Gebze Technical University
CSE344 Systems Programming Course
-
HOMEWORK 3 REPORT

Muhammed OZKAN 151044084

April 23, 2021

Abstract

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.

1 Overview

The homework I submitted is a complete assignment. It supports all the requirements and parameters requested in the homework document and opens the file with the given path. Each process reads the given parameters. It binds the shared memory space to itself according to the name specified in the parameters. Creates semaphorees that will be used for synchronization in program flow and then it holds itself to wait for(barrier) the other fifoes in the fifo list to be received by the processes. The barrier is opened when all fifoes in the list are assigned by a process, and those with potatoes will randomly send them to a process. And it writes the required switch number message on the screen. Those who do not have potatoes wait behind the barrier for others to send. When potatoes come from a process, it writes the receipt information on the screen and checks the cooling number. If the incoming potato has cooled, it will display this and another potato await. If it is not cooled, sends it to another process, as above and writes it to the screen.

2 How did I solve this problem.

First of all, the program binds the shared memory space according to the parameter given to it. It then creates the required semaphorees. He chooses his fifos from the file and starts listening without blocks. In this section, each process creates a semaphore with the name of own fifo. It will use it to prevent busy waiting or continuous reading.

After the preparation and creation stages are finished, if the process starts with potatoes, the first thing to do will be send another process. It will select a random fifo from the file. It will be opened by selected fifo write function. Switch number is updated increased value in the shared memory space, the value is written on the screen and sent to the other process with fifo.

The reading function of the program are done as follows. The arrival information of each incoming potato is immediately written to the screen and the values in the shared memory are updated. It is checked whether the cooling number reaches the desired value or not. otherwise, it is sent to another process as described in the paragraph above. If the desired cooling number has been reached, the relevant message is writed on the screen and the process is finished with that potato. If there are other potatoes that need to be cooled in the system, it is switched to standby mode again. If there are no more potatoes to cool, it sends the termination message to all processes to terminate itself and other processes. Processes that receive this message close and clean all resources and common structures such as fifo, semaphore, etc. Terminates the programs.

3 My design decisions.

Of the given parameters, the -s name was used as shared memory, as well as to create a semaphore to protect it. Thus, data integrity was achieved by taking this semaphore for those who want to access the common area and read and write data. A struct structure was saved in the shared memory area, thus making data access easier. The name given in the -m parameter was used as a barrier semaphore. It was used for the task of holding all processes at the same point and starting with the last incoming process. Apart from these structures, each fifo is opened as a non block, so they created a semaphore in their name. The reason for this is that when the data arrives, the process that writes it open the waiting end of the sent fifo using this semaphore, preventing busywaiting.

Since semaphores, files and fifoes are opened and closed many times, after 500 600 times they give an error "Too many open files", common resources were opened at the beginning of the program and used in this way.

4 What requirements did I meet and which did I fail?

All requirements are fully met. All processes created at the end of the program are terminated and resources are returned to the system.

5 Tests

pithblood@Monster:$ ./player -b 5 -s shmemory -f filewithfifonames -m kilitle
PID=10715 Waiting Other Processes
pid=10715 sending potato number 10715 to fifo4 this is switch number 1
pid=10715 receiving potato number 10717 from fifo1
pid=10715 sending potato number 10717 to fifo3 this is switch number 4
pid=10715 receiving potato number 10715 from fifo1
pid=10715 sending potato number 10715 to fifo4 this is switch number 4
pid=10715 receiving potato number 10716 from fifo1
pid=10715 potato number 10716 has cooled down.

pithblood@Monster:$ ./player -b 4 -s shmemory -f filewithfifonames -m kilitle
PID=10716 Waiting Other Processes
pid=10716 sending potato number 10716 to fifo3 this is switch number 1
pid=10716 receiving potato number 10716 from fifo2
pid=10716 sending potato number 10716 to fifo1 this is switch number 4
pid=10716 receiving potato number 10715 from fifo2
pid=10716 potato number 10715 has cooled down.

pithblood@Monster:$ ./player -b 5 -s shmemory -f filewithfifonames -m kilitle
PID=10717 Waiting Other Processes
pid=10717 sending potato number 10717 to fifo4 this is switch number 1
pid=10717 receiving potato number 10717 from fifo3
pid=10717 sending potato number 10717 to fifo1 this is switch number 3
pid=10717 receiving potato number 10716 from fifo3
pid=10717 sending potato number 10716 to fifo4 this is switch number 2
pid=10717 receiving potato number 10715 from fifo3
pid=10717 sending potato number 10715 to fifo1 this is switch number 3
pid=10717 receiving potato number 10717 from fifo3
pid=10717 sending potato number 10717 to fifo4 this is switch number 5

pithblood@Monster:$ ./player -b 0 -s shmemory -f filewithfifonames -m kili-
tle
PID=10718 Waiting Other Processes
pid=10718 receiving potato number 10717 from fifo4
pid=10718 sending potato number 10717 to fifo3 this is switch number 2
pid=10718 receiving potato number 10716 from fifo4
pid=10718 sending potato number 10716 to fifo2 this is switch number 3
pid=10718 receiving potato number 10715 from fifo4
pid=10718 sending potato number 10715 to fifo3 this is switch number 2
pid=10718 receiving potato number 10715 from fifo4
pid=10718 sending potato number 10715 to fifo2 this is switch number 5
pid=10718 receiving potato number 10717 from fifo4
pid=10718 potato number 10717 has cooled down.

-filewithfifonames
/tmp/fifo1
/tmp/fifo2
/tmp/fifo3
/tmp/fifo4