Login AS fatos40

Name RAMRATAN SHARMA

Roll Number 205120081

5. (P-I) Write a C program to a create child process and display the process ID of parent and child processes also show the Zombie property of it.

```
#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/time.h>

#include <unistd.h>


int main()

{

   int pid;

   printf("\nCalling fork\n");

   pid = fork();

   char cmd[] = "ps -e -o s,pid,ppid";

   if(pid == 0)

   {

      printf("\nChild process started.");

      printf("\nchild process id is: %d", getpid());

      printf("\nParent process id is: %d", getppid());

      system(cmd);

      exit(0);

   }

   else if (pid > 0)
```

```
  {
     printf("\nparent process started.");
     printf("\nParent process id: %d", getpid());
     printf("\nPutting parent process to sleep state");
     sleep(30);
     system(cmd);
  }
  return 0;
}
```

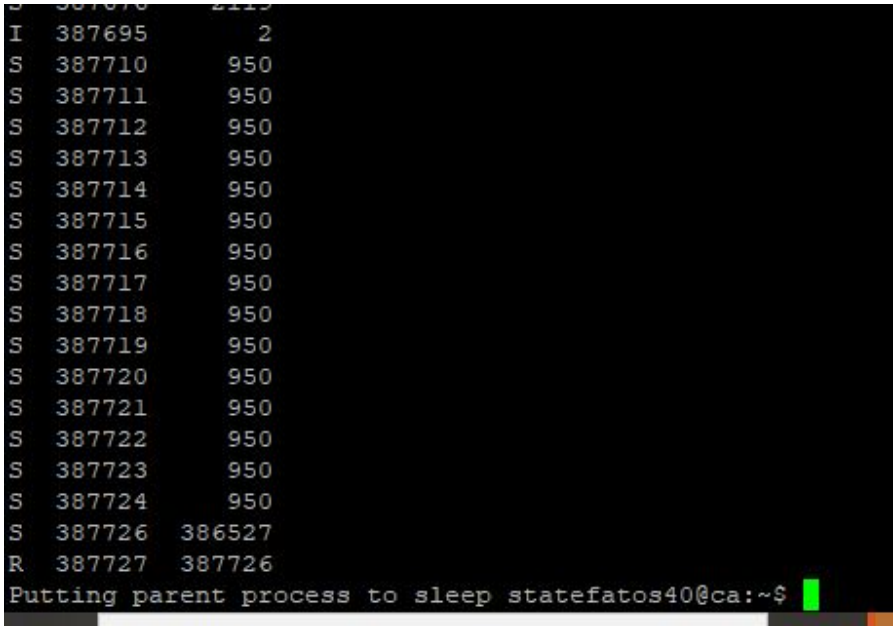
fatos40@ca: ~

```
fatos40@ca:~$ gcc ql.c
fatos40@ca:~$ ./a.out

Calling fork

parent process started.
Parent process id: 386527

Child process started.
child process id is: 386528
S     PID    PPID
S       1       0
S       2       0
I       3       2
I       4       2
I       6       2
I      10       2
S      11       2
I      12       2
S      13       2
S      14       2
S      15       2
S      16       2
S      17       2
S      18       2
```



5 (P-II) Write a C program to implement a task assignment problem for the processor, based on the task burst time (Hint: smallest burst time task get service first-Show the necessary outcomes).

```
#include <stdio.h>

#include <stdlib.h>

void lru(int arr[], int n, int frames);


int search(int arr[], int n, int key)

{

   for (int i = 0; i < n; i++)

   {

      if (arr[i] == key)

         return i;

   }

   return -1;

}
```

```
int main()
{
   int arr[] = {1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6};
   int frames = 4;
   lru(arr, 12, frames);
   return 0;
}
void lru(int arr[], int n, int frames)
{
  printf("Pages: ");
   for (int i = 0; i < n; i++)
   {
      printf(" %d ", arr[i]);
   }
   printf("\n\n");
   int hits = 0, miss = 0, b;
   int *buffer;
   buffer = (int *)malloc(sizeof(int) * frames);
   for (int i = 0; i < frames; i++)
   {
      buffer[i] = 0;
   }
   int index, temp;
   for (int p = 0; p < n; p++)
```

```
{
    index = search(buffer, frames, arr[p]);
    if (index >= 0)
    {
        hits++;
        b = 1;
        temp = buffer[index];
        for (int i = index - 1; i >= 0; i--)
        {
            buffer[i + 1] = buffer[i];
        }
        buffer[0] = temp;
    }
    else
    {
        miss++;
        for (int i = frames - 2; i >= 0; i--)
        {
            buffer[i + 1] = buffer[i];
        }
        buffer[0] = arr[p];
        b = 0;
    }

}
```

```
    float hitp = ((float)(hits*100)/n);
    float misp = ((float)(miss*100)/n);
    printf("Page Hits %d, Page Faults %d, Hit Percent - %f, Page fault Percent- %f\n", hits, miss,hitp,misp);
}
```