CPU Scheduling

Short term scheduling

- Short-term scheduler (CPU scheduler) — determines priorities for ready processes and chooses a process (of highest priority) for execution.
- Dispatcher — prepares the chosen process for execution and starts it (does a context switch).
- Priority function — determines current priority of a process given the processes parameters and state of the system. Arguments to the function may be chosen from process state parameters and system state parameters. Process priority in a given time is the value of priority function for the process and system state parameters in that given time. Arguments to the priority function may include:
    - Waiting time — the time the given process spent in ready state
    - CPU time — current total time of execution for the process
    - Elapsed time — current total time since admission (CPU time + Waiting time + time in the waiting state)
    - Time to deadline — the time before which process must finish execution (in real-time systems).
    - External priority — importance of that process (niceness)
    - Memory requirements (mostly for batch systems)
    - System load — number of processes in the system, memory usage
- Arbitration rule — „breaks ties" when the priority functions yields the same value for more than one processes with the highest priority. The possible arbitration rules:
    - Random
    - Cyclic — processes get the CPU cyclically
    - Chronologically — with respect to admission order (FIFO, LIFO)

    - FIFO — in the getting-ready order (enqueuing on the ready queue)
- Decision mode — specifies the instants in time at which the new process to run is selected. There are two general categories:
    - Nonpreemptive — In this case, once a process is in the running state it continues to execute until it terminates or until it blocks itself to wait for I/O or to request some OS service.
    - Preemptive — The currently running process may be interrupted and moved to the ready state by the OS. The decision to preempt may be performed when a new process arrives, when an interrupt occurs that places a blocked process in the ready state, or periodically, based on a clock interrupt.

Scheduling criteria

- Efficiency:
    - Processor utilization — percentage of time with CPU allocated to user processes
    - Throughput — the number of processes completed per time unit
    - Turnaround time — time between task submission and its termination
    - Response time — time between request submission and response generation start
    - Delay time — time from deadline to task completion
- Other aspects:
    - Fairness with respect to processes
    - External priorities preservation
    - Resource utilization balance
    - Predictability — task execution does not increase drastically with system load

Nonpreemptive scheduling algorithms

FCFS — First Come First Served

- The simplest CPU-scheduling algorithm.
- With this scheme the process that request the CPU first is allocated the CPU first.
- This scheme implements the ready queue as a FIFO queue (normal queue). The processes are added to the end of the queue and when the CPU is free the process at the head of the queue is run.

LCFS — Last Come First Served

- In this scheme, the process that most recently entered the ready queue is allocated the CPU next.
- This scheme implements the ready queue as a LIFO stack. The processes are added to the top of the stack, and when the CPU is free, the process at the top of the stack is run.

SJF (SJN, SPF, SPN) — Shortest (Job/Process) (First/Next)

- This algorithm associates with each process the length of the process's next CPU burst.
- CPU bust it the time a process will spend on a CPU continuously. This is not the same as the total time it will take to finish it's job.
- When the CPU is available, it is assigned to the process that has the smallest next CPU burst. If the next CPU bursts of two processes are the same, FCFS scheduling is used to break the tie.

Preemptive scheduling algorithms

Round Robin (RR)

- This algorithm is similar to FCFS scheduling, but preemption is added to enable the system to switch between processes.
- A small unit of time, called a time quantum or time slice if defined.
- To implement RR scheduling, we again treat the ready queue as a FIFO queue of processes. New processes are added to the tail of the ready queue.
- The CPU scheduler picks the first process from the ready queue, allocating the CPU to each process for a time interval of up to 1 time quantum.
- If the CPU burst of a process is less than 1 quantum the process itself with release the CPU voluntarily. If the CPU burst is longer than 1 quantum then the timer will go off, a context switch will happen and the process will go the the end of the queue.
- Short quantum reduces turnaround time of short processes at the expense of longer processes, but increases context switch overhead.
- To improve interaction with user, time quantum should be slightly greater than the CPU time necessary for interaction.

SRT (Shortest Remaining Time)

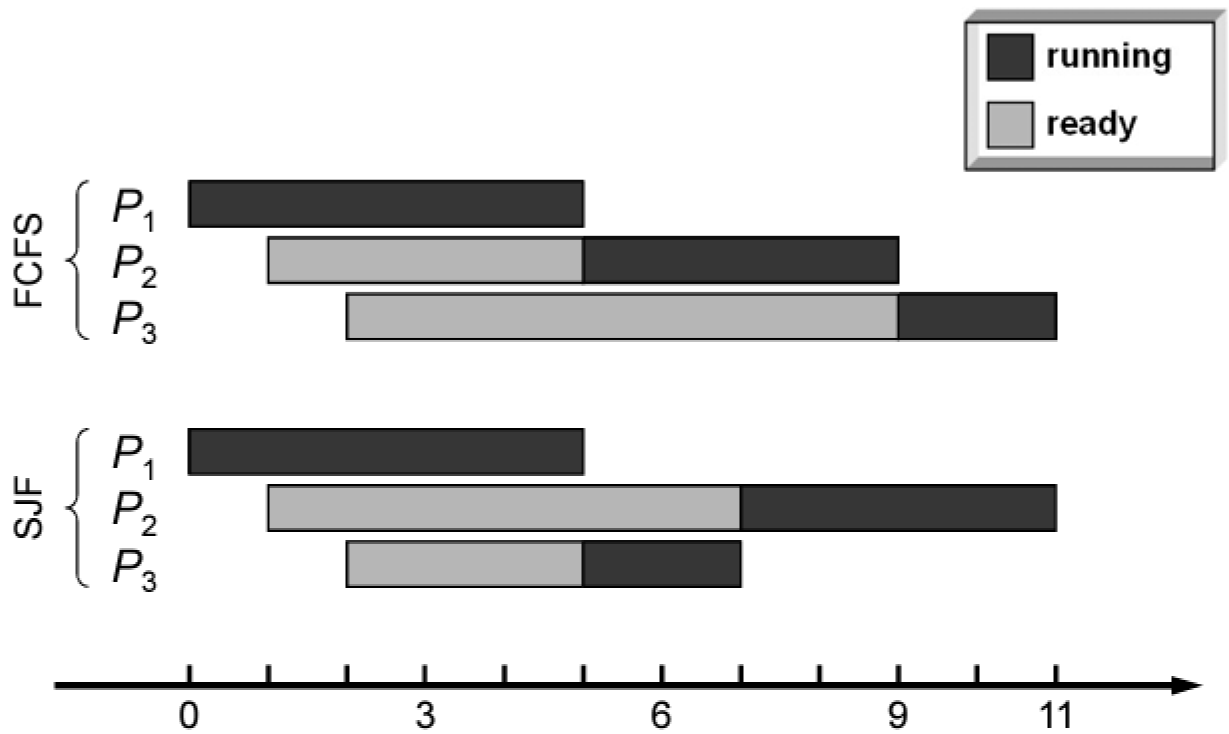
- This is a preemptive version of SJF.
- The process with the smallest amount of time remaining until completion is selected to execute.
- The processes will always run until they complete, or until a new process is added that requires a smaller amount of time to complete.

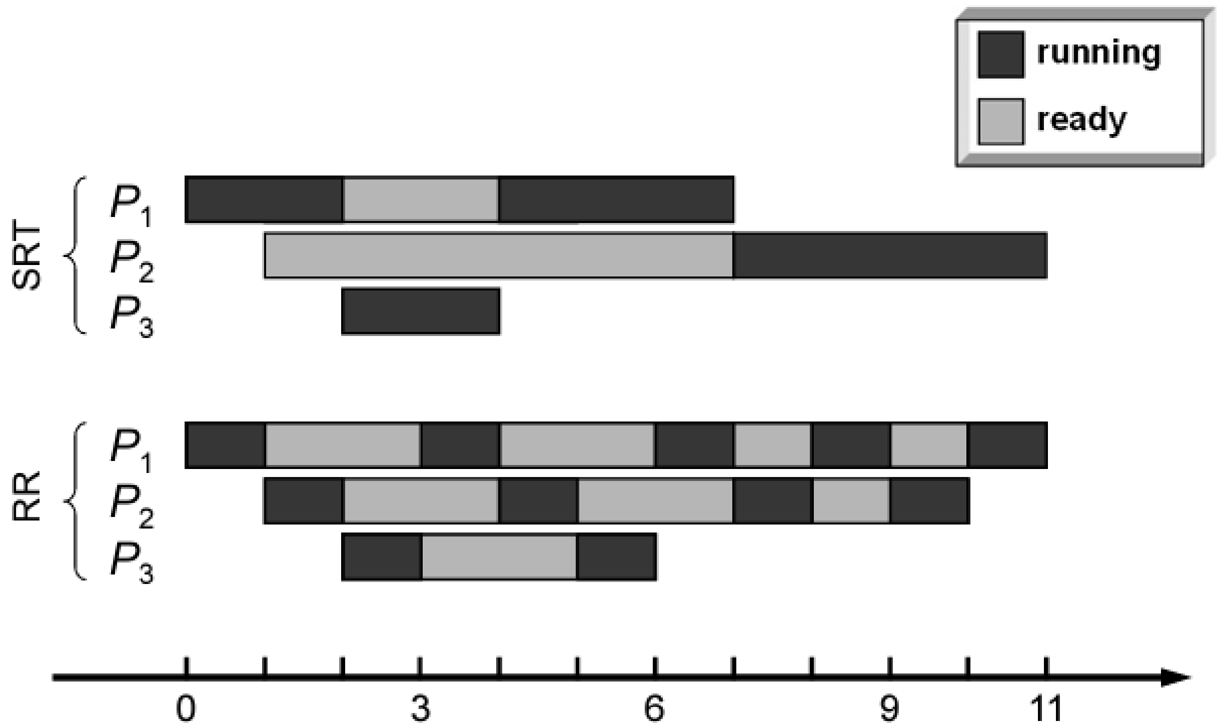
Classification of scheduling algorithms

- We define three parameters that can dictate the priority of processes:
    - a — CPU time
    - r — Elapsed time
    - t — Time remaining to complete

Algorithm	Priority	Decision mode	Arbitration rule
FCFS	r	Nonpreemptive	Random
LCFS	-r	Nonpreemptive	Random
SJF	-t	Nonpreemptive	Random or chronological
SRT	a-t	Preemptive	Random or chronological
RR	constant	Preemptive	Cyclic

Examples of scheduling





Other scheduling algorithms

- Priority scheduling — based on external priority
- Multi-queue scheduling — there are multiple queues and each one can be handled differently
- Deadline scheduling — example: Earliest Deadline First

Priority queue

- A priority queue is a multi-level queue z feedback in which every level corresponds to the priority level or a range of priorities.
- To put a process in a priority queue we need to determine the correct level of the queue, which corresponds to the priority of the new process. Then this process is put at the end of this queue.
- When we want to schedule a process to run on the CPU we choose the first non-empty queue with the highest priority and we take the first process at its head.