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-213Б-23

Студент: Шилов П.В.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 21.11.24

Москва, 2024

Постановка задачи

Вариант 16.

Родительский процесс создает дочерний процесс. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись. Перенаправление стандартных потоков ввода-вывода показано на картинке выше. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1. Процесс child проверяет строки на валидность правилу. Если строка соответствует правилу, то она выводится в стандартный поток вывода дочернего процесса, иначе в pipe2 выводится информация об ошибке. Родительский процесс полученные от child ошибки выводит в стандартный поток вывода. Правило проверки: строка должна оканчиваться на «.» или «;»

Общий метод и алгоритм решения

Использованные системные вызовы:

- pid_t fork() - создание дочернего процесса
- int execlp(const char *file, const char* arg) — дублирование действия оболочки, относящиеся к поиску исполняемого файла
- void exit(int status) - завершения выполнения процесса и возвращение статуса
- int pipe(int pipefd[2]) - создание неименованного канала для передачи данных между процессами
- int open(const char *pathname, int flags, mode_t mode) - открытие\создание файла
- int close(int fd) - закрыть файл

Программа считывает из стандартного ввода имя файла, и создаёт два канала с помощью функции pipe. Далее порождается дочерний процесс, которому подается имя файла через pipe1, и он открывает его с помощью вызова open.

Родительский процесс считывает строки из консоли, пока дочерний процесс не завершит свое действие, и передает их дочернему процессу.

Дочерний процесс считывает данные из pipe1 и проверяет их на соответствие правилу. Данные об ошибках дочерний процесс передает родительскому процессу через pipe2, после чего родительский процесс выводит их в стандартный поток. Строки, прошедшие проверку, записываются в стандартный поток дочернего процесса (файл). При вводе в консоль команды exit действие дочернего процесса завершается, после чего также завершается работа родительского процесса.

Код программы

parent.c

```
#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <string.h>
#include <errno.h>

#define BUFFER_SIZE 1024
```

```
void print_error(const char *msg) {
write(STDERR_FILENO, msg, strlen(msg));
}

int main() {
int pipe1[2] = {3, 4};
int pipe2[2] = {5, 6};
pid_t pid;
char filename[BUFFER_SIZE];
char input[BUFFER_SIZE];
char error_msg[BUFFER_SIZE];

const char start_msg[] = "Введите имя файла: ";
write(STDOUT_FILENO, start_msg, sizeof(start_msg));
input[strcspn(filename, "\n")] = 0;
int filename_len = read(STDIN_FILENO, filename, sizeof(filename));
if (filename_len <= 1) {
print_error("Ошибка ввода имени файла\n");
exit(EXIT_FAILURE);
}
if (pipe(pipe1) == -1 || pipe(pipe2) == -1) {
print_error("Ошибка при создании pipe\n");
exit(EXIT_FAILURE);
}

pid = fork();
if (pid < 0) {
print_error("Ошибка при создании процесса\n");
exit(EXIT_FAILURE);
}
if (pid == 0) {
close(pipe1[1]);
close(pipe2[0]);
execlp("./child", "./child", filename, (char *) NULL);
print_error("Ошибка при запуске дочернего процесса\n");
exit(EXIT_FAILURE);
}

close(pipe1[0]);
close(pipe2[1]);

const char msg1[] = "Введите строку (или 'exit' для выхода):\n";
write(STDOUT_FILENO, msg1, sizeof(msg1));

while (1) {
int input_len = read(STDIN_FILENO, input, sizeof(input));
input[strcspn(input, "\n")] = 0;
if (strcmp(input, "exit") == 0) {
break;
}
if (input_len <= 1) {
print_error("Ошибка ввода строки\n");
break;
}
```

```
if (write(pipe1[1], input, input_len) == -1) {
print_error("Ошибка записи в pipe1\n");
break;
}

int error_len = read(pipe2[0], error_msg, BUFFER_SIZE);
error_msg[error_len] = '\0';
if (error_len > 0) {
if (strcmp(error_msg, "SUCCESS") != 0) {
write(STDERR_FILENO, error_msg, error_len);
}
}
}
close(pipe1[1]);
close(pipe2[0]);
wait(NULL);
return 0;
}
```

child.c

```
#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>

#define BUFFER_SIZE 1024

void print_error(int pipe_fd, const char *msg) {
write(pipe_fd, msg, strlen(msg));
}

int ends_with_dot_or_semicolon(const char *str) {
int len = strlen(str);
return (len > 0 && (str[len - 1] == '.' || str[len - 1] == ';'));
}

int main(int argc, char* argv[]) {
int pipe1[2] = {3, 4};
int pipe2[2] = {5, 6};
char buffer[BUFFER_SIZE];
char* error_msg;

int fd = open(argv[1], O_WRONLY | O_CREAT | O_TRUNC, 0644);
if (fd < 0) {
error_msg = "Ошибка открытия файла\n";
print_error(pipe2[1], error_msg);
close(fd);
close(pipe1[0]);
close(pipe2[1]);
return -1;
}

while (1) {
int len = read(pipe1[0], buffer, BUFFER_SIZE);
if (len <= 0) break;
buffer[len - 1] = '\0';
```

```
if (ends_with_dot_or_semicolon(buffer)) {
write(fd, buffer, strlen(buffer));
write(fd, "\n", 1);
print_error(pipe2[1], "SUCCESS");
} else {
error_msg = "Строка не соответствует правилу!\n";
print_error(pipe2[1], error_msg);
}
}

close(fd);
close(pipe1[0]);
close(pipe2[1]);
return 0;
}
```

Протокол работы программы

Тестирование:

```
$ ./a.out
output.txt
joker
Queen;
.
;
Umbasa
portal;;
exit
```

Вывод(output.txt):	Вывод(stderr):
Queen;	Строка не соответствует правилу!
.	Строка не соответствует правилу!
;	
portal;;	

Strace:

```
execve("./a.out", ["./a.out"], 0x7ffe4d1b2428 /* 71 vars */) = 0

brk(NULL)                               = 0x55eb185b3000

arch_prctl(0x3001 /* ARCH_??? */, 0x7ffda6fd5ae0) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7cad81621000

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=58575, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 58575, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7cad81612000

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\302\211\332Pq\2439\235\350\223\322\257\201\326\243\f"..., 68, 896) =
68

newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, 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, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7cad81200000

mprotect(0x7cad81228000, 2023424, PROT_NONE) = 0

mmap(0x7cad81228000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) =
0x7cad81228000

mmap(0x7cad813bd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) =
0x7cad813bd000

mmap(0x7cad81416000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) =
0x7cad81416000

mmap(0x7cad8141c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x7cad8141c000

close(3)                                = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7cad8160f000

arch_prctl(ARCH_SET_FS, 0x7cad8160f740) = 0

set_tid_address(0x7cad8160fa10)         = 54514

set_robust_list(0x7cad8160fa20, 24)     = 0

rseq(0x7cad816100e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7cad81416000, 16384, PROT_READ) = 0

mprotect(0x55eb1833c000, 4096, PROT_READ) = 0
```

```
mprotect(0x7cad8165b000, 8192, PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

munmap(0x7cad81612000, 58575)           = 0

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\270\320\274\321\217
\321\204\320\260\320\271\320\273\320\260"..., 35Введите имя файла: ) = 35

read(0, output.txt

"output.txt\n", 1024)           = 11

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

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

clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace: Process 54535 attached

, child_tidptr=0x7cad8160fa10) = 54535

[pid 54514] close(3 <unfinished ...>

[pid 54535] set_robust_list(0x7cad8160fa20, 24 <unfinished ...>

[pid 54514] <... close resumed>)        = 0

[pid 54535] <... set_robust_list resumed>) = 0

[pid 54514] close(6)                    = 0

[pid 54535] close(4 <unfinished ...>

[pid 54514] write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\321\201\321\202\321\200\320\276\320\272\321\203 (\320\270\320"..., 66 <unfinished ...>

[pid 54535] <... close resumed>)        = 0

[pid 54535] close(5Введите строку (или 'exit' для выхода):

 <unfinished ...>

[pid 54514] <... write resumed>)        = 66

[pid 54535] <... close resumed>)        = 0

[pid 54514] read(0,  <unfinished ...>

[pid 54535] execve("./child", ["./child", "output.txt\n"], 0x7ffda6fd5cb8 /* 71 vars */) = 0

[pid 54535] brk(NULL)                   = 0x61c65c000000

[pid 54535] arch_prctl(0x3001 /* ARCH_??? */, 0x7fff77cb01d0) = -1 EINVAL (Invalid argument)

[pid 54535] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75493c80f000

[pid 54535] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)

[pid 54535] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 4

[pid 54535] newfstatat(4, "", {st_mode=S_IFREG|0644, st_size=58575, ...}, AT_EMPTY_PATH) = 0

[pid 54535] mmap(NULL, 58575, PROT_READ, MAP_PRIVATE, 4, 0) = 0x75493c800000

[pid 54535] close(4)                    = 0

[pid 54535] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 4

[pid 54535] read(4, "\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

[pid 54535] pread64(4, "\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
```

[pid 54535] pread64(4, "\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

[pid 54535] pread64(4, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\302\211\332Pq\2439\235\350\223\322\257\201\326\243\f"..., 68, 896) = 68

[pid 54535] newfstatat(4, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0

[pid 54535] pread64(4, "\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

[pid 54535] mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 4, 0) = 0x75493c400000

[pid 54535] mprotect(0x75493c428000, 2023424, PROT_NONE) = 0

[pid 54535] mmap(0x75493c428000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x28000) = 0x75493c428000

[pid 54535] mmap(0x75493c5bd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x1bd000) = 0x75493c5bd000

[pid 54535] mmap(0x75493c616000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x215000) = 0x75493c616000

[pid 54535] mmap(0x75493c61c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x75493c61c000

[pid 54535] close(4)   = 0

[pid 54535] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75493c7fd000

[pid 54535] arch_prctl(ARCH_SET_FS, 0x75493c7fd740) = 0

[pid 54535] set_tid_address(0x75493c7fda10) = 54535

[pid 54535] set_robust_list(0x75493c7fda20, 24) = 0

[pid 54535] rseq(0x75493c7fe0e0, 0x20, 0, 0x53053053) = 0

[pid 54535] mprotect(0x75493c616000, 16384, PROT_READ) = 0

[pid 54535] mprotect(0x61c65aff5000, 4096, PROT_READ) = 0

[pid 54535] mprotect(0x75493c849000, 8192, PROT_READ) = 0

[pid 54535] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

[pid 54535] munmap(0x75493c800000, 58575) = 0

[pid 54535] openat(AT_FDCWD, "output.txt\n", O_WRONLY|O_CREAT|O_TRUNC, 0644) = 4

[pid 54535] read(3, joker

 <unfinished ...>

[pid 54514] <... read resumed>"joker\n", 1024) = 6

[pid 54514] write(4, "joker\0", 6)   = 6

[pid 54535] <... read resumed>"joker\0", 1024) = 6

[pid 54514] read(5, <unfinished ...>

[pid 54535] write(6, "\320\241\321\202\321\200\320\276\320\272\320\260 \320\275\320\265 \321\201\320\276\320\276\321\202\320\262\320\265\321\202"..., 61) = 61

[pid 54514] <... read resumed>"\320\241\321\202\321\200\320\276\320\272\320\260 \320\275\320\265 \321\201\320\276\320\276\321\202\320\262\320\265\321\202"..., 1024) = 61

[pid 54535] read(3, <unfinished ...>

[pid 54514] write(2, "\320\241\321\202\321\200\320\276\320\272\320\260 \320\275\320\265

```
\321\201\320\276\320\276\321\202\320\262\320\265\321\202"..., 61Строка не соответствует правилу!
) = 61
[pid 54514] read(0, Queen;
"Queen;\n", 1024)   = 7
[pid 54514] write(4, "Queen;\0", 7)     = 7
[pid 54535] <... read resumed>"Queen;\0", 1024) = 7
[pid 54514] read(5,  <unfinished ...>
[pid 54535] write(4, "Queen;", 6)      = 6
[pid 54535] write(4, "\n", 1)          = 1
[pid 54535] write(6, "SUCCESS", 7 <unfinished ...>
[pid 54514] <... read resumed>"SUCCESS", 1024) = 7
[pid 54535] <... write resumed>)       = 7
[pid 54514] read(0,  <unfinished ...>
[pid 54535] read(3, .
 <unfinished ...>
[pid 54514] <... read resumed>".\n", 1024) = 2
[pid 54514] write(4, ".\0", 2)         = 2
[pid 54535] <... read resumed>".\0", 1024) = 2
[pid 54514] read(5,  <unfinished ...>
[pid 54535] write(4, ".", 1)           = 1
[pid 54535] write(4, "\n", 1)          = 1
[pid 54535] write(6, "SUCCESS", 7 <unfinished ...>
[pid 54514] <... read resumed>"SUCCESS", 1024) = 7
[pid 54535] <... write resumed>)       = 7
[pid 54514] read(0,  <unfinished ...>
[pid 54535] read(3, ;
 <unfinished ...>
[pid 54514] <... read resumed>";\n", 1024) = 2
[pid 54514] write(4, ";\0", 2)         = 2
[pid 54535] <... read resumed>";\0", 1024) = 2
[pid 54514] read(5,  <unfinished ...>
[pid 54535] write(4, ";", 1)           = 1
[pid 54535] write(4, "\n", 1)          = 1
[pid 54535] write(6, "SUCCESS", 7 <unfinished ...>
[pid 54514] <... read resumed>"SUCCESS", 1024) = 7
[pid 54535] <... write resumed>)       = 7
```

[pid 54514] read(0, <unfinished ...>

[pid 54535] read(3, Umbasa

 <unfinished ...>

[pid 54514] <... read resumed>"Umbasa\n", 1024) = 7

[pid 54514] write(4, "Umbasa\0", 7)   = 7

[pid 54535] <... read resumed>"Umbasa\0", 1024) = 7

[pid 54514] read(5, <unfinished ...>

[pid 54535] write(6, "\320\241\321\202\321\200\320\276\320\272\320\260 \320\275\320\265
\321\201\320\276\320\276\321\202\320\262\320\265\321\202"..., 61) = 61

[pid 54514] <... read resumed>"\320\241\321\202\321\200\320\276\320\272\320\260 \320\275\320\265
\321\201\320\276\320\276\321\202\320\262\320\265\321\202"..., 1024) = 61

[pid 54535] read(3, <unfinished ...>

[pid 54514] write(2, "\320\241\321\202\321\200\320\276\320\272\320\260 \320\275\320\265
\321\201\320\276\320\276\321\202\320\262\320\265\321\202"..., 61Строка не соответствует правилу!

) = 61

[pid 54514] read(0, portal;;

"portal;;\n", 1024) = 9

[pid 54514] write(4, "portal;;\0", 9)   = 9

[pid 54535] <... read resumed>"portal;;\0", 1024) = 9

[pid 54514] read(5, <unfinished ...>

[pid 54535] write(4, "portal;;", 8)   = 8

[pid 54535] write(4, "\n", 1)   = 1

[pid 54535] write(6, "SUCCESS", 7)   = 7

[pid 54514] <... read resumed>"SUCCESS", 1024) = 7

[pid 54535] read(3, <unfinished ...>

[pid 54514] read(0, exit

"exit\n", 1024)   = 5

[pid 54514] close(4)   = 0

[pid 54535] <... read resumed>"", 1024) = 0

[pid 54514] close(5 <unfinished ...>

[pid 54535] close(4 <unfinished ...>

[pid 54514] <... close resumed>)   = 0

[pid 54535] <... close resumed>)   = 0

[pid 54514] wait4(-1, <unfinished ...>

[pid 54535] close(3)   = 0

[pid 54535] close(6)   = 0

[pid 54535] exit_group(0)   = ?

```
[pid 54535] +++ exited with 0 +++
<... wait4 resumed>NULL, 0, NULL)      = 54535
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=54535, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---
exit_group(0)                          = ?
+++ exited with 0 +++
```

Вывод

Работа над этой программой потребовала глубокого понимания взаимодействия между родительским и дочерним процессами в C. Необходимо было тщательно спроектировать механизм перенаправления ввода-вывода с помощью трубок, а также реализовать функции для чтения и обработки строк. Особое внимание
пришлось уделить обработке возможных ошибок на каждом этапе выполнения программы, чтобы обеспечить надежную и отказоустойчивую работу. В результате была создана программа, демонстрирующая хорошее понимание концепций межпроцессного взаимодействия и обработки данных в системном программировании на языке C.