CSCI 604 2020

Project One.

Part I.

a) The Fibonacci sequence is the series of numbers 0, 1, 1, 2, 3, 5, 8 … Formally, it can be expressed as:

$fib_0 = 0$
$fib_1 = 1$
$fib_n = fib_{n-1} + fib_{n-2}$

Write a `C` 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 process to output the sequence. Have the parent invoke the `wait( )` call to wait for the child process to complete before exiting the program. (Perform necessary error checking if you can, to ensure that a non-negative number is passed on the command line.)

b) In Part a), the child process must output the Fibonacci sequence, since the parent and child have their own copies of data. Another approach to designing this program is to establish a shared memory object between the parent and child processes. This technique allows the child to write the contents of the Fibonacci sequence to the shared-memory object and has the parent output the sequence when the child completes. Because the memory is shared, any changes the child makes to the shared memory will be reflected in the parent process as well. The program will be structured using POSIX shared memory IPC as discussed in the class. The program first requires creating the data structure for the shared memory segment. (This is most easily accomplished using `struct in C and C++`.)

This data structure will contain two items:
(1) a fixed size 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 100
typedef struct{
int fib_sequence[MAX_SEQUENCE];
int sequence_size;
}shared_data;
```

The parent process will progress through the following steps:
1) Accept the parameter passed on the command line and perform error checking to

ensure that the parameter is ≤`MAX_SEQUENCE`.

2) Establish the shared-memory object using `shm_open( ),ftruncate( ),and mmap( )`.

3) `fork( )` the child process and invoke `wait( )` to wait for the child to finish.

4) Output the value of the Fibonacci sequence in the shared-memory object.

5) Remove the shared-memory object.

Note: - To compile your C program on Linux, use

gcc –o your_target your_c_code.c

To compile your C++ program on Linux, use

g++ –o your_target your_cpp_code.cpp

- You may use a language different than C/C++ that provides support to Pthread. If that is the case, you still need to follow the above instructions and implement all the concepts in parallel with the language you chose. Note that many languages don't support memory allocation as described in b).