Processor Scheduling

H. Levy

Goals for Multiprogramming

- **In a multipogramming system, we try to increase utilzation and thruput by overlapping I/O and CPU activities.**
- **This requires several os *policy* decisions:**
  1. determine the *multipogramming level* -- the number of jobs laoded in primary memory.
  2. decide what job is to run next to guarntee good service
- **These decisions are long-term and short-term scheduling decisions, respectively.**
- **Short-term scheduling executes more frequently, changes of multiprogramming level are more costly.**

Scheduling

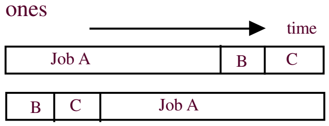
- **The *scheduler* chooses pocesses to run from the ready queue.**
- **The *scheduling algorithm* determines how jobs are scheduled.**
- **In general, the scheduler runs:**
  1. when a process switches from runing to waiting
  2. when an interrupt occurs
  3. when a process is created or terminated
- **In a *preemptive* system, the scheduler can interrupt a process that is running.**
- **In a *non-preemtive* system, the scheduler waits for a running process to explicitly block.**

Scheduling Algorithms

- **There are many possible criteria for evaluating a scheduling algorithm:**
  - CPU utilization
  - thruput
  - turnaround time
  - waiting time
  - response time
- **In an interactive system, predictability may be more important than a low average but high variance.**

Scheduling Algorithms

- **<u>First-come First-served</u> (FCFS) (FIFO)**
  - jobs are scheduled in order of arrival to ready Q
  - typically non-preemptive
- **problem:**
  - avg waiting time can be large if small jobs wait behind long ones



  - may lead to poor overlap of I/O and CPU

Scheduling Algorithms

- **<u>Shortest Job First</u> (SJF)**
  - choose the job with the smallest (expected) CPU burst
  - proveably optimal min. avg. waiting time
- **Problem:**
  - impossible to know size of CPU burst (but can try to predict from previous activity)
- **Can be either preemptive or non-preemptive.**
- **Preemptive SJF is called *shortest remaining time first***

Scheduling Algorithms

- **Priority Scheduling**
  - choose next job based on priority
  - for SJF, priority = expected CPU burst
  - can be either preemptive or non-preemptive
- **Problem:**
  - starvation: jobs can wait indefinitely
- **Solution to starvation**
  - age processes: increase priority as a function of waiting time

Scheduling Algorithms

- **Round Robin**
  - used for timesharing in particular
  - ready queue is treated as a circular queue (FIFO)
  - each process is given a time slice called a *quantum*
  - it is run for that quantum or unti it blocks
- **Problem**
  - context switch overhead of frequent context switch.

Scheduling Algorithms

- **Multi-level Queue**
- **Implement multiple ready Qs based on job "type"**
  - interactive process
  - CPU-bound process
  - batch job
  - system processes
  - editing session
- **Different Qs may be scheduled using different algorithms**
- **Jobs move from Q to Q (feedback)**
- **Very general algorithm, but need to select parameters for: # of queues, sched. algs., when to upgrade a job, when to downgrade a job, when to start a new job.**