Homework 14: Operating Systems and Processes

Instructor: Mehmet Emre

CS 32 Spring '22

Due: 5/25 12:30pm
Name & Perm #: Bharat Kathi (5938444)

Homework buddy (leave blank if you worked alone):

Reading: Chapters 3, 4, and 5 of Operating Systems: Three Easy Pieces (you can download them from the web page, also you can skip the "Homework" sections). Also, the chapters from the UNIX textbook (available from GauchoSpace).

1 *2pts*

In a brief sentence, describe the difference between a process and a program, as defined by Arpaci-Dusseaus.

Answer: A program is simply a set of instructions and corresponding static data, whereas a process is an instance of a program in execution, containing the state of the program.

2 *3pts*

In class, we discussed three objectives of an operating system. List them below:

1. Abstracting `system calls`
2. Isolating `application memory`
3. Managing `system resources`

3 *3pts*

Arpaci-Dusseaus describe a process being in one of 3 states at any point. Describe each of these states **with your own words**:

1. Running: The process is currently executing instructions on a processor.
2. Ready: The process is ready to run, but the OS has not instructed the process to begin running yet.
3. Blocked: The process did something that is preventing it from continuing to execute, it is now waiting on a dependedent event to happen before if can continue execution.

4

The history of Unix as an OS (and then, the history of all Operating Systems, including those derived from Unix, such as Linux, Mac OS X, etc.) was heavily influenced by the choice of programming language in which Unix was first implemented.

1. (1 pts) What programming language was chosen for the first implementation of Unix?

   Answer: UNIX was originally written using assembly, but was soon after rewritten in C.

2. (1 pts) Most other OS implementations, up to that point in time, had been written in what language?

   Answer: Most other OS implementations were written in assembly at the time.

3. (2 pts) According to the authors, as a *direct result of these two facts* (your answers to (a) and (b)), Unix had a distinct advantage over other OSes at the time. What was this advantage?

   Answer: This meant that UNIX shipped with a C compiler by default. This not only made it easy for programmers to get in and start coding, but also made UNIX a very attractive option to build off of.

5

In the Unix system, the "system call interface" provides entry points into ... something.

1. (2 pts) In a single word or short phrase (no more than four words) name what the system call interface provides an entry point into.

   Answer: the UNIX kernel

2. (2 pts) Now, in a 2-3 sentences, describe what the phrase means. What is the "purpose" of the thing you named in part "a"?

   Answer: The UNIX kernel is the part of the UNIX operating system that is always resident in memory and is responsible for managing all the interactions between hardware and software. This includes managing resources such as I/O, memory, CPU, cache, file systems, and network sockets.

6

If you want to "suspend" a program that is running in a shell, in order to get to the shell prompt and do a few commands, then resume that program, what can you do?

1. (2 pts) What do you type to suspend the current running program?

   Answer: You can type Control+Z to suspend the currently running program.

2. (2 pts) What do you type to resume the suspended program?

   Answer: You can use the bg or fg commands followed by the suspended proccess ID to resume a suspended program in the background or foreground, respectively.

7

The command kill is used to, as the name would suggest, manually end processes. kill's primary function is to provide the user with control over processes that are running in the background and thus only accessible by a PID (process ID). kill features varying levels of ability to end process. Why is this fine-tuning useful? Answer the question by succinctly describing at least two different modes of kill and when each might be useful.

1. (2 pts)

   Answer: SIGTERM (15) is the default and safest way to kill a process.

2. (2 pts)

   Answer: SIGKILL (9) is more of a last-resort method to kill a procces as it does not save date or perform any system cleanup.

8

Suppose you have a hello.cpp file that you've compiled into a binary called ./hello. As you know, you can run this with the steps shown below:

```
-bash-4.2$ ./hello
Hello, World
-bash-4.2$
```

In the steps shown, under the hood, somehow something called "fork" and "exec" are involved in getting this program to actually start running. And there is some kind of "parent" and "child", somehow involved. For each of the following, describe its role, relating it to these concepts. Be **brief** but be **just specific enough** that the grader has NO DOUBT that you've read and understood the discussion of these concepts in the reading, and how each of them relates to running a ./hello program.

1. (2 pts) fork

2. (2 pts) exec

3. (2 pts) parent

4. (2 pts) child

9

The `ps` command and the `jobs` command both list processes. Write a few words that distinguishes between the ways that they are used, that is:

1. (2 pts) What is a circumstance where `ps` is more appropriate than `jobs`?

   Answer: ps is an external command which will tell you about all the proccesses running on a system. So if you want to know what procceses are running system-wide this is the command for that.

2. (2 pts) What is a circumstance where `jobs` is more appropriate than `ps`?

   Answer: jobs is a shell internal command that can only show the jobs that the current shell is managing. So if you want to access information that is internal to the current shell session, such as the job numbers, then this would be the command for that.