Assignment 3

All programs are to be written in C

1. The Fibonacci sequence is the series of numbers 0, 1, 1, 2, 3, 5, 8, ... where the nth term is the sum of the two previous terms. Write a program using the fork() system call that generates the Fibonacci sequence in the child process. The number of the sequence will be provided in the command line. For example, if 5 is provided, the first five numbers in the Fibonacci sequence will be output by the child process. Because the parent and child processes have their own copies of the data it will be necessary for the child to output the sequence. Perform necessary error checking to ensure that a non-negative number is passed on the command line.

2. In Question 1, the child process must output the Fibonacci sequence, since the parent and child have their own copies of the data. Another approach to designing this program is to establish a shared memory segment between the parent and child processes. This technique allows the child to write the contents of the Fibonacci sequence to the sharedmemory segment and has the parent output the sequence when the child completes. Because the memory is shared, any changes the child makes will be reflected in the parent process as well.

This program will be structured using POSIX shared memory. The program first requires creating the data structure for the shared-memory segment. This is most easily accomplished using a struct. This data structure will contain two items: (1) a fixed-sized array of size MAX_SEQUENCE that will hold the Fibonacci values and (2) the size of the sequence the child process is to generate sequence_size, where sequence_size <= MAX_SEQUENCE. These items can be represented in a struct as follows:

```
#define MAX_SEQUENCE 10
typedef struct {
  long fib_sequence[MAX_SEQUENCE];
  int sequence_size;
} shared_data;
```

The parent process will progress thmugh the following steps:
a. Accept the parameter passed on the command line and perform
error checking to ensure that the parameter is <= MAX_SEQUENCE.
b. Create a shared-memory segment of size shared_data.
c. Attach the shared-memory segment to its address space.
d. Set the value of sequence_size to the parameter on the command line.
e. Fork the child process and invoke the wait() system call to wait for the child to finish.
f. Output the value of the Fibonacci sequence in the shared-memory segment.
g. Detach and remove the shared-memory segment.

Because the child process is a copy of the parent, the shared-memory region will be attached to the child's address space as well as the parent's. The child process will then write the Fibonacci sequence to shared memory and finally will detach the segment. One issue of concern with cooperating processes involves synchronization issues. In this exercise, the parent and child processes must be synchronized so that the parent does not output the Fibonacci sequence until the child finishes generating the sequence. Think of a way to synchronize the processes without requiring explicit primitives such as Semaphores, etc. [Hint. Refer to Lab Slides]

3. Design a program using ordinary pipes in which one process sends a string message to a second process, and the second process reverses the case of each character in the message and sends it back to the first process. For example, if the first process sends the message Hi There, the second process will return hi tHERE.

4. Design a file-copying program named FileCopy using ordinary pipes. This program will be passed two parameters: the first is the name of the file to be copied, and the second is the name of the copied file. The program will then create an ordinary pipe and write the contents of the file to be copied to the pipe. The child process will read this file from the pipe and write it to the destination file. For example, if we invoke the program as follows:

FileCopy input.txt copy.txt

the file input. txt will be written to the pipe. The child process will read the contents of this file and write it to the destination file copy. Txt.

5. Write a program that is passed an identifier for a shared-memory segment. This program will invoke the shmctl() function. It will then output the following values of the shared-memory segment:

- Segment ID
- Key
- Mode
- Owner UID
- Size
- Number of attaches

The relevant commands and data structures for this exerceise are ipcs, struct shmid_ds.

Submit your solutions in a single zip file to operatingsystemisi@gmail.com with the subject line as "MTechRollNumber_Assignment3" or "Name_Assignment3" for JRFs by Sunday 04 April, 2020.