Netaji Subhas University of Technology

Lab Report

Operating Systems

Name	**Kushagra Lakhwani**
Roll No	***2021UCI8036***
Semester	**4th**
Course	***CICPC09***

Department of Computer Science & Engineering

February 24, 2023

Abstract

The practical lab report *"Operating Systems"* is the original and unmodified content submitted by *Kushagra Lakhwani* (Roll No. 2021UCI8036).

The report is submitted to *Dr. Manoj Kumar* Department of Computer Science and Engineering, NSUT, Delhi, for the partial fulfillment of the requirements of the course (CICPC09).

Contents

1 Process Creation and Termination

1.1 Process Creation

1.1.1 The `fork()` System Call

A process is created by the `fork()` system call. The `fork()` system call creates a child process that is an exact copy of the parent process.

```c
/**
 * Forks a new process and checks for parent and child processes
 */

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

int main(int argc, char *argv[]) {
  pid_t pid;
  pid = fork();

  if (pid == 0) {
    printf("Child process pid: %d\n", (int)getpid());
    exit(EXIT_SUCCESS);
  } else if (pid > 0) {
    printf("Parent process pid: %d", (int)getpid());
  } else {
    printf("Fork failed");
    exit(EXIT_FAILURE);
  }

  return 0;
}
```

The `fork()` system call returns twice. The first time it returns the process ID of the child process to the parent process. The second time it returns 0 to the child process.

```
Parent process pid: 96732
Child process pid: 96733
```

1.2 Process Termination

1.2.1 The exit() System Call

The exit() system call terminates the calling process. The exit() system call takes an integer argument that is returned to the parent process as the child's exit status.

```c
// In file <unistd.h>
/* Terminate program execution with the low-order 8 bits of STATUS.  */
extern void _exit(int __status) __attribute__((__noreturn__));
```

2 CPU Scheduling: FCFS

Implementation of CPU scheduling in a first come first serve (FCFS) manner.

```c
/**
 * Implementation of FCFS scheduling algorithm
 */

#include <stdio.h>

void find_waiting_time(int processes[], int n, int bt[], int wt[]) {
  wt[0] = 0;
  for (int i = 1; i < n; i++) {
    wt[i] = bt[i - 1] + wt[i - 1];
  }
}

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

void find_avg_time(int processes[], int n, int bt[]) {
  int wt[n], tat[n], total_wt = 0, total_tat = 0;
  find_waiting_time(processes, n, bt, wt);
  find_turn_around_time(processes, n, bt, wt, tat);
  printf("Processes  Burst Time   Waiting Time Turn Around Time\n");
  for (int i = 0; i < n; i++) {
    total_wt = total_wt + wt[i];
    total_tat = total_tat + tat[i];
    printf("%d\t   ", i + 1);
    printf("%d\t\t", bt[i]);
    printf("%d\t\t", wt[i]);
    printf("%d\t\n", tat[i]);
  }
  printf("Average waiting time = %f\n", (float)total_wt / (float)n);
```

```
35    printf("Average turn around time = %f", (float)total_tat / (float)n);
36  }
37
38  int main() {
39    int processes[] = {1, 2, 3};
40    int n = sizeof processes / sizeof processes[0];
41    int burst_time[] = {10, 5, 8};
42    find_avg_time(processes, n, burst_time);
43    return 0;
44  }
```

The scheduler selects the process that has been waiting the longest.

```
Processes   Burst Time    Waiting Time     Turn Around Time
1           10            0               10
2           5             10              15
3           8             15              23
Average waiting time = 8.333333
Average turn around time = 16.000000
```

3 CPU Scheduling: Priority

Implementation of CPU scheduling using a "priority" based approach using assigned ranks/priority algorithms.

A major problem with priority scheduling is indefinite blocking or starvation. A solution to the problem of indefinite blockage of the low-priority process is aging. Aging is a technique of gradually increasing the priority of processes that wait in the system for a long period of time.

```
1   /**
2    * CPU priority scheduler.
3    */
4
5   #include <stdio.h>
6
7   typedef struct Process {
8     int pid;
9     int priority;
10    int burst;
11  } Process;
12
13  int compare(Process a, Process b) { return a.priority < b.priority; }
14
```

```
15 void sort(Process *p, int n) {
16   for (int i = 0; i < n; i++) {
17     for (int j = i + 1; j < n; j++) {
18       if (compare(p[j], p[i])) {
19         Process temp = p[i];
20         p[i] = p[j];
21         p[j] = temp;
22       }
23     }
24   }
25 }
26
27 void findWaitingTime(Process proc[], int n, int wt[]) {
28   wt[0] = 0;
29   for (int i = 1; i < n; i++)
30     wt[i] = proc[i - 1].burst + wt[i - 1];
31 }
32
33 void findTurnAroundTime(Process proc[], int n, int wt[], int tat[]) {
34   for (int i = 0; i < n; i++)
35     tat[i] = proc[i].burst + wt[i];
36 }
37
38 void findavgTime(Process proc[], int n) {
39   int wt[n], tat[n], total_wt = 0, total_tat = 0;
40
41   findWaitingTime(proc, n, wt);
42   findTurnAroundTime(proc, n, wt, tat);
43
44   printf("Processes Burst time Waiting time Turn around time\n");
45   for (int i = 0; i < n; i++) {
46     total_wt = total_wt + wt[i];
47     total_tat = total_tat + tat[i];
48     printf("%d\t\t", proc[i].pid);
49     printf("%d\t\t", proc[i].burst);
50     printf("%d\t\t", wt[i]);
51     printf("%d", tat[i]);
52     printf("\n");
53   }
54
55   printf("Average waiting time = %f", (float)total_wt / (float)n);
56   printf("Average turn around time = %f", (float)total_tat / (float)n);
57 }
```

```
58
59 void priorityScheduling(Process proc[], int n) {
60   sort(proc, n);
61   printf("Order in which processes gets executed\n");
62   for (int i = 0; i < n; i++)
63     printf("%d ", proc[i].pid);
64   printf("\n");
65   findavgTime(proc, n);
66 }
67
68 int main() {
69   Process proc[] = {{1, 2, 100}, {2, 1, 19}, {3, 1, 27}, {4, 1, 25}};
70   int n = sizeof proc / sizeof proc[0];
71   priorityScheduling(proc, n);
72   return 0;
73 }
```

```
Order in which processes gets executed
2 3 4 1
Process          BurstTime       WaitingTime      TurnAroundTime
2                19               0                19
3                27               19               46
4                25               46               71
1                100              71               171
Average waiting time = 34.000000
Average turn around time = 76.750000
```