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

Постановка задачи

Составить и отладить программу на языке C/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 clos-
ing 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\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\3\0>\0\1\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\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"..., 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\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

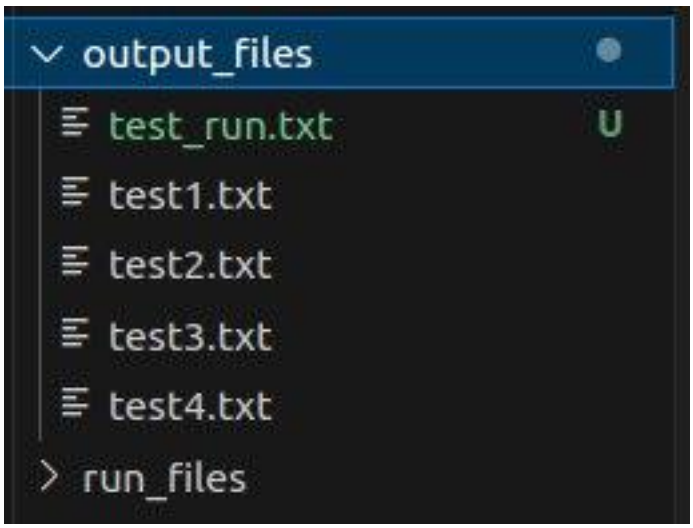
```
mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f55dccb4000

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
```

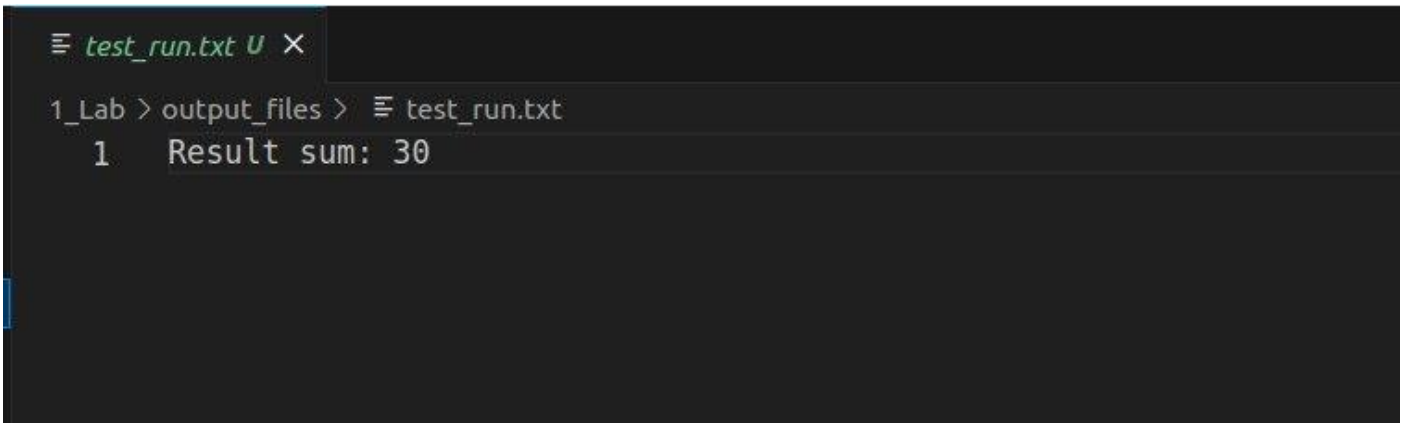
```
write(1, "34\n", 334

)                    = 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

```
baronpipistron@BaronPIpistron:~/MAI_OS/1_Lab$ make run
Running
bin/run_parent
Parent process with pid 3758 started
Input file name: test_run
Parent process with pid 3758
Child process with pid 3899 started
Input numbers:
8 9 10 3
baronpipistron@BaronPIpistron:~/MAI_OS/1_Lab$
```





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.