

Programming Assignment 4

Due Nov 4 by 11:59pm **Points** 100

Overview

The ability for one process to spawn additional processes is a powerful mechanism that can be used to accomplish a variety of tasks. This includes processes that work together and communicate to accomplish one larger task. It also includes using multiple processes to accomplish many smaller tasks in parallel.

For this assignment you will write a program that will use multiple processes to download a collection of files. These processes do not need to communicate to accomplish this task; the goal is to increase efficiency. In particular, if a single process attempts to download a collection of files in serial, then downloading from a slow server will delay all remaining downloads. Instead, when done in parallel, a process connected to a slow server can wait without affecting the other processes.

Course Learning Objectives

This assignment addresses the following course learning objectives.

Direct

- Read and write complex programs in the C programming language.
- Transition to upper-division coursework where programming is a tool used in the exploration of course-specific content.
- Transition from following stepwise, prescribed decompositions to independent problem solving.
- Discuss the architecture of the UNIX operating system from a system-programmer's perspective and be able to write programs that use operating system services (system calls) directly.
- Distinguish language features from operating system features.
- Gain experience with low-level programming in the UNIX environment.
- Consider basic resource management in the development of programs.

Indirect

- Use standard UNIX user-level commands and the UNIX software development environment.

Setup


Clone the repository.

- Accept the [GitHub Classroom Assignment 4 invitation](https://classroom.github.com/a/ShJtXBYF) 
(<https://classroom.github.com/a/ShJtXBYF>).

Requirements

There is, in effect, only a single task, so the following outlines requirements for your program.

Functional Overview

Your program will process a file containing the [URLs](https://en.wikipedia.org/wiki/URL)  (<https://en.wikipedia.org/wiki/URL>) for the files to be downloaded (an example file is provided in the assignment repository). The actual downloading will be handled by the *curl* program; this program is available on the department servers and can be installed in your development environment.

curl

The *curl* program is a command-line tool that can be used to download files specified by URLs. This program allows for configuration using a truly incredible number of command-line options, but we will not consider the vast majority of these options. Instead, your program will use *curl* in the following manner (with the options described below).

```
curl -m <seconds> -o <filename> -s <url>
```

- *-m <seconds>* — This option limits the download time to the specified number of seconds.
- *-o <filename>* — This option specifies the name of the output file to which the contents of the download file will be written.
- *-s* — This option silences the progress meter typically printed by *curl*.
- *<url>* — This is the url of the file to download.

Requirement 1

Your program must take two command-line arguments. The first is the name of the file (discussed below) containing the downloads to be processed. The second is an integer indicating the *maximum* number of downloading processes that may be executing at one time (this is so that the user can control the load from this program).

Your program must verify that the arguments represent a valid file and a valid positive integer.

Requirement 2

Each line of the provided file will contain the information relevant to a file to download. This information will consist of the desired output file name, the url, and an optional maximum number of seconds for the download. These values will be separated by a space character.

Requirement 3

Your program must spawn a single process per download request, but only up to the (command-line argument) specified maximum number of (child) processes that may exist at one time. Each of these processes will execute the appropriate *curl* command with the specified output file, url, and, if appropriate, maximum time.

If there are more download requests than the maximum number of allowed processes, then additional requests must be satisfied as each process completes. It is not sufficient to spawn only a single process at a time; multiple processes (up to the specified limit) must be used.

Requirement 4

The primary (parent) process should print when a download request is starting and when it ends. The output (when starting and when ending) must refer to each download request by the line number in the provided file and the child process id (e.g., `process 3287 processing line #1`). If one of the *curl* processes terminates abnormally, then the primary process must print an error message indicating such.

Requirement 5

Your program must properly manage resources to avoid running out of processes.

Deliverables

- Source Code (and Makefile) - Push all relevant source code and an appropriate Makefile to your repository.

Assignment Rubric

Criteria	Ratings		Pts
Functionality	70 pts Full Marks	0 pts No Marks	70 pts
Code Review	10 pts Full Marks	0 pts No Marks	10 pts
Valgrind Check	10 pts Full Marks	0 pts No Marks	10 pts
Resource Management	10 pts Full Marks	0 pts No Marks	10 pts
			Total Points: 100