LABNUMBER:-LAB-01-(2019/E/118)

DATE: -1/04/2022

PROBLEM SPECIFICATION: -Understanding System Calls and Command Line Arguments With Their Usage

01)

a)

b)What are the system calls you have used in that code and explain the purpose of those calls.

Program 01)

1)fork(): used to make a new process(child process) by duplicating the calling process(parent process)

2) execlp():replace the current process with new process image in this example uses execute the ‘whoami’ command in the child process.

3) getpid(): used to get the process ID (PID) of the current process.

4 wait(): used to wait for the child process to terminate.

5) exit():used to terminate the current process.

Program 02)

1)open():used to open a file and get a file descriptor. In this code, this function is used to open two files named file1 and file2.

2)write():used to write data to a file. In this code, this function is used to write data to file1.

3)scanf():used to read input from the user.

4)lseek():used to move the file pointer to a specified position. In this code, this function is used to move the file pointer to the beginning of file1.

5)read():used to read data from a file. In this code, this function is used to read data from file1 and store it in buf2.

6)close():used to close a file. In this code, this function is used to close file1 and file2.

(c) Distinguish process system calls from I/O system calls?

The "whoami" command is carried out using Program 1's execlp() function. This is an illustration of a process system call since it initiates a new process to carry out a certain instruction or program. Another instance of a process system call is the creation of a new process, which is done via the fork() method.

Program 2 on the other hand, entails file I/O actions including opening, writing, and reading files. These are an illustration of an I/O system call. The write() and read() functions are used to write and read data to and from files, respectively, while the open() method is used to open files for reading or writing. The read/write location in the file is also changed using the lseek() method. Overall, Program 2 manipulates files via a number of I/O system calls.

(d) There are different types of system calls. Consider the following

types, provide examples for each type and explain their purpose.

i. Process Control

fork(): Creates a new process by duplicating the existing process.

exec(): Replaces the current process image with a new process image.

wait(): Suspends the execution of the calling process until one of its child processes terminates.

exit(): Terminates the calling process.

ii. File Management

open(): Creates a new file or opens an existing file.

read(): Reads data from a file.

write(): Writes data to a file.

close(): Closes a file.

unlink(): Deletes a file.

iii. Device Management

ioctl(): Controls the operation of a device.

read(): Reads data from a device.

write(): Writes data to a device.

open(): Opens a device.

iv. Information Maintenance

getpid(): Returns the process ID of the calling process.

getppid(): Returns the process ID of the parent process of the calling process.

getuid(): Returns the user ID of the calling process.

getgid(): Returns the group ID of the calling process.

v. Communication

pipe(): Creates an inter-process communication channel.

socket(): Creates a network communication endpoint.

send(): Sends data to a process or network endpoint.

receive(): Receives data from a process or network endpoint.

helloworld01EX)

INPUT:-

IMPLEMENTATION:-

```c
#include <stdio.h>
int main(){
        printf("hellow world\n");
        return 0;
}
```

OUTPUT:-

```
pc@pc-HP-ProDesk-400-G4-MT:~/Desktop/Lab01_2019e118$ nano hello.c
pc@pc-HP-ProDesk-400-G4-MT:~/Desktop/Lab01_2019e118$ gcc -o hello hello.c
pc@pc-HP-ProDesk-400-G4-MT:~/Desktop/Lab01_2019e118$ ./hello
hellow world
```

example1)

INPUT:-

IMPLEMENTATION:-

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

void main(int argc, char *arg[])
{
        int pid;
        pid=fork();
        if(pid<0)
                        {
                                        printf("fork failed");
                                        exit(1);
                        }

        else if (pid=0)
                        {
                                        execlp("whoami", "ls", NULL);
                                        exit(0);
                        }
        else
                        {
                                        printf("\n Process id is-%d\n",getpid());
                                        wait (NULL);
                                        exit(0);
                        }
}
```

OUTPUT:-

```
pc@pc-HP-ProDesk-400-G4-MT:~/Desktop/Lab01_2019e118$ gcc -o ExCllArg01 ExCllArg
01.c
pc@pc-HP-ProDesk-400-G4-MT:~/Desktop/Lab01_2019e118$ ./ExCllArg01

 Process id is-7740

 Process id is-7741
```

example2)

INPUT:-

IMPLEMENTATION:-

```c
#include<stdio.h>
#include<unistd.h>
#include<string.h>
#include<fcntl.h>
int main()
{
                int fd[2];
                char buf1 [25]= "just a test\n";
                char buf2 [50];
                fd[0]=open("file1", O_RDWR);
                fd[1]=open("file2", O_RDWR);
                write (fd[0], buf1, strlen (buf1));
                printf("\n Enter the text now....");
                scanf("\n %s", buf1);

                printf("\n Cat filel is\n hai");
                write (fd[0], buf1, strlen (buf1));
                lseek (fd[0], SEEK_SET, 0);
                read(fd[0], buf2, sizeof(buf1));
                write(fd[1], buf2, sizeof(buf2));
                close (fd[0]);
                close (fd[1]);
                printf("\n");
                return 0;

}
```

OUTPUT:-

```
pc@pc-HP-ProDesk-400-G4-MT:~/Desktop/Lab01_2019e118$ gcc -o ExCllArg02 ExCllArg
02.c
pc@pc-HP-ProDesk-400-G4-MT:~/Desktop/Lab01_2019e118$ nano ExCllArg02.c
pc@pc-HP-ProDesk-400-G4-MT:~/Desktop/Lab01_2019e118$ ./ExCllArg02

 Enter the text now....2019e118

 Cat filel is
 hai
pc@pc-HP-ProDesk-400-G4-MT:~/Desktop/Lab01_2019e118$
```

DISCUSSION:-

is an important exercise for gaining an understanding of how system calls work and how command line arguments can be used to pass information to a program.

System calls are the fundamental interface between user-level applications and the operating system. They provide a way for applications to request services from the operating system, such as creating a new process, reading and writing files, or communicating with other processes. By understanding how system calls work, we can develop a deeper understanding of how our programs interact with the operating system and the resources that it provides.

Command line arguments, on the other hand, are a way to pass information to a program when it is run. They allow us to provide input to a program in a flexible and easy-to-use way. For example, we can pass filenames, options, or other configuration parameters to a program using command line arguments.

In this lab, we will explore how system calls and command line arguments can be used together to create powerful and flexible programs. We will learn about different types of system calls, such as process control, file management, device management, information maintenance, and communication system calls. We will also learn about different types of command line arguments, such as flags, options, and positional arguments.

By the end of this lab, we will have a better understanding of how system calls work, how to use them in our programs, and how to use command line arguments to make our programs more flexible and useful.

CONCULSION:

A system call is the interface through which the process communicates with the system call. Computers can operate in one of two ways:

user mode and kernel

The process changes from user mode to kernel mode when a system call is made. The user mode process regains control once the system call has finished running. A trap signal is sent to the kernel, which reads the system call code from the register and executes the system call. The primary categories of system calls in an operating system are those for process control, file management, device management, information maintenance, and communication. Our computer system uses several important system calls, such as wait(), fork(), exec(), kill(), and exit().