Preview

- Scheduler
  - Long-Term, Short-Term (CPU), Memory
- Scheduling Queues
- Preemptive or Non-preemptive Scheduling
- Scheduling Criteria
- CPU Scheduling (Shot-term Scheduler)
  - FCFS (First Come First Serve)
  - Shortest Job First
  - Shortest Remaining Time
  - Round Robin
  - Priority Queue
  - Guaranteed Scheduling
  - Lottery Scheduling

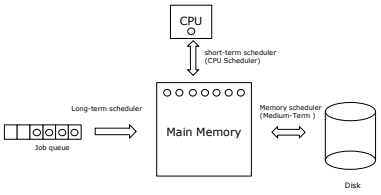
CPU Scheduling

- In a system with a single CPU, only one process can run at a time.
- If CPU is busy for serving a process, others must wait until the CPU become free.
- Once CPU become free, CPU scheduler will select a process from ready Queue and let it use CPU for some calculation.
- The objective of multiprogramming is to have some process running at all times, to maximize CPU utilization.
- The CPU scheduler will select an process from ready queue by scheduling algorithm.
- The ready queue can be implemented as a FIFO queue, priority queue, a tree or unordered linked list.
- The ready queue structure might save pointer to process tables for ready state processes.

Tree Level Scheduler

- Part of operating system that choose the next process for executing its job based on the rule (scheduling algorithm) is called process scheduler.
- Process scheduler is designed for making efficient use of the CPU.
- Level of Scheduler
  - Long-Term Scheduler – Selects a process from the job queue and load into memory for execution (only certain degree of multiprogramming is supported by a system, if more processes need to run, some number of processes must wait in job queue)
  - Short-term scheduler – selects a process from the ready queue and allocates the CPU.
  - Memory Scheduler – schedule which process is in memory and in disk (if memory size is not big enough to support certain degree of multiprogramming, some process might need be in the secondary memory).

Three Level Scheduler



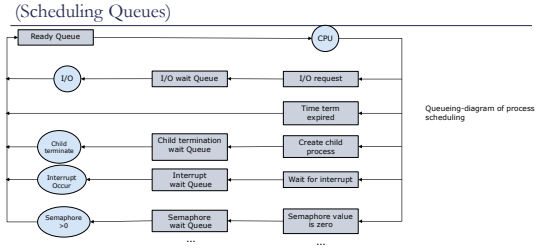
Process Scheduling

(Scheduling Queues)

- Once the process is allocated in the memory and executing on CPU, one of several events could occur:
  - The process could issue an I/O request and then be placed in an I/O wait queue.
  - The process could create a new child process and then be placed in a wait queue while it awaits the child's termination.
  - The process could be removed forcibly from the core, as a result of an interrupt or having its time slice expire, and be put back in the ready queue.
  - The process tries down semaphore but semaphore value is 0, and be placed in a wait queue for the semaphore.
  - …
- Once a CPU core becomes available, short term scheduler (CPU scheduler) need select one of process from the ready queue and let it run again!

Process Scheduling

(Scheduling Queues)



Queueing-diagram of process scheduling

Effective CPU Scheduler is essential

The effective process scheduling algorithm is essential since process switch(context switch) is very expensive.

1. All information for the blocked (become ready state or block state) must be saved (state, register, stack pointer, PC, ..)
2. The memory map (information regarding memory management) must be saved.
3. A new process is selected by the scheduler
4. Information for the new process must be loaded
5. start run new process.

Preemptive vs. Nonpreemptive Scheduling

When to make scheduling decision?

1. When a process switches from the running state to the blocking state, CPU become available! (short term scheduler)
2. When a process switches from the running state to the ready state, CPU become available! (short term scheduler)
3. When a process terminate its job, memory space become available for new process! (long term scheduler)
4. When a process switches from the blocked state to the ready state, CPU become available! (in case short term scheduler use preemptive scheduling)
   - When scheduling takes place only under circumstances 1, 3, we say nonpreemptive scheduling scheme;
   - otherwise, the scheduling scheme is called preemptive scheduling scheme. (possibly, short term scheduler might need select a process for any status changes)

- Under nonpreemptive scheduling, once the CPU has been allocated to a process, the process keeps the CPU until it releases it either by terminating or by switching to the block state.

Preemptive vs. Nonpreemptive Scheduling

- Preemptive scheduling without considering mutual exclusion can result in race conditions when data are shared among several processes
- Ex) two process P_1 and P_2 are updating shared data.
  - T_0: P_1 is currently updating shared data and P_2 is in block state
  - T_1: P_2 change state from block to ready.
  - T_1: P_1 is preempted by CPU shoulder before finishing update shared data.
  - T_2: P_2 is selected by the scheduler and start updating shared data.
- A preemptive kernel requires mechanisms such as mutex locks to prevent race conditions when accessing shared kernel data structures.

Scheduling Criteria

- Many criteria have been suggested for comparing CPU-scheduling algorithms.
  - **CPU Utilization** – want to keep the CPU as busy as possible. Conceptually, CPU utilization can range from 0 to 100 percent. In a real system, it range from 40 percent to 90 percent.
  - **Throughput** – the number of processes that are completed per time unit.
  - **Turnaround time** – The interval from the time of submission of a process to the time of completion.
  - **Waiting time** – Waiting the sum of the periods spent waiting in the ready queue.
  - **Response time** – the time from the submission of a request until the first response is produced
- It is desirable to maximize CPU utilization and throughput and to minimize turnaround time, waiting time, and response time.

Process Scheduling

(CPU Scheduling)

- A process migrates to the ready queue from various wait queues throughout its lifetime.
- The role of the short-term scheduler (the CPU scheduler) is to select from among the processes that are in the ready queue and allocate an available CPU.
- The short-term scheduler select a process from ready queue base on rules.
  - Shortest Job First
  - Shortest Remaining Time
  - Round Robin
  - Priority Queue
  - Guaranteed Scheduling
  - Lottery Scheduling

Scheduling Algorithms

(CPU Scheduling Algorithm: First-Come, First-Served)

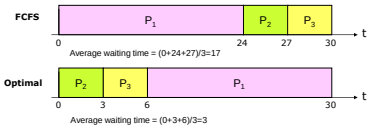
- It is simplest CPU-scheduling algorithm and non-preemptive.
  - Non-preemptive: Once the CPU has been allocated to a process, that process keeps the CPU until it releases the CPU, either by terminating, by requesting I/O or CPU time out. (termination, running ->block ,or running -> ready)
- The implementation of the FCFS policy is easily managed with a FIFO queue.
  - When a process enters the ready queue, its process table (process control block) is linked onto the tail of the queue.
  - When the CPU is free, it is allocated to the process at the head of the queue. The running process is then removed from the queue.
- Drawback of FCFS is that the average waiting time might be quite long.

Scheduling Algorithms
(CPU Scheduling Algorithm: First-Come, First-Served)

Process	Next CPU burst time
P_1	24
P_2	3
P_3	3

three processes are in the ready queue (or Job queue) to run.

$P_1 > P_2 > P_3$



Average waiting time = (0+24+27)/3=17

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

Scheduling Algorithms
(CPU Scheduling Algorithm: Shortest Job First)

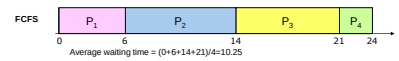
- Shortest Job First Algorithm associates with each process the length of the process's next CPU burst.
  - When the CPU is available, it is assigned to the process that has the smallest next CPU burst.
  - If the next CPU bursts of two processes are the same, FCFS scheduling is used to break the tie.
- Shortest Job First algorithm is provably optimal in that it gives the minimal average waiting time for a given set of processes.

Scheduling Algorithms
(CPU Scheduling Algorithm: Shortest Job First)

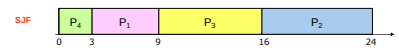
Process	Next CPU burst time
P_1	6 (msec)
P_2	8(msec)
P_3	7(msec)
P_4	3(msec)

Lets assume arriving times as (for FCFS)

$P_1 > P_2 > P_3 > P_4$



Average waiting time = (0+6+14+21)/4=10.25



Average waiting time = (0+3+9+16)/4=7

Scheduling Algorithms
(CPU Scheduling Algorithm: Shortest Job First)

- Although the SJF algorithm is optimal, it cannot be implemented at the level of CPU scheduling, as there is no way to know the length of the next CPU burst.
- One approach to this problem is to try to approximate SJF scheduling based on previous CPU burst time.
  - The next CPU burst is generally predicted as an exponential average of the measured lengths of previous CPU bursts.
  - Let t_n be the length of the n^{th} CPU burst, and let τ_{n+1} be our predicted value for the next CPU burst. Then, for α, $0 \le \alpha \le 1$, define

    $$\tau_{n+1} = \alpha\, t_n + (1 - \alpha)\, \tau_n$$
  - The parameter α controls the relative weight of recent and past history in our prediction.

Scheduling Algorithms
(CPU Scheduling Algorithm: Shortest Job First)

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

t_n :length of the n^{th} CPU burst, τ_{n+1} : predicted next CPU burst

- $\alpha = 0$: $\tau_{n+1} = \tau_n$:recent history has no effect
- $\alpha = 1$: $\tau_{n+1} = t_n$:the most recent CPU burst matters.
- With $\alpha = ½$ and $\tau_0 = 10$, we can estimate sequence of predicted next CPU burst

CPU burst ($t_{i,x,0}$)		6	4	6	4	13	13	13	..
Predicted τ_n	10	8	6	6	5	9	11	12	..

Scheduling Algorithms
(CPU Scheduling Algorithm: Shortest Remaining Time First)

- Preemptive version of the SJF algorithm is Shortest Remaining Time First algorithm.
- When a new process is arrives at the ready queue while previous process is executing, CPU is preempted from the process and change to ready state.
- Short term scheduler select a process from ready queue.
- The newly arrived process might be shorter than what is left of the current executing process.

Scheduling Algorithms
(CPU Scheduling Algorithm: Shortest Remaining Time First)

Process	Arrival Time	Next CPU burst time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

Average waiting time = ((10 – 1) + 0 + (17 – 2) + (5 – 3))/4 = 6.5

Average Turnaround time = ((17- 0) + (5-1) + (26-2)+(10-3))/4 = 13

Scheduling Algorithms
(CPU Scheduling Algorithm: Round-Robin Scheduling)

- The round-robin (RR) scheduling 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) is defined. A time quantum is generally from 10 to 100 milliseconds in length.
- In the RR scheduling algorithm, no process is allocated the CPU for more than 1 time quantum in a row (unless it is the only process in ready queue). If a process's the next CPU burst exceeds 1 time quantum, that process is preempted and is put back in the ready queue.
- It can be implemented by a circular queue. New process or time expired process will be added to the tail of the ready queue.
- The average waiting time under the RR policy is often long.

Scheduling Algorithms
(CPU Scheduling Algorithm: Round-Robin Scheduling)

Process	Next CPU burst time
P_1	24
P_2	10
P_3	14

three processes are in the ready queue to run. With time quantum 4 unit

$P_1 > P_2 > P_3$

Average waiting time = (0+24+27)/3=17

Average waiting time = ((8+8+6+2)+(4+8+8)+(8+8+6+4))/3=23.3

Scheduling Algorithms
(CPU Scheduling Algorithm: Priority Scheduling)

- A priority is associated with each process, and the CPU is allocated to the process with the highest priority. Equal-priority processes are scheduled in FCFS order or use RR.
- Shortest Job First algorithm is a kind of Priority scheduling where shortest CPU burst time process is associated with highest priority.
- A priority scheduling algorithm can leave some low priority processes waiting indefinitely. Rumor has it that when they shut down the IBM 7094 at MIT in 1973, they found a low-priority process that had been submitted in 1967 and had not yet been run.
- Several variation of Priority Scheduling
  - Non-preemptive Priority Scheduling
  - Preemptive Priority Scheduling
  - Priority Scheduling with Round-Robin between processes with same priority

Scheduling Algorithms
(CPU Scheduling Algorithm: Priority Scheduling)

How to set a priority to each process?

- **Internal way** – based on the measurable quantity or quantities to compute the priority of a process.
  Ex)
  - CPU time limits, memory requirements, the number of open files, average I/O bound and CPU/bound ratio.
- **External way** – based on the importance of the process.
  Ex)
  - school: faculty have high priority than student.
  - company: A customer who pay more money can get high priority.

Scheduling Algorithms
(CPU Scheduling Algorithm: Priority Scheduling)

- A major problem with priority scheduling algorithm is the **starvation** of lower priority process – if higher priority process keep coming to ready queue, lower priority process never have a chance to be selected.
- A solution to the problem of starvation of low priority process is **aging**
- A **aging** is a technique of gradually increasing the priorities of processes that wait in the system for a long time.

Scheduling Algorithms
(CPU Scheduling Algorithm: Priority Scheduling)

Process	Arrival Time	priority	CPU time
P_1	0	1	8
P_2	0	3	4
P_3	0	4	9
P_4	0	2	5

High number has high priority

$P_3 > P_2 > P_4 > P_1$

Average waiting time = (18 + 9 + 0 + 13)/4 = 10

Average Turnaround time = (26 + 13 + 9 + 18)/4 = 16.5

- Each processes has different priorities
- All processes are arrived at time T

Scheduling Algorithms
(CPU Scheduling Algorithm: Priority Scheduling)

Process	Arrival Time	priority	CPU time
P_1	0	1	8
P_2	1	3	4
P_3	2	4	9
P_4	3	2	5

High number has high priority

$P_3 > P_2 > P_4 > P_1$

Average waiting time = ((19 – 1) + (11 – 2) + 0 + (14 – 3))/4 = 9.5

Average Turnaround time = ((26 – 0) + (14 – 1) + (11 – 2) + (19 – 3))/4 =16

- Each processes has different priorities
- Each process arrived at different time unit

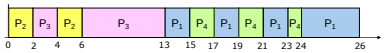
Scheduling Algorithms
(CPU Scheduling Algorithm: Priority Scheduling)

Process	Arrival Time	priority	CPU time
P_1	0	1	8
P_2	0	3	4
P_3	0	3	9
P_4	0	1	5

High number has high priority

$P_2 = P_3 > P_4 = P_1$

R-R time quantum = 2 unit



Average waiting time = ((13+2+2+2+1)+2+(2+2)+(15+2+2)+/4 =

Average Turnaround time = (26 + 6 + 13+ 24)/4 =

- Some processes might have same priorities
- Every processes are arrived at time T

Scheduling Algorithms
(CPU Scheduling Algorithm: Priority Scheduling)

Process	Arrival Time	priority	CPU time
P_1	0	1	8
P_2	1	3	4
P_3	2	3	9
P_4	3	1	5

High number has high priority

$P_2 = P_3 > P_4 = P_1$

R-R time quantum = 2 unit

Average waiting time = ((13+2+2+2+1)+2+(1+2)+(13+2+2)+/4 =

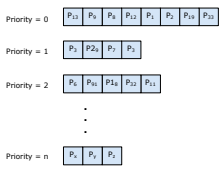
Average Turnaround time = (26 +(7 – 1) +(14 -2) + (25 -3))/4 =

- Some processes might have same priorities
- Each processes are arrived at different time

Scheduling Algorithms
(CPU Scheduling Algorithm: Priority Scheduling with Multilevel Queue)

- Priority scheduling, with a single queue - O(n) search may be necessary to determine the highest-priority process. Multilevel queue can be used to improve search time.
- A system keep multilevel queue each level maintain pointer to process tables with same priority.
- Multilevel queue works well when priority scheduling is combined with round-robin: if there are multiple processes in the highest-priority queue, they are executed in round-robin order

Scheduling Algorithms
(CPU Scheduling Algorithm: Priority Scheduling with Multilevel Queue)



Separate queues for each priority

Scheduling Algorithms
(CPU Scheduling Algorithm: Priority Scheduling with Multilevel Queue)

- A multilevel queue scheduling algorithm can also be used to partition processes into several separate queues based on the process type.
  Example)
  - Real-time processes
  - System processes
  - Interactive processes
  - Batch processes
- Different level queue might have its own scheduling algorithm.

Scheduling Algorithms
(CPU Scheduling Algorithm: Priority Scheduling with Multilevel Feedback Q)

- With a multilevel queue, processes are permanently assigned to a queue when they enter the system.
- With the multilevel feedback queue, allows a process to move between queues.
- Separate processes based on the CPU used. If a process uses too much CPU time, it will be moved to a lower priority queue.

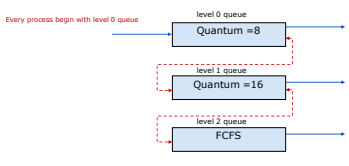
Scheduling Algorithms
(CPU Scheduling Algorithm: Priority Scheduling with Multilevel Feedback Q)

- Parameters for implementing a multilevel feedback queue
  - The number of queues
  - The scheduling algorithm for each queue
  - Determine when to upgrade a process to a higher priority queue
  - Determine when to demote a process to a lower priority queue
  - Determine which queue a process will enter when that process needs service

Scheduling Algorithms
(CPU Scheduling Algorithm: Priority Scheduling with Multilevel Feedback Q)

- Example with three level queue.
  - General rule:
    - The scheduler first executes all processes in queue 0. Only when queue 0 is empty, execute processes in queue 1. Similarly, processes in queue 2 will be executed only if queues 0 and 1 are empty.
    - An entering process is put in queue level 0.
    - A process in queue level 0 is given a time quantum of 8 milliseconds.
    - A process in queue level 1 is given a time quantum of 16 milliseconds.
    - If a process in queue level 0 or 1 cannot finish its job in given quantum, the process move down to the tail of lower level queue.
    - To prevent starvation, a process that waits too long in a lower level queue may gradually be moved to a higher level queue.

Scheduling Algorithms
(CPU Scheduling Algorithm: Priority Scheduling with Multilevel Feedback Q)



Multilevel Feedback Queue Example

Scheduling Algorithms
(CPU Scheduling Algorithm: Guaranteed Scheduling)

- Guaranteed scheduling algorithm guarantees fairness by monitoring the amount of CPU time spent by each user and allocating resources (CPU) accordingly.
- To make fairness, system must keep track of how much CPU each process has had since its creation.
- If there are n processes in the system, each process will receive 1/n of the CPU power.
- Since the amount of CPU time each process has actually used is known, it is fairly easy to compute the ratio of actual CPU time consumed.
- Based ratio, scheduler select a process for CPU.

Scheduling Algorithms

(CPU Scheduling Algorithm: Lottery Scheduling)

- When a program start to run, lottery ticket is given to each process for various system resources.
- Whenever a scheduling decision has to be made, a lottery ticket is selected at random and the process holding that ticket gets the resources.