



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»

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

Студент Кононенко С.С.

Группа ИУ7-53Б

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 потомков. Продемонстрировать с помощью соответствующего вывода информацию об идентификаторах процессов и их группе.

Листинг 1 – Процессы-сироты

```
1 #include <stdio.h>
2 #include <unistd.h>
3
4 #define N 2
5 #define INTERVAL 30
6
7 int pid;
8 int child_pids[N];
9
10 int main()
11 {
12     printf("Parent_process: PID=%d, GROUP=%d\n", getpid(), getpgrp());
13
14     for (size_t i = 0; i < N; ++i)
15     {
16         switch (pid = fork())
17         {
18             case -1:
19                 perror("Can't fork\n");
20
21                 return 1;
22             case 0:
23                 printf("Child_process: PID=%d, GROUP=%d, PPID=%d\n", getpid(),
24                     , getpgrp(), getppid());
25                 sleep(INTERVAL);
26
27                 return 0;
28             default:
29                 child_pids[i] = pid;
30         }
31     }
32
33     printf("Parent_process have children with IDs: %d, %d\n", child_pids
34         [0], child_pids[1]);
35     printf("Parent_process is dead now\n");
36 }
```

```

35     return 0;
36 }

```

```

~/dosbox/os-5th-sem-labs/lab_04/src > master !5 ./task01.exe
Parent process: PID=22628, GROUP=22628
Parent process have children with IDs: 22629, 22630
Parent process is dead now
Child process : PID=22630, GROUP=22628, PPID=22628
Child process : PID=22629, GROUP=22628, PPID=22628

```

Рисунок 1 – Демонстрация работы программы

## Задание 2

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

Листинг 2 – Вызов `wait()`

```

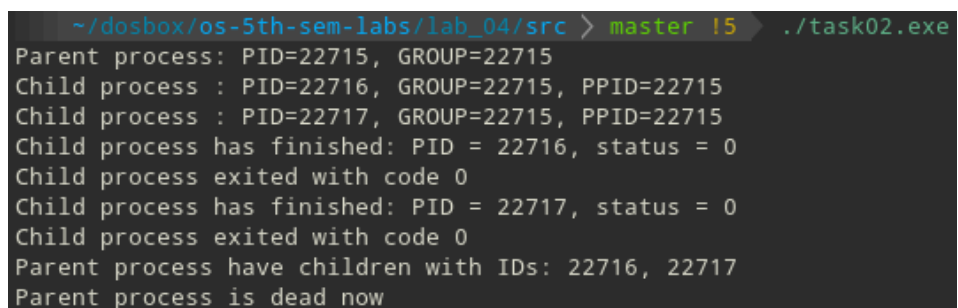
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <sys/wait.h>
4  #include <sys/types.h>
5
6  #define N 2
7  #define INTERVAL 2
8
9  int pid;
10 int child_pids[N];
11
12 int main()
13 {
14     printf("Parent process: PID=%d, GROUP=%d\n", getpid(), getpgrp());
15
16     for (size_t i = 0; i < N; ++i)
17     {
18         switch (pid = fork())
19         {
20             case -1:
21                 perror("Can't fork\n");
22
23                 return 1;
24             case 0:
25                 printf("Child process: PID=%d, GROUP=%d, PPID=%d\n", getpid(),
26                     getpgrp(), getppid());
27                 sleep(INTERVAL);

```

```

28         return 0;
29     default:
30         child_pids[i] = pid;
31     }
32 }
33
34 for (size_t i = 0; i < N; ++i)
35 {
36     int status, stat_val;
37     pid_t childpid = wait(&status);
38
39     printf("Child process has finished: PID=%d, status=%d\n",
40           childpid, status);
41
42     if (WIFEXITED(stat_val))
43     {
44         printf("Child process exited with code %d\n", WEXITSTATUS(
45             stat_val));
46     }
47     else
48     {
49         printf("Child process terminated abnormally\n");
50     }
51 }
52
53 printf("Parent process have children with IDs: %d, %d\n", child_pids
54       [0], child_pids[1]);
55 printf("Parent process is dead now\n");
56
57 return 0;
58 }

```



```

~/.dosbox/os-5th-sem-labs/lab_04/src > master !5 ./task02.exe
Parent process: PID=22715, GROUP=22715
Child process : PID=22716, GROUP=22715, PPID=22715
Child process : PID=22717, GROUP=22715, PPID=22715
Child process has finished: PID = 22716, status = 0
Child process exited with code 0
Child process has finished: PID = 22717, status = 0
Child process exited with code 0
Parent process have children with IDs: 22716, 22717
Parent process is dead now

```

Рисунок 2 – Демонстрация работы программы

## Задание 3

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

Листинг 3 – Вызов `execvp()`

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <sys/wait.h>
4 #include <sys/types.h>
5
6 #define N 2
7
8 int pid;
9 int child_pids[N];
10 const char *const COMMANDS[N] = {"ls", "whoami"};
11
12 int main()
13 {
14     printf("Parent process: PID=%d, GROUP=%d\n", getpid(), getpgrp());
15
16     for (size_t i = 0; i < N; ++i)
17     {
18         switch (pid = fork())
19         {
20             case -1:
21                 perror("Can't fork\n");
22
23                 return 1;
24             case 0:
25                 printf("Child process: PID=%d, GROUP=%d, PPID=%d\n\n", getpid(),
26                     getpgrp(), getppid());
27
28                 switch (execvp(COMMANDS[i], COMMANDS[i], 0))
29                 {
30                     case -1:
31                         perror("Can't exec\n");
32
33                         return 1;
34                     case 0:
35                         return 0;
36                 }
37                 default:
38                     child_pids[i] = pid;
39             }
40         }
41     }
```

```

40
41     for (size_t i = 0; i < N; ++i)
42     {
43         int status, stat_val;
44         pid_t childpid = wait(&status);
45
46         printf("\nChild process has finished: PID=%d, status=%d\n",
47             childpid, status);
48
49         if (WIFEXITED(stat_val))
50         {
51             printf("Child process exited with code %d\n", WEXITSTATUS(
52                 stat_val));
53         }
54         else
55         {
56             printf("Child process terminated abnormally\n");
57         }
58     }
59
60     printf("Parent process have children with IDs: %d, %d\n", child_pids
61         [0], child_pids[1]);
62     printf("Parent process is dead now\n");
63
64     return 0;
65 }

```

```

~/dosbox/os-5th-sem-labs/lab_04/src > master t5 ./task03.exe
Parent process: PID=22751, GROUP=22751
Child process : PID=22752, GROUP=22751, PPID=22751

Child process : PID=22753, GROUP=22751, PPID=22751

task01.c task01.exe task02.c task02.exe task03.c task03.exe task04.c task04.exe task05.c task05.exe

Child process has finished: PID = 22752, status = 0
Child process exited with code 0
hackfeed

Child process has finished: PID = 22753, status = 0
Child process exited with code 0
Parent process have children with IDs: 22752, 22753
Parent process is dead now

```

Рисунок 3 – Демонстрация работы программы

## Задание 4

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

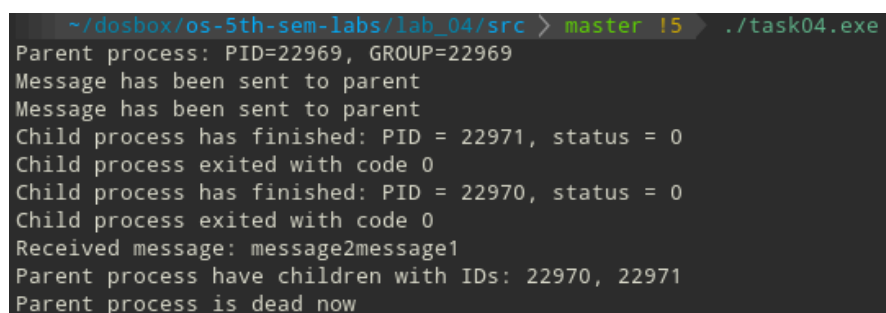
## Листинг 4 – Использование pipe

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <stdlib.h>
4 #include <string.h>
5 #include <sys/wait.h>
6 #include <sys/types.h>
7
8 #define N 2
9 #define BUFLen 100
10
11 int pid;
12 int child_pids[N];
13 const char *PIPEMSG[N] = {"message1", "message2"};
14
15 int main()
16 {
17     int fd[2];
18     char buffer[BUFLen] = {0};
19
20     printf("Parent process: PID=%d, GROUP=%d\n", getpid(), getpgrp());
21
22     if (pipe(fd) == -1)
23     {
24         perror("Can't pipe\n");
25
26         return 1;
27     }
28
29     for (size_t i = 0; i < N; ++i)
30     {
31         switch (pid = fork())
32         {
33             case -1:
34                 perror("Can't fork\n");
35
36                 exit(1);
37             case 0:
38                 close(fd[0]);
39                 write(fd[1], PIPEMSG[i], strlen(PIPEMSG[i]));
40                 printf("Message has been sent to parent\n");
41
42                 exit(0);
43             default:
44                 child_pids[i] = pid;
45         }
46     }
47 }
```

```

48     for (size_t i = 0; i < N; ++i)
49     {
50         int status, stat_val;
51         pid_t childpid = wait(&status);
52
53         printf("Child process has finished: PID=%d, status=%d\n",
54             childpid, status);
55
56         if (WIFEXITED(stat_val))
57         {
58             printf("Child process exited with code %d\n", WEXITSTATUS(
59                 stat_val));
60         }
61         else
62         {
63             printf("Child process terminated abnormally\n");
64         }
65     }
66
67     close(fd[1]);
68     read(fd[0], buffer, BUFLLEN);
69     printf("Received message: %s\n", buffer);
70
71     printf("Parent process have children with IDs: %d, %d\n", child_pids
72         [0], child_pids[1]);
73     printf("Parent process is dead now\n");
74
75     return 0;
76 }

```



```

~/dosbox/os-5th-sem-labs/lab_04/src > master i5 > ./task04.exe
Parent process: PID=22969, GROUP=22969
Message has been sent to parent
Message has been sent to parent
Child process has finished: PID = 22971, status = 0
Child process exited with code 0
Child process has finished: PID = 22970, status = 0
Child process exited with code 0
Received message: message2message1
Parent process have children with IDs: 22970, 22971
Parent process is dead now

```

Рисунок 4 – Демонстрация работы программы

## Задание 5

Предок и потомки обмениваются сообщениями через неименованный программный канал. С помощью сигнала меняется ход выполнения про-



граммы. Предок ждет завершения своих потомков. Вывод соответствующих сообщений на экран.

#### Листинг 5 – Использование сигналов

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <stdlib.h>
4 #include <string.h>
5 #include <signal.h>
6 #include <sys/wait.h>
7 #include <sys/types.h>
8
9 #define N 2
10 #define BUFLen 100
11 #define INTERVAL 5
12
13 int pid;
14 int child_pids[N];
15 const char *PIPEMSG[N] = {"message1", "message2"};
16 int state = 0;
17
18 void reverse(char *x, int begin, int end)
19 {
20     char c;
21
22     if (begin >= end)
23         return;
24
25     c = *(x + begin);
26     *(x + begin) = *(x + end);
27     *(x + end) = c;
28
29     reverse(x, ++begin, --end);
30 }
31
32 void reverse_buf(int sig)
33 {
34     state = 1;
35 }
36
37 int main()
38 {
39     int fd[2];
40     char buffer[BUFLen] = {0};
41
42     printf("Parent process: PID=%d, GROUP=%d\n", getpid(), getpgrp());
43
44     if (pipe(fd) == -1)
```

```

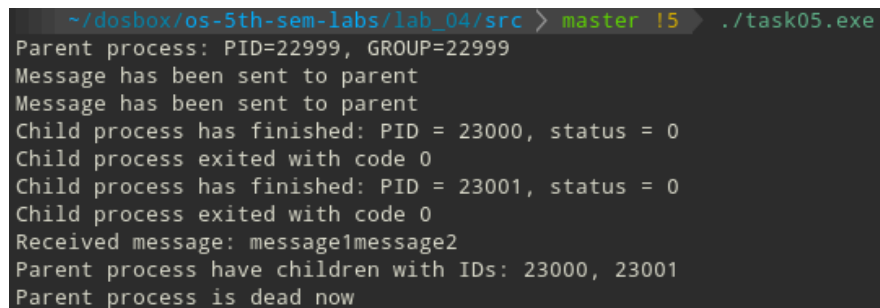
45 {
46     perror("Can't pipe\n");
47
48     return 1;
49 }
50
51 signal(SIGINT, reverse_buf);
52
53 for (size_t i = 0; i < N; ++i)
54 {
55     switch (pid = fork())
56     {
57     case -1:
58         perror("Can't fork\n");
59
60         exit(1);
61     case 0:
62         close(fd[0]);
63         write(fd[1], PIPEMSG[i], strlen(PIPEMSG[i]));
64         printf("Message has been sent to parent\n");
65
66         exit(0);
67     default:
68         child_pids[i] = pid;
69     }
70 }
71
72 for (size_t i = 0; i < N; ++i)
73 {
74     int status, stat_val;
75     pid_t childpid = wait(&status);
76
77     printf("Child process has finished: PID=%d, status=%d\n",
78           childpid, status);
79
80     if (WIFEXITED(stat_val))
81     {
82         printf("Child process exited with code %d\n", WEXITSTATUS(
83             stat_val));
84     }
85     else
86     {
87         printf("Child process terminated abnormally\n");
88     }
89 }
90
91 close(fd[1]);
92 read(fd[0], buffer, BUFLen);

```

```

91     sleep(INTERVAL);
92
93     if (state)
94     {
95         reverse(buffer, 0, strlen(buffer) - 1);
96     }
97
98     printf("Received_message:_%s\n", buffer);
99
100    printf("Parent_process_have_children_with_IDs:_%d,_%d\n", child_pids
        [0], child_pids[1]);
101    printf("Parent_process_is_dead_now\n");
102
103    return 0;
104 }

```

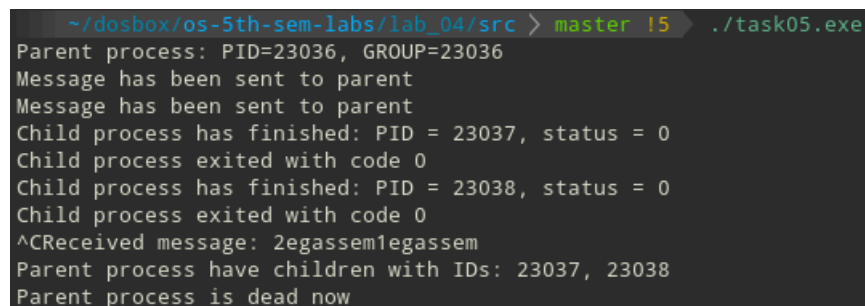


```

~/dosbox/os-5th-sem-labs/lab_04/src > master !5 > ./task05.exe
Parent process: PID=22999, GROUP=22999
Message has been sent to parent
Message has been sent to parent
Child process has finished: PID = 23000, status = 0
Child process exited with code 0
Child process has finished: PID = 23001, status = 0
Child process exited with code 0
Received message: message1message2
Parent process have children with IDs: 23000, 23001
Parent process is dead now

```

Рисунок 5 – Демонстрация работы программы (сигнал не вызывается)



```

~/dosbox/os-5th-sem-labs/lab_04/src > master !5 > ./task05.exe
Parent process: PID=23036, GROUP=23036
Message has been sent to parent
Message has been sent to parent
Child process has finished: PID = 23037, status = 0
Child process exited with code 0
Child process has finished: PID = 23038, status = 0
Child process exited with code 0
^CReceived message: 2egassem1egassem
Parent process have children with IDs: 23037, 23038
Parent process is dead now

```

Рисунок 6 – Демонстрация работы программы (сигнал вызывается)