CS433 Operating Systems

Homework #1

Part I – Overview

1. What are the three main purposes of an operating system? (6 pts)

The three main purposes of an operating system are:

* To execute user programs
* Make the computer system easier and more convenient to use
* Allow the computer hardware to be used in an efficient manner

1. To use cache memory, main memory is divided into cache lines, typically 32 or 64 bytes long. An entire cache line is cached at once. What is the advantage of caching an entire line instead of a single byte or word at a time? (4 pts)

The advantage of caching an entire line instead of a single byte or word at a time is that the processor will not have use CPU cycles to load an instruction or data into memory continuously based on the theory that data is stored sequentially and besides the byte that is needed at that time, the nearby bytes will most likely have needed to be accessed too (concepts of temporal and spatial locality). The end result is saving CPU cycles.

1. Instructions related to accessing I/O devices are typically privileged instructions, that is, they can be executed in kernel mode but not in user mode. Give a reason why these instructions are privileged. (3 pts)

These instructions are privileged because reading and writing to a device will possibly interfere with other processes. If I/O instructions were entrusted to users, then the user programs could misuse them. Examples of this could be: a rogue process reading in your keyboard input or a user program writing into a blocked off section of the hard drive, such as the main OS files which would result in tampering of the OS.

1. What is the main advantage of multiprogramming? (3 pts)

The main advantage of multiprogramming is having multiple programs resident at once and having them all active at the same time. While only one runs in any given instance, all the programs can run in turn. So, the CPU is utilized more effectively which will decrease response time as well as turn around time. An example of this could be when a program is waiting for I/O, another program can run and utilize the CPU.

1. What is the purpose of interrupts? What are the differences between a trap and an interrupt? (4 pts)

Interrupts transfer control to the interrupt service routine through interrupt vector that contains addresses of all service routines. A trip is a software-generated interrupt caused by either an error or an user request.

1. Direct memory access is used for high-speed I/O devices in order to avoid increasing the CPU’s execution load. (6 pts)
   1. How does the CPU interface with the device to coordinate the transfer?

The CPU can write values into special registers which the high-speed I/O device can access. The CPU then issues a command to the device to initiate the corresponding operation.

* 1. How does the CPU know when the memory operations are complete?

When the device is done, it will issue an interrupt which will tell the CPU that the device has finished its memory operations and it is complete.

* 1. The CPU is allowed to execute other programs while the DMAcontroller is transferring data. Does this process interfere withthe execution of the user programs?

Due to the way the memory controller is designed, the memory controller allows both the device and CPU to access memory simultaneously. The memory controller also ensures that access to memory is distributed fairly. So, it may interfere with the execution of user programs as by nature, the CPU cannot perform memory operations at maximum speed when it is having to compete with the DMAcontroller.

1. Describe some of the challenges of designing operating systems for mobile devices compared with designing operating systems for traditional PCs. (4 pts)
2. Storage capacity on a mobile device is far less than of a PC. Thus, memory management becomes more important.
3. There is less processing power and fewer processors, so the operating system must be more efficient as assigning resources as well as assigning processors to applications.
4. Security on an operating system of a mobile device must be more stringent as many people utilize their mobile devices to store a large part of their personal lives. Thus, it is more important to stop malicious code, rogue programs, trojans, viruses, etc.
5. Operating system must balance performance with battery life on a mobile device since it is not plugged into an outlet at all times.
6. Mobile operating systems usually use a touchscreen as a means of input as opposed to a keyboard and mouse, so it may be more difficult to design a variety of touch based gestures and inputs that the operating system would recognize.
7. Which of the following instructions should be allowed only in kernel mode? (4 pts)
   1. Disable all interrupts.
   2. Read the time-of-day clock.
   3. Set the time-of-day clock.
   4. Change the memory map.

C is the correct answer. This is because if a program is allowed to set the time-of-day clock, then it could theoretically constantly set the time-of-day clock back and as a result, gain a bigger time slice to run it’s code.

1. What is the purpose of the command interpreter? Why is it usually separate from the kernel? (4 pts)

The command interpreter reads commands from the user or a file of commands and then executes them. This can be done in one or multiple system calls. The command interpreter changes which is why it is separate from the kernel. For example, one can change from Bash Shell to Korn shell, etc.

1. What are the two models of inter-process communication? (2 pts)

Message passing model and shared memory are the two models of inter-process communication.

1. What is a bootstrap program and where is it stored?

A bootstrap program is the initial program that the computer runs when it is powered up or rebooted. It initializes all aspects of the system, from the CPU registers to device controllers to memory contents. Typically, it is stored in read-only memory (ROM) or electrically erasable programmable memory read-only memory (EEPROM) known by the general term firmware within the computer hardware.

1. There are several design goals in building an operating system, for example,resource utilization, timeliness, robustness, and so on. Give an example of twodesign goals that may contradict one another

Consider fairness and real time. Fairness requires that each process be allocated its resources in a fair way, with no process getting more than its fair share. On the other hand, real time requires that resources be allocated based on the times when different processes must complete their execution. A realtime process may get a disproportionate share of the resources.

1. Can the count = write(fd, buffer, nbytes) call return any value in count other than nbytes? If so, why?

Yes, it can if one takes into account a non-blocking socket described by the FD. When one writes and calls again there may be a chance that the write is trying to wrote more than the space left in socket buffer. So, it will write the number of bytes that will be equal to space left in socket buffer. So, count will not equal to nbytes.

1. What is the difference between kernel and user mode?

In user mode some parts of the memory are access protected and some instructions may not be executed

In kernel mode privileged instructions may be run and protected memory areas may be accessed

because chaos would ensue

1. What are the three general ways that a deadlock can be handled?

Ensure the system will never enter a deadlock state through deadlock prevention (mutual exclusion, hold and waits, no preemption, deadlock avoidance

Allow the system to enter a deadlock state then recover

Ignore the problem and pretend that the deadlock didn’t happen in the system

1. List the different states that a process can exist in at any given time.

New, running, waiting, ready, terminated

1. Describe the dining-philosophers problem and how it relates to operating systems

Shared data and deadlock handling, ex: 4 philosophers to be sitting simaltanously and allow phil to pick up chopsticks only if both are available and pick must be done in the critical section.

Use an asymmetric solution. Odd/even numbers

1. A real-time system needs to handle two voice calls that each run every 6 msec and consume 1 msec of CPU time per burst, plus one video at 25 frames/sec, with each frame requiring 20 msec of CPU time. Is this system schedulable with Rate Monotonic Scheduling?

So, we will do 1/.006 = 166.7 times per second x 2 = 333.4 ms

25 frames/sec with each frame requiring 20 msec = 25 x 20 = 500 ms

So, we get 833.4 ms which is less than 1 full second, so this system is schedulable with Rate Monotonic Scheduling.

1. To find out how many child processes are created upon execution, do 2^n – 1 with n being the number of forks.
2. What role does the dispatcher play in CPU scheduling?

The dispatcher gives control of the CPU to the process selected by the short-term scheduler. To perform this task, a context switch, a switch to user mode, and a jump to the proper location in the user program are all required. The dispatch should be made as fast as possible. The time lost to the dispatcher is termed dispatch latency.

1. The four major categories of the benefits of multithreaded programming is

Responsiveness may allow continued execution if part of a process is blocked, Resource Sharing: threads share resources or process, easier than shared memory or message passing, Economy: cheaper than process creation, thread switching lower overhead than context, Scalability: process can take advantage of multiprocess or architectures.

1. With Gant Charts, Turnaround time = completion time – arrival time

Waiting Time = Turnaround time – burst time

1. Write two short pseudo-code functions that implement the simple semaphore wait() and signal() operations on global variable S.

Wait(s)

{

While(s <=0);

s--;

}

Signal(s){

s++;

}

62. Explain the difference between the first readers–writers problem and the second readers–-writers problem.

Ans: The first readers–writers problem requires that no reader will be kept waiting unless a writer has already obtained permission to use the shared database; whereas the second readers–writers problem requires that 13 once a writer is ready, that writer performs its write as soon as possible. Feedback: 6.6.2 63. Describe the dining-philosophers problem and how it relates to operating systems.

Ans: The scenario involves five philosophers sitting at a round table with a bowl of food and five chopsticks. Each chopstick sits between two adjacent philosophers. The philosophers are allowed to think and eat. Since two chopsticks are required for each philosopher to eat, and only five chopsticks exist at the table, no two adjacent philosophers may be eating at the same time. A scheduling problem arises as to who gets to eat at what time. This problem is similar to the problem of scheduling processes that require a limited number of resources.

1. Which of the following instructions should be privileged? set the value of timer, clear memory, turn off interrupts, switch user to monitor mode
2. What is the difference between timesharing and multiprogramming systems?

Multiprogramming is the effective utilization of the CPU time by allowing several programs to use CPU at same time. Time sharing of computer facility by several users that want to use the same facility at the same time.

1. Discuss three major complications that concurrent processing adds to an operating system.

a) A method of time sharing must be implemented to allow each of several processes to have access to the system. This method involves the preemption of processes that do not voluntarily give up the CPU (by using a system call, for instance) and the kernel being reentrant (so more than one process may be executing kernel code concurrently)  
b)Processes and system resources must have protections and must be protected from each other. Any given process must be limited in the amount of memory it can use and the operations it can perform on devices like disks.  
c) Care must be taken in the kernel to prevent deadlocks between processes, so processes aren't waiting for each other's allocated resources.

27. Give two reasons why caches are useful.

Caching is copying information into faster storage system; main memory can be viewed as a cache for secondary storage. Caching is an important principle because it can be performed at many levels in a computer (hardware, operating system, software). Cache is also useful when two or more components need to exchange data, and the components perform transfers at different speeds (CPU registers vs. hard drive)

Caching can reduce cost of finding data from faster storage. Cache can cache smaller than storage being cached as well hold instructions from CPU. Cache can have consistency problems as well as have a limited size. Increasing cache size will help and can be useful but cache should only use data that is being used. The device itself would still be needed in order to properly store data when power goes off.

1. What are some advantages of using a higher level language to implement an operating system?

The code can be written faster, is more compact and easier to understand. Compiler adds faster process in implementing OS since they are frequently updated to work faster.

1. A system has 4 processes and 5 allocatable resource. Current allocation and max needs are as follows

If x is 0, we have a deadlock immediately. If x is 1, process D can run to completion. When it is finished, the available vector is 1 1 2 2 1. Now A can run to complete, the available vector then becomes 2 1 4 3 2. Then C can run and finish, return the available vector as 3 2 4 4 2. Then B can run to complete. Safe sequence D A C B.

New = when process has been called, ready = process ready to run, running = process is executing, waiting = process is waiting for user input or interrupt to finish, terminated = process has finished executing

1. What is a primary advantage of EDF scheduling algorithm over the rate-monotonic scheduling algorithm?

EDF is optimal, it can schedule processes so that each process can meet its deadline requirements and CPU utilization will be 100%. EDF doesn’t require that processes be periodic, a process doesn’t require a constant amount of CPU time per burst and priorities are dynamic. It can miss deadline though so that’s where RMS is useful as RMS the task with minimum execution period has high priority.

1. Explain how time quantum value and context switching affect each other, in a round-robin scheduling algorithm.

Time quantum and context switch inversely affect each other in round robin. For example, a short time quantum value will mean more context switches or vice versa in round robin. Short quantum is good because it means many processes will get brief chance to run on processor but it’s bad because it means a lot of context switching which is pure overhead.