Assignment 4
CS-351
Fall 2015
Due Date: 11/23/2015 at 11:59 pm (No extensions)

Goals:

1. To appreciate the benefits and challenges of the pipe IPC mechanism.
2. To experiment with POSIX pipes.
3. To use `pipe()`, `read()`, and `write()` system calls in order to implement parent-child process IPC.
4. Employ the pipe IPC mechanism to solve a real-world problem.
5. To implement a program for computing the hashes of files using multiple hash algorithms.

Overview

Hash algorithms map large data sets of variable length (e.g. files), to data sets of a fixed length. For example, the contents of a 1GB file may be hashed into a single 128-bit integer. Many hash algorithms exhibit an important property called an *avalanche effect* – slight changes in the input data trigger significant changes in the hash value.

Hash algorithms are often used for verifying the integrity of files downloaded from the web. For example, the website hosting the file posts the hash of the file computed using the `MD5` hash algorithm. The user then verifies the integrity of the downloaded file, by computing its hash using the `MD5` algorithm and comparing the hash against the hash posted on the website. The download is successful only if the two values match.

In this assignment you will implement a program for computing a hash of the file using `MD5`, `SHA1`, `SHA224`, `SHA256`, `SHA384`, and `SHA512` hashing algorithms. Your program shall take the name of the target file as a command line argument, and then do the following:

1. Check to make sure the file exists.
2. Create two pipes.
3. Create a child process.
4. The parent transmits the name of the file to the child (over the **first pipe**).
5. The child receives the name of the file and computes the hash of the file using the `MD5` algorithm (using Linux program `md5sum` - see Technical Details).
6. The child transmits the computed hash to the parent (over the **second pipe**), and terminates.

7. The parent receives the hash, prints it, and calls `wait()`.
8. Repeat the same process starting with step 3, but using algorithms `SHA1...SHA512`.
9. The parent terminates after all hashes have been computed.

Technical Details

The programs used for computing file hashes have been installed on the Titan server (`ecs.fullerton.edu`) (accessible from the Titan server using the following command line. They are as follows:

- `md5sum`: a program for computing file hashes using MD5 algorithm.
- `sha1sum`: a program for computing file hashes using SHA1 algorithm.
- `sha224sum`: a program for computing file hashes using SHA224 algorithm.
- `sha256sum`: a program for computing file hashes using SHA256 algorithm.
- `sha384sum`: a program for computing file hashes using SHA384 algorithm.
- `sha512sum`: a program for computing file hashes using SHA512 algorithm.

You may use the `popen()` system call in order to launch the above programs and capture their outputs into a character buffer. A sample program illustrating the basic usage of `popen()` can be found on Titanium (`popen.cpp`).

In order to create a one-way pipe, you must use the `pipe()` systemcall discussed in class. The details of `pipe()` usage can be found in the course slides, at http://linux.die.net/man/2/pipe, and at http://beej.us/guide/bgipc/output/html/multipage/pipes.html. Also, a sample program demonstrating the usage of `pipe()` can be found on Titanium (`pipe.cpp`). You must create the following two pipes:

1. **parent-to-child pipe**: used by the parent to transfer the name of the file to the child. The parent writes to this pipe and the child reads it.
2. **child-to-parent pipe**: used by the child to transfer the computed hashes to the parent. The child writes to this pipe and the parent reads it.

The program skeleton code can be found on Titanium (`skel.cpp`). The use of the skeleton code is not required.

Finally, ***your program must include a makefile*** which builds your program when the user types `make`. A sample makefile to build the skeleton code can be found on Titanium. Will cover the details of writing makefiles, in class. You may find the following links useful:

- http://mrbook.org/tutorials/make/
- http://www.cprogramming.com/tutorial/makefiles.html
- http://www.embeddedheaven.com/linux-makefile-tutorial.htm

Running the Program

Your program must be runnable using the following command line: `./multihash <TARGET FILE NAME>`. For example: `./multihash file.txt` where file.txt is the target file.

Your program must output the results in the following format:

```
<HASH ALGORITHM1 NAME> HASH VALUE : <HASH PROGRAM1 OUTPUT>

<HASH ALGORITHM2 NAME> HASH VALUE : <HASH PROGRAM2 OUTPUT>

<HASH ALGORITHM3 NAME> HASH VALUE : <HASH PROGRAM4 OUTPUT>

<HASH ALGORITHM5 NAME> HASH VALUE : <HASH PROGRAM5 OUTPUT>

<HASH ALGORITHM6 NAME> HASH VALUE : <HASH PROGRAM6 OUTPUT>
```

For example:

```
md5sum HASH VALUE: e235b63c02644e219b7bf3668f479c9e ubuntu-12.4.04.1-desktop-i386.iso

sha1sum HASH VALUE: 0a902c243b342406ba1a6c2c07c7fa1b2af70984
ubuntu-12.4.04.1-desktop-
i386.iso

sha224sum HASH VALUE: 614c81bc334e28ed409cb5c87845d0d2f7030a31326a3e4108e0712b
ubuntu
-12.4.04.1-desktop-i386.iso

sha256sum HASH VALUE: 8e5edafb4b59817a9da51545c99c4c780c7cceb511e32d4c5ac519332224081f
ubuntu-12.4.04.1-desktop-i386.iso

sha384sum HASH VALUE: 985da622cf509eb196ab300b8c9d0d53f07e596297e3b66b4f328862e76e7071
63e25ea4b45b6160f5cfde1179b6bd00 ubuntu-12.4.04.1-desktop-i386.iso

sha512sum HASH VALUE: f760995662402eb938e3a23168be0df3524892340f2abdd1a1027ab89399a9ce
f6981d008097975371bc314bcdfff645af9ae6c6790d31e8eee639ebc7d5d729
ubuntu-12.4.04.1-desk
top-i386.iso
```

EVEN MORE BONUS POINTS (+5; challenging!)

Extend your program to parallelize the computation of hashes by forking all children simultaneously. Hint: you will need to use the `select()` function. You may find the following link useful: http://linux.die.net/man/2/select. Hint: each end of the pipe is a file descriptor.

SUBMISSION GUIDELINES:

- ***This assignment MUST be completed using C or C++ on Linux.***
- ***Your assignment must compile and run on the TITAN server.*** Please contact the CS office if you need an account.
- Please hand in your source code along with the makefile electronically (do not submit .o or executable code) through **TITANIUM**.
- You must make sure that the code compiles and runs correctly.
- Write a README file (text file, do not submit a .doc file) which contains
  - Your name and email address.
  - The programming language you used (i.e. C or C++).
  - How to execute your program.
  - Whether you implemented the extra credit.
  - Anything special about your submission that we should take note of.
  - Please include this assignment in the archive of assignment 3, but please include it in the folder called "bonus2".

Academic Honesty:

Academic Honesty: All forms of cheating shall be treated with utmost seriousness. You may discuss the problems with other students, however, you must write your **OWN codes and solutions**. Discussing solutions to the problem is **NOT** acceptable (unless specified otherwise). Copying an assignment from another student or allowing another student to copy your work **may lead to an automatic F for this course**. Moss shall be used to detect plagiarism in programming assignments. If you have any questions about whether an act of collaboration may be treated as academic dishonesty, please consult the instructor before you collaborate. Details posted at http://www.fullerton.edu/senate/documents/PDF/300/UPS300-021.pdf.