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» (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

О Т Ч Е Т

по лабораторной работе № 2

Название: Процессы. Системные вызовы fork, exec

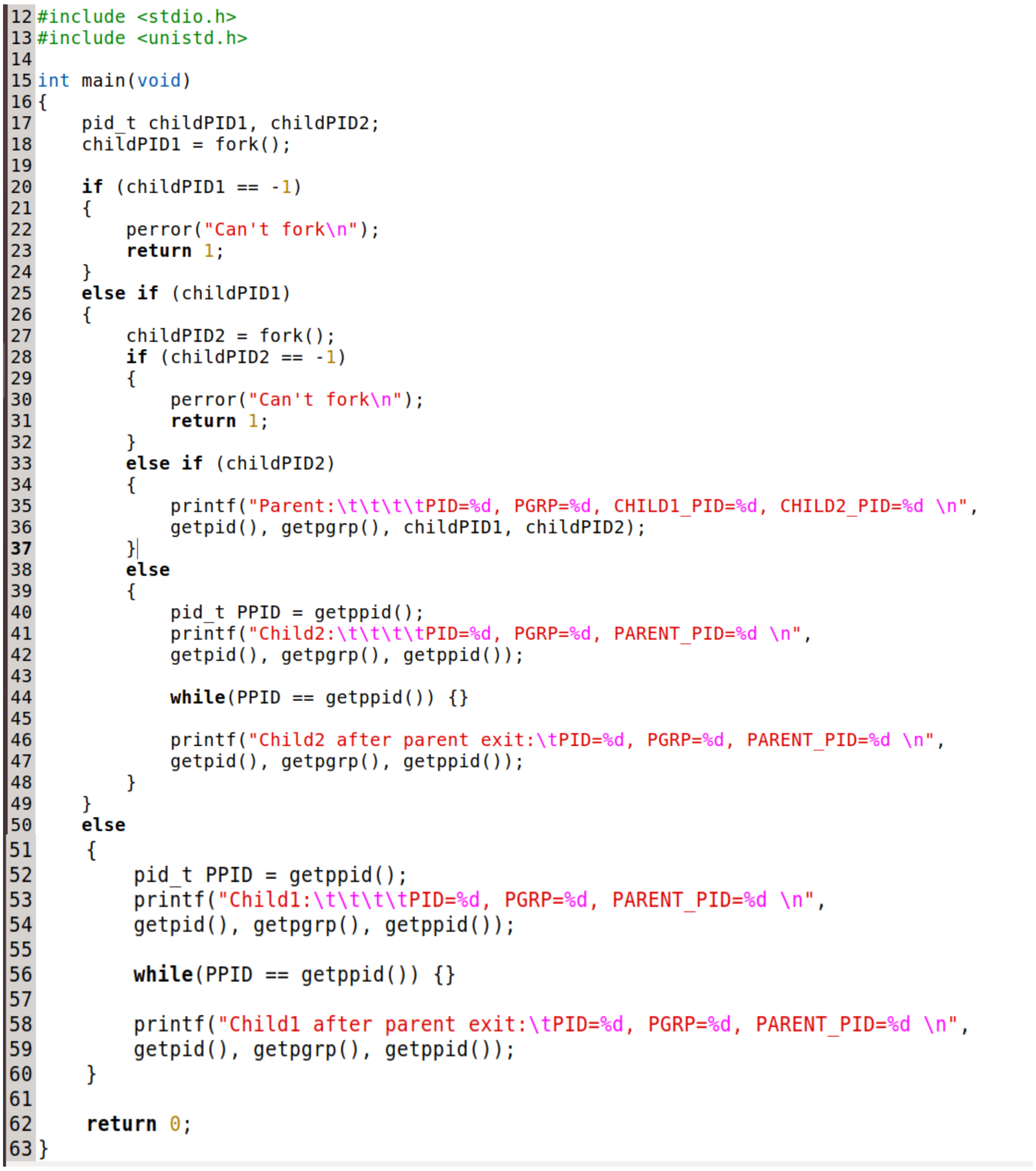
Дисциплина: Операционные системы

Студент	ИУ7-52Б		В.А. Иванов
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			Н.Ю. Рязанова
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2020

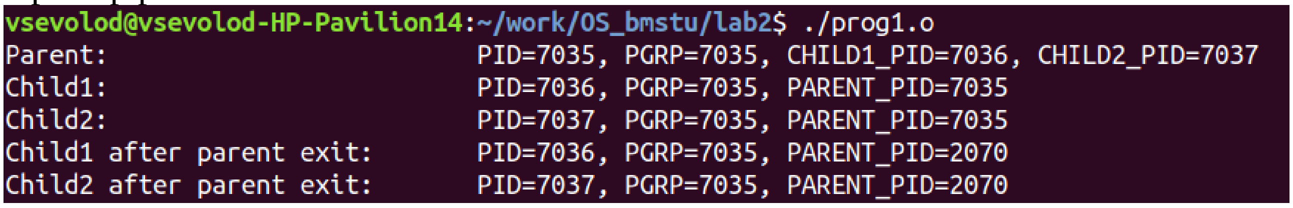
Задание 1

Код программы:



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

int main(void)
{
    pid_t childPID1, childPID2;
    childPID1 = fork();

    if (childPID1 == -1)
    {
        perror("Can't fork\n");
        return 1;
    }
    else if (childPID1)
    {
        childPID2 = fork();
        if (childPID2 == -1)
        {
            perror("Can't fork\n");
            return 1;
        }
        else if (childPID2)
        {
            printf("Parent:\t\t\t\tPID=%d, PGRP=%d, CHILD1_PID=%d, CHILD2_PID=%d \n",
            getpid(), getpgrp(), childPID1, childPID2);
        }
        else
        {
            pid_t PPID = getppid();
            printf("Child2:\t\t\t\tPID=%d, PGRP=%d, PARENT_PID=%d \n",
            getpid(), getpgrp(), getppid());

            while(PPID == getppid()) {}

            printf("Child2 after parent exit:\tPID=%d, PGRP=%d, PARENT_PID=%d \n",
            getpid(), getpgrp(), getppid());
        }
    }
    else
    {
        pid_t PPID = getppid();
        printf("Child1:\t\t\t\tPID=%d, PGRP=%d, PARENT_PID=%d \n",
        getpid(), getpgrp(), getppid());

        while(PPID == getppid()) {}

        printf("Child1 after parent exit:\tPID=%d, PGRP=%d, PARENT_PID=%d \n",
        getpid(), getpgrp(), getppid());
    }

    return 0;
}
```

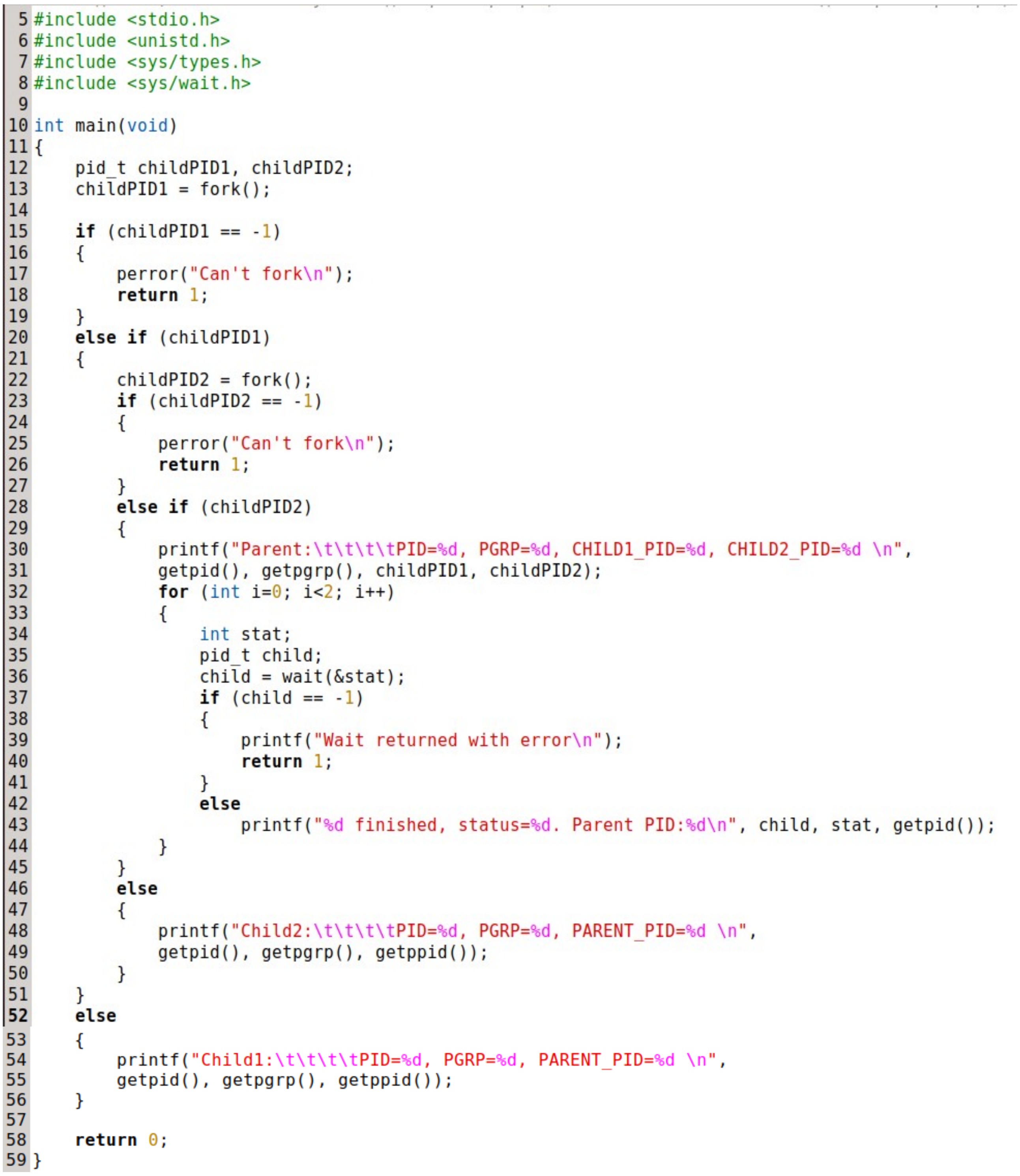
Пример работы:



```
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
```

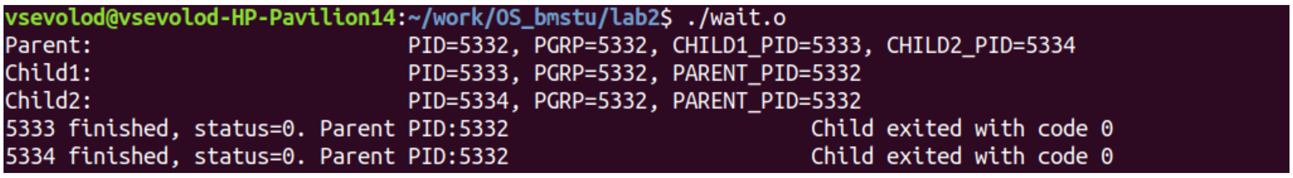
Задание 2

Код программы:



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

int main(void)
{
    pid_t childPID1, childPID2;
    childPID1 = fork();

    if (childPID1 == -1)
    {
        perror("Can't fork\n");
        return 1;
    }
    else if (childPID1)
    {
        childPID2 = fork();
        if (childPID2 == -1)
        {
            perror("Can't fork\n");
            return 1;
        }
        else if (childPID2)
        {
            printf("Parent:\t\t\t\tPID=%d, PGRP=%d, CHILD1_PID=%d, CHILD2_PID=%d \n",
            getpid(), getpgrp(), childPID1, childPID2);
            for (int i=0; i<2; i++)
            {
                int stat;
                pid_t child;
                child = wait(&stat);
                if (child == -1)
                {
                    printf("Wait returned with error\n");
                    return 1;
                }
                else
                    printf("%d finished, status=%d. Parent PID:%d\n", child, stat, getpid());
            }
        }
        else
        {
            printf("Child2:\t\t\t\tPID=%d, PGRP=%d, PARENT_PID=%d \n",
            getpid(), getpgrp(), getppid());
        }
    }
    else
    {
        printf("Child1:\t\t\t\tPID=%d, PGRP=%d, PARENT_PID=%d \n",
        getpid(), getpgrp(), getppid());
    }

    return 0;
}
```

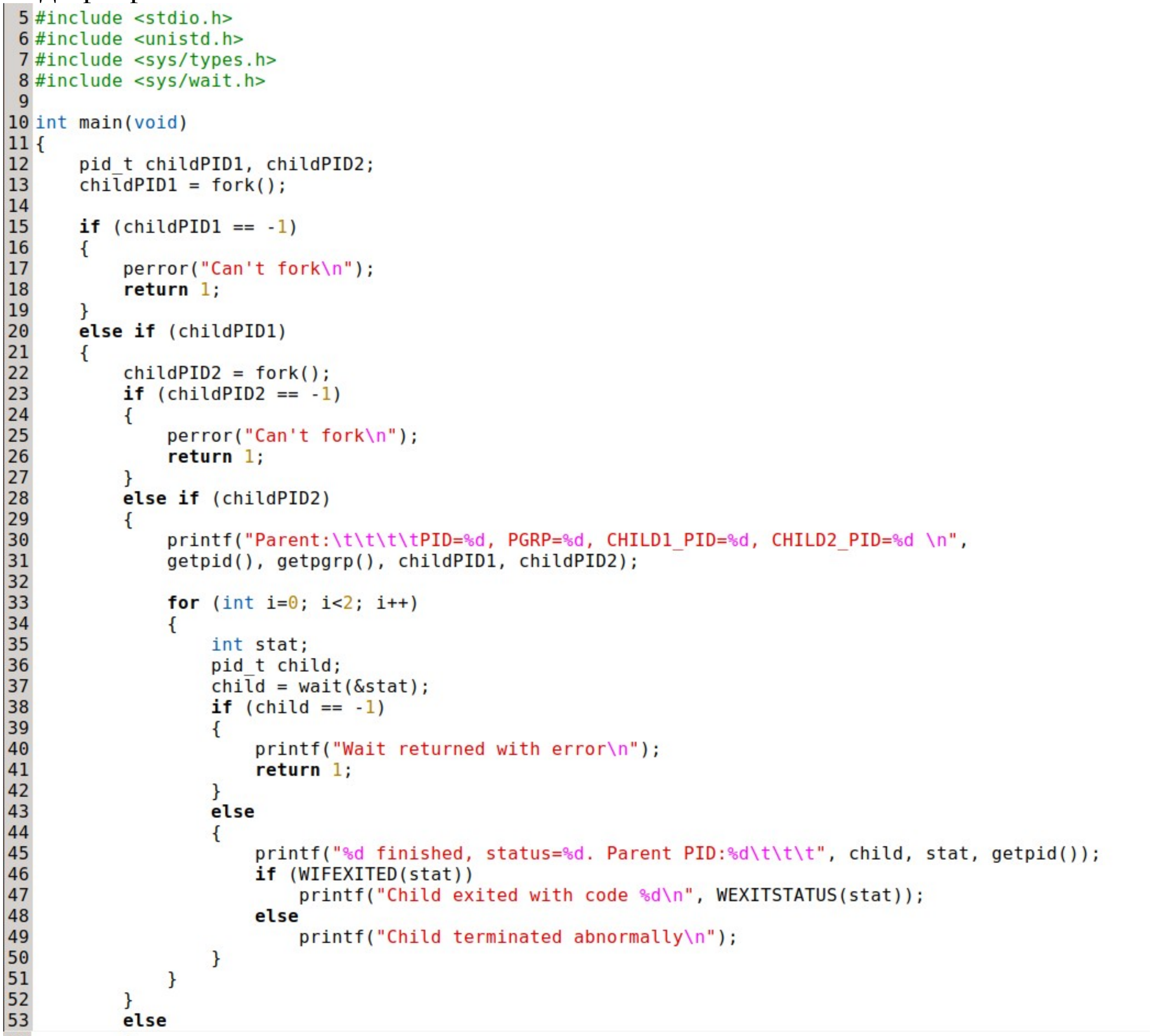
Пример работы:



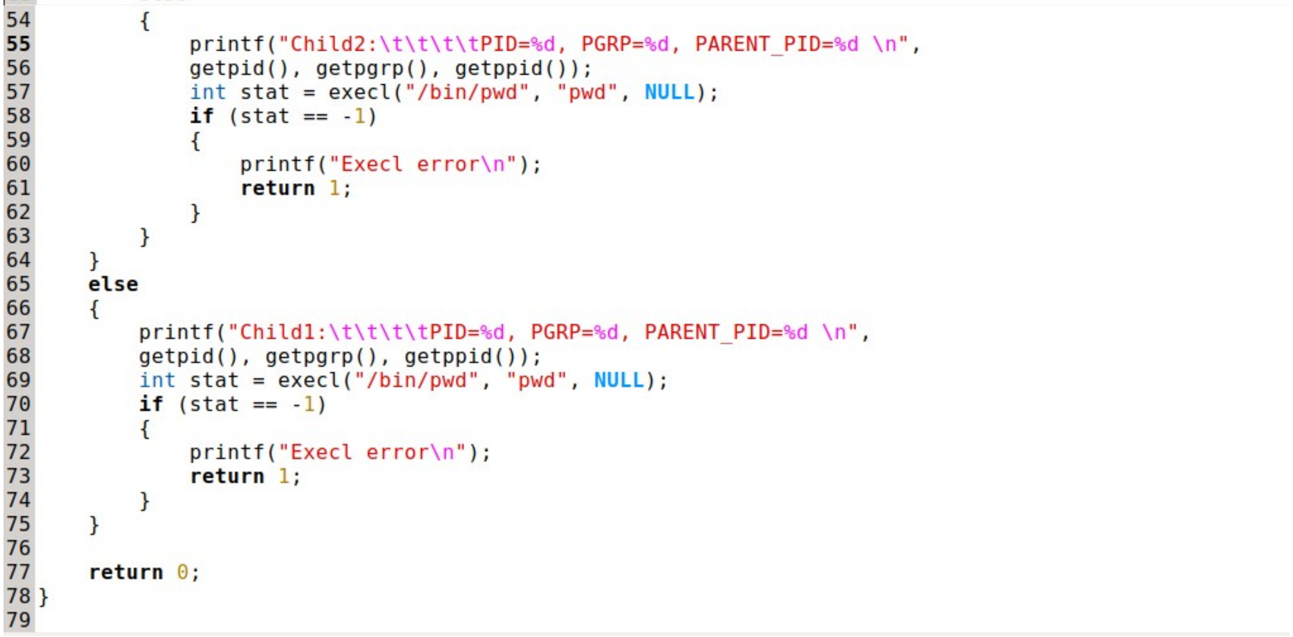
```
vsevolod@vsevolod-HP-Pavilion14:~/work/OS_bmstu/lab2$ ./wait.o
Parent:                         PID=5332, PGRP=5332, CHILD1_PID=5333, CHILD2_PID=5334
Child1:                         PID=5333, PGRP=5332, PARENT_PID=5332
Child2:                         PID=5334, PGRP=5332, PARENT_PID=5332
5333 finished,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

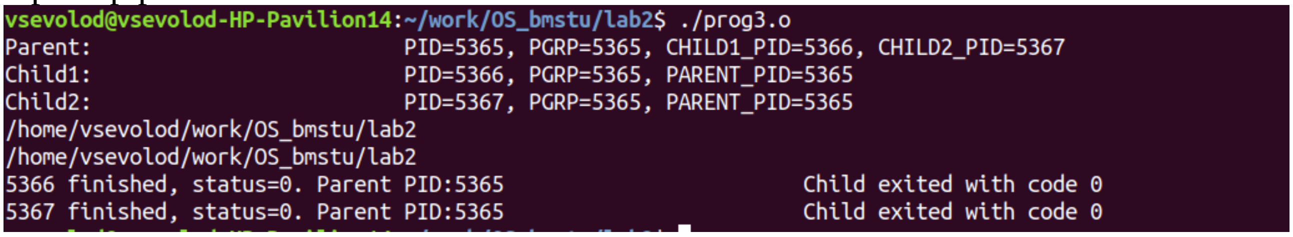


```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main(void)
{
    pid_t childPID1, childPID2;
    childPID1 = fork();

    if (childPID1 == -1)
    {
        perror("Can't fork\n");
        return 1;
    }
    else if (childPID1)
    {
        childPID2 = fork();
        if (childPID2 == -1)
        {
            perror("Can't fork\n");
            return 1;
        }
        else if (childPID2)
        {
            printf("Parent:\t\t\t\tPID=%d, PGRP=%d, CHILD1_PID=%d, CHILD2_PID=%d \n",
            getpid(), getpgrp(), childPID1, childPID2);

            for (int i=0; i<2; i++)
            {
                int stat;
                pid_t child;
                child = wait(&stat);
                if (child == -1)
                {
                    printf("Wait returned with error\n");
                    return 1;
                }
                else
                {
                    printf("%d finished, status=%d. Parent PID:%d\t\t\t", child, stat, getpid());
                    if (WIFEXITED(stat))
                        printf("Child exited with code %d\n", WEXITSTATUS(stat));
                    else
                        printf("Child terminated abnormally\n");
                }
            }
        }
        else
```



```c
        {
            printf("Child2:\t\t\t\tPID=%d, PGRP=%d, PARENT_PID=%d \n",
            getpid(), getpgrp(), getppid());
            int stat = execl("/bin/pwd", "pwd", NULL);
            if (stat == -1)
            {
                printf("Execl error\n");
                return 1;
            }
        }
    }
    else
    {
        printf("Child1:\t\t\t\tPID=%d, PGRP=%d, PARENT_PID=%d \n",
        getpid(), getpgrp(), getppid());
        int stat = execl("/bin/pwd", "pwd", NULL);
        if (stat == -1)
        {
            printf("Execl error\n");
            return 1;
        }
    }

    return 0;
}

```

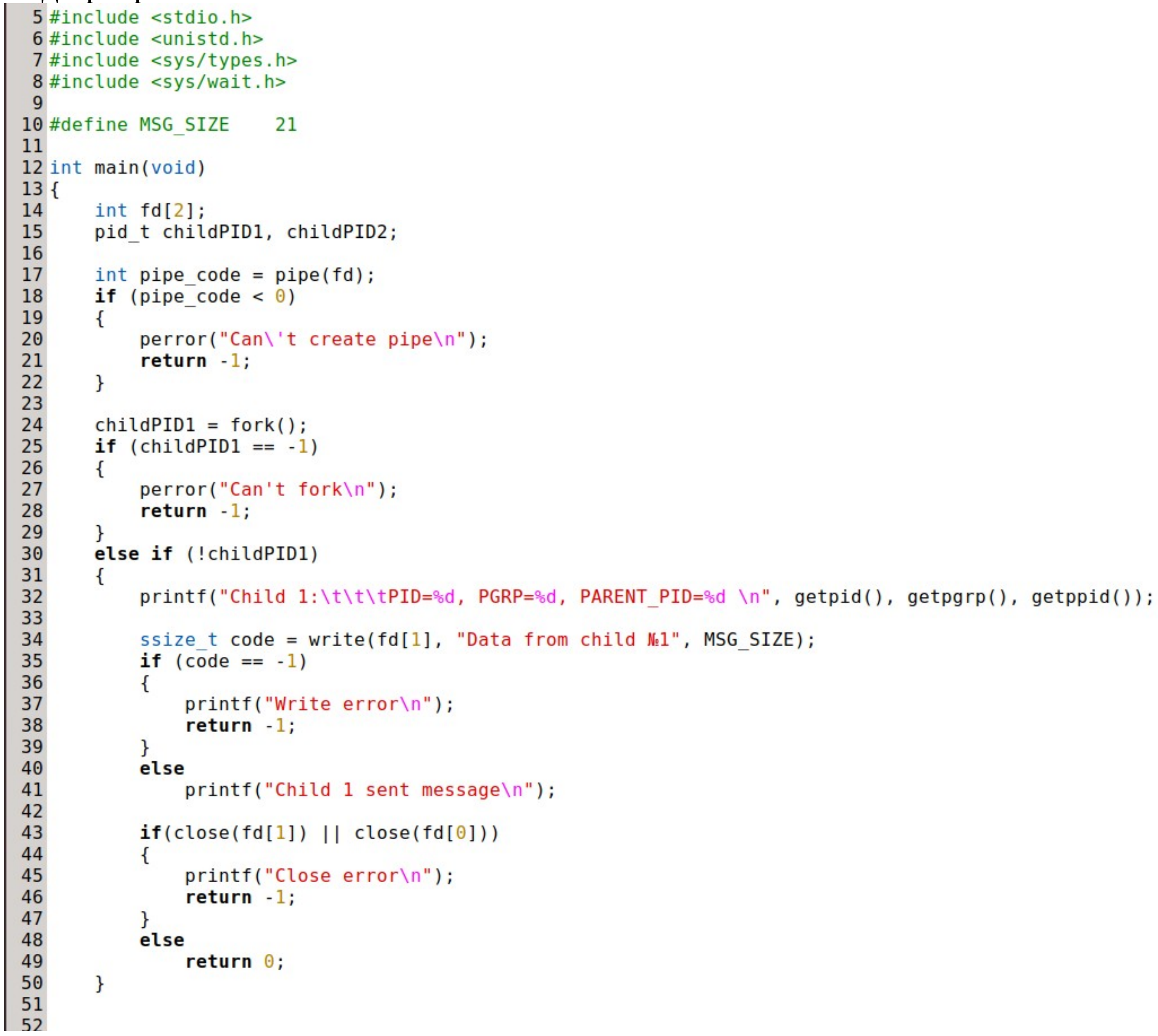
Пример работы:



```
vsevolod@vsevolod-HP-Pavilion14:~/work/OS_bmstu/lab2$ ./prog3.o
Parent:                         PID=5365, PGRP=5365, CHILD1_PID=5366, CHILD2_PID=5367
Child1:                         PID=5366, PGRP=5365, PARENT_PID=5365
Child2:                         PID=5367, PGRP=5365, PARENT_PID=5365
/home/vsevolod/work/OS_bmstu/lab2
/home/vsevolod/work/OS_bmstu/lab2
5366 finished, status=0. Parent PID:5365                        Child exited with code 0
5367 finished, status=0. Parent PID:5365                        Child exited with code 0
```

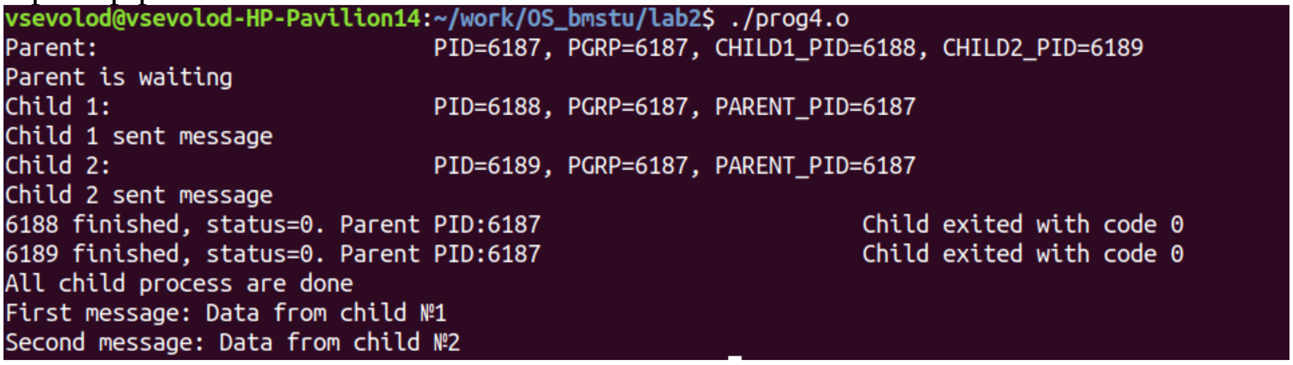
Задание 4

Код программы:



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

#define MSG_SIZE    21

int main(void)
{
    int fd[2];
    pid_t childPID1, childPID2;

    int pipe_code = pipe(fd);
    if (pipe_code < 0)
    {
        perror("Can\'t create pipe\n");
        return -1;
    }

    childPID1 = fork();
    if (childPID1 == -1)
    {
        perror("Can't fork\n");
        return -1;
    }
    else if (!childPID1)
    {
        printf("Child 1:\t\t\tPID=%d, PGRP=%d, PARENT_PID=%d \n", getpid(), getpgrp(), getppid());

        ssize_t code = write(fd[1], "Data from child №1", MSG_SIZE);
        if (code == -1)
        {
            printf("Write error\n");
            return -1;
        }
        else
            printf("Child 1 sent message\n");

        if(close(fd[1]) || close(fd[0]))
        {
            printf("Close error\n");
            return -1;
        }
        else
            return 0;
    }


    childPID2 = fork();
    if (childPID2 == 0)
    {
        printf("Child 2:\t\t\tPID=%d, PGRP=%d, PARENT_PID=%d \n", getpid(), getpgrp(), getppid());

        ssize_t code = write(fd[1], "Data from child №2", MSG_SIZE);
        if (code == -1)
        {
            printf("Write error\n");
            return -1;
        }
        else
            printf("Child 1 sent message\n");

        if (close(fd[1]) || close(fd[0]))
        {
            printf("Close error\n");
            return -1;
        }
        else
            return 0;
    }
    else if (childPID2 == -1)
    {
        perror("Can't fork\n");
        return -1;
    }


    printf("Parent:\t\t\t\tPID=%d, PGRP=%d, CHILD1_PID=%d, CHILD2_PID=%d \n",
    getpid(), getpgrp(), childPID1, childPID2);

    printf("Parent is waiting\n");
    int stat, child;
    for (int i=0; i<2; i++)
    {
        child = wait(&stat);
        if (child == -1)
        {
            printf("Wait returned with error\n");
            return -1;
        }
        else
        {
            printf("%d finished, status=%d. Parent PID:%d\t\t\t", child, stat, getpid());
            if (WIFEXITED(stat))
                printf("Child exited with code %d\n", WEXITSTATUS(stat));
            else
                printf("Child terminated abnormally\n");
        }
    }
    printf("All child process are done\n");

    char res1[MSG_SIZE], res2[MSG_SIZE];
    ssize_t code1 = read(fd[0], res1, MSG_SIZE);
    ssize_t code2 = read(fd[0], res2, MSG_SIZE);
    if (code1 == -1 || code1 == -1)
    {
        printf("Read error\n");
        return -1;
    }
    printf("First message: %s\n", res1);
    printf("Second message: %s\n", res2);

    if (close(fd[1]) || close(fd[0]))
    {
        printf("Close error\n");
        return -1;
    }
    else
        return 0;
}
```

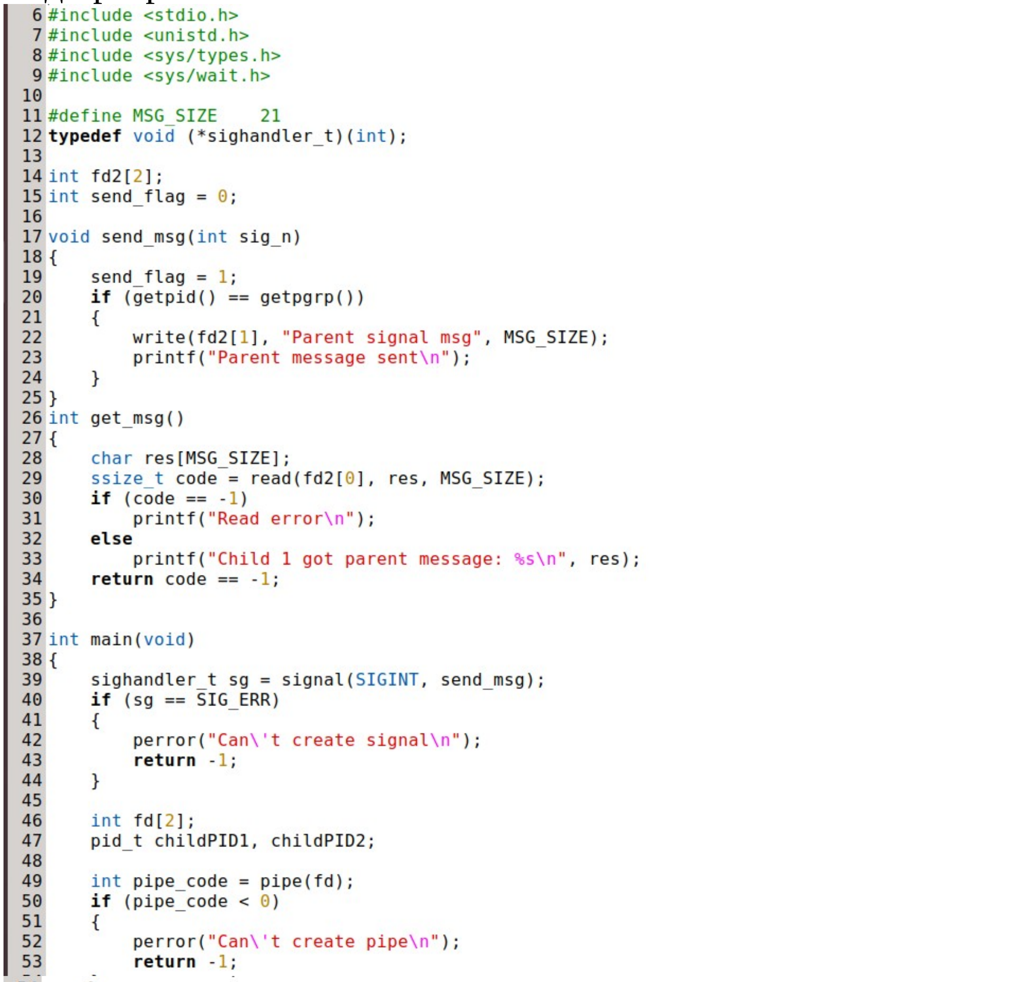
Пример работы:



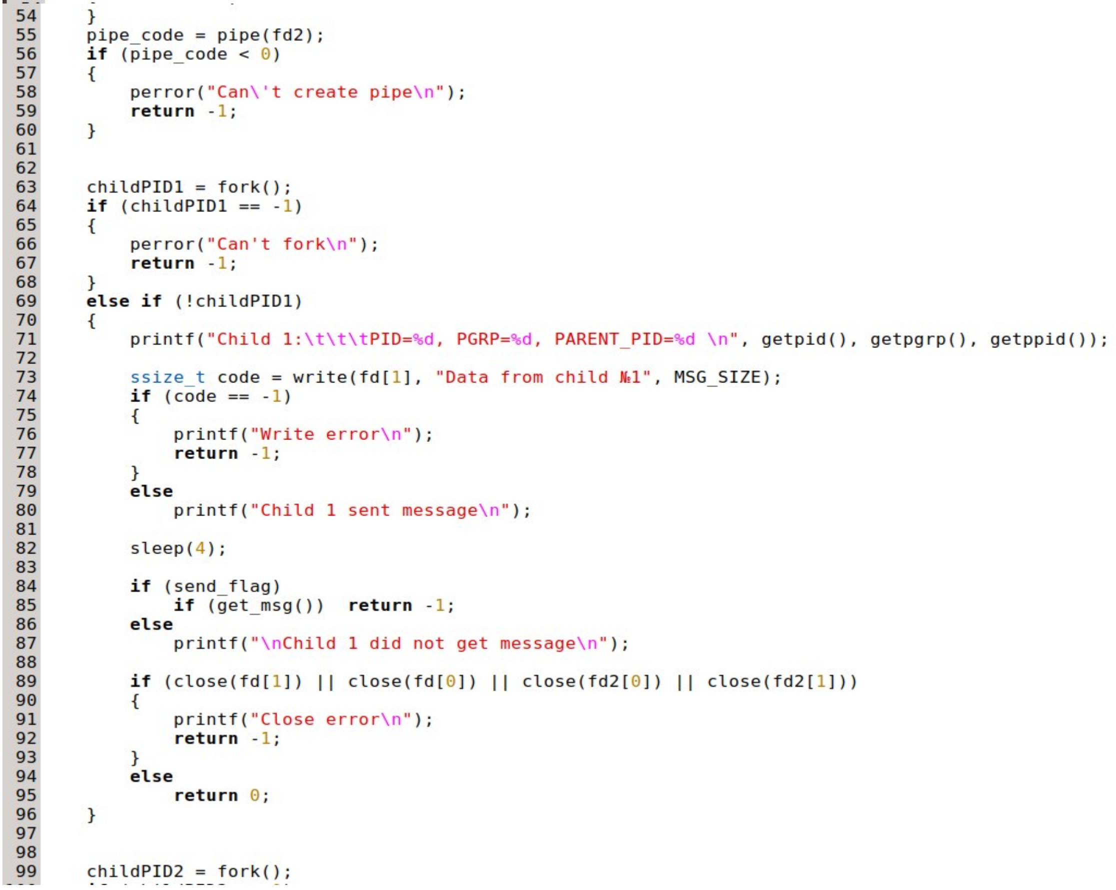
```
vsevolod@vsevolod-HP-Pavilion14:~/work/OS_bmstu/lab2$ ./prog4.o
Parent:                         PID=6187, PGRP=6187, CHILD1_PID=6188, CHILD2_PID=6189
Parent is waiting
Child 1:                        PID=6188, PGRP=6187, PARENT_PID=6187
Child 1 sent message
Child 2:                        PID=6189, 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:

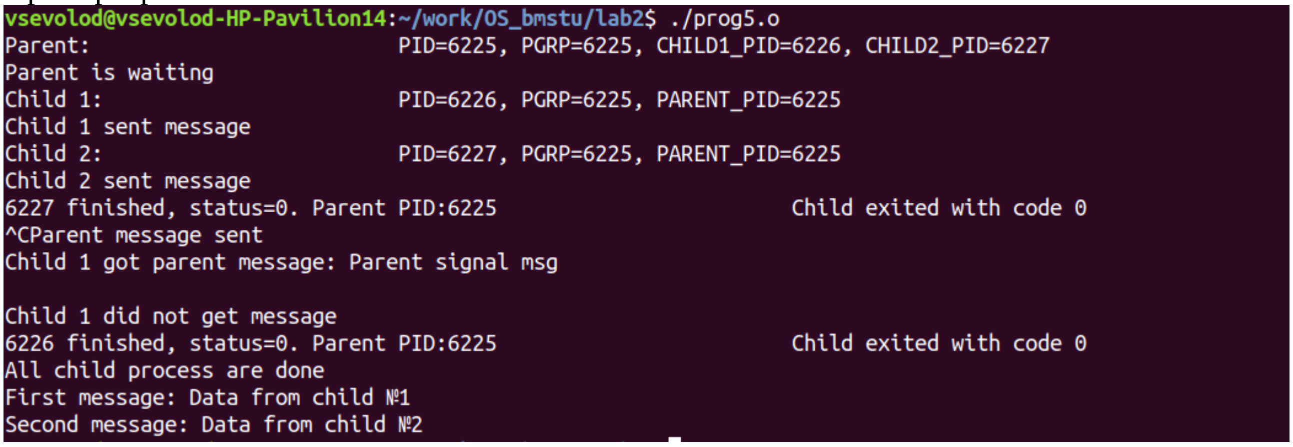


```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

#define MSG_SIZE    21
typedef void (*sighandler_t)(int);

int fd2[2];
int send_flag = 0;

void send_msg(int sig_n)
{
    send_flag = 1;
    if (getpid() == getpgrp())
    {
        write(fd2[1], "Parent signal msg", MSG_SIZE);
        printf("Parent message sent\n");
    }
}
int get_msg()
{
    char res[MSG_SIZE];
    ssize_t code = read(fd2[0], res, MSG_SIZE);
    if (code == -1)
        printf("Read error\n");
    else
        printf("Child 1 got parent message: %s\n", res);
    return code == -1;
}

int main(void)
{
    sighandler_t sg = signal(SIGINT, send_msg);
    if (sg == SIG_ERR)
    {
        perror("Can\'t create signal\n");
        return -1;
    }

    int fd[2];
    pid_t childPID1, childPID2;

    int pipe_code = pipe(fd);
    if (pipe_code < 0)
    {
        perror("Can\'t create pipe\n");
        return -1;
```

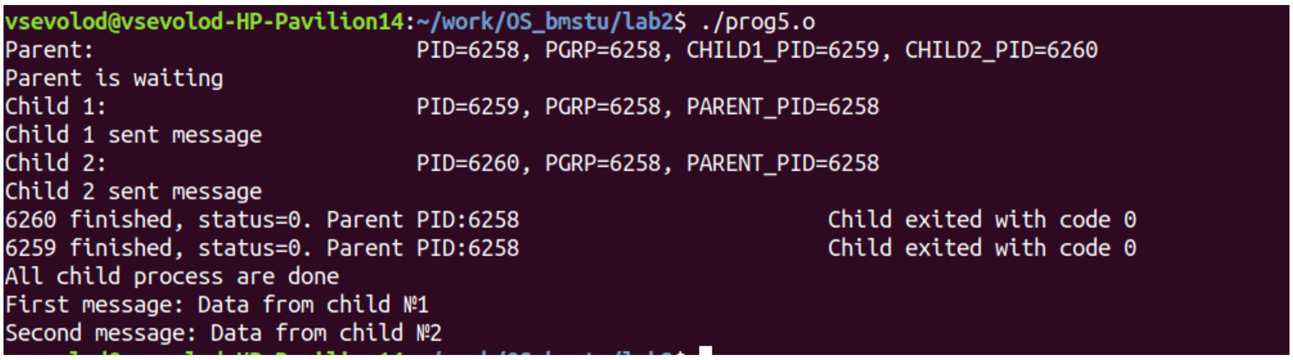


```c
    }
    pipe_code = pipe(fd2);
    if (pipe_code < 0)
    {
        perror("Can\'t create pipe\n");
        return -1;
    }


    childPID1 = fork();
    if (childPID1 == -1)
    {
        perror("Can't fork\n");
        return -1;
    }
    else if (!childPID1)
    {
        printf("Child 1:\t\t\tPID=%d, PGRP=%d, PARENT_PID=%d \n", getpid(), getpgrp(), getppid());

        ssize_t code = write(fd[1], "Data from child №1", MSG_SIZE);
        if (code == -1)
        {
            printf("Write error\n");
            return -1;
        }
        else
            printf("Child 1 sent message\n");

        sleep(4);

        if (send_flag)
            if (get_msg())  return -1;
        else
            printf("\nChild 1 did not get message\n");

        if (close(fd[1]) || close(fd[0]) || close(fd2[0]) || close(fd2[1]))
        {
            printf("Close error\n");
            return -1;
        }
        else
            return 0;
    }


    childPID2 = fork();
    if (childPID2 == 0)
    {
        printf("Child 2:\t\t\tPID=%d, PGRP=%d, PARENT_PID=%d \n", getpid(), getpgrp(), getppid());

        ssize_t code = write(fd[1], "Data from child №2", MSG_SIZE);
        if (code == -1)
        {
            printf("Write error\n");
            return -1;
        }
        else
            printf("Child 2 sent message\n");

        if (close(fd[1]) || close(fd[0]) || close(fd2[0]) || close(fd2[1]))
        {
            printf("Close error\n");
            return -1;
        }
        else
            return 0;

    }
    else if (childPID2 == -1)
    {
        perror("Can't fork\n");
        return -1;
    }

    printf("Parent:\t\t\t\tPID=%d, PGRP=%d, CHILD1_PID=%d, CHILD2_PID=%d \n", getpid(), getpgrp(), childPID1, childPID2);

    printf("Parent is waiting\n");
    int stat, child;
    for (int i=0; i<2; i++)
    {
        child = wait(&stat);
        if (child == -1)
        {
            printf("Wait returned with error\n");
            return -1;
        }
        else
        {
            printf("%d finished, status=%d. Parent PID:%d\t\t\t", child, stat, getpid());
            if (WIFEXITED(stat))
                printf("Child exited with code %d\n", WEXITSTATUS(stat));
            else
                printf("Child terminated abnormally\n");
        }
    }
    printf("All child process are done\n");

    char res1[MSG_SIZE], res2[MSG_SIZE];
    ssize_t code1 = read(fd[0], res1, MSG_SIZE);
    ssize_t code2 = read(fd[0], res2, MSG_SIZE);
    if (code1 == -1 || code1 == -1)
    {
        printf("Read error\n");
        return -1;
    }
    printf("First message: %s\n", res1);
    printf("Second message: %s\n", res2);

    if (close(fd[1]) || close(fd[0]) || close(fd2[0]) || close(fd2[1]))
    {
        printf("Close error\n");
        return -1;
    }
    else
        return 0;

}

```

Примеры работы:



```
vsevolod@vsevolod-HP-Pavilion14:~/work/OS_bmstu/lab2$ ./prog5.o
Parent:                         PID=6225, PGRP=6225, CHILD1_PID=6226, CHILD2_PID=6227
Parent is waiting
Child 1:                        PID=6226, PGRP=6225, PARENT_PID=6225
Child 1 sent message
Child 2:                        PID=6227, PGRP=6225, PARENT_PID=6225
Child 2 sent message
6227 finished, status=0. Parent PID:6225                        Child exited with code 0
^CParent message sent
Child 1 got parent message: Parent signal msg

Child 1 did not get message
6226 finished, status=0. Parent PID:6225                        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=6258, PGRP=6258, CHILD1_PID=6259, CHILD2_PID=6260
Parent is waiting
Child 1:                        PID=6259, PGRP=6258, PARENT_PID=6258
Child 1 sent message
Child 2:                        PID=6260, PGRP=6258, PARENT_PID=6258
Child 2 sent message
6260 finished, status=0. Parent PID:6258                        Child exited with code 0
6259 finished, status=0. Parent PID:6258                        Child exited with code 0
All child process are done
First message: Data from child №1
Second message: Data from child №2
```