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»

Отчёт по лабораторной работе №4 по курсу «Операционные системы»

Тема Процессы. Системные вызовы fork() и exec()

Студент Якуба Д.В.

Группа ИУ7-53Б

Оценка (баллы) ________________

Преподаватель Рязанова Н.Ю.

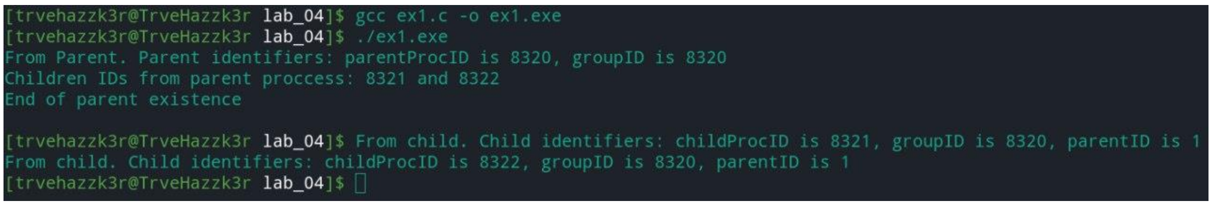
Москва, 2020 г

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 кода 1

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

#define CANT_FORK_ERROR 1
#define SUCCESS 0

 int main(void)
 {
    printf("From Parent. Parent identifiers: parentProcID is %d, groupID is %d\
n", getpid(), getpgrp());
    pid_t childpid;
    int children[2];

    for (int i = 0; i < 2; i++)
    {
        if ((childpid = fork()) == -1)
        {
            perror("Can't fork");
            exit(CANT_FORK_ERROR);
        }
        else if (childpid == 0)
        {
            sleep(1);
            printf("From child. Child identifiers: childProcID is %d, groupID i
s %d, parentID is %d\n", getpid(), getpgrp(), getppid());
            exit(SUCCESS);
        }
        else
            children[i] = childpid;

    }

    printf("Children IDs from parent proccess: %d and %d\nEnd of parent existen
ce\n\n", children[0], children[1]);

    return SUCCESS;
 }
```



```
[trvehazzk3r@TrveHazzk3r lab_04]$ gcc ex1.c -o ex1.exe
[trvehazzk3r@TrveHazzk3r lab_04]$ ./ex1.exe
From Parent. Parent identifiers: parentProcID is 8320, groupID is 8320
Children IDs from parent proccess: 8321 and 8322
End of parent existence

[trvehazzk3r@TrveHazzk3r lab_04]$ From child. Child identifiers: childProcID is 8321, groupID is 8320, parentID is 1
From child. Child identifiers: childProcID is 8322, groupID is 8320, parentID is 1
[trvehazzk3r@TrveHazzk3r lab_04]$ 
```

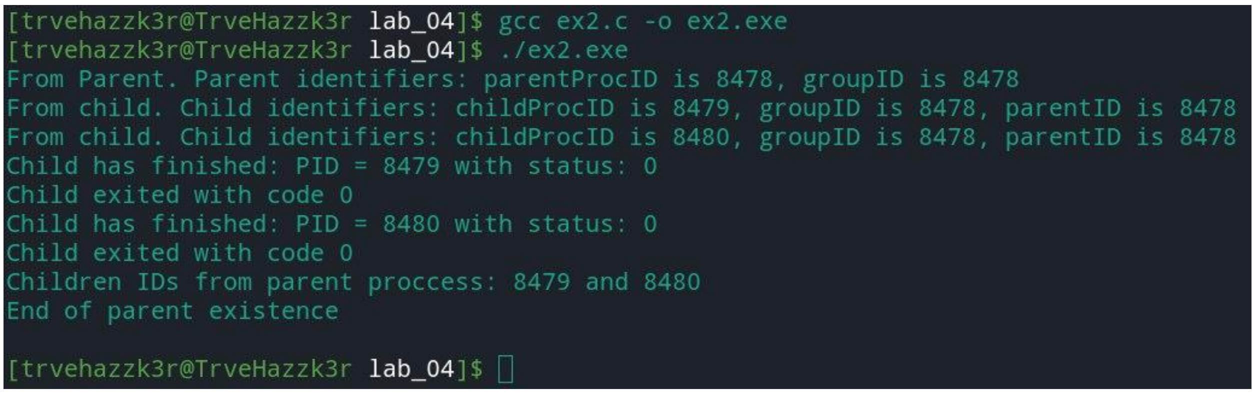
Рисунок 1 Демонстрация работа написанной программы

Задание 2

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

Листинг кода 2

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

#define CANT_FORK_ERROR 1
#define SUCCESS 0

 int main(void)
 {
    printf("From Parent. Parent identifiers: parentProcID is %d, groupID is %d\
n", getpid(), getpgrp());
    pid_t childpid;
    int children[2];

    for (int i = 0; i < 2; i++)
    {
        if ((childpid = fork()) == -1)
        {
            perror("Can't fork");
            return CANT_FORK_ERROR;
        }
        else if (childpid == 0)
        {
            sleep(1);
            printf("From child. Child identifiers: childProcID is %d, groupID i
s %d, parentID is %d\n", getpid(), getpgrp(), getppid());
            return SUCCESS;
        }
        else
            children[i] = childpid;

    }

    int childStatus;
    for (int i = 0; i < 2; i++)
    {
        childpid = wait(&childStatus);
        printf("Child has finished: PID = %d with status: %d\n", childpid, chil
dStatus);
        if (WIFEXITED(childStatus))
            printf("Child exited with code %d\n", WEXITSTATUS(childStatus));
        else if (WIFSTOPPED(childStatus))
            printf("Child process is currently stopped. Code: %d\n", WSTOPSIG(c
hildStatus));
        else if (WIFSIGNALED(childStatus))
            printf("Child process was terminated due to the receipt of a signal
 that was not caught. Code: %d\n", WTERMSIG(childStatus));
    }

    printf("Children IDs from parent proccess: %d and %d\nEnd of parent existen
ce\n\n", children[0], children[1]);

    return SUCCESS;
 }
```



```
[trvehazzk3r@TrveHazzk3r lab_04]$ gcc ex2.c -o ex2.exe
[trvehazzk3r@TrveHazzk3r lab_04]$ ./ex2.exe
From Parent. Parent identifiers: parentProcID is 8478, groupID is 8478
From child. Child identifiers: childProcID is 8479, groupID is 8478, parentID is 8478
From child. Child identifiers: childProcID is 8480, groupID is 8478, parentID is 8478
Child has finished: PID = 8479 with status: 0
Child exited with code 0
Child has finished: PID = 8480 with status: 0
Child exited with code 0
Children IDs from parent proccess: 8479 and 8480
End of parent existence

[trvehazzk3r@TrveHazzk3r lab_04]$
```

Рисунок 2 Демонстрация работы написанной программы

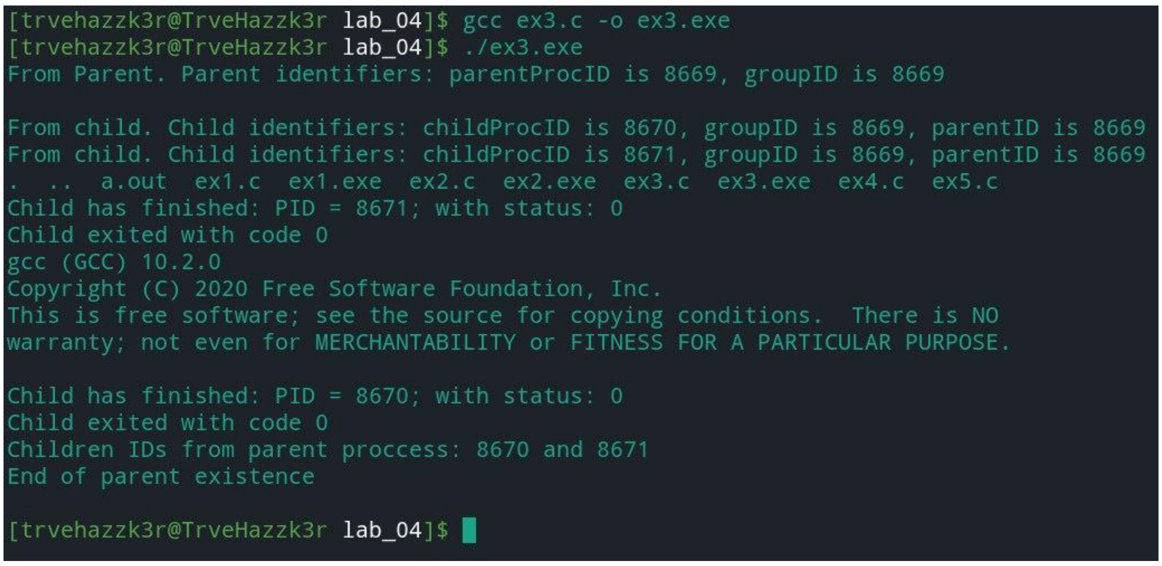
Задание 3

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

Листинг кода 3

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

#define CANT_FORK_ERROR 1
#define CANT_EXECLP_ERROR 2
#define SUCCESS 0
```

```
 int main(void)
 {
    printf("From Parent. Parent identifiers: parentProcID is %d, groupID is %d\
n", getpid(), getpgrp());
    pid_t childpid;
    int children[2];
    char *commands[2] = {"gcc", "ls"};
    char *arguments[2] = {"--version", "-a"};

    for (int i = 0; i < 2; i++)
    {
        if ((childpid = fork()) == -1)
        {
            perror("Can't fork");
            exit(CANT_FORK_ERROR);
        }
        else if (childpid == 0)
        {
            printf("From child. Child identifiers: childProcID is %d, groupID i
s %d, parentID is %d\n", getpid(), getpgrp(), getppid());

            if (execlp(commands[i], commands[i], arguments[i], NULL) == -1)
                {
                    perror("Can't execlp");
                    exit(CANT_EXECLP_ERROR);
                }

            exit(SUCCESS);
        }
        else
            children[i] = childpid;

    }
    printf("\n");

    int childStatus;
    for (int i = 0; i < 2; i++)
    {
        childpid = wait(&childStatus);
        printf("Child has finished: PID = %d with status: %d\n", childpid, chil
dStatus);
        if (WIFEXITED(childStatus))
            printf("Child exited with code %d\n", WEXITSTATUS(childStatus));
        else if (WIFSTOPPED(childStatus))
            printf("Child process is currently stopped. Code: %d\n", WSTOPSIG(c
hildStatus));
        else if (WIFSIGNALED(childStatus))
            printf("Child process was terminated due to the receipt of a signal
 that was not caught. Code: %d\n", WTERMSIG(childStatus));
    }

    printf("Children IDs from parent proccess: %d and %d\nEnd of parent existen
ce\n\n", children[0], children[1]);

    return SUCCESS;
 }
```



```
[trvehazzk3r@TrveHazzk3r lab_04]$ gcc ex3.c -o ex3.exe
[trvehazzk3r@TrveHazzk3r lab_04]$ ./ex3.exe
From Parent. Parent identifiers: parentProcID is 8669, groupID is 8669

From child. Child identifiers: childProcID is 8670, groupID is 8669, parentID is 8669
From child. Child identifiers: childProcID is 8671, groupID is 8669, parentID is 8669
.  ..  a.out  ex1.c  ex1.exe  ex2.c  ex2.exe  ex3.c  ex3.exe  ex4.c  ex5.c
Child has finished: PID = 8671; with status: 0
Child exited with code 0
gcc (GCC) 10.2.0
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Child has finished: PID = 8670; with status: 0
Child exited with code 0
Children IDs from parent proccess: 8670 and 8671
End of parent existence

[trvehazzk3r@TrveHazzk3r lab_04]$
```

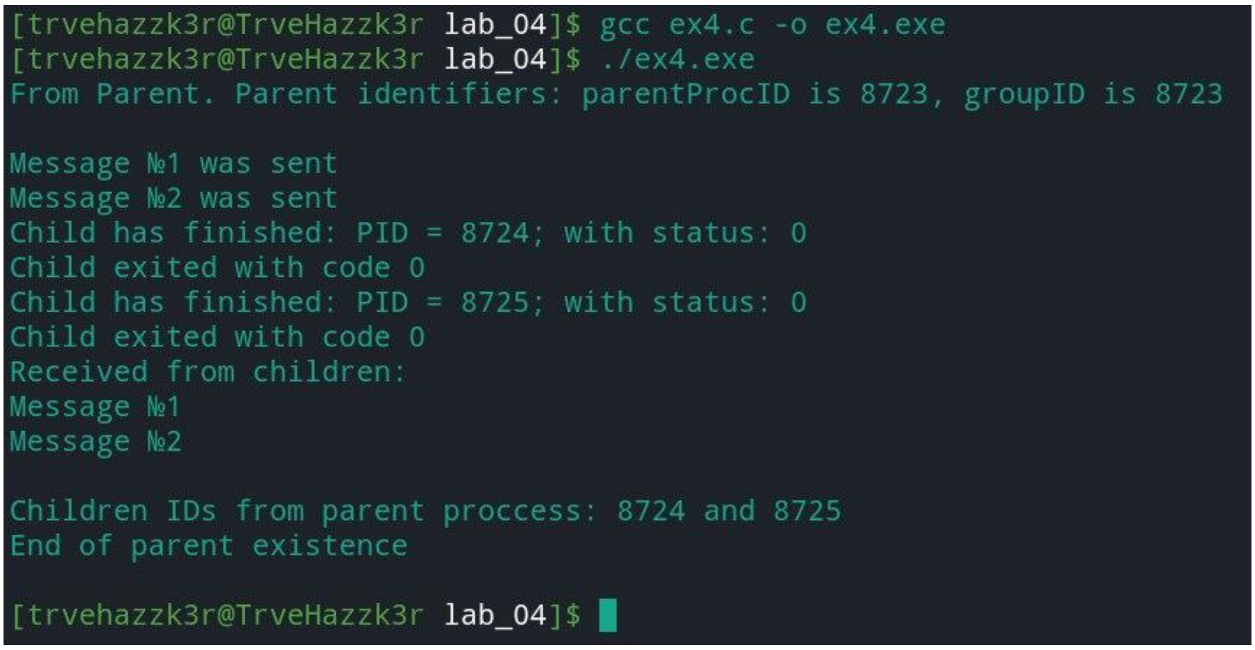
Рисунок 3 Демонстрация работы написанной программы

Задание 4

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

Листинг кода 4

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

#define CANT_FORK_ERROR 1
#define CANT_CREATE_PIPE_ERROR 3
#define SUCCESS 0

 int main(void)
 {
    printf("From Parent. Parent identifiers: parentProcID is %d, groupID is %d\
n", getpid(), getpgrp());
    int fd[2];
    pid_t childpid;
    int children[2];
    char *pipeMessages[2] = {"Message №1\n", "Message №2\n"};

    if (pipe(fd) == -1)
    {
        perror("Can't create pipe");
        exit(CANT_CREATE_PIPE_ERROR);
    }

    for (int i = 0; i < 2; i++)
    {
        if ((childpid = fork()) == -1)
        {
            perror("Can't fork");
            exit(CANT_FORK_ERROR);
        }
        else if (childpid == 0)
        {
            close(fd[0]);
            write(fd[1], pipeMessages[i], strlen(pipeMessages[i]));
            printf("Message №%d was sent\n", i + 1);

            exit(SUCCESS);
        }
        else
            children[i] = childpid;

    }
    printf("\n");

    int childStatus;
    for (int i = 0; i < 2; i++)
    {
        childpid = wait(&childStatus);
        printf("Child has finished: PID = %d with status: %d\n", childpid, chil
dStatus);
        if (WIFEXITED(childStatus))
            printf("Child exited with code %d\n", WEXITSTATUS(childStatus));
        else if (WIFSTOPPED(childStatus))
            printf("Child process is currently stopped. Code: %d\n", WSTOPSIG(c
hildStatus));
        else if (WIFSIGNALED(childStatus))
            printf("Child process was terminated due to the receipt of a signal
 that was not caught. Code: %d\n", WTERMSIG(childStatus));
    }

    char gotMessages[32] = { 0 };
    close(fd[1]);
    if (read(fd[0], gotMessages, 32) > 0)
        printf("Received from children:\n%s\n", gotMessages);
    else
        printf("No messages from children.\n");


    printf("Children IDs from parent proccess: %d and %d\nEnd of parent existen
ce\n\n", children[0], children[1]);

    return SUCCESS;
 }
```



```
[trvehazzk3r@TrveHazzk3r lab_04]$ gcc ex4.c -o ex4.exe
[trvehazzk3r@TrveHazzk3r lab_04]$ ./ex4.exe
From Parent. Parent identifiers: parentProcID is 8723, groupID is 8723

Message №1 was sent
Message №2 was sent
Child has finished: PID = 8724; with status: 0
Child exited with code 0
Child has finished: PID = 8725; with status: 0
Child exited with code 0
Received from children:
Message №1
Message №2

Children IDs from parent proccess: 8724 and 8725
End of parent existence

[trvehazzk3r@TrveHazzk3r lab_04]$
```

Рисунок 4 Демонстрация работы написанной программы

Задание 5

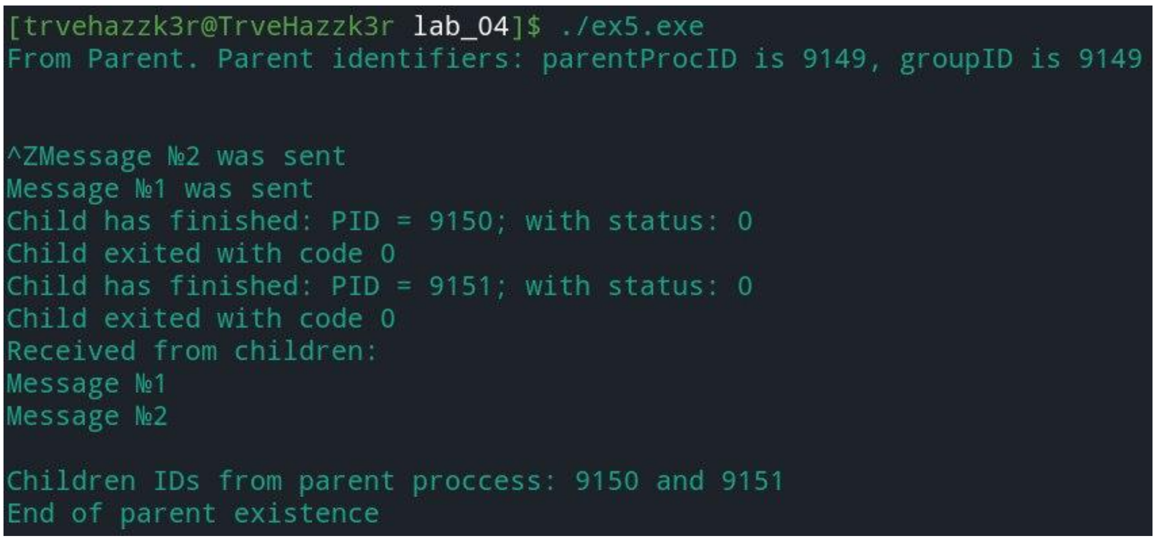
Предок и потомки обмениваются сообщениями через неименованный программный канал. С помощью сигнала меняется ход выполнения программы. Предок ждет завершения своих потомков. Вывод соответствующих сообщений на экран.

Листинг кода 5

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

#define CANT_FORK_ERROR 1
#define CANT_CREATE_PIPE_ERROR 3
#define SUCCESS 0

short canWrite = 0;

void noSIGTSTP()
{
}

void makeCanWriteTrue()
```

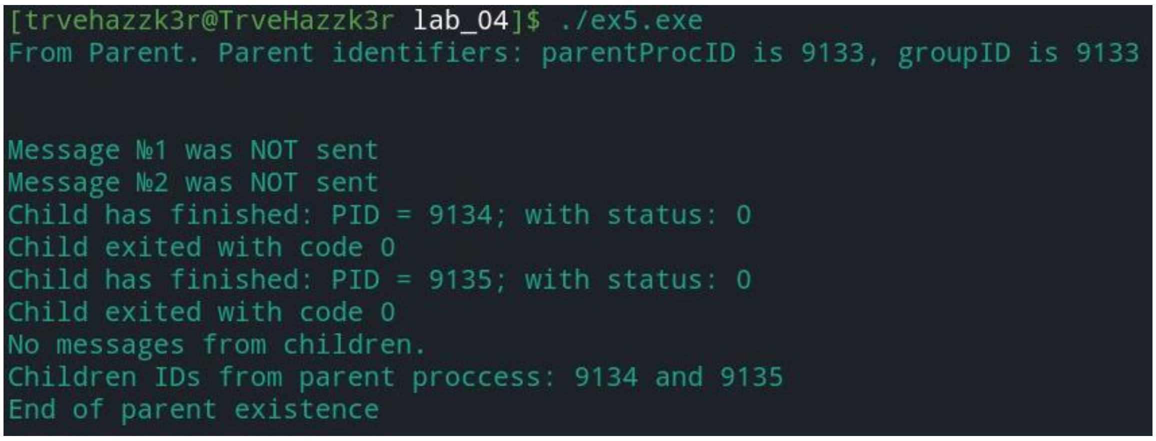
```
{
    canWrite = 1;
}

 int main(void)
 {
    printf("From Parent. Parent identifiers: parentProcID is %d, groupID is %d\
n", getpid(), getpgrp());
    int fd[2];
    pid_t childpid;
    int children[2];
    char *pipeMessages[2] = {"Message №1\n", "Message №2\n"};

    if (pipe(fd) == -1)
    {
        perror("Can't create pipe");
        exit(CANT_CREATE_PIPE_ERROR);
    }

    signal(SIGTSTP, noSIGTSTP);

    for (int i = 0; i < 2; i++)
    {
        if ((childpid = fork()) == -1)
        {
            perror("Can't fork");
            exit(CANT_FORK_ERROR);
        }
        else if (childpid == 0)
        {
            signal(SIGTSTP, makeCanWriteTrue);
            sleep(4);
            if (canWrite)
            {
                close(fd[0]);
                write(fd[1], pipeMessages[i], strlen(pipeMessages[i]));
                printf("Message №%d was sent\n", i + 1);
            }
            else
                printf("Message №%d was NOT sent\n", i + 1);

            exit(SUCCESS);
        }
        else
            children[i] = childpid;

    }
    printf("\n\n");

    int childStatus;
    for (int i = 0; i < 2; i++)
    {
        childpid = wait(&childStatus);
        printf("Child has finished: PID = %d with status: %d\n", childpid, chil
dStatus);
        if (WIFEXITED(childStatus))
            printf("Child exited with code %d\n", WEXITSTATUS(childStatus));
        else if (WIFSTOPPED(childStatus))
```

```
            printf("Child process is currently stopped. Code: %d\n", WSTOPSIG(c
hildStatus));
        else if (WIFSIGNALED(childStatus))
            printf("Child process was terminated due to the receipt of a signal
 that was not caught. Code: %d\n", WTERMSIG(childStatus));
    }

    char gotMessages[32] = { 0 };
    close(fd[1]);
    if (read(fd[0], gotMessages, 32) > 0)
        printf("Received from children:\n%s\n", gotMessages);
    else
        printf("No messages from children.\n");


    printf("Children IDs from parent proccess: %d and %d\nEnd of parent existen
ce\n\n", children[0], children[1]);

    return SUCCESS;
 }
```



```
[trvehazzk3r@TrveHazzk3r lab_04]$ ./ex5.exe
From Parent. Parent identifiers: parentProcID is 9149, groupID is 9149


^ZMessage №2 was sent
Message №1 was sent
Child has finished: PID = 9150; with status: 0
Child exited with code 0
Child has finished: PID = 9151; with status: 0
Child exited with code 0
Received from children:
Message №1
Message №2

Children IDs from parent proccess: 9150 and 9151
End of parent existence
```

Рисунок 5 Демонстрация работы написанной программы, сигнал вызывается



```
[trvehazzk3r@TrveHazzk3r lab_04]$ ./ex5.exe
From Parent. Parent identifiers: parentProcID is 9133, groupID is 9133


Message №1 was NOT sent
Message №2 was NOT sent
Child has finished: PID = 9134; with status: 0
Child exited with code 0
Child has finished: PID = 9135; with status: 0
Child exited with code 0
No messages from children.
Children IDs from parent proccess: 9134 and 9135
End of parent existence
```

Рисунок 6 Демонстрация работы написанной программы, сигнал не вызывается