Федеральное государственное автономное образовательное учреждение высшего образования
"Национальный исследовательский университет "Высшая школа экономики"
Московский институт электроники и математики
им. А. Н. Тихонова НИУ ВШЭ
Департамент компьютерной инженерии

Лабораторная работа № 4
по теме:
«Создание и отслеживание процессов
в операционной системе Linux»

По дисциплине

«Операционные системы»

Группа: **БПМ 243**

Студент: **Николаев Степан Денисович**

Москва 2026

Ход работы

Собрал show_pid.c и запустил три раза. Получил PID 9022, 9024 и 9026; PPID во всех случаях — 3659. PID меняется, потому что каждый запуск создаёт новый процесс. PPID одинаковый, так как родитель — одна и та же оболочка.

```
marmos@marmos-QEMU-Virtual-Machine:~/practice4$ gcc show_pid.c -o show_pid
marmos@marmos-QEMU-Virtual-Machine:~/practice4$ ./show_pid
PID: 9022
PPID: 3659
marmos@marmos-QEMU-Virtual-Machine:~/practice4$ ./show_pid
PID: 9024
PPID: 3659
marmos@marmos-QEMU-Virtual-Machine:~/practice4$ ./show_pid
PID: 9026
PPID: 3659
marmos@marmos-QEMU-Virtual-Machine:~/practice4$
```

Рис. 1. Компиляция show_pid.c и три запуска.

В fork_process.c разделил вывод родителя и ребёнка по результату fork(). Родитель выводит свой PID и PID ребёнка, ребёнок — свой PID и PPID. Например, у ребёнка 9086 PPID равен 9085 — PID родителя. Если родитель завершится раньше ребёнка, PPID может измениться.

```
marmos@marmos-QEMU-Virtual-Machine:~/practice4$ gcc fork_process.c -o fork_process
marmos@marmos-QEMU-Virtual-Machine:~/practice4$ ./fork_process
Я родительский процесс
PID: 9085, PID ребенка: 9086
Я дочерний процесс
PID: 9086, PPID: 9085
marmos@marmos-QEMU-Virtual-Machine:~/practice4$ ./fork_process
Я родительский процесс
PID: 9095, PID ребенка: 9096
Я дочерний процесс
PID: 9096, PPID: 9095
```

Рис. 2. Родительский и дочерний процессы.

В exec_process.c дочерний процесс запускает /bin/date через execl(). После execl() предусмотрен perror() на случай ошибки запуска.

```
marmos@marmos-QEMU-Virtual-Machine:~/practice4$ gcc exec_process.c -o exec_process
marmos@marmos-QEMU-Virtual-Machine:~/practice4$ ./exec_process
Родительский процесс
PID: 9173, PID ребенка: 9174
Дочерний процесс запускает программу date
marmos@marmos-QEMU-Virtual-Machine:~/practice4$ Sun Sep 20 02:37:03 AM MSK 2026
./exec_process                                  ./exec_process
Родительский процесс
PID: 9177, PID ребенка: 9178
Дочерний процесс запускает программу date
marmos@marmos-QEMU-Virtual-Machine:~/practice4$ Sun Sep 20 02:37:09 AM MSK 2026
```

Рис. 3. Запуск date из дочернего процесса.

Дата напечаталась уже после приглашения терминала: родитель завершился раньше ребёнка, и оболочка вернулась к вводу команд. Порядок вывода этих процессов не задан.

Вывод. Проверил получение PID и PPID, создание дочернего процесса и запуск другой программы через execl().

Приложение show_pid.c:

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

int main() {
    // Получаем идентификаторы процесса и его родителя
    pid_t pid = getpid();
    pid_t ppid = getppid();

    printf("PID: %d\n", pid);
    printf("PPID: %d\n", ppid);

    return 0;
}
```

Приложение fork_process.c:

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

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

    if (pid == -1) {
        printf("Ошибка создания процесса\n");
        return 1;
    }

    if (pid == 0) {
        // Дочерний процесс
        printf("Я дочерний процесс\n");
        printf("PID: %d, PPID: %d\n", getpid(), getppid());
    } else {
        // Родительский процесс
        printf("Я родительский процесс\n");
        printf("PID: %d, PID ребенка: %d\n", getpid(), pid);
    }

    return 0;
}
```

Приложение exec_process.c:

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

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

    if (pid == -1) {
        printf("Ошибка создания процесса\n");
        return 1;
    }

    if (pid == 0) {
        // Дочерний процесс
        printf("Дочерний процесс запускает программу date\n");

        execl("/bin/date", "date", NULL);

        // Эта строка выполнится только при ошибке exec
        perror("Ошибка запуска программы");
        return 1;
    } else {
        // Родительский процесс
        printf("Родительский процесс\n");
        printf("PID: %d, PID ребенка: %d\n", getpid(), pid);
    }

    return 0;
}
```