

Ajay Pratap Singh Kulharia

9922103164

F6

OSSP LAB WEEK2

Q1)

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

int main() {
    pid_t pid = getpid();
    pid_t ppid = getppid();
    uid_t ruid = getuid();
    uid_t euid = geteuid();
    int nice_value = getpriority(PRIO_PROCESS, 0);
    char *tty = ttyname(STDIN_FILENO);
    struct passwd *pwd = getpwuid(ruid);
    struct passwd *epwd = getpwuid(euid);

    printf("Process ID (PID): %d\n", pid);
    printf("Parent Process ID (PPID): %d\n", ppid);
    printf("Nice value: %d\n", nice_value);
    printf("Terminal (TTY): %s\n", tty ? tty : "None");
    printf("Real User ID (RUID): %d, User name: %s\n", ruid, pwd ? pwd->pw_name :
"Unknown");
    printf("Effective User ID (EUID): %d, User name: %s\n", euid, epwd ? epwd->pw_name :
"Unknown");

    return 0;
}
```

```
Process ID (PID): 70
Parent Process ID (PPID): 69
Nice value: 0
Terminal (TTY): /dev/pts/3
Real User ID (RUID): 14059, User name: Unknown
Effective User ID (EUID): 14059, User name: Unknown

...Program finished with exit code 0
```

Q2)

```
#include <stdio.h>
```

```
#include <unistd.h>
```

```
#include <sys/types.h>
```

```
int main() {
```

```
    pid_t pid = fork();
```

```
    if (pid < 0) {
```

```
        perror("Fork failed");
```

```
        return 1;
```

```
    } else if (pid == 0) {
```

```
        printf("I am the child process with PID %d, my parent PID is %d\n", getpid(), getppid());
```

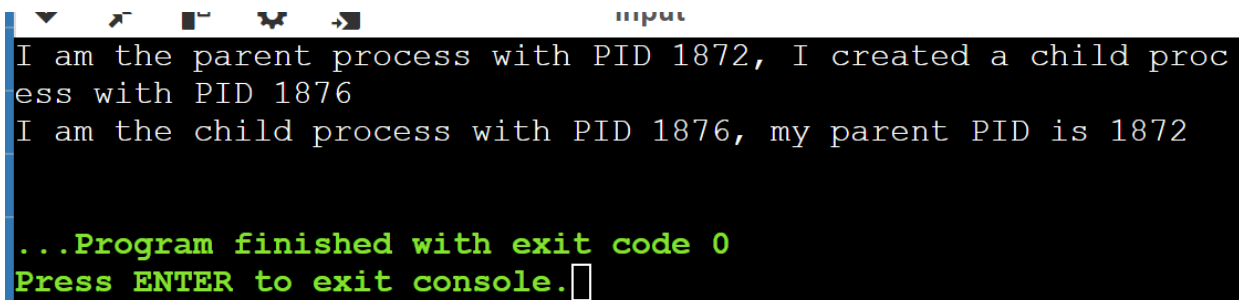
```
    } else {
```

```
        printf("I am the parent process with PID %d, I created a child process with PID %d\n",
getpid(), pid);
```

```
    }
```

```
    return 0;
```

```
}
```



A terminal window with a black background and white text. The text shows a parent process (PID 1872) creating a child process (PID 1876). The child process prints its parent's PID. The program then finishes with exit code 0. The text is as follows:

```

input
I am the parent process with PID 1872, I created a child process with PID 1876
I am the child process with PID 1876, my parent PID is 1872

...Program finished with exit code 0
Press ENTER to exit console.

```

```

Q3) #include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

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

    if (pid < 0) {
        perror("Fork failed");
        return 1;
    } else if (pid == 0) {
        printf("Child process created with PID %d\n", getpid());
        _exit(0); // Child exits
    } else {
        printf("Parent process with PID %d created a zombie process with PID %d\n", getpid(), pid);
        sleep(10); // Parent sleeps, during this time child becomes zombie
        wait(NULL); // Parent waits for child process to complete
        printf("Zombie process with PID %d reaped by parent\n", pid);
    }

    return 0;
}

```

```
Parent process with PID 2794 created a zombie process with PID 2798
Child process created with PID 2798
Zombie process with PID 2798 reaped by parent

...Program finished with exit code 0
Press ENTER to exit console.□
```

Q-4)

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

int main() {
    pid_t pid1, pid2, pid3;

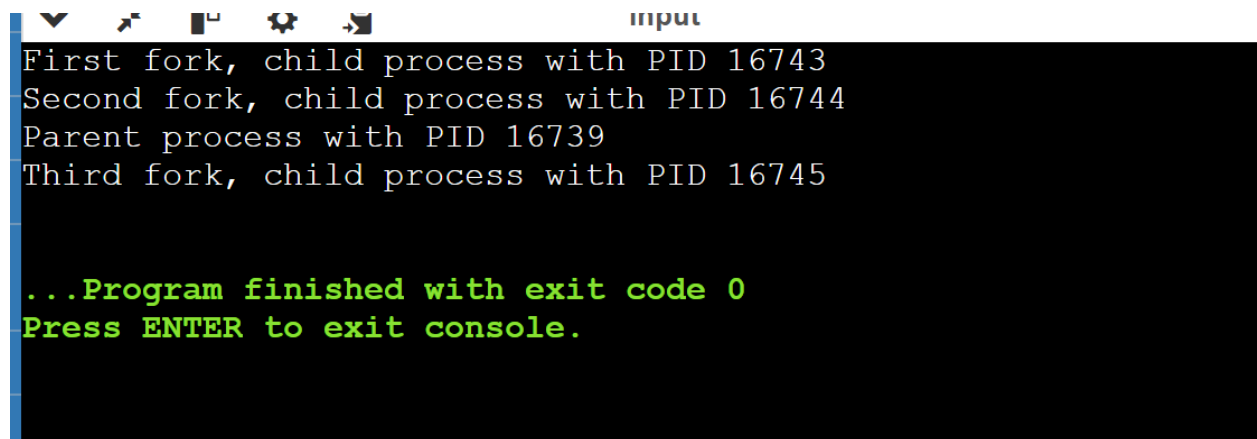
    pid1 = fork();
    if (pid1 < 0) {
        perror("Fork failed");
        return 1;
    } else if (pid1 == 0) {
        printf("First fork, child process with PID %d\n", getpid());
    } else {
        pid2 = fork();
        if (pid2 < 0) {
            perror("Fork failed");
            return 1;
        } else if (pid2 == 0) {
            printf("Second fork, child process with PID %d\n", getpid());
        } else {
            pid3 = fork();
            if (pid3 < 0) {
                perror("Fork failed");
                return 1;
            } else if (pid3 == 0) {
                printf("Third fork, child process with PID %d\n", getpid());
            } else {
```

```

        printf("Parent process with PID %d\n", getpid());
    }
}

return 0;
}

```



```

input
First fork, child process with PID 16743
Second fork, child process with PID 16744
Parent process with PID 16739
Third fork, child process with PID 16745

...Program finished with exit code 0
Press ENTER to exit console.

```

Q5)

```

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

void do_addition() {
    int result = 5 + 3;
    printf("Addition result: %d\n", result);
}

void do_subtraction() {
    int result = 5 - 3;
    printf("Subtraction result: %d\n", result);
}

void do_division() {
    int result = 5 / 3;
    printf("Division result: %d\n", result);
}

int main() {

```

```

pid_t pid1, pid2, pid3;

pid1 = fork();
if (pid1 < 0) {
    perror("Fork failed");
    return 1;
} else if (pid1 == 0) {
    do_addition();
    _exit(0);
} else {
    pid2 = fork();
    if (pid2 < 0) {
        perror("Fork failed");
        return 1;
    } else if (pid2 == 0) {
        do_subtraction();
        _exit(0);
    } else {
        pid3 = fork();
        if (pid3 < 0) {
            perror("Fork failed");
            return 1;
        } else if (pid3 == 0) {
            do_division();
            _exit(0);
        } else {
            wait(NULL);
            wait(NULL);
            wait(NULL);
            printf("All child processes have completed\n");
        }
    }
}

return 0;
}

```



input

Addition result: 8

Subtraction result: 2

Division result: 1

All child processes have completed

...Program finished with exit code 0

Press ENTER to exit console.