Colby Holmes

Cah0448

CSCE-4600 Operating Systems Design

Homework #2

Due 2-16-2017 by 11:59pm

You will be submitting all homework via Blackboard.

1. What is the difference between the fprintf() and the write() function calls, i.e., the following calls:

fprintf(stdout, %s, str);

write(1, str, strlen(str));

You may assume that str is a null-terminated string and that the file descriptor 1 is stdout. In particular, what does the OS do for each function?

Write is a system call and sends the info straight to the OS, while fprintf is a library call, which buffers and sends more uniform packages of data to the OS all at once.

1. Briefly describe the semantics of the following UNIX system calls:
   1. fork(); splits the current program into identical parent and child processes
   2. exit(); closes the current process
   3. getppid(); returns the parent process’s id
   4. wait(); waits until the system tells it otherwise (update message)
   5. exec(); executes the specified system call
2. For the following system calls, provide a reason (or condition) that causes it to fail:
   1. read(); no file open to read
   2. write(); no correct filepath open to write to
   3. fork(); not enough memory to allocate
   4. close(); the socket is being used by another process
   5. waitpid(); there are no child processes
3. Write a C program consisting of 3 processes, that is, one parent process that forks 2 child processes. Each process will print its process ID and its parent’s process ID. While the first process (P1) will wait for the remaining child processes (P2 and P3) to terminate, they will perform the following: process P2 will execute the ‘ps –ael’ command (using one of the exec() commands, not system()) and process P3 will execute the ‘ls’ command. This should be done using one of the exec() commands, not the system() call. The results of P2 and P3 will be written to separate files (i.e., P2\_file.txt and P3\_file.txt). After running your program, you will construct a process tree diagram (see Fig. 3.8 in the textbook) showing the hierarchy of processes all the way up to the init process for these processes in this program. This may require a review of Systems Programming. You may use resources from the Internet, but you must cite your sources. Deliverables for this problem will include your source code, the two files (P2\_file.txt and P3\_file.txt), and the process tree diagram.

Login

Init

Homework 2 Program

Child Process 2

Child Process 1

Bash

Bash

LS

PS

1. Consider a system of 9 processes, **P** = {p1, p2, …, p8, p9}. Associated with the system are 6 memory cells, **M** = {M1, M2, …, M5, M6}. The domain and range for each process is given in the following table:

|  |  |  |
| --- | --- | --- |
| Process pi | Domain D(pi) | Range R(pi) |
| p1 | M1, M2 | M3 |
| p2 | M1 | M5 |
| p3 | M3, M4 | M1 |
| p4 | M3, M4 | M5 |
| p5 | M3 | M4 |
| p6 | M4 | M4 |
| p7 | M5 | M5 |
| p8 | M3, M4 | M2 |
| p9 | M5, M6 | M6 |

In addition, you are given the following precedence relation:

🡺 = {(1,2),(1,6),(2,3),(2,4),(2,5),(3,6),(3,8),(4,6),(4,7),(5,7),(5,8),(6,8),(6,9),(7,9),(8,9)}

* 1. Construct the Precedence Graph (not containing redundant edges).
  2. Determine if the system above is determinate. If it is not, add to 🡺 necessary elements to make it determinate.