Operating Systems I

CS 474

Homework #1

Due date: 10/07/2019 at 11:59 pm
Instructions:
1) To complete assignment one, you need to read Chapters 1, 2, 3, and 4 of the textbook.
2) HW must be submitted on typed pdf or word document.
All questions need to be worked on and answered individually. No collaboration of any form is permitted and will amount to plagiarism if performed.

Questions from Chapter 1

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

1) **To provide an environment for a computer user to execute programs on computer hardware in a convenient and efficient manner.**
2) **To allocate the separate resources of the computer as needed to solve the problem given. The allocation process should be as fair and efficient as possible.**
3) **As a control program it serves two major functions: (1) supervision of the execution of user programs to prevent errors and improper use of the computer, and (2) management of the operation and control of I/O devices.**

1.15 Describe the differences between symmetric and asymmetric multiprocessing. What are three advantages and one disadvantage of multiprocessor systems? (8 pts)

- **Symmetric processing treats all processors as equals; I/O can be processed on any of them. Asymmetric processing designates one CPU as the master, which is the only one capable of performing I/O; the master distributes computational work among the other CPUs.**
- **advantages: Multiprocessor systems can save money, by sharing power supplies, housings, and peripherals. Can execute programs more quickly and can have increased reliability.**
- **disadvantages: Multiprocessor systems are more complex in both hardware and software. Additional CPU cycles are required to manage the cooperation, so per-CPU efficiency goes down.**

1.19 What is the purpose of interrupts? What are the differences between a trap and an interrupt? Can traps be generated intentionally by a user program? If so, for what purpose? (9 pts)

- **An interrupt is a hardware-generated signal that changes the flow within the system.**
- **An interrupt handler is summoned to deal with the cause of the interrupt; control is then returned to the interrupted context and instruction. A trap is a software-generated interrupt.**
- **An interrupt can be used to signal the completion of I/O so that the CPU doesn't have to spend cycles polling the device. A trap can be used to catch arithmetic errors or to call system routines.**

Questions from Chapter 2

2.1 What is the purpose of system calls? (5 pts)

System calls allow user-level processes to request services of the operating system.

2.13 Describe three general methods for passing parameters to the operating system. (6 pts)

1) **Pass parameters in registers**
2) **Registers pass starting addresses of blocks of parameters**
3) **Parameters can be placed, or pushed, onto the stack by the program, and popped off the stack by the operating system**

2.21 What is the main advantage of the microkernel approach to system design? How do user programs and system services interact in a microkernel architecture? What are the disadvantages of using the microkernel approach? (9 pts)

- **Benefits typically include the following: (a) adding a new service does not require modifying the kernel, (b) it is more secure as more operations are done in user mode than in kernel mode, and (c) a simpler kernel design and functionality typically results in a more reliable operating system.**
- **User programs and system services interact in a microkernel architecture by using inter-process communication mechanisms such as messaging. These messages are conveyed by the operating system.**
- **The primary disadvantages of the microkernel architecture are the overheads associated with inter-process communication and the frequent use of the operating system's messaging functions in order to enable the user process and the system service to interact with each other.**

Questions from Chapter 3

3.8 Describe the differences among short-term, medium-term, and long-term scheduling. (7 pts)

- **Short-term (CPU scheduler): selects a process from those that are in memory and ready to execute, and allocates the CPU to it.**
- **Medium-term (memory manager): selects processes from the ready or blocked queue and removes them from memory, then reinstates them later to continue running.**
- **Long-term (job scheduler): determines which jobs are brought into the system for processing.**

3.9 Describe the actions taken by a kernel to context-switch between processes. (6 pts)

In general, the operating system must save the state of the currently running process and restore the state of the process scheduled to be run next. Saving the state of a process typically includes the values of all the CPU registers in addition to memory allocation. Context switches must also perform many architecture-specific operations, including flushing data and instruction caches.

3.12 Including the initial parent process, how many processes are created by the program as shown. (10 pts)

```
#include<stdio.h>
#include<unistd.h>
int main()
{
      int i;
      for (i=0; i<4; i++)
         fork();

      return 0;
 }
```

16 processes including the initial parent process are created.

Q4. Assume that the following program contains no syntax errors. As it executes it will create one or more processes. Simulate the execution of this program and show how processes are created (7+8 pts)

```
#include<stdio.h>
main()
{
int m=10, n=5,count=1, mult=1;
while(count <3)
{
    if(m != 0)
    {
        m = fork(); n = n+25;
    }
    else
    {
        m = fork(); n = n+20; mult = mult*n;
    }
    printf(" n = %d       mult  = %d", n, mult);
    count =count + 1;
  }
}
```

1) What is total number of processes? Show your work.
2) What will this program print on the screen when it executes?

- **4 processes**
- **n=30 mult=1**
  n=30 mult=1
  n=55 mult=1
  n=55 mult=1
  n=50 mult=50
  n=50 mult=50

Questions from Chapter 4

4.7 Under what circumstances does a multithreaded solution using multiple kernel threads provide better performance than a single-threaded solution on a single-processor system? (6 pts)

When a kernel thread suffers a page fault, another kernel thread can be switched in to use the interleaving time in a useful manner. A single-threaded process, on the other hand, will not be capable of performing useful work when a page fault takes place. Therefore, in scenarios where a program might suffer from frequent page faults or has to wait for other

system events, a multithreaded solution would perform better even on a single-processor system.

4.10 Which of the following components of program state are shared across threads in a multithreaded process? (4 pts)

a, Register values

b, Heap memory

c, Global variables

d, Stack memory

4.14 Consider a multiprocessor system and a multithreaded program written using the many-to-many threading model. Let the number of user-level threads in the program be greater than the number of processors in the system. Discuss the performance implications of the following scenarios. (9 pts)

1), The number of kernel threads allocated to the program is less than the number of processors.

2), The number of kernel threads allocated to the program is equal to the number of processors.

3), The number of kernel threads allocated to the program is greater than the number of processors but less than the number of user- level threads.

1) **When the number of kernel threads is less than the number of processors, then some of the processors would remain idle since the scheduler maps only kernel threads to processors and not user-level threads to processors.**
2) **When the number of kernel threads is exactly equal to the number of processors, then it is possible that all of the processors might be utilized simultaneously. However, when a kernel thread blocks inside the kernel (due to a page fault or while invoking system calls), the corresponding processor would remain idle.**
3) **When there are more kernel threads than processors, a blocked kernel thread could be swapped out in favor of another kernel thread that is ready to execute, thereby increasing the utilization of the multiprocessor system.**