OPERATING SYSTEMS: CPU SCHEDULING PROGRAMMING ACTIVITY

A Documentary Research Paper Presented to
Computer Studies Department of College of Science
TECHNOLOGICAL UNIVERSITY OF THE PHILIPPINES

In Partial Fulfillment of the
Requirements for
OPERATING SYSTEMS

BSCS 2C-M
Manansala, Rhea Miguela M.
Morales, Carlos Miguel T.
Torres, Paul Adrian O.

PROF. PEGARINO AMADOR, Jr.

JULY 2023

I. RATIONALE

CPU Scheduling is an essential component of an operating system that manages the execution of processes on a computer's central processing unit (CPU). It determines the order in which processes are allocated CPU time and provides a mechanism for efficient utilization of the CPU's resources.

1. **SJF Scheduling (Shortest Job First)** – Aims to minimize the average waiting time of processes by selecting the shortest job to run next. This algorithm is that shorter jobs tend to be completed faster, leading to reduced waiting times for all processes. It can be either non-preemptive or preemptive, meaning that once a process starts executing, it may or may not be interrupted by another process.
2. **Pre-Emptive Scheduling** – Allows a running process to be interrupted and temporarily halted by another process with higher priority. This is typically used in real-time systems or when time-sharing is required.
3. **Round Robin Scheduling** - Allocates CPU time in fixed time slices called time quantum or time slice to each process in a cyclic manner. If a process does not complete within its time slice, it is preempted and moved to the end of the queue to wait for its turn again.

II. ALGORITHM, CODE AND RUN PROGRAM

CODE FOR SHORTEST JOB FIRST PRIORITY

```
def CPU_SJF(pr):
    num_of_processes = len(pr)
    total_cpu_burst_time = 0
    for i in range(num_of_processes):
        pr[i].process_id = i + 1
        print(f"\nPROCESS #{i+1}")
        print("-------------------------------------------------")
        pr[i].arrival_time = int(input("\tArrival Time: "))
        while True:
            burst_time = input("\tBurst Time: ")
            if burst_time.isdigit() and int(burst_time) > 0:
                pr[i].burst_time = int(burst_time)
                break
            else:
                print("\tInvalid burst time. Please enter a positive integer.")
        pr[i].rem_burst_time = pr[i].burst_time
        total_cpu_burst_time += pr[i].burst_time
        print("-------------------------------------------------")

    # Sort the processes based on arrival time and burst time (shortest job first)
    pr.sort(key=lambda x: (x.arrival_time, x.burst_time))

    pr[0].waiting_time = 0
    pr[0].turnaround_time = pr[0].burst_time
    pr[0].completion_time = pr[0].arrival_time + pr[0].burst_time

    for i in range(1, num_of_processes):
        current_process = pr[i]
        current_process.waiting_time = pr[i - 1].completion_time - current_process.arrival_time
        current_process.turnaround_time = current_process.waiting_time + current_process.burst_time
        current_process.completion_time = current_process.arrival_time + current_process.turnaround_time

    # Display the result
    print("\n-------------------------------------------------------------------------")
    print("| Process | Arrival Time | Burst Time | Waiting Time | Turnaround Time |")
    print("-------------------------------------------------------------------------")

    aveWT = 0
    aveTA = 0
    total_execution_time = pr[-1].completion_time - pr[0].arrival_time
```

```
    for process in pr:
        print(f"|   P{process.process_id:<3d}  |      {process.arrival_time:<6d}  |     {process.burst_time:<6d}
        aveWT += process.waiting_time
        aveTA += process.turnaround_time

    cpu_utilization = total_cpu_burst_time / total_execution_time

    print("-------------------------------------------------------------------------")
    print(f"\tAverage Waiting Time    : {aveWT / num_of_processes:.2f}")
    print(f"\tAverage Turnaround Time : {aveTA / num_of_processes:.2f}")
    print(f"\tSystem Throughput       : {num_of_processes / total_execution_time:.2f}")
    print(f"\tCPU Utilization         : {cpu_utilization:.2f}%")
```

CODE FOR PRE-EMPTIVE PRIORITY

```
def CPU_PP():
    num_of_processes = int(input("Enter the number of processes: "))
    processes = []

    for i in range(num_of_processes):
        process_id = i + 1
        print(f"\nProcess #{process_id}: ")
        print("-----------------------------------------------")
        arrival_time = int(input(f"\tArrival time: "))
        burst_time = int(input(f"\tBurst time: "))
        priority = int(input(f"\tPriority: "))
        print("-----------------------------------------------")
        process = Process(process_id, burst_time, priority)
        process.arrival_time = arrival_time
        processes.append(process)

    current_time = 0
    completed_processes = 0
    pr = []
    total_cpu_burst_time = sum(process.burst_time for process in processes)

    while completed_processes < num_of_processes:
        for process in processes:
            if process.arrival_time <= current_time and process.remaining_time > 0 and process not in pr:
                pr.append(process)

        if pr:
            pr.sort(key=lambda x: (x.priority, x.process_id))

            executing_process = pr[0]
            pr.remove(executing_process)

            executing_process.remaining_time -= 1
            current_time += 1

            if executing_process.remaining_time == 0:
                completed_processes += 1
                executing_process.completion_time = current_time
                executing_process.turnaround_time = executing_process.completion_time - executing_process.arr
                executing_process.waiting_time = executing_process.turnaround_time - executing_process.burst_
        else:
            current_time += 1
    # Display the result
    print("-------------------------------------------------------------------------------------------------")
    print("Process | Arrival Time | Burst Time | Priority | Completion Time | Waiting Time | Turnaround Time")
    print("-------------------------------------------------------------------------------------------------")
    total_waiting_time = 0
    total_turnaround_time = 0

    for process in processes:
        print(
            f"P{process.process_id:<3d}     |   {process.arrival_time:<6d}   |     {process.burst_time:<6d}
        total_waiting_time += process.waiting_time
        total_turnaround_time += process.turnaround_time

    print("-------------------------------------------------------------------------------------------------")

    average_waiting_time = total_waiting_time / num_of_processes
    average_turnaround_time = total_turnaround_time / num_of_processes
    total_execution_time = processes[-1].completion_time - processes[0].arrival_time

    cpu_utilization = total_cpu_burst_time / total_execution_time

    print(f"\nAverage Waiting Time: {average_waiting_time:.2f}")
    print(f"Average Turnaround Time: {average_turnaround_time:.2f}")
    print(f"System Throughput: {num_of_processes / total_execution_time:.2f}")
    print(f"CPU Utilization: {cpu_utilization:.2f}%")
```

CODE FOR ROUND ROBIN PRIORITY

```
def CPU_RR():
    n = int(input("Enter number of processes: "))
    arrival_time = []
    burst_time = []
    remaining_time = []
    waiting_time = [0] * n
    turnaround_time = [0] * n
    quantum = int(input("Enter time quantum: "))

    for i in range(n):
        process_id = i + 1
        print(f"\nProcess #{process_id}: ")
        print("-----------------------------------------------")
        arrival_time.append(int(input("Arrival time: ")))
        burst_time.append(int(input("Burst time: ")))
        remaining_time.append(burst_time[i])
        print("-----------------------------------------------")

    current_time = 0
    completed = 0
    queue = []
    total_cpu_burst_time = sum(burst_time)

    while completed != n:
        for i in range(n):
            if arrival_time[i] <= current_time and remaining_time[i] > 0 and i not in queue:
                queue.append(i)

        if not queue:
            current_time += 1
            continue
```

```
        if not queue:
            current_time += 1
            continue

        i = queue[0]

        if remaining_time[i] <= quantum:
            # Process execution is complete
            current_time += remaining_time[i]
            turnaround_time[i] = current_time - arrival_time[i]
            waiting_time[i] = turnaround_time[i] - burst_time[i]
            remaining_time[i] = 0
            completed += 1
            queue = queue[1:]
        else:
            current_time += quantum
            remaining_time[i] -= quantum

            for j in range(n):
                if arrival_time[j] <= current_time and remaining_time[j] > 0 and j != i and j not in queue:
                    queue.append(j)

            queue = queue[1:] + [i]
    # Display the result
    print("-------------------------------------------------------------------")
    print("Process | Arrival Time | Burst Time | Waiting Time | Turnaround Time")
    print("-------------------------------------------------------------------")
    for i in range(n):
        print(f"P{i+1}      |     {arrival_time[i]:<3d}      |     {burst_time[i]:<6d} |      {waiting_time[i]:<8d}|   { turnaround_time[i]:<
    print("-------------------------------------------------------------------")

    avg_waiting_time = sum(waiting_time) / n
    avg_turnaround_time = sum(turnaround_time) / n
    total_execution_time = max(turnaround_time) - min(arrival_time)
    cpu_utilization = total_cpu_burst_time / total_execution_time

    print("\nAverage Waiting Time: {:.2f}".format(avg_waiting_time))
    print("Average Turnaround Time: {:.2f}".format(avg_turnaround_time))
    print("System Throughput: {:.2f}".format(n / total_execution_time))
    print("CPU Utilization: {:.2f}%".format(cpu_utilization))
```

RUN PROGRAM

FOR SHORTEST JOB FIRST

```
-------------------------------------------------------------------------
| Process | Arrival Time | Burst Time | Waiting Time | Turnaround Time |
-------------------------------------------------------------------------
|   P1    |      0       |     4      |      0       |      4          |
|   P3    |      1       |     9      |      3       |      12         |
|   P2    |      6       |     2      |      7       |      9          |
-------------------------------------------------------------------------
        Average Waiting Time    : 3.33
        Average Turnaround Time : 8.33
        System Throughput       : 0.20
        CPU Utilization         : 1.00%
```

FOR PRE-EMPTIVE PRIORTY

```
-----------------------------------------------------------------------------------------------
Process | Arrival Time | Burst Time | Priority | Completion Time | Waiting Time | Turnaround Time
-----------------------------------------------------------------------------------------------
P1      |  0           |    4       |  5       |  13             |   9          |  13
P2      |  1           |    9       |  1       |  10             |   0          |  9
P3      |  6           |    2       |  6       |  15             |   7          |  9
-----------------------------------------------------------------------------------------------

Average Waiting Time: 5.33
Average Turnaround Time: 10.33
System Throughput: 0.20
CPU Utilization: 1.00%
```

FOR ROUND ROBIN PRIORITY

```
-------------------------------------------------------------------------
Process | Arrival Time | Burst Time | Waiting Time | Turnaround Time
-------------------------------------------------------------------------
P1      |     0        |     1      |      0       |  1
P2      |     5        |     3      |      0       |  3
P3      |     3        |     1      |      0       |  1
-------------------------------------------------------------------------

Average Waiting Time: 0.00
Average Turnaround Time: 1.67
System Throughput: 1.00
CPU Utilization: 1.67%
```