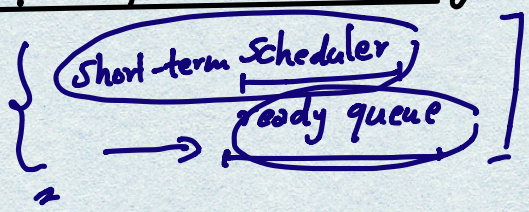
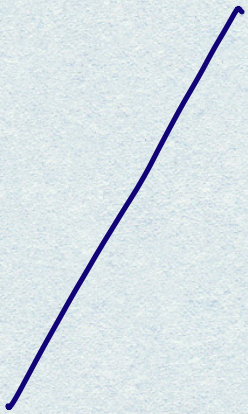
Preemptive vs Nonpreemptive Scheduling

Preemptive Scheduling



- In preemptive scheduling, each process gets a slice of CPU's time to run. (10ms)
- The running process can be forcely removed from the CPU under following conditions:-

1) Timeslice expires (running → ready)
2) An interrupt occurs on the CPU (running → ready)
3) A higher priority process enters the ready queue.
   - → NEW → READY
   - → WAITING → READY



Nonpreemptive Scheduling

- In nonpreemptive scheduling, the running process cannot be forcefully removed from the CPU.
- A running process holds the CPU till :-

1) it terminates.
2) it switches from RUNNING to WAITING state on its own because of :
   a) I/O request → wait()
   b) waiting for the child to complete { fork()
   c) if the mutex or the semaphore is occupied.

Examples :-

Nonpreemptive : Microsoft Windows 3.x

Preemptive : All windows version after windows 95, Mac OS, Linux

Cost of Preemptive scheduling :-

1) Sharing data between processes

- When one process is updating the data, it gets preempted.
- If some other process tries to read this data, it will find this data in inconsistent data.
- Different mechanisms like spin lock / mutex were developed to coordinate access.

2) Overhead of guarding shared data in interrupts

- whenever an interrupt happens, an interrupt handler is called.
- what if the same interrupt happens on two CPU's and the interrupt handler shares data ?
- Need to use (mutex) to protect critical section.

3) Increases the complexity of os design

- what happens when the kernel is changing some important data (I/O queues) and gets preempted in middle ?

- what if the kernel again tries to modify this data on behalf of some different process ?