Name: Amit Nijjar Pid: A11489111 a2nijjar@ucsd.edu email:

CSE 120 Homework #1

Spring 2016 (Kesden)

Answers to these problems were derived from information gathered from lecture, the book, and Wikipedia. I worked alone on this homework assignment.

1. What role does the hardware timer place in the operating systems ability to provide preemptive (vs cooperative) scheduling?

The hardware timer makes sure that various programs have time with the processor, regardless of if the program is set up for preemptive or cooperative scheduling. If there was no hardware timer, the program would hog the processor until the program was done. If all programs had cooperative scheduling, then the processor would be shared among all programs, but since this is not the case, the hardware timer ensures this happens.

1. Do exception handlers typically run in user mode or supervisor mode? Why?

Exception handlers typically run in supervisor mode because the exception handler is typically used during the processors software interrupt instructions. These instructions are done when the processor is in supervisor mode, and therefore the exception handlers are fun in supervisor mode as well.

1. What prevents users from writing assembly code to access other users’ files?

Processor modes restrict operations that could damage the system; they prevent programs from changing or accessing files from other programs or the computer. These cpu restrictions are made by the hardware DRM.

1. What causes the processor to transition back from supervisor mode to user mode?

Security threat causes the processor to transition back from supervisor mode to user mode. The processor will go to user mode to prevent other programs from damaging the computer. The processor does this by going to user mode to execute the program instead of executing the program from supervisor mode.

1. What information is contained within the process control block (PCB)?

The PCB contains data needed to manage a process. This includes the process identification data, state data, and control data.

1. What information is saved upon a context switch?

General registers and the program counter register will be saved

1. How many blocked queues exist within an operating system?

There is one blocked queue for each process that is blocked and is waiting for an event to occur, such as the resource it needs becoming available

1. If fork() clones a process, why does the clone return a different value than the original?

Fork creates a new identical process, to distinguish between the two of them, the return value of fork is different than the original process.

1. What does it mean for a process to be *reaped*? With what entity does the responsibility to reap a process lie?

After a process completes its execution, it goes into a zombie state. From there the zombie process has its exit status read and is them removed from the process table. It is the wait() call that reaps zombies.

1. What is the value of the *zombie* state?

The value of the zombie state is to provide exit status information. Parent processes are able to read the exit status of their zombie child processes.

1. How does *copy on write (COW)* improve system performance?

If multiple callers want to use the same resource, you can provide them all pointers to the same resource. From there, you only need to copy the resource when a caller wants to change it providing it a private copy to change. This is beneficial because if the caller never needs to change the resource, no copy is needed.

1. What happens to the other threads, if one of many threads within a process makes an exec call? Why?

The other threads are replaced with a new program. This is because the program is loaded into the process space and is run from the beginning.

1. Please explain two advantages of threads over user processes.

It is easier to program threads than it is processes

Context switches are faster between threads than between processes

1. Please explain two reasons to choose processes over threads?

Processes are independent of each other, are not linked together – can easily restart them if they crash

Data is not shared so there is no risk of corruption

1. What are the advantages of user-level threads over kernel-supported threads?

Thread switching does not require kernel mode privileges

Can run on any OS

1. What are the advantages of kernel-supported threads over user level threads?

Can schedule multiple threads from the same process on multiple processes

Kernel can schedule another thread if a thread in the same process is blocked

1. How can kernel-supported and user-level threads be combined?

They can be combined if the operating system offers kernel-level support. If it does, then the two types of threads can be combined into hybrid threading, where user level threads are mapped onto kernel level threads

1. Under ideal circumstances, what components of a process appear to be different between threads within the same process?

The program counter, entry in the process table, memory, and schedule are different between threads within the same process.

1. Under less than ideal circumstances, e.g. in the event of bugs, which of the above components can reveal themselves to be shared, if any? Given an example of when this might happen and how the sharing might be revealed.

Memory can reveal itself to be shared in less than ideal circumstances. This can show up when a thread is referencing the wrong memory address, accessing data from a different thread.

1. Are files shared between threads within the same process? If so, by what mechanism?

Yes, files are shared between threads within the same process using the COW mechanism.