CSCI415—Systems Programming

Spring 2017

Programming Assignment 3
Multiprocessing and File Statistics

Problem Statement

A Unix systems programmer has the ability to fork/exec binaries, to establish pipes for synchronous communication between the processes executing those binaries, and to use signals for asynchronous interaction between those processes. This permits her to "reuse" small utilities to produce more complex functions. This is particularly useful when there is no source available for the small utilities.

Components

This assignment requires that you build a five process system from three relatively small programs.

The `totalsize` Program

The `totalsize` program expects that its standard input is a list of file names separated by "whitespace" (see the `isspace(3)` manual page). For all *regular* files whose name are on its stdin, it computes the total size of all those files it can access. That total size is sent to its standard output as a nicely formatted integer (a string that looks like an integer, actually). For example, suppose a directory contains the following files (the `-lu` option requests a long directory listing with time of last access shown):

```
% ls -lu
total 27
-rwxr-xr-x   1 kearns   wheel       24576 Feb 23 13:51 a.out
-rw-r--r--   2 kearns   wheel         612 Feb 21 13:51 fileinfo.c
-rw-r--r--   2 kearns   wheel         612 Feb 21 13:51 link.c
lrwxrwxrwx   1 kearns   wheel          10 Feb 23 14:20 symlink.c -> fileinfo.c
```

Below we show the invocation of `totalsize` assuming that the above directory is the current working directory:

```
% echo ". .. a.out fileinfo.c link.c symlink.c" | ./totalsize
25188
```

This indicates that there are 25118 bytes of storage tied up in the regular files in the list. The directories `.` and `..` are not regular files; `fileinfo.c`, `link.c`, and `symlink.c` all refer to the same regular file (the first two are hard links, and the last is a symbolic link).

If the `UNITS` environment variable has value `"K"` or `"k"` when `totalsize` is invoked, then the total size of the named regular files *in kilobytes* is printed as an integer (truncate, don't round up) followed by the string "kB."

```
% export UNITS=K
% echo ". .. a.out fileinfo.c link.c symlink.c" | ./totalsize
24kB
% export UNITS=bogus
% echo ". .. a.out fileinfo.c link.c symlink.c" | ./totalsize
25188
```

If the `TSTALL` environment variable looks like a positive non-zero integer, then `totalsize` should sleep for that many seconds before inputting a file name.

If the `TMOM` environment variable looks like a positive non-zero integer, then `totalsize` should treat the value as a pid and signal that pid with a SIGUSR1 after it has produced all of its output

The `accessed` Program

Like `totalsize`, the `accessed` program takes a list of file names on its standard input. It also takes a mandatory argument. The program must be invoked as

`accessed` *num*

where *num* is an integer. If *num* is positive, `accessed` outputs, on its standard output, those regular files to which it has access which have *not* been accessed for *num* days. For example, assuming that it is about 3 p.m. on February 23:

```
% echo ". .. a.out fileinfo.c link.c symlink.c" | ./accessed 1
link.c
% echo ". .. a.out fileinfo.c link.c symlink.c" | ./accessed 5
%
```

When there are multiple links to a file, any (single) link will do as the name of the file.

If *num* is negative, `accessed` outputs, on its standard output, those regular files to which it has access which *have* been accessed within *num* days. Again, assuming that it is about 3 p.m. on February 23:

```
% echo ". .. a.out fileinfo.c link.c symlink.c" | accessed -1
a.out
% echo ". .. a.out fileinfo.c link.c symlink.c" | accessed -5
a.out
link.c
```

A value of 0 for the integer argument is invalid. A value that is too large or too small (i.e., negative) to be represented on the system is also invalid.

The Driver Program, `report`

The `report` program will be responsible for creating and interconnecting the other processes of this multiprocess computation.

- It takes a list of filenames on its `stdin`.
- It creates four child processes, two execute the `totalsize` binary and two the `accessed` binary.
- Pipes must be established so that `report` feeds the file names read on its `stdin` onto the `stdin` of *both* children running `accessed`.
- Pipes must be established to bind the `stdout` of each `accessed` process onto the `stdin` of an associated `totalsize binary`.
- Pipes must be established so that `report` can read the `stdout` of *both* `totalsize` children.
- The `report` program accepts a mandatory non-zero positive integer, *num*, as its first command line argument. The integer specifies a duration in days. The first `accessed` process will essentially be invoked as

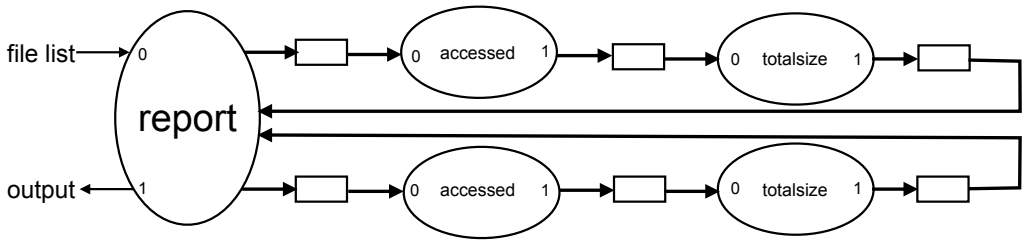
  `accessed` *num*

  The second will essentially be invoked as

  `accessed` *-num*

  The net effect is that the first `accessed` process should output those files which have not been accessed in *num* days. The second should output those files which have been accessed in *num* days.
- It accepts an optional `-k` as a command line argument that causes it to display all size information in kilobytes.
- It accepts an optional `-d` as a command line option. If `-d` is asserted the next command line argument must be a positive non-zero integer that specifies the number of seconds that the `totalsize` processes sleep before inputting a file name.
- After writing the last file name onto its output pipes, it enters a non- terminating loop in which it sleeps for 1 second and outputs an asterisk on its standard output.
- Upon receipt of a SIGUSR1 signal, it exits the loop, reads the output of the first `totalsize` process, and outputs a nicely formatted message. It prints a separator line (perhaps hyphens or asterisks). It then outputs a message incorporating the stdout of the second `totalsize` process.
- On termination, no garbage files or orphan processes should be left on the system.

The following figure shows how a running system should appear.The heavy lines connecting the processes represent pipe-based communication.



In the context of our example, here are some executions of the system:

```
% echo ". .. a.out fileinfo.c link.c symlink.c" | report 1 -d 2
******
A total of 612 bytes are in regular files not accessed for 1 days.
----------
A total of 24576 bytes are in regular files accessed within 1 days.

% echo ". .. a.out fileinfo.c link.c symlink.c" | report 5 -k
A total of 0kB are in regular files not accessed for 5 days.
----------
A total of 24kB are in regular files accessed within 5 days.
```

Note that the exact number of asterisks is roughly proportional to the delay (if any).

Notes

The `totalsize` and `accessed` programs must be able to run as stand-alone binaries. We will test them in stand-alone mode before testing the full-blown system of five processes.

The TA will look at your source code to make sure you follow the termination protocol as specified (i.e., that one of the `totalsize` processes signals `report` to pull it out of the asterisk-printing loop). You are well advised to make your source code neat and readable.

You must use the `fork()` system call and any of the variants of the `exec()` system call/library function that you choose. This implies that you may not use the `system()` and `popen()` library functions (or similar) in this assignment.

All information about a file can be obtained from the `stat()` system call. I will have a special mini-lecture on the use of `stat()` and computations that involve real time.

Due

The submission deadline is 11:59pm on Tuesday, February 28.

You must structure your system to have three source files, `report.c`, `totalsize.c`, and `accessed.c`. Follow these steps to submit:

1. `tar zcvf system.tgz report.c totalsize.c accessed.c`
   This produces a gzipped (compressed) "tarball" consisting of your three source files.

2. `gpg -r CSCI415 --sign -e system.tgz`
   This signs and encrypts your gzipped tarball so the TA can retrieve the submission.

3. `mv system.tgz.gpg ~; chmod 644 ~/system.tgz.gpg`
   This moves your submission to your home directory and protects it so that it can be copied by the TA.