CS240 Operating Systems, Communications and Concurrency

Process Scheduling Overview

Multitasking Concepts
Multiprocessing Concepts
Scheduling Algorithm Performance Criteria
Analysis of FCFS vs SJF

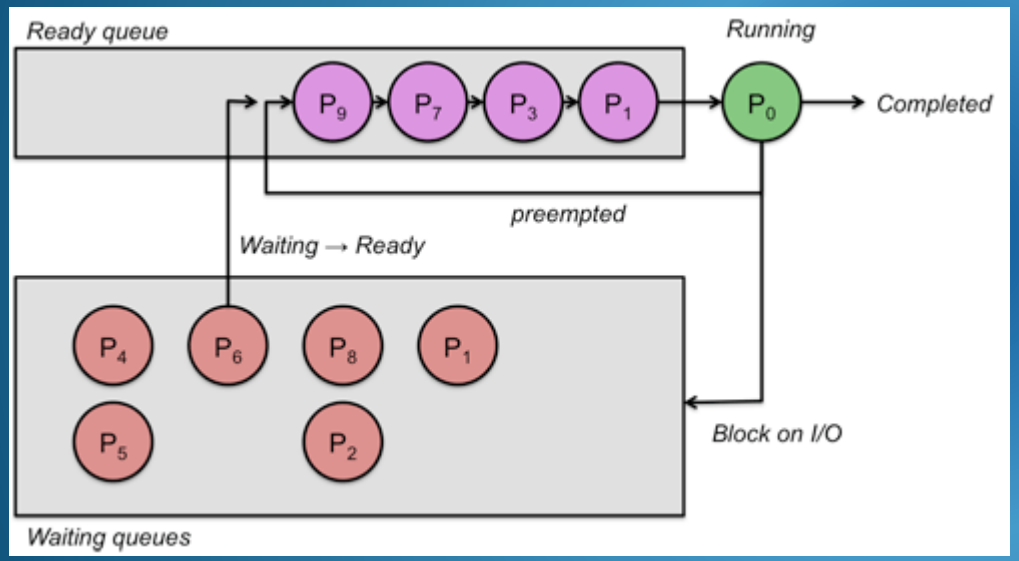
CS240 Operating Systems, Communications and Concurrency

Multitasking

A multitasking operating system allows multiple processes to be resident and active on the computer system at one time.

A multiuser multitasking system, some of these processes may belong to different users.

CS240 Operating Systems, Communications and Concurrency



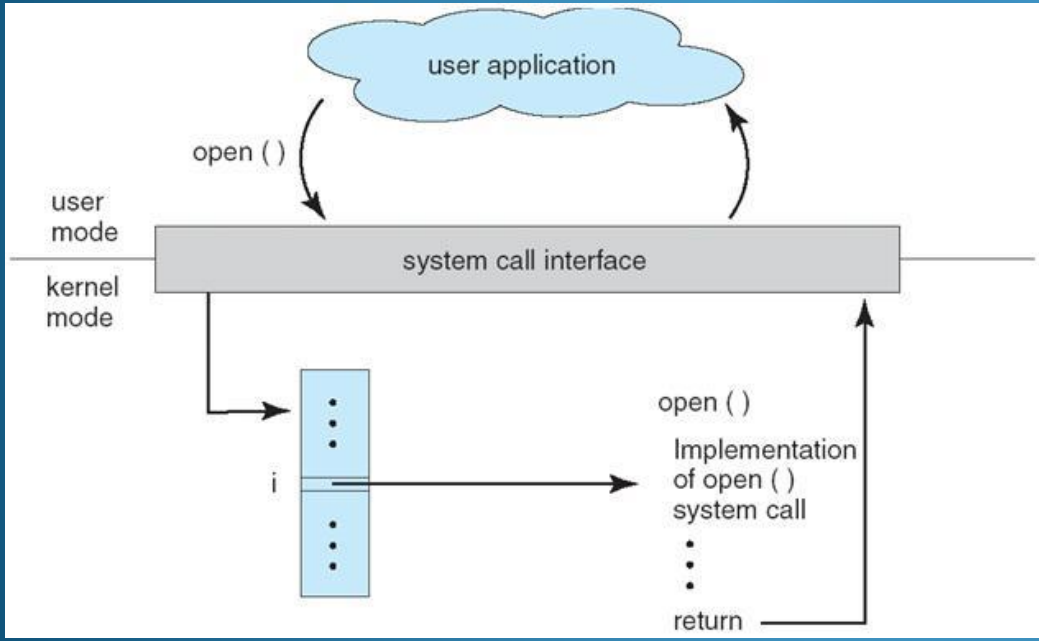
Process Scheduling

The number of processes in a multitasking system is generally much greater than the number of processors.

Each process must compete with others for the available resources.

CPU time is valuable and the operating system must decide which processes are assigned to use the processor(s) and for how long.

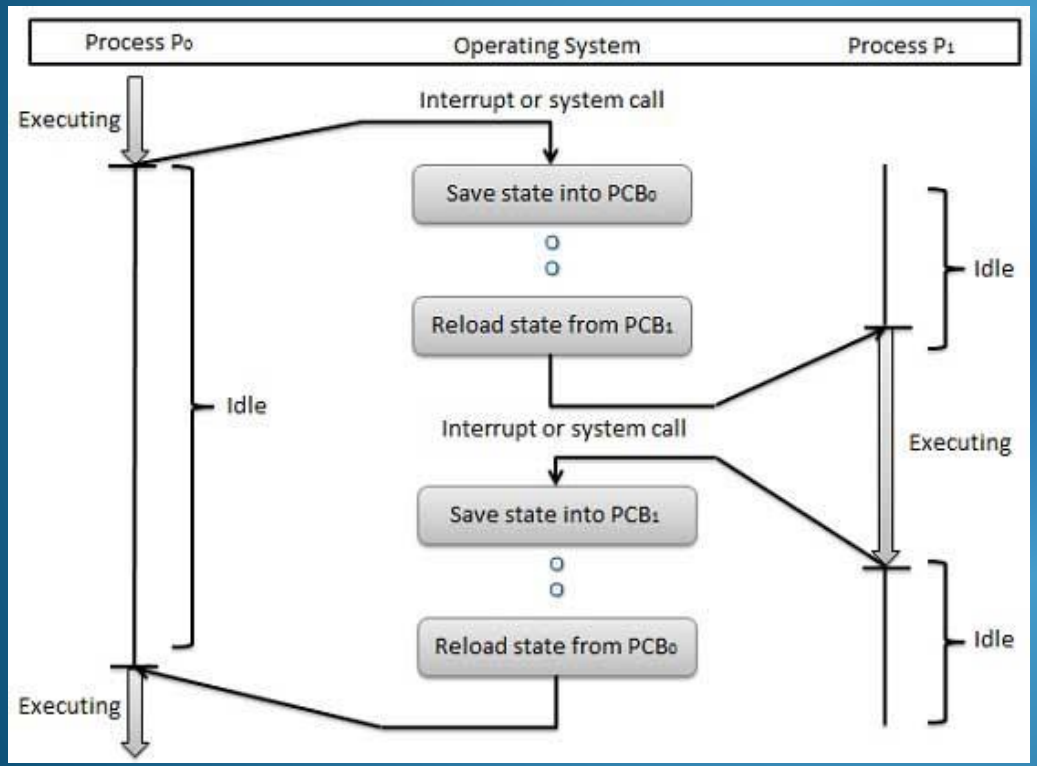
CS240 Operating Systems, Communications and Concurrency



OS Control Mechanism

When a process makes a system call or when its time quantum on the processor expires, the resulting software or hardware interrupt causes the CPU to stop fetching instructions from the currently assigned process and switch to designated code in the kernel for handling the interrupt.

CS240 Operating Systems, Communications and Concurrency

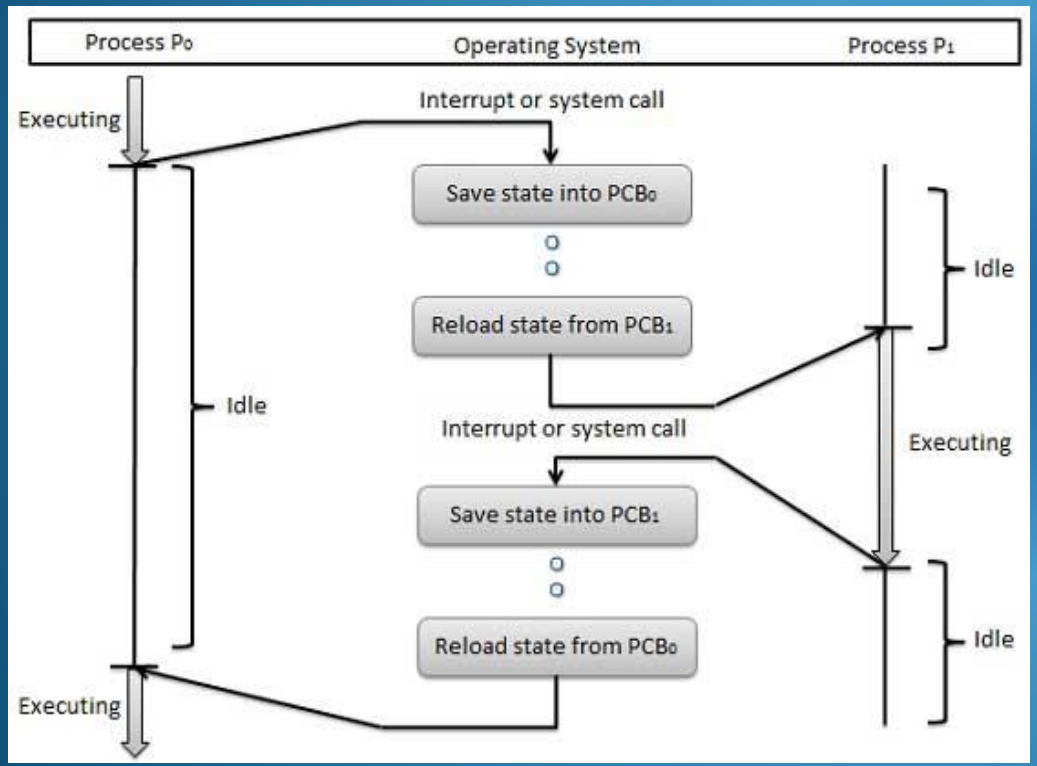


Scheduling Decision

The operating system may then decide to allocate the CPU to another process, in which case it will execute code to perform a context switch.

It saves the run-time state of the current process in the process control block so that it can be continued later, and then executes dispatcher code to load the run-time state of the chosen process for the CPU to continue executing instead.

CS240 Operating Systems, Communications and Concurrency



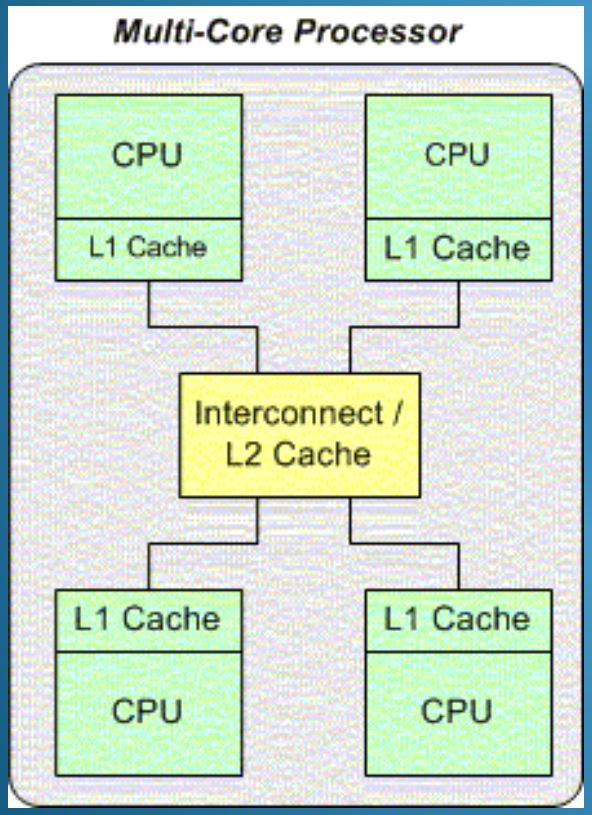
Context Switch

Context Switch code must be efficient

Usually supported by hardware instructions

Incurs Performance Overhead, (CPU is spending time executing operating system code)

CS240 Operating Systems, Communications and Concurrency

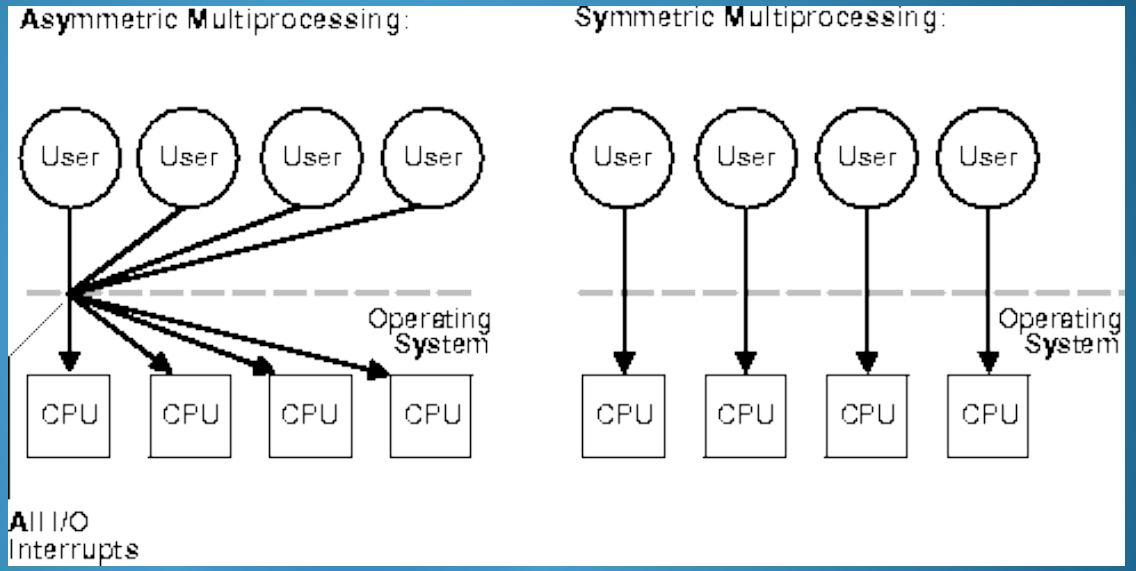


Multiprocessing

Multiprocessing is the use of more than one CPU in a system, usually separate physical CPUs but the idea may also apply to a single CPU with multiple execution cores.

CS240 Operating Systems, Communications and Concurrency

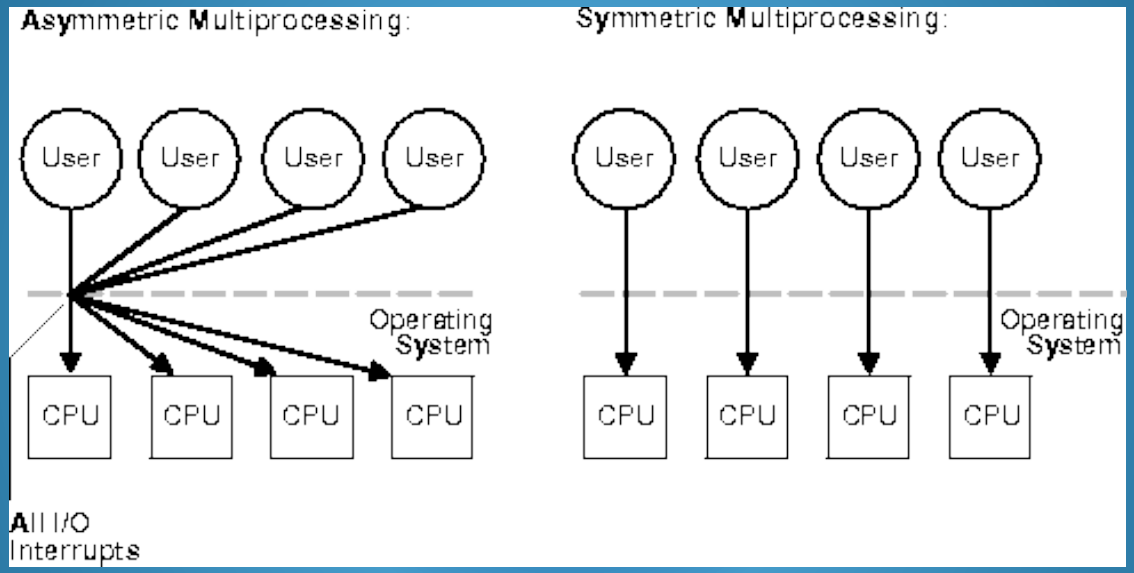
Coordinating Multiprocessing Activity



With **asymmetric multiprocessing**, one processor, the master, centrally executes operating system code and handles I/O operations and the assignment of workloads to the other processors which execute user processes. With this scheme, only one processor is accessing system data structures for resource control.

While this makes it easier to code the operating system functions, in small systems with few processors the master may not have enough slaves to keep it busy, so maximum hardware performance is not achieved.

CS240 Operating Systems, Communications and Concurrency



Symmetric multiprocessing is a system where all processors carry out similar functions and are self scheduling. The identical processors use a shared bus to connect to a single shared main memory and have full access to all I/O devices and are controlled by a single operating system instance.

Each processor will examine and manipulate the operating system queue structures concurrently with others when selecting a process to execute. This access contention must be programmed carefully to protect the integrity of the shared data structures.

CS240 Operating Systems, Communications and Concurrency

Process Scheduling Algorithms

In this section, we are going to focus on servicing the queue associated with processes waiting to be assigned to processor(s), also known as the "Ready" queue.

These processes are ready for their instructions to be executed in contrast to others that may be waiting for I/O operations.

Later we look at approaches to scheduling requests for reading hard disks.

CS240 Operating Systems, Communications and Concurrency

Process Scheduling Algorithms

Some evaluation criteria are necessary to allow us to compare different queuing policies and their effects on system performance.

Processor Utilisation = (Execution Time) / (Total Time)
Throughput = Jobs per Unit time
Turnaround Time = (Time job finishes) - (Time job was submitted)
Waiting Time = Time doing nothing in a queue
Response Time = (Time job is first scheduled on cpu) - (Time job was submitted)

CS240 Operating Systems, Communications and Concurrency

The order in which processes are serviced from a queue can be described by a *Gantt Chart*. It represents a scheduling sequence.

Each process receives a number of units of time on a particular resource in the order described.

Simpler format to describe a scheduling sequence on a processor

CS240 Operating Systems, Communications and Concurrency

Scheduling algorithms can be *preemptive* or *non-preemptive*.

All multitasking systems would employ preemptive scheduling as otherwise, one process might never give up control of the CPU.

We will compare some scheduling algorithms using deterministic modeling, i.e. we will define a workload pattern and compare the results across different algorithms.

CS240 Operating Systems, Communications and Concurrency

First let's look at some **non-preemptive** algorithms.

FCFS (First Come First Served)

SJF (Shortest Job First)

CS240 Operating Systems, Communications and Concurrency

Consider the following, where three jobs arrive in the following order and a scheduling decision must be made:- The CPU Burst Time is the immediate period of execution time required by each task on the CPU. Note this would not normally be known in advance.

Job	CPU (Burst Time)
1	24
2	3
3	3

For academic purposes let's say the burst time is known, in advance of the scheduling decision.

CS240 Operating Systems, Communications and Concurrency

Job	CPU (Burst Time)
1	24
2	3
3	3

With the FCFS algorithm we get the following order:-

CS240 Operating Systems, Communications and Concurrency

FCFS gives the following performance results:-

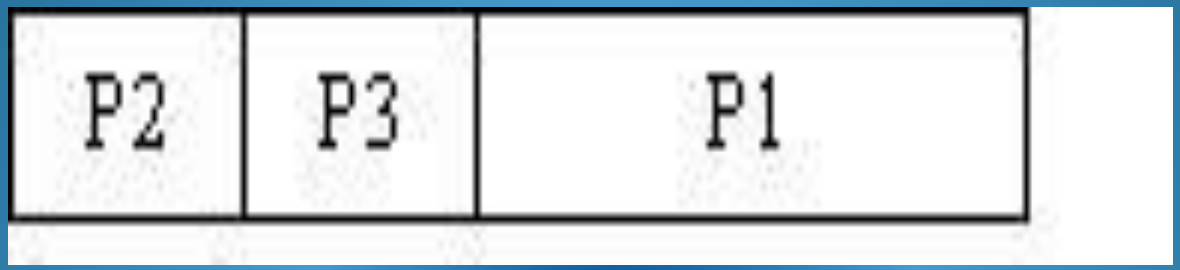
Job	Waiting Time	Response Time	Turnaround Time
1	0	0	24
2	24	24	27
3	27	27	30
Average	17	17	27

CS240 Operating Systems, Communications and Concurrency

Job	CPU (Burst Time)
1	24
2	3
3	3

With the SJF algorithm we get the following order

CS240 Operating Systems, Communications and Concurrency



SJF gives the following performance results:-

Job	Waiting Time	Response Time	Turnaround Time
1	6	6	30
2	0	0	3
3	3	3	6
Average	3	3	13

CS240 Operating Systems, Communications and Concurrency

Analysis of non-preemptive algorithms used

FCFS is an inherently fair algorithm but performs badly for interactive systems where the response time is important and in situations where the job length differs greatly.

SJF is provably the optimal algorithm in terms of throughput, waiting time and response performance but is not fair.

SJF favours short jobs over longer ones. The arrival of shorter jobs in the ready queue postpones the scheduling of the longer ones indefinitely even though they may be in the ready queue for quite some time. This is known as starvation.

CS240 Operating Systems, Communications and Concurrency

Estimating CPU Burst Length for SJF

In practice it is not possible to know the CPU burst length of jobs in advance but this can be estimated using a smoothing formula (a heuristic guess) based on its historical performance as follows:-

Let t_n = length of burst n
Let T_n = average of previous n bursts
The next burst t_{n+1} of a process may be estimated from the formula:-

$$t_{n+1} = at_n + (1-a)T_n$$

where $0 <= a <= 1$

A chosen weighted value based on recent activity and past activity.

CS240 Operating Systems, Communications and Concurrency

Other aspects of non-preemptive SJF

SJF calculation with approximated CPU burst is more complex than FCFS

You must maintain cumulative history information and perform the calculations required for predicting burst length each time you come to choose the next task.

CS240 Operating Systems, Communications and Concurrency

Need Other Algorithms

With FCFS, long jobs hold up short jobs.

While SJF solves this, a continual stream of short jobs will block long jobs indefinitely.

It may be better that the longer a job is in the system waiting for the CPU the greater chance it has of being scheduled.

CS240 Operating Systems, Communications and Concurrency

Non preemptive Algorithms (Continued)

The **(HRN) highest response ratio next** algorithm endeavours to meet this objective. Type of priority based algorithm.

The Response Ratio of a job is calculated as follows:-

Response Ratio = (Waiting Time) / (Service Time)

The Response Ratio determines the ordering of the jobs. As the job waits in the ready queue, its priority increases.