Hot Potato With 5 Child Processes

1. Problem Statement

The assigned task was to have a parent process create five child processes that will play hot potato with a message placed on the POSIX message queue by the parent process. The first child process to receive the message 1000 times sends a signal to the parent process that it has lost. The parent will then send a signal to all child processes and the child processes will have to respond with their process ID and current count, and then their count is reset to zero. The remaining child processes will continue the game until only one child process is left. The final child process is declared the winner and the game is over.

1. Approach

I used the CLion IDE to develop my code and test my code. I also ran my code on UT Dallas cslinux1 terminal. I approached this problem by using a for loop to create 5 child processes. In the for loop I used an if statement that holds a while loop that waits for a message to appear in the message queue. Once a message is received the counter is incremented and the message is sent back to the queue. This task is repeated until one process’s counter reaches 1000, at which point the process sends a signal to the parent process which will then send a signal to all child process requiring the child processes to print their ID and current counter value, and then set their counter to zero. The signals are handled by the signal handler, signal handler 1 will handle the signal of the parent and the signal handler 2 will handle the signal of the child. Signal handler 1 will decrement the status variable every time a signal is received from a child. When the last signal is received from the child process the parent will exit its loop and print the winner.

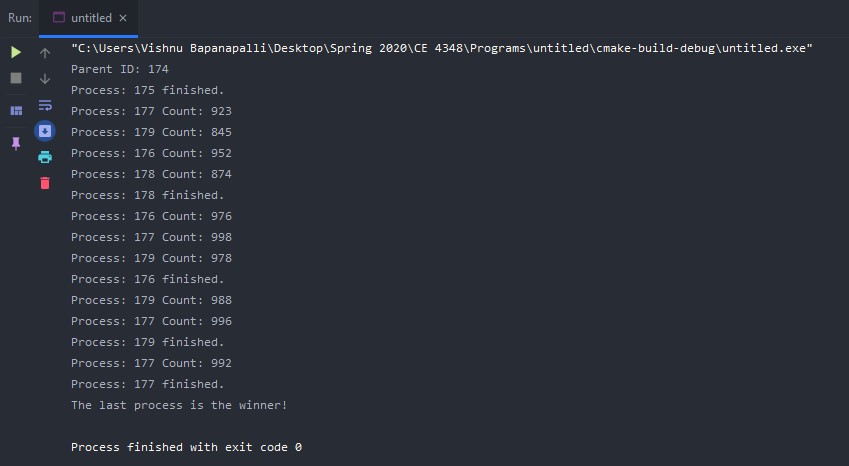
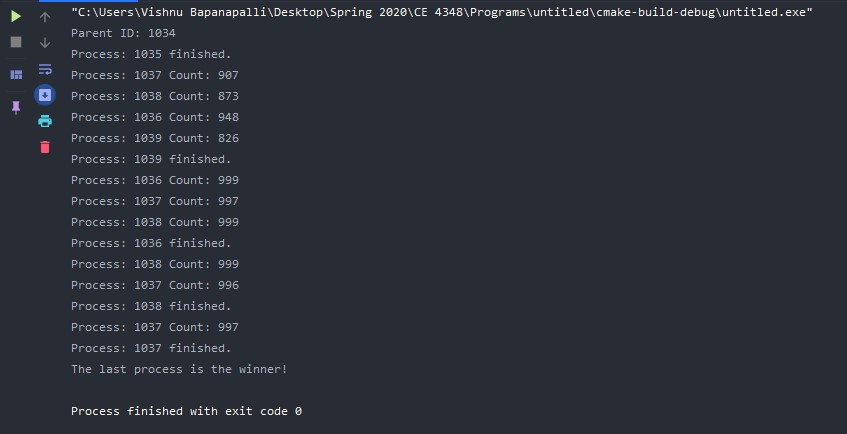
1. Solution

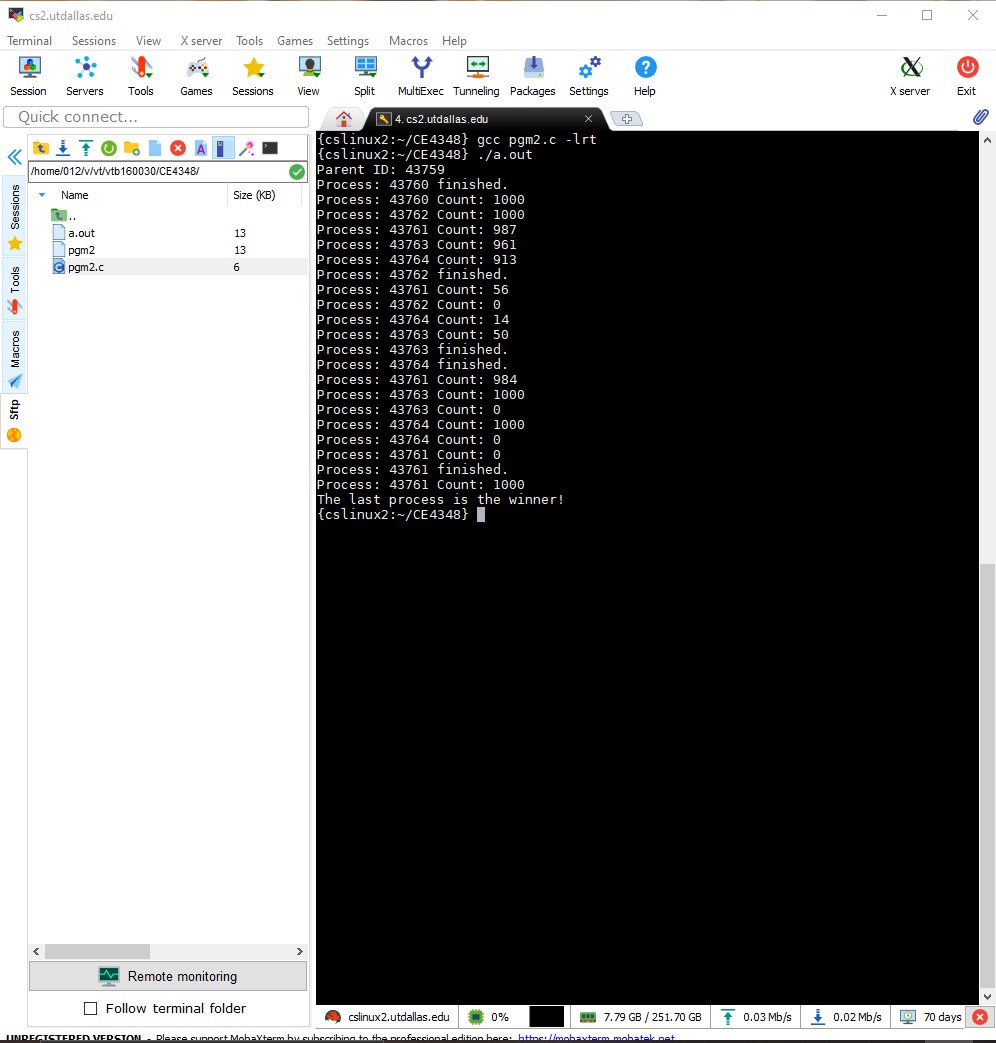
To run this code on the Linux terminal please use the following command:

**$ gcc pgm2.c -lrt**

**$./a.out**

This will execute the code and display the child processes in order of completion. After the parent ID is printed the first process to finish will print its process id and state its finished. The remaining processes will print their ID and current counter status. This will repeat until only one process remains, at this point the final process displayed is the winner. Its important to note that each process receives the number of messages it receives because the OS determines which process is first readily available to receive the next message.

The above images are examples of program output, when the first process finishes the remaining processes will print their ID and counter status. As each process finishes the remaining processes decrement, as seen in the example first there are four remaining members, then three remaining members, then two remaining members and finally one remaining member. Then the last process is declared the winner.



In this screenshot I am using the cslinux2 terminal to test my code. The code runs similarly to my test on my personal PC. In this child processes are finishing faster and sending signals to the parent than child processes receiving signals which causes the count to be interrupted by the second signal it receives. Despite this minor inconvenience the final child process to finish is still being determined as the winner.