**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: Соколов Даниил Витальевич

Группа: М8О-207Б-20

Преподаватель: Е. С. Миронов

Вариант: 19

Дата:

Оценка:

# **Лабораторная работа №2**

## **1. Описание**

Программа должна создавать два дочерних процесса через fork(), с помощью pipe создавать канал для отправки данных процессам.

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.

Вариант 19) Правило фильтрации: с вероятностью 80% строки отправляются в pipe1, иначе в pipe2. Дочерние процессы удаляют все гласные из строк.

**2. Исходный код**

#include <unistd.h>

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <sys/wait.h>

#include <ctype.h>

#define MAX 255

char \* WorkProc(char \* name, char \* newstring);

char \* gets\_s(char \* begin, int max); // svoyz realizaciya gets()

int checker(char x);

int main(void) {

char string[MAX];

char string1[MAX];

char string2[MAX];

char \* str = NULL;

int num = 0, exitfromprog = 1;

int pipefd1[2], pipefd2[2];

if (pipe(pipefd1) < 0 || pipe(pipefd2) < 0) {

perror("pipe");

exit(EXIT\_FAILURE);

}

pid\_t child1 = fork();

if (child1 == -1) {

perror("fork");

exit(EXIT\_FAILURE);

}

if (child1) {

pid\_t child2 = fork();

if (child2 == -1) {

perror("fork");

exit(EXIT\_FAILURE);

}

if (!child2) { // child2

close(pipefd2[1]);

char name[MAX];

read(pipefd2[0], name, MAX);

while (1) {

char newstring[MAX];

printf("WORK 2\n");

read(pipefd2[0], newstring, MAX);

WorkProc(name, newstring);

printf("WORK 2 (finished)\n");

}

} else { // parent

printf("Enter filename for child1:\n");

gets\_s(string1, MAX);

printf("Enter filename for child2:\n");

gets\_s(string2, MAX);

write(pipefd1[1], string1, MAX);

write(pipefd2[1], string2, MAX);

while(1) {

srand(time(NULL));

num = rand() % 100 + 1;

gets\_s(string, MAX);

str = strchr(string, '\0');

\*str = '\n';

printf("%s", string);

if (num >= 1 && num <= 80) {

printf("work 1 process\n");

write(pipefd1[1], string, MAX);

} else {

printf("work 2 process\n");

write(pipefd2[1], string, MAX);

}

}

}

} else if (!child1) { // child1

close(pipefd1[1]);

char name[MAX];

read(pipefd1[0], name, MAX);

while(1) {

printf("WORK 1\n");

char newstring[MAX];

read(pipefd1[0], newstring, MAX);

WorkProc(name, newstring);

printf("WORK 1 (finished)\n");

}

}

return 0;

}

char \* gets\_s(char \* begin, int max) {

char \* start = NULL;

char tmp[max];

char \* sym;

start = fgets(tmp, max, stdin);

if (start) {

sym = strchr(tmp, '\n');

if (sym) {

\*sym = '\0';

} else {

\*(sym + max - 1) = '\0';

}

strncpy(begin, tmp, max);

} else

exit(EXIT\_FAILURE);

return begin;

}

char \* WorkProc(char \* name, char \* newstring) {

FILE \* file = fopen(name, "a+");

char sym;

while ((sym = \*newstring) != '\n' && (sym = \*newstring) != '\0') {

if (!checker(sym)) {

putc(sym, file);

putchar(sym);

}

newstring++;

}

putc('\n', file);

putchar('\n');

fclose(file);

}

int checker(char x) {

x = tolower(x);

if (x == 'a' || x == 'e' || x == 'y' || x == 'u' || x == 'o') {

return 1;

} else

return 0;

}

## **3. Тестирование программы и вывод команды strace**

## danchik@danchik-MS-7A20:~/Desktop/repositories/Prac2/OS$ ./lab2

## Enter filename for child1:

## h1

## Enter filename for child2:

## h2

## WORK 1

## WORK 2

## wegwegw

## wegwegw

## work 1 process

## wgwgw

## WORK 1 (finished)

## WORK 1

## qwrqwtttqw

## qwrqwtttqw

## work 1 process

## qwrqwtttqw

## WORK 1 (finished)

## WORK 1

## erwyhewh

## erwyhewh

## work 1 process

## rwhwh

## WORK 1 (finished)

## WORK 1

## efkjwkerg

## efkjwkerg

## work 1 process

## fkjwkrg

## WORK 1 (finished)

## WORK 1

## wegkwkge

## wegkwkge

## work 1 process

## wgkwkg

## WORK 1 (finished)

## WORK 1

## wegjweigweg

## wegjweigweg

## work 1 process

## wgjwigwg

## WORK 1 (finished)

## WORK 1

## wegpjwpkegw

## wegpjwpkegw

## work 1 process

## wgpjwpkgw

## WORK 1 (finished)

## WORK 1

## gwegkwegweigl;oweghko;woeh

## gwegkwegweigl;oweghko;woeh

## work 1 process

## gwgkwgwigl;wghk;wh

## WORK 1 (finished)

## WORK 1

## Hi 2 proc

## Hi 2 proc

## work 2 process

## Hi 2 prc

## WORK 2 (finished)

## WORK 2

## ^C

## 

## danchik@danchik-MS-7A20:~/Desktop/repositories/Prac2/OS$ strace ./lab2

## execve("./lab2", ["./lab2"], 0x7fff5540b480 /\* 59 vars \*/) = 0

## brk(NULL) = 0x55b837aab000

## arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffc5508a140) = -1 EINVAL (Недопустимый аргумент)

## access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (Нет такого файла или каталога)

## openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

## fstat(3, {st\_mode=S\_IFREG|0644, st\_size=87927, ...}) = 0

## mmap(NULL, 87927, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f33cca9f000

## 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\360q\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

## pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32

## pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\t\233\222%\274\260\320\31\331\326\10\204\276X>\263"..., 68, 880) = 68

## fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2029224, ...}) = 0

## mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f33cca9d000

## 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

## pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32

## pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\t\233\222%\274\260\320\31\331\326\10\204\276X>\263"..., 68, 880) = 68

## mmap(NULL, 2036952, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f33cc8ab000

## mprotect(0x7f33cc8d0000, 1847296, PROT\_NONE) = 0

## mmap(0x7f33cc8d0000, 1540096, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x25000) = 0x7f33cc8d0000

## mmap(0x7f33cca48000, 303104, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x19d000) = 0x7f33cca48000

## mmap(0x7f33cca93000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e7000) = 0x7f33cca93000

## mmap(0x7f33cca99000, 13528, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f33cca99000

## close(3) = 0

## arch\_prctl(ARCH\_SET\_FS, 0x7f33cca9e540) = 0

## mprotect(0x7f33cca93000, 12288, PROT\_READ) = 0

## mprotect(0x55b835b0a000, 4096, PROT\_READ) = 0

## mprotect(0x7f33ccae2000, 4096, PROT\_READ) = 0

## munmap(0x7f33cca9f000, 87927) = 0

## pipe([3, 4]) = 0

## pipe([5, 6]) = 0

## clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f33cca9e810) = 4302

## clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f33cca9e810) = 4303

## fstat(1, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}) = 0

## brk(NULL) = 0x55b837aab000

## brk(0x55b837acc000) = 0x55b837acc000

## write(1, "Enter filename for child1:\n", 27Enter filename for child1:

## ) = 27

## fstat(0, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}) = 0

## read(0, g1

## "g1\n", 1024) = 3

## write(1, "Enter filename for child2:\n", 27Enter filename for child2:

## ) = 27

## read(0, g2

## "g2\n", 1024) = 3

## write(4, "g1\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 255) = 255

## write(6, "g2\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 255) = 255

## read(0, WORK 1

## WORK 2

## qwerty

## "qwerty\n", 1024) = 7

## write(1, "qwerty\n", 7qwerty

## ) = 7

## write(1, "work 1 process\n", 15work 1 process

## ) = 15

## write(4, "qwerty\n\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 255) = 255

## read(0, qwrt

## WORK 1 (finished)

## WORK 1

## dfh

## "dfh\n", 1024) = 4

## write(1, "dfh\n", 4dfh

## ) = 4

## write(1, "work 1 process\n", 15work 1 process

## ) = 15

## write(4, "dfh\n\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 255) = 255

## dfh

## read(0, WORK 1 (finished)

## WORK 1

## dhfetrj

## "dhfetrj\n", 1024) = 8

## write(1, "dhfetrj\n", 8dhfetrj

## ) = 8

## write(1, "work 1 process\n", 15work 1 process

## ) = 15

## write(4, "dhfetrj\n\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 255) = 255

## dhftrj

## read(0, WORK 1 (finished)

## WORK 1

## 3ley54oy[h54

## "3ley54oy[h54\n", 1024) = 13

## write(1, "3ley54oy[h54\n", 133ley54oy[h54

## ) = 13

## write(1, "work 1 process\n", 15work 1 process

## ) = 15

## write(4, "3ley54oy[h54\n\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 255) = 255

## 3l54[h54

## read(0, WORK 1 (finished)

## WORK 1

## rdjptpjr

## "rdjptpjr\n", 1024) = 9

## write(1, "rdjptpjr\n", 9rdjptpjr

## ) = 9

## write(1, "work 2 process\n", 15work 2 process

## ) = 15

## write(6, "rdjptpjr\n\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 255) = 255

## read(0, rdjptpjr

## WORK 2 (finished)

## WORK 2

## ^Cstrace: Process 4301 detached

## <detached ...>

## 

## **4. Вывод**

В ходе лабораторной работы я научился работать с процессами и создавать дочерние процессы и взаимодейстововать между ними через pipe. Я приобрёл навыки по распараллеливанию программы на процессы. Для выполнения программы мне пришлось писать собственную функцию вводу строк (замена gets()), с помощью функции rand() и time создал вероятность попадания строки в отдельный файл, и все это в разных процессах. Интересная лабораторная работа, которая заставила меня вникнуть в тему многопроцессного программирования.