Date : 22-10-24

Exp. No. 10

Exp. Title Write a program to stimulate the work of shortest remaining time first scheduling. Experiment with different length jobs.

Page No. 33

ALGORITHM : Shortest remaining Time Implementation

PURPOSE : To represent the working of the shortest remaining time algorithm.

Input :- No. of process, Burst time of each process and Quantum time.

Output :- Average waiting time and Turnaround time

Date :

Exp. No.

Exp. Title

PROGRAM:

SOURCE CODE:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
int main()
{
int pid[15];
int bt[15];
int n;
Printf("Enter the number of processes:");
scanf("%d", &n);
int n;
Printf("Enter the number of process:");
Scanf("%d", &n)
Printf("Enter the process id of all the process:");
for (int i=0; i<n; i++)
{
scanf("%d", &pid[i]);
}
printf("Enter burst time of all the processes\n");
for (int i=0; i<n; i++)
{
scanf("%d", &bt[i]);
}
int i, wt[n];
wt[0] = 0;
for (i=1; i<n; i++)
{
```

Output:

Enter the number of process : 5
Enter process id of all process : 1 2 3 4 5
Enter burst time of all process : 10 8 5 7 9

Process ID	Burst time	Waiting time	Turnaround Time
1	10	0	10
2	8	10	18
3	5	18	23
4	7	23	30
5	9	30	39

Avg. waiting time : 16.200001
Avg. turn around time : 24.000000

Date :
Exp. No.

Exp. Title

Page No. 35

```
w[i] = bt[i-1] + wt[i-1];
}
printf ("Process\t Burst Time\t Waiting Time\t
        Turnaround Time\n");
float tat = 0.0;
float wat = 0.0;
for (i=0; i<n; i++)
{
printf ("%d\t\t", pid[i]);
printf ("%d\t\t", bt[i]);
printf ("%d\t\t", wt[i]);
printf ("%d\t\t", bt[i] + wt[i]);
printf ("\n");
twt = wt[i] + twt + wt[i];
tat = (wt[i] + bt[i]) tat + bt[i];
}
float att, awt;
awt = twt/n;
att = tat/n;
printf ("Avg. waiting time = %f\n", awt);
printf ("Avg. turn around time = %f", att);
}
```

Date : 15-10-24 Exp. No. 04	Exp. Title Write the program to implement the system call opendir(), readdir(), closedir().	Page No. 11

ALGORITHUM:

step 1: start the program.
step 2: In main function pass the arguements.
step 3: Create structures as stat buff and the variables as integer.
step 4: Open the directory.
step 5: Read the content of the directory
step 6): Display the contents of the directory.
step 7: close the directory.
step 8: Stop the program.

Date : Exp. No.	Exp. Title	Page No. 12

PROGRAM:

SOURCE CODE:

```
/* Recursively descent a directory hierachy pointing a file */

#include <stdio.h>
#include <dirent.h>
#include <errno.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>
Void read_file (const char * file path)
{
 FILE *file = fopen (filepath, "r");
 if (file == NULL)
 {
  perror ("Failed to open file");
 }
 char line [256];
 while (fgets (line, sizeof (line), file) != NULL)
 {
  Printf ("%.s", line);
 }
 fclose (file);
}
int main (int argc, char * argv [])
{
 if (argc != 2)
 {
  fprintf (stderr, "usage: %.s <directory_name>\n", argv[0]);
  return EXIT_FAILURE;
 }
```

output :-

```
file : 1.C
content of 1.C:
#include <stdio.h>
void main()
{
printf ("Hello");
printf ("welcome to OS lab");
}
```

Date :
Exp. No.

Exp. Title

Page No.
13

```
DIR *dirp = opendir (argv[1]);
if (dirp == NULL)
{
perror ("Failed to open directory");
return EXIT_FAILURE;
}
struct dirent *direntp;
while ((direntp = readdir (dirp)) != NULL)
{
if (strcmp (direntp->d_name, ".") != 0 && strcmp
                    (direntp->d_name, "..") != 0)
{
printf ("File: %s\n", direntp->d_name);
if (direntp->d_type == DT_REG)
{
char filepath [1024];
snprintf (filepath, sizeof (filepath), "%s/%s", argv[1],
                    direntp->d_name);
printf ("contents of %s:\n", filepath);
read_file (filepath);
}
printf ("\n");
}
}
if (closedir (dirp) == -1)
{
perror ("Failed to close directory");
return EXIT_FAILURE;
}
return EXIT_SUCCESS;
}
```

Anitha