RIPHAH
INTERNATIONAL UNIVERSITY

LAB 08

Name : Muhammad Hammad
Sap id : 56765
Course : Operating System
Section: SE 5-2

Presented to :

Sir. Shehzad Ahmed

RIPHAH INTERNATIONAL UNIVERSITY

Objective of Lab 08: System Calls

To understand how user programs interact with the operating-system kernel through **system calls** in Linux.
The main goal is to learn:

- How to **create, manage, and terminate processes** using system calls.
- How a **parent** and **child** process communicate and execute concurrently.
- How to use the **`fork()`**, **`exec()`**, **`wait()`**, and **`exit()`** system calls to control program execution flow.
- How process identifiers (PID and PPID) help identify and manage running processes.

Topics Covered

Topic	Description
System Calls	Interface between user programs and the OS kernel
Process Management	Using `fork()`, `wait()`, and `exit()` to create and manage processes
Program Execution	Running external programs with the `exec()` family of system calls
Process Identification	Getting process IDs using `getpid()` and `getppid()`
Parallel Execution	Demonstrating concurrent execution of parent and child processes
Complete Workflow	Implementing full process flow: `fork()` → `exec()` → `wait()` → `exit()`

Task1:

The following program prints process, and parent Ids. Execute this program

RIPHAH
INTERNATIONAL UNIVERSITY

```
hammad56765@Ubuntu:~$ ls
Desktop         infinite    lab06     pstudent.txt  student      task1.java
Documents       infinite.c  Music     Public        student.txt  Templates
Downloads       lab04       os        RIPHAH        task1        Videos
fstudents.txt   lab05       Pictures  snap          task1.c
hammad56765@Ubuntu:~$ mkdir lab08
hammad56765@Ubuntu:~$ cd lab08
hammad56765@Ubuntu:~/lab08$ touch task1.c
hammad56765@Ubuntu:~/lab08$ cat>task1.c
#include <stdio.h>
#include <unistd.h>

int main(void) {
    pid_t pid = getpid();
    pid_t ppid = getppid();
    printf("PID = %d, PPID = %d\n", (int)pid, (int)ppid);
    return 0;
}
```

Output :

```
hammad56765@Ubuntu:~/lab08$ gcc task1.c -o task1
hammad56765@Ubuntu:~/lab08$ ./task1
ProcessID = 3160, ParentprocessPID = 3053
```

java Program:

```
hammad56765@Ubuntu:~/lab08$ touch task1.java
hammad56765@Ubuntu:~/lab08$ cat>task1.java
```

task1.c × task1.java ×

```
1 package org.kodejava.lang.management;
2 import java.lang.management.ManagementFactory;
3 import java.lang.management.RuntimeMXBean;
4 public class GetProcessID {
5 public static void main(String[] args) {
6 RuntimeMXBean bean= ManagementFactory.getRuntimeMXBean();
7 String jvmName = bean.getName();
8 System.out.println("Name = " + jvmName);
9 long pid=Long.parseLong(jvmName.split("@")[0]);
10 System.out.println("PID= " + pid);
11 }
12 }
```

Output :

```
hammad56765@Ubuntu:~/lab08$ java task1.java -o task1
Name = 3322@Ubuntu
PID= 3322
```

Task # 2:

Run following program and save it with named **task2.c.**

```
hammad56765@Ubuntu:~/lab08$ touch task2.c
hammad56765@Ubuntu:~/lab08$ cat>task2.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    fork();
    int x=5;
    pid_t pid = getpid();
    printf("Value of X in PID= %d is %d\n",pid,x);
    return 0;
}
```

Output :

```
hammad56765@Ubuntu:~/lab08$ gcc task2.c -o task2
hammad56765@Ubuntu:~/lab08$ ./task2
Value of X in PID= 3354 is 5
Value of X in PID= 3355 is 5
```

RIPHAH
INTERNATIONAL UNIVERSITY

Task # 3:

Run following program and save it with named **task3.c**

```
hammad56765@Ubuntu:~/lab08$ touch task3.c
hammad56765@Ubuntu:~/lab08$ cat>task2.c
hammad56765@Ubuntu:~/lab08$ cat>task3.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>

int main(void)
{

    fork();
    pid_t pid = getpid();
    int i;
    for (i = 1; i <= 200; i++)
    {
    printf("This line is from PID %d, value = %d\n", pid, i);
   }
}
```

Output :

```
hammad56765@Ubuntu:~/lab08$ gcc task3.c -o task3
hammad56765@Ubuntu:~/lab08$ ./task3
This line is from PID 3377, value = 1
This line is from PID 3377, value = 2
This line is from PID 3377, value = 3
This line is from PID 3377, value = 4
This line is from PID 3377, value = 5
This line is from PID 3377, value = 6
This line is from PID 3377, value = 7
This line is from PID 3377, value = 8
This line is from PID 3377, value = 9
This line is from PID 3377, value = 10
This line is from PID 3377, value = 11
This line is from PID 3377, value = 12
This line is from PID 3377, value = 13
This line is from PID 3377, value = 14
This line is from PID 3377, value = 15
This line is from PID 3377, value = 16
This line is from PID 3377, value = 17
This line is from PID 3377, value = 18
This line is from PID 3377, value = 19
This line is from PID 3377, value = 20
This line is from PID 3377, value = 21
This line is from PID 3377, value = 22
```

RIPHAH INTERNATIONAL UNIVERSITY

Task4 :

Write a code in your terminal and compile it and show the output.

```
hammad56765@Ubuntu:~/lab08$ touch task4.c
hammad56765@Ubuntu:~/lab08$ cat>task4.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main(void) {
    pid_t pid = fork();  // Step 1: Create a new process

    if (pid < 0) {  // If fork fails
        perror("fork failed");
        exit(1);
    }

    if (pid == 0) {
        // Child Process
        printf("Child: My PID is %d. I'm running exec now...\n", getpid());

        // Step 2: Replace child process with another program
        execl("/bin/ls", "ls", "-l", (char *)NULL);

        // Only runs if exec fails
        perror("exec failed");
        exit(1);
    }
```

RIPHAH
INTERNATIONAL UNIVERSITY

```
    else {
        // Parent Process
        printf("Parent: My PID is %d. Waiting for child %d...\n", getpid(), pid);

        int status;
        pid_t wpid = waitpid(pid, &status, 0);  // Step 3: Wait for child

        if (wpid == -1) {
            perror("waitpid failed");
            exit(1);
        }

        if (WIFEXITED(status)) {
            printf("Parent: Child exited normally with status %d.\n", WEXITSTATUS(status));
        } else {
            printf("Parent: Child terminated abnormally.\n");
        }

        printf("Parent: Exiting now.\n");
    }

    return 0;  // Step 4: Exit
}
```

Output :

```
hammad56765@Ubuntu:~/lab08$ gcc task4.c -o task4
hammad56765@Ubuntu:~/lab08$ ./task4
Parent: My PID is 3391. Waiting for child 3392...
Child: My PID is 3392. I’m running exec now...
total 84
-rwxrwxr-x 1 hammad56765 hammad56765 16048 Oct 19 12:17 task1
-rw-rw-r-- 1 hammad56765 hammad56765   203 Oct 19 12:16 task1.c
-rw-rw-r-- 1 hammad56765 hammad56765   419 Oct 19 12:39 task1.java
-rwxrwxr-x 1 hammad56765 hammad56765 16048 Oct 19 12:41 task2
-rw-rw-r-- 1 hammad56765 hammad56765   198 Oct 19 12:43 task2.c
-rwxrwxr-x 1 hammad56765 hammad56765 16048 Oct 19 12:44 task3
-rw-rw-r-- 1 hammad56765 hammad56765   261 Oct 19 12:43 task3.c
-rwxrwxr-x 1 hammad56765 hammad56765 16312 Oct 19 12:48 task4
-rw-rw-r-- 1 hammad56765 hammad56765  1213 Oct 19 12:47 task4.c
Parent: Child exited normally with status 0.
Parent: Exiting now.
```

RIPHAH INTERNATIONAL UNIVERSITY

Task # 5:

Try to implement following code and save your program with nam**task5.c**

```
hammad56765@Ubuntu:~/lab08$ touch task5.c
hammad56765@Ubuntu:~/lab08$ cat>task5.c
#include <stdio.h>
     int main()
     {
        int fd;
        if ( fork() != 0) // for parent
                wait ((int *) 0);
        else   // for child
        {
           execl ("/bin/mkdir", "mkdir", "newdir", (char *) NULL);
           fprintf (stderr, "exec failed!\n");
           exit (1);
         }

        exit (0);
     }
hammad56765@Ubuntu:~/lab08$ gcc task5.c -o task5
task5.c: In function ‘main’:
task5.c:5:14: warning: implicit declaration of function ‘fork’ [-Wimplicit-function-declaration]
    5 |         if ( fork() != 0) // for parent
      |              ^~~~
task5.c:6:18: warning: implicit declaration of function ‘wait’ [-Wimplicit-function-declaration]
    6 |                 wait ((int *) 0);
      |                 ^~~~
task5.c:9:12: warning: implicit declaration of function ‘execl’ [-Wimplicit-function-declaration]
    9 |            execl ("/bin/mkdir", "mkdir", "newdir", (char *) NULL);
      |            ^~~~~
```

After adding libraries:

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

int main()
{
    int fd;
    if ( fork() != 0) // for parent
        wait ((int *) 0);
    else   // for child
    {
        execl ("/bin/mkdir", "mkdir", "newdir", (char *) NULL);
        fprintf (stderr, "exec failed!\n");
        exit (1);
    }
    exit (0);
}
```

RIPHAH
INTERNATIONAL UNIVERSITY

Output :

```
hammad56765@Ubuntu:~/lab08$ gcc task5.c -o task5
hammad56765@Ubuntu:~/lab08$ ./task5
hammad56765@Ubuntu:~/lab08$ ./task5
mkdir: cannot create directory ‘newdir’: File exists
hammad56765@Ubuntu:~/lab08$ ^C
hammad56765@Ubuntu:~/lab08$ ls -ld newdir
drwxrwxr-x 2 hammad56765 hammad56765 4096 Oct 19 13:01 newdir
hammad56765@Ubuntu:~/lab08$ rmdir newdir
```

----------END----------