LAB Assignment 5

Operating Systems (UCS-303)

Q1. Write a program to:

- Create a child process using fork().
- Print the process IDs (PID, PPID) in both parent and child.

Q2. Write a program to:

- Demonstrate creation of an orphan process using fork() and sleep().
- Demonstrate creation of a zombie process (child exits before parent calls wait ()).

Q3. Write a program that accepts two small numbers (< 50) as arguments and then sums the two in a child process. The sum should be returned by the child to the parent as its exit status and the parent should print the sum.

Q4. Predict the output of the following program.

```
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
int main() {
   int x = 10;
   pid_t pid;
   pid = fork();                          // create child process
   if (pid < 0) {                         // error in fork
      perror("fork failed");
      return 1;
   }
   if (pid == 0) {                        // Child process
      printf("Child: Initially x = %d\n", x);
      x = x + 5;                          // modify x in child
      printf("Child: Modified x = %d\n", x);
   } else {
                                          // Parent process
      wait(NULL);                         // wait for child to finish
      printf("Parent: x = %d\n", x);
   }

   return 0;
}
```

Q5: Demonstrate use of lseek() System call.

Function Prototype:

off_t lseek(int fd, off_t offset, int whence);

Parameters:

- fd → File descriptor (from open())
- offset → How far to move the pointer
- whence → Reference point for movement:
  - SEEK_SET → Beginning of file
  - SEEK_CUR → Current file position
  - SEEK_END → End of file

Create a file 'seeking' and write some content e.g. 1234567890abcdefghijklmnopqrstuvwxyz and save it. Predict the output of following code:

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
int main() {
   int f;
   char buff[10];
   f = open("seeking", O_RDWR);      // open the file "seeking" in read+write mode
   if (f < 0) {
      perror("open");
      return 1;
   }
   // read first 10 bytes and print
   read(f, buff, 10);
   write(1, buff, 10);
   // move file pointer 10 bytes ahead from current
   lseek(f, 10, SEEK_CUR);
   // read next 10 bytes and print
   read(f, buff, 10);
   write(1, buff, 10);
```

```
    close(f);
    return 0;
}
```

Q6. Predict output of following codes as well for the seeking file mentioned in Q5.

Prog1.

```c
#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<stdio.h>
#include<unistd.h>
int main()
{
        int n,f,f1;
        char buff[10];
        f=open("seeking",O_RDWR);
        f1=lseek(f,10,SEEK_SET);
        printf("Pointer is at %d position\n",f1);
        read(f,buff,10);
        write(1,buff,10);
}
```

Prog2.

```c
#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<stdio.h>
#include<unistd.h>
int main()
{
        int n,f,f1;
        char buff[20];
        f=open("seeking",O_RDWR);//1234567890abcdefghijx1x2x3x4x5
        f1=lseek(f,-11,SEEK_END);
        read(f,buff,10);
        write(1,buff,10);
}
```

Q7. Predict output of the following programs.

```
 main()
{
    int a = 10;
    if(fork()) == 0))
        a++;
    printf("%d\n",a);
}
```

```c
#include <stdio.h>
#include <unistd.h>
int main()
{
    fork();
    fork() && fork() || fork();
    fork();

    printf("forked\n");
    return 0;
}
```