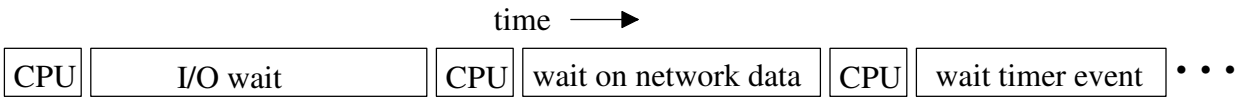
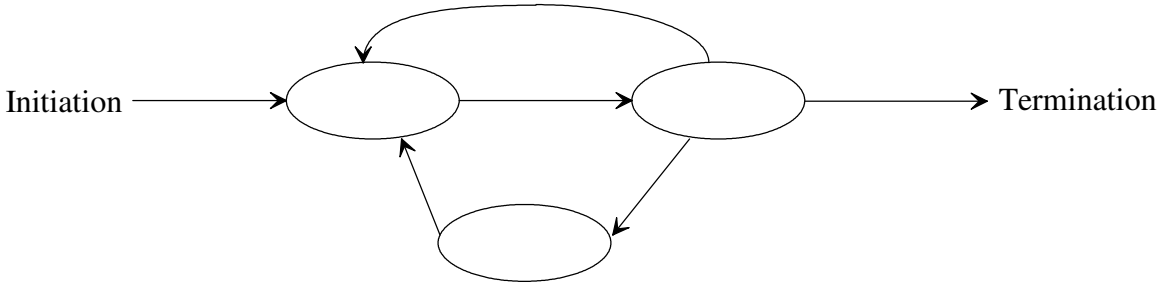
1) During its lifetime, a thread alternates between needing the CPU and waiting for events (input device, signal from another thread, timer to expire, etc.)

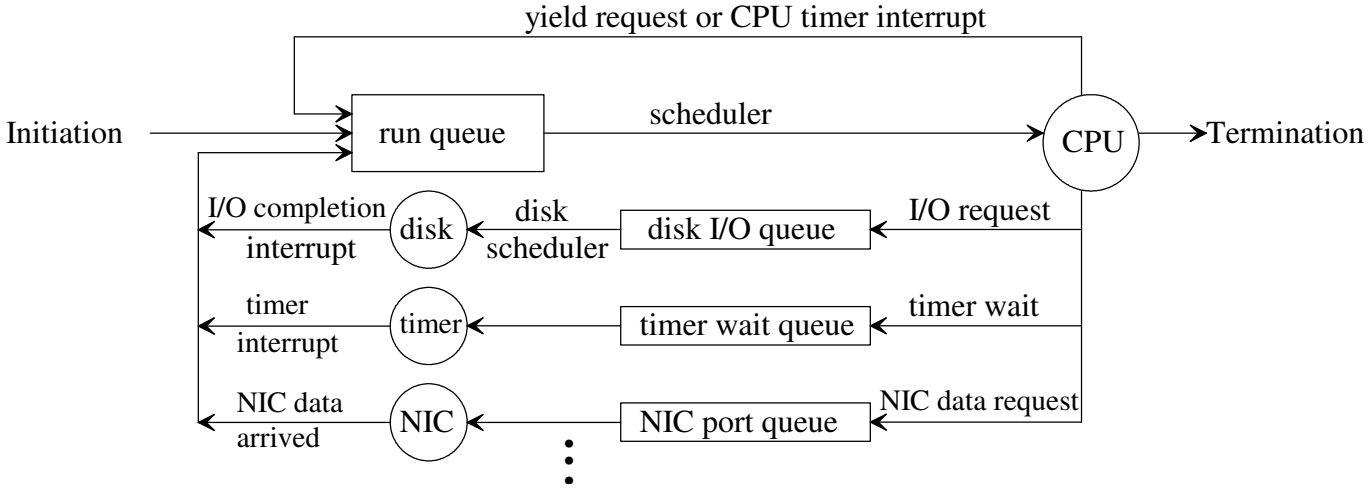


The *scheduler* chooses which thread to *dispatch*, i.e., run next. In the thread state diagram below:
a) Label the states: "waiting", "runnable", "running"
b) Label the transistion between states: "I/O or wait request", "dispatch", "yield or preemption", "I/O complete or event signaled"

Thread State Diagram



2) Think of the TCB (thread control block) for a thread as moving around from queue to queue depending on its state.



The wait queues might not be first-in-first-out (FIFO) queues. What might the following queues use for scheduling?

a) disk I/O queue

b) timer wait queue

3) The scheduler for the run queue tries to "make all users happy," but its goals are generally:
- maximize system performance
- allow users to exert control on scheduling

To maximize a system performance, why should the scheduler try to keep as much hardware (processors, I/O devices, etc.) as possible busy?

4) One measure of system performance is *throughput* - amount of "work" per unit time. If work equates to CPU work, then we want to keep threads running on the CPUs as much as possible.

a) How does a cache memory help a CPU work more efficiently?

b) On a thread switch, how does the cache impact performance?