



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 main()
8 {
9     printf("Parent_process: PID=%d, GROUP=%d\n", getpid(), getpgrp());
10
11     for (size_t i = 0; i < N; ++i)
12     {
13         switch (fork())
14         {
15             case -1:
16                 perror("Can't fork\n");
17
18                 return 1;
19             case 0:
20                 printf("Child_process: PID=%d, GROUP=%d, PPID=%d\n",
21                     getpid(), getpgrp(), getppid());
22                 sleep(INTERVAL);
23
24                 return 0;
25         }
26     }
27
28     printf("Parent_process is dead now\n");
29
30     return 0;
31 }
```

```
~/dosbox/os-5th-sem-labs/lab_04/src > master ?1 ./task01.exe
Parent process: PID=31071, GROUP=31071
Parent process is dead now
Child process : PID=31072, GROUP=31071, PPID=31071
Child process : PID=31073, GROUP=31071, PPID=31071
```

Рисунок 1 – Демонстрация работы программы

## Задание 2

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

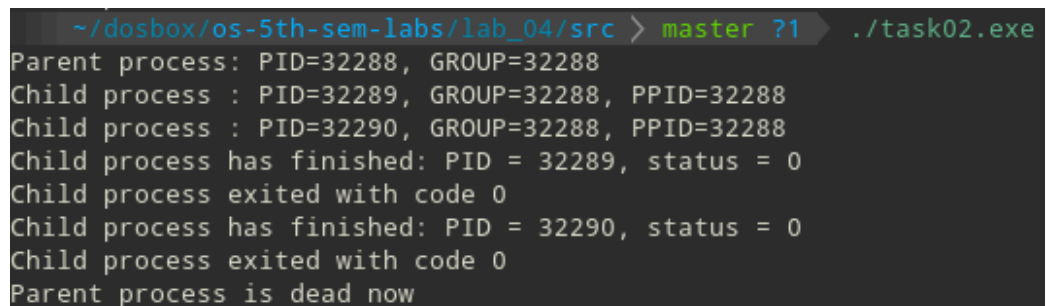
Листинг 2 – Вызов `wait()`

```
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 5
8
9 int main()
10 {
11     printf("Parent process: PID=%d, GROUP=%d\n", getpid(), getpgrp());
12
13     for (size_t i = 0; i < N; ++i)
14     {
15         switch (fork())
16         {
17             case -1:
18                 perror("Can't fork\n");
19
20                 return 1;
21             case 0:
22                 printf("Child process: PID=%d, GROUP=%d, PPID=%d\n",
23                     getpid(), getpgrp(), getppid());
24                 sleep(INTERVAL);
25
26                 return 0;
27         }
28     }
29
30     for (size_t i = 0; i < N; ++i)
31     {
32         int status, stat_val;
33         pid_t childpid = wait(&status);
```

```

34
35     printf("Child process has finished: PID=%d, status=%d\n",
36           childpid, status);
37
38     if (WIFEXITED(stat_val))
39     {
40         printf("Child process exited with code %d\n",
41               WEXITSTATUS(stat_val));
42     }
43     else
44     {
45         printf("Child process terminated abnormally\n");
46     }
47 }
48
49 printf("Parent process is dead now\n");
50
51 return 0;
52 }

```



```

~/.dosbox/os-5th-sem-labs/lab_04/src > master ?1 ./task02.exe
Parent process: PID=32288, GROUP=32288
Child process : PID=32289, GROUP=32288, PPID=32288
Child process : PID=32290, GROUP=32288, PPID=32288
Child process has finished: PID = 32289, status = 0
Child process exited with code 0
Child process has finished: PID = 32290, status = 0
Child process exited with code 0
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  const char *const COMMANDS[N] = {"ls", "whoami"};
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 (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\n",
24                     getpid(), getpgrp(), getppid());
25
26                 switch (execlp(COMMANDS[i], COMMANDS[i], 0))
27                 {
28                     case -1:
29                         perror("Can't_exec\n");
30
31                         return 1;
32                     case 0:
33                         return 0;
34                 }
35             }
36     }
37
38     for (size_t i = 0; i < N; ++i)
39     {
40         int status, stat_val;
41         pid_t childpid = wait(&status);
42
43         printf("\nChild_process_has_finished:_PID=_%d,_status=_%d\n",
44             childpid, status);
45
46         if (WIFEXITED(stat_val))
47         {
48             printf("Child_process_exited_with_code_%d\n",
49                 WEXITSTATUS(stat_val));
50         }
51         else
52         {
53             printf("Child_process_terminated_abnormally\n");
54         }
55     }

```

```

56
57     printf("Parent process is dead now\n");
58
59     return 0;
60 }

```

```

~/dosbox/os-5th-sem-labs/lab_04/src > master ?1 ./task03.exe
Parent process: PID=32366, GROUP=32366
Child process : PID=32367, GROUP=32366, PPID=32366

Child process : PID=32368, GROUP=32366, PPID=32366

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 = 32367, status = 0
Child process exited with code 0
hackfeed

Child process has finished: PID = 32368, status = 0
Child process exited with code 0
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 const char *PIPEMSG[N] = {"message1", "message2"};
12
13 int main()
14 {
15     int fd[2];
16     char buffer[BUFLen] = {0};
17
18     if (pipe(fd) == -1)
19     {
20         perror("Can't pipe\n");

```

```

21
22     return 1;
23 }
24
25 for (size_t i = 0; i < N; ++i)
26 {
27     switch (fork())
28     {
29         case -1:
30             perror("Can't fork\n");
31
32             exit(1);
33         case 0:
34             close(fd[0]);
35             write(fd[1], PIPEMSG[i], strlen(PIPEMSG[i]));
36             printf("Message has been sent to parent\n");
37
38             exit(0);
39     }
40 }
41
42 for (size_t i = 0; i < N; ++i)
43 {
44     int status, stat_val;
45     pid_t childpid = wait(&status);
46
47     printf("Child process has finished: PID=%d, status=%d\n",
48           childpid, status);
49
50     if (WIFEXITED(stat_val))
51     {
52         printf("Child process exited with code %d\n",
53               WEXITSTATUS(stat_val));
54     }
55     else
56     {
57         printf("Child process terminated abnormally\n");
58     }
59 }
60
61 close(fd[1]);
62 read(fd[0], buffer, BUFLen);
63 printf("Received message: %s\n", buffer);
64
65 printf("Parent process is dead now\n");
66
67 return 0;
68 }

```

```
~/dosbox/os-5th-sem-labs/lab_04/src > master ?1 > ./task04.exe
Message has been sent to parent
Message has been sent to parent
Child process has finished: PID = 32429, status = 0
Child process exited with code 0
Child process has finished: PID = 32430, status = 0
Child process exited with code 0
Received message: message1message2
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 программы. 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 const char *PIPEMSG[N] = {"message1", "message2"};
14 int state = 0;
15
16 void reverse(char *x, int begin, int end)
17 {
18     char c;
19
20     if (begin >= end)
21         return;
22
23     c = *(x + begin);
24     *(x + begin) = *(x + end);
25     *(x + end) = c;
26
27     reverse(x, ++begin, --end);
```



```

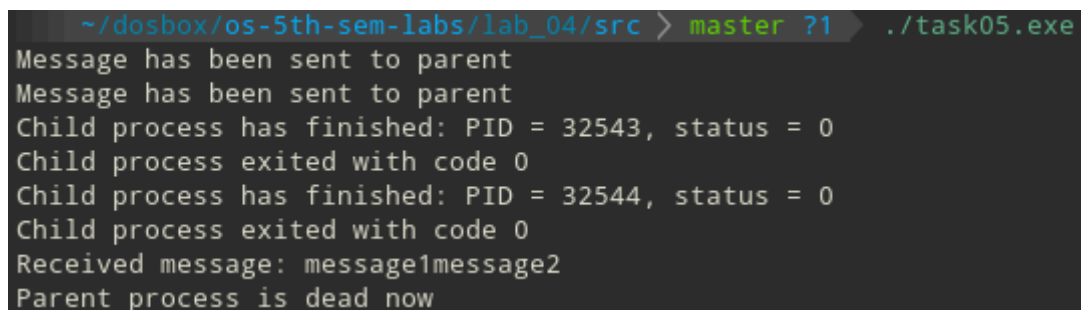
28 }
29
30 void reverse_buf(int sig)
31 {
32     state = 1;
33 }
34
35 int main()
36 {
37     int fd[2];
38     char buffer[BUFLen] = {0};
39
40     if (pipe(fd) == -1)
41     {
42         perror("Can't pipe\n");
43
44         return 1;
45     }
46
47     signal(SIGINT, reverse_buf);
48
49     for (size_t i = 0; i < N; ++i)
50     {
51         switch (fork())
52         {
53             case -1:
54                 perror("Can't fork\n");
55
56                 exit(1);
57             case 0:
58                 close(fd[0]);
59                 write(fd[1], PIPEMSG[i], strlen(PIPEMSG[i]));
60                 printf("Message has been sent to parent\n");
61
62                 exit(0);
63         }
64     }
65
66     for (size_t i = 0; i < N; ++i)
67     {
68         int status, stat_val;
69         pid_t childpid = wait(&status);
70
71         printf("Child process has finished: PID=%d, status=%d\n",
72               childpid, status);
73
74         if (WIFEXITED(stat_val))
75         {

```

```

76         printf("Child process exited with code %d\n",
77                WEXITSTATUS(stat_val));
78     }
79     else
80     {
81         printf("Child process terminated abnormally\n");
82     }
83 }
84
85 close(fd[1]);
86 read(fd[0], buffer, BUFLen);
87 sleep(INTERVAL);
88
89 if (state)
90 {
91     reverse(buffer, 0, strlen(buffer) - 1);
92 }
93
94 printf("Received message: %s\n", buffer);
95
96 printf("Parent process is dead now\n");
97
98 return 0;
99 }

```

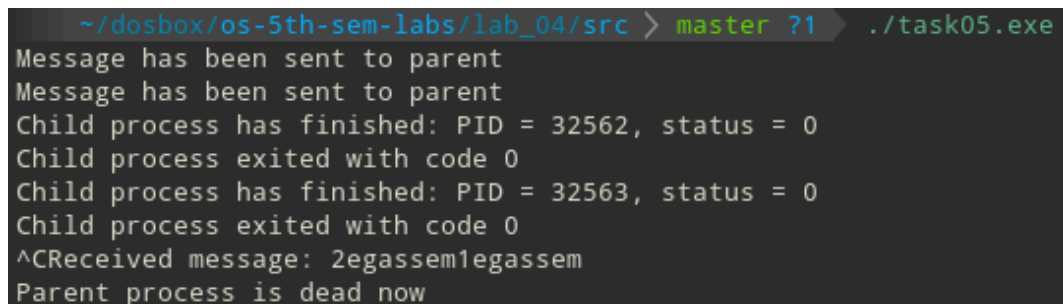


```

~/dosbox/os-5th-sem-labs/lab_04/src > master ?1 ./task05.exe
Message has been sent to parent
Message has been sent to parent
Child process has finished: PID = 32543, status = 0
Child process exited with code 0
Child process has finished: PID = 32544, status = 0
Child process exited with code 0
Received message: message1message2
Parent process is dead now

```

Рисунок 5 – Демонстрация работы программы (сигнал не вызывается)



```

~/dosbox/os-5th-sem-labs/lab_04/src > master ?1 ./task05.exe
Message has been sent to parent
Message has been sent to parent
Child process has finished: PID = 32562, status = 0
Child process exited with code 0
Child process has finished: PID = 32563, status = 0
Child process exited with code 0
^CReceived message: 2egassem1egassem
Parent process is dead now

```

Рисунок 6 – Демонстрация работы программы (сигнал вызывается)