Practical 4

Natya Vidhan Biswas

B.Sc. (H) C.S. 25771

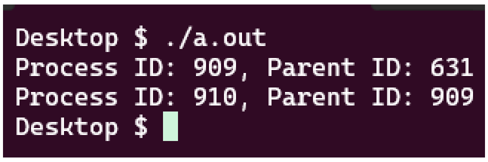
Same program, Same code

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

int main() {
    pid_t pid;
    pid = fork();

    if (pid < 0) {
        perror("Fork failed");
        return 1;
    }
    printf("Process ID: %d, Parent ID: %d\n", getpid(), getppid());

    return 0;
}
```



```
Desktop $ ./a.out
Process ID: 909, Parent ID: 631
Process ID: 910, Parent ID: 909
Desktop $
```

Same Program, Different Code

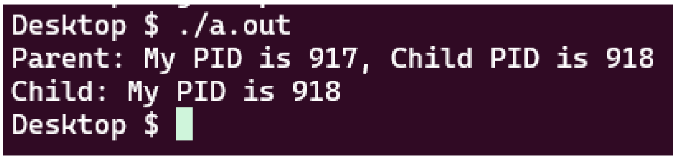
```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main() {
    pid_t pid;
    pid = fork();

    if (pid < 0) {
        perror("Fork failed");
        return 1;
    }

    if (pid == 0) {
        printf("Child: My PID is %d\n", getpid());
    } else {
        printf("Parent: My PID is %d, Child PID is %d\n", getpid(), pid);
    }

    return 0;
}
```



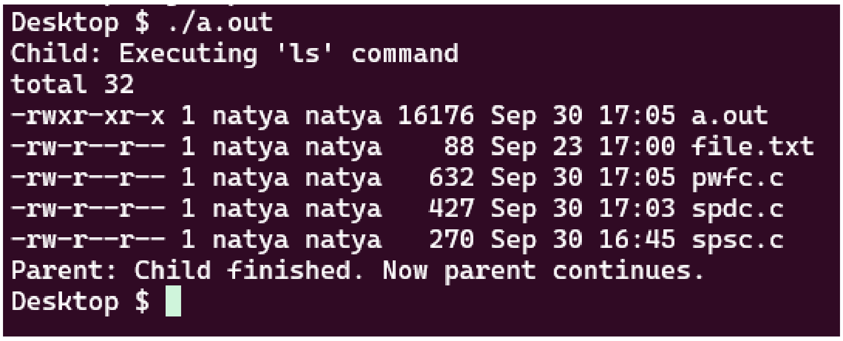
```
Desktop $ ./a.out
Parent: My PID is 917, Child PID is 918
Child: My PID is 918
Desktop $
```

Parent Waits for Child

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

int main() {
    pid_t pid;
    pid = fork();

    if (pid == 0) {
        printf("Child: Executing 'ls' command\n");
        execlp("ls", "ls", "-l", NULL);
    } else {
        int status;
        wait(&status);
        printf("Parent: Child finished. Now parent continues.\n");
    }
    return 0;
}
```



```
Desktop $ ./a.out
Child: Executing 'ls' command
total 32
-rwxr-xr-x 1 natya natya 16176 Sep 30 17:05 a.out
-rw-r--r-- 1 natya natya    88 Sep 23 17:00 file.txt
-rw-r--r-- 1 natya natya   632 Sep 30 17:05 pwfc.c
-rw-r--r-- 1 natya natya   427 Sep 30 17:03 spdc.c
-rw-r--r-- 1 natya natya   270 Sep 30 16:45 spsc.c
Parent: Child finished. Now parent continues.
Desktop $
```