Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Лабораторная работа №1 по курсу

«Операционные системы»

Группа: М8О-214БВ-24

Студент: Кириенко А. И.

Преподаватель Бахарев В. Д.

Оценка: _____

Дата: 01.10.24

Москва, 2024

Постановка задачи

Вариант 13.

Child1 переводит строки в нижний регистр. Child2 превращает все пробельные символы в символ «_».

Общий метод и алгоритм решения

Использованные системные вызовы:

- pid_t fork(void); – создает дочерний процесс.
- int pipe(int *fd); – создание неименованного канала для передачи данных между процессами.
- ssize_t write(int fd, const void *buf, size_t count); - запись данных из памяти процесса.
- ssize_t read(int fd, void *buf, size_t count); - чтение данных из файлового дескриптора.
- int dup2(int oldfd, int newfd); - переназначение файлового дескриптора.
- int close(int fd); - закрытие файлового дескриптора.
- pid_t wait(int *status); - ожидание завершения дочернего процесса.
- int execl (const char *Path, const char *Argument0, const char *Argument1, ... , NULL); - замена текущего изображения процесса.

В рамках лабораторной была написана и отлажена программа на языке C, осуществляющая работу с процессами и взаимодействие между ними в операционной системе Linux.

Я написала три отдельные программы parent.c, child1.c, child2.c. Родительский процесс создает два дочерних процесса с помощью системного вызова fork(), каждый из которых выполняет свою логику. child1 переводит строки в нижний регистр, child2 превращает все пробельные символы в символ "_". Для запуска программ child1 и child2 родительский процесс использует системный вызов execl().

Для организации взаимодействия процессов были созданы три неименнованых канала с помощью системного вызова pipe(). pd1 - связь родителя с первым дочерним процессом, pd2 - связь между двумя дочерними процессами, pd3 - связь второго дочернего процесса с родительским процессом.

Алгоритм работы программы:

Родительский процесс считывает строку введенную пользователем (с помощью системного вызова read()) и передает её первому дочернему процессу с помощью вызова write() в канал fd1.

Первый дочерний процесс получает данные через read(), преобразует строку в нижний регистр и передает результат второму дочернему процессу (вызов write() в канал fd2).

Второй дочерний процесс заменяет пробельные символы на "_" ("\t" считается как один пробельный символ) и передает окончательный результат родительскому процессу (write() в pd3).

Родительский процесс выводит результат на экран. С помощью системного вызова wait() он ждет окончания дочерних процессов.

Код программы

parent.c

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


int main(){
    int fd1[2];
    int fd2[2];
    int fd3[2];

    if (pipe(fd1) == -1){ //создаем канал
        const char *err1 = "Error: failed to create pipe1\n";
        write(2, err1, strlen(err1));
        return 1;
    }

    if (pipe(fd2) == -1){
        const char *err2 = "Error: failed to creat pipe2\n";
        write(2, err2, strlen(err2));
        return 1;
    }

    if (pipe(fd3) == -1){
        const char *err3 = "Error: failed to create pipe3\n";
        write (2, err3, strlen(err3));
        return 1;
    }
    pid_t pid1 = fork(); //создаем дочерний процесс 1
```

```
    if (pid1 == -1){
        const char *error1 = "Error: failed to spawn new process1\n";
        write (2, error1, strlen(error1));
        return 1;
    } else if (pid1 == 0){
        if (dup2(fd1[0], STDIN_FILENO) == -1){
            const char *er5 = "Error: failed to dup2 fd1 STDIN\n";
            write(2, er5, strlen(er5));
            return 1;
        }

        if (dup2(fd2[1], STDOUT_FILENO) == -1){
            const char *er6 = "Error: failed to dup2 fd2 STDOUT\n";
            write(2, er6, strlen(er6));
            return 1;
        }

        if (close(fd1[1]) == -1){
            const char *e1 = "Error: failed to close fd1[1] in child1\n";
            write(2, e1, strlen(e1));
            return 1;
        }
        if (close(fd2[0]) == -1){
            const char *e2 = "Error: failed to close fd2[0] in child1\n";
            write(2, e2, strlen(e2));
            return 1;
        }
        if (close(fd3[0]) == -1){
            const char *e3 = "Error: failed to close fd3[0] in child1\n";
```

```
            write(2, e3, strlen(e3));
            return 1;
        }
        if (close(fd3[1]) == -1){
            const char *e4 = "Error: failed to close fd3[1] in child1\n";
            write(2, e4, strlen(e4));
            return 1;
        }

        if (execl("./child1", "child1", (char *)NULL) == -1){
            const char *erro1 = "Error: failed to execl child1\n";
            write(2, erro1, strlen(erro1));
            return 1;
        }
    }

    pid_t pid2 = fork(); //создаем дочерний процесс 2
    if (pid2 == -1){
        const char *error2 = "Error: failed to spawn new process2\n";
        write(2, error2, strlen(error2));
        return 1;
    }
    if (pid2 == 0){
        if (dup2(fd2[0], STDIN_FILENO) == -1){
            const char *er3 = "Error: failed to dup2 fd2 STDIN\n";
            write(2, er3, strlen(er3));
            return 1;
        }
```

```
    if (dup2(fd3[1], STDOUT_FILENO) == -1){
        const char *er4 = "Error: failed to dup2 fd3 STDOUT\n";
        write(2, er4, strlen(er4));
        return 1;
    }

    if (close(fd1[0]) == -1){
        const char *e5 = "Error: failed to close fd1[0] in child2\n";
        write(2, e5, strlen(e5));
        return 1;
    }
    if (close(fd1[1]) == -1){
        const char *e6 = "Error: failed to close fd1[1] in child2\n";
        write(2, e6, strlen(e6));
        return 1;

    }
    if (close(fd2[1]) == -1){
        const char *e7 = "Error: failed to close fd2[1] in child2\n";
        write(2, e7, strlen(e7));
        return 1;
    }
    if (close(fd3[0]) == -1){
        const char *e8 = "Error: failed to close fd3[0] in child2\n";
        write(2, e8, strlen(e8));
        return 1;
    }

    if (execl("./child2", "child2", (char *)NULL) == -1) {
```

```
            const char *erro2 = "Error: failed to execl child2\n";
            write(2, erro2, strlen(erro2));
            return 1;
        }
    }


    if (close(fd1[0]) == -1){
        const char *e9 = "Error: failed to close fd1[0] in parent\n";
        write(2, e9, strlen(e9));
        return 1;
    }


    if (close(fd2[0]) == -1){
        const char *e10 = "Error: failed to close fd2[0] in parent\n";
        write(2, e10, strlen(e10));
        return 1;
    }
    if (close(fd2[1]) == -1){
        const char *e11 = "Error: failed to close fd2[1] in parent\n";
        write(2, e11, strlen(e11));
        return 1;
    }
    if (close(fd3[1]) == -1){
        const char *e12 = "Error: failed to close fd3[1] in parent\n";
        write(2, e12, strlen(e12));
        return 1;
    }


    char input[1024];
```

```
    ssize_t r;
    char buf2[1024];
    ssize_t w1;

    while ((r = read(STDIN_FILENO, input, sizeof(input))) > 0){ //читаем
пользователя
        if (write(fd1[1], input, r) == -1){ //отправляем ребенку 1
            const char *er1 = "Error: failed to write to fd1 in parent\n";
            write(2, er1, strlen(er1));
            return 1;
        }
        //читаем от ребенка 2
        w1 = read(fd3[0], buf2, sizeof(buf2));
        if (w1 == -1){
            const char *error4 = "Error: failed to read from fd3\n";
            write(2, error4, strlen(error4));
            return 1;
        }
        if (w1 > 0) {
            if (write (1, buf2, w1) == -1){ //выводим итог
                const char *er2 = "Error: failed to write to stdout in parent\n";
                write(2, er2, strlen(er2));
                return 1;

            }
        }
    }

    if (r == -1){
```

```
        const char *error3 = "Error: failed to read from stdin\n";
        write(2, error3, strlen(error3));
        return 1;
    }

    if (close(fd1[1]) == -1){
        const char *e13 = "Error: failed to close fd1[1] in parent\n";
        write(2, e13, strlen(e13));
        return 1;
    }
    if (close(fd3[0]) == -1){
        const char *e14 = "Error: failed to close fd3[0] in parent\n";
        write(2, e14, strlen(e14));
        return 1;
    }
    if (wait(NULL) == -1) {
        const char *erro3 = "Error: failed to wait child\n";
        write (2, erro3, strlen(erro3));
        return 1;
    }
    if (wait(NULL) == -1) {
        const char *erro3 = "Error: failed to wait child\n";
        write (2, erro3, strlen(erro3));
        return 1;
    }
    return 0;
}
```

child1.c

```c
#include <unistd.h>
#include <ctype.h> //для нижнего регистра tolower
#include <string.h>


int main(){
    char buf[1024];
    ssize_t r1;

    while ((r1 = read(STDIN_FILENO, buf, sizeof(buf))) > 0){
        for (ssize_t i = 0; i < r1; i++){
            buf[i] = tolower((unsigned char)buf[i]);

        }
        if (write(STDOUT_FILENO, buf, r1) == -1){
            const char *err = "Error: failed to write to fd2 in child1\n";
            write(2, err, strlen(err));
            return 1;
        }
    }

    if (r1 == -1){
        const char *err2 = "Error: failed to read from fd1\n";
        write(2, err2, strlen(err2));
        return 1;
    }
    return 0;
}
```

Child2.c

```c
#include <unistd.h>
#include <string.h>
int main(){
    char buf[1024];
    ssize_t r1;

    while ((r1 = read(STDIN_FILENO, buf, sizeof(buf))) > 0){
        for (ssize_t i = 0; i < r1; i++){
            if (buf[i] == ' '  || buf[i] == '\t'){
                buf[i] = '_';
            }
        }
        if (write(STDOUT_FILENO, buf, r1) == -1){
            const char *err = "Error: failed to write to fd3 in child2\n";
            write(2, err, strlen(err));
            return 1;
        }
    }
    if (r1 == -1){
        const char *err2 = "Error: failed to read from fd2\n";
        write(2, err2, strlen(err2));
        return 1;
    }
    return 0;
}
```

Протокол работы программы

Тестирование:

```
nastik@HUAWEI:/mnt/c/Users/kp982/OneDrive/Рабочий стол/настя/лаба1 OS$ gcc parent.c -o parent
nastik@HUAWEI:/mnt/c/Users/kp982/OneDrive/Рабочий стол/настя/лаба1 OS$ gcc child1.c -o child1
nastik@HUAWEI:/mnt/c/Users/kp982/OneDrive/Рабочий стол/настя/лаба1 OS$ gcc child2.c -o child2
nastik@HUAWEI:/mnt/c/Users/kp982/OneDrive/Рабочий стол/настя/лаба1 OS$ ./parent
HHHeeeLo KKKK I`m FINE
hhheeelo_kkkk_i`m_fine
SSSSSSSSSISISISIssssssssssssssssssssssssssssssssssssssssssssss
sssssssssisisisissssssssssssssssssssssssssssssssssssssssssssss
DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddd
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF   KKKK ooopppdsjdjdjd wwwwww RRRR                PP
ffffffffffffffffffffffffffffffffffffff___kkkk_ooopppdsjdjdjd_wwwwww_rrrr__pp
nastik@HUAWEI:/mnt/c/Users/kp982/OneDrive/Рабочий стол/настя/лаба1 OS$ ./parent
222 &&&& PaPa mama DAD SISTER
222_&&&&_papa_mama_dad_sister
MAMA    PAPA    SISTER  meeee
mama_papa_sister_meeee
```

.

Strace:

nastik@HUAWEI:/mnt/c/Users/kp982/OneDrive/Рабочий стол/настя/лаба1 OS$ strace -f ./parent

execve("./parent", ["./parent"], 0x7fffb9c9c188 /* 25 vars */) = 0

brk(NULL) = 0x57f6df3ca000

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7c160ad6b000

access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

fstat(3, {st_mode=S_IFREG|0644, st_size=19151, ...}) = 0

mmap(NULL, 19151, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7c160ad66000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0\0"..., 832) = 832

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7c160aa00000

mmap(0x7c160aa28000, 1605632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7c160aa28000

mmap(0x7c160abb0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1b0000) = 0x7c160abb0000

mmap(0x7c160abff000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x7c160abff000

mmap(0x7c160ac05000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7c160ac05000

close(3)   = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7c160ad63000

arch_prctl(ARCH_SET_FS, 0x7c160ad63740) = 0

set_tid_address(0x7c160ad63a10)   = 1595

set_robust_list(0x7c160ad63a20, 24)   = 0

rseq(0x7c160ad64060, 0x20, 0, 0x53053053) = 0

mprotect(0x7c160abff000, 16384, PROT_READ) = 0

mprotect(0x57f6bfe12000, 4096, PROT_READ) = 0

mprotect(0x7c160ada3000, 8192, PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

munmap(0x7c160ad66000, 19151)   = 0

pipe2([3, 4], 0)   = 0

pipe2([5, 6], 0)   = 0

pipe2([7, 8], 0)   = 0

clone(child_stack=NULL,
**flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace: Process 1596
attached**

, child_tidptr=0x7c160ad63a10) = 1596

[pid 1596] set_robust_list(0x7c160ad63a20, 24 <unfinished ...>

**[pid 1595] clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD <unfinished ...>**

[pid 1596] <... set_robust_list resumed>) = 0

[pid 1596] dup2(3, 0strace: Process 1597 attached

<unfinished ...>

[pid 1595] <... clone resumed>, child_tidptr=0x7c160ad63a10) = 1597

[pid 1596] <... dup2 resumed>)   = 0

[pid 1597] set_robust_list(0x7c160ad63a20, 24 <unfinished ...>

[pid 1595] close(3 <unfinished ...>

[pid 1596] dup2(6, 1 <unfinished ...>

[pid 1595] <... close resumed>)   = 0

[pid 1597] <... set_robust_list resumed>) = 0

[pid 1595] close(5 <unfinished ...>

[pid 1596] <... dup2 resumed>)   = 1

[pid 1595] <... close resumed>)   = 0

[pid 1597] dup2(5, 0 <unfinished ...>

[pid 1595] close(6 <unfinished ...>

[pid 1596] close(4 <unfinished ...>

[pid 1595] <... close resumed>)   = 0

[pid 1597] <... dup2 resumed>)   = 0

[pid 1595] close(8 <unfinished ...>

[pid 1596] <... close resumed>)   = 0

[pid 1595] <... close resumed>)   = 0

[pid 1597] dup2(8, 1 <unfinished ...>

[pid 1595] read(0, <unfinished ...>

[pid 1596] close(5 <unfinished ...>

[pid 1597] <... dup2 resumed>)   = 1

[pid 1596] <... close resumed>)   = 0

[pid 1597] close(3 <unfinished ...>

[pid 1596] close(7 <unfinished ...>

[pid 1597] <... close resumed>)   = 0

[pid 1596] <... close resumed>)   = 0

[pid 1597] close(4 <unfinished ...>

[pid 1596] close(8 <unfinished ...>

[pid 1597] <... close resumed>)   = 0

[pid 1596] <... close resumed>)   = 0

[pid 1597] close(6 <unfinished ...>

[pid 1596] execve("./child1", ["child1"], 0x7ffcd8d4aab8 /* 25 vars */ <unfinished ...>

[pid 1597] <... close resumed>)   = 0

[pid 1597] close(7)   = 0

[pid 1597] execve("./child2", ["child2"], 0x7ffcd8d4aab8 /* 25 vars */) = 0

[pid 1597] brk(NULL <unfinished ...>

[pid 1596] <... execve resumed>)   = 0

[pid 1597] <... brk resumed>)   = 0x617540ac6000

[pid 1596] brk(NULL <unfinished ...>

[pid 1597] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 1596] <... brk resumed>)   = 0x5b59a99ca000

[pid 1597] <... mmap resumed>)   = 0x7e0dea3f6000

[pid 1597] access("/etc/ld.so.preload", R_OK <unfinished ...>

[pid 1596] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 1597] <... access resumed>)   = -1 ENOENT (No such file or directory)

[pid 1596] <... mmap resumed>)   = 0x72e7e6b3d000

[pid 1597] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>

[pid 1596] access("/etc/ld.so.preload", R_OK <unfinished ...>

[pid 1597] <... openat resumed>)   = 3

[pid 1596] <... access resumed>)   = -1 ENOENT (No such file or directory)

[pid 1597] fstat(3, <unfinished ...>

[pid 1596] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>

[pid 1597] <... fstat resumed>{st_mode=S_IFREG|0644, st_size=19151, ...}) = 0

[pid 1596] <... openat resumed>)   = 4

[pid 1597] mmap(NULL, 19151, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>

[pid 1596] fstat(4, <unfinished ...>

[pid 1597] <... mmap resumed>)   = 0x7e0dea3f1000

[pid 1596] <... fstat resumed>{st_mode=S_IFREG|0644, st_size=19151, ...}) = 0

[pid 1597] close(3 <unfinished ...>

[pid 1596] mmap(NULL, 19151, PROT_READ, MAP_PRIVATE, 4, 0 <unfinished ...>

[pid 1597] <... close resumed>) = 0

[pid 1596] <... mmap resumed>) = 0x72e7e6b38000

[pid 1597] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC <unfinished ...>

[pid 1596] close(4 <unfinished ...>

[pid 1597] <... openat resumed>) = 3

[pid 1596] <... close resumed>) = 0

[pid 1597] read(3, <unfinished ...>

[pid 1596] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC <unfinished ...>

[pid 1597] <... read resumed>"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0"..., 832) = 832

[pid 1596] <... openat resumed>) = 4

[pid 1597] pread64(3, <unfinished ...>

[pid 1596] read(4, <unfinished ...>

[pid 1597] <... pread64 resumed>"\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

[pid 1596] <... read resumed>"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0"..., 832) = 832

[pid 1597] fstat(3, <unfinished ...>

[pid 1596] pread64(4, <unfinished ...>

[pid 1597] <... fstat resumed>{st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0

[pid 1596] <... pread64 resumed>"\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

[pid 1597] pread64(3, <unfinished ...>

[pid 1596] fstat(4, <unfinished ...>

[pid 1597] <... pread64 resumed>"\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

[pid 1596] <... fstat resumed>{st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0

[pid 1597] mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished ...>

[pid 1596] pread64(4, <unfinished ...>

[pid 1597] <... mmap resumed>)   = 0x7e0dea000000

[pid 1596] <... pread64 resumed>"\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

[pid 1597] mmap(0x7e0dea028000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000 <unfinished ...>

[pid 1596] mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 4, 0 <unfinished ...>

[pid 1597] <... mmap resumed>)   = 0x7e0dea028000

[pid 1596] <... mmap resumed>)   = 0x72e7e6800000

[pid 1597] mmap(0x7e0dea1b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000 <unfinished ...>

[pid 1596] mmap(0x72e7e6828000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x28000 <unfinished ...>

[pid 1597] <... mmap resumed>)   = 0x7e0dea1b0000

[pid 1596] <... mmap resumed>)   = 0x72e7e6828000

[pid 1597] mmap(0x7e0dea1ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000 <unfinished ...>

[pid 1596] mmap(0x72e7e69b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x1b0000 <unfinished ...>

[pid 1597] <... mmap resumed>)   = 0x7e0dea1ff000

[pid 1596] <... mmap resumed>)   = 0x72e7e69b0000

[pid 1597] mmap(0x7e0dea205000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 1596] mmap(0x72e7e69ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x1fe000 <unfinished ...>

[pid 1597] <... mmap resumed>)   = 0x7e0dea205000

[pid 1596] <... mmap resumed>)   = 0x72e7e69ff000

[pid 1597] close(3)   = 0

[pid 1596] mmap(0x72e7e6a05000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 1597] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 1596] <... mmap resumed>)   = 0x72e7e6a05000

[pid 1597] <... mmap resumed>)   = 0x7e0dea3ee000

[pid 1597] arch_prctl(ARCH_SET_FS, 0x7e0dea3ee740 <unfinished ...>

[pid 1596] close(4 <unfinished ...>

[pid 1597] <... arch_prctl resumed>)   = 0

[pid 1596] <... close resumed>)   = 0

[pid 1597] set_tid_address(0x7e0dea3eea10 <unfinished ...>

[pid 1596] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 1597] <... set_tid_address resumed>) = 1597

[pid 1596] <... mmap resumed>)   = 0x72e7e6b35000

[pid 1597] set_robust_list(0x7e0dea3eea20, 24 <unfinished ...>

[pid 1596] arch_prctl(ARCH_SET_FS, 0x72e7e6b35740 <unfinished ...>

[pid 1597] <... set_robust_list resumed>) = 0

[pid 1596] <... arch_prctl resumed>)   = 0

[pid 1597] rseq(0x7e0dea3ef060, 0x20, 0, 0x53053053 <unfinished ...>

[pid 1596] set_tid_address(0x72e7e6b35a10 <unfinished ...>

[pid 1597] <... rseq resumed>)   = 0

[pid 1596] <... set_tid_address resumed>) = 1596

[pid 1596] set_robust_list(0x72e7e6b35a20, 24 <unfinished ...>

[pid 1597] mprotect(0x7e0dea1ff000, 16384, PROT_READ <unfinished ...>

[pid 1596] <... set_robust_list resumed>) = 0

[pid 1597] <... mprotect resumed>)   = 0

[pid 1596] rseq(0x72e7e6b36060, 0x20, 0, 0x53053053 <unfinished ...>

[pid 1597] mprotect(0x61753c6a7000, 4096, PROT_READ <unfinished ...>

[pid 1596] <... rseq resumed>)   = 0

[pid 1597] <... mprotect resumed>)   = 0

[pid 1597] mprotect(0x7e0dea42e000, 8192, PROT_READ <unfinished ...>

[pid 1596] mprotect(0x72e7e69ff000, 16384, PROT_READ <unfinished ...>

[pid 1597] <... mprotect resumed>)   = 0

[pid 1596] <... mprotect resumed>)   = 0

[pid 1597] prlimit64(0, RLIMIT_STACK, NULL, <unfinished ...>

[pid 1596] mprotect(0x5b59904d9000, 4096, PROT_READ <unfinished ...>

[pid 1597] <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

[pid 1596] <... mprotect resumed>)   = 0

[pid 1597] munmap(0x7e0dea3f1000, 19151 <unfinished ...>

[pid 1596] mprotect(0x72e7e6b75000, 8192, PROT_READ <unfinished ...>

[pid 1597] <... munmap resumed>)   = 0

[pid 1596] <... mprotect resumed>)   = 0

[pid 1597] read(0, <unfinished ...>

[pid 1596] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0

[pid 1596] munmap(0x72e7e6b38000, 19151) = 0

[pid 1596] read(0, HELLLLOOOOOO

<unfinished ...>

[pid 1595] <... read resumed>"\320\240HELLLLOOOOOO\n", 1024) = 15

[pid 1595] write(4, "\320\240HELLLLOOOOOO\n", 15) = 15

[pid 1596] <... read resumed>"\320\240HELLLLOOOOOO\n", 1024) = 15

[pid 1595] read(7, <unfinished ...>

[pid 1596] write(1, "\320\240hellllooooo\n", 15 <unfinished ...>

[pid 1597] <... read resumed>"\320\240hellllooooo\n", 1024) = 15

[pid 1596] <... write resumed>)   = 15

[pid 1597] write(1, "\320\240hellllooooo\n", 15 <unfinished ...>

[pid 1596] read(0, <unfinished ...>

[pid 1595] <... read resumed>"\320\240hellllooooo\n", 1024) = 15

[pid 1597] <... write resumed>)   = 15

[pid 1595] write(1, "\320\240hellllooooo\n", 15 <unfinished ...>

Phellllooooo

[pid 1597] read(0, <unfinished ...>

[pid 1595] <... write resumed>)   = 15

[pid 1595] read(0, "", 1024)   = 0

[pid 1595] close(4) = 0

[pid 1596] <... read resumed>"", 1024) = 0

[pid 1595] close(7) = 0

[pid 1596] exit_group(0 <unfinished ...>

[pid 1595] wait4(-1, <unfinished ...>

[pid 1596] <... exit_group resumed>) = ?

[pid 1597] <... read resumed>"", 1024) = 0

[pid 1597] exit_group(0) = ?

[pid 1596] +++ exited with 0 +++

[pid 1595] <... wait4 resumed>NULL, 0, NULL) = 1596

[pid 1595] --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=1596, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---

[pid 1595] wait4(-1, <unfinished ...>

[pid 1597] +++ exited with 0 +++

<... wait4 resumed>NULL, 0, NULL) = 1597

--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=1597, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---

exit_group(0) = ?

+++ exited with 0 +++

Вывод

В ходе выполнения лабораторной работы я получила новые знания и навыки в области работы процессами и системными вызовами. В результате работы программа создала два дочерних процесса для решения поставленной задачи. Трудностей у меня не возникало, за исключением выделения тех вызовов, которые относятся именно к моей программе.