|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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»

**Лабораторная работа 2-я по Unix**

|  |  |
| --- | --- |
| **Тема: «Процессы. Системные вызовы fork() и exec()»**  **Студент** Гарасев Н.А.  **Группа** ИУ7-52Б  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель** Рязанова Н.Ю. |  |

Москва.

2020 г.

Задание 1.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main()

{

    int child1 = fork();

    if (child1 == -1)

    {

        perror("Can't fork");

        exit(1);

    }

    else if (child1 == 0)

    {

        printf("Child: pid=%d, pidid=%d, groupid=%d\n", getpid(), getppid(), getpgrp());

        sleep(2);

        printf("Child: pid=%d, pidid=%d, groupid=%d\n", getpid(), getppid(), getpgrp());

        return 0;

    }

    printf("Parent: pid=%d, childpid=%d, groupid=%d\n", getpid(), child1, getpgrp());

    int child2 = fork();

    if (child2 == -1)

    {

        perror("Can't fork");

        exit(1);

    }

    else if (child2 == 0)

    {

        printf("Child: pid=%d, pidid=%d, groupid=%d\n", getpid(), getppid(), getpgrp());

        sleep(2);

        printf("Child: pid=%d, pidid=%d, groupid=%d\n", getpid(), getppid(), getpgrp());

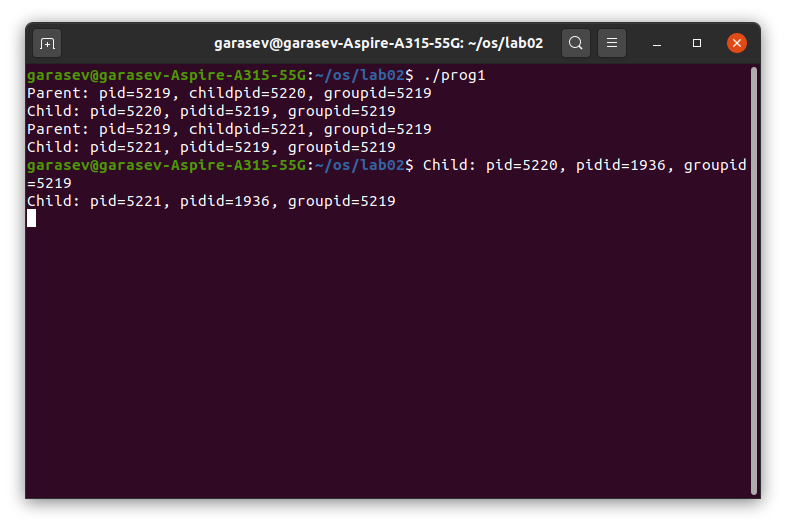
        return 0;

    }

    printf("Parent: pid=%d, childpid=%d, groupid=%d\n", getpid(), child2, getpgrp());

    return 0;

}



Задание 2.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

int main()

{

    int child1 = fork();

    if (child1 == -1)

    {

        perror("Can't fork");

        exit(1);

    }

    else if (child1 == 0)

    {

        printf("Child: pid=%d, pidid=%d, groupid=%d\n", getpid(), getppid(), getpgrp());

        return 0;

    }

    printf("Parent: pid=%d, childpid=%d, groupid=%d\n", getpid(), child1, getpgrp());

    int child2 = fork();

    if (child2 == -1)

    {

        perror("Can't fork");

        exit(1);

    }

    else if (child2 == 0)

    {

        printf("Child: pid=%d, pidid=%d, groupid=%d\n", getpid(), getppid(), getpgrp());

        return 0;

    }

    printf("Parent: pid=%d, childpid=%d, groupid=%d\n", getpid(), child2, getpgrp());

    if (child1 != 0 && child2 != 0)

    {

        int status1;

        pid\_t ret1 = wait(&status1);

        if (WIFEXITED(status1))

            printf("Parent: child %d finished with %d code.\n", ret1, WEXITSTATUS(status1));

        else if (WIFSIGNALED(status1))

            printf("Parent: child %d finished from signal with %d code.\n", ret1, WTERMSIG(status1));

        else if (WIFSTOPPED(status1))

            printf("Parent: child %d finished from signal with %d code.\n", ret1, WSTOPSIG(status1));

        int status2;

        pid\_t ret2 = wait(&status2);

        if (WIFEXITED(status2))

            printf("Parent: child %d finished with %d code.\n", ret2, WEXITSTATUS(status2));

        else if (WIFSIGNALED(status2))

            printf("Parent: child %d finished from signal with %d code.\n", ret2, WTERMSIG(status2));

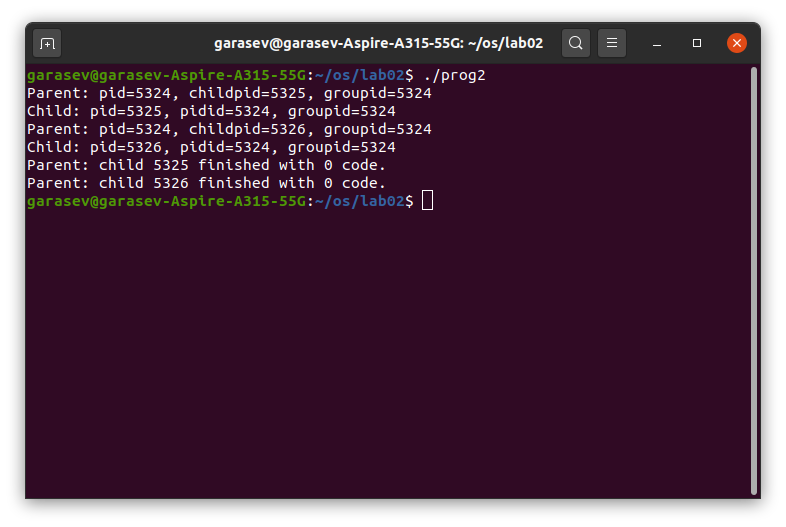
        else if (WIFSTOPPED(status2))

            printf("Parent: child %d finished from signal with %d code.\n", ret2, WSTOPSIG(status2));

    }

    return 0;

}



Задание 3.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

int main()

{

    int child1 = fork();

    if (child1 == -1)

    {

        perror("Can't fork");

        exit(1);

    }

    else if (child1 == 0)

    {

        printf("Child: pid=%d, pidid=%d, groupid=%d\n", getpid(), getppid(), getpgrp());

        if (execlp("/bin/ps", "ps", "al", (char\*)NULL) == -1)

        {

            perror("Child couldn't exec");

            exit(1);

        }

        return 0;

    }

    printf("Parent: pid=%d, childpid=%d, groupid=%d\n", getpid(), child1, getpgrp());

    int child2 = fork();

    if (child2 == -1)

    {

        perror("Can't fork");

        exit(1);

    }

    else if (child2 == 0)

    {

        printf("Child: pid=%d, pidid=%d, groupid=%d\n", getpid(), getppid(), getpgrp());

        if (execlp("/bin/ls", "ls", "-a", (char\*)NULL) == -1)

        {

            perror("Child couldn't exec");

            exit(1);

        }

        return 0;

    }

    printf("Parent: pid=%d, childpid=%d, groupid=%d\n", getpid(), child2, getpgrp());

    if (child1 != 0 && child2 != 0)

    {

        int status1;

        pid\_t ret1 = wait(&status1);

        if (WIFEXITED(status1))

            printf("Parent: child %d finished with %d code.\n", ret1, WEXITSTATUS(status1));

        else if (WIFSIGNALED(status1))

            printf("Parent: child %d finished from signal with %d code.\n", ret1, WTERMSIG(status1));

        else if (WIFSTOPPED(status1))

            printf("Parent: child %d finished from signal with %d code.\n", ret1, WSTOPSIG(status1));

        int status2;

        pid\_t ret2 = wait(&status2);

        if (WIFEXITED(status2))

            printf("Parent: child %d finished with %d code.\n", ret2, WEXITSTATUS(status2));

        else if (WIFSIGNALED(status2))

            printf("Parent: child %d finished from signal with %d code.\n", ret2, WTERMSIG(status2));

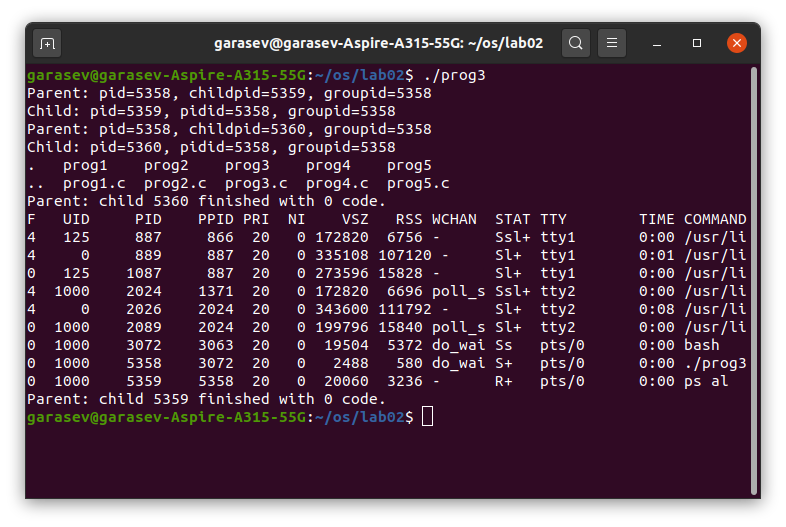
        else if (WIFSTOPPED(status2))

            printf("Parent: child %d finished from signal with %d code.\n", ret2, WSTOPSIG(status2));

    }

    return 0;

}



Задание 4.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <string.h>

int main()

{

    int fd[2];

    if (pipe(fd) == -1)

    {

        perror("Couldn't pipe");

        exit(1);

    }

    int child1 = fork();

    if (child1 == -1)

    {

        perror("Can't fork");

        exit(1);

    }

    else if (child1 == 0)

    {

        close(fd[0]);

        char msg1[] = "Message from child1";

        write(fd[1], msg1, 64);

        exit(0);

    }

    int child2 = fork();

    if (child2 == -1)

    {

        perror("Can't fork");

        exit(1);

    }

    else if (child2 == 0)

    {

        close(fd[0]);

        char msg2[] = "Message from child2";

        write(fd[1], msg2, 64);

        exit(0);

    }

    if (child1 != 0 && child2 != 0)

    {

        close(fd[1]);

        char msg1[64];

        read(fd[0], msg1, 64);

        char msg2[64];

        read(fd[0], msg2, 64);

        printf("Parent: read %s %s\n", msg1, msg2);

        int status1;

        pid\_t ret1 = wait(&status1);

        if (WIFEXITED(status1))

            printf("Parent: child %d finished with %d code.\n", ret1, WEXITSTATUS(status1));

        else if (WIFSIGNALED(status1))

            printf("Parent: child %d finished from signal with %d code.\n", ret1, WTERMSIG(status1));

        else if (WIFSTOPPED(status1))

            printf("Parent: child %d finished from signal with %d code.\n", ret1, WSTOPSIG(status1));

        int status2;

        pid\_t ret2 = wait(&status2);

        if (WIFEXITED(status2))

            printf("Parent: child %d finished with %d code.\n", ret2, WEXITSTATUS(status2));

        else if (WIFSIGNALED(status2))

            printf("Parent: child %d finished from signal with %d code.\n", ret2, WTERMSIG(status2));

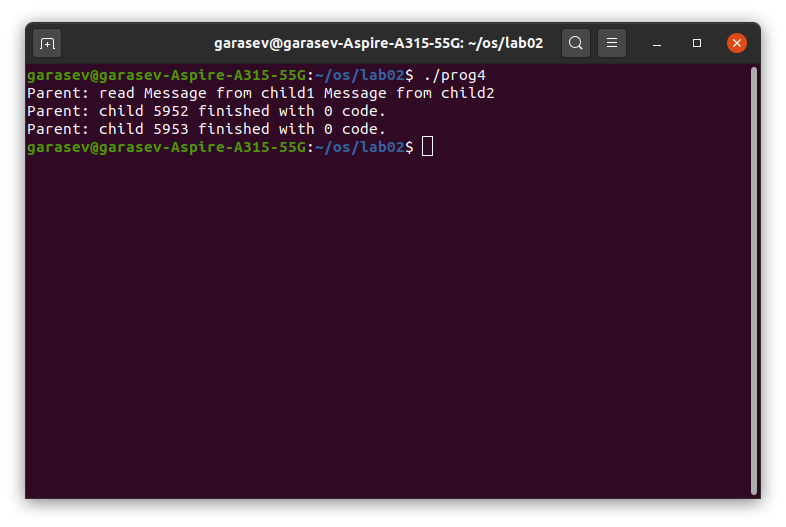
        else if (WIFSTOPPED(status2))

            printf("Parent: child %d finished from signal with %d code.\n", ret2, WSTOPSIG(status2));

    }

    return 0;

}



Задание 5.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <string.h>

#include <signal.h>

int flag = 0;

void catch\_sig(int sig)

{

    printf("catch signal %d\n", sig);

    flag = 1;

}

int main()

{

    int fd[2];

    if (pipe(fd) == -1)

    {

        perror("Couldn't pipe");

        exit(1);

    }

    void (\*old\_handler)(int) = signal(SIGINT, catch\_sig);

    int child1 = fork();

    if (child1 == -1)

    {

        perror("Can't fork");

        exit(1);

    }

    else if (child1 == 0)

    {

        sleep(5);

        if (flag)

        {

            close(fd[0]);

            char msg1[] = "Message from child1";

            write(fd[1], msg1, 64);

        }

        exit(0);

    }

    int child2 = fork();

    if (child2 == -1)

    {

        perror("Can't fork");

        exit(1);

    }

    else if (child2 == 0)

    {

        sleep(5);

        if (flag)

        {

            close(fd[0]);

            char msg2[] = "Message from child2";

            write(fd[1], msg2, 64);

        }

        exit(0);

    }

    if (child1 != 0 && child2 != 0)

    {

sleep(5);

        if (flag)

        {

            close(fd[1]);

            char msg1[64];

            read(fd[0], msg1, 64);

            char msg2[64];

            read(fd[0], msg2, 64);

            printf("Parent: read\n %s\n %s\n", msg1, msg2);

        }

        int status1;

        pid\_t ret1 = wait(&status1);

        if (WIFEXITED(status1))

            printf("Parent: child %d finished with %d code.\n", ret1, WEXITSTATUS(status1));

        else if (WIFSIGNALED(status1))

            printf("Parent: child %d finished from signal with %d code.\n", ret1, WTERMSIG(status1));

        else if (WIFSTOPPED(status1))

            printf("Parent: child %d finished from signal with %d code.\n", ret1, WSTOPSIG(status1));

        int status2;

        pid\_t ret2 = wait(&status2);

        if (WIFEXITED(status2))

            printf("Parent: child %d finished with %d code.\n", ret2, WEXITSTATUS(status2));

        else if (WIFSIGNALED(status2))

            printf("Parent: child %d finished from signal with %d code.\n", ret2, WTERMSIG(status2));

        else if (WIFSTOPPED(status2))

            printf("Parent: child %d finished from signal with %d code.\n", ret2, WSTOPSIG(status2));

    }

    return 0;

}

