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-214Б-23

Студент: Миронов Д.А. Преподаватель: Бахарев В.Д.

Оценка:

Дата: 28.10.24

Москва, 2024

# Постановка задачи

**Вариант 1.**

**Пользователь вводит строки произвольной длины, которые принимаются родительским процессом. Родительский процесс отправляет их первому дочернему процессу, если длина строки меньше 10, или второму – если больше. Дочерние процессы принимают строки и записывают их в собственные выходные файлы, удаляя из строк все гласные.**

# Общий метод и алгоритм решения

Использованные системные вызовы:

* int channel1[2]; pipe(channel1); int channel1[2]; pipe(channel1); - создает два канала связи.
* const pid\_t child = fork(); – создает дочерний процесс.
* pid\_t pid = getpid(); – получает id текущего процесса.
* dup2(channel1[STDIN\_FILENO], STDIN\_FILENO); и dup2(channel2[STDIN\_FILENO], STDIN\_FILENO); – перенаправляет стандартный ввод на дескрипторы канала связи.
* int32\_t status = execv(path, args); – заменяет код новым программным кодом, указанным в path.
* wait(&child\_status); – родительский процесс ждет завершения дочернего процесса. Решение:

1. Обрабатываю путь переданный через аргументы командной строки.
2. Открываю два канала связи.
3. Создаю дочерний процесс. Связываю его ввод с первым каналом связи.
4. Находясь в родительском процессе, создаю еще один дочерний процесс. Связываю его ввод со вторым каналом связи.
5. Находясь в родительском процессе, считываю строки, вводимые пользователем, и в зависимости от длины отправляю их в первый или второй каналы связи.
6. В дочерних процессах получаю строку, переданную от родительского процесса и удаляю из неё гласные.
7. Записываю полученную строку в файл.

# Код программы

**Server.c**

#include <stdint.h>

#include <unistd.h>

#include <sys/wait.h>

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

**static** **char** CLIENT1\_PROGRAM\_NAME[] = "client1";

**static** **char** CLIENT2\_PROGRAM\_NAME[] = "client2";

**int** main(**int** argc, **char** \*\*argv) {

**if** (argc == 2) {

**char** msg[1024];

uint32\_t len = snprintf(msg, **sizeof**(msg) - 1, "usage: %s filename\n", argv[0]);

write(STDERR\_FILENO, msg, len);

exit(EXIT\_SUCCESS);

}

**char** progpath[1024];

{

// NOTE: Read full program path, including its name

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);

}

// NOTE: Trim the path to first slash from the end

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

--len;

progpath[len] = '\0';

}

**char** buf[4096];

ssize\_t bytes;

**int** channel1[2];

**if** (pipe(channel1) == -1) {

**const** **char** msg[] = "error: failed to create pipe1\n";

write(STDERR\_FILENO, msg, **sizeof**(msg));

exit(EXIT\_FAILURE);

}

**int** channel2[2];

**if** (pipe(channel2) == -1) {

**const** **char** msg[] = "error: failed to create pipe2\n";

write(STDERR\_FILENO, msg, **sizeof**(msg));

exit(EXIT\_FAILURE);

}

**const** pid\_t child = fork();

**switch** (child) {

**case** -1: {

**const** **char** msg[] = "error: failed to spawn new process\n";

write(STDERR\_FILENO, msg, **sizeof**(msg));

exit(EXIT\_FAILURE);

} **break**;

**case** 0: { // NOTE: We're a child, child doesn't know its pid after fork

pid\_t pid = getpid(); // NOTE: Get child PID

dup2(channel1[STDIN\_FILENO], STDIN\_FILENO);

//dup2(STDOUT\_FILENO, channel1[0]);

{

**char** msg[64];

**const** int32\_t length = snprintf(msg, **sizeof**(msg),

"%d: I'm a child\n", pid);

write(STDOUT\_FILENO, msg, length);

}

{

**char** path[1024];

snprintf(path, **sizeof**(path) - 1, "%s/%s", progpath, CLIENT1\_PROGRAM\_NAME);

**char** \***const** args[] = {CLIENT1\_PROGRAM\_NAME, argv[1], NULL};

int32\_t status = execv(path, 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**;

**default**: { // NOTE: We're a parent, parent knows PID of child after fork

**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: { // NOTE: We're a child, child doesn't know its pid after fork

pid\_t pid2 = getpid(); // NOTE: Get child PID

dup2(channel2[STDIN\_FILENO], STDIN\_FILENO);

{

**char** msg[64];

**const** int32\_t length = snprintf(msg, **sizeof**(msg),

"%d: I'm a child\n", pid2);

write(STDOUT\_FILENO, msg, length);

}

{

**char** path[1024];

snprintf(path, **sizeof**(path) - 1, "%s/%s", progpath, CLIENT2\_PROGRAM\_NAME);

**char** \***const** args[] = {CLIENT2\_PROGRAM\_NAME, argv[2], NULL};

int32\_t status = execv(path, 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**;

**default**: { // NOTE: We're a parent, parent knows PID of child after fork

//close(channel1[1]);

pid\_t pid = getpid(); // NOTE: Get parent PID

{

**char** msg[64];

**const** int32\_t length = snprintf(msg, **sizeof**(msg),

"%d: I'm a parent, my child has PID %d\n", pid, child);

write(STDOUT\_FILENO, msg, length);

}

**while** ((bytes = read(STDIN\_FILENO, buf, **sizeof**(buf) - 1)) > 0) {

//printf("Bytes: %lld; Str: <%s>", bytes, buf);

**if** (bytes == 1) {

exit(EXIT\_SUCCESS);

}

**if** (bytes > 10) {

write(channel2[STDOUT\_FILENO], buf, bytes);

} **else** {

write(channel1[STDOUT\_FILENO], buf, bytes);

}

}

// NOTE: `wait` blocks the parent until child exits

**int** child\_status;

wait(&child\_status);

**if** (child\_status != EXIT\_SUCCESS) {

**const** **char** msg[] = "error: child exited with error\n";

write(STDERR\_FILENO, msg, **sizeof**(msg));

exit(child\_status);

}

} **break**;

}

} **break**;

}

}

**Client1.c**

#include <stdint.h>

#include <stdbool.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include <string.h>

**int** main(**int** argc, **char** \*\*argv) {

**char** buf[4096];

**char** data[4096];

**const** **char** \*vowels = "aeiouAEIOU";

ssize\_t bytes;

int32\_t file = open(argv[1], O\_WRONLY | O\_CREAT | O\_APPEND, 0600);

**if** (file == -1) {

**const** **char** msg[] = "error: failed to open requested file\n";

write(STDERR\_FILENO, msg, **sizeof**(msg));

exit(EXIT\_FAILURE);

}

pid\_t pid = getpid();

**while** ((bytes = read(STDIN\_FILENO, data, **sizeof**(data) - 1)) > 0) {

data[bytes] = '\0';

**int** i = 0;

**int** ind = 0;

**while** (data[i] != '\0') {

**if** (!strchr(vowels, data[i])) {

buf[ind] = data[i];

ind++;

}

i++;

}

buf[ind - 1] = '\n';

int32\_t len = ind;

int32\_t written = write(file, buf, len);

**if** (written != len) {

**const** **char** msg[] = "error: failed to write to file\n";

write(STDERR\_FILENO, msg, **sizeof**(msg));

exit(EXIT\_FAILURE);

}

}

close(file);

}

**Client2.c**

#include <stdint.h>

#include <stdbool.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include <string.h>

**int** main(**int** argc, **char** \*\*argv) {

**char** buf[4096];

**char** data[4096];

**const** **char** \*vowels = "aeiouAEIOU";

ssize\_t bytes;

int32\_t file = open(argv[1], O\_WRONLY | O\_CREAT | O\_APPEND, 0600);

**if** (file == -1) {

**const** **char** msg[] = "error: failed to open requested file\n";

write(STDERR\_FILENO, msg, **sizeof**(msg));

exit(EXIT\_FAILURE);

}

pid\_t pid = getpid();

**while** ((bytes = read(STDIN\_FILENO, data, **sizeof**(data) - 1)) > 0) {

data[bytes] = '\0';

**int** i = 0;

**int** ind = 0;

**while** (data[i] != '\0') {

**if** (!strchr(vowels, data[i])) {

buf[ind] = data[i];

ind++;

}

i++;

}

buf[ind - 1] = '\n';

int32\_t len = ind;

int32\_t written = write(file, buf, len);

**if** (written != len) {

**const** **char** msg[] = "error: failed to write to file\n";

write(STDERR\_FILENO, msg, **sizeof**(msg));

exit(EXIT\_FAILURE);

}

}

close(file);

}

**Протокол работы программы**

$ ./servok File1.txt File2.txt

941: I'm a child

940: I'm a parent, my child has PID 941

942: I'm a child

text

some more text

string

really big string

$ cat File1.txt

txt

strng

$ cat File2.txt

sm mr txt

rlly bg strng

Strace:

$ strace -f ./servok

execve("./servok", ["./servok"], 0x7ffdb13d2c18 /\* 26 vars \*/) = 0

brk(NULL) = 0x5607260b2000

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fa4dcb6c000

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=19711, ...}) = 0

mmap(NULL, 19711, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fa4dcb67000

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) = 0x7fa4dc955000

mmap(0x7fa4dc97d000, 1605632, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7fa4dc97d000

mmap(0x7fa4dcb05000, 323584, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1b0000) = 0x7fa4dcb05000

mmap(0x7fa4dcb54000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1fe000) = 0x7fa4dcb54000

mmap(0x7fa4dcb5a000, 52624, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fa4dcb5a000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fa4dc952000

arch\_prctl(ARCH\_SET\_FS, 0x7fa4dc952740) = 0

set\_tid\_address(0x7fa4dc952a10) = 949

set\_robust\_list(0x7fa4dc952a20, 24) = 0

rseq(0x7fa4dc953060, 0x20, 0, 0x53053053) = 0

mprotect(0x7fa4dcb54000, 16384, PROT\_READ) = 0

mprotect(0x56072484b000, 4096, PROT\_READ) = 0

mprotect(0x7fa4dcba4000, 8192, PROT\_READ) = 0

prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

munmap(0x7fa4dcb67000, 19711) = 0

readlink("/proc/self/exe", "/mnt/c/Users/begemot/ClionProjec"..., 1023) = 54

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

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

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLDstrace: Process 950 attached

, child\_tidptr=0x7fa4dc952a10) = 950

[pid 950] set\_robust\_list(0x7fa4dc952a20, 24 <unfinished ...>

[pid 949] clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD <unfinished ...>

[pid 950] <... set\_robust\_list resumed>) = 0

strace: Process 951 attached

[pid 950] getpid( <unfinished ...>

[pid 949] <... clone resumed>, child\_tidptr=0x7fa4dc952a10) = 951

[pid 951] set\_robust\_list(0x7fa4dc952a20, 24 <unfinished ...>

[pid 949] getpid( <unfinished ...>

[pid 950] <... getpid resumed>) = 950

[pid 949] <... getpid resumed>) = 949

[pid 951] <... set\_robust\_list resumed>) = 0

[pid 949] write(1, "949: I'm a parent, my child has "..., 40 <unfinished ...>

949: I'm a parent, my child has PID 950

[pid 950] dup2(3, 0 <unfinished ...>

[pid 949] <... write resumed>) = 40

[pid 951] getpid( <unfinished ...>

[pid 949] read(0, <unfinished ...>

[pid 950] <... dup2 resumed>) = 0

[pid 951] <... getpid resumed>) = 951

[pid 950] write(1, "950: I'm a child\n", 17 <unfinished ...>

950: I'm a child

[pid 951] dup2(5, 0 <unfinished ...>

[pid 950] <... write resumed>) = 17

[pid 951] <... dup2 resumed>) = 0

[pid 950] execve("/mnt/c/Users/begemot/ClionProjects/OS-labs/Lab1/client1", ["client1"], 0x7ffeded7eef8 /\* 26 vars \*/ <unfinished ...>

[pid 951] write(1, "951: I'm a child\n", 17951: I'm a child

) = 17

[pid 951] execve("/mnt/c/Users/begemot/ClionProjects/OS-labs/Lab1/client2", ["client2", "SHELL=/bin/bash"], 0x7ffeded7eef8 /\* 26 vars \*/ <unfinished ...>

[pid 950] <... execve resumed>) = 0

[pid 950] brk(NULL) = 0x5654aea7c000

[pid 951] <... execve resumed>) = 0

[pid 951] brk(NULL <unfinished ...>

[pid 950] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

[pid 951] <... brk resumed>) = 0x55e35d723000

[pid 950] <... mmap resumed>) = 0x7fef3b0c3000

[pid 951] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

[pid 950] access("/etc/ld.so.preload", R\_OK <unfinished ...>

[pid 951] <... mmap resumed>) = 0x7f862899a000

[pid 950] <... access resumed>) = -1 ENOENT (No such file or directory)

[pid 951] access("/etc/ld.so.preload", R\_OK <unfinished ...>

[pid 950] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC <unfinished ...>

[pid 951] <... access resumed>) = -1 ENOENT (No such file or directory)

[pid 950] <... openat resumed>) = 7

[pid 951] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC <unfinished ...>

[pid 950] fstat(7, <unfinished ...>

[pid 951] <... openat resumed>) = 7

[pid 950] <... fstat resumed>{st\_mode=S\_IFREG|0644, st\_size=19711, ...}) = 0

[pid 951] fstat(7, <unfinished ...>

[pid 950] mmap(NULL, 19711, PROT\_READ, MAP\_PRIVATE, 7, 0 <unfinished ...>

[pid 951] <... fstat resumed>{st\_mode=S\_IFREG|0644, st\_size=19711, ...}) = 0

[pid 950] <... mmap resumed>) = 0x7fef3b0be000

[pid 951] mmap(NULL, 19711, PROT\_READ, MAP\_PRIVATE, 7, 0 <unfinished ...>

[pid 950] close(7 <unfinished ...>

[pid 951] <... mmap resumed>) = 0x7f8628995000

[pid 950] <... close resumed>) = 0

[pid 951] close(7 <unfinished ...>

[pid 950] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC <unfinished ...>

[pid 951] <... close resumed>) = 0

[pid 950] <... openat resumed>) = 7

[pid 951] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC <unfinished ...>

[pid 950] read(7, <unfinished ...>

[pid 951] <... openat resumed>) = 7

[pid 950] <... 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\0"..., 832) = 832

[pid 951] read(7, <unfinished ...>

[pid 950] pread64(7, <unfinished ...>

[pid 951] <... 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\0"..., 832) = 832

[pid 950] <... 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 951] pread64(7, <unfinished ...>

[pid 950] fstat(7, <unfinished ...>

[pid 951] <... 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 950] <... fstat resumed>{st\_mode=S\_IFREG|0755, st\_size=2125328, ...}) = 0

[pid 951] fstat(7, <unfinished ...>

[pid 950] pread64(7, <unfinished ...>

[pid 951] <... fstat resumed>{st\_mode=S\_IFREG|0755, st\_size=2125328, ...}) = 0

[pid 950] <... 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 951] pread64(7, <unfinished ...>

[pid 950] mmap(NULL, 2170256, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 7, 0 <unfinished ...>

[pid 951] <... 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 950] <... mmap resumed>) = 0x7fef3aeac000

[pid 951] mmap(NULL, 2170256, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 7, 0 <unfinished ...>

[pid 950] mmap(0x7fef3aed4000, 1605632, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 7, 0x28000 <unfinished ...>

[pid 951] <... mmap resumed>) = 0x7f8628783000

[pid 950] <... mmap resumed>) = 0x7fef3aed4000

[pid 951] mmap(0x7f86287ab000, 1605632, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 7, 0x28000 <unfinished ...>

[pid 950] mmap(0x7fef3b05c000, 323584, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 7, 0x1b0000 <unfinished ...>

[pid 951] <... mmap resumed>) = 0x7f86287ab000

[pid 950] <... mmap resumed>) = 0x7fef3b05c000

[pid 951] mmap(0x7f8628933000, 323584, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 7, 0x1b0000 <unfinished ...>

[pid 950] mmap(0x7fef3b0ab000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 7, 0x1fe000 <unfinished ...>

[pid 951] <... mmap resumed>) = 0x7f8628933000

[pid 950] <... mmap resumed>) = 0x7fef3b0ab000

[pid 951] mmap(0x7f8628982000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 7, 0x1fe000 <unfinished ...>

[pid 950] mmap(0x7fef3b0b1000, 52624, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

[pid 951] <... mmap resumed>) = 0x7f8628982000

[pid 950] <... mmap resumed>) = 0x7fef3b0b1000

[pid 951] mmap(0x7f8628988000, 52624, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

[pid 950] close(7 <unfinished ...>

[pid 951] <... mmap resumed>) = 0x7f8628988000

[pid 950] <... close resumed>) = 0

[pid 951] close(7 <unfinished ...>

[pid 950] mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

[pid 951] <... close resumed>) = 0

[pid 950] <... mmap resumed>) = 0x7fef3aea9000

[pid 951] mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

[pid 950] arch\_prctl(ARCH\_SET\_FS, 0x7fef3aea9740 <unfinished ...>

[pid 951] <... mmap resumed>) = 0x7f8628780000

[pid 950] <... arch\_prctl resumed>) = 0

[pid 951] arch\_prctl(ARCH\_SET\_FS, 0x7f8628780740 <unfinished ...>

[pid 950] set\_tid\_address(0x7fef3aea9a10 <unfinished ...>

[pid 951] <... arch\_prctl resumed>) = 0

[pid 950] <... set\_tid\_address resumed>) = 950

[pid 951] set\_tid\_address(0x7f8628780a10 <unfinished ...>

[pid 950] set\_robust\_list(0x7fef3aea9a20, 24 <unfinished ...>

[pid 951] <... set\_tid\_address resumed>) = 951

[pid 950] <... set\_robust\_list resumed>) = 0

[pid 951] set\_robust\_list(0x7f8628780a20, 24 <unfinished ...>

[pid 950] rseq(0x7fef3aeaa060, 0x20, 0, 0x53053053 <unfinished ...>

[pid 951] <... set\_robust\_list resumed>) = 0

[pid 950] <... rseq resumed>) = 0

[pid 951] rseq(0x7f8628781060, 0x20, 0, 0x53053053) = 0

[pid 950] mprotect(0x7fef3b0ab000, 16384, PROT\_READ <unfinished ...>

[pid 951] mprotect(0x7f8628982000, 16384, PROT\_READ <unfinished ...>

[pid 950] <... mprotect resumed>) = 0

[pid 951] <... mprotect resumed>) = 0

[pid 950] mprotect(0x5654ae275000, 4096, PROT\_READ <unfinished ...>

[pid 951] mprotect(0x55e35c45b000, 4096, PROT\_READ <unfinished ...>

[pid 950] <... mprotect resumed>) = 0

[pid 951] <... mprotect resumed>) = 0

[pid 950] mprotect(0x7fef3b0fb000, 8192, PROT\_READ <unfinished ...>

[pid 951] mprotect(0x7f86289d2000, 8192, PROT\_READ <unfinished ...>

[pid 950] <... mprotect resumed>) = 0

[pid 951] <... mprotect resumed>) = 0

[pid 950] prlimit64(0, RLIMIT\_STACK, NULL, <unfinished ...>

[pid 951] prlimit64(0, RLIMIT\_STACK, NULL, <unfinished ...>

[pid 950] <... prlimit64 resumed>{rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

[pid 951] <... prlimit64 resumed>{rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

[pid 950] munmap(0x7fef3b0be000, 19711 <unfinished ...>

[pid 951] munmap(0x7f8628995000, 19711 <unfinished ...>

[pid 950] <... munmap resumed>) = 0

[pid 951] <... munmap resumed>) = 0

[pid 950] openat(AT\_FDCWD, NULL, O\_WRONLY|O\_CREAT|O\_APPEND, 0600 <unfinished ...>

[pid 951] openat(AT\_FDCWD, "SHELL=/bin/bash", O\_WRONLY|O\_CREAT|O\_APPEND, 0600 <unfinished ...>

[pid 950] <... openat resumed>) = -1 EFAULT (Bad address)

[pid 950] write(2, "error: failed to open requested "..., 38error: failed to open requested file

) = 38

[pid 950] exit\_group(1 <unfinished ...>

[pid 951] <... openat resumed>) = -1 ENOENT (No such file or directory)

[pid 950] <... exit\_group resumed>) = ?

[pid 951] write(2, "error: failed to open requested "..., 38error: failed to open requested file

) = 38

[pid 951] exit\_group(1 <unfinished ...>

[pid 950] +++ exited with 1 +++

[pid 951] <... exit\_group resumed>) = ?

[pid 949] <... read resumed>0x7ffeded7ddb0, 4095) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

[pid 949] --- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=950, si\_uid=1000, si\_status=1, si\_utime=0, si\_stime=0} ---

[pid 949] read(0, <unfinished ...>

[pid 951] +++ exited with 1 +++

<... read resumed>0x7ffeded7ddb0, 4095) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=951, si\_uid=1000, si\_status=1, si\_utime=0, si\_stime=0} ---

read(0,

"\n", 4095) = 1

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**

В результате выполнения лабораторной работы удалось познакомиться с системными вызовами (такими как pipe(), fork(), dup2(), execv(), wait()) и реализовать программу записи строк в разные файлы. Проблем при выполнении работы не возникло.