ID	Name	Subject	Dep
201-2204008	Ahmad Seyam Nasser	Operating system	BSC

Operating system assingment

Scheduling Algorithms and its types

Priority Scheduling:

Priority Scheduling is a method of scheduling processes that is based on priority. In this algorithm, the scheduler selects the tasks to work as per the priority. The processes with higher priority should be carried out first, whereas jobs with equal priorities are carried out on a round-robin or FCFS basis. Priority depends upon memory requirements, time requirements, etc.

Types of Priority Scheduling

1. **Preemptive Scheduling**
   In Preemptive Scheduling, the tasks are mostly assigned with their priorities. Sometimes it is important to run a task with a higher priority before another lower priority task, even if the lower priority task is still running. The lower priority task holds for some time and resumes when the higher priority task finishes its execution.

2. **Non-Preemptive Scheduling**

In this type of scheduling method, the CPU has been allocated to a specific process. The process that keeps the CPU busy, will release the CPU either by switching context or terminating. It is the only method that can be used for various hardware platforms. That is because it does not need special hardware(for example, a timer) like preemptive scheduling.

Characteristics of Priority Scheduling

- A CPU algorithm that schedules processes based on priority ● It is used in Operating Systems for performing batch processes.
- If two jobs having the same priority are READY, it works on a FIRST COME, FIRST SERVE basis.
- In Priority Scheduling, a number is assigned to each process that indicates its priority level.
- The lower the number, the higher the priority is.
- In this type of scheduling algorithm, if a new process arrives that has higher priority than the currently-running process(es), then they are preempted.

Round-robin Scheduling and its characteristics

Round-robin is one of the algorithms employed by process and network schedulers in computing. As the term is generally used, time slices (also known as time quanta) are assigned to each process in equal portions and in circular order, handling all processes without priority (also known as cyclic execution). Round-robin scheduling is simple, easy to implement, and starvation-free. Round-robin scheduling can be applied to other scheduling problems, such as data packet scheduling in computer networks. It is an Operating system concept.

Characteristics of Round-robin Scheduling

- **Time slicing:** It uses a fixed time quantum or time slice which is a predetermined amount of time that each process is allowed to execute.
- **Preemptive:** Which means that the operating system can interrupt a running process and switch to another process when the time quantum expires. This ensures that no single process can monopolize the CPU for an extended period.
- **First-Come-First-Serve ready queue:** Processes are placed in a queue, and they are executed in the order they arrive in the queue. The process at the front of the queue is given CPU time until its quantum is used up, or it voluntarily yields the CPU.
- **Overhead:** The Round-robin algorithm has a low scheduling overhead, as it only requires maintaining a simple queue and a time to switch processes when the time quantum expires. This simplicity makes it efficient and suitable for real-time systems.

Task 1: Scheduling Algorithm and its types:

1. there are serval process scheduling algorithms commonly used in operating system:

a. [first-come,first-served]:in FCFS the process that arrives first gets executed first.it simple but not suitable for time-sensitive task as it may lead too long waiting times.

Characteristics of FCFS

- **Non-Preemptive:** FCFS is a non-preemptive scheduling algorithm, which means that once a process starts executing, it continues until it completes its execution or voluntarily gives up the CPU. Other processes have to wait their turn.
- **Queue:** Processes are placed in a queue in the order in which they arrive. The first process to arrive is the first to be executed, hence the name "First-Come-First-Serve."
- **Fairness:** FCFS provides fairness in the sense that processes are executed in the order they arrive. However, this fairness may not always lead to optimal performance, especially for long-running processes.
- **Simplicity:** FCFS is easy to understand and implement. It doesn't require complex data structures or extensive bookkeeping.
- **Low scheduling overhead:** FCFS has low scheduling overhead because it doesn't require frequent context switches. It simply selects the next process in the queue when the currently running process finishes.

- **Inefficiency for short processes:** While FCFS is straightforward, it may not be efficient in terms of waiting time for short processes if long processes are ahead in the queue. Short processes can be held up by long processes, leading to a phenomenon known as "convoy effect."
- **No consideration for process priorities**: FCFS does not take into account the priority or importance of processes. All processes are treated equally, which may not be suitable for systems with varying task priorities.
- **High average waiting time**: FCFS scheduling can lead to a high average waiting time, especially if processes with long execution times arrive before shorter ones. This can result in poor performance for some workloads.
- **Deterministic behavior:** FCFS provides deterministic behavior, meaning that the order of execution is entirely predictable and follows a strict first-come-first-serve order.

c. [shortest job first]: SJF schedules the process with the shortest execution time first.it minimizes the waiting time but can suffer from the "starvation" problem if short process keep arriving.

2. Comparison:

-FCFS: simple to implement, but not suitable for time-sensitive tasks.

-SJF: minimize waiting time but can lead to starvation.

-RR: provides fairness but may not be efficient for long task.

3.Real-word scenarios:

-FCFS: A print queue, where the first document submitted is the first to be printed, as all print jobs are equally important.

-SJF: CPU scheduling in a supercomputer center, where tasks are submitted, and the goal is to minimize execution time for all user

-RR: A shared computer lab where multiple user run tasks. RR ensure everyone gets equal access to the CPU.

What is Multithreading

Multithreading is the ability of a program or an operating system to enable more than one user at a time without requiring multiple copies of the programming running on the computer. Multithreading can also handle the multiple request from same user.

Task 2: Practical application

1. Here is a simple python codes that demonstrates multithreading by calculating the squares of number concurrently: PYTHON

```
Import threading

Def calculate_square(number):

    Result = number * number

    Print(“the square of {number} is {result}”)   If

__name__ == “__main__”:

    Numbers = [1, 2, 3, 4, 5]

    Threads = []


   For num in numbers:

       Threads = threading.thread(target=calculate_square,args=(num,))

       Threads.append(thread)

       Thread.start()
```