Name: _____

CS 33211 **Exam #1** OS

Monday 1 October 2007

1. **Most modern CPUs provide both user mode and privileged (kernel) mode instructions.**

   a. **Why is it necessary to differentiate between two sets of instructions in this manner? (10 points)**

   The privileged instructions access machine functions, such as I/O and memory control, which should be restricted to authorized use only. Thus user programs and OS support programs are only permitted to use user mode instructions, while the OS kernel is permitted to use both user mode and privileged mode instructions.

   b. **Explain how the CPU transitions from user mode to kernel mode and vice-versa. (10 points)**

   When a user process makes a system call, a trap (software-generated interrupt) occurs, which causes (1) the appropriate trap handler to be invoked and (2) kernel mode to be set. This is the only method for the CPU to go into kernel mode. When the trap handler finishes, user mode is set as control returns to the calling process.

   Note that it's important to realize that a user program can not just put itself into kernel mode; going into kernel mode is a CPU function that happens only through one carefully-managed mechanism (the OS's set of traps).

2. **Define "multiprogramming". (10 points)**

   Multiprogramming refers to the OS keeping multiple programs in memory and switching between them.

   Note that multiprogramming differs from timesharing in that multiprogramming may be nonpremptive, choosing a new program to execute only when the current one terminates or blocks, whereas a timesharing system must provide adequate response time to support interactive computing, which usually requires a preemptive scheduling algorithm.

3. **What is the distinction between application programs, system programs, and the micro-kernel in a micro-kernel-based OS such as Mach? (15 points)**

   The micro-kernel is the "minimal" kernel possible, consisting of only those few core services that must run in privileged (kernel) mode. The rest of the "OS" — that portion which can

run in user mode — is considered to by system processes / programs. Application programs are the non-OS programs such as word processors, multimedia players, and games.

4. Give a few examples to illustrate different reasons for a process to be created. (10 points)

When the OS boots, it starts the OS kernel and various support processes. When a user logs on, a shell process may be started. When a user runs a program, a process is created, and that process in turn may create other processes (i.e., email client starting a JPEG viewer).

5. Consider the 5-state process model, with the states labeled as new, running, ready, waiting (or blocked), and terminated.

a. At a particular point in time, how many processes can be in each state, assuming a system with a single OS and a single CPU? (10 points)

Only one process can be in the running state, but an arbitrary number of processes can be in the new, ready, waiting, and terminated states.

b. What is the difference between the ready state and the waiting state? (10 points)

Processes in the ready state are waiting to be dispatched onto the CPU and will execute once that happens, while processes in the waiting state are waiting for some event (e.g., I/O operation to complete, or a signal to be received) and would not execute even if dispatched onto the CPU.

6. In the UNIX process model, both states 3 and 5 are "ready" states. Why are two ready states needed? (10 points)

State 3 is for processes that are "ready" and are physically in memory. State 5 is for processes that are "ready" but have been swapped out to disk; for these processes to run they must be swapped back into memory and thus move into State 3.

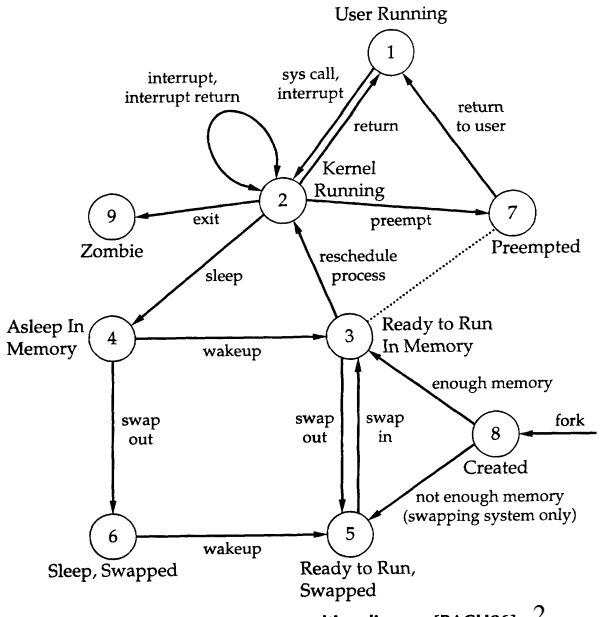

FIGURE 3.16 UNIX process state transition diagram [BACH86]

Name: _____

7. The name "remote procedure call" (RPC) was chosen to highlight the similarity that mechanism provides to normal procedure calls.

a. In what way(s) are RPCs similar to normal procedure calls? (5 points)

They are similar to a normal procedure call in that parameters are passed to an RPC, the calling process waits while the RPC executes, the RPC may return values to the calling process, and when the RPC returns control to the calling process it continues its normal execution.

a. In what way(s) are RPCs different from normal procedure calls? (10 points)

A normal procedure call calls a procedure within the same process, on the same machine, while RPC will likely call a procedure in a different process on a different machine. Further, a normal procedure call passes data to the procedure via a stack or in memory, while an RPC passes data to the procedure via messages over a network. A normal procedure call is written by the programmer and compiled directly, whereas with RPCs the programmer includes extra information that the system uses to generate stub / skeleton code, which is then compiled along with the program and handles the details of the communication.