edu/courses/cs114/2007fa/magic.php`. If you've followed the instructions above, you should see the magic word. Write this magic word in a file called `magic.txt`, and submit it in CMS by the due date. Do not share the magic word with your classmates.

2 A shell script: finding big files

Have you ever wanted to figure out what's eating up all the space on your hard drive? In this problem, you will write a short script to help with this. The script you will write is called `listbig.sh`. It should be a Bourne shell script. It will take one argument on the command line, which is the path to a file or directory. It will write out one number to standard output, as well as further information to a file called `bigfiles`.

The size of a file is the number of bytes taken up by a file (you can find this out, for example, by looking at the output of `ls -l`). Now let's define the size of a directory as the sum of the sizes of all the files in the directory, plus the sizes of all the subdirectories in the directory, plus the size of the directory itself (as shown by `ls -ld`).

Now let's define a concept similar to size, but slightly different – *bigness*. The *bigness* of a file is defined as the size of the file. A *big file or directory* is one whose bigness is greater than 2048 bytes (2K). The *bigness* of a directory is similar to the size of a directory, but with a twist. The bigness of a directory is the total size of the directory (as defined above) **minus** the bignesses of any directories or files below it which are themselves big.

Consider this example.

```
% ls -lR top
total 16
-rw-r--r--   1 ebreck  ebreck  1000 Oct 17 17:57 1k
-rw-r--r--   1 ebreck  ebreck  3000 Oct 17 17:57 3k
drwxr-xr-x   5 ebreck  ebreck   170 Oct 17 17:58 sub/
drwxr-xr-x   3 ebreck  ebreck   102 Oct 17 17:59 sub2/

top/sub:
total 24
-rw-r--r--   1 ebreck  ebreck  1000 Oct 17 17:58 one
-rw-r--r--   1 ebreck  ebreck  1000 Oct 17 17:58 three
-rw-r--r--   1 ebreck  ebreck  1000 Oct 17 17:58 two

top/sub2:
total 8
-rw-r--r--   1 ebreck  ebreck  1000 Oct 17 17:59 one
```

In this example, the file `top/3k` is big, and the directories `top/sub` and `top` are big, but no other file or directory is big.

The script should take one argument on the command line - the argument is a path to a file or directory that exists. The script should always print to standard output one number - either 0, if the argument is big, or otherwise the bigness of the argument. Also, for any file or directory which is a child of the argument and is big, it should append to the file `bigfiles` the name of that child and its bigness.

If the script had been invoked as follows `listbig.sh top`, it would print out `0`, and append to the file `bigfiles` the following

```
top/3k 3000
top/sub 3170
top 2306
```

Here is one possible pseudocode for the script. Your goal is to implement this as a Bourne shell script, and submit it in CMS by the due date.

```
size = (figure out size based on looking at the output of ls -l)
arg = first argument to this script
if arg is a directory then
  for every child of this directory
    childsize = (recursively call this script on child)
    size = size + childsize
if size > 2048 then
  append arg and size to the file bigfiles
  print 0 to standard output
else
  print size to standard output
```