# IT– Year 2



## Lab Exercise 2

**IT2060 – Operating Systems and System Administration**

**Semester 1, 2022**

**Learning Objectives:** Students will be able to learn UNIX process management system calls

and library functions.

### Exercise 1

Write a C program to print the process ID of the process and it’s parent process ID.

Code:

**#include <stdio.h>**

**int main()**

**{**

**printf("Process ID: %d\n",getpid());**

**printf("Parent Process ID: %d\n",getppid());**

**return 0;**

**}**

Text

Description automatically generated

### fork ( ) System call

**Exercise 2**

#include <stdio.h> main()

{

printf("I am Parent\n"); fork();

printf("Hello World...!\n");

}

// In fork, after the parent process, child process will be duplicated.

Graphical user interface, text, application, email

Description automatically generated

### Exercise 3

#include <stdio.h> main()

{

int ret;

printf("I am Parent\n"); ret = fork();

printf("Return Value: %d\n", ret);

}

**//If the fork() is returning minus values, fork is failed**

**// If the fork is returning zero values, it is returning child values**

**// If the fork is returning plus values, it is returning parent values**

# IT– Year 2

## Lab Exercise 2

**IT2060 – Operating Systems and System Administration**

**Semester 1, 2018**

### Exercise 4

#include <stdio.h> main()

{

int ret;

printf("Hello World\n"); ret = fork();

if(ret == 0)

printf("I am Child and Return Value=%d\n", ret);

else

printf("I am Parent and Return Value=%d\n", ret);

}

//This clearly validates that if the return value decrements to zero child class appears.

### getpid ( ) and getppid ( ) system calls

**Exercise 5**

#include <stdio.h> main()

{

int ret;

printf("Hello World\n"); ret = fork();

if(ret == 0){

printf("I am Child and Return Value=%d\n", ret); printf("Child PID: %d\n",

getpid());



}

else{

}

sleep(20);

}

//sleep functon ensures that the terminal sleeps for argumenent

printf("Child's Parent PID: %d\n", getppid());

printf("I am Parent and Return Value=%d\n", ret); printf("Parent PID: %d\n", getpid());

# IT– Year 2



## Lab Exercise 2

**IT2060 – Operating Systems and System Administration**

**Semester 1, 2018**

### execl ( ) system call

**Exercise 6**

#include <stdio.h> main()

{

printf(“Here comes the date. \n”);

execl(“/bin/date”, “date”, 0); /\*0 means end-of-arguments \*/ printf(“That was the date. \n”);

}

### Exercise 7

#include <stdio.h>

main()

{

printf(“Here comes the date. \n”); fork();

execl(“/bin/date”, “date”, 0);

printf(“That was the date. \n”);

}

Why did you get date two times? and Why didn’t you get first print statement two times?

Because fork creates a duplicate of the parent function and execute the child function

### Exercise 8

**system ( ) library function**

#include <stdio.h>

main()

{

printf(“Here comes the date. \n”); system(“date”);

printf(“That was the date”);

}

# IT– Year 2

## Lab Exercise 2

**IT2060 – Operating Systems and System Administration**

**Semester 1, 2018**

### CPU Time Slicing

**Exercise 9**

#include <stdio.h> main()

{

int i=0, j=0, pid; pid=fork();

if (pid == 0)

{

for (i=0; i<500000; i++)

printf(“Child: %d\n”,i++);

}

else

{

for (j=0; j<500000; j++)

printf(“Parent: %d\n”, j++);

}

}

### Zombi Process Exercise 10

#include <stdio.h> main()

{

int id;

if ((id = fork())== 0)

{



}

else

{

}

}

printf(“I am child process \n”);

while(1) sleep(100);

# IT– Year 2

## Lab Exercise 2

**IT2060 – Operating Systems and System Administration**

**Semester 1, 2018**

### Orphan Process Exercise 11

#include <stdio.h> main()

{

int id;

if ((id = fork())== 0)

{



}

else

{

}

}

printf(“I am child process \n”); sleep(10);

printf(“I am parent process \n”);