Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

Группа: М80-206Б-18

Студент: Тугарев М. А.

Преподаватель: Соколов А.А.

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

Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2019.

# Содержание

1. Постановка задачи.
2. Общие сведения о программе.
3. Общий метод и алгоритм решения.
4. Код программы.
5. Демонстрация работы программы.
6. Вывод.

## Постановка задачи

Составить и отладить программу на языке Си, где дочерний процесс представляет собой сервер по работе с деревом общего вида и принимает команды со стороны родительского процесса.

## Общий метод и алгоритм решения

* Удалить файлы — именованные каналы, если они существуют. Далее создаем 2 именованных канала для передачи от клиента серверу и наоборот.
* Создаем дочерний процесс, вызываем программу - сервер, передаем ей списком имена каналов.
* После ввода команды она передается серверу, где она считывается, выполняется и с помощью метода writeToFile возвращает результат работы клиенту
* После ввода команды «stop» программа закрывает и удаляет именованные каналы, прекращая свою работу.

## Код программы

**main.cpp:**

void readFromServer(const char\* toClient, std::vector<std::string>& in)

{

in.clear();

int fd;

if((fd = open(toClient, O\_RDONLY)) < 0)

{

std::cout << "can not open FIFO for writing\n";

exit(-1);

}

int size;

char forRead[100];

size = read(fd, forRead, 100);

if(size < 0)

{

printf("Can\'t read string\n");

exit(-1);

}

int times = std::atoi(forRead);

for(int i = 0; i < times; ++i)

{

size = read(fd, forRead, 100);

if(size < 0)

{

printf("Can\'t read string\n");

exit(-1);

}

in.push\_back(forRead);

}

close(fd);

}

int main(int argc, char \*argv[])

{

int fd, result;

int size;

char toServer[] = "/tmp/toServer";

char toClient[] = "/tmp/toClient";

// Creating named Channel

std::cout << "STARTING" << std::endl;

if(remove(toServer) != 0){}

if(remove(toClient) != 0){}

if(mknod(toServer, S\_IFIFO | 0666, 0) < 0) {

printf("Can\'t create FIFO\n");

}

if(mknod(toClient, S\_IFIFO | 0666, 0) < 0) {

printf("Can\'t create FIFO\n");

}

pid\_t pid = fork();

if(pid < 0)

{

perror("FORK ERROR");

exit(1);

}

else if(pid == 0)

{

execl("./server", toServer, toClient, NULL);

perror("EXEC ERROR");

exit(1);

}

else if(pid >0)

{

std::string buf = "";

while(buf != "stop")

{

std::getline(std::cin, buf);

if(buf.size() > 100)

{

std::cout << "Too Big buf" << std::endl;

}

if((fd = open(toServer, O\_WRONLY)) < 0)

{

std::cout << "can not open FIFO for writing CLIENT\n";

exit(-1);

}

size = write(fd, buf.c\_str(), 100);

if(size != 100)

{

std::cout << "Can't write all string to FIFO CLIENT\n";

exit(-1);

}

close(fd);

std::vector<std::string> feedback;

readFromServer(toClient, feedback);

for(auto &str : feedback)

{

std::cout << str << std::endl;

}

}

//delete named channels

if(remove(toServer) != 0)

{

std::cout << "File Delete Error" << std::endl;

}

if(remove(toClient) != 0)

{

std::cout << "File Delete Error" << std::endl;

}

}

return 0;

}

server.cpp:

void writeToClient(const char\* toClient, const std::vector<std::string>& toWrite)

{

int fd;

if((fd = open(toClient, O\_WRONLY)) < 0) {

std::cout << "Can\'t open FIFO for writing SERVER\n";

exit(-1);

}

int size;

size = write(fd, std::to\_string(toWrite.size()).c\_str(), 100);

if(size != 100)

{

std::cout << "Can't write all string to FIFO SERVER\n";

exit(-1);

}

//std::cout << "THIS IS WRITE TO CLIENT " << toWrite.size() << std::endl;

for(auto &str : toWrite)

{

//std::cout << str << std::endl;

size = write(fd, str.c\_str(), 100);

if(size != 100)

{

std::cout << "Can't write all string to FIFO SERVER\n";

exit(-1);

}

}

close(fd);

}

int main(int argc, char \*argv[])

{

char\* toServer = argv[0];

char\* toClient = argv[1];

int fd;

int size;

char buf[100];

std::string com = "";

Tree::Tree<int> tree(0);

while(com != "stop")

{

if((fd = open(toServer, O\_RDONLY)) < 0){

printf("Can\'t open FIFO for reading SERVER\n");

exit(-1);

}

size = read(fd, buf, 100);

if(size < 0)

{

printf("Can\'t read string\n");

exit(-1);

}

close(fd);

com = buf;

std::vector<std::string> toWrite;

if(com == "stop")

{

toWrite.push\_back("Stopping Server");

}

if(com.size() > 0 && com[0] == '+')

{

if(tree.addLeaf(com) == -1)

{

toWrite.push\_back("Badbuf");

}

else

{

toWrite.push\_back("Leaf added");

}

}

if(com.size() > 0 && com[0] == '-')

{

toWrite.push\_back(std::to\_string(tree.getLeaf(com)));

}

writeToClient(toClient, toWrite);

}

exit(0);

}

## Демонстрация работы программы

miksail@miksail:~/Desktop/OC\_lab/OC\_lab1$ ./client

STARTING

+s 1231

Bad command

+ 7

Leaf added

+ 9

Leaf added

+s 12

Leaf added

+sb 35

Leaf added

-s

Returned value: 7

-sb

Returned value: 9

-ss

Returned value: 12

-sbs

Returned value: 35

stop

Stopping Server

## Вывод

В результате данной лабораторной работы я научился создавать процессы, работать с методом «exec». Понял как работать с именованными каналами и передавать данные между процессами.