Hacettepe University

Computer Science and Engineering Department

BIL342 Experiment-1

Subject	Interprocess Communication
Subject	Processes and Pipes
Programming Language	ANSI C
Date Due	20/03/2017
Advisors	Asst. Prof. Dr. Ahmet Burak CAN Asst. Prof. Dr. Kayhan İMRE Assoc. Prof. Dr. Harun ARTUNER Dr. Ali Seydi KEÇELİ Dr. Aydın KAYA

Introduction

To hide the effects of interrupts, operating systems provide a conceptual model consisting of sequential processes running in parallel. Processes can be created and terminated dynamically. Each process has its own address space. Processes can communicate with each other using interprocess communication primitives, such as semaphores, monitor, pipes or messages. These primitives are used to ensure that no two processes are ever in their critical regions at the same time, a situation that leads to chaos. A process can be running, runnable (ready), or blocked and can change state when it or another process executes one of the interprocess communication primitives.

A pipe is a form of redirection that is used in Linux and other Unix-like operating systems to send the output of one program to another program for further processing.

Experiment

The aim of this experiment is to make you familiar with some concepts of processes and interprocess communication and have practical experience on interprocess communication programming using UNIX and POSIX facilities.

You will implement a multi process matrix multiplication application that communicates via pipes. The application will fork itself n times after it is started. $n \leq 10$ will be given as a command line argument to the program. The child processes will communicate with pipes, which are defined before fork() operations. Each process will process the data provided from the previous child process and pass the result to the next child process.

The main process reads an input matrix from the file named `matrix.txt`. There will be only one square matrix in the input file, where the first line gives the size (number of columns/rows) of the square matrix. Matrix columns are delimited with commas. Each line corresponds to a row of the matrix. The format of the `matrix.txt` is given below.

```
4
3,8,12,34
45,6,7,9
5,6,7,8
1,5,6,76
```

Each child process will get an input matrix from previous child process and compute the square of the matrix. For example: if process count is given as 3, the first child process will get a matrix m from the parent process and compute $m.m=m^2$. The second child process will get m^2 from the first one and compute $m^2.m^2 =m^4$. Then, the nth process will compute m^{2n}. Each process will read the input from the previous child process and send the result matrix to the next process via a pipe. Then the last child process sends the result to the parent process via a pipe and the result is displayed to the screen. A visualization of the pipes and process are show in the Figure below. The result computed in each process should be printed to standard output and output files named <process_number>.txt with the process id. Each process will print its output to a separate file. Sample output format is given below:

1.txt

```
Process-1 <id>

463   314   380   2854
449   483   685   2324
328   158   199   888
334   454   545   5903
```

2.txt

```
Process-2 <id>

1433231   1652800   2022080   19235740
1425650   1537601   1904370   16730790
584670    613900    756431    6721880
2508850   3090230   3763500   37337701
```

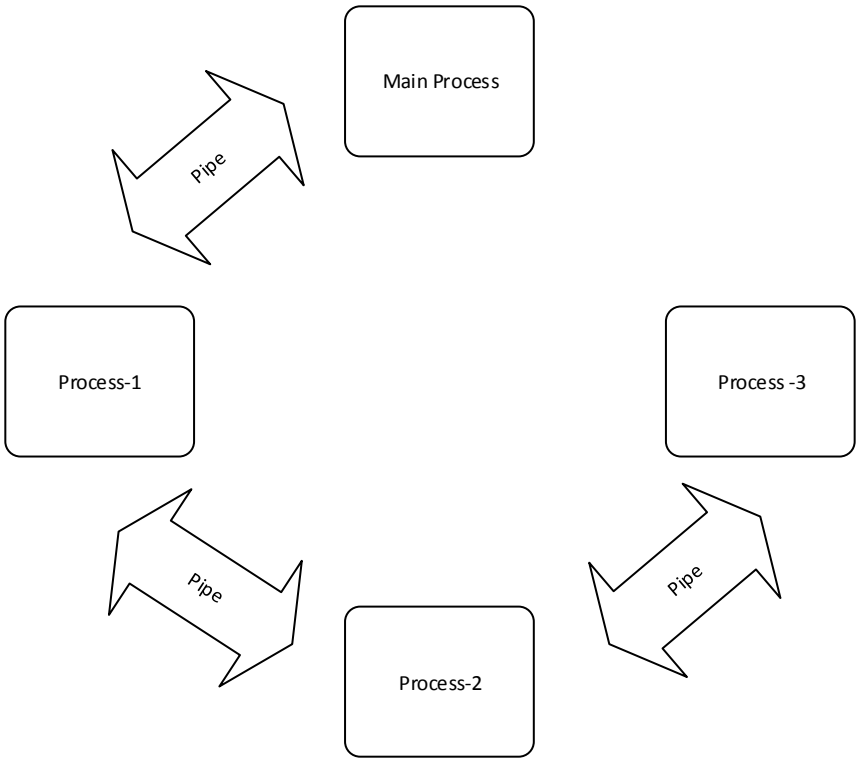

Figure-1 Process and pipe structure of the program.

Notes

- Before submission, you have to compile and test your code on dev.cs.hacettepe.edu.tr machine. Your programs will be tested on this machine, so make sure that your code runs on this machine properly.
- Save all your work until the experiment is graded.
- The assignment must be original, INDIVIDUAL work. Shared source codes will be considered as cheating. The students who share their works will be punished in the same way.
- You can ask your questions via piazza group.
- You are expected to follow the news group and you will be responsible for the announcements made there.

References

[1] Unix Multi-Process Programming and Inter-Process Communications (IPC), http://users.actcom.co.il/~choo/lupg/tutorials/multiprocess/multi-process.html.

[2] How to Use a Pipe, https://www.cs.rutgers.edu/~pxk/416/notes/c-tutorials/pipe.html.