CSE344 – System Programming - Homework 3
IPC with fifos and shared memory

Ozan GECKIN - 1801042103

April 23, 2021

1 Problem

In our problem ,You'll have N ¿ 1 processes, that will play the hot potato game. Some processes will start with a hot potato, and in order not to burn their hands, will immediately give it to another process, and the potato will keep changing processes, and eventually cool down after a predefined number of switches and stop switching.

2 Solution

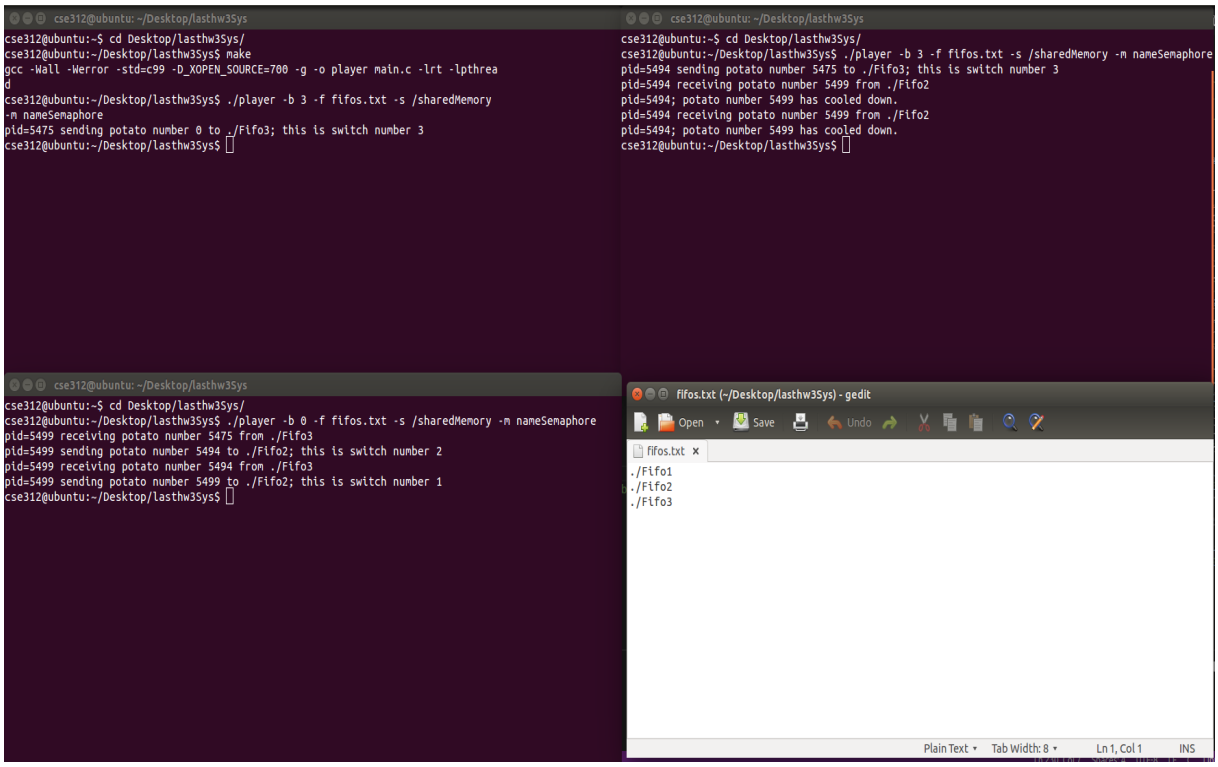
In this game, I used semaphores as a solution to synchronization problems. I generated the processes with fifos and created a share memory that they all have access to. Semaphores have done the trick for syncronization issues in this shared memory. I used getopt () to get arguments from the terminal and test them.

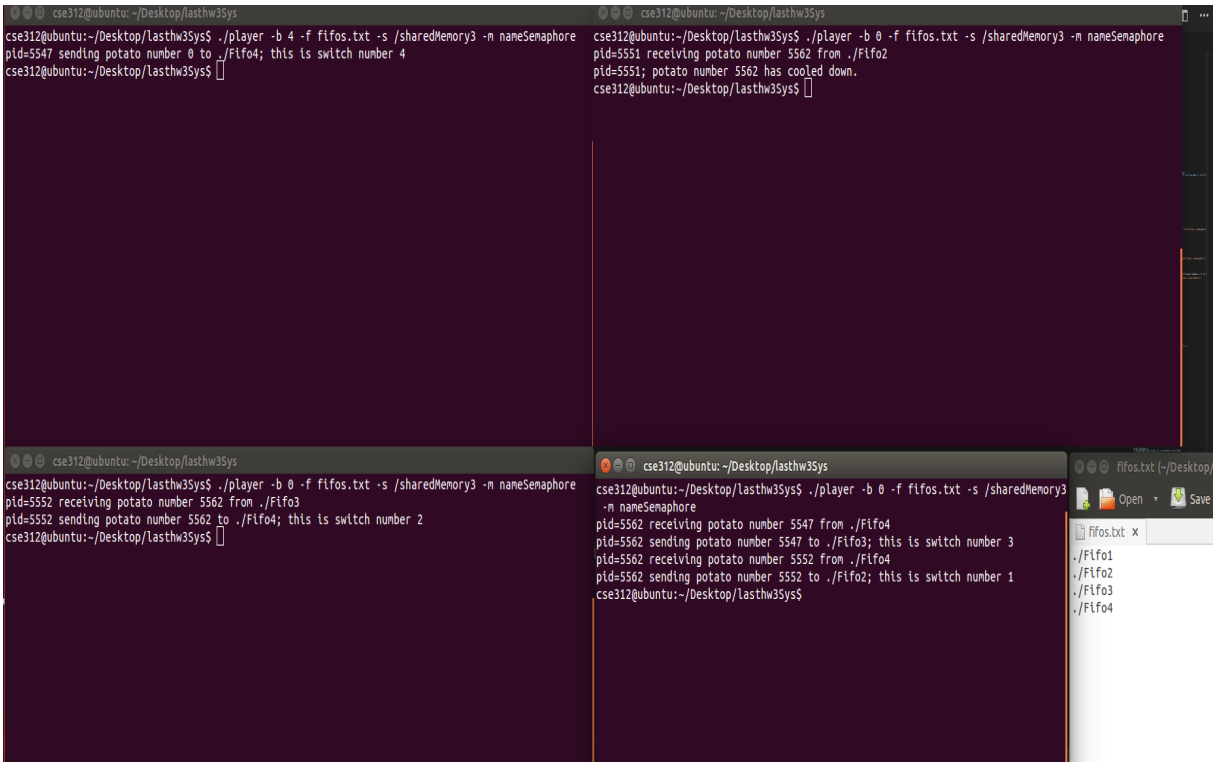
3 Algorithm

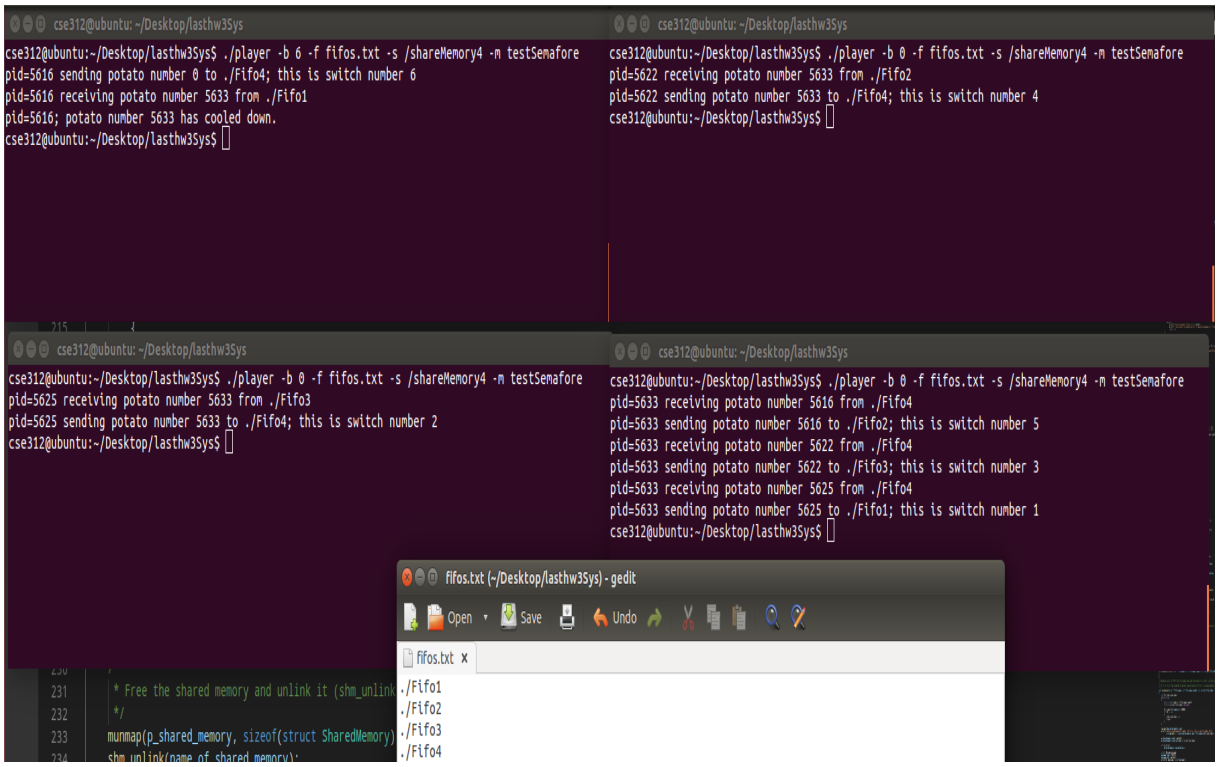
First, I used getopt to read the specified arguments from the user. -b was checked for correct number used strtol. Invalid argument shows error and quits the program.Then, all arguments are checked for validity because getopt can give error on invalid arguments but not for not enough arguments.Then, the FIFO file is opened. For every line, we reallocate a pointer to get one more line, and that read line is added to end of it.Now we open the shared memory. It involves few steps. First, we try to create it along with OEXCL argument because OEXCL gives error if the shared memory is already made. If the shared memory is already opened, we just get the mmap pointer and return it, else we set its length, get the mmap pointer and initialize all necessary variables in it. Now from that list of FIFOs, I make all FIFO with mkfifo (this fails if it is already created, but passes if otherwise, no need for check for error), I open first FIFO that is not opened already. I used a string to which I concatenate

name of FIFO, I just opened which is later checked to ensure, it is not opened already.If we have to send potato, I open a random FIFO from the list, waited on shared semaphore and sent the potato to it and increase number of potato counter in shared memory segment (this is done on just this step).Now in while loop, I wait for a message on my own FIFO, then I wait for shared semaphore, lower the cooldown timer and send the potato again to a random FIFO. If the cooldown timer hits 0, I decrease the number of potato counter in the shared memory segment, which if hits 0, I send exit message to all FIFOs and we are done.

4 Test




cse312@ubuntu: ~/Desktop/lasthw3Sys
cse312@ubuntu:~/Desktop/lasthw3Sys$ ./player -b 4 -f fifos.txt -s /sharedMemory3 -m nameSemaphore
pid=5547 sending potato number 0 to ./Fifo4; this is switch number 4
cse312@ubuntu:~/Desktop/lasthw3Sys$
cse312@ubuntu: ~/Desktop/lasthw3Sys
cse312@ubuntu:~/Desktop/lasthw3Sys$ ./player -b 0 -f fifos.txt -s /sharedMemory3 -m nameSemaphore
pid=5551 receiving potato number 5562 from ./Fifo2
pid=5551; potato number 5562 has cooled down.
cse312@ubuntu:~/Desktop/lasthw3Sys$
cse312@ubuntu: ~/Desktop/lasthw3Sys
cse312@ubuntu:~/Desktop/lasthw3Sys$ ./player -b 0 -f fifos.txt -s /sharedMemory3 -m nameSemaphore
pid=5552 receiving potato number 5562 from ./Fifo3
pid=5552 sending potato number 5562 to ./Fifo4; this is switch number 2
cse312@ubuntu:~/Desktop/lasthw3Sys$
cse312@ubuntu: ~/Desktop/lasthw3Sys
cse312@ubuntu:~/Desktop/lasthw3Sys$ ./player -b 0 -f fifos.txt -s /sharedMemory3
-m nameSemaphore
pid=5562 receiving potato number 5547 from ./Fifo4
pid=5562 sending potato number 5547 to ./Fifo3; this is switch number 3
pid=5562 receiving potato number 5552 from ./Fifo4
pid=5562 sending potato number 5552 to ./Fifo2; this is switch number 1
cse312@ubuntu:~/Desktop/lasthw3Sys$
fifos.txt (~/Desktop/
Open
Save
fifos.txt
./Fifo1
./Fifo2
./Fifo3
./Fifo4


cse312@ubuntu: ~/Desktop/lasthw3Sys
cse312@ubuntu:~/Desktop/lasthw3Sys$ ./player -b 6 -f fifos.txt -s /shareMemory4 -m testSemafore
pid=5616 sending potato number 0 to ./Fifo4; this is switch number 6
pid=5616 receiving potato number 5633 from ./Fifo1
pid=5616; potato number 5633 has cooled down.
cse312@ubuntu:~/Desktop/lasthw3Sys$
cse312@ubuntu: ~/Desktop/lasthw3Sys
cse312@ubuntu:~/Desktop/lasthw3Sys$ ./player -b 0 -f fifos.txt -s /shareMemory4 -m testSemafore
pid=5622 receiving potato number 5633 from ./Fifo2
pid=5622 sending potato number 5633 to ./Fifo4; this is switch number 4
cse312@ubuntu:~/Desktop/lasthw3Sys$
cse312@ubuntu: ~/Desktop/lasthw3Sys
cse312@ubuntu:~/Desktop/lasthw3Sys$ ./player -b 0 -f fifos.txt -s /shareMemory4 -m testSemafore
pid=5625 receiving potato number 5633 from ./Fifo3
pid=5625 sending potato number 5633 to ./Fifo4; this is switch number 2
cse312@ubuntu:~/Desktop/lasthw3Sys$
cse312@ubuntu: ~/Desktop/lasthw3Sys
cse312@ubuntu:~/Desktop/lasthw3Sys$ ./player -b 0 -f fifos.txt -s /shareMemory4 -m testSemafore
pid=5633 receiving potato number 5616 from ./Fifo4
pid=5633 sending potato number 5616 to ./Fifo2; this is switch number 5
pid=5633 receiving potato number 5622 from ./Fifo4
pid=5633 sending potato number 5622 to ./Fifo3; this is switch number 3
pid=5633 receiving potato number 5625 from ./Fifo4
pid=5633 sending potato number 5625 to ./Fifo1; this is switch number 1
cse312@ubuntu:~/Desktop/lasthw3Sys$
fifos.txt (~/Desktop/lasthw3Sys) - gedit
Open
Save
Undo
fifos.txt
./Fifo1
./Fifo2
./Fifo3
./Fifo4
* Free the shared memory and unlink it (shm_unlin
*/
munmap(p_shared_memory, sizeof(struct SharedMemory)
shm_unlink(name_of_shared_memory);