Assignment 1

## Due July 26th at 11:55PM

Goals:

1. To gain hands-on experience with fork() and wait() system calls.

2. To implement a multi-process downloader application.

3. Overview Process Performance comparisons

For extra resources and vm setup, please visit https://gamble.ecs.fullerton.edu/resources. Download all the related files on Titanium.

## System Calls:

Write a simple shell program called shell. When ran, the parent process will print out the prompt which looks like:

cmd>

and will then wait for the user to enter a command such as ls, ps, or any other command. For example:

cmd> ls

The parent process will then use fork() in order to create a child and then call wait() in order to wait for the child to terminate. The child process will use execlp() in order to replace its program with the program entered at the command line. After the child produces the output, the parent will prompt the user for another command. For example:

cmd>ls

file.txt fork fork.c mystery1 mystery1.cpp mystery2 shell shell.cpp

The parent will repeat the above sequence until the user types the exit command, which will cause the parent process to exit.

# File Downloaders:

File downloaders are programs used for downloading les from the Internet. In this assignment, you will implement two different types of multi-process downloaders (i.e., le downloaders that comprise multiple processes):

1. a serial file downloader which downloads files one by one.

2. a parallel file downloader which downloads multiple files in parallel.

You will then compare the performance of the two types of downloaders. Both downloaders will use the Linux wget program in order to perform the actual downloading.

The usage of the wget is simple: wget <FILE URL>. For example, running from command line the following command:

# wget http://ftp.gnu.org/gnu/wget/wget-1.5.3.tar.gz

will download the wget Linux downloader to the current directory. Before proceeding with the assignment, you may want to take a moment to experiment with the wget command.

In your program, the parent process shall first read the file, urls.txt, containing the URLs of the files to be downloaded. urls.txt shall have the following format:

<URL1>

<URL2>

.

.

.

For example:

http://ftp.gnu.org/gnu/wget/wget-1.5.3.tar.gz

https://www.gnu.org/s/bash/manual/bash.pdf

Next, the parent process shall fork the child processes. Each created child process shall use the execlp() system call to replace its executable image with that of the wget program. The two types downloaders are described in detail below.

The two downloaders shall be implemented as separate programs. The serial downloader program shall be called serial.c (or .cpp extension if you use C++). The parallel downloader program shall be called parallel.c (or .cpp extension if you use C++).

### Serial Downloader

The serial downloader shall download les one by one. After the parent process has read and parsed the urls.txt file, it shall proceed as follows:

1. The parent process forks off a child process.

2. The child uses execlp("/usr/bin/wget", "wget", <URL STRING1>, NULL) system call in order to replace its program with wget program that will download the first file in urls.txt (i.e. the le at URL <URL STRING1>).

3. The parent executes a wait() system call until the child exits.

4. The parent forks off another child process which downloads the next le specified in urls.txt.

5. Repeat the above steps until all files are downloaded.

### Parallel Downloader

1. The parent forks off n children, where n is the number of URLs in urls.txt.

2. Each child executes execlp("/usr/bin/wget", "wget", <URL STRING>, NULL) system call where each <URL STRING> is a distinct URL in urls.txt.

3. The parent calls wait() (n times in a row) and waits for all children to terminate.

4. The parent exits.

Please note:

While the parallel downloader executes, the outputs from different children may intermingle. This is acceptable.

fork.c file posted on Titanium provides an example of using fork(), execlp(), and wait() system calls. Please feel free to modify it in order to complete the above tasks. (Check the Samples.zip file on Titanium)

### Performance Comparison

Use the time program to measure the execution time for the two downloaders. For example:

time ./serial

real 0m10.009s

user 0m0.008s

sys 0m0.000s

The column titled real gives the execution time in seconds. Please get the execution times for both downloaders using the following urls.txt file: (Check the url.txt file on Titanium)

http://delaunay.ecs.fullerton.edu/~mshafae/srl/CSUF\_Ubuntu\_Mate\_16\_04\_01\_amd64\_20170122.ova

https://www.dropbox.com/s/9frc9x5blloo7a4/CSUF\_Ubuntu\_Mate\_16\_04\_01\_amd64\_20170122.ova?dl=0

Your execution times should be submitted along with your code (see the section titled \Submission Guidelines".

## Extra Questions:

In your README.TXT submission, please include the answers to the following questions (you may need to do some research):

1. In the output of time, what is the difference between real, user, and sys times?

2. Which is longer: user time or sys time? Use your knowledge to explain why.

3. When downloading the files above, which downloader finishes faster? Why? Please Explain.

4. Repeat the experiment for 10 files (any reasonably large-sized les, e.g., 100 MB, will do). Is the downloader in the previous question still faster? If not so, why? Please Explain.

# SUBMISSION GUIDELINES:

This assignment MUST be completed using C or C++ on Linux. You may work in groups of 2 to 4.

Your assignment must compile and run on the TITAN server. Please contact the CS office if you have trouble logging in and questions about the Titan server.

Please hand in your source code electronically (do not submit .o or executable code). TWO links are provided on 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 names and email addresses of all the students in your group.
* The programming language you used (C or C++).
* How to execute your program. (Details about how to run your code, Programming language + environment + manual with steps)
* The execution times for both downloaders.
* The answers to all questions in this assignment.
* Whether you are volunteering to present and who will present what part of the assignment. July 27th
* Anything special about your submission that we should take note.
* Hard Time/drama? Please explain your hard times and how you conquered them.
* Invalid/incomplete README.txt shall revoke the grade for this assignment

You will upload part of your assignment (README.txt) to Turnitin.com on titanium, which means you will get a similarity percentage of how much data you have used from your references on your Raedme.txt. For an undergrad class, less than 15% is acceptable. More than acceptable percentage is not a candidate for grading. If you get under 10%, it means you have done your task right. Do not use direct quotes for this class’ assignments. The length of the quotes will be counted towards your similarity percentage.

Coding style must conform to professional norms. At a minimum, code must be commented, have descriptive names for identifiers, and contain a comment at the top of each file with pertinent information such as the students’ name, email address, and assignment name.

If you are volunteering to present the assignment in class for a bonus, you need to be ready the next session of the class after the submission due date. (July 27th)

For each assignment, you will upload

1. A tar file (per group) including your codes including your Readme.txt on Titanium, (Place all your files under one directory with a unique name)
2. Readme.txt separately on the link provided for the Readme.txt on Titanium.

Place all your files under one directory with a unique name (such as p1-[userid] for assignment 1, e.g. p1-shariri1).

Tar the contents of this directory using the following command.

tar cvf [directory name].tar [directory name] E.g. tar -cvf p1-shariri1.tar p1-shariri1/

http://www.bic.mni.mcgill.ca/users/kate/Howto/tar\_notes.html

https://stackoverflow.com/questions/18855850/create-a-tar-xz-in-one-command

## Grading Rubric :

All programs compile: 10%

Correctly shell: 20%

Correct serial downloader: 20%

Correct parallel downloader: 20%

Execution times for both downloaders: 10%

README file: True/False

Answers to extra questions: 20%

Bonus: Great! Let’s see how good your presentation is!

Late submissions shall be penalized 10%. No assignments shall be accepted after 24 hours.

## Grading Criteria

Your program will be evaluated according to the following criteria:

1. Correctness – The program must work as described by this document. Correctness also applies to the management of operating system resources. Resources utilized by your program should be properly released after the program ends.
2. Error reporting – Implementation of error checking and reporting is required. It is acceptable to report errors to the terminal. Error handling should properly cleanup allocated operating system resources.
3. Coding style – The program code should be neat, consistently formatted, and well commented
4. Target platform – Titan server

## 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 specied 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.

url.txt content:

https://www.dropbox.com/s/9frc9x5blloo7a4/CSUF\_Ubuntu\_Mate\_16\_04\_01\_amd64\_20170122.ova?dl=0

http://delaunay.ecs.fullerton.edu/~mshafae/srl/CSUF\_Ubuntu\_Mate\_16\_04\_01\_i386\_20170122.ova

https://www.dropbox.com/s/05k878x9t93k6wn/CSUF\_Ubuntu\_Mate\_16\_04\_01\_i386\_20170122.ova?dl=0

http://delaunay.ecs.fullerton.edu/~mshafae/srl/CSUF\_Ubuntu\_Mate\_16\_04\_01\_amd64\_20170122.ova