

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 <u>Процессы. Системные вызовы fork() и exec()</u>
Студент Артемьев И. О.
Группа <u>ИУ7-53Б</u>
Оценка (баллы)
Преподаватели Рязанова Н. Ю.

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

```
| #include < stdio.h>
2 #include <unistd.h>
  int main(void)
  {
      int childpid first = fork(), childpid second;
      if (childpid_first == -1)
9
           perror("Can't fork\n");
10
           return -1;
11
12
      else if (childpid first == 0)
13
14
           sleep(1);
15
           printf("\nFirst child: id: %d ppid: %d pgroup: %d\n", getpid(),
16
              getppid(), getpgrp());
           return 0;
17
      }
18
19
      childpid second = fork();
20
      if (childpid second == -1)
21
22
           perror("Can't fork\n");
23
           return -1;
24
25
      else if (childpid_second == 0)
26
27
           sleep(1);
28
           printf("\nSecond child: id: %d ppid: %d pgroup: %d\n\n", getpid(),
29
              getppid(), getpgrp());
           return 0;
30
      }
31
32
      printf("Parent: id: %d pgroup: %d first child: %d second child: %d\n",
33
          getpid(), getpgrp(), childpid first, childpid second);
34
      return 0;
35
36 }
```

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

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

Листинг 2: Вызов функции wait()

```
#include "status.h"
  int main(void)
3
  {
4
      int status;
5
      int childpid first = fork(), childpid second;
      if (childpid first == -1)
      {
           perror("Can't fork\n");
10
           return -1;
11
12
      else if (childpid first == 0)
13
           printf("\nFirst child: id: %d ppid: %d pgroup: %d\n", getpid(),
16
              getppid(), getpgrp());
           return 0;
17
      }
18
19
      childpid second = fork();
20
      if (childpid\_second == -1)
21
22
           perror("Can't fork\n");
23
           return -1;
24
25
      else if (childpid second = 0)
26
27
28
           printf("\nSecond child: id: %d ppid: %d pgroup: %d\n\n", getpid(),
29
              getppid(), getpgrp());
           return 0;
30
      }
31
32
      wait(&status);
33
      check status(status);
34
      wait(&status);
35
      check status(status);
37
      printf("Parent: id: %d pgroup: %d first child: %d second child: %d\n",
38
          getpid(), getpgrp(), childpid first, childpid second);
39
      return 0;
40
41 }
```

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

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

Листинг 3: Вызов функции execlp()

```
#include "status.h"
  int main(void)
3
  {
4
      int status;
      int childpid first = fork(), childpid second;
      if (childpid first == -1)
           perror("Can't fork\n");
10
           return -1;
11
12
      else if (childpid first == 0)
13
           printf("\nFirst child: id: %d ppid: %d pgroup: %d\n", getpid(),
15
              getppid(), getpgrp());
16
           if (execlp("./levenstein.exe", "./levenstein.exe", NULL) == -1)
17
18
               perror("Can't execlp first child\n\n");
19
               return -1;
20
          }
21
           return 0;
23
      }
24
25
      childpid second = fork();
26
      if (childpid second == -1)
27
      {
28
           perror("Can't fork\n");
29
           return -1;
30
31
      else if (childpid second = 0)
32
33
           printf("\nSecond child: id: %d ppid: %d pgroup: %d\n\n", getpid(),
34
              getppid(), getpgrp());
35
           sleep(8);
           if (execlp("./sort.exe", "./sort.exe", NULL) == -1)
37
           {
38
               perror("Can't execlp second child\n\n");
39
               return -1;
40
          }
41
42
```

```
return 0;
43
      }
44
45
      wait(&status);
46
      check status(status);
47
      wait(&status);
48
      check status(status);
49
50
      printf("Parent: id: %d pgroup: %d first child: %d second child: %d\n",
51
          getpid(), getpgrp(), childpid first, childpid second);
52
      return 0;
53
54 }
```

```
ilyaartemev@Air-Ilya > ~/Desktop/IU7-OS/lab_04/src > / master • ? > ./case 3.exe
First child: id: 16930 ppid: 16929 pgroup: 16929
Second child: id: 16931 ppid: 16929 pgroup: 16929
Введите первую строку: aba
Введите вторую строку: ba
Редакционное расстояние: 1
Дочерний процесс завершен нормально
Код завершения дочернего процесса: 0
Введите количество элементов массива: 4
Введите элементы массива: 0 4 1 -43
Отсортированный массив: -43 0 1 4
Дочерний процесс завершен нормально
Код завершения дочернего процесса: 0
Parent: id: 16929 pgroup: 16929 first child: 16930 second child: 16931
 🕨 🖢 master 🎍 ?
```

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

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

Листинг 4: Использование ріре

```
#include "status.h"
  int main(void)
  {
4
      const char *msg 1 = "Overleaf" is used by over eight million students and
           academics at 3,600 institutions worldwide.\n";
      const char *msg 2 = "Just want to say that I am really grateful for
          Overleaf, it has enabled a slew of research and teaching development
         in my work that would have been annoyingly difficult before. Even
          people who dont know LaTeX are participating with me on research
          proposals and thats saying something if you know what LaTeX is
         like for the uninitiated.\n";
      char readbuffer[1000];
      int fd[2];
10
      int status:
11
      int childpid first , childpid second;
12
13
      if (pipe(fd) = -1)
14
      {
15
           printf("1\n");
          perror("Can't pipe\n");
17
          return -1;
18
      }
19
20
      childpid first = fork();
21
      if (childpid first == -1)
22
      {
23
          perror("Cant fork.\n");
24
          return -1;
26
      else if (childpid first == 0)
27
28
           close (fd [0]);
29
          write (fd[1], msg 1, strlen(msg 1));
30
31
          return 0;
32
      }
33
34
      childpid second = fork();
35
      if (childpid second == -1)
36
      {
37
           perror("Can't fork\n");
38
```

```
return -1;
39
40
      else if (childpid second = 0)
41
42
           close (fd [0]);
43
           write (fd[1], msg 2, strlen(msg 2));
44
45
           return 0;
46
      }
47
48
      wait(&status);
49
      check status(status);
50
      wait(&status);
51
      check status(status);
52
53
      printf("Parent: id: %d pgroup: %d first child: %d second child: %d\n\n",
54
           getpid(), getpgrp(), childpid_first, childpid_second);
55
      close (fd[1]);
56
      read(fd[0], readbuffer, sizeof(readbuffer));
57
      printf("%s", readbuffer);
      return 0;
60
61
  }
```

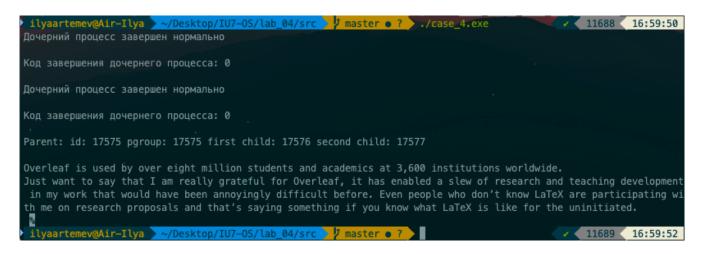


Рис. 4: Демонстрация работы программы (задание №4).

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

```
#include "status.h"
  int flag = 0;
  void ctrl c(const int signal numb)
      flag = 1;
      printf("\nSignal number: %d\n", signal numb);
  }
10
  void ctrl z(const int signal numb)
11
  {
12
      flag = 0;
13
      printf("\nSignal number: %d\n", signal numb);
15
16
  int main(void)
17
  {
18
      const char *msg_1 = "Overleaf" is used by over eight million students and
19
          academics at 3,600 institutions worldwide.\n";
      const char *msg 2 = "Just want to say that I am really grateful for
20
         Overleaf, it has enabled a slew of research and teaching development
         in my work that would have been annoyingly difficult before. Even
         people who dont know LaTeX are participating with me on research
         proposals and thats saying something if you know what LaTeX is
         like for the uninitiated.\n";
21
      char readbuffer[1000];
22
      int fd[2];
      int status:
25
      int childpid first , childpid second;
26
27
      signal(SIGINT, ctrl c);
28
      signal(SIGTSTP, ctrl z);
29
      if (pipe(fd) = -1)
31
32
          printf("1\n");
33
          perror("Can't pipe\n");
34
          return -1;
35
      }
36
37
```

```
childpid first = fork();
38
       if (childpid first == -1)
39
40
           perror("Cant fork.\n");
41
           return -1;
42
43
       else if (childpid first == 0)
44
45
           close (fd [0]);
46
           write (fd[1], msg 1, strlen(msg 1));
47
48
           return 0;
49
      }
50
51
       childpid second = fork();
52
       if (childpid second == -1)
53
54
           perror("Can't fork\n");
55
           return -1;
56
57
       else if (childpid second = 0)
58
59
           close (fd [0]);
60
           write(fd[1], msg_2, strlen(msg_2));
61
62
           return 0;
63
      }
64
65
       wait(&status);
66
       check status(status);
67
       wait(&status);
68
       check status(status);
69
70
       printf("Parent: id: %d pgroup: %d first child: %d second child: %d\n\n",
71
           getpid(), getpgrp(), childpid first, childpid second);
72
       printf("To show a messages from childs, press: \TCTRL+C\T.\n");
73
       printf("To not show messages from childs, press: \TCTRL+Z\T.\n\n");
74
       sleep (1000);
75
76
       if (flag == 1)
77
       {
78
           close (fd[1]);
79
           read(fd[0], readbuffer, sizeof(readbuffer));
80
           printf("%s", readbuffer);
81
      }
82
83
       return 0;
84
85 }
```

```
ilyaartemev@Air—Ilya  ~/Desktop/IU7—OS/lab_04/src  pmaster  ? ./case_5.exe
Дочерний процесс завершен нормально

Код завершения дочернего процесса: 0

Дочерний процесс завершен нормально

Код завершения дочернего процесса: 0

Parent: id: 18136 pgroup: 18136 first child: 18137 second child: 18138

To show a messages from childs, press: "CTRL+C".

To not show messages from childs, press: "CTRL+Z".

^Z

Signal M: 18

ilyaartemev@Air—Ilya  ~/Desktop/IU7—OS/lab_04/src  pmaster  ? pmaster  pmaster  ? pmaster  pmaster
```

Рис. 5: Демонстрация работы программы, сигнал не вызывается (задание №5).



Рис. 6: Демонстрация работы программы, сигнал вызывается (задание №5).

Дополнительные файлы

Листинг 6: Проверка статуса завершившегося дочернего процесса

```
|#include "status.h"
  void check status(const int status)
  {
      if (WIFEXITED(status))
           printf("Child process completed successfully \n\n");
           printf("Return code: %d\n\n", WEXITSTATUS(status));
      else if (WIFSIGNALED(status))
11
           printf("The child process terminates with an un-intercepted signal\n
12
              \n");
           printf("Signal number: %d\n\n", WTERMSIG(status));
13
14
      else if (WIFSTOPPED(status))
           printf("The child process has stopped\n\n");
17
           printf("Signal number: %d\n\n", WSTOPSIG(status));
18
      }
19
20
      return;
^{21}
22 }
```

Листинг 7: Хедер для подключения к файлам вызывающим check status

```
#ifndef __STATUS_H__
#define __STATUS_H__

#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/wait.h>

void check_status(const int status);

#endif // __STATUS_H__
```

Дополнительное задание

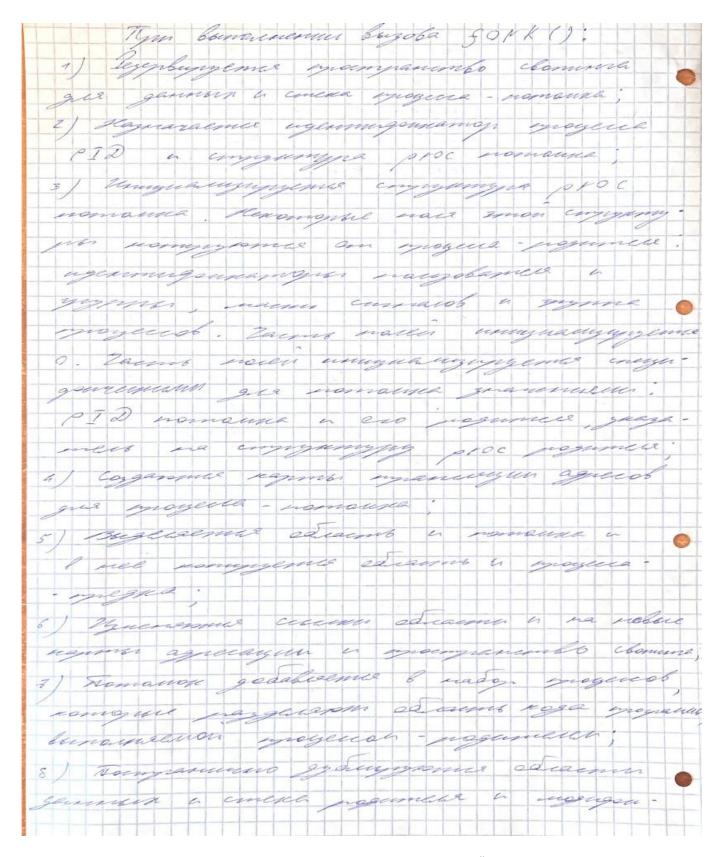


Рис. 7: Очередность при вызове fork().

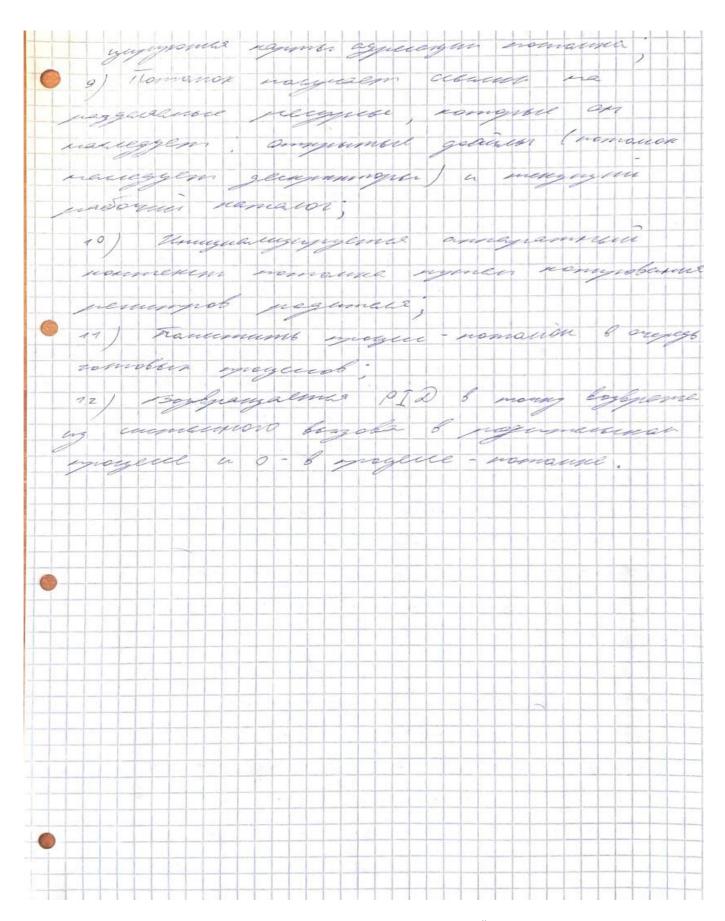


Рис. 8: Очередность при вызове fork().

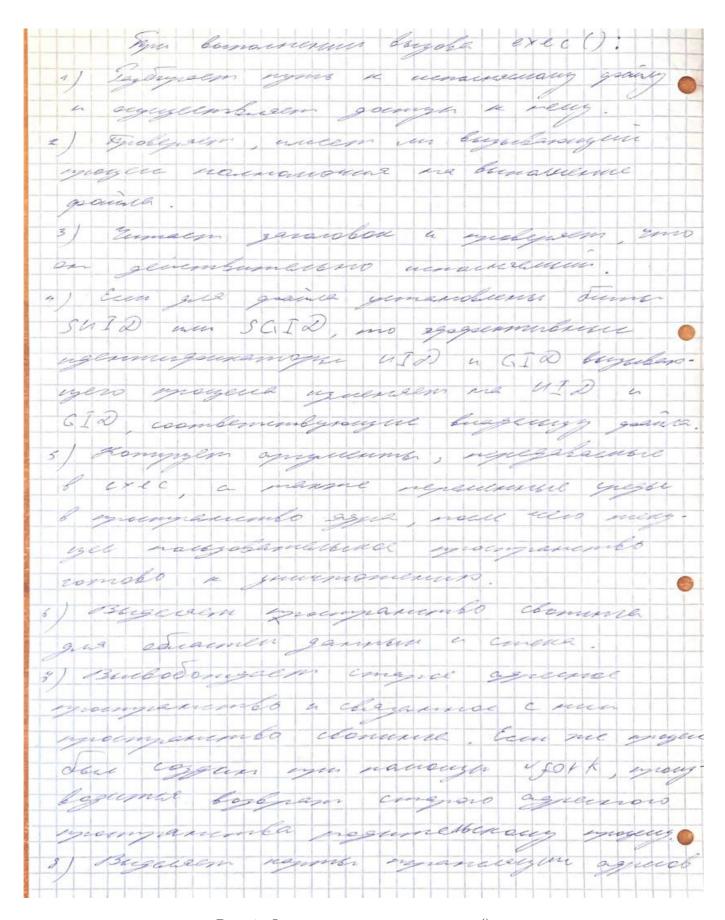


Рис. 9: Очередность при вызове exec().

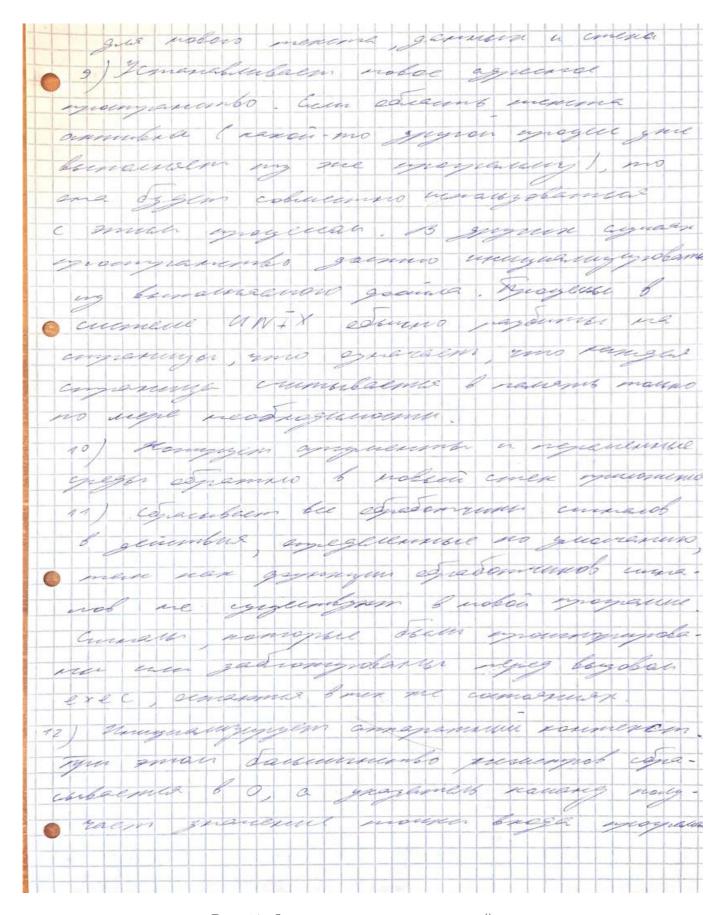


Рис. 10: Очередность при вызове exec().