Practice Questions #1

(1) How is a system call different from a function call?

(2) How is system call processing different from interrupt servicing?

(3) What is the significance of mode switching in system call processing? That is, can we have a system call without the mode switch?

(4) Consider a uni-programing system. The user wants to run job A and job B. A has the following characteristics: runs for 10 ms waits for 990 ms and the pattern continues for many iterations. Job B runs for 100ms and waits for 400ms and the pattern continues for many iterations. If you are measuring CPU utilization values over half second intervals, what are the possible values you could get?

(5) In the system given in (4), we enabled multi-programming. There is a single processor in the system and just jobs A and B. What are the time measured utilization values over half second intervals in this case?

(6) Why is memory protection an important concern with multi-programming?

(7) We have a computer with 512 MB of RAM. We have 3 applications. Application A needs 300 MB of RAM, B needs 200 MB, and C needs 300 MB. How can a virtual memory system load all three programs and still provide good performance? What characteristics of the applications are exploited by the virtual memory system?

(8) Can a micro-kernel OS be more reliable than a monolithic OS? Provide reasons why micro-kernel could provide a more reliable OS. Could micro-kernel make an OS less reliable too?

(9) Consider a web server. Each incoming request is processed as follows: input side processing (done on all incoming requests) – 10ms, disk access – 900ms, cache access – 100ms. The input side processing determines whether a request can be served by the cache or disk access is required. Assume that the web server has a single disk that serves the requests one after the other. The cache will hold popular requests after warm up. If the web server uses a single thread or process, what is the best and worst request rates achievable by the server?

(10) Suppose we use two threads in the above web server, what is the best request rate we could achieve?

(11) What is the maximum number of threads we can engage in the web server? That is find the number of threads beyond which the web server's request rate is not going to increase?

(12) Consider the following C code fragment. How many processes run soon after the execution of the given fragment?

```
for (i=0; i<4; i++) {
        fork();
        run_compute (i);
}
```

(13) You are expected to run programs in a computer with a sleeping beauty approach for resuming the dispatcher.

The support code provided by the professor for your assignment has a following core segment.

```
while (1) {
    obtainData (data)
    processData (data)
    outputData (data)
}
```

You immediately realize that there is a problem because the code does not relinquish the CPU and the program would not even yield the CPU for the OS. As a result, the OS would not be able to run and perform certain housekeeping tasks necessary for the code's execution.

```
while (1) {
    obtainData(data)
    processData(data)
    outputData(data)
}
```

yield();

You suggest the insertion of yield() as shown above.

What does yield() do? Can you provide the pseudo code for yield()?

(14) You are given a computing system where a system call takes 2000 ns (nano seconds), saving or restoring registers takes 200 ns, invoking interrupt servicing routine takes 500 ns. Approximately, how long would it take to switch from one process to another process?

(15) Consider the following code fragment.

```
close(1);
fd = open("temp2.txt", ...)
if (fork() == 0) {
        printf(" Message from A");
}
printf("Message from B");
```

What will be the contents of temp2.txt after executing this fragment?

(16) Lets say you don't want "Message from A" in your temp2.txt file. What modifications would you do?

(17) What modifications could you do to the code fragment to ensure that the output of "Message from A" shows up on the standard output?

(18) Can a program generate an address that is not in its address space?

(19) When a program is running in a modern operating system, portion of its address space is occupied by the kernel. Why is it necessary for the kernel to hold portion of the address space?

(20) Briefly explain how input redirection works in a UNIX like operating system. Show kernel-level data structures, system calls, and the order of manipulating them in in your answer.

(21) Briefly explain how command piping works in a UNIX like operating system. Show kernel-level data structures, system calls, and the order of their manipulations in your answer.

(22) An application took 600s to run in a single core machine. It took 170s on a four core machine. What is the speedup you got when runng it on the four core machine?

(23) What portion of the application is actually parallel?

(24) What is the expected runtine in a 16 core machine?

(25) Lets say the actual runtine you got when you run the application is less than the value computed above. Explain why you got better performance.

(26) Making a single application instance run over multiple cores involves additional programs like creating threads, distributing data, consolidating the results, etc. How can such overhead be factored into the speedup equation we saw in the class?

(27) You are asked to write a large program in C or C++ to solve a computational problem. You use a threading library because you realize that the application can be broken down to concurrent subtasks. By implementing the subtasks using threads you expect the application to run faster with more processor cores.

To your disappointment, your applicatn using threads actually runs slightly slower than the original serial version in a single core. You

explain it as threading overhead. With multiple cores, you are still at the same speed! What is going on? You go over the program and find no synchronization problems.

(28) What are the advantages of user-level threading over kernel level threading?

(29) What are the advantages of kernel level threading over user level threading?

(30) Can you think of an application that is better suited for user-level threading?