Assignment 2

Aryaman Srivastava 2210110206

1. Write a C program to create child processes of a main process using fork system call for N times. N can be any number ranging from 1 to 10. Also print a message with each process saying "Hello I am a child process\ parent process. My Pid is ="__". Here, the child process should print “I am a child process” and exit. The parent should wait to reap the child, print a message “I am a parent process” after reaping the child, and then exit. Analyze the number of child processes while varying the value of N & mention in the report.

Ans. This program creates a specified number of child processes (1 to 10) using fork(). Each child process prints its PID and the PID of its parent. After creating all child processes, the parent process waits for each child to complete using wait() and prints a message indicating it has reaped a child. This demonstrates basic process creation and management using fork() and wait(). The program ensures the number of child processes is within the valid range and handles errors in forking. Finally, it ensures proper termination of both parent and child processes using exit() with appropriate exit statuses.

```
aryaman@DESKTOP-4MHM220:~/ASSIGNMENT2/Q1$ gcc child_processes.c -o child_processes
aryaman@DESKTOP-4MHM220:~/ASSIGNMENT2/Q1$ ./child_processes
Enter the number of child processes (1 to 10): 3
Hello, I am a child process. My PID is 15690. My parent's PID is 15688.
Hello, I am a child process. My PID is 15691. My parent's PID is 15688.
Hello, I am a child process. My PID is 15692. My parent's PID is 15688.
I am a parent process. Reaped a child.
I am a parent process. Reaped a child.
I am a parent process. Reaped a child.
```

2. Write a C program in which the main program accepts the integers to be sorted. Main program uses the FORK system call to create a new process called a child process. Parent process sorts the integers using sorting algorithm and waits for child process using WAIT system call to sort the integers using any sorting algorithm. Also demonstrate zombie and orphan states.

Ans. This program takes input of an array size and elements from the user. It then creates a child process using fork(). The child process sorts a copy of the array using bubble sort, while the parent process sorts the original array using selection sort. After sorting, both processes display their respective sorted arrays. The parent process also waits for the child process to complete to prevent a zombie state. This program demonstrates the use of fork() to create a child process, allowing for parallel execution of different sorting algorithms on the same array.

```
aryaman@DESKTOP-4MHM220:~/ASSIGNMENT2/Q2$ gcc ZombieOrphan.c -o ZombieOrphan
aryaman@DESKTOP-4MHM220:~/ASSIGNMENT2/Q2$ ./ZombieOrphan
Please provide the count of numbers: 3
Enter 3 numbers: 9
8
6
Parent process is sorting using selection sort:
Sorted array by parent process: 6 8 9
Child process is sorting using bubble sort:
Sorted array by child process: 6 8 9
```

3. Write a C program in which the main program accepts an integer array. Main program uses the FORK system call to create a new process called a child process. Parent process sorts an integer array and passes the sorted array to child process through the command line arguments of EXEC system call. The child process uses EXEC system call to load new program that uses this sorted array for performing the binary search to search the item in the array.

Ans.

Child.c : This program implements binary search to find a specified element within a sorted array. It first takes input for the size of the array and then fills the array with elements provided as arguments. After that, it prompts the user to enter the element to search for. The binary search algorithm locates the element in the array, printing its index if found, or indicating that the element is not present if not found. This algorithm divides the search space in half at each step, making it highly efficient for large data.

Parent.c : This program sorts an array of integers using the bubble sort algorithm. It prompts the user to enter the number of elements and the elements as well. Then, it forks a child process to perform the sorting. The child process sorts the array using the bubble sort function and executes another program named "child" with the sorted array as arguments. The parent process then waits for the child process to finish executing "child" and then prints a message indicating its completion. This program demonstrates inter-process communication and execution via forking and exec functions in Linux.

```
aryaman@DESKTOP-4MHM220:~/ASSIGNMENT2/Q3$ gcc -o parent parent.c
aryaman@DESKTOP-4MHM220:~/ASSIGNMENT2/Q3$ gcc -o child child.c
aryaman@DESKTOP-4MHM220:~/ASSIGNMENT2/Q3$ ./parent
Enter the number of elements: 4
Enter the elements:
1 2 3 4
Enter the element to search: 3
Element 3 found at index 2
Parent process done.
```

4. Write a c program to simulate the FCFS scheduling algorithm.

Ans. This program simulates First Come First Serve (FCFS) scheduling algorithm for a set of processes. It prompts the user to input the number of processes and their burst times. Then, it calculates the waiting

time and turnaround time for each process. Finally, it computes the average waiting time and average turnaround time across all processes and displays them along with the individual process. The waiting time is the total time a process spends waiting in the ready queue, and the turnaround time is the total time taken by a process from submission to completion.

```
aryaman@DESKTOP-4MHM220:~$ cd ASSIGNMENT2/Q4
aryaman@DESKTOP-4MHM220:~/ASSIGNMENT2/Q4$ gcc -o main main.c
aryaman@DESKTOP-4MHM220:~/ASSIGNMENT2/Q4$ ./main
Enter the number of processes in the ready queue: 5
Enter the burst time for each process:
Burst time for process 1: 1
Burst time for process 2: 2
Burst time for process 3: 3
Burst time for process 4: 4
Burst time for process 5: 5

Process Burst Time      Waiting Time    Turnaround Time
1       1               0               1
2       2               1               3
3       3               3               6
4       4               6               10
5       5               10              15

Average Waiting Time: 4.00
Average Turnaround Time: 7.00
```

5. Write a program in C to stimulate the CPU scheduling algorithm Shortest job first (Non-Preemption).

Ans. This program sorts tasks by their shortest time needed to finish, following the Shortest Job First (SJF) algorithm. It calculates how long each task has to wait by adding up the time spent on earlier tasks. It also calculates the total time each task takes from when it arrives until it is done. After these calculations the program finds the average waiting and turnaround times and displays the results.

```
aryaman@DESKTOP-4MHM220:~/ASSIGNMENT2/Q5$ gcc -o main main.c
aryaman@DESKTOP-4MHM220:~/ASSIGNMENT2/Q5$ ./main
Enter the number of jobs: 6
Enter duration for each job:
Duration for job 1: 8
Duration for job 2: 6
Duration for job 3: 4
Duration for job 4: 2
Duration for job 5: 5
Duration for job 6: 3

Job     Duration        Waiting Time    Turnaround Time
4       2               0               2
6       3               2               5
3       4               5               9
5       5               9               14
2       6               14              20
1       8               20              28

Average Waiting Time: 8.33
Average Turnaround Time: 13.00
```

6. Write a C program to simulate the Shortest Remaining Time First scheduling algorithm.

Ans. This program implements the Shortest Remaining Time First (SRTF) scheduling algorithm. It prompts the user to input the number of processes, their arrival times, and burst times. Then, it simulates the scheduling process, selecting the process with the shortest remaining time at each time step. It calculates the turnaround time and waiting time for each process, updating them as processes complete. Finally, it prints a table displaying process details along with their respective turnaround and waiting times, as well as the average turnaround and waiting times for all processes.

```
aryaman@DESKTOP-4MHM220:~/ASSIGNMENT2/Q6$ gcc -o main main.c
aryaman@DESKTOP-4MHM220:~/ASSIGNMENT2/Q6$ ./main
Enter the number of processes: 5
Enter arrival time and burst time for each process:
Arrival time for process 1: 6
Burst time for process 1: 4
Arrival time for process 2: 8
Burst time for process 2: 3
Arrival time for process 3: 7
Burst time for process 3: 2
Arrival time for process 4: 5
Burst time for process 4: 3
Arrival time for process 5: 6
Burst time for process 5: 7

Process Arrival Time    Burst Time      Turnaround Time Waiting Time
1       6               4               18.00           14.00
2       8               3               16.00           13.00
3       7               2               17.00           15.00
4       5               3               19.00           16.00
5       6               7               18.00           11.00

Average Waiting Time: 4.20
Average Turnaround Time: 8.00
```

7. Write a C program to simulate the CPU scheduling algorithm round-robin.

Ans. This program simulates the Round Robin scheduling algorithm for a set of processes. It prompts the user to input the number of processes and the time quantum. Then, it takes the burst time for each process as input. It iterates through the processes, executing them in a round-robin manner, with each process given a time quantum to execute. It calculates waiting time and turnaround time for each process. Finally, it prints out a table displaying process details including burst time, waiting time, and turnaround time, along with average waiting time and average turnaround time for all processes.

```
aryaman@DESKTOP-4MHM220:~/ASSIGNMENT2/Q7$ gcc -o main main.c
aryaman@DESKTOP-4MHM220:~/ASSIGNMENT2/Q7$ ./main
Enter the number of processes: 6
Enter time quantum: 9
Enter burst time for each process:
Burst time for process 1: 7
Burst time for process 2: 5
Burst time for process 3: 6
Burst time for process 4: 4
Burst time for process 5: 2
Burst time for process 6: 3

Process Burst Time      Waiting Time    Turnaround Time
1       7               0               7
2       5               7               12
3       6               12              18
4       4               18              22
5       2               22              24
6       3               24              27

Average Waiting Time: 13.83
Average Turnaround Time: 18.33
```