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»

Выполнил: Е. Г. Туймуков
Группа: М8О-208БВ-24
Преподаватель: Е. С. Миронов

Москва, 2025

Условие

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.

Цель работы: Изучение механизмов создания процессов, организации межпроцессного взаимодействия через pipes и обработки данных в многопроцессной архитектуре.

Задание: Правило фильтрации: строки длины больше 10 символов отправляются в pipe2, иначе в pipe1. Дочерние процессы инвертируют строки.

Вариант: 20

Метод решения

Данная программа реализует многопроцессную обработку текстовых данных с использованием каналов (pipes) для межпроцессного взаимодействия. Основной алгоритм: родительский процесс запрашивает имена двух файлов, читает строки из стандартного ввода и направляет их в два дочерних процесса по правилу фильтрации: строки длиной более 10 символов отправляются второму процессу, остальные - первому. Каждый дочерний процесс получает строки из своего канала, инвертирует их (переворачивает задом наперёд) и записывает результаты в указанный файл, а также выводит в стандартный вывод (stdout).

Ключевые компоненты: ParentProcess - управляет каналами и дочерними процессами Pipe - кроссплатформенная реализация каналов ChildProcess - запускает дочерние процессы utils - содержит функцию для инверсии строк

Системные вызовы: Windows: CreatePipe, CreateProcess, ReadFile, WriteFile, CloseHandle, WaitForSingleObject Linux/macOS: pipe, fork, execvp, read, write, close, waitpid, dup2

Программа использует объектно-ориентированный подход с инкапсуляцией платформозависимых особенностей, что обеспечивает кроссплатформенность и четкое разделение ответственности между модулями.

Описание программы

Программа реализует многопроцессную обработку текстовых данных через каналы (pipes). Родительский процесс запрашивает у пользователя имена двух файлов для записи результатов, читает строки из стандартного ввода и распределяет их между двумя дочерними процессами: строки длиной более 10 символов отправляются второму процессу, остальные - первому. Каждый дочерний процесс читает строки из своего канала, переворачивает их задом наперёд и записывает результат в указанный файл, а также выводит в стандартный вывод (stdout).

Архитектура программы включает несколько модулей. В parent.cpp находится точка входа, создающая объект ParentProcess. Класс ParentProcess (parentprocess.cpp) управляет всей работой: создает каналы, запрашивает имена файлов, запускает дочерние процессы и распределяет строки по правилу фильтрации. Класс Pipe (pipe.cpp) инкапсулирует работу с каналами, используя CreatePipe на Windows и pipe на Linux/macOS. Класс ChildProcess (childprocess.cpp) отвечает за запуск дочерних процессов через CreateProcess (Windows)

или fork/execvp (Linux/macOS). Файл child.cpp реализует дочерний процесс, который читает строки из стандартного ввода (подключенного к каналу), инвертирует их с помощью функции reverseString из utils.cpp и записывает в файл и stdout. Модули oslinux.cpp и oswin.cpp обеспечивают кроссплатформенность, реализуя системные вызовы для соответствующих платформ.

Результаты

Разработанная программа успешно реализует многопроцессную архитектуру для параллельной обработки текстовых данных.
В ходе решения были достигнуты следующие ключевые результаты:
Корректная работа системы межпроцессного взаимодействия
Реализованы два независимых канала передачи данных между родительским и дочерними процессами
Обеспечено четкое распределение строк по длине строк
Достигнута синхронизация процессов через блокирующие операции чтения/записи
Кросс-платформенная функциональность
Программа корректно работает как в Windows, так и в Linux/Unix системах
Реализована унифицированная абстракция для работы с каналами через класс Pipe
Обеспечен единый интерфейс для создания процессов на разных платформах

Выводы

В ходе лабораторной работы успешно разработана многопроцессная система обработки текстовых данных с использованием межпроцессного взаимодействия через каналы. Программа демонстрирует корректную работу в кросс-платформенном режиме на Windows и Unix системах.

Исходная программа

childprocess.cpp

```
#include "../include/child.process.hpp"
#include "../include/os.h"

ChildProcess::ChildProcess(Pipe* p,const std::string& f,bool is.c1)
: pipe(p),file.name(f),is.child1(is.c1),pid(INVALID.PIPE.HANDLE) {}

void ChildProcess::execute() {
#ifdef .WIN32
const char* exe = "child.exe";
#else
const char* exe = "./child";
#endif
char* argv[] = { (char*)"child",(char*)file.name.c.str(),NULL };
pid = CreateChildProcess(exe,argv,pipe->getReadFd());
}
```

child.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include "../include/utils.h"

int main(int argc,char** argv) {
if (argc <2) {
std::cerr <<"No filename provided" <<std::endl;
return 1;
}
std::string filename = argv[1];
std::ofstream file(filename);
if (!file.is.open()) {
std::cerr <<"Failed to open file: " <<filename <<std::endl;
return 1;
}

std::string line;
while (std::getline(std::cin,line)) {
std::string reversed = reverseString(line);
file <<reversed <<std::endl;
std::cout <<reversed <<std::endl;  // Вывод в stdout
}

file.close();
return 0;
}
```

os.linux.cpp

```
#include "../include/os.h"
#include <fcntl.h>
#include <unistd.h>
#include <sys/wait.h>
#include <cstdio> // Для perror

int CreatePipe(PipeHandle fd[2]) {
return pipe(fd);
}

int PipeRead(PipeHandle fd,char* buf,size.t size) {
return read(fd,buf,size);
}

int PipeWrite(PipeHandle fd,const char* buf,size.t size) {
return write(fd,buf,size);
}

int ClosePipe(PipeHandle fd) {
return close(fd);
}

ProcessHandle CreateChildProcess(const char* exe,char* const* argv,PipeHandle
stdin.handle) {
pid.t pid = fork();
if (pid == -1) {
perror("fork");
return -1;
}
if (pid == 0) {
// Child
if (stdin.handle != INVALID.PIPE.HANDLE) {
Dup2(stdin.handle,STDIN.FILENO);
ClosePipe(stdin.handle);
}
// Закрыть write end,но поскольку pipe в parent,child не имеет его
execvp(exe,argv);
perror("execvp");
.exit(1);
}
return pid;
}

int WaitProcess(ProcessHandle pid) {
int status;
return waitpid(pid,&status,0);
}
```

```
int Dup2(PipeHandle oldfd,int newfd) {
return dup2(oldfd,newfd);
}

os.win.cpp
#include "../include/os.h"
#include <iostream> // Для std::cerr

int CreatePipe(PipeHandle fd[2]) {
SECURITY.ATTRIBUTES saAttr;
saAttr.nLength = sizeof(SECURITY.ATTRIBUTES);
saAttr.bInheritHandle = TRUE;
saAttr.lpSecurityDescriptor = NULL;

if (!CreatePipe(&fd[0],&fd[1],&saAttr,0)) {
std::cerr <<"CreatePipe failed" <<std::endl;
return -1;
}

// Make write end non-inheritable
if (!SetHandleInformation(fd[1],HANDLE.FLAG.INHERIT,0)) {
std::cerr <<"SetHandleInformation failed" <<std::endl;
return -1;
}

return 0;
}

int PipeRead(PipeHandle fd,char* buf,size.t size) {
DWORD bytesRead;
if (!ReadFile(fd,buf,size,&bytesRead,NULL)) {
return -1;
}
return bytesRead;
}

int PipeWrite(PipeHandle fd,const char* buf,size.t size) {
DWORD bytesWritten;
if (!WriteFile(fd,buf,size,&bytesWritten,NULL)) {
return -1;
}
return bytesWritten;
}

int ClosePipe(PipeHandle fd) {
return !CloseHandle(fd);
}
```

```
ProcessHandle CreateChildProcess(const char* exe,char* const* argv,PipeHandle
stdin.handle) {
PROCESS.INFORMATION pi;
STARTUPINFO si;
ZeroMemory(&si,sizeof(si));
si.cb = sizeof(si);
ZeroMemory(&pi,sizeof(pi));

si.dwFlags |= STARTF.USESTDHANDLES;
si.hStdInput = stdin.handle;
si.hStdOutput = GetStdHandle(STD.OUTPUT.HANDLE);  // Оставляем stdout как есть
si.hStdError = GetStdHandle(STD.ERROR.HANDLE);

std::string cmdline;
for (int i = 0; argv[i]; ++i) {
if (i >0) cmdline += " ";
cmdline += argv[i];
}

if (!CreateProcessA(NULL,(LPSTR)cmdline.c.str(),NULL,NULL,TRUE,0,NULL,NULL,&si,&pi))
{
std::cerr <<"CreateProcess failed (" <<GetLastError() <<")" <<std::endl;
return INVALID.HANDLE.VALUE;
}

// Close thread handle
CloseHandle(pi.hThread);

return pi.hProcess;
}

int WaitProcess(ProcessHandle pid) {
WaitForSingleObject(pid,INFINITE);
CloseHandle(pid);
return 0;
}

int Dup2(PipeHandle oldfd,int newfd) {
// No-op on Windows,since we set in STARTUPINFO
return 0;
}

parentprocess.cpp
#include "../include/parent.process.hpp"
#include <iostream>

ParentProcess::ParentProcess()
: pipe1(new Pipe()),pipe2(new Pipe()),child1(nullptr),child2(nullptr) {}
```

```
ParentProcess::~ParentProcess() {
delete pipe1;
delete pipe2;
delete child1;
delete child2;
}

void ParentProcess::start() {
std::cout <<"Enter filename for child1: ";
std::getline(std::cin,file1);
std::cout <<"Enter filename for child2: ";
std::getline(std::cin,file2);

child1 = new ChildProcess(pipe1,file1,true);
child1->execute();
// Close read end in parent
ClosePipe(pipe1->getReadFd());

child2 = new ChildProcess(pipe2,file2,false);
child2->execute();
// Close read end in parent
ClosePipe(pipe2->getReadFd());

std::cout <<"Enter lines (empty line to end):" <<std::endl;
std::string line;
while (std::getline(std::cin,line)) {
if (line.empty()) break;
Pipe* target.pipe = (line.length() >10) ? pipe2 : pipe1;
target.pipe->write(line.c.str(),line.length());
target.pipe->write("\n",1);
}

ClosePipe(pipe1->getWriteFd());
ClosePipe(pipe2->getWriteFd());

WaitProcess(child1->getPid());
WaitProcess(child2->getPid());

std::cout <<"Parent: children finished." <<std::endl;
}

parent.cpp
#include "../include/parent.process.hpp"

int main() {
ParentProcess parent;
parent.start();
```

```
return 0;
}

pipe.cpp
#include "../include/pipe.hpp"

Pipe::Pipe() {
if (CreatePipe(fd) != 0) {
// Обработка ошибки,но для простоты пропустим
}
}

Pipe::~Pipe() {
ClosePipe(fd[0]);
ClosePipe(fd[1]);
}

ssize.t Pipe::read(char* buf,size.t size) {
return PipeRead(fd[0],buf,size);
}

ssize.t Pipe::write(const char* buf,size.t size) {
return PipeWrite(fd[1],buf,size);
}

cmake
#include "../include/utils.h"

std::string reverseString(const std::string& s) {
std::string rev;

// Просто идем с конца строки и добавляем символы в новую строку
for (int i = s.size() -1; i >= 0; i--) {
rev += s[i];
}

return rev;
}


utils.cpp
cmake.minimum.required(VERSION 3.10)
project(lab1)

set(CMAKE.CXX.STANDARD 11)
set(CMAKE.CXX.STANDARD.REQUIRED ON)

include.directories(include)
```

```
set(UTILS.SOURCES src/utils.cpp)

if(UNIX)
set(PLATFORM.SOURCES src/os.linux.cpp)
elseif(WIN32)
set(PLATFORM.SOURCES src/os.win.cpp)
endif()

add.executable(parent
src/parent.cpp
src/parent.process.cpp
src/pipe.cpp
src/child.process.cpp
${UTILS.SOURCES}
${PLATFORM.SOURCES}
)

add.executable(child
src/child.cpp
${UTILS.SOURCES}
)

if(WIN32)
target.link.libraries(parent ws2.32)
endif()
```

Логи

```
9410  execve("./parent",["./parent"],["SHELL=/bin/bash","SESSION.MANAGER=local/242432
ERROR;JS LOG","HOME=/home/modmod","USERNAME=modmod","IM.CONFIG.PHASE=1","LANG=ru.RU.U
%s %s","XDG.SESSION.CLASS=user","TERM=xterm-256color","GTK.PATH=/snap/code/206/usr/li
/usr/bin/lesspipe %s","USER=modmod","GTK.PATH.VSCODE.SNAP.ORIG=","VSCODE.GIT.IPC.HAND
","FONTCONFIG.FILE=/etc/fonts/fonts.conf","VSCODE.GIT.ASKPASS.MAIN=/snap/code/206/usr
"GDK.BACKEND=x11","PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:
= 0
9410  brk(NULL)                         = 0x631811d88000
9410  mmap(NULL,8192,PROT.READ|PROT.WRITE,MAP.PRIVATE|MAP.ANONYMOUS,-1,0) =
0x70c54d0a5000
9410  access("/etc/ld.so.preload",R.OK) = -1 ENOENT (Нет такого файла или каталога)
9410  openat(AT.FDCWD,"/etc/ld.so.cache",O.RDONLY|O.CLOEXEC) = 3
9410  fstat(3,{st.dev=makedev(0x103,0x8),st.ino=7080911,st.mode=S.IFREG|0644,st.nlink
/* 2025-10-13T12:49:14.467681259+0300 */,st.atime.nsec=467681259,st.mtime=1760348954
/* 2025-10-13T12:49:14.435683805+0300 */,st.mtime.nsec=435683805,st.ctime=1760348954
/* 2025-10-13T12:49:14.437683646+0300 */,st.ctime.nsec=437683646}) = 0
9410  mmap(NULL,94467,PROT.READ,MAP.PRIVATE,3,0) = 0x70c54d08d000
9410  close(3)                          = 0
9410  openat(AT.FDCWD,"/lib/x86.64-linux-gnu/libstdc++.so.6",O.RDONLY|O.CLOEXEC)
= 3
```

```
ed>)           = 0
9446  <... mprotect resumed>)           = 0
9447  prlimit64(0,RLIMIT.STACK,NULL, <unfinished ...>
9446  mprotect(0x7cfcb3861000,8192,PROT.READ <unfinished ...>
9447  <... prlimit64 resumed>{rlim.cur=8192*1024,rlim.max=RLIM64.INFINITY})
= 0
9446  <... mprotect resumed>)           = 0
9447  munmap(0x742159ad8000,94467 <unfinished ...>
9446  prlimit64(0,RLIMIT.STACK,NULL, <unfinished ...>
9447  <... munmap resumed>)             = 0
9446  <... prlimit64 resumed>{rlim.cur=8192*1024,rlim.max=RLIM64.INFINITY})
= 0
9446  munmap(0x7cfcb380b000,94467 <unfinished ...>
9447  futex(0x742159a7a7bc,FUTEX.WAKE.PRIVATE,2147483647) = 0
9446  <... munmap resumed>)             = 0
9447  getrandom( <unfinished ...>
9446  futex(0x7cfcb367a7bc,FUTEX.WAKE.PRIVATE,2147483647 <unfinished ...>
9447  <... getrandom resumed>"\xf6\xc1\x7b\x8c\xdf\xc3\x52\x88",8,GRND.NONBLOCK)
= 8
9446  <... futex resumed>)              = 0
9447  brk(NULL)                         = 0x57cc21206000
9447  brk(0x57cc21227000)               = 0x57cc21227000
9446  getrandom( <unfinished ...>
9447  openat(AT.FDCWD,"2",O.WRONLY|O.CREAT|O.TRUNC,0666 <unfinished ...>
9446  <... getrandom resumed>"\xeb\x77\x86\xb7\x81\x6d\x36\x63",8,GRND.NONBLOCK)
= 8
9446  brk(NULL <unfinished ...>
9447  <... openat resumed>)             = 3
9446  <... brk resumed>)                = 0x60a15ad00000
9446  brk(0x60a15ad21000 <unfinished ...>
9447  fstat(0, <unfinished ...>
9446  <... brk resumed>)                = 0x60a15ad21000
9447  <... fstat resumed>{st.dev=makedev(0,0xf),st.ino=75970,st.mode=S.IFIFO|0600,st.
/* 2025-10-13T12:56:16.575245764+0300 */,st.atime.nsec=575245764,st.mtime=1760349376
/* 2025-10-13T12:56:16.575245764+0300 */,st.mtime.nsec=575245764,st.ctime=1760349376
/* 2025-10-13T12:56:16.575245764+0300 */,st.ctime.nsec=575245764}) = 0
9446  openat(AT.FDCWD,"1",O.WRONLY|O.CREAT|O.TRUNC,0666 <unfinished ...>
9447  read(0, <unfinished ...>
9446  <... openat resumed>)             = 3
9446  fstat(0,{st.dev=makedev(0,0xf),st.ino=75969,st.mode=S.IFIFO|0600,st.nlink=1,st.
/* 2025-10-13T12:56:16.575245764+0300 */,st.atime.nsec=575245764,st.mtime=1760349376
/* 2025-10-13T12:56:16.575245764+0300 */,st.mtime.nsec=575245764,st.ctime=1760349376
/* 2025-10-13T12:56:16.575245764+0300 */,st.ctime.nsec=575245764}) = 0
9446  read(0, <unfinished ...>
9410  <... read resumed>"123\n",1024)  = 4
9410  write(4,"123",3)                 = 3
9446  <... read resumed>"123",4096)    = 3
9410  write(4,"\n",1 <unfinished ...>
```

```
9446  read(0, <unfinished ...>
9410  <... write resumed>)              = 1
9446  <... read resumed>"\n",4096)      = 1
9410  read(0, <unfinished ...>
9446  write(3,"321\n",4)                = 4
9446  fstat(1,{st.dev=makedev(0,0x1a),st.ino=3,st.mode=S.IFCHR|0620,st.nlink=1,st.uid
/* 2025-10-13T12:56:16+0300 */,st.atime.nsec=0,st.mtime=1760349376 /* 2025-10-13T12:5
*/,st.mtime.nsec=0,st.ctime=1760348474 /* 2025-10-13T12:41:14.222497184+0300
*/,st.ctime.nsec=222497184}) = 0
9446  write(1,"321\n",4)                = 4
9446  read(0, <unfinished ...>
9410  <... read resumed>"papa\n",1024) = 5
9410  write(4,"papa",4)                 = 4
9446  <... read resumed>"papa",4096)    = 4
9410  write(4,"\n",1)                   = 1
9410  read(0, <unfinished ...>
9446  read(0,"\n",4096)                 = 1
9446  write(3,"apap\n",5)               = 5
9446  write(1,"apap\n",5)               = 5
9446  read(0, <unfinished ...>
9410  <... read resumed>"popa\n",1024) = 5
9410  write(4,"popa",4)                 = 4
9446  <... read resumed>"popa",4096)    = 4
9410  write(4,"\n",1)                   = 1
9410  read(0, <unfinished ...>
9446  read(0,"\n",4096)                 = 1
9446  write(3,"apop\n",5)               = 5
9446  write(1,"apop\n",5)               = 5
9446  read(0, <unfinished ...>
9410  <... read resumed>"pobeda\n",1024) = 7
9410  write(4,"pobeda",6)               = 6
9410  write(4,"\n",1)                   = 1
9446  <... read resumed>"pobeda\n",4096) = 7
9410  read(0, <unfinished ...>
9446  write(3,"adebop\n",7)             = 7
9446  write(1,"adebop\n",7)             = 7
9446  read(0, <unfinished ...>
9410  <... read resumed>0x631811d9a700,1024) = ? ERESTARTSYS (To be restarted
if SA.RESTART is set)
9446  <... read resumed>0x60a15ad144a0,4096) = ? ERESTARTSYS (To be restarted
if SA.RESTART is set)
9447  <... read resumed>0x57cc2121a4a0,4096) = ? ERESTARTSYS (To be restarted
if SA.RESTART is set)
9410  ---SIGINT {si.signo=SIGINT,si.code=SI.KERNEL} ---
9446  ---SIGINT {si.signo=SIGINT,si.code=SI.KERNEL} ---
9447  ---SIGINT {si.signo=SIGINT,si.code=SI.KERNEL} ---
9410  +++ killed by SIGINT +++
9446  +++ killed by SIGINT +++
```

```
9447  +++ killed by SIGINT +++
```