



НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

## по лабораторной работе № 2

Дисциплина: Операционные системы

Преподаватель	<hr/>	Н.Ю. Рязанова
	(Подпись, дата)	(И.О. Фамилия)

Москва, 2020

# Задание 1

Код программы:

```
12#include <stdio.h>
13#include <unistd.h>
14
15int main(void)
16{
17    pid_t childPID1, childPID2;
18    childPID1 = fork();
19
20    if (childPID1 == -1)
21    {
22        perror("Can't fork\n");
23        return 1;
24    }
25    else if (childPID1)
26    {
27        childPID2 = fork();
28        if (childPID2 == -1)
29        {
30            perror("Can't fork\n");
31            return 1;
32        }
33        else if (childPID2)
34        {
35            printf("Parent:\t\t\t\tPID=%d, PGRP=%d, CHILD1_PID=%d, CHILD2_PID=%d \n",
36                getpid(), getpgrp(), childPID1, childPID2);
37        }
38        else
39        {
40            pid_t PPID = getppid();
41            printf("Child2:\t\t\t\tPID=%d, PGRP=%d, PARENT_PID=%d \n",
42                getpid(), getpgrp(), getppid());
43
44            while(PPID == getppid()) {}
45
46            printf("Child2 after parent exit:\tPID=%d, PGRP=%d, PARENT_PID=%d \n",
47                getpid(), getpgrp(), getppid());
48        }
49    }
50    else
51    {
52        pid_t PPID = getppid();
53        printf("Child1:\t\t\t\tPID=%d, PGRP=%d, PARENT_PID=%d \n",
54            getpid(), getpgrp(), getppid());
55
56        while(PPID == getppid()) {}
57
58        printf("Child1 after parent exit:\tPID=%d, PGRP=%d, PARENT_PID=%d \n",
59            getpid(), getpgrp(), getppid());
60    }
61
62    return 0;
63}
```

Пример работы:

```
vsevolod@vsevolod-HP-Pavilion14:~/work/OS_bmstu/lab2$ ./prog1.o
Parent: PID=7035, PGRP=7035, CHILD1_PID=7036, CHILD2_PID=7037
Child1: PID=7036, PGRP=7035, PARENT_PID=7035
Child2: PID=7037, PGRP=7035, PARENT_PID=7035
Child1 after parent exit: PID=7036, PGRP=7035, PARENT_PID=2070
Child2 after parent exit: PID=7037, PGRP=7035, PARENT_PID=2070
```

## Задание 2

Код программы:

```
5 #include <stdio.h>
6 #include <unistd.h>
7 #include <sys/types.h>
8 #include <sys/wait.h>
9
10 int main(void)
11 {
12     pid_t childPID1, childPID2;
13     childPID1 = fork();
14
15     if (childPID1 == -1)
16     {
17         perror("Can't fork\n");
18         return 1;
19     }
20     else if (childPID1)
21     {
22         childPID2 = fork();
23         if (childPID2 == -1)
24         {
25             perror("Can't fork\n");
26             return 1;
27         }
28         else if (childPID2)
29         {
30             printf("Parent:\t\t\tPID=%d, PGRP=%d, CHILD1_PID=%d, CHILD2_PID=%d \n",
31                 getpid(), getpgrp(), childPID1, childPID2);
32             for (int i=0; i<2; i++)
33             {
34                 int stat;
35                 pid_t child;
36                 child = wait(&stat);
37                 if (child == -1)
38                 {
39                     printf("Wait returned with error\n");
40                     return 1;
41                 }
42                 else
43                 {
44                     printf("%d finished, status=%d. Parent PID:%d\t\t\t", child, stat, getpid());
45                     if (WIFEXITED(stat))
46                         printf("Child exited with code %d\n", WEXITSTATUS(stat));
47                     else
48                         printf("Child terminated abnormally\n");
49                 }
50             }
51         }
52     }
53     else
54     {
55         printf("Child2:\t\t\tPID=%d, PGRP=%d, PARENT_PID=%d \n",
56             getpid(), getpgrp(), getppid());
57     }
58 }
59
60 {
61     printf("Child1:\t\t\tPID=%d, PGRP=%d, PARENT_PID=%d \n",
62         getpid(), getpgrp(), getppid());
63 }
64
65 return 0;
66 }
```

Пример работы:

```
vsevolod@vsevolod-HP-Pavilion14:~/work/OS_bmstu/lab2$ ./wait.o
Parent: PID=5332, PGRP=5332, CHILD1_PID=5333, CHILD2_PID=5334
Child1: PID=5333, PGRP=5332, PARENT_PID=5332
Child2: PID=5334, PGRP=5332, PARENT_PID=5332
5333 finished, status=0. Parent PID:5332 Child exited with code 0
5334 finished, status=0. Parent PID:5332 Child exited with code 0
```

## Задание 3

Код программы:

```
5 #include <stdio.h>
6 #include <unistd.h>
7 #include <sys/types.h>
8 #include <sys/wait.h>
9
10 int main(void)
11 {
12     pid_t childPID1, childPID2;
13     childPID1 = fork();
14
15     if (childPID1 == -1)
16     {
17         perror("Can't fork\n");
18         return 1;
19     }
20     else if (childPID1)
21     {
22         childPID2 = fork();
23         if (childPID2 == -1)
24         {
25             perror("Can't fork\n");
26             return 1;
27         }
28         else if (childPID2)
29         {
30             printf("Parent:\t\t\t\tPID=%d, PGRP=%d, CHILD1_PID=%d, CHILD2_PID=%d \n",
31                 getpid(), getpgrp(), childPID1, childPID2);
32
33             for (int i=0; i<2; i++)
34             {
35                 int stat;
36                 pid_t child;
37                 child = wait(&stat);
38                 if (child == -1)
39                 {
40                     printf("Wait returned with error\n");
41                     return 1;
42                 }
43                 else
44                 {
45                     printf("%d finished, status=%d. Parent PID:%d\t\t\t", child, stat, getpid());
46                     if (WIFEXITED(stat))
47                         printf("Child exited with code %d\n", WEXITSTATUS(stat));
48                     else
49                         printf("Child terminated abnormally\n");
50                 }
51             }
52         }
53     }
54     else
55     {
56         printf("Child2:\t\t\t\tPID=%d, PGRP=%d, PARENT_PID=%d \n",
57             getpid(), getpgrp(), getppid());
58         int stat = execl("demo2.o", " ", NULL);
59         if (stat == -1)
60         {
61             printf("Execl error\n");
62             return 1;
63         }
64     }
65     else
66     {
67         printf("Child1:\t\t\t\tPID=%d, PGRP=%d, PARENT_PID=%d \n",
68             getpid(), getpgrp(), getppid());
69         int stat = execl("demo1.o", " ", NULL);
70         if (stat == -1)
71         {
72             printf("Execl error\n");
73             return 1;
74         }
75     }
76
77     return 0;
78 }
79
```

Запускаемые программы:

[illegible]

### Пример работы:

```
vsevolod@vsevolod-HP-Pavilion14:~/work/OS_bmstu/lab2$ ./prog3.o
Parent:                PID=3867, PGRP=3867, CHILD1_PID=3868, CHILD2_PID=3869
Child2:                PID=3869, PGRP=3867, PARENT_PID=3867
Child1:                PID=3868, PGRP=3867, PARENT_PID=3867
>>>>>>>>>>>>>>>>
Second program message
>>>>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<
First program message
<<<<<<<<<<<<<<<<<
3869 finished, status=0. Parent PID:3867           Child exited with code 0
3868 finished, status=0. Parent PID:3867           Child exited with code 0
```

## Задание 4

Код программы:

```
5 #include <stdio.h>
6 #include <unistd.h>
7 #include <sys/types.h>
8 #include <sys/wait.h>
9
10 #define MSG_SIZE 21
11
12 int main(void)
13 {
14     int fd[2];
15     pid_t childPID1, childPID2;
16
17     int pipe_code = pipe(fd);
18     if (pipe_code < 0)
19     {
20         perror("Can't create pipe\n");
21         return -1;
22     }
23
24     childPID1 = fork();
25     if (childPID1 == -1)
26     {
27         perror("Can't fork\n");
28         return -1;
29     }
30     else if (!childPID1)
31     {
32         printf("Child 1:\t\t\tPID=%d, PGRP=%d, PARENT_PID=%d \n", getpid(), getpgrp(), getppid());
33
34         ssize_t code = write(fd[1], "Data from child №1", MSG_SIZE);
35         if (code == -1)
36         {
37             printf("Write error\n");
38             return -1;
39         }
40         else
41             printf("Child 1 sent message\n");
42
43         if(close(fd[1]) || close(fd[0]))
44         {
45             printf("Close error\n");
46             return -1;
47         }
48         else
49             return 0;
50     }
51 }
52
```

```

53 childPID2 = fork();
54 if (childPID2 == 0)
55 {
56     printf("Child 2:\t\t\tPID=%d, PGRP=%d, PARENT_PID=%d \n", getpid(), getpgrp(), getppid());
57
58     ssize_t code = write(fd[1], "Data from child #2", MSG_SIZE);
59     if (code == -1)
60     {
61         printf("Write error\n");
62         return -1;
63     }
64     else
65         printf("Child 1 sent message\n");
66
67     if (close(fd[1]) || close(fd[0]))
68     {
69         printf("Close error\n");
70         return -1;
71     }
72     else
73         return 0;
74 }
75 else if (childPID2 == -1)
76 {
77     perror("Can't fork\n");
78     return -1;
79 }
80
81
82 printf("Parent:\t\t\t\tPID=%d, PGRP=%d, CHILD1_PID=%d, CHILD2_PID=%d \n",
83       getpid(), getpgrp(), childPID1, childPID2);
84
85 printf("Parent is waiting\n");
86 int stat, child;
87 for (int i=0; i<2; i++)
88 {
89     child = wait(&stat);
90     if (child == -1)
91     {
92         printf("Wait returned with error\n");
93         return -1;
94     }
95     else
96     {
97         printf("%d finished, status=%d. Parent PID:%d\t\t\t", child, stat, getpid());
98         if (WIFEXITED(stat))
99             printf("Child exited with code %d\n", WEXITSTATUS(stat));
100         else
101             printf("Child terminated abnormally\n");
102     }
103 }
104 printf("All child process are done\n");
105
106 char res1[MSG_SIZE], res2[MSG_SIZE];
107 ssize_t code1 = read(fd[0], res1, MSG_SIZE);
108 ssize_t code2 = read(fd[0], res2, MSG_SIZE);
109 if (code1 == -1 || code2 == -1)
110 {
111     printf("Read error\n");
112     return -1;
113 }
114 printf("First message: %s\n", res1);
115 printf("Second message: %s\n", res2);
116
117 if (close(fd[1]) || close(fd[0]))
118 {
119     printf("Close error\n");
120     return -1;
121 }
122 else
123     return 0;
124 }
125

```



Пример работы:

```
vsevolod@vsevolod-HP-Pavilion14:~/work/OS_bmstu/lab2$ ./prog4.o
Parent: PID=6187, PGRP=6187, CHILD1_PID=6188, CHILD2_PID=6189
Parent is waiting
Child 1: PID=6188, PGRP=6187, PARENT_PID=6187
Child 1 sent message
Child 2: PID=6189, PGRP=6187, PARENT_PID=6187
Child 2 sent message
6188 finished, status=0. Parent PID:6187 Child exited with code 0
6189 finished, status=0. Parent PID:6187 Child exited with code 0
All child process are done
First message: Data from child №1
Second message: Data from child №2
```



## Задание 5

### Код программы:

```
6 #include <stdio.h>
7 #include <unistd.h>
8 #include <sys/types.h>
9 #include <sys/wait.h>
10
11 #define MSG_SIZE 21
12 typedef void (*sighandler_t)(int);
13
14 int send_flag = 0;
15
16 void send_msg(int sig_n)
17 {
18     send_flag = 1;
19 }
20
21 int main(void)
22 {
23     sighandler_t sg = signal(SIGINT, send_msg);
24     if (sg == SIG_ERR)
25     {
26         perror("Can't create signal\n");
27         return -1;
28     }
29
30     int fd[2];
31     pid_t childPID1, childPID2;
32
33     int pipe_code = pipe(fd);
34     if (pipe_code < 0)
35     {
36         perror("Can't create pipe\n");
37         return -1;
38     }
39
40     childPID1 = fork();
41     if (childPID1 == -1)
42     {
43         perror("Can't fork\n");
44         return -1;
45     }
46     else if (!childPID1)
47     {
48         printf("Child 1:\t\t\tPID=%d, PGRP=%d, PARENT_PID=%d \n", getpid(), getpgrp(), getppid());
49
50         ssize_t code = write(fd[1], "Data from child №1", MSG_SIZE);
51         if (code == -1)
52         {
53             printf("Write error\n");
54             return -1;
55         }
56     }
```

```

57     else
58         printf("Child 1 sent message\n");
59
60     sleep(4);
61
62     if (send_flag)
63         printf("\nChild 1: Ctrl-C flag was setted\n");
64     else
65         printf("\nChild 1: Ctrl-C flag was not setted\n");
66
67     if (close(fd[1]) || close(fd[0]))
68     {
69         printf("Close error\n");
70         return -1;
71     }
72     else
73         return 0;
74 }
75
76
77 childPID2 = fork();
78 if (childPID2 == 0)
79 {
80     printf("Child 2:\t\t\tPID=%d, PGRP=%d, PARENT_PID=%d \n", getpid(), getpgrp(), getppid());
81
82     ssize_t code = write(fd[1], "Data from child №2", MSG_SIZE);
83     if (code == -1)
84     {
85         printf("Write error\n");
86         return -1;
87     }
88     else
89         printf("Child 2 sent message\n");
90
91     if (close(fd[1]) || close(fd[0]))
92     {
93         printf("Close error\n");
94         return -1;
95     }
96     else
97         return 0;
98 }
99
100 else if (childPID2 == -1)
101 {
102     perror("Can't fork\n");
103     return -1;
104 }
105
106 printf("Parent:\t\t\t\tPID=%d, PGRP=%d, CHILD1_PID=%d, CHILD2_PID=%d \n", getpid(), getpgrp(), childPID1, childPID2);
107
108
109 printf("Parent is waiting\n");
110 int stat, child;
111 for (int i=0; i<2; i++)
112 {
113     child = wait(&stat);
114     if (child == -1)
115     {
116         printf("Wait returned with error\n");
117         return -1;
118     }
119     else
120     {
121         printf("%d finished, status=%d. Parent PID:%d\t\t\t", child, stat, getpid());
122         if (WIFEXITED(stat))
123             printf("Child exited with code %d\n", WEXITSTATUS(stat));
124         else
125             printf("Child terminated abnormally\n");
126     }
127 }
128 printf("All child process are done\n");
129
130 char res1[MSG_SIZE], res2[MSG_SIZE];
131 ssize_t code1 = read(fd[0], res1, MSG_SIZE);
132 ssize_t code2 = read(fd[0], res2, MSG_SIZE);
133 if (code1 == -1 || code2 == -1)
134 {
135     printf("Read error\n");
136     return -1;
137 }
138 printf("First message: %s\n", res1);
139 printf("Second message: %s\n", res2);
140
141 if (close(fd[1]) || close(fd[0]))
142 {
143     printf("Close error\n");
144     return -1;
145 }
146 else
147     return 0;
148 }

```

### Примеры работы:

```
vsevolod@vsevolod-HP-Pavilion14:~/work/OS_bmstu/lab2$ ./prog5.o
Parent: PID=3085, PGRP=3085, CHILD1_PID=3086, CHILD2_PID=3087
Parent is waiting
Child 1: PID=3086, PGRP=3085, PARENT_PID=3085
Child 1 sent message
Child 2: PID=3087, PGRP=3085, PARENT_PID=3085
Child 2 sent message
3087 finished, status=0. Parent PID:3085 Child exited with code 0
^C
Child 1: Ctrl-C flag was setted
3086 finished, status=0. Parent PID:3085 Child exited with code 0
All child process are done
First message: Data from child №1
Second message: Data from child №2
vsevolod@vsevolod-HP-Pavilion14:~/work/OS_bmstu/lab2$ ./prog5.o
Parent: PID=3090, PGRP=3090, CHILD1_PID=3091, CHILD2_PID=3092
Parent is waiting
Child 1: PID=3091, PGRP=3090, PARENT_PID=3090
Child 1 sent message
Child 2: PID=3092, PGRP=3090, PARENT_PID=3090
Child 2 sent message
3092 finished, status=0. Parent PID:3090 Child exited with code 0

Child 1: Ctrl-C flag was not setted
3091 finished, status=0. Parent PID:3090 Child exited with code 0
All child process are done
First message: Data from child №1
Second message: Data from child №2
```