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»

Группа: М8О-210Б-23

Студент: Сетраков Ф.С.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 09.10.24

Москва, 2024

Постановка задачи

Вариант 22.

Необходимо составить и отладить программу на Си. Родительский процесс создаёт два дочерних процесса. Первой строкой пользователь вводит имя файла, которое будет использовано для открытия File с таким именем на запись для child1. Аналогично для второй строки и процесса child2. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их с вероятностью 80% в child1, иначе в child2. Дочерние процессы инвертируют строки.

Общий метод и алгоритм решения

Использованные системные вызовы:

- pid_t fork(void); – создает дочерний процесс.
- int pipe(int pipefd[2]) - создание неименованного канала для передачи данных между процессами
- pid_t waitpid(pid_t pid, int *status, int options) - Ожидание завершения дочернего процесса
- void exit(int status) - завершения выполнения процесса и возвращение статуса
- int open(const char *pathname, int flags, mode_t mode) - открытие\создание файла
- int close(int fd) - закрыть файл

Для работы со стандартным вводом/выводом без использования iostream, я написал несколько функций, которые пишут/считывают строки из стандартного потока, при помощи системных вызовов read и write. Моя программа считывает имена двух файлов, валидирует их. Далее создаются 2 пайпа - один для транспортировки данных между parent и child1, второй для child2. При помощи вызова fork, создаются 2 дочерних процесса, каждый из которых считывает строки из своего пайпа и пишет инвертированную строку в файл, пока не встретит строку STOP. Строка STOP является сигналом окончания ввода. Родительский процесс считывает строки из стандартного ввода, пока не встретит строку STOP, после чего, генерируется случайное число от 0 до 99 - если число больше 79 - то строка перенаправляется через пайп в процесс child2, иначе в child1.

Код программы

main.cpp

```
#include "multitasking/parent.h"

#include "errlib/errors.h"

#include "errlib/error_handlers.h"


int main() {

    error err = parent_process();

    if (err != STATE_OK) {
```

```
        log_error(err);
        }
        return err;
}
```

parent.cpp

```
#include "parent.h"

error parent_loop(int write_pipe_fd1, int write_pipe_fd2) {
        std::string buffer;
        int bytes = std_in(&buffer);
        int target_fd;

        while (bytes > 0 && buffer != "STOP\n") {
        target_fd = (std::rand() % 100 >= 80) ? write_pipe_fd2 : write_pipe_fd1;
        if (file_print(target_fd, buffer) == -1) {
        return ERROR_PIPE_FAILED;
        }
        bytes = std_in(&buffer);
        }
        if (file_print(write_pipe_fd1, "STOP\n") == -1 || file_print(write_pipe_fd2, "STOP\n") ==
-1) {
        return ERROR_STOP_FAILED;
        }
        return STATE_OK;
}

error parent_process() {
        std::srand(std::time(nullptr));

        std::string file1_name, file2_name;
```

```
int bytes = file_scan(STDIN_FILENO, &file1_name);
if (bytes <= 1) {
return ERROR_INVALID_INPUT;
}
file1_name.erase(file1_name.length() - 1);

bytes = file_scan(STDIN_FILENO, &file2_name);
if (bytes <= 1) {
return ERROR_INVALID_INPUT;
}
file2_name.erase(file2_name.length() - 1);

if (file1_name == file2_name) {
return ERROR_INVALID_INPUT;
}

int pipe1_fd[2], pipe2_fd[2];
if (pipe(pipe1_fd) == -1 || pipe(pipe2_fd) == -1) {
perror("Error creating pipes");
return ERROR_PIPE_FAILED;
}

pid_t pid1 = fork();
if (pid1 == -1) {
perror("Error forking first child");
return ERROR_FORK_FAILED;
}
if (pid1 == 0) {
close(pipe1_fd[1]);
```

```
        child_process(pipe1_fd[0], file1_name);
        }

        pid_t pid2 = fork();
        if (pid2 == -1) {
        perror("Error forking second child");
        return ERROR_FORK_FAILED;
        }
        if (pid2 == 0) {
        close(pipe2_fd[1]);
        child_process(pipe2_fd[0], file2_name);
        }

        close(pipe1_fd[0]);
        close(pipe2_fd[0]);

        error err = parent_loop(pipe1_fd[1], pipe2_fd[1]);

        waitpid(pid1, nullptr, 0);
        waitpid(pid2, nullptr, 0);

        close(pipe1_fd[1]);
        close(pipe2_fd[1]);
        return err;
}
```

child.cpp

```
#include "child.h"
#include "../iolib/io.h"

void invert_string(std::string* s) {
```

```
	int p1 = 0, p2 = (*s).size() - 1;
	while (p1 < p2) {
	std::swap((*s)[p1], (*s)[p2]);
	p1++;
	p2--;
	}
}

void child_process(int read_pipe_fd, const std::string& file_name) {
	int write_fd = open(file_name.c_str(), O_WRONLY | O_CREAT | O_APPEND, S_IRUSR | S_IWUSR);
	if (write_fd == -1) {
	log_stderr("Error opening file in child process");
	exit(1);
	}

	std::string buffer;
	int bytes = file_scan(read_pipe_fd, &buffer);
	while (bytes > 0 && buffer != "STOP\n") {
	buffer.erase(buffer.size() - 1);
	invert_string(&buffer);
	if (file_print(write_fd, buffer + "\n") == -1) {
	log_stderr("Error writing to file");
	exit(1);
	}
	bytes = file_scan(read_pipe_fd, &buffer);
	}

	close(read_pipe_fd);
	close(write_fd);
```

```
        exit(0);
}
```

io.cpp

```
#include "io.h"

int std_in(std::string* message) {
        return file_scan(STDIN_FILENO, message);
}

void std_out(const std::string& message) {
        file_print(STDOUT_FILENO, message);
}

void log_stderr(const std::string& message) {
        file_print(STDERR_FILENO, message);
}

int file_scan(int input_file, std::string* message) {
        char buffer[DEFAULT_BUFF_SIZE];
        std::string output;
        ssize_t bytes = read(input_file, buffer, DEFAULT_BUFF_SIZE);
        for (int i = 0; i < bytes; i++) {
        output += buffer[i];
        }
        *message = output;
        return (int) bytes;
}

int file_print(int output_file, const std::string& message) {
        ssize_t bytes = write(output_file, message.c_str(), message.size());
```

```
        return (int) bytes;
}
```

error_handlers.cpp

```
#include "error_handlers.h"


void log_error(error err) {
        switch (err) {
        case ERROR_STOP_FAILED: {
        std_out("ERROR: CANT STOP ONE OF CHILD PROCESSES\n");
        break;
        }
        case ERROR_PIPE_FAILED: {
        std_out("ERROR: CANT CREATE PIPES\n");
        break;
        }
        case ERROR_FORK_FAILED: {
        std_out("ERROR: FORK FAILED\n");
        break;
        }
        case ERROR_INVALID_INPUT: {
        std_out("ERROR: INVALID INPUT\n");
        break;
        }
        case ERROR_CANT_WRITE_FILE: {
        std_out("ERROR: CANT WRITE FILE\n");
        break;
        }
        default: {
        std_out("UNKNOWN ERROR CODE\n");
        break;
```

```
        }
        }
}
```

errors.h

```
#ifndef LAB1_ERRORS_H
#define LAB1_ERRORS_H

typedef enum {
        STATE_OK,
        ERROR_CANT_WRITE_FILE,
        ERROR_PIPE_FAILED,
        ERROR_FORK_FAILED,
        ERROR_INVALID_INPUT,
        ERROR_STOP_FAILED,
} error;
#endif
```

io.h

```
#ifndef LAB1_IO_H
#define LAB1_IO_H
#include <fcntl.h>
#include <unistd.h>
#include <string>

#define DEFAULT_BUFF_SIZE 1024

void log_stderr(const std::string&);
int std_in(std::string*);
void std_out(const std::string&);
int file_scan(int input_file, std::string *message);
int file_print(int output_file, const std::string &message);
#endif
```

parent.h

```
#ifndef LAB1_PARENT_H

#define LAB1_PARENT_H


#include "../iolib/io.h"

#include "../errlib/errors.h"

#include "child.h"


#include <ctime>

#include <sys/wait.h>

#include <cstdlib>


error parent_process();


#endif
```

child.h

```
#ifndef LAB1_CHILD_H

#define LAB1_CHILD_H


#include <string>


#include "../iolib/io.h"

void child_process(int, const std::string&);


#endif
```

error_handlers.h

```
#ifndef LAB1_ERROR_HANDLERS_H

#define LAB1_ERROR_HANDLERS_H


#include "errors.h"

#include "../iolib/io.h"


void log_error(error);

#endif
```

Протокол работы программы

Тестирование

```
$ ./a.out
file1.txt
file2.txt
string 1
string 2
dovod
line 4
STOP
$ cat file1.txt
1 gnirts
2 gnirts
4 enil
$ cat file2.txt
dovod
```

Strace

```
$ strace -f ./a.out
execve("./a.out", ["./a.out"], 0x7ffd0ed6f768 /* 58 vars */) = 0
brk(NULL)                               = 0x5a44218c8000
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=151211, ...}) = 0
mmap(NULL, 151211, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fcc4f857000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/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
fstat(3, {st_mode=S_IFREG|0755, st_size=22040176, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fcc4f855000
mmap(NULL, 2641984, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fcc4f400000
mmap(0x7fcc4f497000, 1363968, PROT_READ|PROT_EXEC,
```

```
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x97000) = 0x7fcc4f497000
mmap(0x7fcc4f5e4000, 589824, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1e4000) = 0x7fcc4f5e4000
mmap(0x7fcc4f674000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x274000) = 0x7fcc4f674000
mmap(0x7fcc4f682000, 12352, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7fcc4f682000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/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"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=973144, ...}) = 0
mmap(NULL, 975176, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fcc4f766000
mmap(0x7fcc4f774000, 536576, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xe000) = 0x7fcc4f774000
mmap(0x7fcc4f7f7000, 376832, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x91000) = 0x7fcc4f7f7000
mmap(0x7fcc4f853000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xec000) = 0x7fcc4f853000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/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
fstat(3, {st_mode=S_IFREG|0644, st_size=915712, ...}) = 0
mmap(NULL, 184808, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fcc4f738000
mmap(0x7fcc4f73c000, 147456, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x4000) = 0x7fcc4f73c000
mmap(0x7fcc4f760000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x28000) = 0x7fcc4f760000
mmap(0x7fcc4f764000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x2b000) = 0x7fcc4f764000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/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\340_\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\0"..., 784,
64) = 784
fstat(3, {st_mode=S_IFREG|0755, st_size=2014520, ...}) = 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\0"..., 784,
64) = 784
mmap(NULL, 2034616, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fcc4f20f000
mmap(0x7fcc4f233000, 1511424, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x24000) = 0x7fcc4f233000
mmap(0x7fcc4f3a4000, 319488, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x195000) = 0x7fcc4f3a4000
mmap(0x7fcc4f3f2000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1e3000) = 0x7fcc4f3f2000
mmap(0x7fcc4f3f8000, 31672, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7fcc4f3f8000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
```

```
0x7fcc4f736000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fcc4f734000
arch_prctl(ARCH_SET_FS, 0x7fcc4f7375c0) = 0
set_tid_address(0x7fcc4f737890)        = 44739
set_robust_list(0x7fcc4f7378a0, 24)    = 0
rseq(0x7fcc4f737ee0, 0x20, 0, 0x53053053) = 0
mprotect(0x7fcc4f3f2000, 16384, PROT_READ) = 0
mprotect(0x7fcc4f764000, 4096, PROT_READ) = 0
mprotect(0x7fcc4f853000, 4096, PROT_READ) = 0
mprotect(0x7fcc4f674000, 53248, PROT_READ) = 0
mprotect(0x5a44130ea000, 4096, PROT_READ) = 0
mprotect(0x7fcc4f8b6000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7fcc4f857000, 151211)         = 0
futex(0x7fcc4f6826bc, FUTEX_WAKE_PRIVATE, 2147483647) = 0
getrandom("\xeb\x45\x67\xde\xec\xa9\x0f\xbd", 8, GRND_NONBLOCK) = 8
brk(NULL)                              = 0x5a44218c8000
brk(0x5a44218e9000)                    = 0x5a44218e9000
read(0, file1.txt
"file1.txt\n", 1024)            = 10
read(0, file2.txt
"file2.txt\n", 1024)            = 10
pipe2([3, 4], 0)                       = 0
pipe2([5, 6], 0)                       = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7fcc4f737890) = 44742
strace: Process 44742 attached
[pid 44739]
clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD <unfinished ...>
[pid 44742]
set_robust_list(0x7fcc4f7378a0, 24) = 0
strace: Process 44743 attached
[pid 44739] <... clone resumed>, child_tidptr=0x7fcc4f737890) = 44743
[pid 44743] set_robust_list(0x7fcc4f7378a0, 24 <unfinished ...>
[pid 44742] close(4 <unfinished ...>
[pid 44739] close(3 <unfinished ...>
[pid 44743] <... set_robust_list resumed>) = 0
[pid 44739] <... close resumed>)       = 0
[pid 44742] <... close resumed>)       = 0
[pid 44739] close(5)                   = 0
[pid 44742] openat(AT_FDCWD, "file1.txt", O_WRONLY|O_CREAT|O_APPEND, 0600 <unfinished
...>
[pid 44739] read(0,  <unfinished ...>
[pid 44743] close(6 <unfinished ...>
[pid 44742] <... openat resumed>)      = 4
[pid 44743] <... close resumed>)       = 0
[pid 44742] read(3,  <unfinished ...>
```

```
[pid 44743] openat(AT_FDCWD, "file2.txt", O_WRONLY|O_CREAT|O_APPEND, 0600) = 6
[pid 44743] read(5, string 1
 <unfinished ...>
[pid 44739] <... read resumed>"string 1\n", 1024) = 9
[pid 44739] write(4, "string 1\n", 9)   = 9
[pid 44739] read(0,  <unfinished ...>
[pid 44742] <... read resumed>"string 1\n", 1024) = 9
[pid 44742] write(4, "1 gnirts\n", 9)   = 9
[pid 44742] read(3, string 2
 <unfinished ...>
[pid 44739] <... read resumed>"string 2\n", 1024) = 9
[pid 44739] write(4, "string 2\n", 9)   = 9
[pid 44742] <... read resumed>"string 2\n", 1024) = 9
[pid 44739] read(0,  <unfinished ...>
[pid 44742] write(4, "2 gnirts\n", 9)   = 9
[pid 44742] read(3, dovod
 <unfinished ...>
[pid 44739] <... read resumed>"dovod\n", 1024) = 6
[pid 44739] write(4, "dovod\n", 6)     = 6
[pid 44742] <... read resumed>"dovod\n", 1024) = 6
[pid 44739] read(0,  <unfinished ...>
[pid 44742] write(4, "dovod\n", 6)     = 6
[pid 44742] read(3, line 4
 <unfinished ...>
[pid 44739] <... read resumed>"line 4\n", 1024) = 7
[pid 44739] write(4, "line 4\n", 7)    = 7
[pid 44742] <... read resumed>"line 4\n", 1024) = 7
[pid 44739] read(0,  <unfinished ...>
[pid 44742] write(4, "4 enil\n", 7)    = 7
[pid 44742] read(3, STOP
 <unfinished ...>
[pid 44739] <... read resumed>"STOP\n", 1024) = 5
[pid 44739] write(4, "STOP\n", 5)      = 5
[pid 44739] write(6, "STOP\n", 5)      = 5
[pid 44743] <... read resumed>"STOP\n", 1024) = 5
[pid 44742] <... read resumed>"STOP\n", 1024) = 5
[pid 44739] wait4(44742,  <unfinished ...>
[pid 44743] close(5 <unfinished ...>
[pid 44742] close(3 <unfinished ...>
[pid 44743] <... close resumed>)       = 0
[pid 44742] <... close resumed>)       = 0
[pid 44743] close(6 <unfinished ...>
[pid 44742] close(4 <unfinished ...>
[pid 44743] <... close resumed>)       = 0
[pid 44742] <... close resumed>)       = 0
[pid 44743] exit_group(0 <unfinished ...>
[pid 44742] exit_group(0 <unfinished ...>
[pid 44743] <... exit_group resumed>)   = ?
[pid 44742] <... exit_group resumed>)   = ?
```

```
[pid 44743] +++ exited with 0 +++
[pid 44742] +++ exited with 0 +++
<... wait4 resumed>NULL, 0, NULL)      = 44742
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=44743, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} ---
wait4(44743, NULL, 0, NULL)            = 44743
close(4)                               = 0
close(6)                               = 0
exit_group(0)                          = ?
+++ exited with 0 +++
```

Вывод

Я изучил, как управлять процессами в ОС при помощи системных вызовов на языке Си. Изучил способы обмена данных между процессами, способы синхронизации процессов. Данная лабораторная работа показалась мне интересной, т.к ранее я не работал с процессами напрямую. В ходе выполнения данной лабораторной работы я столкнулся с многими проблемами, например, отладкой такого кода. Отлаживать такой код довольно тяжело, так ошибка может находится в одном из нескольких процессов, и отследить ее довольно трудно