Name - Bhuvnesh Verma

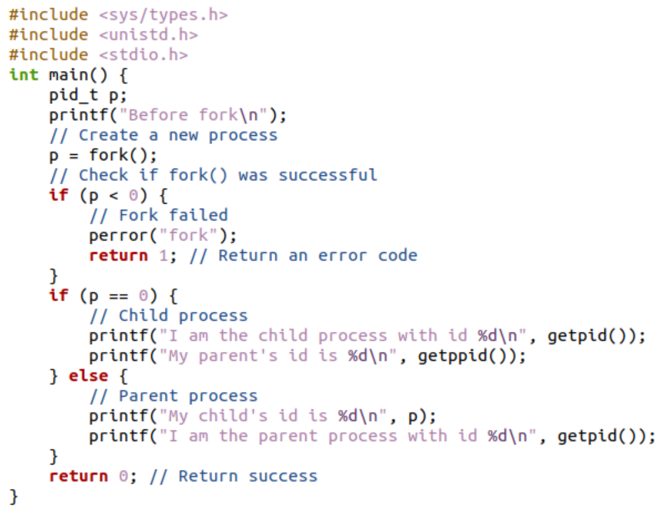
Roll No . - 28

Batch - A2

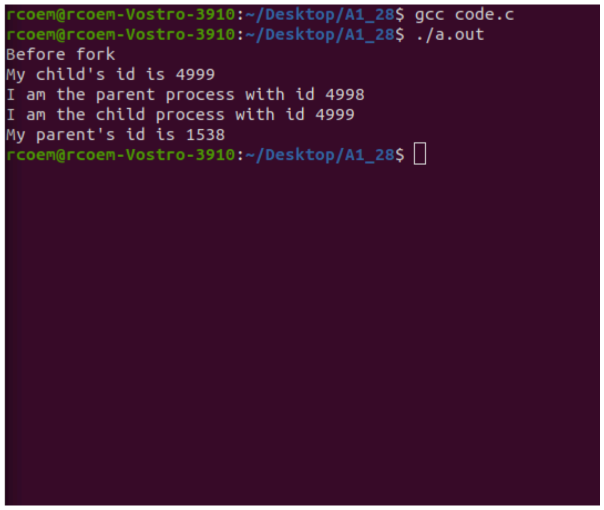
Date - 20/08/24

Lab no . - 3

Fork system call

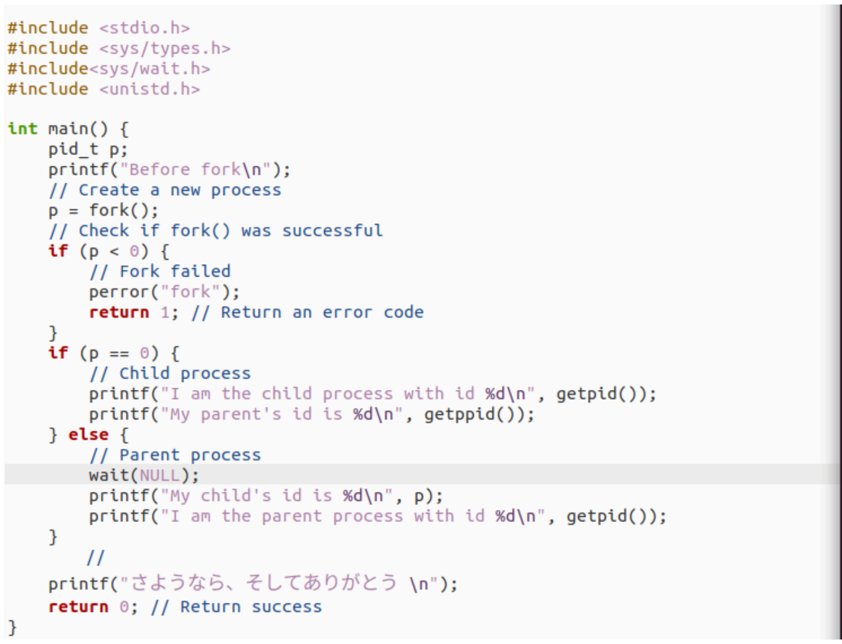


```c
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
int main() {
    pid_t p;
    printf("Before fork\n");
    // Create a new process
    p = fork();
    // Check if fork() was successful
    if (p < 0) {
        // Fork failed
        perror("fork");
        return 1; // Return an error code
    }
    if (p == 0) {
        // Child process
        printf("I am the child process with id %d\n", getpid());
        printf("My parent's id is %d\n", getppid());
    } else {
        // Parent process
        printf("My child's id is %d\n", p);
        printf("I am the parent process with id %d\n", getpid());
    }
    return 0; // Return success
}
```

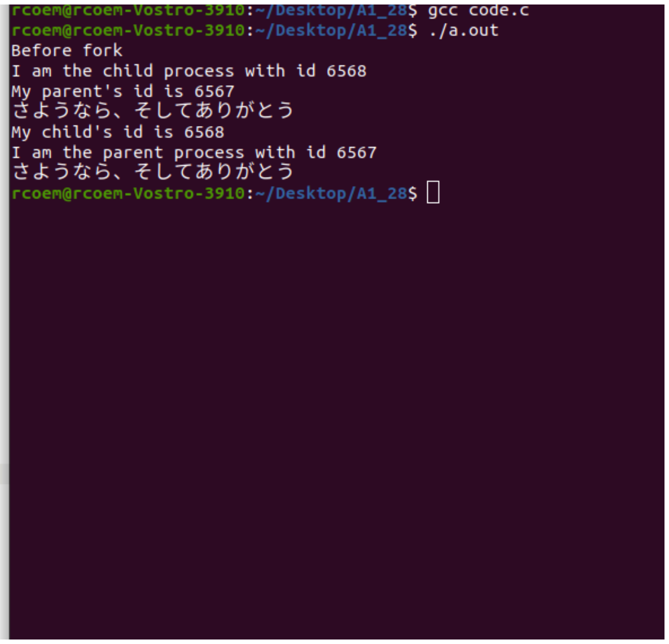


```
rcoem@rcoem-Vostro-3910:~/Desktop/A1_28$ gcc code.c
rcoem@rcoem-Vostro-3910:~/Desktop/A1_28$ ./a.out
Before fork
My child's id is 4999
I am the parent process with id 4998
I am the child process with id 4999
My parent's id is 1538
rcoem@rcoem-Vostro-3910:~/Desktop/A1_28$ 
```

Wait

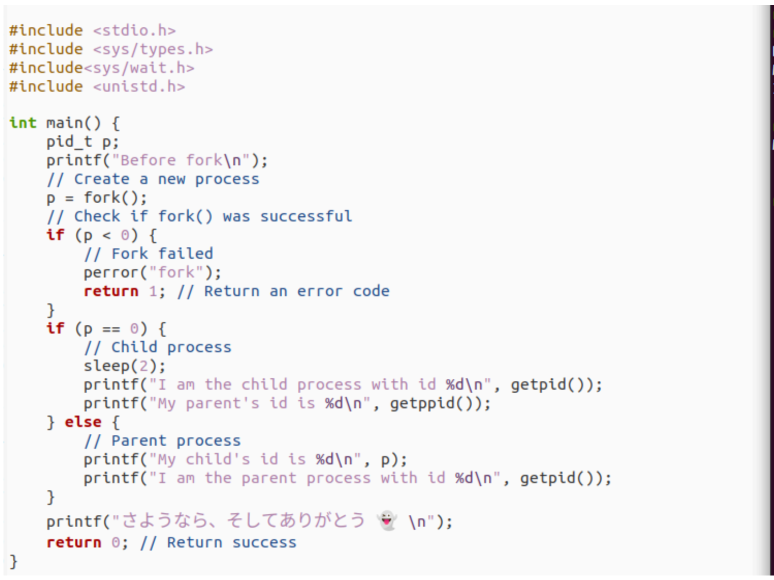


```c
#include <stdio.h>
#include <sys/types.h>
#include<sys/wait.h>
#include <unistd.h>

int main() {
    pid_t p;
    printf("Before fork\n");
    // Create a new process
    p = fork();
    // Check if fork() was successful
    if (p < 0) {
        // Fork failed
        perror("fork");
        return 1; // Return an error code
    }
    if (p == 0) {
        // Child process
        printf("I am the child process with id %d\n", getpid());
        printf("My parent's id is %d\n", getppid());
    } else {
        // Parent process
        wait(NULL);
        printf("My child's id is %d\n", p);
        printf("I am the parent process with id %d\n", getpid());
    }
        //
    printf("さようなら、そしてありがとう \n");
    return 0; // Return success
}
```

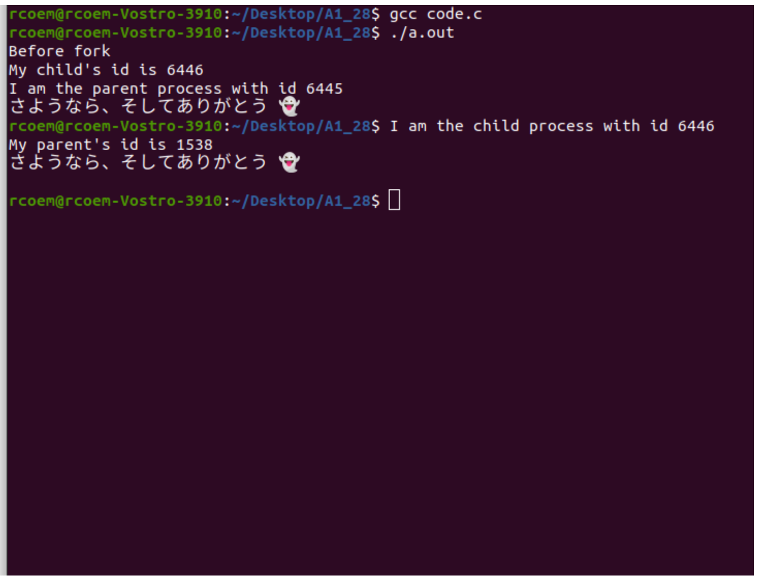


```
rcoem@rcoem-Vostro-3910:~/Desktop/A1_28$ gcc code.c
rcoem@rcoem-Vostro-3910:~/Desktop/A1_28$ ./a.out
Before fork
I am the child process with id 6568
My parent's id is 6567
さようなら、そしてありがとう
My child's id is 6568
I am the parent process with id 6567
さようなら、そしてありがとう
rcoem@rcoem-Vostro-3910:~/Desktop/A1_28$ 
```

Orphan

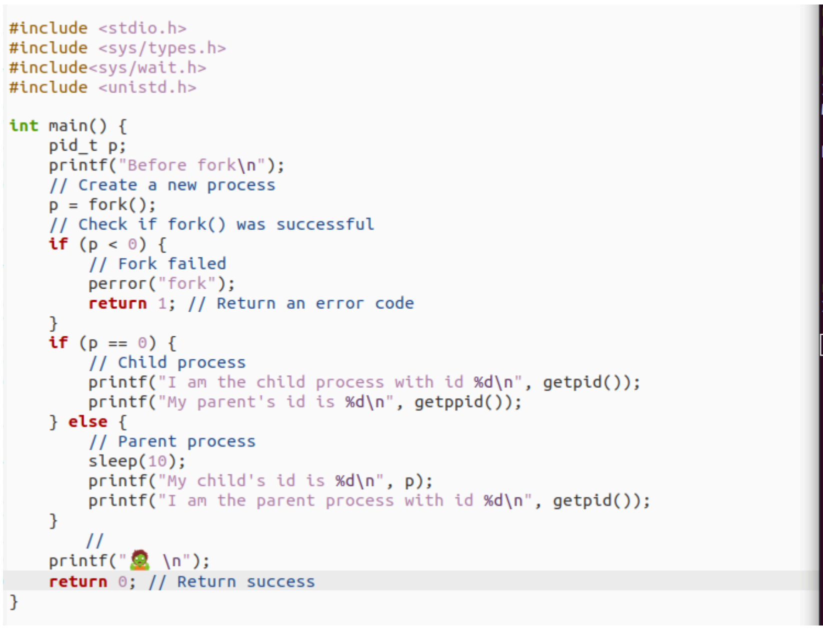


```c
#include <stdio.h>
#include <sys/types.h>
#include<sys/wait.h>
#include <unistd.h>

int main() {
    pid_t p;
    printf("Before fork\n");
    // Create a new process
    p = fork();
    // Check if fork() was successful
    if (p < 0) {
        // Fork failed
        perror("fork");
        return 1; // Return an error code
    }
    if (p == 0) {
        // Child process
        sleep(2);
        printf("I am the child process with id %d\n", getpid());
        printf("My parent's id is %d\n", getppid());
    } else {
        // Parent process
        printf("My child's id is %d\n", p);
        printf("I am the parent process with id %d\n", getpid());
    }
    printf("さようなら、そしてありがとう 👻 \n");
    return 0; // Return success
}
```

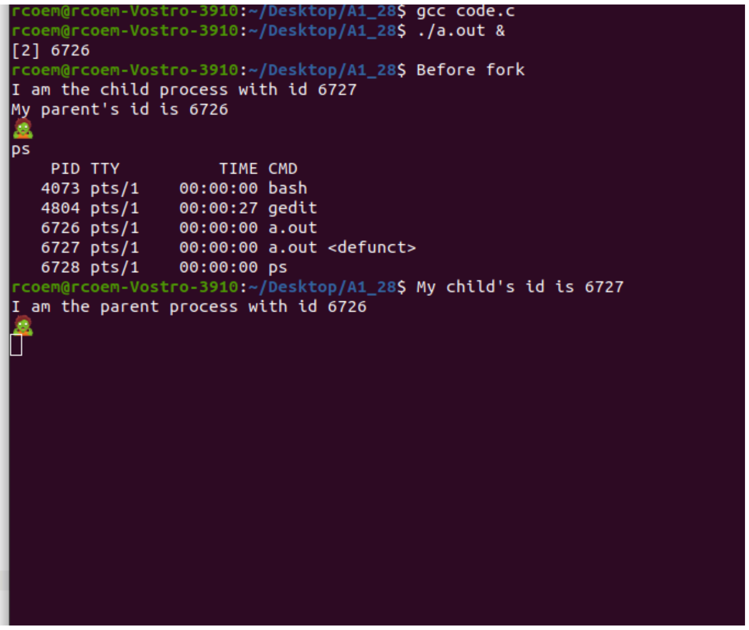


```
rcoem@rcoem-Vostro-3910:~/Desktop/A1_28$ gcc code.c
rcoem@rcoem-Vostro-3910:~/Desktop/A1_28$ ./a.out
Before fork
My child's id is 6446
I am the parent process with id 6445
さようなら、そしてありがとう 👻
rcoem@rcoem-Vostro-3910:~/Desktop/A1_28$ I am the child process with id 6446
My parent's id is 1538
さようなら、そしてありがとう 👻

rcoem@rcoem-Vostro-3910:~/Desktop/A1_28$ 
```

Zombie

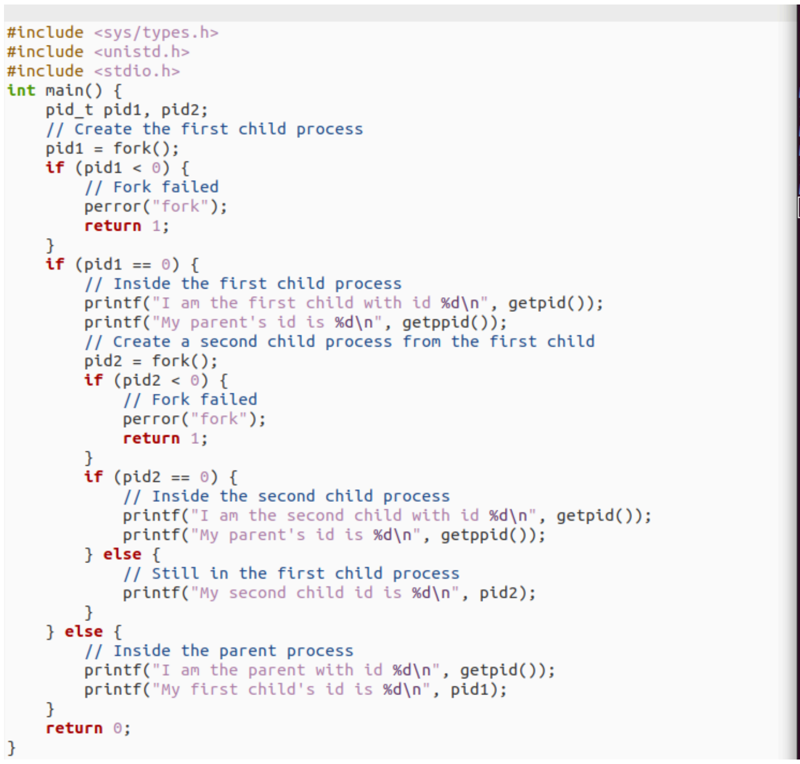


```c
#include <stdio.h>
#include <sys/types.h>
#include<sys/wait.h>
#include <unistd.h>

int main() {
    pid_t p;
    printf("Before fork\n");
    // Create a new process
    p = fork();
    // Check if fork() was successful
    if (p < 0) {
        // Fork failed
        perror("fork");
        return 1; // Return an error code
    }
    if (p == 0) {
        // Child process
        printf("I am the child process with id %d\n", getpid());
        printf("My parent's id is %d\n", getppid());
    } else {
        // Parent process
        sleep(10);
        printf("My child's id is %d\n", p);
        printf("I am the parent process with id %d\n", getpid());
    }
        //
    printf("🧟 \n");
    return 0; // Return success
}
```

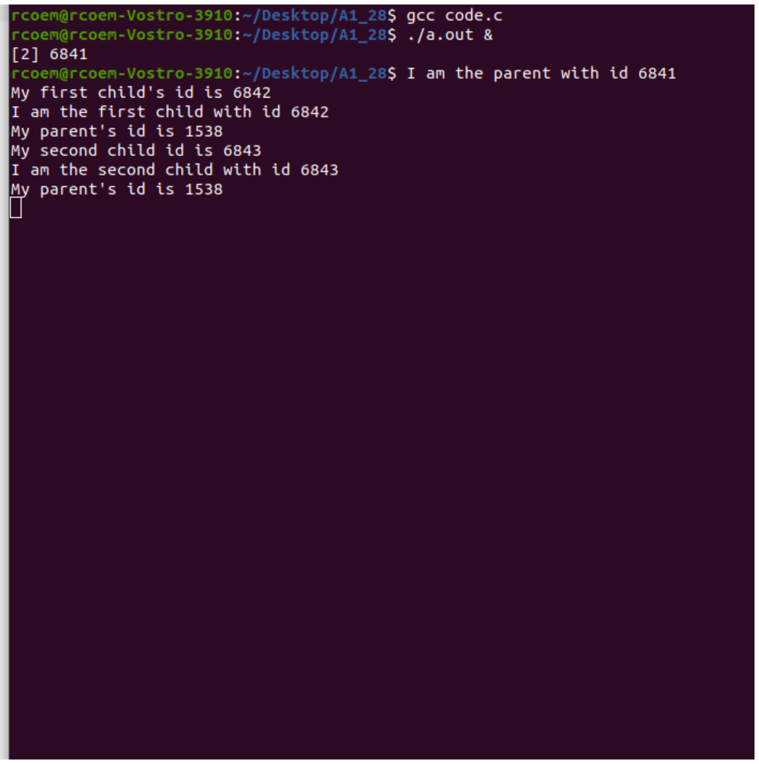


```
rcoem@rcoem-Vostro-3910:~/Desktop/A1_28$ gcc code.c
rcoem@rcoem-Vostro-3910:~/Desktop/A1_28$ ./a.out &
[2] 6726
rcoem@rcoem-Vostro-3910:~/Desktop/A1_28$ Before fork
I am the child process with id 6727
My parent's id is 6726
🧟
ps
    PID TTY          TIME CMD
   4073 pts/1    00:00:00 bash
   4804 pts/1    00:00:27 gedit
   6726 pts/1    00:00:00 a.out
   6727 pts/1    00:00:00 a.out <defunct>
   6728 pts/1    00:00:00 ps
rcoem@rcoem-Vostro-3910:~/Desktop/A1_28$ My child's id is 6727
I am the parent process with id 6726
🧟
```

Parents have children who then have their own children.



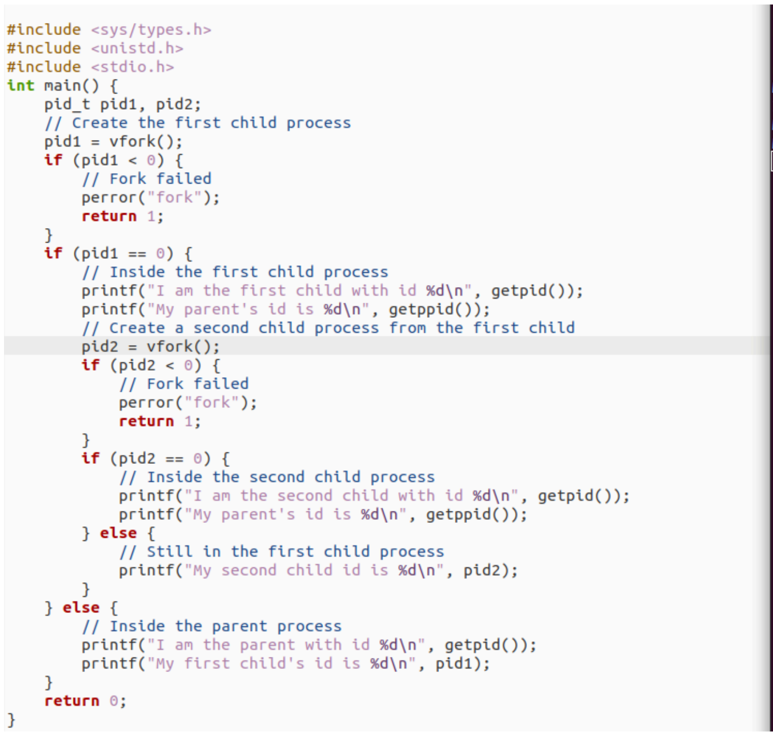
```c
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
int main() {
    pid_t pid1, pid2;
    // Create the first child process
    pid1 = fork();
    if (pid1 < 0) {
        // Fork failed
        perror("fork");
        return 1;
    }
    if (pid1 == 0) {
        // Inside the first child process
        printf("I am the first child with id %d\n", getpid());
        printf("My parent's id is %d\n", getppid());
        // Create a second child process from the first child
        pid2 = fork();
        if (pid2 < 0) {
            // Fork failed
            perror("fork");
            return 1;
        }
        if (pid2 == 0) {
            // Inside the second child process
            printf("I am the second child with id %d\n", getpid());
            printf("My parent's id is %d\n", getppid());
        } else {
            // Still in the first child process
            printf("My second child id is %d\n", pid2);
        }
    } else {
        // Inside the parent process
        printf("I am the parent with id %d\n", getpid());
        printf("My first child's id is %d\n", pid1);
    }
    return 0;
}
```



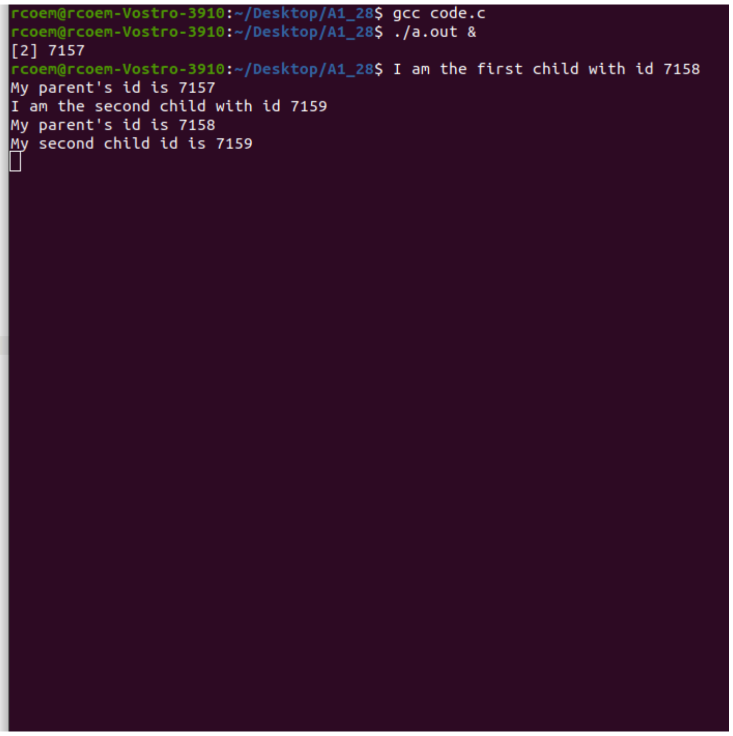
```
rcoem@rcoem-Vostro-3910:~/Desktop/A1_28$ gcc code.c
rcoem@rcoem-Vostro-3910:~/Desktop/A1_28$ ./a.out &
[2] 6841
rcoem@rcoem-Vostro-3910:~/Desktop/A1_28$ I am the parent with id 6841
My first child's id is 6842
I am the first child with id 6842
My parent's id is 1538
My second child id is 6843
I am the second child with id 6843
My parent's id is 1538
```

---

vfork



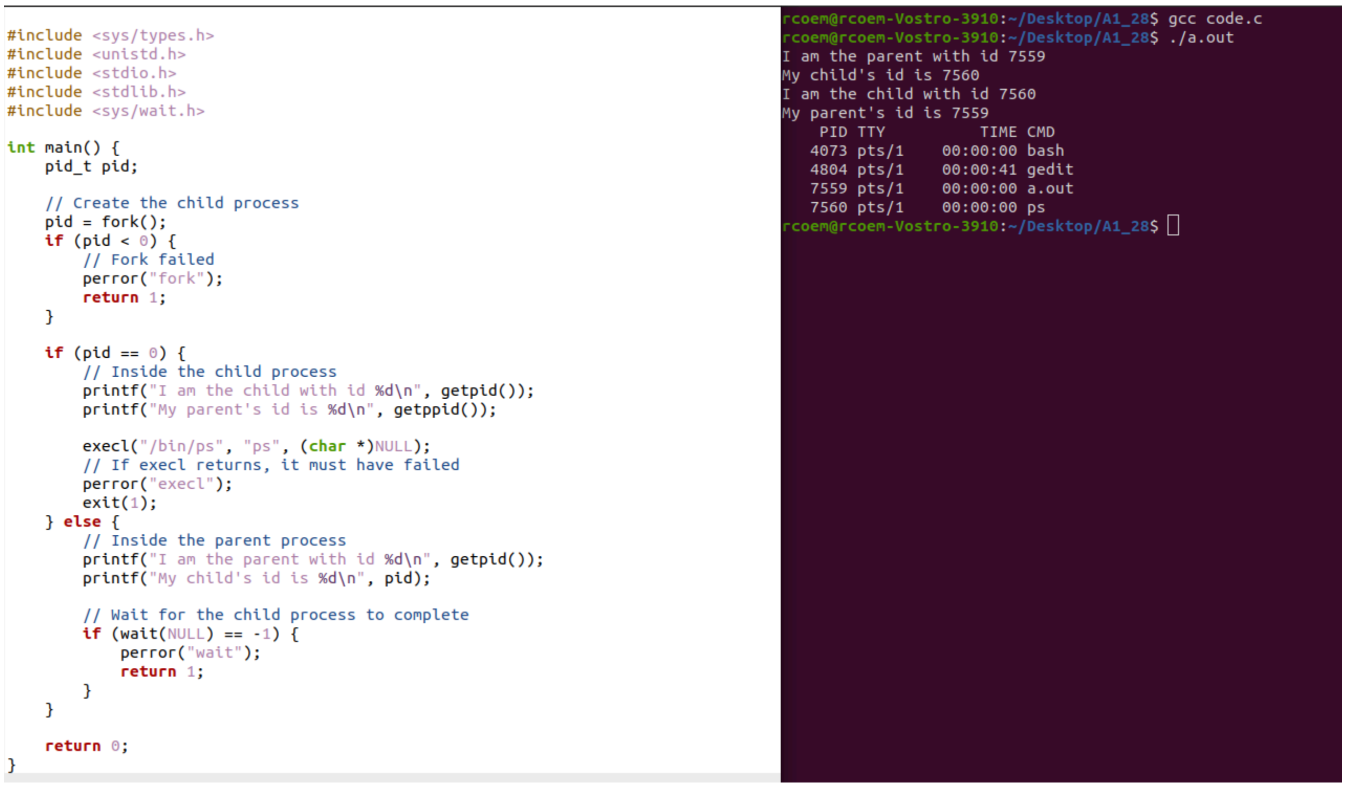
```c
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
int main() {
    pid_t pid1, pid2;
    // Create the first child process
    pid1 = vfork();
    if (pid1 < 0) {
        // Fork failed
        perror("fork");
        return 1;
    }
    if (pid1 == 0) {
        // Inside the first child process
        printf("I am the first child with id %d\n", getpid());
        printf("My parent's id is %d\n", getppid());
        // Create a second child process from the first child
        pid2 = vfork();
        if (pid2 < 0) {
            // Fork failed
            perror("fork");
            return 1;
        }
        if (pid2 == 0) {
            // Inside the second child process
            printf("I am the second child with id %d\n", getpid());
            printf("My parent's id is %d\n", getppid());
        } else {
            // Still in the first child process
            printf("My second child id is %d\n", pid2);
        }
    } else {
        // Inside the parent process
        printf("I am the parent with id %d\n", getpid());
        printf("My first child's id is %d\n", pid1);
    }
    return 0;
}
```



```
rcoem@rcoem-Vostro-3910:~/Desktop/A1_28$ gcc code.c
rcoem@rcoem-Vostro-3910:~/Desktop/A1_28$ ./a.out &
[2] 7157
rcoem@rcoem-Vostro-3910:~/Desktop/A1_28$ I am the first child with id 7158
My parent's id is 7157
I am the second child with id 7159
My parent's id is 7158
My second child id is 7159
```

---

exec



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

int main() {
    pid_t pid;

    // Create the child process
    pid = fork();
    if (pid < 0) {
        // Fork failed
        perror("fork");
        return 1;
    }

    if (pid == 0) {
        // Inside the child process
        printf("I am the child with id %d\n", getpid());
        printf("My parent's id is %d\n", getppid());

        execl("/bin/ps", "ps", (char *)NULL);
        // If execl returns, it must have failed
        perror("execl");
        exit(1);
    } else {
        // Inside the parent process
        printf("I am the parent with id %d\n", getpid());
        printf("My child's id is %d\n", pid);

        // Wait for the child process to complete
        if (wait(NULL) == -1) {
            perror("wait");
            return 1;
        }
    }

    return 0;
}
```

```
rcoem@rcoem-Vostro-3910:~/Desktop/A1_28$ gcc code.c
rcoem@rcoem-Vostro-3910:~/Desktop/A1_28$ ./a.out
I am the parent with id 7559
My child's id is 7560
I am the child with id 7560
My parent's id is 7559
    PID TTY          TIME CMD
   4073 pts/1    00:00:00 bash
   4804 pts/1    00:00:41 gedit
   7559 pts/1    00:00:00 a.out
   7560 pts/1    00:00:00 ps
rcoem@rcoem-Vostro-3910:~/Desktop/A1_28$ 
```
