Homework 5 (due November 12th, midnight) - 20 points.

Misha Ward

Structures:

2.8) What is the main advantage of the layered approach to system design? What are the disadvantages of the layered approach?

The main advantage of the layered approach is the simplicity of the system and ease debugging. By breaking up the OS into smaller pieces, you can create a modular approach to the design of the OS. The main disadvantage would be that the operating system is usually less efficient than other methods.

Processes:

3.12) Including the initial parent process, how many processes are created by the program shown in Figure 3.32?

There should be in total 15 processes. The processes would be (2*4) – 1 = 15 (4 forks).

3.17) Using the program shown in Figure 3.35, explain what the output will be at lines X and Y.

Output: CHILD: 0 CHILD: -1 CHILD: -2 CHILD: -3 CHILD: -4 PARENT: 0 PARENT: 1 PARENT: 2 PARENT: 3 PARENT: 4
NOTE: Line X is the CHILD: -# and Line Y is the PARENT: #

The output of 3.35 is going to show the child's first as the pid will be set to 0 after the fork and enter the for loop printing line X. The for loop will print the numbers in the array but only negative due to the nums[i] *= -1 line. The parent process pid will then be read and thus printing line Y, with the parent showing the positive numbers.

Threads:

4.15) Consider the following code segment:…

a) For unique processes, there are either 5 or 6 processes created depending if the first process is counted. If it is, then there are 6 processes. The first process, **P1 (1)** uses the fork() and fork() after if statement which creates sub process 1 (**SP1**) (2) and **SP2 (3)**. SP1 goes through the if statement and encounters the inner fork() and spawns **SSP1 (4)** and also creates a thread. SP1 hits another fork() outside of the if statement spawning **SSP2 (5)**. SSP1 continues and creates a thread, then forks creating **SSSP1 (6)**.

b) To build off the answer for a), theoretically, each process is a thread so there could be up to 8 or 7 threads. But if you just look at threads created specifically in the code, then there are two threads created.

Homework 5 (due November 12th, midnight) - 20 points.

Misha Ward

4.17) The program shown in Figure 4.16 uses the Pthreads API. What would be the output from the program LINE C and LINE P?

The output for line C is CHILD: value = 5 while the output for line P is PARENT: value = 0.

Synchronization:

5.8) The structure of process of Pi (I == 0 or 1) is shown in Figure 5.21. The other process is Pj (j== 1 or 0). Prove that the algorithm satisfies all three requirements for the critical-section problem.

Mutual Exclusion:
To prove mutual exclusion, Pi can only enter the critical section only if flag[j] = false or turn == i. This pretty much means that if Pi got to the critical section first, that the flag turn will be set to j while the flag for I will be true preventing the other process from entering into the critical section until the Pi is completed.

Progress: Progress is achieved because no matter how many processes there are, there are always processes that get their turn since Pi for example would need to complete for Pj to enter the critical process.

Bounded Waiting: Bounded waiting occurs when Pi is processing, Pj is waiting and pretty much locked out from entering the critical section and vice versa. For the most part, the process should complete the critical section once and then the other process will be able to complete.

5.12) The Linux kernel has a policy that a process cannot hold a spinlock while attempting to acquire a semaphore. Explain why this policy is in place.

The reason for not holding a spinlock during a semaphore is because you might need to sleep when you wait for the semaphore which is not allowed to sleep when in a spin lock. If you have the sleep, you might just stay in spinlock forever and the computer cannot interrupt the process.

5.21) Servers can be designed to limit the number of open connections. For example, a server may wish to have only N socket connections at any point in time. As soon as N connections are made, the server will not accept another incoming connection until an existing connection is released. Explain how semaphores can be used by a server to limit the number of concurrent connections.

Semaphores could be used to limit the amount of connections for the server as a semaphore is pretty much a queue that monitors traffic. By using acquire() and release(),

Homework 5 (due November 12th, midnight) - 20 points.

Misha Ward

the semaphore would block traffic that was beyond the allowable connections and put those connections into sleep to wait for a connection that is terminated and the release() method is invoked.

Scheduling:

6.11) Discuss how the following pairs of scheduling criteria conflict in certain settings.

a) For CPU utilization, the biggest issue is when you have a scheduling algorithm that breaks up processes, if that occurs, you risk increasing the overheads for the computer which would lower performance. By doing this however, you increase the response time for processes.
b) For average turnaround time and maximum waiting time, scheduling algorithms can impact these metrics by executing the shortest task first. By using the shortest task first, the average process time is reduced as turnaround time is minimized. By doing this however, doing this algorithm could cause long processes to starve and increase their time to process.
c) For I/O device and CPU utilization, scheduling algorithms can impact these metrics as CPU utilization is maximized when long running CPU tasks process without context switches. Input and Output utilization is minimized when scheduling jobs you do not schedule jobs immediately so they wait for the context switching. To maximize, you would want to start I/O jobs as soon as possible and ensure the use of context switching.

6.24) Explain the differences in how much the following scheduling algorithms discriminate in favor of short processes:

a) FCFS - FCFS does discriminate for short processes as any process that is first will be ran. The first process could be very long and ran even through a shorter or more important process needs to be run.
b) RR - Round robin is an equal opportunity scheduler which does not discriminate assuming equal burst in CPU. This allows short processes to complete when their turn is up.
c) Multilevel feedback queries - This works very simulariy to RR by giving each process a time to complete, with longer and longer processing time after each “loop”. This scheduling algorithm discriminates towards short processes as they allow the short processes to complete given the CPU burst time.