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: Пронин А. С.

Группа: ИУ7-52Б

Преподаватель: Рязанова Н. Ю.

Оценка: _____

Москва

2021

Листинг 1: Программа 1

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main()
{
    int childpid_1, childpid_2;

    if ((childpid_1 = fork()) == -1)
    {
        perror("Can\'t␣fork.\n");
        return EXIT_FAILURE;
    }
    else if (childpid_1 == 0)
    {
        sleep(2);
        printf("\nFirst␣child:␣pid␣=␣%d;␣ppid␣=␣%d;␣␣pgrp␣=␣%d\n", getpid(),
            getppid(), getpgrp());
        exit(EXIT_SUCCESS);
    }

    if ((childpid_2 = fork()) == -1)
    {
        perror("Can\'t␣fork.\n");
        return EXIT_FAILURE;
    }
    else if (childpid_2 == 0)
    {
        sleep(3);
        printf("Second␣child:␣pid␣=␣%d;␣ppid␣=␣%d;␣␣pgrp␣=␣%d\n", getpid(), getppid(),
            getpgrp());
        exit(EXIT_SUCCESS);
    }

    printf("Parent:␣pid␣=␣%d;␣pgrp␣=␣%d;␣child1␣=␣%d;␣child2␣=␣%d\n", getpid(),
        getpgrp(), childpid_1, childpid_2);

    printf("Parent␣will␣die␣now.\n");
    return EXIT_SUCCESS;
}
```

```
arseny@arseny-VirtualBox:~/shared/Lab4$ gcc 1.c
arseny@arseny-VirtualBox:~/shared/Lab4$ ./a.out
Parent: pid = 7554; pgrp = 7554; child1 = 7555; child2 = 7556
Parent will die now.
arseny@arseny-VirtualBox:~/shared/Lab4$
First child: pid = 7555; ppid = 705;  pgrp = 7554
Second child: pid = 7556; ppid = 705;  pgrp = 7554
```

Рис. 1: Результат работы программы 1

```
arseny@arseny-VirtualBox:~/shared/Lab4$ ps -al
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  1000     787     782  0  80   0 - 146832 ep_pol tty2     00:00:45 Xorg
0 S  1000     916     782  0  80   0 - 49908 poll_s tty2      00:00:00 gnome-session-b
0 S  1000    7554    7369  0  80   0 -   624 hrtime pts/0     00:00:00 a.out
1 S  1000    7555    7554  0  80   0 -   591 hrtime pts/0     00:00:00 a.out
1 S  1000    7556    7554  0  80   0 -   591 hrtime pts/0     00:00:00 a.out
0 R  1000    7557    7487  0  80   0 -  5021 -      pts/1     00:00:00 ps
```

```
arseny@arseny-VirtualBox:~/shared/Lab4$ ps -al
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  1000     787     782  0  80   0 - 146812 ep_pol tty2     00:00:45 Xorg
0 S  1000     916     782  0  80   0 - 49908 poll_s tty2      00:00:00 gnome-session-b
1 S  1000    7555     705  0  80   0 -   591 hrtime pts/0     00:00:00 a.out
1 S  1000    7556     705  0  80   0 -   591 hrtime pts/0     00:00:00 a.out
0 R  1000    7558    7487  0  80   0 -  5021 -      pts/1     00:00:00 ps
```

Рис. 2: Демонстрация "усыновления"

Листинг 2: Программа 2

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>

void checkStatus(int child_pid, int status);

int main()
{
    int childpid_1, childpid_2;

    if ((childpid_1 = fork()) == -1)
    {
        perror("Can\'t fork.\n");
        return EXIT_FAILURE;
    }
    else if (childpid_1 == 0)
    {
        //sleep(1);
        printf("First child: pid = %d; ppid = %d;  pgrp = %d\n", getpid(), getppid(),
            getpgrp());
        exit(EXIT_SUCCESS);
    }

    if ((childpid_2 = fork()) == -1)
    {
        perror("Can\'t fork.\n");
        return EXIT_FAILURE;
    }
    else if (childpid_2 == 0)
    {
        //sleep(2);
        printf("Second child: pid = %d; ppid = %d;  pgrp = %d\n", getpid(), getppid(),
            getpgrp());
        exit(EXIT_SUCCESS);
    }

    printf("Parent: pid = %d; pgrp = %d; child1 = %d; child2 = %d\n", getpid(),
        getpgrp(), childpid_1, childpid_2);

    int status;
    pid_t child_pid;

    printf("Waiting...\n");
    child_pid = wait(&status);
    checkStatus(child_pid, status);
```

```
45
46      printf("Waiting...\n");
47      child_pid = wait(&status);
48      checkStatus(child_pid, status);
49
50      printf("Parent␣will␣die␣now.\n");
51      return EXIT_SUCCESS;
52  }
53
54  void checkStatus(int child_pid, int status)
55  {
56      if (WIFEXITED(status))
57          printf("Child␣with␣pid␣=␣%d␣has␣terminated␣normally.\n\n", child_pid);
58      else if (WEXITSTATUS(status))
59          printf("Child␣with␣pid␣=␣%d␣has␣terminated␣with␣code␣%d.\n", child_pid,
                WIFEXITED(status));
60      else if (WIFSIGNALED(status))
61      {
62          printf("Child␣with␣pid␣=␣%d␣has␣terminated␣with␣an␣un-intercepted␣signal.\n",
                child_pid);
63          printf("Signal␣number␣=␣%d.\n", WTERMSIG(status));
64      }
65      else if (WIFSTOPPED(status))
66      {
67          printf("Child␣with␣pid␣=␣%d␣has␣stopped.\n", child_pid);
68          printf("Signal␣number␣=␣%d.", WSTOPSIG(status));
69      }
70  }
```

```
arseny@arseny-VirtualBox:~/shared/Lab4$ gcc 2wait.c
arseny@arseny-VirtualBox:~/shared/Lab4$ ./a.out
Parent: pid = 7615; pgrp = 7615; child1 = 7616; child2 = 7617
Waiting...
Second child: pid = 7617; ppid = 7615;  pgrp = 7615
Child with pid = 7617 has terminated normally.

Waiting...
First child: pid = 7616; ppid = 7615;  pgrp = 7615
Child with pid = 7616 has terminated normally.

Parent will die now.
```

Рис. 3: Результат работы программы 2

Листинг 3: Программа 3

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>

void checkStatus(int child_pid, int status);

int main()
{
    int childpid_1, childpid_2;

    if ((childpid_1 = fork()) == -1)
    {
        perror("Can\'t␣fork.\n");
        return EXIT_FAILURE;
    }
    else if (childpid_1 == 0)
    {
        printf("First␣child:␣pid␣=␣%d;␣ppid␣=␣%d;␣␣pgrp␣=␣%d\n", getpid(), getppid(),
            getpgrp());
        if (execlp("ls", "ls", NULL) == -1)
        {
            perror("First␣child␣can\'t␣exec");
            exit(EXIT_FAILURE);
        }
        exit(EXIT_SUCCESS);
    }

    if ((childpid_2 = fork()) == -1)
    {
        perror("Can\'t␣fork.\n");
        return EXIT_FAILURE;
    }
    else if (childpid_2 == 0)
    {
        printf("Second␣child:␣pid␣=␣%d;␣ppid␣=␣%d;␣␣pgrp␣=␣%d\n", getpid(), getppid(),
            getpgrp());
        if (execl("sort", "sort", "999", "111", "9", "1", "11", "99", "55", "555",
            "5", NULL) == -1)
        {
            perror("Second␣child␣can\'t␣exec");
            exit(EXIT_FAILURE);
        }
        exit(EXIT_SUCCESS);
    }

```

```
45      printf("Parent: pid = %d; pgrp = %d; child1 = %d; child2 = %d\n", getpid(),
            getpgrp(), childpid_1, childpid_2);
46
47      int status;
48      pid_t child_pid;
49
50      printf("Waiting...\n");
51      child_pid = wait(&status);
52      checkStatus(child_pid, status);
53
54      printf("Waiting...\n");
55      child_pid = wait(&status);
56      checkStatus(child_pid, status);
57
58      printf("Parent will die now.\n");
59      return EXIT_SUCCESS;
60  }
61
62  void checkStatus(int child_pid, int status)
63  {
64      if (WIFEXITED(status))
65          printf("Child with pid = %d has terminated normally.\n\n", child_pid);
66      else if (WEXITSTATUS(status))
67          printf("Child with pid = %d has terminated with code %d.\n", child_pid,
                WIFEXITED(status));
68      else if (WIFSIGNALED(status))
69      {
70          printf("Child with pid = %d has terminated with an un-intercepted signal.\n",
                child_pid);
71          printf("Signal number = %d.\n", WTERMSIG(status));
72      }
73      else if (WIFSTOPPED(status))
74      {
75          printf("Child with pid = %d has stopped.\n", child_pid);
76          printf("Signal number = %d.", WSTOPSIG(status));
77      }
78  }
```

Листинг 4: Программа sort для потомка

```c
#include <stdio.h>
#include <stdlib.h>

void printMas(const int *mas, int size);
void selectionSort(int *l, int *r);

int main(int argc, char *argv[])
{
    int n = argc - 1;
    int mas[n];

    for (int i = 0; i < n; i++)
        mas[i] = atoi(argv[i + 1]);

    printf("Inputed␣array␣=␣");
    printMas(mas, n);

    selectionSort(&mas[0], &mas[n-1]);
    printf("Sorted␣array␣=␣");
    printMas(mas, n);

    return 0;
}

void printMas(const int *mas, int size)
{
    for (int i = 0; i < size; i++)
    {
        printf("%d␣", mas[i]);
    }
    printf("\n");
}

void swap(int *el1, int *el2)
{
    int temp = *el1;
    *el1 = *el2;
    *el2 = temp;
}
```

```
48 void selectionSort(int *l, int *r)
49 {
50     for (int *i = l; i <= r; i++)
51     {
52         int minz = *i, *ind = i;
53         for (int *j = i + 1; j <= r; j++)
54         {
55             if (*j < minz)
56             {
57                 minz = *j;
58                 ind = j;
59             }
60         }
61         swap(i, ind);
62     }
63 }
```

```
arseny@arseny-VirtualBox:~/shared/Lab4$ gcc 3.c
arseny@arseny-VirtualBox:~/shared/Lab4$ ./a.out
Parent: pid = 7628; pgrp = 7628; child1 = 7629; child2 = 7630
Waiting...
Second child: pid = 7630; ppid = 7628;  pgrp = 7628
First child: pid = 7629; ppid = 7628;  pgrp = 7628
Inputed array = 999 111 9 1 11 99 55 555 5
Sorted array = 1 5 9 11 55 99 111 555 999
Child with pid = 7630 has terminated normally.

Waiting...
1.c  2wait.c  3.c  4.c  5.c  a.out  SelectionSort.c  sort
Child with pid = 7629 has terminated normally.

Parent will die now.
```

Рис. 4: Результат работы программы 3

Листинг 5: Программа 4

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/wait.h>

#define LEN 50
#define TEXT1 "My name is Proffesional\n"
#define TEXT2 "There is no meaning in this words\n"

void checkStatus(int child_pid, int status);

int main()
{
    int childpid_1, childpid_2;
    int fd[2];

    if (pipe(fd) == -1)
    {
        perror("Can\'t␣pipe.\n");
        return EXIT_FAILURE;
    }

    if ((childpid_1 = fork()) == -1)
    {
        perror("Can\'t␣fork.\n");
        return EXIT_FAILURE;
    }
    else if (childpid_1 == 0)
    {
        close(fd[0]);
        write(fd[1], TEXT1, strlen(TEXT1) + 1);
        exit(EXIT_SUCCESS);
    }

    if ((childpid_2 = fork()) == -1)
    {
        perror("Can\'t␣fork.\n");
        return EXIT_FAILURE;
    }
    else if (childpid_2 == 0)
    {
        close(fd[0]);
        write(fd[1], TEXT2, strlen(TEXT2) + 1);
        exit(EXIT_SUCCESS);
    }
```

```
48
49      printf("Parent: pid = %d; pgrp = %d; child1 = %d; child2 = %d\n", getpid(),
            getpgrp(), childpid_1, childpid_2);
50
51      char text1[LEN], text2[LEN];
52
53      close(fd[1]);
54      read(fd[0], text1, LEN);
55      read(fd[0], text2, LEN);
56
57      printf("Text1: %s", text1);
58      printf("Text2: %s", text2);
59
60      int status;
61      pid_t child_pid;
62
63      printf("Waiting...\n");
64      child_pid = wait(&status);
65      checkStatus(child_pid, status);
66
67      printf("Waiting...\n");
68      child_pid = wait(&status);
69      checkStatus(child_pid, status);
70
71      printf("Parent will die now.\n");
72      return EXIT_SUCCESS;
73  }
74
75  void checkStatus(int child_pid, int status)
76  {
77      if (WIFEXITED(status))
78          printf("Child with pid = %d has terminated normally.\n\n", child_pid);
79      else if (WEXITSTATUS(status))
80          printf("Child with pid = %d has terminated with code %d.\n", child_pid,
                WIFEXITED(status));
81      else if (WIFSIGNALED(status))
82      {
83          printf("Child with pid = %d has terminated with an un-intercepted signal.\n",
                child_pid);
84          printf("Signal number = %d.\n", WTERMSIG(status));
85      }
86      else if (WIFSTOPPED(status))
87      {
88          printf("Child with pid = %d has stopped.\n", child_pid);
89          printf("Signal number = %d.", WSTOPSIG(status));
90      }
91  }
```

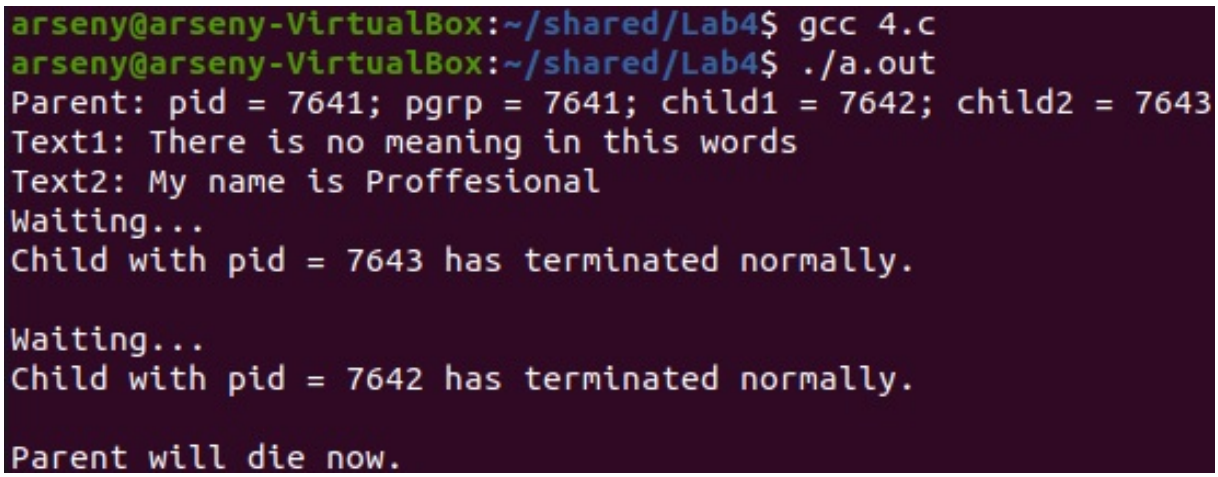

Рис. 5: Результат работы программы 4

Листинг 6: Программа 5

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/wait.h>

#define LEN 50
#define TEXT1 "My name is Proffesional\n"
#define TEXT2 "There is no meaning in this words\n"

void checkStatus(int child_pid, int status);
void catch_sig(int sig_numb);

int flag = 0;

int main()
{
    signal(SIGINT, catch_sig);

    int childpid_1, childpid_2;
    int fd[2];

    if (pipe(fd) == -1)
    {
        perror("Can\'t␣pipe.\n");
        return EXIT_FAILURE;
    }

    if ((childpid_1 = fork()) == -1)
    {
        perror("Can\'t␣fork.\n");
        return EXIT_FAILURE;
    }
    else if (childpid_1 == 0)
    {
        close(fd[0]);
        write(fd[1], TEXT1, strlen(TEXT1) + 1);
        exit(EXIT_SUCCESS);
    }

    if ((childpid_2 = fork()) == -1)
    {
        perror("Can\'t␣fork.\n");
        return EXIT_FAILURE;
    }
```

```
48      else if (childpid_2 == 0)
49      {
50          close(fd[0]);
51          write(fd[1], TEXT2, strlen(TEXT2) + 1);
52          exit(EXIT_SUCCESS);
53      }
54
55      printf("Parent: pid = %d; pgrp = %d; child1 = %d; child2 = %d\n", getpid(),
            getpgrp(), childpid_1, childpid_2);
56      printf("Press \"CTRL+C\", to see message from second child.\n");
57      printf("In other case you will see message from first child.\n\n");
58
59      char text1[LEN], text2[LEN];
60
61      close(fd[1]);
62      read(fd[0], text1, LEN);
63      read(fd[0], text2, LEN);
64
65      sleep(2);
66
67      if (flag)
68          printf("Message: %s", text2);
69      else
70          printf("Message: %s", text1);
71
72      int status;
73      pid_t child_pid;
74
75      printf("Waiting...\n");
76      child_pid = wait(&status);
77      checkStatus(child_pid, status);
78
79      printf("Waiting...\n");
80      child_pid = wait(&status);
81      checkStatus(child_pid, status);
82
83      printf("Parent will die now.\n");
84      return EXIT_SUCCESS;
85  }
86
87  void checkStatus(int child_pid, int status)
88  {
89      if (WIFEXITED(status))
90          printf("Child with pid = %d has terminated normally.\n\n", child_pid);
91      else if (WEXITSTATUS(status))
92          printf("Child with pid = %d has terminated with code %d.\n", child_pid,
                WIFEXITED(status));
93
```

```
94      else if (WIFSIGNALED(status))
95      {
96          printf("Child␣with␣pid␣=␣%d␣has␣terminated␣with␣an␣un-intercepted␣signal.\n",
                child_pid);
97          printf("Signal␣number␣=␣%d.\n", WTERMSIG(status));
98      }
99      else if (WIFSTOPPED(status))
100     {
101         printf("Child␣with␣pid␣=␣%d␣has␣stopped.\n", child_pid);
102         printf("Signal␣number␣=␣%d.", WSTOPSIG(status));
103     }
104 }
105
106 void catch_sig(int sig_numb)
107 {
108     flag = 1;
109     printf("\ncatch_sig:␣%d\n", sig_numb);
110 }
```

```
arseny@arseny-VirtualBox:~/shared/Lab4$ gcc 5.c
arseny@arseny-VirtualBox:~/shared/Lab4$ ./a.out
Parent: pid = 7654; pgrp = 7654; child1 = 7655; child2 = 7656
Press "CTRL+C", to see message from second child.
In other case you will see message from first child.

Message: There is no meaning in this words
Waiting...
Child with pid = 7655 has terminated normally.

Waiting...
Child with pid = 7656 has terminated normally.

Parent will die now.
```

```
arseny@arseny-VirtualBox:~/shared/Lab4$ ./a.out
Parent: pid = 7692; pgrp = 7692; child1 = 7693; child2 = 7694
Press "CTRL+C", to see message from second child.
In other case you will see message from first child.

^C
catch_sig: 2
Message: My name is Proffesional
Waiting...
Child with pid = 7693 has terminated normally.

Waiting...
Child with pid = 7694 has terminated normally.

Parent will die now.
```

Рис. 6: Результат работы программы 5