Operating Systems Project Report

Mini Project

Title: To-Do List Priority Scheduler

Team Members:

1. Parva Mehrotra - 200905116 , Batch A1
2. Naman Kumar - 200905176 , Batch A1
3. Harshika Sofat - 200905117 , Batch A1
4. Mehul Chauhan - 200905174 , Batch A1
5. Bipin Livingston - 200905036 , Batch A1

➔ Abstract

In this mini-project, we will be addressing and making a to-do list priority scheduler that will prioritize the everyday tasks of a person based on priority. Time management is something that a lot of people struggle with, especially students. They are involved in numerous activities that affect their holistic development overall. Thus, keeping track of their daily tasks and completing it on time becomes a struggle. Therefore, to solve this problem, we develop a to-do list scheduler that prioritizes the activities throughout the day, based on their priority.

To do so, we use operating systems concepts to implement a priority scheduler that firstly inputs their name, their sleep time, duration of sleep, and their meal timings. Next, we ask the user to input the number of activities they need to do with the priority and the time it will take to complete each task. Additionally, we even take the arrival time of the activity into consideration.

The to-do list is implementable as a daily activity scheduler which has a lot of practical applications.

➔ Aim

To demonstrate tasks being scheduled based on the priority of activities using preemptive scheduling.

➔Hardware and Programming Languages

Programming Language: C

System: Linux- ubuntu 20.04LTS, Windows Subsystem for Linux (WSL)

Hardware:

Processor: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 1.80 GHz

GPU: Nvidia GeForce 1660 Ti GDDR6 @ 6GB(192 bits)

Libraries: string.h, stdio.h, stdlib.h

➔ Pre-Requisites

CPU Scheduling is a process of determining which process will own the CPU for execution while another process is on hold. The main task of CPU scheduling is to make sure that whenever the CPU remains idle, the OS at least selects one of the processes available in the ready queue for execution. The selection process

will be carried out by the CPU scheduler. It selects one of the processes in memory that are ready for execution.
Two major types of CPU scheduling algorithms are examined for his analysis.

1. Non-Preemptive Scheduling
Non-preemptive Scheduling is used when a process terminates, or a process switches from running to the waiting state. In this scheduling, once the resources (CPU cycles) are allocated to a process, the process holds the CPU till it gets terminated or reaches a waiting state. In the case of non-preemptive scheduling, the CPU does not interrupt a process running CPU in the middle of the execution. Instead, it waits till the process completes its CPU burst time, and then it can allocate the CPU to another process.

2. Preemptive Scheduling
Preemptive scheduling is used when a process switches from running state to ready state or from the waiting state to ready state. The resources (mainly CPU cycles) are allocated to the process for a limited amount of time and then taken away, and the process is again placed back in the ready queue if that process still has CPU burst time remaining. That process stays in the ready queue till it gets its next chance to execute.
For our task implementation, we employ preemptive schedulers due to the following reasons:

1. A processor can be preempted to execute the different processes in the middle of any current process execution. Allows better computation utility.
2. CPU utilization is more efficient compared to Non-Preemptive Scheduling.
3. Waiting and response time of Preemptive Scheduling is less.
4. Preemptive Scheduling is prioritized. The highest priority process is a process that is currently utilized.
5. Preemptive Scheduling is flexible.

➔Implementation Methodology

A brief overview of keywords specific to preemptive scheduling:

1. **Arrival Time:** Time at which the process arrives in the ready queue.
2. **Completion Time:** Time at which process completes its execution.
3. **Burst Time:** Time required by a process for CPU execution.
4. **Turn Around Time:** Time Difference between completion time and arrival time.
5. **Waiting Time:** Time Difference between turn around time and burst time.
6. **Priority:** The priority given to a particular task at hand.

For our study of scheduling daily activities using preemptive scheduling, we discuss key variables. We then make a comparison and analyse them with respect to keywords discussed under preemptive schedulers.

1. **arrivalTime:** The time at which the activity for the day becomes available to the user.
2. **burstTime:** The duration required to complete an activity.
3. **waitingTime:** Then an activity has to wait after its availability before being executed.
4. **turnaroundTime:** The time after which an activity becomes available and before it is finished.
5. **priority:** The set priority for an activity (higher will be executed first).

Algorithm:

1. The user is asked for sleeping time, breakfast, lunch, and dinner time (to ensure that they are not active during that time).

2. Then, they are asked to enter the activity details. For each activity, they will be the activity name, arrival time, burst time, and priority.

3. A for loop is run for 24 iterations. In each iteration, the activity with the highest priority is extracted, excluding those with burst time equal to zero.

4. As the activity name is known, a corresponding activity ID is assigned and stored in a character hours[] array. This array is responsible for

keeping track of each process' execution with respect to the time lapsed in the day.

5. The next day begins and activities are executed in a preemptive fashion on the basis of their priority. Here, the burst time for each executed activity is decremented as we proceed and the loop iterates over all activities until completion.

6. Simultaneously, we keep storing the Activity ID according to each hour of the day in the act_name array. This represents activities for each hour of the day and if hours are free or not.

7. The logic for breakfast, lunch, dinner hours is also put in the above step.

8. Finally, when all activities for the day/the hours in the day are over, we print them according to activities executed in certain hours.

Formulae Used:
Waiting Time= Turnaround Time — Burst Time
Turnaround Time= Completion Time — Arrival Time
We now develop the code for the aforementioned algorithm.

➔ Code & Result

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct process
{
  char activ[50];
  int waitingTime, arrivalTime, burstTime, turnaroundTime, priority, id;
};
char hours[24];
char act_name[24][50];


struct process a[24], b;
int n;
int extractHour(char *string)
{
  int i, time_int;
  char time_temp[3];
  for (i = 0; i < 3; i++)
  {
    if (string[i] == ':')
    {
      break;
    }
    time_temp[i] = string[i];
  }
  time_int = atoi(time_temp);
  return time_int;
}

int main()
{
  int temp[10], t, count = 0, highest_p, wake_time;
  char break_time[10], lun_time[10], din_time[10];
```

```
  int sleep_hours = 0;

  char string[256], name[25], sleep[10];
  float total_waitingTime = 0, total_turnaroundTime = 0, Avg_waitingTime,
Avg_turnaroundTime;
  printf("What is your name?\n");
  scanf("%s", name);
  printf("Greetings, %s! :)\n", name);
  printf("When do you plan to sleep tonight?\n");
  scanf("%s", sleep);
  printf("How many hours, %s?\n", name);
  scanf("%d", &sleep_hours);
  int sleep_time = extractHour(sleep);
  if (sleep_time + sleep_hours > 24)
  {
    wake_time = sleep_time + sleep_hours - 24;
    printf("So you'll wake up at %d:00, great!\n", wake_time);
  }
  else
  {
    wake_time = sleep_time + sleep_hours;
    printf("So you'll wake up at %d:00 tomorrow, great!\n", wake_time);
  }
  for (int i = 0; i < 24; ++i)
  {
      strcpy(act_name[i],"Free");
  }

  char c;
  printf("Do you have breakfast? y-yes n-no\n");
  scanf(" %c", &c);
  if (c == 'y')
  {
    printf("Enter breakfast and grooming time\n");
    scanf("%s", break_time);
  }
  int b_time= extractHour(break_time);
```

```
printf("Enter lunch time\n");
scanf("%s", lun_time);
int l_time= extractHour(lun_time);

printf("Enter dinner time\n");
scanf("%s", din_time);
int d_time= extractHour(din_time);

printf("Enter the number of the activities:\n");
scanf("%d", &n);
printf("\nEnter the arrival time, duration and importance of the activities:\n");
for (int i = 0; i < n; i++)
{
  printf("\nInput details of activity %d:\n", i + 1);
  printf("Enter activity name:\n");
  scanf("%s", a[i].activ);
  printf("Enter Arrival-Time:- ");
  scanf("%s", string);
  a[i].arrivalTime = extractHour(string);
  printf("Number of hours needed for the activity:- ");
  scanf("%d", &a[i].burstTime);
  printf("Number of priority for the activity:- ");
  scanf("%d", &a[i].priority);
  temp[i] = a[i].burstTime;
  a[i].id = i;
}

a[9].priority = -1;
for (t = wake_time; t < 24; t++)
{
  if (t == b_time)
  {
    hours[t] = 'B';
    strcpy(act_name[t],"Breakfast");
    continue;
  }
```

```
    if (t == l_time)
    {
      hours[t] = 'L';
      strcpy(act_name[t],"Lunch");
      continue;
    }
    if (t == d_time)
    {
      hours[t] = 'D';
      strcpy(act_name[t],"Dinner");
      continue;
    }
    highest_p = 9;
    for (int i = 0; i < n; i++)
    {
      if (a[highest_p].priority < a[i].priority && a[i].arrivalTime <= t && a[i].burstTime
> 0)
      {
        highest_p = i;
        strcpy(act_name[t],a[i].activ);
        hours[t] = a[highest_p].id + '0';

      }
    }

    a[highest_p].burstTime = a[highest_p].burstTime - 1;
    if (a[highest_p].burstTime == 0)
    {
      count++;
      a[highest_p].waitingTime = t + 1 - a[highest_p].arrivalTime - temp[highest_p];
      a[highest_p].turnaroundTime = t + 1 - a[highest_p].arrivalTime;
      total_waitingTime = total_waitingTime + a[highest_p].waitingTime;
      total_turnaroundTime = total_turnaroundTime + a[highest_p].turnaroundTime;
    }
  }
  for (int i = 0; i < sleep_hours; i++)
  {
```

```
    hours[sleep_time] = 'S';
    strcpy(act_name[sleep_time],"Sleep");
    sleep_time++;
    if (sleep_time > 23)
      sleep_time = 0;
  }
  Avg_waitingTime = total_waitingTime / n;
  Avg_turnaroundTime = total_turnaroundTime / n;
  printf("\n\nActivity-ID\tBreak-Time\tTotal-Submission-Time\n");
  for (int i = 0; i < n; i++)
  {
    printf("%d\t\t%d\t\t%d\n", i , a[i].waitingTime,
          a[i].turnaroundTime);
  }
  printf("\n");
  printf("\nAverage break time of the activities is %f\n", Avg_waitingTime);
  printf("Average turnaround time of the activities is %f\n\n",Avg_turnaroundTime);

  printf("Your schedule for the day is as follows:\n\n");

  for (int i = 0; i < 24; i++)
  {
    if(strcmp(act_name[i],act_name[i+1])==0)
    {
        int j=i;
        while(strcmp(act_name[i],act_name[i+1])==0)
        {
            i++;
        }
        printf("Hours: %d to %d\t\tActivity ID: %c\tActivity: %s\n", j, i+1, hours[i],
act_name[i] );
    }
    else
    {
        if(i<10)
        {
```

```
            printf("Hour: %d to %d\t\tActivity ID: %c\tActivity: %s\n", i,i+1, hours[i],
act_name[i]);
        }
        else
        printf("Hour: %d to %d\t\tActivity ID: %c\tActivity: %s\n", i,i+1, hours[i],
act_name[i]);
    }
  }
  if(count!=n)
  {
    printf("\nDon't worry as you didn't finish in time, you can do the remaining
activities tomorrow, %s! :)\n",name);
  }
  return 0;
  return 0;
}
```

```
PS C:\Users\naman\Downloads> gcc OS_final.c
PS C:\Users\naman\Downloads> ./a.exe
What is your name?
Rahul
Greetings, Rahul! :)
When do you plan to sleep tonight?
23:00
How many hours, Rahul?
8
So you'll wake up at 7:00, great!
Do you have breakfast? y-yes n-no
y
Enter breakfast and grooming time
7:00
Enter lunch time
12:00
Enter dinner time
18:00
Enter the number of the activities:
5

Enter the arrival time, duration and importance of the activities:

Input details of activity 1:
Enter activity name:
Gym
Enter Arrival-Time:- 7:00
Number of hours needed for the activity:- 3
Number of priority for the activity:- 2

Input details of activity 2:
Enter activity name:
Study
Enter Arrival-Time:- 8:00
Number of hours needed for the activity:- 2
```

```
Study
Enter Arrival-Time:- 8:00
Number of hours needed for the activity:- 2
Number of priority for the activity:- 4

Input details of activity 3:
Enter activity name:
Cricket
Enter Arrival-Time:- 13:00
Number of hours needed for the activity:- 3
Number of priority for the activity:- 6

Input details of activity 4:
Enter activity name:
Class
Enter Arrival-Time:- 14:00
Number of hours needed for the activity:- 2
Number of priority for the activity:- 9

Input details of activity 5:
Enter activity name:
Meditation

Activity-ID     Break-Time      Total-Submission-Time
0               10              13
1               0               2
2               2               5
3               0               2
4               0               1


Average break time of the activities is 2.400000
Average turnaround time of the activities is 4.600000

Your schedule for the day is as follows:
```

```
Input details of activity 5:
Enter activity name:
Meditation

Activity-ID     Break-Time      Total-Submission-Time
0               10              13
1               0               2
2               2               5
3               0               2
4               0               1


Average break time of the activities is 2.400000
Average turnaround time of the activities is 4.600000

Your schedule for the day is as follows:

Hours: 0 to 7           Activity ID: S  Activity: Sleep
Hour: 7 to 8            Activity ID: B  Activity: Breakfast
Hours: 8 to 10          Activity ID: 1  Activity: Study
Hours: 10 to 12         Activity ID: 0  Activity: Gym
Hour: 12 to 13          Activity ID: L  Activity: Lunch
Hour: 13 to 14          Activity ID: 2  Activity: Cricket
Hours: 14 to 16         Activity ID: 3  Activity: Class
Hours: 16 to 18         Activity ID: 2  Activity: Cricket
Hour: 18 to 19          Activity ID: D  Activity: Dinner
Hour: 19 to 20          Activity ID: 0  Activity: Gym
Hours: 20 to 22         Activity ID:    Activity: Free
Hour: 22 to 23          Activity ID: 4  Activity: Meditation
Hour: 23 to 24          Activity ID: S  Activity: Sleep
PS C:\Users\naman\Downloads>
```

➔Learning Outcomes

By the end of this mini-project, we should have a basic understanding and knowledge of preemptive scheduling, non-preemptive scheduling and their different types. We also learn the characteristics, downfalls of priority scheduling. The program's well-organized structure also aided us in improving our coding etiquette by requiring us to maintain clean code with indentation. It's also helped us improve our understanding of the above-mentioned areas, which will come in handy for our OS lab examinations. Overall, working on this project has been an educational and fruitful experience.

➔Contribution from each student

Parva Mehrotra: Responsible for proper input/output format of the to-do list (interaction with the User)
Naman Kumar : Implementing preemption on the basis of priority so that the activities are executed efficiently
Harshika Sofat: Responsible for the deep domain research of the topic and adding operation (method)
Mehul Chauhan: Looking into minute aspects of the code such that the different possible errors are eliminated
Bipin Livingston: Brainstorming and ideation a practical idea that is relevant to the real-world scenario (especially students)

➔Conclusion & Future Scope

This to-do list priority scheduler can come in handy, especially while organizing daily activities. This will help the user keep track of all activities that they have to perform throughout the day according to the time they arrive and the duration in which they'll be completed. Pre-emption allows an efficient way of execution so that activities execute such that the system always has a process to execute.
In all, whenever a user is confused about the activities they have to execute in a day, they can refer to a to-do list such as this and rely on it for efficient execution, so that their work gets done in time.

➔ References

[1] https://www.sciencedirect.com/topics/engineering/preemptive-scheduling
[2] https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/
[3] https://www.tutorialspoint.com/preemptive-and-non-preemptive-scheduling
[4] https://stackoverflow.com/
[5] OS Lab manual
[6] Operating System Concepts (9e)