**Московский авиационный институт  
(Национальный исследовательский университет)**

**Факультет информационных технологий и прикладной математики  
Кафедра вычислительной математики и программирования**

**Лабораторная работа № 2 по курсу «Операционные системы»**

Студент: Абросимов Алексей Дмитриевич

Группа: М8О-207Б-20

Преподаватель: Е. С. Миронов

Вариант: 21

Дата:

Оценка:

Москва, 2021

## **1. Описание**

Данная лабораторная работа будет выполняться в ОС Unix.

**Процесс** – абстракция, описывающая выполняющуюся программу.

**Задание:**

Имеется три процесса: родительский и два дочерних. Взаимодействие между процессами будет осуществляться с помощью двух неименованых каналов (pipe’ов) «odd» и «even». Процесс-родитель осуществляет чтение названий двух файлов из стандартного ввода, в который процесс-child будет осуществлять вывод инвертированных строк, пришедших с неименованного канала. Процесс-ребенок перенаправляет поток ввода на один из пайпов («even» или «odd») и поток вывода на файл и будет инвертировать строки,находящиеся в неименованном канале, и выводить их в нужный файл.

Для работы программы нужно научиться создавать неименованые каналы и передавать информацию между процессами. Совместная работа обеспечивается благодаря системным вызовам read, write, close, которые используют файловые дескрипторы.

**Использованные системные вызовы:**

* **pid\_t fork(void)** – создает дочерний процесс. Если это значение будет отрицательным, следовательно при порождении процесса произошла ошибка. Процесс-потомок получает в качестве кода возврата значение 0, если вызов fork()оказался успешным;
* **int pipe(int pipefd[2])** – создание неименованного канала для передачи данных между процессами;
* **int close(int fd)** – закрывает файловый дескриптор;
* **Int write(int fd, void \*buf,size\_t count)** – записывает до count байтов из буфера buf в файл, на который ссылается файловый дескриптор fd.

## **2. Исходный код**

Parent.c

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <string.h>

#include <sys/wait.h>

#include <sys/stat.h>

#include <fcntl.h>

int main()

{

int even[2];

int odd[2];

char temp = 'c';

char \*filename;

char \*filename2;

int i = 0;

filename = (char \*)malloc(sizeof(char));

printf("Enter file1 name: ");

while (temp != '\n') {

scanf("%c", &temp);

i++;

filename = (char \*)realloc(filename, i \* sizeof(char));

filename[i - 1] = temp;

}

filename[i - 1] = '\0';

int file = open(filename, O\_WRONLY);

if (file == -1) {

perror("open");

exit(EXIT\_FAILURE);

}

temp='c';

filename2 = (char \*)malloc(sizeof(char));

i=0;

printf("Enter file2 name: ");

while (temp != '\n') {

scanf("%c", &temp);

i++;

filename2 = (char \*)realloc(filename2, i \* sizeof(char));

filename2[i - 1] = temp;

}

filename2[i - 1] = '\0';

int file2 = open(filename2, O\_WRONLY);

if (file2 == -1) {

perror("open");

exit(EXIT\_FAILURE);

}

if (pipe(even) == -1) {

perror("odd");

return -5;

}

if (pipe(odd) == -1) {

perror("even");

return -5;

}

pid\_t pid = fork();

if (pid == 0) {

if (dup2(file, fileno(stdout)) == -1) {

perror("dup2 stdout");

exit(EXIT\_FAILURE);

}

if (dup2(odd[0], fileno(stdin)) == -1) {

perror("dup2 stdin");

exit(EXIT\_FAILURE);

}

execl("child.out", "child.out", NULL);

}

if (pid > 0) {

char str;

pid\_t pid2;

int counter = 0;

pid2 = fork();

if (pid2 == 0) {

if (dup2(file2, fileno(stdout)) == -1) {

perror("dup2 stdout");

exit(EXIT\_FAILURE);

}

if (dup2(even[0], fileno(stdin)) == -1) {

perror("dup2 stdin");

exit(EXIT\_FAILURE);

}

execl("child.out", "child.out", NULL);

}

if (pid2 > 0) {

while ((str = getchar()) != EOF) {

if ((counter == 0) && str != '\n') {

write(even[1], &str, sizeof(char));

}

else if ((counter == 0) && str == '\n') {

write(even[1], &str, sizeof(char));

counter += 1;

}

else if (counter == 1 && str != '\n') {

write(odd[1], &str, sizeof(char));

}

else if (counter == 1 && str == '\n') {

write(odd[1], &str, sizeof(char));

counter = 0;

}

}

}

}

close(file);

close(file2);

close(even[1]);

close(odd[1]);

close(even[0]);

close(odd[0]);

return 0;

}

Child.c

#include <stdio.h>

#include <unistd.h>

int main()

{

char string[100];

char nstring[100];

int i = 0;

int ni = 0;

char c;

while ((c = getchar()) != EOF)

{

if (c != '\n')

{

string[i] = c;

i += 1;

}

else

{

while (ni < i)

{

nstring[i - 1 - ni] = string[ni];

ni += 1;

}

ni = 0;

while (ni < i)

{

printf("%c", nstring[ni]);

fflush(stdout);

ni += 1;

}

i = 0;

ni = 0;

printf("\n");

fflush(stdout);

}

}

close(STDOUT\_FILENO);

return 0;

}

## **3. Консоль**

**$ make**

gcc parent.c -o prog.out -Wall -Wextra

gcc child.c -o child.out -Wall –Wextra

**$ strace ./prog.out**

execve("./prog.out", ["./prog.out"], 0x7ffd1b24bb70 /\* 52 vars \*/) = 0

brk(NULL) = 0x5603f1c26000

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

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=104259, ...}) = 0

mmap(NULL, 104259, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f46182b9000

close(3) = 0

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

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\20\35\2\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2030928, ...}) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f46182b7000

mmap(NULL, 4131552, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f4617cb9000

mprotect(0x7f4617ea0000, 2097152, PROT\_NONE) = 0

mmap(0x7f46180a0000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e7000) = 0x7f46180a0000

mmap(0x7f46180a6000, 15072, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f46180a6000

close(3) = 0

arch\_prctl(ARCH\_SET\_FS, 0x7f46182b8500) = 0

mprotect(0x7f46180a0000, 16384, PROT\_READ) = 0

mprotect(0x5603f1b35000, 4096, PROT\_READ) = 0

mprotect(0x7f46182d3000, 4096, PROT\_READ) = 0

munmap(0x7f46182b9000, 104259) = 0

brk(NULL) = 0x5603f1c26000

brk(0x5603f1c47000) = 0x5603f1c47000

fstat(1, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(136, 1), ...}) = 0

fstat(0, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(136, 1), ...}) = 0

write(1, "Enter file1 name: ", 18Enter file1 name: ) = 18

read(0, res

"res\n", 1024) = 4

openat(AT\_FDCWD, "res", O\_WRONLY) = 3

write(1, "Enter file2 name: ", 18Enter file2 name: ) = 18

read(0, res2

"res2\n", 1024) = 5

openat(AT\_FDCWD, "res2", O\_WRONLY) = 4

pipe([5, 6]) = 0

pipe([7, 8]) = 0

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f46182b87d0) = 910

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f46182b87d0) = 911

read(0, hello kitty

"hello kitty\n", 1024) = 12

write(6, "h", 1) = 1

write(6, "e", 1) = 1

write(6, "l", 1) = 1

write(6, "l", 1) = 1

write(6, "o", 1) = 1

write(6, " ", 1) = 1

write(6, "k", 1) = 1

write(6, "i", 1) = 1

write(6, "t", 1) = 1

write(6, "t", 1) = 1

write(6, "y", 1) = 1

write(6, "\n", 1) = 1

read(0, bye kitty

"bye kitty\n", 1024) = 10

write(8, "b", 1) = 1

write(8, "y", 1) = 1

write(8, "e", 1) = 1

write(8, " ", 1) = 1

write(8, "k", 1) = 1

write(8, "i", 1) = 1

write(8, "t", 1) = 1

write(8, "t", 1) = 1

write(8, "y", 1) = 1

write(8, "\n", 1) = 1

read(0, "", 1024) = 0

close(3) = 0

close(4) = 0

exit\_group(0) = ?

+++ exited with 0 +++

**$ cat res**

yttik eyb

**$ cat res2**

yttik olleh

**4. Вывод**

Лабораторная работа помогла мне понять принцип работы с неименованными каналами для взаимодействия между процессами, принцип работы с дескрипторами и переопределением потоков ввода и вывода.