

Processor related OS issues

Processor Scheduling

- Why do we need to do that?
- What does a program do?
- Goals of processor scheduling
 - provide good service
 - response time
 - throughput

Scheduling: First Principles

- Unit of scheduling
 - process or job
- What is a program?
- What is a process?
- What is a scheduler?
- Scheduler Classification
 - non-preemptive
 - preemptive
 - resume
 - restart

- What is the input to the scheduler program?
- Sources of priority
 - static properties
 - running time, memory, I/O
 - dynamic properties
 - available memory, memory requirements
 - urgency, \$\$\$\$
- Explicit info to the scheduler
- Implicit info inferred by the scheduler

Data Structures

- Process control block (PCB)

- state of a running program

```
enum state_type {new, ready, running, waiting, halted};  
typedef struct _control_block {  
    enum state_type state;  
    address PC;  
    int reg_file[NumRegs];  
    struct control_block *next_pcb;  
    int priority;  
    address page_table;  
    .....  
} control_block;
```

- Scheduling queues
 - ready queue
 - device queues
- Environments
 - dedicated
 - multiprogramming
 - interactive or time-sharing
- Types of schedulers
 - long term scheduler: multiprogramming
 - short term scheduler: time-sharing
 - medium term scheduler: reduce thrashing
 - dispatcher: transfer control to a selected process

Scheduling Algorithms

- Metrics
 - CPU utilization
 - throughput
 - turnaround time
 - waiting time
 - response time

Scheduling Algorithms

- FCFS
 - non-preemptive
 - process has to voluntarily relinquish by terminating or by making a blocking system call
 - intrinsic property used: arrival time
 - expected performance
 - no starvation
 - variance in turnaround time
 - convoy effect: CPU followed by I/O bursts
=> poor utilization

- SJF
 - non-preemptive
 - intrinsic property: CPU burst time required
 - how to know this?
 - better response for short jobs
 - provably optimal
 - min avg. waiting time for a pool of jobs

- Priority
 - extrinsic property: associate priority with each process
 - one queue for each priority level
 - FCFS within each level
 - SJF is a special case of this where
$$p = 1/\text{burst-time}$$
 - problem with priority?
 - solution?
 - increase priority with aging

- Preemptive algorithms
 - FCFS intrinsically non-preemptive
 - SJF and Priority algorithms can be made preemptive
 - how?
 - SJF with preemption referred to as “shortest remaining time first”
- Scheduler for time-sharing systems
 - preemptive or non-preemptive?

- Round robin
 - preemptive
 - good for time-sharing systems
 - when to preempt?
 - processor assigned in “time quantum” units
 - FCFS is a special case of RR
 - $q = \text{infinity}$
 - processor sharing
 - $q = 1$
 - as though every process on a processor $1/n$ speed
 - balancing q with context switch time

Algorithms:

select:

get head of queue;set timer;

dispatch;

timer interrupt:

save context and enqueue in ready queue;

goto select;

I/O request:

save context and enqueue in device queue;

goto select;

I/O completion:

move context from device to ready queue;

termination:

free context; goto select;

Multilevel Queues

- ready queue partitioned
- each queue has its own scheduling alg.
 - E.g. foreground (interactive) and background (batch) queues
- time slice between the queues
- add feedback to the queues
 - why?
 - adjust priority dynamically
 - move jobs between the queues

Evaluation

- modeling
- simulation
- implement it and test it on a job mix