Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №1 по курсу**

**«Операционные системы»**

Группа: М8О-209Б-22

Студент: Концебалов О.С.

Преподаватель: Пономарев Н.В.

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

Дата: 28.10.2023

Москва, 2023.

# 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 С/C++, осуществляющую работу с процессами и взаимодействие между ними в одной из двух операционных систем. В результате работы программа (основной процесс) должен создать для решение задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или каналы (pipe).

Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

## Общие сведения о программе

Программа состоит из 4 папок: include, src, run\_files, output\_files. В папке include находятся 2 папки calls, в которой находится файл с необходимыми системными вызовами calls.h, и processes, в котором находятся два файла parent\_process.cpp и child\_process.cpp с классами родительского и дочернего процессов соответственно. В папке src так же, как и в include есть две папки calls и processes с исходным кодом моего проекта. Папка run\_files содержит два файла run\_parent.cpp и run\_child.cpp с функцией main, которые будут скомпилированы в исполняемые. В output\_files находятся файлы полученные в результате работы программы.

Также есть Makefile для удобной сборки и запуска программы.

## Общий метод и алгоритм решения

Сначала запускается родительский процесс, который запрашивает у пользователя имя файла, который будет создан при работе дочернего процесса. После этого создаются пайпы, перенаправляются потоки ввода-вывода и создается дочерний процесс, в который передается полученное от пользователя имя файла. После этого получаем числа, введенные пользователем, передаем их в дочерний процесс, там суммируем и записываем в файл.

На каждом этапе проверяем выполнение всех функций на ошибки.

## Код программы

./include/calls/calls.hpp

#pragma once

#include <unistd.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <iostream>

#include <string>

namespace calls {

[[nodiscard]] pid\_t create\_process();

[[nodiscard]] int open\_file(const char\*);

void create\_dup2FD(const int, const int);

void closeFD(const int);

void create\_pipe(int\*);

void run\_process(const char\*, const char\*);

}; // namespace calls

./include/processes/parent\_process.hpp

#pragma once

#include <iostream>

#include <string>

namespace processes {

class ParentProcess final {

public:

    static void parent\_process\_handler();

private:

    [[nodiscard]] static std::string get\_file\_name();

};

}; // namespace processes

./include/processes/child\_process.hpp

#pragma once

#include <iostream>

namespace processes {

class ChildProcess final {

public:

    static void child\_process\_handler(const char\*);

};

}; // namespace processes

./src/calls/calls.cpp

#include "../../include/calls/calls.hpp"

namespace calls {

pid\_t create\_process() {

    pid\_t pid = fork();

    if (pid == -1) throw std::runtime\_error("Failed with creating child process");

    return pid;

}

int open\_file(const char\* file\_name) {

    int file = open(file\_name, O\_CREAT | O\_WRONLY | O\_TRUNC);

    if (file == -1) throw std::runtime\_error("Failed with opening file");

    return file;

}

void create\_dup2FD(const int old\_fd, const int new\_fd) {

    int dup2\_num = dup2(old\_fd, new\_fd);

    if (dup2\_num == -1) throw std::runtime\_error("Failed with creating dup2 to this file directories");

    return;

}

void closeFD(const int fd) {

    int closeFD\_num = close(fd);

    if (closeFD\_num == -1) throw std::runtime\_error("Failed with closing fd");

    return;

}

void create\_pipe(int\* fd) {

    int pipe\_num = pipe(fd);

    if (pipe\_num == -1) throw std::runtime\_error("Failed with creating pipe");

    return;

}

void run\_process(const char\* child\_process, const char\* file\_name) {

    int execl\_num = execl(child\_process, child\_process, file\_name, NULL);

    if (execl\_num == -1) throw std::runtime\_error("Failed with run child process");

    return;

}

}; // namespace calls

./src/processes/parent\_process.cpp

#include "../../include/processes/parent\_process.hpp"

#include "../../include/calls/calls.hpp"

#define CHILD\_NAME "./bin/run\_child"

using namespace processes;

using namespace calls;

void ParentProcess::parent\_process\_handler() {

    std::cout << "Parent process with pid " << getpid() << " started" << std::endl;

    std::string file\_name = get\_file\_name();

    int fd\_1[2];

    int fd\_2[2];

    create\_pipe(fd\_1);

    create\_pipe(fd\_2);

    int write\_1 = fd\_1[1], write\_2 = fd\_2[1];

    int read\_1 = fd\_1[0], read\_2 = fd\_2[0];

    pid\_t pid = create\_process();

    if (pid == 0) {

        closeFD(write\_1);

        closeFD(read\_2);

        create\_dup2FD(read\_1, STDIN\_FILENO);

        closeFD(write\_2);

        closeFD(read\_1);

        run\_process(CHILD\_NAME, file\_name.c\_str());

    } else {

        closeFD(write\_2);

        closeFD(read\_1);

        std::cout << "Parent process with pid " << getpid() << std::endl;

        uint64\_t num;

        while (std::cin >> num) {

            dprintf(write\_1, "%ld ", num);

        }

        closeFD(write\_1);

        closeFD(read\_2);

    }

    return;

}

std::string ParentProcess::get\_file\_name() {

    std::string file\_name;

    std::cout << "Input file name: ";

    std::cin >> file\_name;

    if (file\_name.length() > 255) throw std::invalid\_argument("File name must be less than 256 symbols");

    for (char c: file\_name) {

        if (c == '/' || c == '\\' || c == '?' || c == '<' || c == '>' || c == '\*' || c == '|') {

            throw std::invalid\_argument("File name can't contains /, \\, ?, <, >, \*, |");

        }

    }

    file\_name = "output\_files/" + file\_name + ".txt";

    return file\_name;

}

./src/processes/child\_process.cpp

#include "../../include/processes/child\_process.hpp"

#include "../../include/calls/calls.hpp"

using namespace processes;

using namespace calls;

void ChildProcess::child\_process\_handler(const char\* file\_name) {

    std::cout << "Child process with pid " << getpid() << " started\nInput numbers: " << std::endl;

    int file = open\_file(file\_name);

    create\_dup2FD(file, STDOUT\_FILENO);

    closeFD(file);

    int64\_t sum { 0 };

    int64\_t num;

    while (std::cin >> num) {

        sum += num;

    }

    std::cout << "Result sum: " << sum;

    return;

}

./run\_files/run\_parent.cpp

#include "../include/processes/parent\_process.hpp"

int main() {

    processes::ParentProcess::parent\_process\_handler();

    return 0;

}

./run\_files/run\_child.cpp

#include "../include/processes/child\_process.hpp"

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

    processes::ChildProcess::child\_process\_handler(argv[1]);

    return 0;

}

## Использование утилиты strace

baronpipistron@BaronPIpistron:~/MAI\_OS/1\_Lab$ strace ./bin/run\_parent

execve("./bin/run\_parent", ["./bin/run\_parent"], 0x7fff623cb7f0 /\* 55 vars \*/) = 0

brk(NULL) = 0x5612f1ced000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7fffc96e6340) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f55dcdd0000

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

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=65771, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 65771, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f55dcdbf000

close(3) = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libstdc++.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\0\0\0\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=2260296, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 2275520, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f55dca00000

mprotect(0x7f55dca9a000, 1576960, PROT\_NONE) = 0

mmap(0x7f55dca9a000, 1118208, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x9a000) = 0x7f55dca9a000

mmap(0x7f55dcbab000, 454656, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ab000) = 0x7f55dcbab000

mmap(0x7f55dcc1b000, 57344, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x21a000) = 0x7f55dcc1b000

mmap(0x7f55dcc29000, 10432, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f55dcc29000

close(3) = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libgcc\_s.so.1", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=125488, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 127720, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f55dcd9f000

mmap(0x7f55dcda2000, 94208, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7f55dcda2000

mmap(0x7f55dcdb9000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1a000) = 0x7f55dcdb9000

mmap(0x7f55dcdbd000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1d000) = 0x7f55dcdbd000

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\0P\237\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 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\244;\374\204(\337f#\315I\214\234\f\256\271\32"..., 68, 896) = 68

newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2216304, ...}, AT\_EMPTY\_PATH) = 0

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

mmap(NULL, 2260560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f55dc600000

mmap(0x7f55dc628000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7f55dc628000

mmap(0x7f55dc7bd000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7f55dc7bd000

mmap(0x7f55dc815000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x214000) = 0x7f55dc815000

mmap(0x7f55dc81b000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f55dc81b000

close(3) = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libm.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\0\0\0\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=940560, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 942344, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f55dccb8000

mmap(0x7f55dccc6000, 507904, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xe000) = 0x7f55dccc6000

mmap(0x7f55dcd42000, 372736, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x8a000) = 0x7f55dcd42000

mmap(0x7f55dcd9d000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xe4000) = 0x7f55dcd9d000

close(3) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f55dccb6000

arch\_prctl(ARCH\_SET\_FS, 0x7f55dccb73c0) = 0

set\_tid\_address(0x7f55dccb7690) = 5535

set\_robust\_list(0x7f55dccb76a0, 24) = 0

rseq(0x7f55dccb7d60, 0x20, 0, 0x53053053) = 0

mprotect(0x7f55dc815000, 16384, PROT\_READ) = 0

mprotect(0x7f55dcd9d000, 4096, PROT\_READ) = 0

mprotect(0x7f55dcdbd000, 4096, PROT\_READ) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f55dccb4000

mprotect(0x7f55dcc1b000, 45056, PROT\_READ) = 0

mprotect(0x5612f0789000, 4096, PROT\_READ) = 0

mprotect(0x7f55dce0a000, 8192, PROT\_READ) = 0

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

munmap(0x7f55dcdbf000, 65771) = 0

getrandom("\x5c\x9d\x27\x34\x0c\x78\x68\x97", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0x5612f1ced000

brk(0x5612f1d0e000) = 0x5612f1d0e000

futex(0x7f55dcc2977c, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

newfstatat(1, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}, AT\_EMPTY\_PATH) = 0

getpid() = 5535

write(1, "Parent process with pid 5535 sta"..., 37Parent process with pid 5535 started

) = 37

write(1, "Input file name: ", 17Input file name: ) = 17

newfstatat(0, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}, AT\_EMPTY\_PATH) = 0

read(0, strace\_test

"strace\_test\n", 1024) = 12

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

write(1, "34\n", 334

) = 3

pipe2([5, 6], 0) = 0

write(1, "56\n", 356

) = 3

write(1, "34\n", 334

) = 3

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f55dccb7690) = 5537

close(6) = 0

close(3) = 0

getpid() = 5535

write(1, "Parent process with pid 5535\n", 29Parent process with pid 5535

) = 29

read(0, Child process with pid 5537 started

Input numbers:

7 7 7

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

newfstatat(4, "", {st\_mode=S\_IFIFO|0600, st\_size=0, ...}, AT\_EMPTY\_PATH) = 0

lseek(4, 0, SEEK\_CUR) = -1 ESPIPE (Illegal seek)

write(4, "7 ", 2) = 2

newfstatat(4, "", {st\_mode=S\_IFIFO|0600, st\_size=0, ...}, AT\_EMPTY\_PATH) = 0

lseek(4, 0, SEEK\_CUR) = -1 ESPIPE (Illegal seek)

write(4, "7 ", 2) = 2

newfstatat(4, "", {st\_mode=S\_IFIFO|0600, st\_size=0, ...}, AT\_EMPTY\_PATH) = 0

lseek(4, 0, SEEK\_CUR) = -1 ESPIPE (Illegal seek)

write(4, "7 ", 2) = 2

read(0, "", 1024) = 0

close(4) = 0

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=5537, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

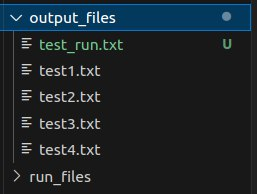
close(5) = 0

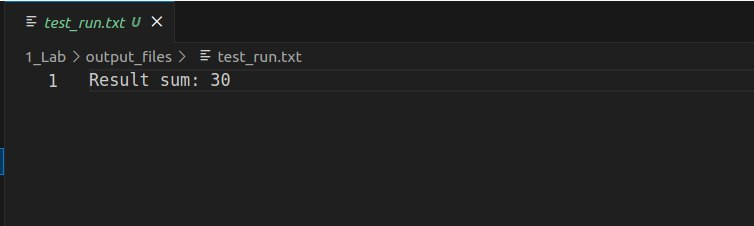
exit\_group(0) = ?

+++ exited with 0 +++

## Демонстрация работы программы







## Вывод

Во время выполнения лабораторной работы возникло много трудностей, связанных с прокидыванием пайпов и перенаправлением потоков ввода-вывода. Много времени ушло на правильную обработку общения между процессами и фикс багов. В целом лаба интересная, но лично мне не зашла. Много слишком низкоуровневых вещей, применение которым пока что не могу представить.