NAME : K DHANUSH

REGISTER NUMBER : 192311094

COURSE : OPERATING SYSTEMS FOR MOBILE APPLICATIONS

COURSE CODE : CSA0497

1. PROCESS CREATION

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

int main() {     pid_t
pid = fork();

   if (pid < 0) {
perror("Fork failed");
exit(EXIT_FAILURE);     } else
if (pid == 0) {         // Child
process         execlp("/bin/ls",
"ls", NULL);
perror("execlp failed");
exit(EXIT_FAILURE);
   } else {
      // Parent process         wait(NULL);
prin ("Child process completed.\n");
   }

   return 0;
}
```

INPUT :

OUTPUT :

Main

Main.c

Child process completed.

2. FILE COPYING

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

void copyFile(const char *source, const char *des na on) {
FILE *srcFile, *destFile;     char ch;

   srcFile = fopen(source, "r");     if
(srcFile == NULL) {         perror("Error
opening source file");
exit(EXIT_FAILURE);
   }

   destFile = fopen(des na on, "w");     if
(destFile == NULL) {         perror("Error
opening des na on file");
       fclose(srcFile);
exit(EXIT_FAILURE);
   }

   while ((ch = fgetc(srcFile)) != EOF) {
fputc(ch, destFile);
   }
```

```
    fclose(srcFile);
fclose(destFile);
}

int main() {          const char *sourceFile =
"source.txt";     const char *des na onFile = "des
na on.txt";

    copyFile(sourceFile, des na onFile);
prin ("File copied successfully.\n");

    return 0;
}
INPUT :
OUTPUT :
Usage: /tmp/949K6IhL8a/main.o <source> <des na on>
```

3. FCFS SCHEDULING

```
#include <stdio.h>

struct Process {
    int id;
    int burst_ me;      int
wai ng_ me;         int
turnaround_ me;
};

void findWai ngTime(struct Process proc[], int n) {
proc[0].wai ng_ me = 0;

    for (int i = 1; i < n; i++) {
```

```
        proc[i].wai ng_ me = proc[i - 1].wai ng_ me + proc[i -
1].burst_ me;
    }
}

void findTurnaroundTime(struct Process proc[], int n) {
    for (int i = 0; i < n; i++) {
        proc[i].turnaround_ me = proc[i].wai ng_ me +
proc[i].burst_ me;
    }
}

void findavgTime(struct Process proc[], int n)
{                    findWai    ngTime(proc,    n);
findTurnaroundTime(proc, n);

    float total_wai ng_ me = 0, total_turnaround_ me = 0;

    for (int i = 0; i < n; i++) {
        total_wai ng_ me += proc[i].wai ng_ me;
total_turnaround_ me += proc[i].turnaround_ me;
    }

    prin ("Average wai ng me: %.2f\n", total_wai ng_ me / n);
    prin ("Average turnaround me: %.2f\n", total_turnaround_
me / n);
}

int main() {    struct Process proc[] = { {1, 10},
{2,  5},  {3,  8}  };        int  n  =  sizeof(proc)  /
```

```
sizeof(proc[0]);    findavgTime(proc, n);    return
0;
}
```

INPUT :

OUTPUT:

Average wai ng me: 8.33

Average turnaround me: 16.00

4. SJF SCHEDULING

```
#include <stdio.h>

struct Process {
   int id;
   int burst_ me;
};

void findWai ngTime(struct Process proc[], int n, int
wai ng_ me[]) {     wai ng_ me[0] = 0;

   for (int i = 1; i < n; i++) {
      wai ng_ me[i] = wai ng_ me[i - 1] + proc[i -
1].burst_ me;
   }
}

void findTurnAroundTime(struct Process proc[], int n, int
wai ng_ me[], int turn_around_ me[]) {
   for (int i = 0; i < n; i++) {
      turn_around_ me[i] = proc[i].burst_ me + wai ng_ me[i];
   }
```

```
}

void findavgTime(struct Process proc[], int n) {
int wai ng_ me[n], turn_around_ me[n];

   findWai ngTime(proc, n, wai ng_ me);
findTurnAroundTime(proc, n, wai ng_ me, turn_around_
me);

   float total_wai ng_ me = 0, total_turn_around_ me = 0;

   for (int i = 0; i < n; i++) {
      total_wai ng_ me += wai ng_ me[i];
total_turn_around_ me += turn_around_ me[i];
   }

   prin ("Average wai ng me: %.2f\n", total_wai ng_ me / n);
   prin ("Average turn around me: %.2f\n", total_turn_around_
me / n);
}

void sortProcesses(struct Process proc[], int n) {
struct Process temp;
   for (int i = 0; i < n - 1; i++) {         for (int j = 0; j <
n - i - 1; j++) {            if (proc[j].burst_ me > proc[j +
1].burst_ me) {               temp = proc[j];
proc[j] = proc[j + 1];
            proc[j + 1] = temp;
         }
      }
   }
```

```
}

int main() {    struct Process proc[] = { {1, 6}, {2, 8},
{3, 7}, {4, 3} };          int n = sizeof(proc) /
sizeof(proc[0]);

   sortProcesses(proc, n);
findavgTime(proc, n);

   return 0;
}
```

INPUT :

OUTPUT :

Average wai ng me: 7.00

Average turn around me: 13.00

5. PRIORITY SCHEDULING

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

struct Process {
int id;
   int burst_ me;
int priority;
};

void priority_scheduling(struct Process proc[], int n) {
struct Process temp;

   // Sort processes by priority     for (int i = 0;
i < n - 1; i++) {        for (int j = 0; j < n - i - 1;
```

```
j++) {            if (proc[j].priority > proc[j +
1].priority) {                temp = proc[j];
proc[j] = proc[j + 1];                proc[j + 1] =
temp;
            }
        }
    }

    prin ("Process ID\tBurst Time\tPriority\n");
    for (int i = 0; i < n; i++) {
        prin ("%d\t\t%d\t\t%d\n", proc[i].id, proc[i].burst_ me,
proc[i].priority);
    }
}

int main() {
    struct Process proc[] = {
        {1, 10, 2},
        {2, 5, 1},
        {3, 8, 3}
    };
    int n = sizeof(proc) / sizeof(proc[0]);
priority_scheduling(proc, n);

    return 0;
}
```

INPUT :

OUTPUT :

Process ID	Burst Time	Priority
2	5	
1 1	10	
2 3	8	
3		