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-210Б-23

Студент: Жданович Е.Т. Преподаватель: Бахарев В.Д. Оценка:

Дата: 26.12.24

Москва, 2024

# Постановка задачи

**Вариант 11.**

Составить и отладить программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними в одной из двух операционных систем. Родительский процесс создает два дочерних процесса. Child1 и Child2 можно

«соединить» между собой дополнительным каналом. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1.

Процесс child1 и child2 производят работу над строками. Child2 пересылает результат своей работы родительскому процессу. Родительский процесс полученный результат выводит в стандартный поток вывода. 11 вариант) Child1 переводит строки в верхний регистр. Child2 превращает все пробельные символы в символ «\_».

# Общий метод и алгоритм решения

Использованные системные вызовы:

* pid\_t fork(void) – используется для создания дочернего процесса.
* int pipe(int fd) создает канал для однонаправленной связи между процессами. fd[0] используется для чтения из канала, a fd[1] — для записи в него.
* ssize\_t write(int fd, const void buf, size\_t count) записывает данные из буфера buf в файл, связанный с файловым дескриптором fd, в количестве байтов, указанном в count.
* ssize\_t read(int fd, void buf, size\_t count) читает данные из файла или канала, связанного с файловым дескриптором fd, в буфер buf в количестве байтов, указанном в count.
* int execv(const char path, char const argv[]) заменяет текущий процесс новым процессом, запускающим указанную программу.
* int32\_t open(const char\* file, int oflag, ...); – открывает файл и возвращает файловый дескриптор.
* int close int fd) — закрывает файл.
* int dup2(int oldfd, int newfd) дублирует файловый дескриптор oldfd, заменяя им дескриптор newfd. Перенаправление стандартного ввода дочернего процесса на канал.
* int wait(int status) приостанавливает выполнение родительского процесса до завершения дочернего процесса.

В рамках лабораторной работы я реализовала три программы на С: Parent.c, Child1.c и Child2.c, которые работают совместно для выполнения работы над строками. Программа Parent.c запрашивает у пользователя входные данные, создает ріре для связи с дочерними процессами и использует fork() для их создания. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их Child1, который переводит строки в верхний регистр. После чего Child2 получает данные от Child1 и превращает все пробельные символы в символ «\_».

В случае ошибок, таких как неверный ввод или деление на ноль, программа выводит соответствующие сообщения об ошибках. Также родительский процесс ожидает завершение дочерних процессов после вывода результатов.

# Код программы

**Parent.c**

#include <stdint.h> #include <stdlib.h> #include <sys/wait.h> #include <unistd.h>

#include <string.h>

static char CHILD1\_PROGRAM\_NAME[] = "./child1"; static char CHILD2\_PROGRAM\_NAME[] = "./child2"; int main(int argc, char \*\*argv) {

if (argc != 1) { char msg[] = "usage: ./{filename}\n"; write(STDOUT\_FILENO, msg, strlen(msg)); exit(EXIT\_SUCCESS); }

char progpath[1024];

{

ssize\_t len = readlink("/proc/self/exe", progpath,

sizeof(progpath) - 1);

if (len == -1) {

const char msg[] = "error: failed to read full program path\n"; write(STDERR\_FILENO, msg, sizeof(msg));

exit(EXIT\_FAILURE);

}

while (progpath[len] != '/')

--len;

progpath[len + 1] = '\0';

}

int pipe1[2], pipe2[2], pipe3[2];

if (pipe(pipe1) == -1 || pipe(pipe2) == -1 || pipe(pipe3) == -1) { const char msg[] = "error: failed to create pipe\n"; write(STDERR\_FILENO, msg, sizeof(msg));

exit(EXIT\_FAILURE);

}

const pid\_t child1 = fork();

switch (child1) { case -1: {

const char msg[] = "error: failed to spawn new process\n"; write(STDERR\_FILENO, msg, sizeof(msg)); exit(EXIT\_FAILURE);

} break; case 0: {

dup2(pipe1[STDIN\_FILENO], STDIN\_FILENO); dup2(pipe2[STDOUT\_FILENO], STDOUT\_FILENO);

close(pipe1[STDOUT\_FILENO]); close(pipe2[STDIN\_FILENO]); close(pipe3[STDIN\_FILENO]); close(pipe3[STDOUT\_FILENO]);

{

char \*const args[] = {CHILD1\_PROGRAM\_NAME, NULL}; int32\_t status = execv(CHILD1\_PROGRAM\_NAME, args); if (status == -1) {

const char msg[] = "error: failed to exec into new

exectuable image\n";

write(STDERR\_FILENO, msg, sizeof(msg)); exit(EXIT\_FAILURE);

}

}

} break;

}

const pid\_t child2 = fork(); switch (child2) {

case -1: {

const char msg[] = "error: failed to spawn new process\n"; write(STDERR\_FILENO, msg, sizeof(msg)); exit(EXIT\_FAILURE);

} break; case 0: {

dup2(pipe2[STDIN\_FILENO], STDIN\_FILENO); dup2(pipe3[STDOUT\_FILENO], STDOUT\_FILENO);

close(pipe1[STDIN\_FILENO]); close(pipe1[STDOUT\_FILENO]);

close(pipe2[STDOUT\_FILENO]); close(pipe3[STDIN\_FILENO]);

{

char \*const args[] = {CHILD2\_PROGRAM\_NAME, NULL}; int32\_t status = execv(CHILD2\_PROGRAM\_NAME, args); if (status == -1) {

const char msg[] = "error: failed to exec into new

exectuable image\n";

write(STDERR\_FILENO, msg, sizeof(msg)); exit(EXIT\_FAILURE);

}

}

} break;

}

close(pipe1[0]);

close(pipe2[0]);

close(pipe3[1]);

ssize\_t bytes; char buf[1024];

char msg\_of\_hint[] = "Enter your string or (Enter / CTRL + D) for stop: \n"; int len\_of\_msg\_of\_hint = strlen(msg\_of\_hint);

write(STDOUT\_FILENO, msg\_of\_hint, len\_of\_msg\_of\_hint); while (bytes = read(STDIN\_FILENO, buf, sizeof(buf))) {

if (bytes < 0) {

const char msg[] = "error: failed to read from stdin\n"; write(STDERR\_FILENO, msg, sizeof(msg)); exit(EXIT\_FAILURE);

} else if (buf[0] == '\n') { break;

}

buf[bytes] = '\0';

write(pipe1[1], buf, strlen(buf));

char result[1024];

ssize\_t bytes\_read = read(pipe3[0], result, sizeof(result) - 1); if (bytes\_read > 0) {

result[bytes\_read] = '\0';

char msg[] = "Processed result: "; write(STDOUT\_FILENO, msg, strlen(msg)); write(STDOUT\_FILENO, result, bytes\_read - 1); write(STDOUT\_FILENO, "\n\n", 2);

write(STDOUT\_FILENO, msg\_of\_hint, len\_of\_msg\_of\_hint);

}

}

close(pipe1[1]);

close(pipe3[0]);

close(pipe2[1]);

wait(NULL); wait(NULL);

return 0;

}

# Child1.c

#include <ctype.h> #include <unistd.h> #include <string.h> int main() {

char input[1024]; ssize\_t bytes\_read;

while ((bytes\_read = read(STDIN\_FILENO, input, sizeof(input))) > 0) {

input[bytes\_read] = '\0';

for (int i = 0; i < bytes\_read; i++) { input[i] = toupper(input[i]);

}

write(STDOUT\_FILENO, input, bytes\_read);

}

return 0;

}

# Child2.c

#include <unistd.h> #include <string.h> int main() {

char input[1024]; ssize\_t bytes\_read;

while ((bytes\_read = read(STDIN\_FILENO, input, sizeof(input))) > 0) {

input[bytes\_read] = '\0';

for (int i = 0; i < bytes\_read; i++) {

if (input[i] == ' ') {

input[i] = '\_';

}

}

write(STDOUT\_FILENO, input, bytes\_read);

}

return 0;

}

# Протокол работы программы

## Тестирование:

Enter your string or (Enter / CTRL + D) for stop: ijon kj lk Processed result: IJON\_KJ\_LK

Enter your string or (Enter / CTRL + D) for stop: knm kl kjl Processed result: KNM\_KL\_KJL

Enter your string or (Enter / CTRL + D) for stop: Hello you Processed result: HELLO\_YOU

## Strace:

lizka@LizaAlisa:~/ЛАБЫ\_ОС/Лаба1$ strace -f ./parent

execve("./parent", ["./parent"], 0x7ffd3fb6b5a8 /\* 26 vars */) = 0*

*brk(NULL) = 0x563b94154000*

*mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f18b0dd5000*

*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=19163,...})=0*

*mmap(NULL,19163,PROT\_READ,MAP\_PRIVATE,3,0)=0x7f18b0dd0000*  
***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\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)=0x7f18b0bbe000*  
*mmap(0x7f18b0be6000,1605632,PROT\_READ|PROT\_EXEC,MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE,3,0x28000)=0x7f18b0be6000*  
*mmap(0x7f18b0d6e000,323584,PROT\_READ,MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE,3,0x1b0000)=0x7f18b0d6e000*  
*mmap(0x7f18b0dbd000,24576,PROT\_READ|PROT\_WRITE,MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE,3,0x1fe000)=0x7f18b0dbd000*  
*mmap(0x7f18b0dc3000,52624,PROT\_READ|PROT\_WRITE,MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS,-1,0)=0x7f18b0dc3000*

***close(3)=0***  
*mmap(NULL,12288,PROT\_READ|PROT\_WRITE,MAP\_PRIVATE|MAP\_ANONYMOUS,-1,0)=0x7f18b0bbb000*  
*arch\_prctl(ARCH\_SET\_FS,0x7f18b0bbb740)=0*  
*set\_tid\_address(0x7f18b0bbba10)=1424*  
*set\_robust\_list(0x7f18b0bbba20,24)=0*  
*rseq(0x7f18b0bbc060,0x20,0,0x53053053)=0*  
*mprotect(0x7f18b0dbd000,16384,PROT\_READ)=0*  
*mprotect(0x563b6b8f6000,4096,PROT\_READ)=0*  
*mprotect(0x7f18b0e0d000,8192,PROT\_READ)=0*  
*prlimit64(0,RLIMIT\_STACK,NULL,{rlim\_cur=81921024,rlim\_max=RLIM64\_INFINITY})=0*  
*munmap(0x7f18b0dd0000,19163)=0*  
***readlink("/proc/self/exe","/home/lizka/\320\233\320\220\320\221\320\253\_\320\236\320\241/\320\233\320\260\320\261"...,1023)=42***  
***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:Process1425attached,child\_tidptr=0x7f18b0bbba10)=1425***  
*[pid1425]set\_robust\_list(0x7f18b0bbba20,24*  
*[pid1424]clone(child\_stack=NULL,flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD*  
*[pid1425]<...set\_robust\_listresumed>)=0*  
*[pid1425]dup2(3,0)=0*  
*[pid1425]dup2(6,1)=1*  
*[pid1425]close(4)=0*  
*[pid1425]close(5)=0*  
*[pid1425]close(7)=0*  
*[pid1425]close(8)=0*  
*[pid1425]execve("./child1",["./child1"],0x7ffd91bd4648/\*26vars/strace:Process1426attached*  
*[pid1424]<...cloneresumed>,child\_tidptr=0x7f18b0bbba10)=1426*  
*[pid1426]set\_robust\_list(0x7f18b0bbba20,24*  
*[pid1424]close(3*  
*[pid1426]<...set\_robust\_listresumed>)=0*  
*[pid1424]<...closeresumed>)=0*  
*[pid1424]close(5*  
*[pid1426]dup2(5,0*  
*[pid1424]<...closeresumed>)=0*  
*[pid1426]<...dup2resumed>)=0*  
*[pid1424]close(8*  
*[pid1425]<...execveresumed>)=0*  
*[pid1424]<...closeresumed>)=0*  
*[pid1426]dup2(8,1*  
*[pid1424]write(1,"Enteryourstringor(Enter/CT"...,51Enteryourstringor(Enter/CTRL+D)forstop:*  
*[pid1425]brk(NULL*  
*[pid1424]<...writeresumed>)=51*  
*[pid1426]<...dup2resumed>)=1*  
*[pid1424]read(0,*  
*[pid1425]<...brkresumed>)=0x5619e92c1000*  
*[pid1426]close(3)=0*  
*[pid1425]mmap(NULL,8192,PROT\_READ|PROT\_WRITE,MAP\_PRIVATE|MAP\_ANONYMOUS,-1,0*  
*[pid1426]close(4*  
*[pid1425]<...mmapresumed>)=0x7f1f9312c000*  
*[pid1426]<...closeresumed>)=0*  
*[pid1426]close(6*  
*[pid1425]access("/etc/ld.so.preload",R\_OK*  
*[pid1426]<...closeresumed>)=0*  
*[pid1425]<...accessresumed>)=-1ENOENT(Nosuchfileordirectory)*  
*[pid1426]close(7)=0*  
*[pid1425]openat(AT\_FDCWD,"/etc/ld.so.cache",O\_RDONLY|O\_CLOEXEC*  
*[pid1426]execve("./child2",["./child2"],0x7ffd91bd4648/26vars/*  
*[pid1425]<...openatresumed>)=4*  
*[pid1425]fstat(4,{st\_mode=S\_IFREG|0644,st\_size=19163,...})=0*  
*[pid1425]mmap(NULL,19163,PROT\_READ,MAP\_PRIVATE,4,0*  
*[pid1426]<...execveresumed>)=0*  
*[pid1425]<...mmapresumed>)=0x7f1f93127000*  
*[pid1426]brk(NULL)=0x55b9b3e14000*  
*[pid1425]close(4*  
*[pid1426]mmap(NULL,8192,PROT\_READ|PROT\_WRITE,MAP\_PRIVATE|MAP\_ANONYMOUS,-1,0)=0x7f452f70c000*  
*[pid1426]access("/etc/ld.so.preload",R\_OK*  
*[pid1425]<...closeresumed>)=0*  
*[pid1426]<...accessresumed>)=-1ENOENT(Nosuchfileordirectory)*  
*[pid1425]openat(AT\_FDCWD,"/lib/x86\_64-linux-gnu/libc.so.6",O\_RDONLY|O\_CLOEXEC*  
*[pid1426]openat(AT\_FDCWD,"/etc/ld.so.cache",O\_RDONLY|O\_CLOEXEC)=3*  
*[pid1426]fstat(3,*  
*[pid1425]<...openatresumed>)=4*  
*[pid1426]<...fstatresumed>{st\_mode=S\_IFREG|0644,st\_size=19163,...})=0*  
*[pid1425]read(4,*  
*[pid1426]mmap(NULL,19163,PROT\_READ,MAP\_PRIVATE,3,0*  
*[pid1425]<...readresumed>"\177ELF\2\1\1\3\0\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*  
*[pid1426]<...mmapresumed>)=0x7f452f707000*  
*[pid1425]pread64(4,*  
*[pid1426]close(3*  
*[pid1425]<...pread64resumed>"\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*  
*[pid1426]<...closeresumed>)=0*  
*[pid1425]fstat(4,*  
*[pid1426]openat(AT\_FDCWD,"/lib/x86\_64-linux-gnu/libc.so.6",O\_RDONLY|O\_CLOEXEC*  
*[pid1425]<...fstatresumed>{st\_mode=S\_IFREG|0755,st\_size=2125328,...})=0*  
*[pid1426]<...openatresumed>)=3*  
*[pid1425]pread64(4,*  
*[pid1426]read(3,*  
*[pid1425]<...pread64resumed>"\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*  
*[pid1426]<...readresumed>"\177ELF\2\1\1\3\0\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*  
*[pid1425]mmap(NULL,2170256,PROT\_READ,MAP\_PRIVATE|MAP\_DENYWRITE,4,0*  
*[pid1426]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*  
*[pid1425]<...mmapresumed>)=0x7f1f92f15000*  
*[pid1426]fstat(3,*  
*[pid1425]mmap(0x7f1f92f3d000,1605632,PROT\_READ|PROT\_EXEC,MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE,4,0x28000*  
*[pid1426]<...fstatresumed>{st\_mode=S\_IFREG|0755,st\_size=2125328,...})=0*  
*[pid1426]pread64(3,*  
*[pid1425]<...mmapresumed>)=0x7f1f92f3d000*  
*[pid1426]<...pread64resumed>"\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*  
*[pid1425]mmap(0x7f1f930c5000,323584,PROT\_READ,MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE,4,0x1b0000*  
*[pid1426]mmap(NULL,2170256,PROT\_READ,MAP\_PRIVATE|MAP\_DENYWRITE,3,0*  
*[pid1425]<...mmapresumed>)=0x7f1f930c5000*  
*[pid1426]<...mmapresumed>)=0x7f452f4f5000*  
*[pid1426]mmap(0x7f452f51d000,1605632,PROT\_READ|PROT\_EXEC,MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE,3,0x28000*  
*[pid1425]mmap(0x7f1f93114000,24576,PROT\_READ|PROT\_WRITE,MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE,4,0x1fe000*  
*[pid1426]<...mmapresumed>)=0x7f452f51d000*  
*[pid1425]<...mmapresumed>)=0x7f1f93114000*  
*[pid1426]mmap(0x7f452f6a5000,323584,PROT\_READ,MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE,3,0x1b0000)=0x7f452f6a5000*  
*[pid1425]mmap(0x7f1f9311a000,52624,PROT\_READ|PROT\_WRITE,MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS,-1,0*  
*[pid1426]mmap(0x7f452f6f4000,24576,PROT\_READ|PROT\_WRITE,MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE,3,0x1fe000*  
*[pid1425]<...mmapresumed>)=0x7f1f9311a000*  
*[pid1426]<...mmapresumed>)=0x7f452f6f4000*  
*[pid1426]mmap(0x7f452f6fa000,52624,PROT\_READ|PROT\_WRITE,MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS,-1,0*  
*[pid1425]close(4*  
*[pid1426]<...mmapresumed>)=0x7f452f6fa000*  
*[pid1425]<...closeresumed>)=0*  
*[pid1426]close(3*  
*[pid1425]mmap(NULL,12288,PROT\_READ|PROT\_WRITE,MAP\_PRIVATE|MAP\_ANONYMOUS,-1,0*  
*[pid1426]<...closeresumed>)=0*  
*[pid1425]<...mmapresumed>)=0x7f1f92f12000*  
*[pid1426]mmap(NULL,12288,PROT\_READ|PROT\_WRITE,MAP\_PRIVATE|MAP\_ANONYMOUS,-1,0)=0x7f452f4f2000*  
*[pid1425]arch\_prctl(ARCH\_SET\_FS,0x7f1f92f12740*  
*[pid1426]arch\_prctl(ARCH\_SET\_FS,0x7f452f4f2740*  
*[pid1425]<...arch\_prctlresumed>)=0*  
*[pid1426]<...arch\_prctlresumed>)=0*  
*[pid1425]set\_tid\_address(0x7f1f92f12a10*  
*[pid1426]set\_tid\_address(0x7f452f4f2a10*  
*[pid1425]<...set\_tid\_addressresumed>)=1425*  
*[pid1426]<...set\_tid\_addressresumed>)=1426*  
*[pid1425]set\_robust\_list(0x7f1f92f12a20,24*  
*[pid1426]set\_robust\_list(0x7f452f4f2a20,24*  
*[pid1425]<...set\_robust\_listresumed>)=0*  
*[pid1426]<...set\_robust\_listresumed>)=0*  
*[pid1425]rseq(0x7f1f92f13060,0x20,0,0x53053053*  
*[pid1426]rseq(0x7f452f4f3060,0x20,0,0x53053053*  
*[pid1425]<...rseqresumed>)=0*  
*[pid1426]<...rseqresumed>)=0*  
*[pid1426]mprotect(0x7f452f6f4000,16384,PROT\_READ)=0*  
*[pid1425]mprotect(0x7f1f93114000,16384,PROT\_READ*  
*[pid1426]mprotect(0x55b98f436000,4096,PROT\_READ*  
*[pid1425]<...mprotectresumed>)=0*  
*[pid1426]<...mprotectresumed>)=0*  
*[pid1425]mprotect(0x5619c3df3000,4096,PROT\_READ*  
*[pid1426]mprotect(0x7f452f744000,8192,PROT\_READ*  
*[pid1425]<...mprotectresumed>)=0*  
*[pid1426]<...mprotectresumed>)=0*  
*[pid1425]mprotect(0x7f1f93164000,8192,PROT\_READ*  
*[pid1426]prlimit64(0,RLIMIT\_STACK,NULL,*  
*[pid1425]<...mprotectresumed>)=0*  
*[pid1426]<...prlimit64resumed>{rlim\_cur=81921024,rlim\_max=RLIM64\_INFINITY})=0*  
*[pid1426]munmap(0x7f452f707000,19163*  
*[pid1425]prlimit64(0,RLIMIT\_STACK,NULL,*  
*[pid1426]<...munmapresumed>)=0*  
*[pid1425]<...prlimit64resumed>{rlim\_cur=81921024,rlim\_max=RLIM64\_INFINITY})=0*  
*[pid1426]read(0,*  
*[pid1425]munmap(0x7f1f93127000,19163)=0*  
*[pid1425]read(0,*  
*[pid1424]<...readresumed>"\n",1024)=1*  
*[pid1424]close(4)=0*  
*[pid1425]<...readresumed>"",1024)=0*  
*[pid1424]close(7)=0*  
*[pid1425]exit\_group(0*  
*[pid1424]close(6*  
*[pid1425]<...exit\_groupresumed>)=?[pid1424]<...closeresumed>)=0*  
*[pid1424]wait4(-1,*  
*[pid1426]<...readresumed>"",1024)=0*  
*[pid1425]+++exitedwith0+++*  
*[pid1424]<...wait4resumed>NULL,0,NULL)=1425*  
*[pid1426]exit\_group(0*  
*[pid1424]---****SIGCHLD{si\_signo=SIGCHLD,si\_code=CLD\_EXITED,si\_pid=1425,si\_uid=1000,si\_status=0,si\_utime=0,si\_stime=2/0.02s/}---***  
*[pid1426]<...exit\_groupresumed>)=?*  
*[pid1424]wait4(-1,*  
*[pid1426]+++exitedwith0+++*  
*...wait4resumed>NULL,0,NULL)=1426---****SIGCHLD{si\_signo=SIGCHLD,si\_code=CLD\_EXITED,si\_pid=1426,si\_uid=1000,si\_status=0,si\_utime=0,si\_stime=2/0.02s/\*/}---***  
*exit\_group(0)=?+++exitedwith0+++*

# Вывод

Во время выполнения лабораторной работы я разработала программу, которая использует несколько процессов для обработки строк, вводимых пользователем.

Основная сложность возникла из-за не закрытых каналов (ріpes), что приводило к зависанию процессов: дочерние процессы не завершались, поскольку продолжали ждать ввода. Я исправила это, убедившись, что все ненужные дескрипторы закрыты после их использования. В будущем хотелось бы уделить больше времени отладке и тестированию процессов, чтобы избежать подобных проблем. В целом, работа была полезной и помогла мне лучше понять