OPERATING SYSTEMS
CSCI-SHU215

Lecture 06 – Scheduling

2

Fundamental Concepts

Running a Program

3

1. Program = sequence of instructions
2. Task = [1] + data
3. Process = [2] + execution context (PCB)

Monoprogramming

Process retains the processor until its termination

Sharing the Processor

4

CPU–I/O Cycle

The lifecycle of a process consists of a sequence of cycles

[CPU execution + I/O wait]

Distribution of the CPU cycles

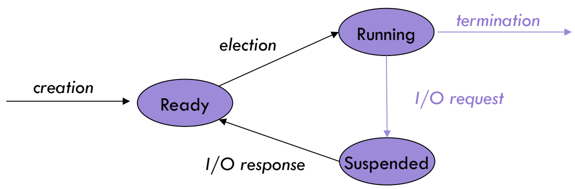
Multiprogramming + DMA => Maximum CPU usage

Multiprogramming

Batch processing

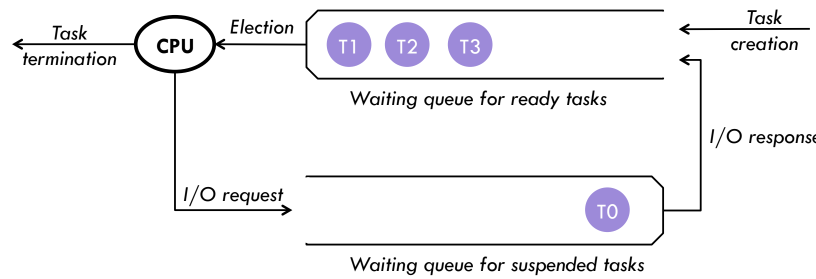
The active process hands over control:

- upon termination
- upon requesting an I/O



Multiprogramming

Batch processing scheduler

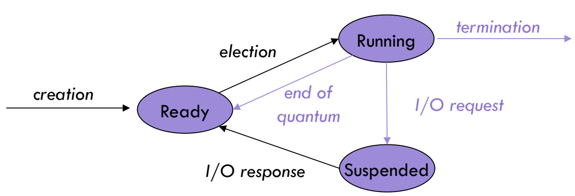


Multiprogramming

Time-sharing

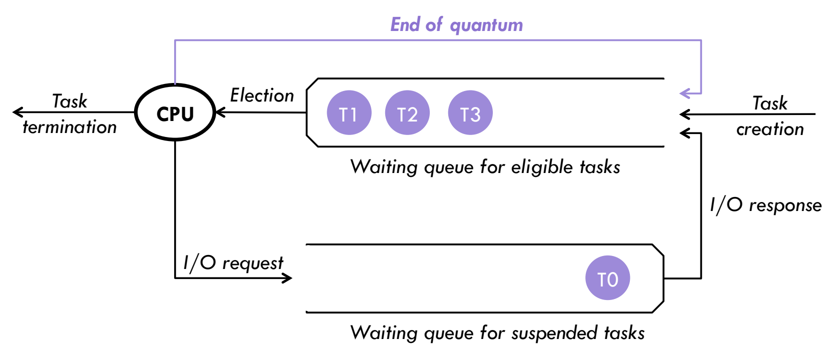
The active process hands over control:

- upon termination
- upon requesting an I/O
- upon spending its quantum



Multiprogramming

Time-sharing scheduler



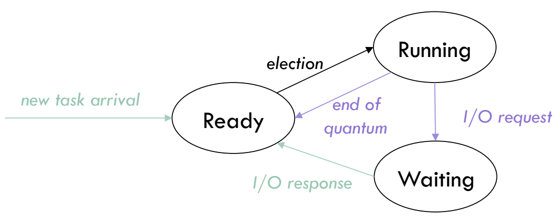
CPU Scheduler

- Elects one of the processes that are ready in memory

  Allocates the CPU to one of them

- Triggers of an election

additional triggers: preemptive scheduling



Dispatcher

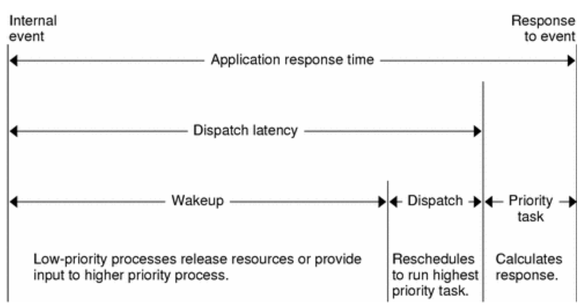
Hands over CPU to the elected process

1. Context switch
2. Shift to user mode
3. Restart execution from PC

Dispatch induces overhead

Dispatch Latency

Dispatch latency describes the amount of time a system takes to respond to a request for a process to begin operation.



Scheduling Criteria

- CPU utilization
- Throughput

  # of processes completed per unit of time
- Turnaround time

  Time required for a particular process to complete, from submission time to completion (*wall clock time*)
- Waiting time

  Time spent by a process in the Ready queue
- Response time

  Time taken in an interactive program from the issuance of a command to the commence of a response to that command

Scheduling Policy Objectives

Aim towards optimal satifaction of criteria

- In particular:
  - Guarantee fairness (*not equality*!)
  - prevent famines

Many algorithms

- *eg. FIFO, SJF, RR, EDF, RMS, …*
- Choosing the right algorithm depends on system usage (number of tasks, types, insertion rate, …)

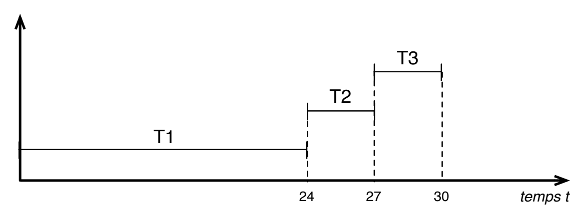
Optimization Criteria

- Maximum CPU utilization
- Maximum throughput
- Minimum turnaround time
- Minimum waiting time
- Minimum response time

Common Scheduling Policies

First Come, First Served (FCFS *ou* FIFO)

Process	CPU time	$t_{insertion}$ (order)	$T_{response}$	T_{wait}
P_1	24	0(1)	24	0
P_2	3	0(2)	27	24
P_3	3	0(3)	30	27



- Response time ($T_{response}$) = t_{end} - $t_{insertion}$
- Waiting time = [$T_{response}$] – [Computation time]

Average waiting time: (0 + 24 + 27)/3 = 17 Convoy Effect!

FCFS Scheduling

Let's change the order of insertions

Process	Comp. Time	$t_{insertion}$ (order)	$T_{response}$	T_{wait}
P_1	24	0(3)	30	6
P_2	3	0(1)	3	0
P_3	3	0(2)	6	3

Average waiting time: $(6 + 0 + 3)/3 = 3$

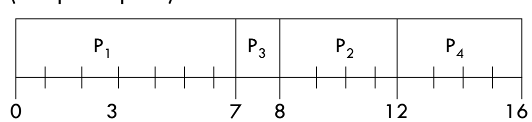
Shortest Job First (SJF)

- Principle
  - Predict CPU time of each process
  - Elect the process with the shortest CPU time
- Two startegies
  - Non preemptive
    - Once a process obtains the CPU, it will only release it upon termination
  - Preemptive, or *Shortest-Remaining-Time-First (SRTF)*
    - Upon insertion of a new process, switch to the latter if its CPU time is shorter than that of the running process
- SJF is optimal
  - Guarantees that average waiting time is minimal

Example of Non-Preemptive SJF

Process	Insertion Time	CPU Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (non preemptive)

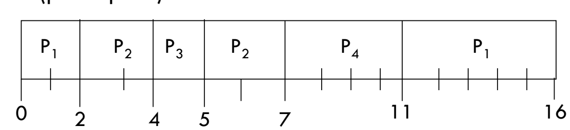


- Average waiting time = $(0 + 6 + 3 + 7)/4 = 4$

Example of Preemptive SJF

Process	Insertion Time	CPU Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SRTF (preemptive)



- Average waiting time = $(9 + 1 + 0 + 2)/4 = 3$

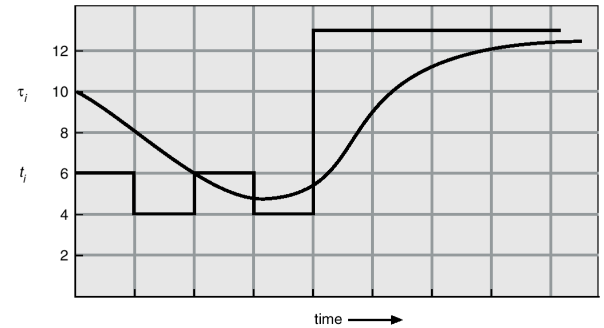
Prediciting the Next Computation Time

One simple, fast, and relatively accurate method is the exponential average

t_n	duration of the n^{th} CPU burst
τ_{n+1}	prediction for the duration of the next burst
α	weighting factor ($0 \le \alpha \le 1$)

$$\tau_{n+1} = \alpha\ t_n + (1 - \alpha)\ \tau_n$$

Prediciting the Next Computation Time



CPU burst (t_i)		6	4	6	4	13	13	13	...
"guess" (τ_i)	10	8	6	6	5	9	11	12	...

Priority Scheduling

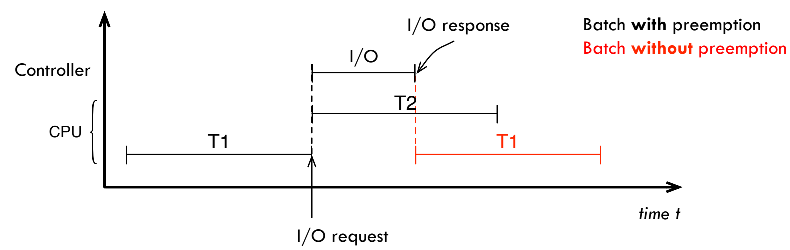
- A priority (integer > 0) gets assigned to each process
- Scheduler allocates the CPU to the process with the highest priority
  - Preemptive
  - Non preemptive
- SJF is a priority-oriented algorithm

  Priority value is the expected CPU time
- Problem: Famine

  Low priority processes may never run
- Solution: Graceful Ageing

  Priority of processes in ready queue increases periodically

Priority Scheduling



Round Robin (RR)

Principle

Each process gets the CPU for a limited duration

quantum: generally 10-100 milliseconds

When it has spent its quantum, a process goes back to the Ready queue

Fairness

Consider n processes in the Ready queue, let the quantum value be q

No process ever waits more than $(n-1)q$

Performance tradeoff

- q is large $\Rightarrow$ FIFO
- q is small (close to duration of context switch) $\Rightarrow$ high overhead

Example of RR with Q = 20

Process	CPU time
P_1	53
P_2	17
P_3	68
P_4	24

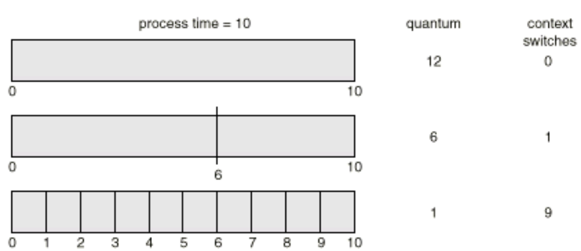
Gantt diagram:

P_1	P_2	P_3	P_4	P_1	P_3	P_4	P_1	P_3	P_3

0 20 37 57 77 97 117 121 134 154 162

Typically induces a higher average turnaround time than SJF, but a smaller *response time*

Quantum vs. Switch

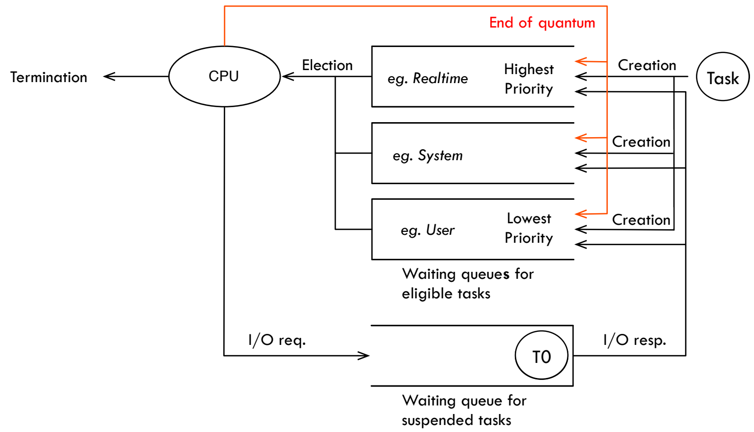


Multilevel Queues

- Separate waiting queues
  - foreground (interactive)
  - background (batch)
- Each queue enforces a different scheduling policy
  - foreground – RR
  - background – FCFS
- Requires inter-level scheduling
  - Fixed priority scheduling
    - ie. serve all processes of FG queue, then those of BG queue
    - => May induce famine
  - Time slice
    - Each queue gets a percentage of the available CPU time
    - eg. 80% for FG with RR and 20% for BG with FCFS

Multilevel Queue Scheduling

29



Multilevel Feedback Queues (MFQ)

30

A process can be moved to another queue

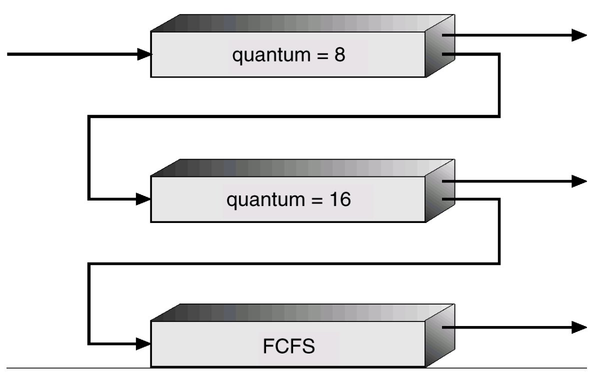
Implementation of ageing

Required settings for MFQ

- # of queues
- Scheduling policy for each queue
- Formula for promoting a process to a higher priority queue
- Formula for demoting a process to a lower priority queue
- Formula for assigning a queue to a new process

Example of MFQ Scheduling

31



32

Extended Scheduling Policies

Multicore Scheduling

- CPU scheduling is more complex
- *Homogeneous processors*
- *Load sharing*
- *Asymmetric Multiprocessing*

  One processor manages all OS structures

  No data sharing

Realtime Scheduling

- Common misconception: real time does not mean fast!
  - Real-time processes have timing constraints
  - Expressed as deadlines or rate requirements
- *Hard Realtime Systems*

  Guarantee that every process respects deadlines/rates
- *Soft Realtime Systems*

  May miss deadlines, but QoS then degrades

Common RT Scheduling Policies

- Rate monotonic
  - Just one scalar priority related to the periodicity of the job
  - Priority = 1/rate
  - Static
- Earliest deadline first (EDF)
  - Priority = deadline
  - Dynamic but more complex
- Both require admission control to provide guarantees

Linux Scheduling

Two policies: time sharing and real time

Time sharing
- Priority based on credits

  processus with the largest amount of credits gets elected
- Credit subtraction upon every clock tick
- When credit = 0, election of another process
- General recreditation when all processes are down to 0 credit
  - Per process recreditation based on priority and history

Real time
- Soft real time
- Posix.1b – two classes
  - FCFS and RR
  - Process with highest priority runs first

Linux Scheduling