**CS 375 Unix/Linux Programming**

**Fall, 2021**

**Programming Project #3**

**Due: Wednesday, December 1, 2021**

Write three programs that communic

ate using pipes and System V IPC methods. The three program files should be named **master.cpp**, **reverse.cpp** and **upper.cpp**. One message queue, one shared memory segment, two semaphores, and a pipe are required. The master program should prompt the user for a line of input. The master program should pass the user input to the reverse program via a message queue. The reverse program should reverse the line of input and place it in a shared memory segment for further processing by the upper program. The upper program should read the text from the shared memory segment then convert it to upper case and then send it to the master program via a pipe that has been dup’d over stdout. (I.e., as far as the upper program is concerned, it should just write its results to stdout.) The master program should then print out the result to the screen.

The message queue provides the necessary synchronization mechanism between the master and reverse programs. Use a pair of semaphores to provide for exclusive access (and synchronization) to the data shared between the reverse and upper programs. One should be set to 1 (call it the empty semaphore) and one should be set to 0 (call it the full semaphore). The reverse process should acquire the empty semaphore before accessing the shared memory and release the full semaphore after accessing the shared memory. The upper process should acquire the full semaphore before accessing the shared memory and release the empty semaphore after accessing the shared memory. The pipe provides the necessary synchronization mechanism between the upper and master programs. After reception of the result on the pipe, the master program should prompt the user for another line of input. You may assume that the longest possible line of user input is 256 characters.

The master program should create the IPC objects (message queue, shared memory segment, semaphores, and pipe). The master program also is responsible for forking and exec'ing the reverse and upper programs. The reverse program should take the message queue, shared memory segment, and semaphore ids as its arguments (four total). The upper program should accept the shared memory segment and semaphore ids as its arguments (three total), but make sure the pipe is dup’d before it is exec’d. Reminder: all of the program arguments must be C-strings.

When the master program reads end-of-file on standard input (user types Ctrl-d), it should send a shutdown message (via the message queue) to the reverse program. Before the reverse program exits and it should send a shutdown message (via the shared memory segment) to the upper program. The master program should wait for both the reverse and upper programs to exit, before cleaning up and exiting. No IPC objects should remain after the master program exits.

Example session is shown below (user input is shown in **bold**):

$ **./master**

> **Hello there!**

!EREHT OLLEH

> **How are you?**

?UOY ERA WHO

> **^D**

$

Note: Each invocation of the master program should fork and exec the two children, the reverse and upper programs, only once. Do not fork and exec the programs for each line of input.

**What to submit**

* Provide a makefile named **Makefile** that will make all three programs for this assignment as the default target (typically called **all**). Each program must be a separate target.
* Create a tarfile or zipfile containing your well-commented program source files and makefile.
* Submit your archive to Blackboard.