CSci 144: Homework #1

Due on September 9, 2016 at 11:59pm

Professor Ming Li

Cole Burrell

Chapter 1

Problem 4

Suppose a computer system and all of its applications were completely bug free. Suppose further that everyone in the world were completely honest and trustworthy. In other words, we need not consider fault isolation.

Part 1 *How should an operating system allocate time on the processor? Should it give the entire processor to each application until it no longer needs it? If there were multiple tasks ready to go at the same time, should it schedule first the task with the least amount of work to do or the one with the most? Justify your answer.*
The operating system should allocate time on the processor as a mixture of fairness and priority of the process. The tasks can be completed on a first come, first serve basis, unless they require a higher priority. If multiple tasks reach the processor at the same time and they all have even priority, the ones with the least amount of work should be completed first. Once those tasks are completed, they are able to move on to other tasks, if applicable, and they would not be hung up on the tasks requiring more work.

Part 2 *How should the operating system allocate physical memory to applications? What should happen if the set of applications does not fit in the memory at the same time?*
Since application bugs do not exist, and there is no worry about malicious software infiltrating the memory of other software, the process would not have to be isolated in memory. If the software does not fit in the memory, it would need to wait until memory is freed up.

Part 3 *How should the operating system allocate its disk space? Should the first user to ask acquire all of the free space? What would the likely outcome be for that policy?*
Disk space should be allocated on a need basis. If a user is low on disk space, more disk space is allocated to the user. The first user should not just take all of the disk space on a multi-user system, as it leaves no room for more users.

Problem 6

How should an operating system support communication between applications? Explain your reasoning.
Modern operating system system calls is a very secure and efficient method for providing communication between the operating system and the application. The application has access to a very specific set of commands that are well coded and secure. These commands transparently allow user mode applications to ask for kernel levels commands to be executed.

Problem 9

How would you design a system to update complex data structures on disk in a consistent fashion despite machine crashes?
One key to preserving data on a disk is to provide the ability for the operating system to check the integrity of files on the event of a crash. The system can update complex data structures in chunks and not as one long process to reduce the amount of corrupted data in a crash.

Chapter 2

Problem 5

Define three types of user-mode to kernel-mode transfers.

- Interrupts: On a single core system, an interupt stops a process and takes immediate priority to complete. This interupts mostly stem from I/O on the system (such as the mouse, keyboard, network, hard drive, etc.), and require attention to complete. These interrupts usually consist of a handler and a timer to complete. On a multicore system, this task is only sent to a single core while the other cores continue on.
- Processor Exceptions: Processor exceptions occur from user programs that usually attempt to extend outside the scope of the process. Such events are: accessing data outside of the memory region, divide by zero, writing to read-only data, and setting a breakpoint to send to the debugger. The process is stopped and an error handler is sent to the user.
- System Calls: These are extremely common mode transfers. They occur when a user software needs to perform a kernel level task. System calls are very carefully coded access points within the kernel for security's sake. Such events are: create/delete files, read/write files, creating new process, establishing a connection to a web server, and sending/recieving packets over the network. After the system call is completed, the program returns to user level execution.

Problem 7

Most hardware architectures provide an instruction to return from an interrupt, such as iret. This instruction switches the mode of operation from kernel-mode to user-mode.

Part 1 *Explain where in the operating system this instruction would be used.*
This instruction is used to return an interrupted process's state back to the processor. This is usually used after an interrupt to return the process to the correct location.

Part 2 *Explain what happens if an application program executes this instruction.*
If an application program at the user level executes this instruction, a processor exception will occur because *iret* is outside of the scope of user level software.