

**DEHRADUN CAMPUS**

# PRACTICAL FILE / TERM WORK

**OS LAB PCS-502**

**B.Tech CSE V**

**2023-24**

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GRAPHIC ERA HILL UNIVERSITY, DEHRADUN

**SUBMITTED TO SUBMITTED BY**

Mr. Nitin Thapliyal NAME: Nilesh Bhanot

ASST. PROFESSOR Examination Roll No.: 2118851

DEPARTMENT OF COMPUTER Course / Sem: B.Tech/5th SCIENCE & ENGG. SEC/ROLL NO: A/42

**COLLEGE ROLL NO. 42 EXAMINATION ROLL NO. 2118851**

A red and yellow logo

Description automatically generated

**DEHRADUN CAMPUS**

THIS IS TO CERTIFY THAT Mr. / Ms. Nilesh Bhanot HAS SATISFACTORILY COMPLETED ALL THE EXPERIMENTS IN THE LABORATORY OF THIS COLLEGE. THE COURSE OF THE EXPERIMENTS / TERM WORK of Operating System Lab(PCS-502) in partial fulfillment of requirement in 5th Semester of B.TECH (CSE) DEGREE COURSE PRESCRIBED BY GRAPHIC ERA HILL UNIVERSITY, DEHRADUN, DURING THE YEAR 2023-2024 .

CONCERNED FACULTY HEAD OF DEPARTMENT

NAME OF EXAMINER: SIGNATURE OF EXAMINER:

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING STUDENT LAB REPORT SHEET**

**OS LAB (PCS-502)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of Student ………Nilesh Bhanot………………. Mo. No……9528029993** | | | | |
| **Address Permanent: Banjarawala, Dehradun, Uttarakhand** | | | | |
| **Father’s Name: Surender Bhanot** | **Mo No: 8791047160** | |  |  |
| **Mother’s Name: Vidushi Bhanot** | **Mo No: 941151252** | |  |  |
| **Section: A Branch: CSE Semester: 5th** | **Class Roll No.: 42** | | |  |
| **Local Address: same as permanent Email:**[**nileshbhanot264@gmail.com**](mailto:nileshbhanot264@gmail.com)  **Grade** | | **A** | **B** | **C** |

**Marks** 5 3 1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S.** | **Name of the Experiment** | **D.O.P.** | **D.O.S** | **Grade** | **Grade** | **Total** | **Student’s** | **Teacher’s** |
| **No.** | **(Viva)** | **(Report** | **Marks** | **Signature** | **Signature** |
|  |  | **File)** | **(out of** |  |  |
|  |  |  | **10)** |  |  |
| **1** |  |  |  |  |  |  |  |  |
| **2** |  |  |  |  |  |  |  |  |
| **3** |  |  |  |  |  |  |  |  |
| **4** |  |  |  |  |  |  |  |  |
| **5** |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **6** |  |  |  |  |  |  |  |  |
| **7** |  |  |  |  |  |  |  |  |
| **8** |  |  |  |  |  |  |  |  |
| **9** |  |  |  |  |  |  |  |  |
| **10** |  |  |  |  |  |  |  |  |
| **11** |  |  |  |  |  |  |  |  |
| **12** |  |  |  |  |  |  |  |  |
| **13** |  |  |  |  |  |  |  |  |
| **14** |  |  |  |  |  |  |  |  |
| **15** |  |  |  |  |  |  |  |  |

PROGRAM 1

### Q: Write a C program using fork command.

**Fork( ) :**

The Fork system call is used for creating a new process in Linux, and Unix systems, which is called the child process, which runs concurrently with the process that makes the fork() call (parent process). After a new child process is created, both processes will execute the next instruction following the fork() system call.

### CODE:

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

int main()

{

fork();

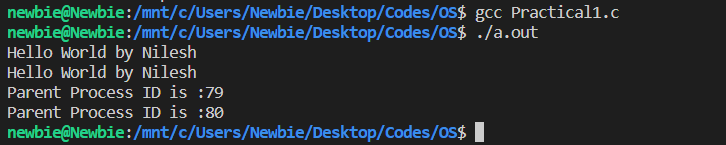
printf("Hello World by Rashmi\n");

printf("Parent Process ID is :%d\n",getpid());

return 0;

}

## OUTPUT



PROGRAM 2

### Q: WAP to demonstrate sum of even number when parent process is called and sum of odd numbers when child process is called using fork() system call.

### CODE:

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

int main()

{

int n;

printf("Total no of elements in an array\n");

scanf("%d",&n);

int arr[n];

printf("Enter the elements in an array\n");

for(int i=0;i<n;i++)

{

scanf("%d",&arr[i]);

}

int countEven=0,countOdd=0;

int pid=fork();

if(pid==0)

{

printf("Child Process is running\n");

printf("child Process id :%d\n",getpid());

printf("Parent Process id:%d\n",getppid());

for(int i=0;i<n;i++)

{

if(arr[i]%2!=0)

{

countOdd+=arr[i];

}

}

printf("Odd count:%d\n",countOdd);

printf("Child Process Completed\n");

}

else

{

printf("Parent Process is running\n");

printf("Parent Process id:%d\n",getpid());

for(int i=0;i<n;i++)

{

if(arr[i]%2==0)

{

countEven+=arr[i];

}

}

printf("Even count:%d\n",countEven);

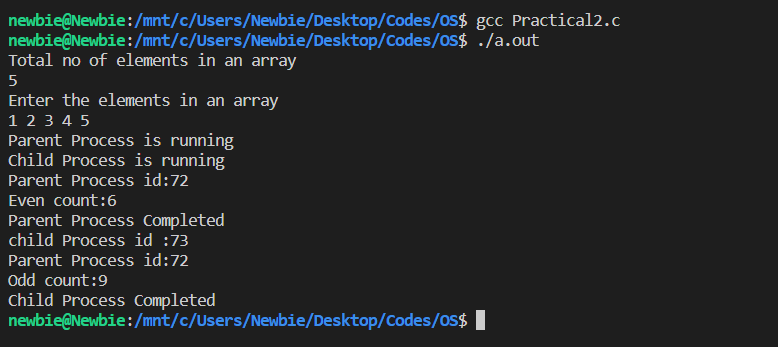
printf("Parent Process Completed\n");

}

return 0;

}

## OUTPUT



PROGRAM 3

**Q.WAP in C to Implement the wait System Call**

**CODE**

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

#include<sys/types.h>

#include<sys/wait.h>

int main()

{

int pid;

pid=fork();

if(pid==0)

{

printf("Inside the child Process\n");

exit(0);

}

else

{

wait(NULL);

printf("Inside the Parent Process\n");

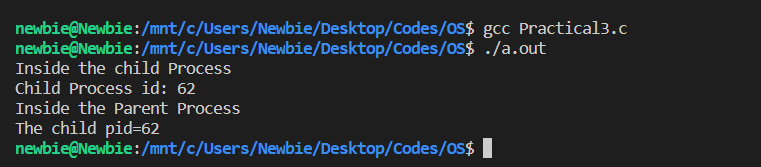
printf("The chid pid=%d\n",pid);

}

return 0;

}

**OUTPUT**



PROGRAM 4

**Q. Write a Program in C to Implement the First Come First Serve Algorithm(FCFS).**

**CODE:**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <string.h>

typedef struct Process

{

int at, bt, wt, ct, tat;

} Process;

int comp(const void \*a, const void \*b)

{

Process \*p1 = (Process \*)a;

Process \*p2 = (Process \*)b;

return (p1->at - p2->at);

}

int main()

{

int n;

scanf("%d", &n);

Process p[n];

for (int i = 0; i < n; i++)

scanf("%d %d", &p[i].at, &p[i].bt);

qsort(p, n, sizeof(Process), comp);

int time = 0;

p[0].wt = 0;

for (int i = 0; i < n; i++)

{

p[i].wt = time - p[i].at;

time += p[i].bt;

p[i].ct = time;

p[i].tat = p[i].bt + p[i].wt;

}

printf("PID\tAT\tBT\tWT\tCT\tTAT\n");

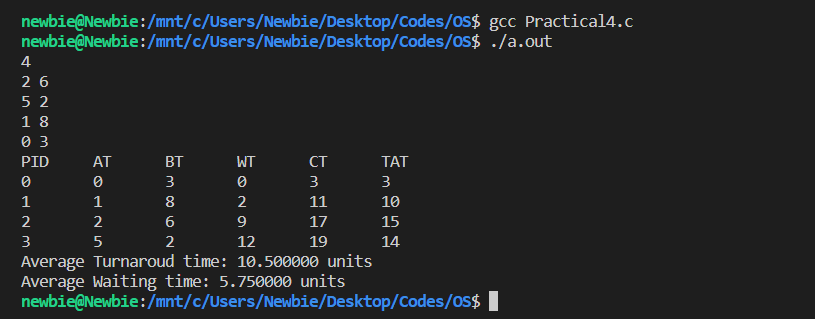
for (int i = 0; i < n; i++)

printf("%d\t%d\t%d\t%d\t%d\t%d\n", i, p[i].at, p[i].bt, p[i].wt, p[i].ct, p[i].tat);

return 0;

}

**OUTPUT**

****

PROGRAM 5

**Q. Write a Program in c to Implement the shortest job first Scheduling Algorithm**.

**CODE:**

#include <stdio.h>

#include <stdlib.h>

struct PCB

{

int pid, arr, brst, ct, wt, tat, isCompleted;

};

void sort(struct PCB p[], int n)

{

for (int i = 0; i < n - 1; i++)

{

for (int j = 0; j < n - i - 1; j++)

{

if (p[j].brst > p[j + 1].brst)

{

struct PCB temp = p[j];

p[j] = p[j + 1];

p[j + 1] = temp;

}

}

}

}

int main()

{

int n;

float aTat = 0.0, aWt = 0.0;

printf("Enter the no. of Process: ");

scanf("%d", &n);

printf("Enter the PID: ");

struct PCB \*p = malloc(n \* sizeof(struct PCB));

for (int i = 0; i < n; i++)

{

scanf("%d", &p[i].pid);

}

printf("Enter the Arrival Time: ");

for (int i = 0; i < n; i++)

{

scanf("%d", &p[i].arr);

}

printf("Enter the Burst Time: ");

for (int i = 0; i < n; i++)

{

scanf("%d", &p[i].brst);

p[i].isCompleted = 0;

}

sort(p, n);

int currTime = 0;

int countComplete = 0;

while (countComplete < n)

{

int sji = -1;

for (int i = 0; i < n; i++)

{

if (!p[i].isCompleted && p[i].arr <= currTime)

{

if (sji == -1 || p[i].brst < p[sji].brst)

sji = i;

}

}

if (sji == -1)

currTime++;

else

{

struct PCB \*process = &p[sji];

process->ct = currTime + process->brst;

process->tat = process->ct - process->arr;

process->wt = process->tat - process->brst;

aTat = aTat + process->tat;

aWt = aWt + process->wt;

currTime = process->ct;

process->isCompleted = 1;

countComplete++;

}

}

printf("PID\t\t AT\t\t BT\t\t CT\t\t TAT\t\t WT\t\t\n");

printf("------------------------------------------------------------------------------------------\n");

for (int i = 0; i < n; i++)

{

printf("%d\t\t %d\t\t %d\t\t %d\t\t %d\t\t %d\n", p[i].pid, p[i].arr, p[i].brst, p[i].ct,

p[i].tat, p[i].wt);

}

printf("\nAverage Turn Around Time is: %f", aTat / n);

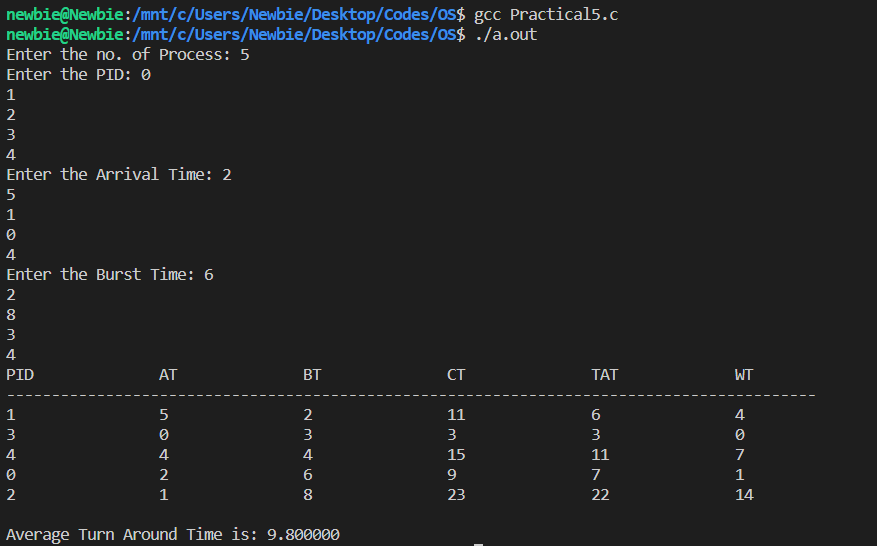
printf("\nAverage Waiting Time is: %f", aWt / n);

free(p);

return 0;

}

**OUTPUT**



PROGRAM 6

**Q. Write a Program in c to Implement the shortest Remaining time first Scheduling Algorithm**.

**CODE:**

#include <stdio.h>

#define max 100

int main()

{

int n, AT[max], WT[max], ET[max], TAT[max], CT[max], PID[max], RT[max], i, min, j, temp, maxCT;

double avg\_WT = 0, avg\_TAT = 0, thrpt;

printf("Please enter number of processes\n");

scanf("%d", &n);

if (n > max)

{

printf("Limit exceeded.");

}

else if (n < 1)

{

printf("No process exist or negative input.");

}

else

{

for (i = 0; i < n; i++)

{

scanf("%d", &AT[i]);

scanf("%d", &ET[i]);

PID[i] = i + 1;

}

// sorting for arrival time

for (i = 0; i < -1; i++)

{

min = i;

for (j = i + 1; j < n; j++)

{

if (AT[j] < AT[min])

{

min = j;

}

}

temp = AT[i];

AT[i] = AT[min];

AT[min] = temp;

temp = ET[i];

ET[i] = ET[min];

ET[min] = temp;

temp = PID[i];

PID[i] = PID[min];

PID[min] = temp;

}

for (i = 0; i < n; i++)

{

RT[i] = ET[i];

}

RT[n] = 9999;

j = 0;

for (min = 0; j != n; min++)

{

temp = n;

for (i = 0; i < n; i++)

{

if (AT[i] <= min && RT[i] < RT[temp] && RT[i] > 0)

{

temp = i;

}

}

RT[temp]--;

if (RT[temp] == 0)

{

j++;

thrpt = min + 1;

CT[temp] = thrpt;

if (maxCT < thrpt)

{

maxCT = thrpt;

}

}

}

printf("Process\tArrival Time\tExecution Time\tCompletion Time\tTAT\tWaiting Time\n");

for (i = 0; i < n; i++)

{

TAT[i] = CT[i] - AT[i];

WT[i] = TAT[i] - ET[i];

avg\_TAT = avg\_TAT + TAT[i];

avg\_WT = avg\_WT + WT[i];

printf("P%d\t%d\t\t%d\t\t%d\t\t%d\t%d\n", PID[i], AT[i], ET[i], CT[i], TAT[i], WT[i]);

}

avg\_TAT = (double)avg\_TAT / n;

avg\_WT = (double)avg\_WT / n;

thrpt = (double)n / (maxCT - AT[0]);

printf("\nAvg.TAT=%.2f\n", avg\_TAT);

printf("Avg.WT=%.2f\n", avg\_WT);

printf("Throughput=%.2f", thrpt);

}

return 0;

}

**OUTPUT**

