Experiment 8
System Calls

Done By: Rohit Karunakaran
Roll No: 58
Date Of Submission: 15-09-2021

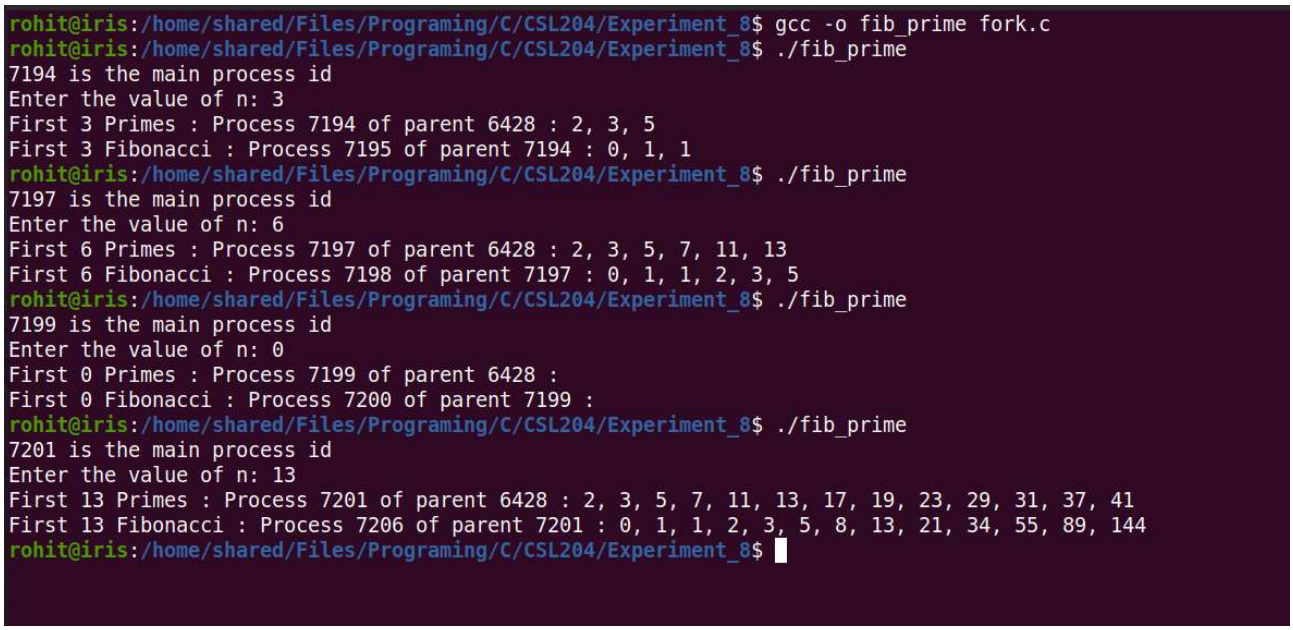
1. Program to accept the limiting value ‘n’as input and generate the Fibonacci sequence of n numbers using the child process while the parent process generate the first n prime numbers

Program Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/wait.h>

int* fibonacci(int n){
    if(n<0) return NULL;
    int* fib = (int *)malloc (sizeof(int)*n);
    fib[0] = 0;
    if(n>0){
        fib[1] = 1;
        int f3;
        for(int i = 2; i<n; i++){
            f3 = fib[i-1]+fib[i-2];
            fib[i] = f3;
        }
    }
    return fib;
}

int* n_primes(int n){
    if(n<0) return NULL;
    int* prime = (int *) malloc(sizeof(int)*n);
    prime[0] = 2;
    int i = 3;
    int k = 1;
    while(k<n){
        int j;
        for( j = 2; j≤i/2; j++){
            if(i%j == 0) break;
        }
        if(!(j≤i/2))
            prime[k++] = i;
        i++;
    }
    return prime;
}

void print_arr(int* arr,int n){
    printf("Process %d of parent %d : ", (int)getpid(), (int)getppid());
    for(int i=0;i<n;i++){
        if(i==n-1)
        printf("%d ",arr[i]);
        else
        printf("%d, ",arr[i]);
    }
    printf("\n");
```

```
}

int main(){
    int n;
    pid_t parentid = getpid();
    printf("%d is the main process id\n", getpid());
    printf("Enter the value of n: ");
    scanf("%d",&n);
    int cpid = (int)fork();
    if(cpid == 0 ){
        int* fib_arr = fibonacci(n);
        printf("First %d Fibonacci : ",n);
        print_arr(fib_arr, n);
        exit(0);
    }
    int* primes = n_primes(n);
    printf("First %d Primes : ",n);
    print_arr(primes,n);
    wait(NULL);
    return 0;

 }
```

Screenshots:



2. Generate an N level hierarchy of processes and also display the parent id of process.

Program Code:

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

int main(){
    int n ;
    int i = 0;
    int p = 0;
    int ch = 0;
    printf("Enter the number of process to be created: ");
    scanf("%d", &n);

    for(i =0; i<n ; i++){
        p++;
        if(ch == 0){
            printf("Line %d:  Process id: %d, Parent id: %d \n",p,getpid(),getppid());
            ch = fork();
        }else{
            break;
        }
    }
    wait(NULL);
    return 0;
}
```

Screenshots:

```
rohit@iris:/home/shared/Files/Programing/C/CSL204/Experiment_8$ ./n_fork
Enter the number of process to be created: 5
Line 1:  Process id: 8395, Parent id: 6428
Line 2:  Process id: 8397, Parent id: 8395
Line 3:  Process id: 8398, Parent id: 8397
Line 4:  Process id: 8399, Parent id: 8398
Line 5:  Process id: 8400, Parent id: 8399
rohit@iris:/home/shared/Files/Programing/C/CSL204/Experiment_8$ ./n_fork
Enter the number of process to be created: 10
Line 1:  Process id: 8402, Parent id: 6428
Line 2:  Process id: 8405, Parent id: 8402
Line 3:  Process id: 8406, Parent id: 8405
Line 4:  Process id: 8407, Parent id: 8406
Line 5:  Process id: 8408, Parent id: 8407
Line 6:  Process id: 8409, Parent id: 8408
Line 7:  Process id: 8410, Parent id: 8409
Line 8:  Process id: 8411, Parent id: 8410
Line 9:  Process id: 8412, Parent id: 8411
Line 10:  Process id: 8413, Parent id: 8412
rohit@iris:/home/shared/Files/Programing/C/CSL204/Experiment_8$
```

3.

Program Code:

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

int main(){
    printf("In process A, pid: %d\n",getpid());
    int b_id = fork();
    if(b_id == 0){ // create B
        printf("In process B pid: %d, Parent pid: %d\n",getpid(),getppid());
        int d_id = fork();
        if(d_id == 0){
            printf("In process D, pid: %d, Parent pid: %d\n",getpid(),getppid());
            int h_id = fork();
            if(h_id == 0){
                printf("In process H, pid: %d, Parent pid: %d\n",getpid(),getppid());
                int i_id = fork();
                if(i_id == 0){
                    printf("In process I, pid: %d, Parent pid: %d\
n",getpid(),getppid());
                }
            }
        }
        else{
            int e_id = fork();
            if(e_id == 0){
                printf("In process E, pid: %d, Parent pid: %d\n",getpid(),getppid());

            }else{
                int f_id = fork();
                if(f_id == 0){
                    printf("In process F, pid: %d, Parent pid: %d\
n",getpid(),getppid());

                }
            }
        }
    }else{
        int c_id = fork();
        if(c_id == 0){ // Create C
            printf("In process C pid: %d, Parent pid: %d\n",getpid(),getppid());
            int g_id = fork();
            if(g_id == 0){ // Create G
                printf("In process G pid: %d, Parent pid: %d\n",getpid(),getppid());
            }
        }
    }
    wait(NULL);
    return 0;
}
```

Screenshots: