



Министерство науки и высшего образования Российской
Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Московский государственный технический университет имени
Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4
По курсу: "Операционные системы"

Студент _____ Сукочева Алис
Группа _____ ИУ7-53Б
Название предприятия _____ МГТУ им. Н. Э. Баумана, каф. ИУ7
Тема _____ Процессы. Системные вызовы `fork()` и `exec()`.

Студент:	_____	Сукочева А.
	подпись, дата	Фамилия, И.О.
Преподаватель:	_____	Рязанова Н.Ю.
	подпись, дата	Фамилия, И. О.

Листинг 1 — Программа 1.

```

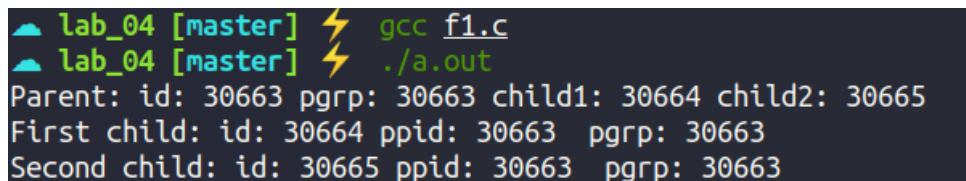
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <stdlib.h>
4
5 #define OK 0
6 #define ERROR 1
7 #define SLEEP_TIME 2
8 #define ERROR_FORK -1
9
10 int main()
11 {
12     int childpid_1, childpid_2;
13
14     // Первый процесс.
15     // Создается дочерний процесс
16     if ((childpid_1 = fork()) == ERROR_FORK)
17     {
18         // Если при порождении процесса произошла ошибка.
19         perror("Can\'t fork.\n");
20         return ERROR;
21     }
22     else if (!childpid_1)
23     {
24         // Это процесс потомок.
25         printf("First child: id: %d ppid: %d  pgrp: %d\n", getpid(),
26               getppid(), getpgrp());
27         sleep(SLEEP_TIME);
28         exit(OK);
29     }
30
31     // Аналогично 2 процесс.
32     if ((childpid_2 = fork()) == ERROR_FORK)
33     {
34         perror("Can\'t fork.\n");
35         return ERROR;
36     }
37     else if (!childpid_2)
38     {
39         // Это процесс потомок.
40         printf("Second child: id: %d ppid: %d  pgrp: %d\n", getpid(),
41               getppid(), getpgrp());
42         sleep(SLEEP_TIME);
43         exit(OK);
44     }
45 }

```

```

44     printf("Parent: id: %d pgrp: %d child1: %d child2: %d\n", getpid(),
45           getpgrp(), childpid_1, childpid_2);
46     return OK;
47 }

```



```

lab_04 [master] gcc f1.c
lab_04 [master] ./a.out
Parent: id: 30663 pgrp: 30663 child1: 30664 child2: 30665
First child: id: 30664 ppid: 30663 pgrp: 30663
Second child: id: 30665 ppid: 30663 pgrp: 30663

```

Рисунок 0.1 — Результат работы программы 1.

Листинг 2 — Программа 2.

```

1  #include <stdio.h>
2  #include <unistd.h>
3  #include <sys/wait.h>
4  #include <stdlib.h>
5
6  #define OK 0
7  #define ERROR 1
8  #define ERROR_FORK -1
9  #define SLEEP_TIME 2
10
11 void check_status(int status);
12
13 int main()
14 {
15     int childpid_1, childpid_2;
16
17     if ((childpid_1 = fork()) == ERROR_FORK)
18     {
19         // Если при порождении процесса произошла ошибка.
20         perror("Can\'t fork.\n");
21         return ERROR;
22     }
23     else if (!childpid_1)
24     {
25         // Это процесс потомок.
26         printf("First child: id: %d ppid: %d pgrp: %d\n", getpid(),
27               getppid(), getpgrp());
28         sleep(SLEEP_TIME);
29         exit(OK);
30     }

```

```

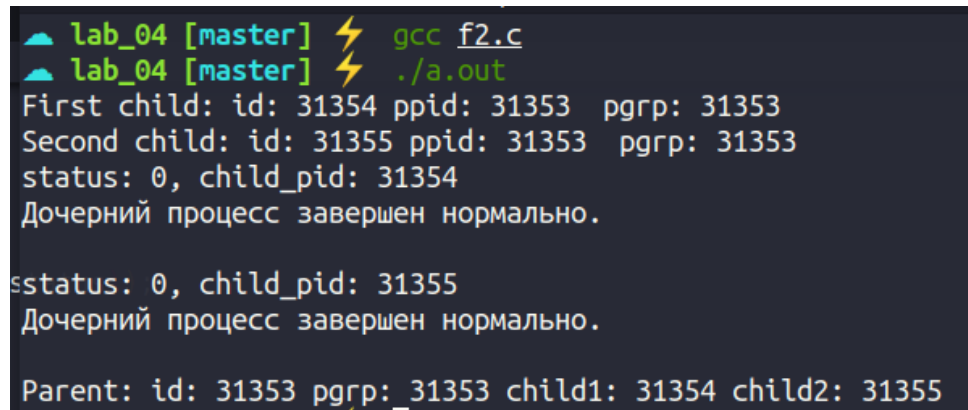
31 // Аналогично 2 процесс.
32 if ((childpid_2 = fork()) == ERROR_FORK)
33 {
34     perror("Can\'t fork.\n");
35     return ERROR;
36 }
37 else if (!childpid_2)
38 {
39     // Это процесс потомок.
40     printf("Second child: id: %d ppid: %d pgrp: %d\n", getpid(),
41           getppid(), getpgrp());
42     sleep(SLEEP_TIME);
43     exit(OK);
44 }
45
46 int status;
47 pid_t child_pid;
48
49 child_pid = wait(&status);
50 printf("status: %d, child_pid: %d\n", status, child_pid);
51 check_status(status);
52
53 child_pid = wait(&status);
54 printf("status: %d, child_pid: %d\n", status, child_pid);
55 check_status(status);
56
57 printf("Parent: id: %d pgrp: %d child1: %d child2: %d\n", getpid(),
58       getpgrp(), childpid_1, childpid_2);
59
60 return OK;
61 }
62
63 void check_status(int status)
64 {
65     if (WIFEXITED(status))
66     {
67         printf("Дочерний процесс завершен нормально.\n\n");
68         return;
69     }
70
71     if (WEXITSTATUS(status))
72     {
73         printf("Код завершения дочернего процесса %d.\n",
74               WIFEXITED(status));
75         return;
76     }
77 }

```

```

75     if (WIFSIGNALED(status))
76     {
77         printf("Дочерний процесс завершается перехватываемым сигналом\n");
78         printf("Номер сигнала %d.\n", WTERMSIG(status));
79         return;
80     }
81
82     if (WIFSTOPPED(status))
83     {
84         printf("Дочерний процесс остановился.\n");
85         printf("Номер сигнала %d.", WSTOPSIG(status));
86     }
87 }

```



```

lab_04 [master] ⚡ gcc f2.c
lab_04 [master] ⚡ ./a.out
First child: id: 31354 ppid: 31353 pgrp: 31353
Second child: id: 31355 ppid: 31353 pgrp: 31353
status: 0, child_pid: 31354
Дочерний процесс завершен нормально.

status: 0, child_pid: 31355
Дочерний процесс завершен нормально.

Parent: id: 31353 pgrp: 31353 child1: 31354 child2: 31355

```

Рисунок 0.2 — Результат работы программы 2.