### Concepts

- **Scheduler:**
  - OS entity which decides in which order and how long a process in the ready list will execute on the CPU.

- **CPU I/O Burst Cycle:**
  - processes execution generally cycles through CPU execution and IO wait.

- **Preemptive Scheduling:**
  - a process already in the CPU may be suspended dynamically, even when it did not finish or did not voluntary request wait.

---

**CPU Scheduling**

```
Job queue

Ready queue

CPU

Time expired

I/O

IO queue

IO requests Q

Child terminates

Child executes

Interrupt occurs

Interrupt Q

Fork a child Q
```
Concepts (contd..)

- **Dispatcher**
  - the specific module which gives control of CPU to the process selected by the short-term scheduler

- **Dispatch Latency**
  - the time it takes to dispatch; that is: switching context+switching to user mode+jump to the correct location of the new user process

- **Starvation**
  - A situation when a process is deprived of any CPU time for indefinite period.

Scheduling Performance Criteria

- **CPU utilization:**
  - time CPU executed user process/ total time

- **Throughput**
  - # of completed process/ total time

- **Turn-around time**
  - total waiting time+total execution time= time submitted-time completed

- **Waiting time**
  - total time spent waiting in the ready queue

- **Response time**
  - the initial time in the waiting queue

Desirable Goals?
Scheduling Policies

- First-In-First-Out (FIFO)
- Shortest-Job-First (SJF)
- Preemptive Shortest-Job-First (pSJF)
- Priority Scheduling
- Round Robin (RR)
- Multilevel Queue
- Multilevel Feedback Queue

Multilevel Queue

- Priority 1
- Priority 2
- ....
- ....

- System process
- Interactive process
- Interactive editing process
- Batch process
- Student process

- High Priority First or CPU time slice
- Processes are permanently assigned
- Each Queue can have its own scheduling algorithm
- Starvation possible
Thread Scheduling

- **Load Sharing**
  - Very effective on multiprocessor system
  - First Come First Served
  - Smallest number of threads first
  - Preemptive smallest number of threads first
- **Gang Scheduling**
  - Less number of blocks
  - Less context switching
- **Dedicated Processor Assignment**
  - Absolutely no context switching
  - Suitable for massively multiprocessor system
- **Dynamic Scheduling**
  - Compiler+OS controls the number of threads

Real Time Scheduling

- Hard realtime vs. soft realtime.
- Periodic vs. aperiodic realtime jobs.
- Preemption points

Multiprocessor Scheduling

- Heterogenenous vs. Homogeneous
- Self Scheduling vs. Master-Slave
  - queue access
  - master bottleneck
Example UNIX 4.3 BSD

- Multilevel Feedback with Round Robin in each group.
- Bands:
  - Swapper
  - Block I/O device control
  - File manipulation
  - Character I/O device control
  - User Processes
- Priority Computation:

\[ CPU_j(t) = \frac{1}{2} U_j(t) + \frac{1}{2} CPU_j(t - 1) \]

\[ p_j(t) = Base_j + CPU_j(t - 1) + nice_j \]

---

WINDOWS NT Scheduling

Design Objective: Maximum Responsiveness
- Priority Driven Preemptive Scheduler
- Two priority bands
  - real-time (31-16)
  - variable (15-0)
- Real time processed have fixed priority
- Other processes have dynamic priority
  - P=process-base (0-15) +thread-base (-2 to +2)
  - for preempted threads --thread -base
  - for on I/O, interrupt threads ++thread-base;
  - for wait on keyboard/display I/O thread ++++thread-base;
- On N-multiprocessor system
  - (N-1) processors gets (N-1) top priority threads.
  - Remaining 1 processor runs remaining threads.
- Allows processor affinity
  - bound thread blocks until the affined processor is free.