

- [CPSC 275: Introduction to Computer Systems](#)

[CPSC 275: Introduction to Computer Systems](#)

Fall 2025

- [Syllabus](#)
- [Schedule](#)
- [Resources](#)
- [Upload](#)
- [Solution](#)

Homework 33

NOTE: You are not required to hand in the following exercises, but you are strongly encouraged to complete them to strengthen your understanding of the concepts covered in class.

Exercise 1: Creating new processes

Enter the following code in `proc1.c`, compile and run the program:

```
/*
 * proc1.c - creates a new process.
 */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(void) {
    printf("Hi.\n");
    fork();
    printf("Bye.\n");
    exit(0);
}
```

What is the output from the program? Explain the result. Now, modify the program by inserting

```
sleep(10);
```

before and after the `fork` system call. Compile and run the program in the background:

```
$ ./proc1 &
```

immediately followed by a command to view a list of current processes:

```
$ ps
```

What is the PID of the parent process? Issue the `ps` command after 10 seconds. How many processes created by the program do you see? What is the PID of the child process?

Exercise 2: Executing different parts of the same program

Enter the following code in `proc2.c`:

```

/*
 * proc2.c - parent and child processes execute different parts of the same program.
 */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main()
{
    int pid;
    char *message;
    int i;

    printf("fork program starting\n");
    pid = fork();
    switch(pid)
    {
        case -1:
            exit(1);
        case 0:
            message = "This is the child";
            break;
        default:
            message = "This is the parent";
            break;
    }

    for (i = 0; i < N; i++)
        puts(message);

    exit(0);
}

```

Compile it with:

```
$ gcc -o proc2 proc2.c -D N=nmsg
```

where *nmsg* is the number messages each process will print. Run the program several times for different values of *nmsg*. What is the output from the program? Explain the results.

Exercise 3: Running different programs in separate processes

Enter the following code in `proc3.c`:

```

/*
 * proc3.c - parent and child processes run different programs in its own process space
 */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <wait.h>

int main(void)
{
    int pid;

    printf("fork program starting\n");
    pid = fork();
    switch(pid)
    {
        case -1:
            exit(1);

```

```

case 0:
    execl("./child", "child", (char *) 0);
    exit(1); /* should never get here */

default:
    printf("Process[%d]: parent in execution ...\n", getpid());
    sleep(5);
    if (wait(NULL) > 0) /* waiting for child */
        printf("Process[%d]: parent terminating child ...\n", getpid());
    printf("Process[%d]: parent terminating ...\n", getpid());
    exit(0);
}
}

```

and in child.c:

```

/*
 * child.c - This child program replaces the parent's program.
 */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(void)
{
    printf("Process[%d]: child in execution ...\n", getpid());
    sleep(5);
    printf("Process[%d]: child in execution ...\n", getpid());
    exit(0);
}

```

Compile both programs separately and run proc3 in the background as follows:

```
$ ./proc3 &
```

immediately followed by a command to view a list of current processes:

```
$ ps
```

What is the PID of the parent process? What is the PID of the child process? Do you see two distinct programs running concurrently?

- **Welcome: Sean**

- [LogOut](#)

