

CS162, Spring 2004
Section 3 Quiz 1
Steve Martin

1. What are the execution states for a thread/process? Draw a state diagram for a thread and label the edges.

States:

New

Ready

Running

Waiting

Done

Edges:

Admitted (New->Ready)

I/O or Event Completion (Waiting->Ready)

Interrupt (Running->Ready)

Scheduler Dispatch (Ready->Running)

Exit or Complete (Running->Done)

I/O or Event Wait (Running->Waiting)

2. What happens when a thread is 'Run'?

First, must load state, which includes:

- *reload registers*
- *reload memory state*
 - o *stack pointer*
 - o *page table register (if **process**)*
 - o *flush tlb (if **process**)*
- *reload program counter.*

Then jump to PC.

3. What is an internal event? How are they different from an external event?

Internal event: run-time exception, system call exception.

External event: hardware interrupt--i.e. i/o complete, key hit on board, timer fired, etc.

4. What are the differences between forking a thread and forking a process?

Thread forks are all within a single address space—i.e. we only need to create data structures describing the thread and a stack for it.

Process forks do much more. A process encapsulates more state than a thread, and forking must duplicate this state. The most expensive part of this is duplicating complete address spaces—including their contents. A forked process is a copy of the original.

5. Are thread forks or UNIX process forks more efficient? How could we improve efficiency?

Thread forks are much more efficient because less state must be created.

UNIX forks can be made more efficient through a technique called copy-on-write. This is basically sharing memory between the child and parent process, and only doing the duplication when the memory is written to. See the UNIX manpage for `vfork()`.

6. What is the difference between an asynchronous and a synchronous system call?

Asynch - a system call that does not yield control, but is called and then the program can continue to be executed. Example: `fork()`

Synch - a system call that blocks and only returns when the system call is complete. Example: `read()`

7. What is the difference between a `thread.join()` and a `thread.sleep()` call?

Join waits for another thread to finish. Once the thread join is called with finishes, the thread becomes runnable.

Sleep causes a thread to sleep--i.e. not be runnable--until an outside agent calls `Ready()` on it.