Name: _____

CS 4/53201 **Final Exam** **Operating Systems**

Monday 10 December 2001

1. **For the operating system to provide effective protection against accidental or intentional damage by a user program, it must have good support from the CPU architecture.**

   a. **Explain how the operating system can provide protection if the CPU architecture provides two modes for instruction execution: user mode, and kernel / privileged / monitor mode. (10 points)**

      The operating system kernel runs in kernel mode, which means it can use the privileged instructions provided by the CPU. In contrast, user programs run in user mode and can not use the privileged instructions, so the only way they can perform certain actions (such as accessing a printer or network port) is to make a system call, which traps into kernel mode, essentially asking the operating system to perform that access for them. Thus the operating system can prevent user programs from taking certain actions.

   b. **Explain how the operating system can provide protection if the CPU architecture provides base and limit registers. (10 points)**

      The operating system computes a physical address by adding the virtual address to the base value. It also compares the virtual address to the limit, and if the virtual address is above the limit, then the process is trying to access memory above the process, so an address exception is generated. Since the virtual address will never be negative, the process will not try to access memory below the process.

2. **Besides waiting on an I/O operation to complete, for what other reason might a process or thread be in the blocked state? (Hint — there are 3 other reasons.) (10 points)**

   I miscounted, there are actually four other reasons!

   Waiting on a message to arrive after executing a blocking RECV operation

   Waiting on a semaphore V or signal, or a condition variable signal

   Waiting to acquire a resource

   Waiting for a page to be loaded into memory after a page fault

3. **Shortest-Job-First (SJF) is a non-preemptive algorithm for CPU scheduling.**

   a. **How effective is this algorithm with respect to scheduling processes with a <u>short</u> CPU burst? (5 points)**

In general, it is very effective for scheduling short processes, as it gives them higher priority in the scheduling process. The only potential problem is that if a long process gets scheduled (because there are no shorter processes in the ready queue) and then a short process arrives while the long process is executing, the short process has to wait until the long process completes because SJF is not preemptive.

b. How effective is this algorithm with respect to scheduling processes with a <u>long</u> CPU burst? (5 points)

Long processes have to wait until all shorter processes have been processed first, regardless of how long they have been waiting. Using SJF, long processes could potentially starve, waiting to execute indefinitely.

c. The overhead that is required for the operating system to use this algorithm is high. Explain why this is true. (5 points)

The algorithm requires the CPU scheduler to (1) predict the CPU burst length, and then (2) sort the lengths to find the shortest, both of which are time-consuming.

4. This question examines deadlock and deadlock prevention.

a. List the 4 necessary and sufficient conditions for deadlock. (5 points)

Mutual exclusion, no preemption, hold and wait, and circular wait

b. Consider the following proposal: if a process wants to acquire a new resource, it must first release all of its current resources, and then it must make a single request to acquire both the new resource plus those that it just released (all at the same time).

Is deadlock still possible? If so, explain how it might arise. If not, indicate which of 4 necessary and sufficient conditions does not hold, and briefly explain why. (5 points)

No, deadlock is not possible, because this proposal eliminates the hold and wait condition.

5. In demand paging, Least Recently Used (LRU) is the ideal algorithm. Why is LRU not used in practice? (10 points)

Because recording the time the page is accessed would have to be done every time there is a memory access, which would slow the system down too much. The algorithm also has a moderately large overhead with respect to the memory space needed to store this information and the time required to compute the least recently used page.

6. The operating system represents each file using a file descriptor (in UNIX terms, an inode).

a. What information is stored in the file descriptor? (10 points)

File type, size, access times, owner, group, access permissions, and pointers to the blocks where the file is located on the disk.

b. Why is the file name not stored in the file descriptor? Be specific. (5 points)

Because if links (or "aliases" or "shortcuts") are used, the same file may be referred to by different names from different directories. Therefore, the name must be stored with the directory, not with the file or file descriptor.

7. Explain the basic idea of multi-level indexing, with respect to file systems. Draw a diagram if that will aid in your explanation. (15 points)

The inode has some number of pointers in the inode itself, which point directly to blocks storing the file. If the file is small, those inodes may be sufficient to point to the file. If the file is larger, the inode has another pointer to a block containing more pointers to blocks in the file. If the file is larger still, the inode has another pointer to a block containing pointers to blocks containing pointers to blocks in the file. See the diagram in Lecture 32 for details.

8. Consider the various disk head scheduling algorithms discussed in class.

a. Briefly describe the LOOK algorithm. (5 points)

Move the disk head from one side of the disk to the other, picking up requests as it goes along. Once it reaches the last request in a particular direction, it turns around and heads back the other way.

b. What potential problem does the SSTF algorithm have? Explain your answer. (5 points)

The main problem is that the head can stay in one part of the disk indefinitely, leading to the starvation of requests far away from that part of the disk.

c. Would it be a good idea for SSTF to consider rotational delay as well? Explain your answer. (5 points)

If the goal is to service the request that can be reached in the shortest amount of time, then rotational delay should be considered as well, since it is roughly as time-consuming as seek time. However, rotational delay is harder to estimate, and would add to the overhead of the algorithm, which is why it is not considered in practice.

9. What are the major goals of a distributed file system? (10 points)

Network transparency — provide the same operations for accessing both remote and local files, ideally with the clients not having to know the location of the files.

Availability / robustness — the file service should be maintained even in the presence of partial system failures.

Performance — the file system should overcome the bottlenecks of a centralized file system.

Name: _____

10. Consider the Central Coordinator algorithm for distributed mutual exclusion.

a. What does a thread do when it receives a *reply* message? (5 points)

It enters the critical section (it requested permission to do so earlier, via a *request* message).

b. What are the major problems with this algorithm? (5 points)

The coordinator is a performance bottleneck. The coordinator is also a single point of failure (if it crashes the algorithm no longer works). Lost reply or release messages also result in the failure of the algorithm.

11. Linux represents processes and threads using the same internal representation, but uses the fork system call to create new processes and the clone system call to create new threads. How do these two system calls differ? Be specific with respect to the Linux process properties. (10 points)

Under Linux, processes have several sets of properties: process identity (process ID, process credentials, personality), process environment, and process context (scheduling context, accounting information, file table, file-system context, and virtual memory context).

When fork creates a new process, it creates new identity and scheduling contexts, and copies the other contexts. When clone creates a new thread, it also creates new identity and scheduling contexts, but the cloned thread shares the other contexts with the parent.

12. A unique feature of Windows NT/2000 is its Environmental Subsystems. Briefly explain the functionality provided by these Environmental Subsystems, and give an example of one such Environmental Subsystem. (10 points)

The Environmental Subsystems allow binaries that were originally compiled to run under another operating system, to run under Windows NT/2000. These include the OS/2 subsystem (which runs OS/2 applications), the Win32 subsystem (which runs most programs under Windows NT/2000), the Virtual DOS Machine (VDM, which runs DOS applications), and the POSIX subsystem (which runs applications that follow the POSIX.1 standard).