



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 дисциплине "Операционные системы"

Тема Процессы. Системные вызовы fork() и exec()

Студент Фролов Е.А.

Группа ИУ7-55Б

Оценка (баллы)                     

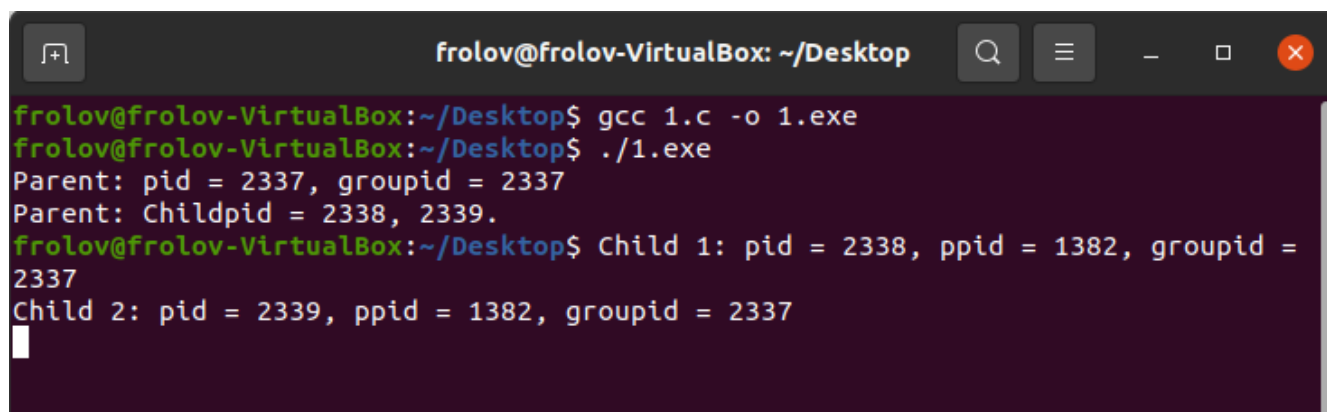
Преподаватели Рязанова Н.Ю.

## Задание №1

Процессы-сироты. В программе создаются не менее двух потомков. В потомках вызывается `sleep()`. Чтобы предок гарантированно завершился раньше своих помков. Продемонстрировать с помощью соответствующего вывода информацию об идентификаторах процессов и их группе.

Листинг 1: Процессы-сироты

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4
5 int main()
6 {
7     int first_child = fork();
8     if (first_child == -1)
9     {
10         perror("Can't fork.");
11         exit(1);
12     }
13     else if (first_child == 0)
14     {
15         sleep(2);
16         printf("Child 1: pid = %d, ppid = %d, groupid = %d\n", getpid(),
17             getppid(), getpgrp());
18         return 0;
19     }
20
21     printf("Parent: pid = %d, groupid = %d\n", getpid(), getpgrp());
22
23     int second_child = fork();
24     if (second_child == -1)
25     {
26         perror("Can't fork.");
27         exit(1);
28     }
29     else if (second_child == 0)
30     {
31         sleep(2);
32         printf("Child 2: pid = %d, ppid = %d, groupid = %d\n", getpid(),
33             getppid(), getpgrp());
34         return 0;
35     }
36
37     printf("Parent: Childpid = %d, %d.\n", first_child, second_child);
38
39     return 0;
40 }
```

A terminal window titled 'frolov@frolov-VirtualBox: ~/Desktop'. The window has a dark background with light-colored text. The terminal shows the following commands and output:

```
frolov@frolov-VirtualBox:~/Desktop$ gcc 1.c -o 1.exe
frolov@frolov-VirtualBox:~/Desktop$ ./1.exe
Parent: pid = 2337, groupid = 2337
Parent: Childpid = 2338, 2339.
frolov@frolov-VirtualBox:~/Desktop$ Child 1: pid = 2338, ppid = 1382, groupid = 2337
Child 2: pid = 2339, ppid = 1382, groupid = 2337
```

Рис. 1: Демонстрация работы программы (задание №1).

## Задание №2

Предок ждет завершения своих потомков, используя системный вызов `wait()`. Вывод соответствующих сообщений на экран.

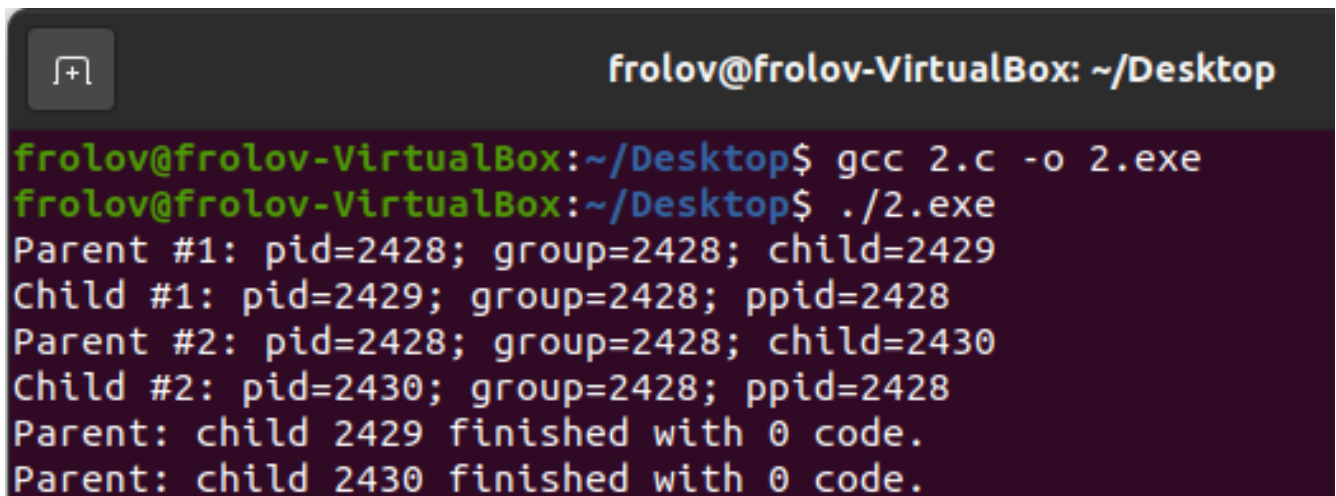
Листинг 2: Вызов функции `wait()`

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/types.h>
4 #include <sys/wait.h>
5 #include <unistd.h>
6
7 int main()
8 {
9     int first_child = fork();
10
11     if (first_child == -1)
12     {
13         perror("Couldn't fork.");
14         exit(1);
15     }
16     else if (first_child == 0)
17     {
18         printf("Child #1: pid=%d; group=%d; ppid=%d\n", getpid(), getpgrp(),
19               getppid());
20         return 0;
21     }
22     printf("Parent #1: pid=%d; group=%d; child=%d\n", getpid(), getpgrp(),
23           first_child);
24
25     int second_child = fork();
26     if (second_child == -1)
27     {
28         perror("Couldn't fork.");
29         exit(1);
30     }
31     else if (second_child == 0)
32     {
33         printf("Child #2: pid=%d; group=%d; ppid=%d\n", getpid(), getpgrp(),
34               getppid());
35         return 0;
36     }
37     printf("Parent #2: pid=%d; group=%d; child=%d\n", getpid(), getpgrp(),
38           second_child );
39
40     if (first_child != 0 && second_child != 0)
41     {
42         int status1;
43         int status2;
44         pid_t ret_value1 = wait(&status1);
45
46         if (WIFEXITED(status1))
47             printf("Parent: child %d finished with %d code.\n", ret_value1,
48                   WEXITSTATUS(status1));
49         else if (WIFSIGNALED(status1))
```

```

45     printf( "Parent: child %d finished from signal with %d code.\n",
46             ret_value1, WTERMSIG(status1));
47     else if (WIFSTOPPED(status1))
48         printf("Parent: child %d finished from signal with %d code.\n",
49             ret_value1, WSTOPSIG(status1));
50
51     pid_t ret_value2 = wait(&status2);
52
53     if (WIFEXITED(status2))
54         printf("Parent: child %d finished with %d code.\n", ret_value2,
55             WEXITSTATUS(status2));
56     else if (WIFSIGNALED(status2))
57         printf( "Parent: child %d finished from signal with %d code.\n",
58             ret_value2, WTERMSIG(status2));
59     else if (WIFSTOPPED(status2))
60         printf("Parent: child %d finished from signal with %d code.\n",
61             ret_value2, WSTOPSIG(status2));
62 }
63
64 return 0;
65 }

```



```

frolov@frolov-VirtualBox: ~/Desktop
frolov@frolov-VirtualBox:~/Desktop$ gcc 2.c -o 2.exe
frolov@frolov-VirtualBox:~/Desktop$ ./2.exe
Parent #1: pid=2428; group=2428; child=2429
Child #1: pid=2429; group=2428; ppid=2428
Parent #2: pid=2428; group=2428; child=2430
Child #2: pid=2430; group=2428; ppid=2428
Parent: child 2429 finished with 0 code.
Parent: child 2430 finished with 0 code.

```

Рис. 2: Демонстрация работы программы (задание №2).

## Задание №3

Потомки переходят на выполнение других программ. Предок ждет завершения своих потомков. Вывод соответствующих сообщений на экран.

Я взял стрые две программы из курса СИ.

Листинг 3: Среднее арифметическое отрицательных элементов массива

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define N 10
5
6 typedef enum
7 {
8     ok,
9     er_empty_input_file,
10    er_invalid_array_size,
11    er_no_task_answer,
12    er_invalid_count_of_elements
13 } status_code;
14
15 status_code array_in(int (*a)[N], int *n)
16 {
17     printf("Vvedite razmer massiva: ");
18     if (scanf("%d", n) != 1)
19         return er_empty_input_file;
20     if (*n < 1 || *n > N)
21         return er_invalid_array_size;
22
23     //
24
25     int count = 0;
26     for (int i = 0; i < *n; i++)
27     {
28         count += scanf("%d", &(*a)[i]);
29     }
30
31     return ok;
32 }
33
34 status_code mean_negative(int a[N], int n, double *mean)
35 {
36     int sum = 0;
37     int count = 0;
38     for (int i = 0; i < n; i++)
39     {
40         if (a[i] < 0)
41         {
42             count += 1;
43             sum += a[i];
44         }
45     }
46 }
```

```

47     if (count == 0)
48     {
49         *mean = 0;
50         return er_no_task_answer;
51     }
52     else
53     {
54         *mean = (double)sum / count;
55         return ok;
56     }
57 }
58
59 int main()
60 {
61     int a[N];
62     status_code err;
63     int n;
64
65     err = array_in(&a, &n);
66     if (err)
67     {
68         printf("Input error.");
69         return err;
70     }
71
72     double mean;
73     err = mean_negative(a, n, &mean);
74     if (err)
75         printf("Net otricatel'nyx.\n");
76     else
77         printf("%.3lf\n", mean);
78
79     return err;
80 }

```

Листинг 4: Площадь треугольника по двум сторонам и углу

```

1  #include <stdio.h>
2  #include <math.h>
3
4  int main()
5  {
6      double a, alpha, b, h = 0, s = 0, pi = 3.1415926;
7      printf("Input (a, b, cos(a,b): ");
8      scanf("%lf %lf %lf", &a, &b, &alpha);
9
10     alpha = alpha * pi / 180;
11
12     h = fabs(((b - a) / 2) * tan(alpha));
13
14     s = ((a + b) / 2) * h;
15
16     printf("%.2lf\n", s);
17     return 0;
18 }

```

## Листинг 5: Вызов функции `execvp()`

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/wait.h>
5
6 #define F_PROG "neg_avrg.exe"
7 #define S_PROG "squad.exe"
8
9 int main()
10 {
11     int first_child = fork();
12     if ( first_child == -1 )
13     {
14         perror("Couldn't fork.");
15         exit(1);
16     }
17     else if ( first_child == 0 )
18     {
19         printf( "Child: pid=%d; group=%d; parent=%d\n\n", getpid(), getpgrp(),
20                getppid());
21         if (execl(F_PROG, F_PROG, NULL, (char *)NULL) == -1)
22         {
23             perror( "Child couldn't exec." );
24             exit(1);
25         }
26     }
27
28     int second_child = fork();
29     if (second_child == -1)
30     {
31         perror("Couldn't fork.");
32         exit(1);
33     }
34     else if (second_child == 0)
35     {
36         printf( "Child: pid=%d, group=%d; parent=%d\n\n", getpid(), getpgrp(),
37                getppid());
38         if ( execl(S_PROG, S_PROG, NULL, (char *)NULL) == -1)
39         {
40             perror( "Child couldn't exec." );
41             exit(1);
42         }
43     }
44
45     if (first_child != 0 && second_child != 0)
46     {
47         printf( "Parent: pid=%d; group=%d; first_child=%d, second_child=%d\n",
48                getpid(), getpgrp(), first_child, second_child );
49         int status1, status2;
50
51         pid_t ret_value1 = wait( &status1 );
52         if ( WIFEXITED(status1) )
53             printf("Parent: child %d finished with %d code.\n\n", ret_value1,
54                    WEXITSTATUS(status1) );
55         else if ( WIFSIGNALED(status1) )

```



```

52     printf( "Parent: child %d finished from signal with %d code.\n\n",
           ret_value1, WTERMSIG(status1));
53     else if ( WIFSTOPPED(status1) )
54         printf("Parent: child %d finished from signal with %d code.\n\n",
           ret_value1, WSTOPSIG(status1));
55
56     pid_t ret_value2 = wait( &status2 );
57     if ( WIFEXITED(status2) )
58         printf("Parent: child %d finished with %d code.\n\n", ret_value2,
           WEXITSTATUS(status2) );
59     else if ( WIFSIGNALED(status2) )
60         printf( "Parent: child %d finished from signal with %d code.\n\n",
           ret_value2, WTERMSIG(status2));
61     else if ( WIFSTOPPED(status2) )
62         printf("Parent: child %d finished from signal with %d code.\n\n",
           ret_value2, WSTOPSIG(status2));
63 }
64 return 0;
65 }

```

```

frolov@frolov-VirtualBox: ~/Desktop
frolov@frolov-VirtualBox:~/Desktop$ ./a.out
Parent: pid=3206; group=3206; first_child=3207, second_child=3208
Child: pid=3207; group=3206; parent=3206

Child: pid=3208, group=3206; parent=3206

Vvedite razmer massiva: 4
1 -1 -3 2
-2.000
Parent: child 3207 finished with 0 code.

Input (a, b, cos(a,b): 5 10 40
16.07
Parent: child 3208 finished with 0 code.

```

Рис. 3: Демонстрация работы программы (задание №3).

## Задание №4

Предок и потомки обмениваются сообщениями через неименованный программный канал. Предок ждет завершения своих потомков. Вывод соответствующих сообщений на экран.

Листинг 6: Использование pipe

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <string.h>
5 #include <wait.h>
6
7 int main()
8 {
9     int fd[2];
10
11     printf("Parent: PID=%d, GROUP=%d\n", getpid(), getpgrp());
12
13     if (pipe(fd) == -1)
14     {
15         perror("Couldn't pipe.");
16         exit(1);
17     }
18
19     int first_child = fork();
20     if (first_child == -1)
21     {
22         perror("Couldn't fork.");
23         exit(1);
24     }
25
26     else if (first_child == 0)
27     {
28         char msg[] = "(The first child.)";
29         close(fd[0]);
30         write(fd[1], msg, sizeof msg - 1);
31         exit(0);
32     }
33
34     int second_child = fork();
35     if (second_child == -1)
36     {
37         perror("Couldn't fork.");
38         exit(1);
39     }
40
41     else if (second_child == 0)
42     {
43         char msg[] = "(Message from the second child of a different length.)";
44         write(fd[1], msg, sizeof msg - 1);
45         exit(0);
46     }
47
48
49
50
```

```

51  if (first_child != 0 && second_child != 0)
52  {
53      close(fd[1]);
54      char msg[150];
55      read(fd[0], msg, sizeof msg);
56
57      printf("\nParent reads: %s\n", msg);
58
59      int status1, status2;
60      pid_t val1 = wait(&status1);
61
62      if (WIFEXITED(status1))
63          printf("Parent: (child %d) finished with %d code.\n\n", val1,
64                  WEXITSTATUS(status1));
65
66      else if (WIFSIGNALED(status1))
67          printf("Parent: (child %d) finished from signal with %d
68                  code.\n\n", val1, WTERMSIG(status1));
69
70      else if (WIFSTOPPED(status1))
71          printf("Parent: (child %d) finished from signal with %d
72                  code.\n\n", val1, WSTOPSIG(status1));
73
74      pid_t val2 = wait(&status2);
75
76      if (WIFEXITED(status2))
77          printf("Parent: (child %d) finished with %d code.\n\n", val2,
78                  WEXITSTATUS(status2));
79
80      else if (WIFSIGNALED(status2))
81          printf("Parent: (child %d) finished from signal with %d
82                  code.\n\n", val2, WTERMSIG(status2));
83
84      else if (WIFSTOPPED(status2))
85          printf("Parent: (child %d) finished from signal with %d
86                  code.\n\n", val2, WSTOPSIG(status2));
87
88  }
89
90  return 0;
91 }

```

```

frolov@frolov-VirtualBox: ~/Desktop
frolov@frolov-VirtualBox:~/Desktop$ gcc 4.c -o 4.exe
frolov@frolov-VirtualBox:~/Desktop$ ./4.exe
Parent: PID=3372, GROUP=3372

Parent reads: (The first child.)(Message from the second child of a different length.)
Parent: (child 3373) finished with 0 code.

Parent: (child 3374) finished with 0 code.

```

Рис. 4: Демонстрация работы программы (задание №4).

## Задание №5

Предок и потомки обмениваются сообщениями через неименованный программный канал. С помощью сигнала меняется ход выполнения программы. Предок ждет завершения своих потомков. Вывод соответствующих сообщений на экран.

Листинг 7: Использование сигналов

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <string.h>
5 #include <wait.h>
6 #include <signal.h>
7 #include <time.h>
8 int mode = 1;
9
10 void catcher(int sigint)
11 {
12     printf("Process caught signal #%d\n", sigint);
13     mode = 1;
14 }
15
16 int main()
17 {
18     int fd[2];
19     if (pipe(fd) == -1)
20     {
21         perror("Couldn't pipe.");
22         exit(1);
23     }
24
25     signal(SIGINT, catcher);
26
27     int first_child = fork();
28     if (first_child == -1)
29     {
30         perror("Couldn't fork.");
31         exit(1);
32     }
33     else if (first_child == 0)
34     {
35         sleep(3);
36         char msg[];
37         if (mode)
38             sprintf(msg, "(The first child. The signal was caught.)");
39         else
40             sprintf(msg, "(The first child. The signal wasnt caught.)");
41         close(fd[0]);
42         (write(fd[1], msg, sizeof msg - 1));
43         exit(0);
44     }
45
46
47
48
```

```

49     int second_child = fork();
50
51     if (second_child == -1)
52     {
53         perror("Couldn't fork.");
54         exit(1);
55     }
56     else if (second_child == 0)
57     {
58         sleep(3);
59         char msg[];
60         if (mode)
61             sprintf(msg, "(Message from the second child of a different
62                 length. The signal was caught.)");
63         else
64             sprintf(msg, "(Message from the second child of a different
65                 length. The signal wasnt caught.)");
66         close(fd[0]);
67         (write(fd[1], msg, sizeof msg - 1));
68         exit(0);
69     }
70
71     if (first_child != 0 && second_child != 0) {
72
73         int status1, status2;
74         pid_t val1 = wait(&status1);
75         if (WIFEXITED(status1))
76             printf("Parent: (child %d) finished with %d code.\n\n", val1,
77                 WEXITSTATUS(status1));
78         else if (WIFSIGNALED(status1))
79             printf("Parent: (child %d) finished from signal with %d
80                 code.\n\n", val1, WTERMSIG(status1));
81         else if (WIFSTOPPED(status1))
82             printf("Parent: (child %d) finished from signal with %d
83                 code.\n\n", val1, WSTOPSIG(status1));
84
85         pid_t val2 = wait(&status2);
86         if (WIFEXITED(status2))
87             printf("Parent: (child %d) finished with %d code.\n\n", val2,
88                 WEXITSTATUS(status2));
89         else if (WIFSIGNALED(status2))
90             printf("Parent: (child %d) finished from signal with %d
91                 code.\n\n", val2, WTERMSIG(status2));
92         else if (WIFSTOPPED(status2))
93             printf("Parent: (child %d) finished from signal with %d
94                 code.\n\n", val2, WSTOPSIG(status2));
95
96         close(fd[1]);
97         char msg[150];
98         read(fd[0], msg, sizeof msg);
99
100        printf("\nParent reads: %s\n", msg);
101        return 0;
102    }
103 }

```

```
frolov@frolov-VirtualBox: ~/Desktop
frolov@frolov-VirtualBox:~/Desktop$ gcc 5.c -o 5.exe
frolov@frolov-VirtualBox:~/Desktop$ ./5.exe
Parent: (child 3700) finished with 0 code.
Parent: (child 3701) finished with 0 code.
Parent: reads: (The first child. The signal wasnt caught.)(Message from the second child of a different length. The signal wasnt caught.)
```

Рис. 5: Демонстрация работы программы, если сигнал не был получен. (задание №5).

```
frolov@frolov-VirtualBox: ~/Desktop
frolov@frolov-VirtualBox:~/Desktop$ gcc 5.c -o 5.exe
frolov@frolov-VirtualBox:~/Desktop$ ./5.exe
Parent: (child 3668) finished with 0 code.
Parent: (child 3669) finished with 0 code.
Parent: reads: (The first child. The signal was caught.)(Message from the second child of a different length. The signal was caught.)
```

Рис. 6: Демонстрация работы программы, если сигнал был получен. (задание №5).