



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 Богаченко А.Е.

Группа ИУ7-56Б

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 – Рекурсивный

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

```
(root👤NebuchadnezzaR) - [~/bmstu-os/lab4/src]
# ./a.out
Parent process: PID=243926, GROUP=243926
Parent process have children with IDs: 243927, 243928
Parent process is dead now

(root👤NebuchadnezzaR) - [~/bmstu-os/lab4/src]
# Child process : PID=243928, GROUP=243926, PPID=923
Child process : PID=243927, GROUP=243926, PPID=923
```

Рисунок 1 – Демонстрация работы

Задание 2

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

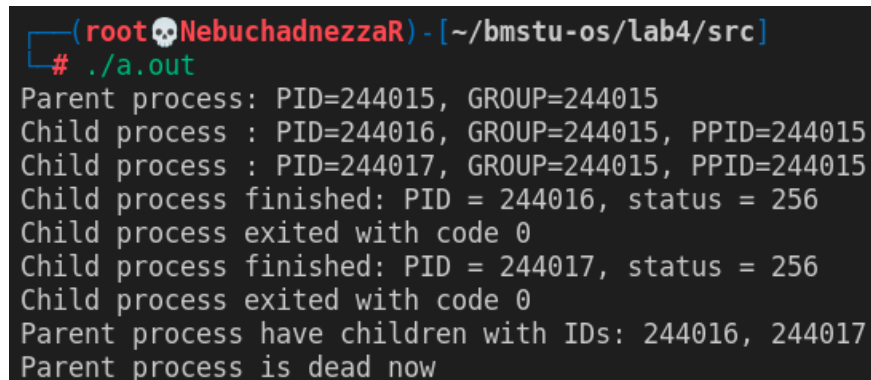
Листинг 2 – `wait()`

```
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 pid;
8 int child_pids[2];
9
10 int main(void) {
11     printf("Parent process: PID=%d, GROUP=%d\n", getpid(), getpgrp());
12
13     for (size_t i = 0; i < 2; ++i) {
14         switch (pid = fork()) {
15             case -1:
16                 perror("Can't fork\n");
17                 return EXIT_FAILURE;
18             case 0:
19                 sleep(2);
20                 printf("Child process: PID=%d, GROUP=%d, PPID=%d\n", getpid(),
21                     getpgrp(), getppid());
22                 return EXIT_FAILURE;
23             default:
24                 child_pids[i] = pid;
25         }
26     }
27
28     for (size_t i = 0; i < 2; ++i) {
```

```

29     int status;
30     pid_t childpid = wait(&status);
31     printf("Child process finished: PID=%d, status=%d\n", childpid, status);
32
33     int stat_val;
34     if (WIFEXITED(stat_val)) {
35         printf("Child process exited with code %d\n",
36             WEXITSTATUS(stat_val));
37     } else {
38         printf("Child process terminated abnormally\n");
39     }
40 }
41
42 printf("Parent process have children with IDs: %d, %d\n", child_pids[0],
43     child_pids[1]);
44 printf("Parent process is dead now\n");
45
46 return EXIT_SUCCESS;
47 }

```



```

(root NebuchadnezzaR) - [~/bmstu-os/lab4/src]
# ./a.out
Parent process: PID=244015, GROUP=244015
Child process : PID=244016, GROUP=244015, PPID=244015
Child process : PID=244017, GROUP=244015, PPID=244015
Child process finished: PID = 244016, status = 256
Child process exited with code 0
Child process finished: PID = 244017, status = 256
Child process exited with code 0
Parent process have children with IDs: 244016, 244017
Parent process is dead now

```

Рисунок 2 – Демонстрация работы

Задание 3

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

Листинг 3 – `execlp()`

```

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 pid;
8 int child_pids[2];
9 const char volatile *const cmd[2] = {"ls", "whoami"};
10
11 int main() {
12     printf("Parent_process: PID=%d, GROUP=%d\n", getpid(), getpgrp());
13
14     for (size_t i = 0; i < 2; ++i) {
15         switch (pid = fork()) {
16             case -1:
17                 perror("Can't fork\n");
18                 return EXIT_FAILURE;
19             case 0:
20                 printf("Child_process: PID=%d, GROUP=%d, PPID=%d\n\n",
21                     getpid(), getpgrp(), getppid());
22
23                 switch (execlp(cmd[i], cmd[i], 0)) {
24                     case -1:
25                         perror("Can't exec\n");
26                         return EXIT_FAILURE;
27                     case 0:
28                         return EXIT_SUCCESS;
29                 }
30             default:
31                 child_pids[i] = pid;
32         }
33     }
34
35     for (size_t i = 0; i < 2; ++i) {
36         int status;
37         pid_t childpid = wait(&status);
38         printf("\nChild_process_finished: PID=%d, status=%d\n", childpid, status);
39         int stat_val;
40         if (WIFEXITED(stat_val)) {
41             printf("Child_process_exited_with_code%d\n",
42                 WEXITSTATUS(stat_val));
43         } else {
44             printf("Child_process_terminated_abnormally\n");
45         }
46     }
47
48     printf("Parent_process_have_children_with_IDs: %d, %d\n", child_pids[0],
49         child_pids[1]);
50     printf("Parent_process_is_dead_now\n");
51
52     return EXIT_SUCCESS;
53 }

```

```

(root👁NebuchadnezzaR) - [~/bmstu-os/lab4/src]
# ./a.out
Parent process: PID=244246, GROUP=244246
Child process : PID=244247, GROUP=244246, PPID=244246

Child process : PID=244248, GROUP=244246, PPID=244246

root

Child process finished: PID = 244248, status = 0
Child process exited with code 0
1.c 2.c 3.c 4.c 5.c a.out

Child process finished: PID = 244247, status = 0
Child process exited with code 0
Parent process have children with IDs: 244247, 244248
Parent process is dead now

```

Рисунок 3 – Демонстрация работы

Задание 4

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

Листинг 4 – pipe

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <sys/types.h>
5  #include <sys/wait.h>
6  #include <unistd.h>
7
8  int pid;
9  int child_pids[2];
10 const char volatile *msg[] = {"message1", "message2"};
11
12 int main() {
13     int fd[2];
14     char buffer[50] = {0};
15
16     printf("Parent process: PID=%d, GROUP=%d\n", getpid(), getpgrp());
17     if (pipe(fd) == -1) {
18         perror("Can't pipe\n");
19         return EXIT_FAILURE;
20     }
21

```

```

22     for (size_t i = 0; i < 2; ++i) {
23         switch (pid = fork()) {
24             case -1:
25                 perror("Can't fork\n");
26                 exit(EXIT_FAILURE);
27             case 0:
28                 close(fd[0]);
29                 write(fd[1], msg[i], strlen(msg[i]));
30                 printf("Message has been sent to parent\n");
31                 exit(EXIT_SUCCESS);
32             default:
33                 child_pids[i] = pid;
34         }
35     }
36
37     for (size_t i = 0; i < 2; ++i) {
38         int status;
39         pid_t childpid = wait(&status);
40         printf("Child process finished: PID=%d, status=%d\n", childpid, status);
41
42         int stat_val;
43         if (WIFEXITED(stat_val)) {
44             printf("Child process exited with code %d\n",
45                 WEXITSTATUS(stat_val));
46         } else {
47             printf("Child process terminated abnormally\n");
48         }
49     }
50
51     close(fd[1]);
52     read(fd[0], buffer, sizeof(buffer));
53     printf("Received message: %s\n", buffer);
54
55     printf("Parent process have children with IDs: %d, %d\n", child_pids[0],
56         child_pids[1]);
57     printf("Parent process is dead now\n");
58
59     return EXIT_SUCCESS;
60 }

```

```
(root👤NebuchadnezzaR) - [~/bmstu-os/lab4/src]
# ./a.out
Parent process: PID=244331, GROUP=244331
Message has been sent to parent
Child process finished: PID = 244333, status = 0
Child process exited with code 0
Message has been sent to parent
Child process finished: PID = 244332, status = 0
Child process exited with code 0
Received message: message2message1
Parent process have children with IDs: 244332, 244333
Parent process is dead now
```

Рисунок 4 – Демонстрация работы

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 – Сигналы

```
1 #include <signal.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <string.h>
5 #include <sys/types.h>
6 #include <sys/wait.h>
7 #include <unistd.h>
8
9 int pid;
10 int child_pids[2];
11 const char volatile *msg[] = {"message1", "message2"};
12
13 int state = 0;
14
15 void ignore_sigint(int sig) {}
16
17 void is_writing(int sig) { state = 1; }
```



```

18
19 int main() {
20     int fd[2];
21     char buffer[50] = {0};
22
23     printf("Parent_process:PID=%d,GROUP=%d\n", getpid(), getpgrp());
24
25     if (pipe(fd) == -1) {
26         perror("Can't pipe\n");
27         return EXIT_FAILURE;
28     }
29
30     signal(SIGINT, ignore_sigint);
31     for (size_t i = 0; i < 2; ++i) {
32         switch (pid = fork()) {
33             case -1:
34                 perror("Can't fork\n");
35                 exit(EXIT_FAILURE);
36             case 0:
37                 signal(SIGINT, is_writing);
38                 sleep(5);
39                 if (state) {
40                     close(fd[0]);
41                     write(fd[1], msg[i], strlen(msg[i]));
42                     printf("Message_has_been_sent_to_parent\n");
43                 } else {
44                     printf("No_signal_sent,writing_will_not_be_completed\n");
45                 }
46
47                 exit(EXIT_SUCCESS);
48             default:
49                 child_pids[i] = pid;
50         }
51     }
52
53     for (size_t i = 0; i < 2; ++i) {
54         int status;
55         pid_t childpid = wait(&status);
56         printf("Child_process_finished:PID=%d,status=%d\n", childpid, status);
57
58         int stat_val;
59         if (WIFEXITED(stat_val)) {
60             printf("Child_process_exited_with_code%d\n",
61                    WEXITSTATUS(stat_val));
62         } else {
63             printf("Child_process_terminated_abnormally\n");
64         }
65     }

```

```

66
67     close(fd[1]);
68     read(fd[0], buffer, sizeof(buffer));
69
70     printf("Received message: %s\n", buffer);
71     printf("Parent process have children with IDs: %d, %d\n", child_pids[0],
72           child_pids[1]);
73     printf("Parent process is dead now\n");
74
75     return EXIT_SUCCESS;
76 }

```

```

(root👤NebuchadnezzaR) - [~/bmstu-os/lab4/src]
# ./a.out
Parent process: PID=244426, GROUP=244426
No signal sent, writing will not be completed
No signal sent, writing will not be completed
Child process finished: PID = 244427, status = 0
Child process exited with code 0
Child process finished: PID = 244428, status = 0
Child process exited with code 0
Received message:
Parent process have children with IDs: 244427, 244428
Parent process is dead now

```

Рисунок 5 – Демонстрация работы (без сигнала)

```

(root👤NebuchadnezzaR) - [~/bmstu-os/lab4/src]
# ./a.out
Parent process: PID=244502, GROUP=244502
^CMessage has been sent to parent
Message has been sent to parent
Child process finished: PID = 244503, status = 0
Child process exited with code 0
Child process finished: PID = 244504, status = 0
Child process exited with code 0
Received message: message1message2
Parent process have children with IDs: 244503, 244504
Parent process is dead now

```

Рисунок 6 – Демонстрация работы (с сигналом)