

**NATIONAL TEXTILE**

**UNIVERSITY**

DEPARTMENT OF COMPUTER SCIENCE

**SUBMITTED BY:**

Eman Faisal 23-NTU-CS-1149

**SECTION SE: 5th (A)**

**Operating System-LAB3**

**SUBMITTED TO:**

Sir Nasir Mahmood

**SUBMISSION DATE:** 10/3/25

**TASK-1**

CODE:

#include <stdio.h>

#include <unistd.h>

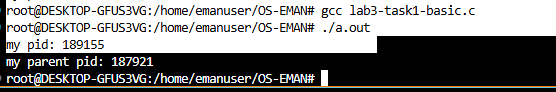
int main(){

    printf("my pid: %d\n",getpid());

    printf("my pid: %d\n",getppid());

    return 0;

}

**OUTPUT:**

**TASK-2**

CODE:  
#include <stdio.h>

#include <unistd.h>

int main() {

pid\_t pid = fork();

if (pid == 0) {

// This block runs in the child process

printf("Child: PID=%d, Parent=%d\n", getpid(), getppid());

}

else {

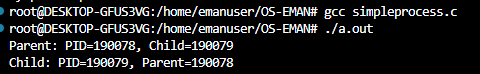
// This block runs in the parent process

printf("Parent: PID=%d, Child=%d\n", getpid(), pid);

}

return 0;

}

OUTPUT:

**TASK-3**

CODE:  
#include <stdio.h>

#include <unistd.h>

int main() {

pid\_t pid = fork();

if (pid == 0) {

execlp("ls", "ls", "-l", NULL);

printf("This will not print if exec succeeds.\n");

} else {

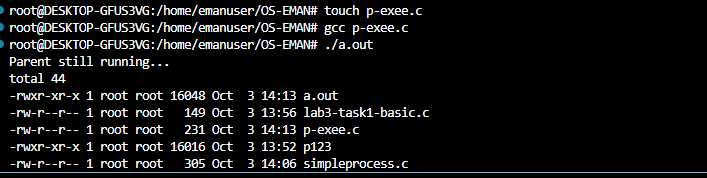
printf("Parent still running...\n");

}

return 0;

}

OUTPUT:



**Difference between wait and no wait:**

**TASK-4**

CODE:

#include <stdio.h>

#include <unistd.h>

#include <sys/wait.h>

int main() {

pid\_t pid = fork();

if (pid == 0) {

execlp("ls", "ls", "-l", NULL);

printf("This will not print if exec succeeds.\n");

} else {

waitpid(pid, NULL, 0); // Wait for the child process to finish

printf("Parent still running...\n");

}

return 0;

}

OUTPUT:  
