Practical 05

Name: Rahul Baser
Roll No.: A4-75

Aim:

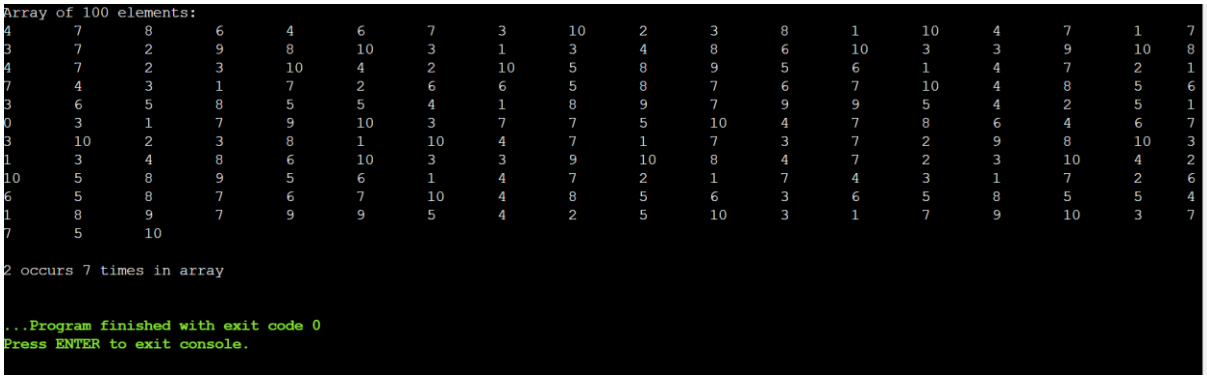
1. **Write a program to count the frequency of an element in an array of 100 elements using Using the concept of shared memory used in IPC**
2. **Using the concept of pipes used in IPC. Use pipes and write a C program in Linux to generate Fibonacci series in child process and pass numbers to parent process using pipe. Parent process should separate the odd and even numbers and print them.**

Code:

```
#include<stdio.h>
#include<sys/shm.h>
#include<sys/stat.h>
#include<sys/wait.h>
#include<sys/types.h>
#include<stdlib.h>
#include<unistd.h>
int main(int argc, char const *argv[])
{
int numArr[100];
int countOf = 2, status;
int segment_id;
int *shared_memory;
int lowerBound = 1;
int upperBound = 10;
const int size = 4096;
segment_id = shmget(IPC_PRIVATE, size, S_IRUSR | S_IWUSR);
shared_memory = (int *)shmat(segment_id, NULL, 0);
for (int i = 0; i < 100; i++) //for randomly assigning 100
elements in
{
int num = (rand() %(upperBound-lowerBound+1))+lowerBound;
numArr[i] = num;
}
printf("Array of 100 elements:\n");

for (int i = 0; i < 100; i++) //for randomly assigning 100
elements in
{
printf("%d\t", numArr[i]);
}
int pid = fork();
if(pid==0){
//child
```

```
for (int i = 50; i < 100; i++)
{
if(numArr[i]==countOf)
*shared_memory+=1;

}
}
else{
//parent
for (int i = 0; i < 50; i++)
{
if(numArr[i]==countOf)
*shared_memory+=1;

}
waitpid(-1, &status, 0);
printf("\n\n%d occurs %d times in array\n", countOf,
*shared_memory);
}
return 0;
}
```

Output:



Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <string.h>
#include <sys/wait.h>
// child process is writer (fd1) and parent is reader(fd0).
int main()
{
    printf("Hello\n");
    int n;
    int fd[2], i;
                                                    input j = 0;
```

```
pid_t pid;
printf("Enter no. of terms in Fibbonacci series: ");
scanf("%d", &n);

if (pipe(fd) == -1)
{
    printf("Pipe failed\n");
    return 1;
}
pid = fork();
if (pid < 0)
{
    printf("fork failed\n");
    return 1;
}
// Parent Process
else if (pid > 0) // parent process condition
{
    int inbuf[n];
    int even[n];
    int odd[n];
    int j, k;
    j = 0;
    k = 0;
    close(fd[1]);
    read(fd[0], inbuf, sizeof(inbuf));
    wait(NULL);
    // printf("Printing from Parent process\n");
    for (i = 0; i < n; i++)
                                                                {
        // printf("%d\t", inbuf[i]);
        if (inbuf[i] % 2 == 0)
        {
            even[j] = inbuf[i];
            j++;
        }
        else
        {
            odd[k] = inbuf[i];
            k++;
        }
    }
    // Printing even and odd integers
    printf("Even Numbers are: ");
    for (i = 0; i < j; i++)
    {
        printf("%d\t", even[i]);
    }

    printf("Odd Numbers are: ");
    for (i = 0; i < k; i++)
    {
        printf("%d\t", odd[i]);
    }
}
```

```
    else
    {
        // child Process
        int fib[n];
        close(fd[0]);
        for (i = 0; i < n; i++)
        {
            if (i == 0 || i == 1)
            {
                fib[i] = i;
                continue;
            }
            else
                fib[i] = fib[i - 1] + fib[i - 2];
        }
        write(fd[1], fib, sizeof(fib));
        printf("Fib from child :\t");
        for (i = 0; i < n; i++)
        {
            printf("%d\t", fib[i]);
        }
    }
    return 0;
}
```

Output:

```
Hello
Enter no. of terms in Fibbonacci series: 9
Fib from child :        0       1       1       2       3       5       8       13      21      Even Numbers are: 0     2       8O
dd Numbers are: 1       1       3       5       13      21

...Program finished with exit code 0
Press ENTER to exit console.
```