Operating Systems Assignment 2

Ritikesh Vali
2019UEE5095

Q1. Implement Round Robin Scheduling algorithm taking time slice as 2 ms. Implement it using queue data structure: read the process number, its entering time and its CPU burst time and store them in a queue data structure called ready queue. Take at least 4 processes and find the average waiting time and average turnaround time for each process

```
Processes  Burst time  Waiting time  Turn around time
 1       10    37       47
 2       5     22       27
 3       8     31       39
 4       12    43       55
5        15    49       64
6        32    50       82
Average waiting time is 38.6667 and average turn around time is 52.3333
```

.

round-robin.cpp - OS - Visual Studio Code

round-robin.cpp > averagetime(int [], int, int [], int)

```cpp
        if(done==true)
            break;
    }
}

void turnaround(int processes[],int n,int bursttime[],int wt[],int tat[])
{
    for(int i=0;i<n;i++)
        tat[i]=bursttime[i]+wt[i];
}

void averagetime(int processes[],int n,int bursttime[],int quantum)
{
    int total_wt=0, total_tat=0;
    int *wt=new int[n];
    int *tat=new int[n];

    waitingtime(processes,n, bursttime, wt, quantum);
    turnaround(processes, n, bursttime, wt, tat);
    cout<<"Processes "<<" Burst time "<<" Waiting time "<<" Turn around time\n";

    for(int i=0;i<n;i++)
    {
        total_wt += wt[i];
        total_tat +=tat[i];
        cout<<" "<<i+1<<"\t\t"<<bursttime[i]<<"\t"<<wt[i]<<"\t\t "<<tat[i]<<'\n';
    }
    cout<<"Average waiting time is "<<(float)total_wt/(float)n;
    cout<<" and average turn around time is "<<(float)total_tat/(float)n;
}
```

Ln 57, Col 64 Spaces: 4 UTF-8 CRLF C++ Go Live Win32

round-robin.cpp - OS - Visual Studio Code

round-robin.cpp > averagetime(int [], int, int [], int)

```cpp
#include<iostream>
using namespace std;

void waitingtime(int processes[],int n,int bursttime[],int wt[],int quantum)
{
    int *remburstime = new int[n];
    for(int i=0;i<n;i++)
        remburstime[i]=bursttime[i];

    int t=0;
    while (1)
    {
        bool done=true;
        for(int i=0;i<n;i++)
        {
            if(remburstime[i]>0)
            {
                done=false;
                if(remburstime[i]>quantum)
                {
                    t+=quantum;
                    remburstime[i]-=quantum;
                }
                else
                t+=remburstime[i];
                wt[i]=t-bursttime[i];
                remburstime[i]=0;
            }
        }
        if(done==true)
```

Ln 57, Col 64 Spaces: 4 UTF-8 CRLF C++ Go Live Win32

round-robin.cpp - OS - Visual Studio Code

round-robin.cpp > averagetime(int [], int, int [], int)

```cpp
{
    int total_wt=0, total_tat=0;
    int *wt=new int[n];
    int *tat=new int[n];

    waitingtime(processes,n, bursttime, wt, quantum);
    turnaround(processes, n, bursttime, wt, tat);
    cout<<"Processes "<<" Burst time "<<" Waiting time "<<" Turn around time\n";

    for(int i=0;i<n;i++)
    {
        total_wt += wt[i];
        total_tat +=tat[i];
        cout<<" "<<i+1<<"\t\t"<<bursttime[i]<<"\t"<<wt[i]<<"\t\t "<<tat[i]<<'\n';
    }
    cout<<"Average waiting time is "<<(float)total_wt/(float)n;
    cout<<" and average turn around time is "<<(float)total_tat/(float)n;
}

int main()
{
    int processes[] = {1,2,3,4,5,6};
    int n=6;
    int bursttime[] = {10,5,8,12,15,32};
    int quantum=2;
    averagetime(processes, n, bursttime, quantum);

    return 0;
}
```

Ln 57, Col 64 Spaces: 4 UTF-8 CRLF C++ Go Live Win32

Q2. Implement Priority Scheduling algorithm with pre-emption. Implement it using priority queue data structure: read the process number, its entering time, its priority and its CPU burst time. Take at least 6 processes and find the average waiting time and average turnaround time for each process.

```
Process_no  Start_time  Complete_time   Turn_Around_Time
    Waiting_Time
1       1       7       6           0
2       7       12      10          5
3       12      22      19          9
4       22      24      20          18
5       24      31      26          19
6       31      32755       32724           0
Average waiting time is : 8.5and average turnaround time is : 5467.5
```

priority-scheduling.cpp - OS - Visual Studio Code

priority-scheduling.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;

struct process{
    int arrivaltime;
    int bursttime;
    int pr;
    int pno;
};

process proc[10];

bool cmp(process a,process b)
{
    if(a.arrivaltime==b.arrivaltime)
        return a.pr<b.pr;
    else
        return a.arrivaltime<b.arrivaltime;
}

void waitingtime(int waitingtime[],int totalprocess)
{
    int service[10];
    service[0]=proc[0].arrivaltime;
    waitingtime[0]=0;

    for(int i=1;i<totalprocess;i++)
    {
        service[i]=proc[i-1].bursttime+service[i-1];
        waitingtime[i]=service[i]-proc[i].arrivaltime;
```

priority-scheduling.cpp - OS - Visual Studio Code

priority-scheduling.cpp

```cpp
void waitingtime(int waitingtime[],int totalprocess)
{
    int service[10];
    service[0]=proc[0].arrivaltime;
    waitingtime[0]=0;

    for(int i=1;i<totalprocess;i++)
    {
        service[i]=proc[i-1].bursttime+service[i-1];
        waitingtime[i]=service[i]-proc[i].arrivaltime;
        if(waitingtime[i]<0)
            waitingtime[i]=0;
    }
}

void turnaround(int tarrivaltime[],int waitingtime[],int totalprocess)
{
    for(int i=0;i<totalprocess;i++)
        tarrivaltime[i]=proc[i].bursttime+waitingtime[i];
}

void ganttchart(int totalprocess)
{
    int waitingTime[10], tarrivaltime[10];
    double wavg=0, tavg=0;
    waitingtime(waitingTime,totalprocess);
    turnaround(tarrivaltime,waitingTime,totalprocess);

    int stime[10], ctime[10];
    stime[0]=proc[0].arrivaltime;
```

priority-scheduling.cpp - OS - Visual Studio Code

priority-scheduling.cpp

```cpp
void ganttchart(int totalprocess)
{
    int waitingTime[10], tarrivaltime[10];
    double wavg=0, tavg=0;
    waitingtime(waitingTime,totalprocess);
    turnaround(tarrivaltime,waitingTime,totalprocess);

    int stime[10], ctime[10];
    stime[0]=proc[0].arrivaltime;
    ctime[0]=stime[0]+tarrivaltime[0];

    for(int i=1;i<totalprocess;i++)
    {
        stime[i]=ctime[i-1];
        ctime[i]=stime[i]+tarrivaltime[i]-waitingTime[i];
    }
    cout<<"Process_no\tStart_time\tComplete_time\tTurn+Around_Time\tWaiting_Time"<<'\n';

    for(int i=0;i<totalprocess;i++)
    {
        wavg += waitingTime[i];
        tavg += tarrivaltime[i];

        cout<<proc[i].pno<<"\t\t"<<stime[i]<<"\t\t"<<ctime[i]<<"\t\t"<<tarrivaltime[i]<<"\t\t\t"<<waitingTime[i]<<'\n';
    }

    cout<<"Average waiting time is : "<<wavg/(float)totalprocess;
    cout<<" and average turnaround time is : "<<tavg/(float)totalprocess<<'\n';
}
```

Ln 10, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Win32

priority-scheduling.cpp - OS - Visual Studio Code

priority-scheduling.cpp

```cpp
        cout<<proc[i].pno<<"\t\t"<<stime[i]<<"\t\t"<<ctime[i]<<"\t\t"<<tarrivaltime[i]<<"\t\t\t"<<waitingTime[i]<<'\n';
    }

    cout<<"Average waiting time is : "<<wavg/(float)totalprocess;
    cout<<" and average turnaround time is : "<<tavg/(float)totalprocess<<'\n';
}
```

```cpp

int main()
{
    int arrivaltime[]={1,2,3,4,5};
    int bursttime[]={3,5,1,7,4};
    int priority[]={3,4,1,7,8};
    int totalprocess;
    cin>>totalprocess;

    for(int i=0;i<totalprocess;i++)
    {
        proc[i].arrivaltime=arrivaltime[i];
        proc[i].bursttime=bursttime[i];
        proc[i].pr=priority[i];
        proc[i].pno=i+1;
    }

    sort(proc,proc+totalprocess,cmp);
    ganttchart(totalprocess);

    return 0;
}
```

Ln 10, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Win32

Q3. Write a program using fork() and pipe() where one child process sends a message from one pipe to another child process and then returns a new message acknowledging the first message through another pipe to the first child process.

```
first string: Ritikesh is here
second string: Hello, it's Ritikesh
```

pipe-message.c - OS - Visual Studio Code

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/types.h>
#include<unistd.h>

int main()
{
    int fd1[2], fd2[2];
    char fixed_str[]="Ritikesh is here";
    printf("first string: %s\n", fixed_str);
    char input_str[100];
    pid_t p;

    if(pipe(fd1)==-1)
    {
        fprintf(stderr, "Pipe failed");
        return 1;
    }
    if(pipe(fd2)==-1)
    {
        fprintf(stderr, "Pipe failed");
        return 1;
    }

    scanf("%s", input_str);
    p=fork();
    if(p<0){
        fprintf(stderr,"Fork failed");
        return 1;
```

pipe-message.c - OS - Visual Studio Code

```c
    scanf("%s", input_str);
    p=fork();
    if(p<0){
        fprintf(stderr,"Fork failed");
        return 1;
    }
    else if(p>0){
        char finalstr[100];
        close(fd1[0]);
        write(fd1[1], input_str, strlen(input_str)+1);
        close(fd1[1]);
        wait(NULL);
        close(fd2[1]);
        read(fd2[0], finalstr, 100);
        printf("second string: %s\n", finalstr);
        close(fd2[0]);
    }
    else{
        close(fd1[1]);
        char finalstr[100];
        read(fd1[0], finalstr, 100);
        int k=strlen(finalstr);
        finalstr[k]='\0';
        close(fd1[0]);
        close(fd2[0]);
        write(fd2[1], finalstr, strlen(finalstr)+1);
        close(fd2[1]);
        exit(0);
    }
}
```