Tribhuvan University
Faculty of Humanities and Social Sciences

Lab report on:
Operating System Lab 3:
Process

Submitted to:
Mr. Roshan Maharjan,
Er. Himal Chand Thapa ,
Department of Computer Application,
Himalaya College of Engineering,
Chyasal,Lalitpur

Submitted by:
Sujal Gurung
Roll no: 34
BCA II/IV
September 20, 2023

1 Objectives

- to learn about system call APIs available through `unistd.h` header file
- to write our own implementation of the **FCFS** scheduling algorithm

2 Introduction

System calls are an OS feature through which programs can request services from OS. The `unistd.h` file allows using these calls in POSIX operating systems (these include UNIX, linux & macOS).

A process is a running instance of a program. Each process is assigned unique process id(`pid`). A process can launch another process, creating a parent-child relation between them. It is upto the OS developers to implement how the OS schedules which process to run when many of them are waiting. Process scheduling algorithms are concerned with some metrics like:

- **Arrival time**: timestamp when process arrives in memory
- **Completion time**: timestamp when process finishes execution
- **Burst time**: How much time a process needs to execute.
- **Turn-around time**: How much time it took from arriving to executing process.
  formula: `completion time - arrival time`
- **Wait time**: How much time a process has to spend waiting to be run.
  formula: `turnaround time - burst time`

Out of the many popular process scheduling algorithms, we implemented First Come First Serve(FCFS). This simply runs whichever process arrived first.

3 Lab Work

3.1 Write a program to print the process Id.

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

void main() {
    printf("Process id is: %d", getpid());
    printf("\nParent process id is: %d", getppid());
}
```

Output

```
Process id is: 85715          # pid of running c program
Parent process id is: 85364   # pid of parent(terminal that executed the program)
```

3.2 Write a program to create child process using `fork()` system call

The `fork()` system call creats a child process that runs the same program as the parent. It returns:

- 0 to child process
- child's `pid` that is a positive integer to parent process
- -1 if error occurs

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

void main() {
    int child = fork();
    if (child == -1) {
        printf("\nError creating child process");
        exit(0);
    }
    else if (child == 0) {
        printf("\nCurrently in child(pid %d)", getpid());
    }
    else {
        printf("\nCurrently in parent(pid %d) that created child(pid %d)",
        ↪  getpid(), child);
    }
}
```

Output

```
Currently in parent(pid 94004) that created child(pid 94005)
Currently in child(pid 94005)
```

3.3 Write a program to find the Turnaround time, waiting time using First come first serve scheduling algorithm.

Here, we assume arrival time for every process is 0.

```
int main() {
  int bt[20], wt[20], tat[20], i, n;
  float wtsum, tatsum;

  printf("Enter the number of processes -- ");
  scanf("%d", &n);
  for(i=0; i<n; i++) {
      printf("Enter Burst Time for Process %d -- ", i);
      scanf("%d", &bt[i]);
  }

  wt[0] = wtsum = 0;
  tat[0] = tatsum = bt[0];
  for(i=1;i<n;i++) {
      wt[i] = wt[i-1] + bt[i-1];
      tat[i] = tat[i-1] +bt[i];
      wtsum = wtsum + wt[i];
```

```
      tatsum = tatsum + tat[i];
  }

  printf("\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");
  for(i=0;i<n;i++) {
    printf("\t P%d \t\t %d \t\t %d \t\t %d", i, bt[i], wt[i], tat[i]);
  }
  printf("\nAverage Waiting Time -- %f", wtsum/n);
  printf("\nAverage Turnaround Time -- %f", tatsum/n);
  return 0;
}
```

Output

```
Enter the number of processes  4
Enter Burst Time for Process 0 -- 4
Enter Burst Time for Process 1 -- 1
Enter Burst Time for Process 2 -- 2
Enter Burst Time for Process 3 -- 3

PROCESS        BURST TIME       WAITING TIME    TURNAROUND TIME
P0              4               0               4
P1              1               4               5
P2              2               5               7
P3              3               7               10
Average Waiting Time -- 4.000000
Average Turnaround Time -- 6.500000
```

4 Conclusion

Thus, we learned about basic system call APIs that allow us to handle processes. We also implemented the FCFS process scheduling algorithm that we had learned in class.