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: Попов А.В.

Преподаватель: Бахарев В.Д.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 28.10.24

Москва, 2024

**Постановка задачи**

**Вариант 21.**

Родительский процесс создает два дочерних процесса. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись для child1. Аналогично для второй строки и процесса child2. Родительский и дочерний процесс должны быть представлены разными программами.

Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1 или в pipe2 в зависимости от правила фильтрации. Процесс child1 и child2 производят работу над строками. Процессы пишут результаты своей работы в стандартный вывод.

Правило фильтрации: нечетные строки отправляются в pipe1, четные в pipe2. Дочерние процессы инвертируют строки.

**Общий метод и алгоритм решения**

Использованные системные вызовы:

* pid\_t fork(void); – создает дочерний процесс.
* int pipe(int fd[2]); – создает pipe, однонаправленный канал связи между процессами
* ssize\_t write(int fd, const void buf[.count], size\_t count); - пишет count байтов из буфера в файл, на который ссылается файловый дескриптор fd
* ssize\_t read(int fd, void buf[.count], size\_t count) – пытается прочитать count байтов из файлового дескриптора fd в буфер buff
* pid\_t getpid(void); - получить pid текущего или родительского процесса
* int open(const char \*pathname, int flags, mode\_t mode ); - открыть файл с указанными флагами или создать, если указаны специальные флаги, возвращает файловый дескриптор
* int close(int fd); - закрывает файловый дескриптор
* int execv(const char \*pathname, char \*const argv[]); - заменяет образ текущего процесса на новый образ, создается новый стек, кучу и сегменты данных
* pid\_t waidpid (pid\_t pid, int \*stat\_loc, int options); - ждет пока процесс с pid завершится и получается код выхода

Программа получает из стандартного ввода имена файлов, создает для каждого из детей pipe и создает дочерний процесс с помощью fork. Перед запуском кода дочернего процесса выход из pipe присоединяется к стандартному входу дочернего процесса через dup2. Далее в дочерний процесс через аргументы передается имя файла для записи и код дочернего запускается через execv. Данные операции повторяются еще раз для второго дочернего процесса. Родительский процесс получается строку из стандартного ввода и пересылает ее в один из pipe в зависимости от четности строки. Когда на вход поступает EOF или пустая строка, дочерние процессы завершаются, а родительский процесс ждет их заверения. Все процессы закрывают в конце закрывают открытые файлы и pipe

**Код программы**

**parent.c**

#include <stdlib.h>  
#include <unistd.h>  
  
  
#include <sys/wait.h>  
#include <libio/io.h>  
#include <parent/processes.h>  
  
#define MAX\_LINE\_LENGTH 1024  
  
  
  
  
int main(void) {  
 // Get child exec path from environment variable  
 const char \*env\_var\_name = "CHILD\_EXEC\_PATH";  
 char \*exec\_path = getenv(env\_var\_name);  
 if (exec\_path == NULL) {  
 print\_fd(STDERR\_FILENO, "%s: environment variable %s not set\n", exec\_path, env\_var\_name);  
 exit(EXIT\_FAILURE);  
 }  
  
 child\_t child[2] = {  
 create\_empty\_child("child1"),  
 create\_empty\_child("child2"),  
 };  
 const size\_t child\_len = sizeof(child) / sizeof(child[0]);  
  
 // Read file path from stdin  
 for (int i = 0; i < child\_len; i++) {  
 ssize\_t read\_bytes = reads\_fd(STDIN\_FILENO, child[i].file\_path, PATH\_MAX);  
 if (read\_bytes == -1) {  
 print\_fd(STDERR\_FILENO, "Error: Failure during reading file path");  
 exit(EXIT\_FAILURE);  
 }  
 child[i].file\_path[read\_bytes - 1] = '\0'; // Remove trailing newline  
 }  
  
 // Start child processes  
 for (int i = 0; i < child\_len; i++) {  
 const pid\_t status = start\_child\_process(exec\_path, &child[i]);  
 if (status == -1) {  
 exit(EXIT\_FAILURE);  
 }  
 if (status == 0) {  
 // Child process  
 exit(EXIT\_SUCCESS);  
 }  
 }  
  
 // Send to child processes  
 char line[MAX\_LINE\_LENGTH];  
 int line\_number = 1;  
 while (reads\_fd(STDIN\_FILENO, line, MAX\_LINE\_LENGTH) > 0) {  
 if (line[0] == '\n') {  
 // Close on empty line  
 for (int i = 0; i < child\_len; i++) {  
 const ssize\_t bytes = write\_str(child[i].channel[PIPE\_WRITE\_END], line);  
 if (bytes == -1) {  
 print\_fd(STDERR\_FILENO, "Error: Failure during writing to pipe for %s\n", child[i].name);  
 exit(EXIT\_FAILURE);  
 }  
 }  
 break;  
 }  
 child\_t current\_child = child[line\_number % child\_len];  
 const ssize\_t bytes = write\_str(current\_child.channel[PIPE\_WRITE\_END], line);  
 if (bytes == -1) {  
 print\_fd(STDERR\_FILENO, "Error: Failure during writing to pipe for %s\n", current\_child.name);  
 exit(EXIT\_FAILURE);  
 }  
 line\_number++;  
 }  
 // Close all pipes and wait for children  
 for (int i = 0; i < child\_len; i++) {  
 close\_child\_process(child[i]);  
 }  
 return 0;  
}

**processes.c**

*/\*\*  
 \* @file  
 \* @brief  
 \* @details  
 \* @author xsestech   
 \* @date 28.10.2024  
 \*/*#include <parent/processes.h>  
  
child\_t create\_empty\_child(const char\* name) {  
 child\_t child;  
 strncpy(child.name, name, sizeof(child.name));  
 child.pid = -1;  
 child.channel[0] = -1;  
 child.channel[1] = -1;  
 return child;  
}  
  
pid\_t start\_child\_process(char \*exec\_path, child\_t \*child) {  
 if (pipe(child->channel) == -1) {  
 print\_fd(STDERR\_FILENO, "Error during pipe creation for %s", child->name);  
 return -1;  
 }  
  
 pid\_t fork\_pid = fork();  
  
 if (fork\_pid == -1) {  
 print\_fd(STDERR\_FILENO, "Error during creating process for %s\n", child->name);  
 return -1;  
 }  
  
 if (fork\_pid == 0) {  
 // We are child  
 pid\_t child\_pid = getpid();  
  
 printf("%s: child pid %d\n", exec\_path, child\_pid);  
 // Redirect pipe output to child stdin, before execv  
 if (dup2(child->channel[PIPE\_READ\_END],STDIN\_FILENO) == -1) {  
 print\_fd(STDERR\_FILENO, "Error during dup2 for %s\n", child->name);  
 return -1;  
 }  
 close(child->channel[PIPE\_WRITE\_END]);  
 const char \*args[] = {  
 child->name,  
 child->file\_path,  
 NULL,  
 };  
 const int32\_t status = execv(exec\_path, args);  
 close(child->channel[PIPE\_READ\_END]);  
 if (status != 0) {  
 print\_fd(STDERR\_FILENO, "Error executing %s in %s", exec\_path, child->name);  
 return -1;  
 }  
 return 0;  
 }  
 // We are parent  
 close(child->channel[PIPE\_READ\_END]);  
 const pid\_t parent\_pid = getppid();  
 child->pid = fork\_pid;  
 print\_fd(STDOUT\_FILENO, "Parent %d: created child with pid %d\n", parent\_pid, fork\_pid);  
 return fork\_pid;  
}  
void close\_child\_process(const child\_t child) {  
 close(child.channel[PIPE\_WRITE\_END]);  
 int child\_status;  
 waitpid(child.pid, &child\_status, 0);  
 print\_fd(STDOUT\_FILENO, "Child %d: exit status %d\n", child.pid, child\_status);  
}

**io.c**

/\*\*  
 \* @file  
 \* @brief  
 \* @details  
 \* @author xsestech   
 \* @date 27.10.2024  
 \*/  
  
#include <libio/io.h>  
  
ssize\_t print\_fd(const int fd, char \*fmt, ...) {  
 va\_list args;  
 va\_start(args, fmt);  
 char buff[IO\_MAX\_STR\_LEN];  
 size\_t len = vsnprintf(buff, IO\_MAX\_STR\_LEN - 1, fmt, args);  
 const ssize\_t writen\_bytes = write(fd, buff, len);  
 va\_end(args);  
 return writen\_bytes;  
}  
ssize\_t write\_str(const int fd, const char \*buff) {  
 return write(fd, buff, strlen(buff));  
}  
  
ssize\_t reads\_fd(const int fd, char \*buff, const size\_t buff\_size) {  
 ssize\_t read\_bytes = 0;  
 return read(fd, buff, buff\_size);  
}

**child.c**

*/\*\*  
 \* @file  
 \* @brief  
 \* @details  
 \* @author xsestech   
 \* @date 26.10.2024  
 \*/*#include <stdio.h>  
#include <stdlib.h>  
#include <unistd.h>  
#include <fcntl.h>  
#include <libconfig/config.h>  
  
#include <libio/io.h>  
  
int main(const int argc, char \*argv[]) {  
 if (argc != 2) {  
 print\_fd(STDERR\_FILENO, "No file specified");  
 exit(EXIT\_FAILURE);  
 }  
 const pid\_t pid = getpid();  
 int file = open(argv[1], O\_WRONLY | O\_CREAT | O\_TRUNC | O\_APPEND, 0600);  
  
 if (file == -1) {  
 print\_fd(STDERR\_FILENO, "%d: Error opening file %s\n", pid, argv[1]);  
 exit(EXIT\_FAILURE);  
 }  
  
 print\_fd(STDOUT\_FILENO, "%d: opened file %s\n", getpid(), argv[1]);  
 char buffer[MAX\_LINE\_LENGTH];  
 ssize\_t bytes = 0;  
 while ((bytes = read(STDIN\_FILENO, buffer, MAX\_LINE\_LENGTH)) > 0) {  
 if (bytes == -1) {  
 print\_fd(STDERR\_FILENO, "%d: Error reading from pipe\n", pid);  
 exit(EXIT\_FAILURE);  
 }  
 if (buffer[0] == '\n') {  
 break;  
 }  
  
  
 buffer[bytes - 1] = '\0'; // remove newline  
 print\_fd(STDOUT\_FILENO, "%d: got: %s\n", pid, buffer);  
 if (write(file, buffer, bytes - 1) != bytes - 1) {  
 print\_fd(STDERR\_FILENO, "%d: Error writing to file\n", pid);  
 exit(EXIT\_FAILURE);  
 }  
 }  
 const char term = '\0';  
 write(file, &term, sizeof(term));  
 close(file);  
 return 0;  
}

**Протокол работы программы**

**Здесь нужно показать тесты программы (текст или скриншоты), а затем показать полный вывод утилиты strace (или какой-либо другой утилиты на Windows, если вы выполняете лабы на этой операционной системе).**

**В strace нужно обязательно выделить, где происходят системные вызовы, которые вы использовали в лабораторной работе (например, где в первой лабораторной работе был вызван fork и другие вызовы). Полный список вызовов, которые нужно будет выделить в выводе strace, будет указан при выдаче лабы в нашем канале.**

**Тестирование:**

builder@4c1c3dd98286:/app$ ./build/parent/parent

file1.out

file2.out

Parent 1: created child with pid 398

build/child/child: child pid 398

Parent 1: created child with pid 399

build/child/child: child pid 399

398: opened file file1.out

399: opened file file2.out

123

399: got: 123

234

398: got: 234

123

399: got: 123

234

398: got: 234

123

399: got: 123

234

398: got: 234

Child 398: exit status 0

Child 399: exit status 0

cat file1.out

234234234

$ cat < file2.out

123123123

**Strace:**

builder@4c1c3dd98286:/app$ strace -f ./build/parent/parent

execve("./build/parent/parent", ["./build/parent/parent"], 0xffffc9c0d538 /\* 9 vars \*/) = 0

brk(NULL) = 0xaaaae7b11000

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0xffffa3fb0000

faccessat(AT\_FDCWD, "/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=13739, ...}) = 0

mmap(NULL, 13739, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0xffffa3fac000

close(3) = 0

openat(AT\_FDCWD, "/lib/aarch64-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\267\0\1\0\0\0\360\206\2\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=1722920, ...}) = 0

mmap(NULL, 1892240, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_DENYWRITE, -1, 0) = 0xffffa3da9000

mmap(0xffffa3db0000, 1826704, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0) = 0xffffa3db0000

munmap(0xffffa3da9000, 28672) = 0

munmap(0xffffa3f6e000, 36752) = 0

mprotect(0xffffa3f4a000, 77824, PROT\_NONE) = 0

mmap(0xffffa3f5d000, 20480, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x19d000) = 0xffffa3f5d000

mmap(0xffffa3f62000, 49040, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0xffffa3f62000

close(3) = 0

set\_tid\_address(0xffffa3fb0fb0) = 393

set\_robust\_list(0xffffa3fb0fc0, 24) = 0

rseq(0xffffa3fb1600, 0x20, 0, 0xd428bc00) = 0

mprotect(0xffffa3f5d000, 12288, PROT\_READ) = 0

mprotect(0xaaaab884f000, 4096, PROT\_READ) = 0

mprotect(0xffffa3fb5000, 8192, PROT\_READ) = 0

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

munmap(0xffffa3fac000, 13739) = 0

**read**(0, file1.in

"file1.in\n", 4096) = 9

**read**(0, file2.in

"file2.in\n", 4096) = 9

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

**clone**(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLDstrace: Process 394 attached

, child\_tidptr=0xffffa3fb0fb0) = 394

[pid 394] set\_robust\_list(0xffffa3fb0fc0, 24 <unfinished ...>

[pid 393] **close**(3 <unfinished ...>

[pid 394] <... set\_robust\_list resumed>) = 0

[pid 393] <... close resumed>) = 0

[pid 394] **getpid**( <unfinished ...>

[pid 393] **getppid**( <unfinished ...>

[pid 394] <... getpid resumed>) = 394

[pid 393] <... getppid resumed>) = 390

[pid 394] fstat(1, <unfinished ...>

[pid 393] **write**(1, "Parent 390: created child with p"..., 39 <unfinished ...>

[pid 394] <... fstat resumed>{st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}) = 0

Parent 390: created child with pid 394

[pid 393] <... write resumed>) = 39

[pid 394] getrandom( <unfinished ...>

[pid 393] **pipe2**( <unfinished ...>

[pid 394] <... getrandom resumed>"\x83\x57\xae\xda\x68\x11\xd2\x5f", 8, GRND\_NONBLOCK) = 8

[pid 393] <... pipe2 resumed>[3, 5], 0) = 0

[pid 394] brk(NULL <unfinished ...>

[pid 393] **clone**(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD <unfinished ...>

[pid 394] <... brk resumed>) = 0xaaaae7b11000

[pid 394] brk(0xaaaae7b32000 <unfinished ...>

[pid 393] <... clone resumed>, child\_tidptr=0xffffa3fb0fb0) = 395

[pid 394] <... brk resumed>) = 0xaaaae7b32000

strace: Process 395 attached

[pid 393] **close**(3 <unfinished ...>

[pid 394] **write**(1, "build/child/child: child pid 394"..., 33 <unfinished ...>

[pid 393] <... close resumed>) = 0

build/child/child: child pid 394

[pid 393] **getppid**( <unfinished ...>

[pid 395] set\_robust\_list(0xffffa3fb0fc0, 24 <unfinished ...>

[pid 394] <... write resumed>) = 33

[pid 393] <... getppid resumed>) = 390

[pid 395] <... set\_robust\_list resumed>) = 0

[pid 394] **dup3**(3, 0, 0 <unfinished ...>

[pid 393] **write**(1, "Parent 390: created child with p"..., 39 <unfinished ...>

[pid 395] **getpid**( <unfinished ...>

[pid 394] <... dup3 resumed>) = 0

Parent 390: created child with pid 395

[pid 393] <... write resumed>) = 39

[pid 394] **close**(4 <unfinished ...>

[pid 393] **read**(0, <unfinished ...>

[pid 395] <... getpid resumed>) = 395

[pid 394] <... close resumed>) = 0

[pid 395] fstat(1, <unfinished ...>

[pid 394] execve("build/child/child", ["child1", "file1.in"], 0xfffff197aa28 /\* 9 vars \*/ <unfinished ...>

[pid 395] <... fstat resumed>{st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}) = 0

[pid 395] getrandom("\xd3\x0f\x43\x8a\xb8\x72\x59\x3d", 8, GRND\_NONBLOCK) = 8

[pid 395] brk(NULL) = 0xaaaae7b11000

[pid 395] brk(0xaaaae7b32000) = 0xaaaae7b32000

[pid 395] **write**(1, "build/child/child: child pid 395"..., 33build/child/child: child pid 395

) = 33

[pid 395] **dup3**(3, 0, 0) = 0

[pid 395] **close**(5) = 0

[pid 395] **execve**("build/child/child", ["child2", "file2.in"], 0xfffff197aa28 /\* 9 vars \*/ <unfinished ...>

[pid 394] <... execve resumed>) = 0

[pid 394] brk(NULL) = 0xaaab12ee1000

[pid 394] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

[pid 395] <... execve resumed>) = 0

[pid 394] <... mmap resumed>) = 0xffffb9203000

[pid 395] brk(NULL <unfinished ...>

[pid 394] faccessat(AT\_FDCWD, "/etc/ld.so.preload", R\_OK <unfinished ...>

[pid 395] <... brk resumed>) = 0xaaaaec976000

[pid 394] <... faccessat resumed>) = -1 ENOENT (No such file or directory)

[pid 395] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

[pid 394] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 4

[pid 394] fstat(4, <unfinished ...>

[pid 395] <... mmap resumed>) = 0xffffb8d35000

[pid 394] <... fstat resumed>{st\_mode=S\_IFREG|0644, st\_size=13739, ...}) = 0

[pid 395] faccessat(AT\_FDCWD, "/etc/ld.so.preload", R\_OK <unfinished ...>

[pid 394] mmap(NULL, 13739, PROT\_READ, MAP\_PRIVATE, 4, 0 <unfinished ...>

[pid 395] <... faccessat resumed>) = -1 ENOENT (No such file or directory)

[pid 394] <... mmap resumed>) = 0xffffb91ff000

[pid 394] close(4 <unfinished ...>

[pid 395] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 5

[pid 394] <... close resumed>) = 0

[pid 395] fstat(5, <unfinished ...>

[pid 394] openat(AT\_FDCWD, "/lib/aarch64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC <unfinished ...>

[pid 395] <... fstat resumed>{st\_mode=S\_IFREG|0644, st\_size=13739, ...}) = 0

[pid 394] <... openat resumed>) = 4

[pid 395] mmap(NULL, 13739, PROT\_READ, MAP\_PRIVATE, 5, 0 <unfinished ...>

[pid 394] read(4, <unfinished ...>

[pid 395] <... mmap resumed>) = 0xffffb8d31000

[pid 394] <... read resumed>"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0\267\0\1\0\0\0\360\206\2\0\0\0\0\0"..., 832) = 832

[pid 395] **close**(5 <unfinished ...>

[pid 394] fstat(4, {st\_mode=S\_IFREG|0755, st\_size=1722920, ...}) = 0

[pid 395] <... close resumed>) = 0

[pid 395] openat(AT\_FDCWD, "/lib/aarch64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC <unfinished ...>

[pid 394] mmap(NULL, 1892240, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_DENYWRITE, -1, 0 <unfinished ...>

[pid 395] <... openat resumed>) = 5

[pid 394] <... mmap resumed>) = 0xffffb8ffc000

[pid 395] **read**(5, <unfinished ...>

[pid 394] mmap(0xffffb9000000, 1826704, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0 <unfinished ...>

[pid 395] <... read resumed>"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0\267\0\1\0\0\0\360\206\2\0\0\0\0\0"..., 832) = 832

[pid 394] <... mmap resumed>) = 0xffffb9000000

[pid 395] fstat(5, <unfinished ...>

[pid 394] munmap(0xffffb8ffc000, 16384 <unfinished ...>

[pid 395] <... fstat resumed>{st\_mode=S\_IFREG|0755, st\_size=1722920, ...}) = 0

[pid 394] <... munmap resumed>) = 0

[pid 395] mmap(NULL, 1892240, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_DENYWRITE, -1, 0 <unfinished ...>

[pid 394] munmap(0xffffb91be000, 49040 <unfinished ...>

[pid 395] <... mmap resumed>) = 0xffffb8b2e000

[pid 394] <... munmap resumed>) = 0

[pid 395] mmap(0xffffb8b30000, 1826704, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 5, 0 <unfinished ...>

[pid 394] mprotect(0xffffb919a000, 77824, PROT\_NONE <unfinished ...>

[pid 395] <... mmap resumed>) = 0xffffb8b30000

[pid 394] <... mprotect resumed>) = 0

[pid 395] munmap(0xffffb8b2e000, 8192 <unfinished ...>

[pid 394] mmap(0xffffb91ad000, 20480, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x19d000 <unfinished ...>

[pid 395] <... munmap resumed>) = 0

[pid 394] <... mmap resumed>) = 0xffffb91ad000

[pid 395] munmap(0xffffb8cee000, 57232 <unfinished ...>

[pid 394] mmap(0xffffb91b2000, 49040, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

[pid 395] <... munmap resumed>) = 0

[pid 394] <... mmap resumed>) = 0xffffb91b2000

[pid 395] mprotect(0xffffb8cca000, 77824, PROT\_NONE <unfinished ...>

[pid 394] **close**(4 <unfinished ...>

[pid 395] <... mprotect resumed>) = 0

[pid 395] mmap(0xffffb8cdd000, 20480, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 5, 0x19d000 <unfinished ...>

[pid 394] <... close resumed>) = 0

[pid 395] <... mmap resumed>) = 0xffffb8cdd000

[pid 394] set\_tid\_address(0xffffb9203fb0 <unfinished ...>

[pid 395] mmap(0xffffb8ce2000, 49040, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0 <unfinished ...>

[pid 394] <... set\_tid\_address resumed>) = 394

[pid 395] <... mmap resumed>) = 0xffffb8ce2000

[pid 394] set\_robust\_list(0xffffb9203fc0, 24 <unfinished ...>

[pid 395] **close**(5 <unfinished ...>

[pid 394] <... set\_robust\_list resumed>) = 0

[pid 395] <... close resumed>) = 0

[pid 394] rseq(0xffffb9204600, 0x20, 0, 0xd428bc00 <unfinished ...>

[pid 395] set\_tid\_address(0xffffb8d35fb0 <unfinished ...>

[pid 394] <... rseq resumed>) = 0

[pid 395] <... set\_tid\_address resumed>) = 395

[pid 394] mprotect(0xffffb91ad000, 12288, PROT\_READ <unfinished ...>

[pid 395] set\_robust\_list(0xffffb8d35fc0, 24 <unfinished ...>

[pid 394] <... mprotect resumed>) = 0

[pid 395] <... set\_robust\_list resumed>) = 0

[pid 394] mprotect(0xaaaae4fff000, 4096, PROT\_READ <unfinished ...>

[pid 395] rseq(0xffffb8d36600, 0x20, 0, 0xd428bc00 <unfinished ...>

[pid 394] <... mprotect resumed>) = 0

[pid 395] <... rseq resumed>) = 0

[pid 394] mprotect(0xffffb9208000, 8192, PROT\_READ <unfinished ...>

[pid 395] mprotect(0xffffb8cdd000, 12288, PROT\_READ <unfinished ...>

[pid 394] <... mprotect resumed>) = 0

[pid 395] <... mprotect resumed>) = 0

[pid 394] prlimit64(0, RLIMIT\_STACK, NULL, <unfinished ...>

[pid 395] mprotect(0xaaaab828f000, 4096, PROT\_READ <unfinished ...>

[pid 394] <... prlimit64 resumed>{rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

[pid 394] munmap(0xffffb91ff000, 13739 <unfinished ...>

[pid 395] <... mprotect resumed>) = 0

[pid 395] mprotect(0xffffb8d3a000, 8192, PROT\_READ <unfinished ...>

[pid 394] <... munmap resumed>) = 0

[pid 395] <... mprotect resumed>) = 0

[pid 394] **getpid**( <unfinished ...>

[pid 395] prlimit64(0, RLIMIT\_STACK, NULL, <unfinished ...>

[pid 394] <... getpid resumed>) = 394

[pid 395] <... prlimit64 resumed>{rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

[pid 394] **openat**(AT\_FDCWD, "file1.in", O\_WRONLY|O\_CREAT|O\_TRUNC|O\_APPEND, 0600 <unfinished ...>

[pid 395] munmap(0xffffb8d31000, 13739) = 0

[pid 395] **getpid**() = 395

[pid 395] **openat**(AT\_FDCWD, "file2.in", O\_WRONLY|O\_CREAT|O\_TRUNC|O\_APPEND, 0600 <unfinished ...>

[pid 394] <... openat resumed>) = 4

[pid 395] <... openat resumed>) = 5

[pid 394] getpid( <unfinished ...>

[pid 395] getpid( <unfinished ...>

[pid 394] <... getpid resumed>) = 394

[pid 395] <... getpid resumed>) = 395

[pid 394] **write**(1, "394: opened file file1.in\n", 26394: opened file file1.in

<unfinished ...>

[pid 395] **write**(1, "395: opened file file2.in\n", 26 <unfinished ...>

[pid 394] <... write resumed>) = 26

395: opened file file2.in

[pid 395] <... write resumed>) = 26

[pid 394] **read**(0, <unfinished ...>

[pid 395] **read**(0, 123

<unfinished ...>

[pid 393] <... read resumed>"123\n", 1024) = 4

[pid 393] **write**(5, "123\n", 4) = 4

[pid 395] <... read resumed>"123\n", 1024) = 4

[pid 393] **read**(0, <unfinished ...>

[pid 395] **write**(1, "395: got: 123\n", 14395: got: 123

) = 14

[pid 395] **write**(5, "123", 3) = 3

[pid 395] **read**(0, 234

<unfinished ...>

[pid 393] <... read resumed>"234\n", 1024) = 4

[pid 393] **write**(4, "234\n", 4) = 4

[pid 394] <... read resumed>"234\n", 1024) = 4

[pid 393] **read**(0, <unfinished ...>

[pid 394] **write**(1, "394: got: 234\n", 14394: got: 234

) = 14

[pid 394] **write**(4, "234", 3) = 3

[pid 394] **read**(0, 123

<unfinished ...>

[pid 393] <... read resumed>"123\n", 1024) = 4

[pid 393] **write**(5, "123\n", 4) = 4

[pid 395] <... read resumed>"123\n", 1024) = 4

[pid 395] **write**(1, "395: got: 123\n", 14395: got: 123

<unfinished ...>

[pid 393] **read**(0, <unfinished ...>

[pid 395] <... write resumed>) = 14

[pid 395] **write**(5, "123", 3) = 3

[pid 395] **read**(0, 234

<unfinished ...>

[pid 393] <... read resumed>"234\n", 1024) = 4

[pid 393] **write**(4, "234\n", 4) = 4

[pid 394] <... read resumed>"234\n", 1024) = 4

[pid 393] **read**(0, <unfinished ...>

[pid 394] **write**(1, "394: got: 234\n", 14394: got: 234

) = 14

[pid 394] **write**(4, "234", 3) = 3

[pid 394] **read**(0, 123

<unfinished ...>

[pid 393] <... read resumed>"123\n", 1024) = 4

[pid 393] **write**(5, "123\n", 4) = 4

[pid 395] <... read resumed>"123\n", 1024) = 4

[pid 393] **read**(0, <unfinished ...>

[pid 395] **write**(1, "395: got: 123\n", 14395: got: 123

) = 14

[pid 395] **write**(5, "123", 3) = 3

[pid 395] **read**(0, 234

<unfinished ...>

[pid 393] <... read resumed>"234\n", 1024) = 4

[pid 393] **write**(4, "234\n", 4) = 4

[pid 394] <... read resumed>"234\n", 1024) = 4

[pid 393] **read**(0, <unfinished ...>

[pid 394] **write**(1, "394: got: 234\n", 14394: got: 234

) = 14

[pid 394] **write**(4, "234", 3) = 3

[pid 394] **read**(0,

<unfinished ...>

[pid 393] <... read resumed>"\n", 1024) = 1

[pid 393] **write**(4, "\n34\n", 4) = 4

[pid 394] <... read resumed>"\n34\n", 1024) = 4

[pid 393] **write**(5, "\n34\n", 4 <unfinished ...>

[pid 394] **write**(4, "\0", 1 <unfinished ...>

[pid 393] <... write resumed>) = 4

[pid 395] <... read resumed>"\n34\n", 1024) = 4

[pid 393] **close**(4 <unfinished ...>

[pid 395] **write**(5, "\0", 1 <unfinished ...>

[pid 393] <... close resumed>) = 0

[pid 393] **wait4**(394, <unfinished ...>

[pid 394] <... write resumed>) = 1

[pid 394] **close**(4 <unfinished ...>

[pid 395] <... write resumed>) = 1

[pid 395] **close**(5) = 0

[pid 394] <... close resumed>) = 0

[pid 395] **exit**\_**group**(0 <unfinished ...>

[pid 394] **exit\_group**(0 <unfinished ...>

[pid 395] <... exit\_group resumed>) = ?

[pid 394] <... exit\_group resumed>) = ?

[pid 395] +++ exited with 0 +++

[pid 394] +++ exited with 0 +++

<... wait4 resumed>[{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 394

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=395, si\_uid=501, si\_status=0, si\_utime=0, si\_stime=0} ---

**write**(1, "Child 394: exit status 0\n", 25Child 394: exit status 0

) = 25

close(5) = 0

**wait4**(395, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 395

**write**(1, "Child 395: exit status 0\n", 25Child 395: exit status 0

) = 25

**exit\_group**(0) = ?

+++ exited with 0 +++

**Вывод**

В ходе данной работы я научился создавать процессы, налаживать общение между ними. Я столкнулся с проблемами при пересылке из входа pipe в стандартный ввод дочернего процесса, т.к. очень легко перепутать индексы и порядок в dup2. В целом я даже рад данному обстоятельству, т.к. больше смог разобраться в теме.