Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Лабораторная работа №3 по курсу

«Операционные системы»

Группа: М8О-213Б-23

Студент: Шилов П.В.

Преподаватель: Бахарев В.Д

Оценка: _____

Дата: 22.11.24

Москва, 2024

Постановка задачи

Вариант 6.

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 дочернему процессу. Процесс child проверяет строки на валидность правилу. Если строка соответствует правилу, то она выводится в стандартный поток вывода дочернего процесса, иначе родительскому процессу передается информация об ошибке. Родительский процесс полученные от child ошибки выводит в стандартный поток вывода. Правило проверки: строка должна оканчиваться на «.» или «;» Для связи между родительским и дочернимпроцессом должны быть использованы общая память и семафоры.

Общий метод и алгоритм решения

Использованные системные вызовы:

- pid_t fork(void); – создает дочерний процесс.
- int read(int fd, void* buf, size_t count); – считывает count байт из fd в buf.
- int write(int fd, void* buf, size_t count); – записывает count байт из buf в fd.
- int open(const char *pathname, int flags, ...); – открывает файловый дескриптор.
- int close(int fd); – закрывает файловый дескриптор.
- int shm_open(const char *name, int oflag, mode_t mode); – создает или открываетобъект общей памяти.
- int shm_unlink(const char *name); – удаляет объект общей памяти.
- void *mmap(void addr, size_t length, int prot, int flags, int fd, off_t offset); – выполняетотображение файла или устройства на память.
- int munmap(void addr, size_t length); – удаляет отображение файла или устройства напамять.
- sem_t *sem_open(const char *name, int oflag, mode_t mode, unsigned int value); – создает или открывает семафор. Если создается новый (oflag & O_CREAT > 0), тонеобходимо указать разрешения на доступ и начальное значение.
- int sem_wait(sem_t *sem); – ожидает пока значение, хранимое в семафоре, не станетбольше нуля, после чего уменьшает его на 1.
- int sem_post(sem_t *sem); – увеличивает значение, хранимое в семафоре на 1.
- int sem_unlink(const char *name); – удаляет семафор.

Программа считывает из стандартного ввода имя файла, которую родительский процесс записывает в общую память. Далее порождается дочерний процесс.

Родительский процесс считывает строки из консоли, и записывает их в общую память, о чем сигнализирует дочернему процессу.

Дочерний процесс, дождавшись сигнала, считывает имя файла и открывает его, а

потом начинает считывать строки из общей памяти и проверяет их на соответствие правилу. Данные об ошибках дочерний процесс передает родительскому процессу через другой участок общей памяти, отведенной для ошибок, и сигнализирует о завершении чтения родителю. После чего родительский процесс выводит ошибки в свой стандартный поток. Строки, прошедшие проверку, записываются в стандартный поток дочернего процесса (файл). При вводе в консоль команды exit действие обоих процессов завершается.

Код программы

parent.c

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

#define SHM_NAME "/shared_memory"
#define SHM_ERROR_NAME "/shared_error_memory"
#define SEM_PARENT_WRITE "/sem_parent_write"
#define SEM_CHILD_READ "/sem_child_read"
#define BUFFER_SIZE 1024

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

int main() {
pid_t pid;
char filename[BUFFER_SIZE];
char input[BUFFER_SIZE];
char *shared_memory;
char *error_memory;

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

int shm_fd = shm_open(SHM_NAME, O_CREAT | O_RDWR, 0666);
int shm_error_fd = shm_open(SHM_ERROR_NAME, O_CREAT | O_RDWR, 0666);
if (shm_fd == -1 || shm_error_fd == -1) {
print_error("Ошибка при создании shared memory\n");
exit(EXIT_FAILURE);
}
ftruncate(shm_fd, BUFFER_SIZE);
ftruncate(shm_error_fd, BUFFER_SIZE);
```

```
shared_memory = mmap(NULL, BUFFER_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, shm_fd, 0);
error_memory = mmap(NULL, BUFFER_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, shm_error_fd, 0);
if (shared_memory == MAP_FAILED || error_memory == MAP_FAILED) {
print_error("Ошибка при отображении shared memory\n");
exit(EXIT_FAILURE);
}

sem_t *sem_parent_write = sem_open(SEM_PARENT_WRITE, O_CREAT, 0666, 0);
sem_t *sem_child_read = sem_open(SEM_CHILD_READ, O_CREAT, 0666, 0);
if (sem_parent_write == SEM_FAILED || sem_child_read == SEM_FAILED) {
print_error("Ошибка при создании семафоров\n");
exit(EXIT_FAILURE);
}

pid = fork();
if (pid < 0) {
print_error("Ошибка при создании процесса\n");
exit(EXIT_FAILURE);
}

if (pid == 0) {
execlp("./child", "./child", filename, (char *)NULL);
print_error("Ошибка при запуске дочернего процесса\n");
exit(EXIT_FAILURE);
}

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));
if (input_len <= 0) {
print_error("Ошибка ввода строки\n");
break;
}
input[strcspn(input, "\n")] = 0;

strncpy(shared_memory, input, BUFFER_SIZE);

sem_post(sem_parent_write);

if (strcmp(input, "exit") == 0) {
break;
}

sem_wait(sem_child_read);

if (strlen(error_memory) > 0) {
write(STDERR_FILENO, error_memory, strlen(error_memory));
memset(error_memory, 0, BUFFER_SIZE);
}
}

wait(NULL);
munmap(shared_memory, BUFFER_SIZE);
```

```
munmap(error_memory, BUFFER_SIZE);
shm_unlink(SHM_NAME);
shm_unlink(SHM_ERROR_NAME);
sem_close(sem_parent_write);
sem_close(sem_child_read);
sem_unlink(SEM_PARENT_WRITE);
sem_unlink(SEM_CHILD_READ);

return 0;
}
```

child.c

```
#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>
#include <semaphore.h>
#include <sys/mman.h>

#define SHM_NAME "/shared_memory"
#define SHM_ERROR_NAME "/shared_error_memory"
#define SEM_PARENT_WRITE "/sem_parent_write"
#define SEM_CHILD_READ "/sem_child_read"
#define BUFFER_SIZE 1024

void report_error(const char *error_memory, const char *msg) {
strncpy((char *)error_memory, msg, BUFFER_SIZE);
}

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[]) {
if (argc < 2) {
write(STDERR_FILENO, "Не указано имя файла\n", 22);
return -1;
}

int shm_fd = shm_open(SHM_NAME, O_RDWR, 0666);
int shm_error_fd = shm_open(SHM_ERROR_NAME, O_RDWR, 0666);
if (shm_fd == -1 || shm_error_fd == -1) {
write(STDERR_FILENO, "Ошибка при открытии shared memory\n", 34);
return -1;
}
char *shared_memory = mmap(NULL, BUFFER_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, shm_fd,
0);
char *error_memory = mmap(NULL, BUFFER_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED,
shm_error_fd, 0);
if (shared_memory == MAP_FAILED || error_memory == MAP_FAILED) {
```

```
write(STDERR_FILENO, "Ошибка при отображении shared memory\n", 38);
return -1;
}

sem_t *sem_parent_write = sem_open(SEM_PARENT_WRITE, 0);
sem_t *sem_child_read = sem_open(SEM_CHILD_READ, 0);
if (sem_parent_write == SEM_FAILED || sem_child_read == SEM_FAILED) {
write(STDERR_FILENO, "Ошибка при открытии семафоров\n", 31);
return -1;
}

int fd = open(argv[1], O_WRONLY | O_CREAT | O_TRUNC, 0644);
if (fd < 0) {
report_error(error_memory, "Ошибка открытия файла\n");
sem_post(sem_child_read);
return -1;
}

while (1) {
sem_wait(sem_parent_write);

char buffer[BUFFER_SIZE];
strncpy(buffer, shared_memory, BUFFER_SIZE);

if (strcmp(buffer, "exit") == 0) {
break;
}

if (ends_with_dot_or_semicolon(buffer)) {
write(fd, buffer, strlen(buffer));
write(fd, "\n", 1);
} else {
report_error(error_memory, "Строка не соответствует правилу!\n");
}

sem_post(sem_child_read);
}

close(fd);
munmap(shared_memory, BUFFER_SIZE);
munmap(error_memory, BUFFER_SIZE);
sem_close(sem_parent_write);
sem_close(sem_child_read);

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"], 0x7ffe33fb7228 /* 71 vars */) = 0
brk(NULL)   = 0x648a46c40000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffd80e63f30) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7a85caa6b000
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) = 0x7a85caa5c000
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"..., 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) = 0x7a85ca800000
mprotect(0x7a85ca828000, 2023424, PROT_NONE) = 0
mmap(0x7a85ca828000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7a85ca828000
mmap(0x7a85ca9bd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7a85ca9bd000
mmap(0x7a85caa16000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x7a85caa16000
mmap(0x7a85caa1c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7a85caa1c000
close(3)   = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7a85caa59000
arch_prctl(ARCH_SET_FS, 0x7a85caa59740) = 0
set_tid_address(0x7a85caa59a10)   = 57299
set_robust_list(0x7a85caa59a20, 24)   = 0
rseq(0x7a85caa5a0e0, 0x20, 0, 0x53053053) = 0
mprotect(0x7a85caa16000, 16384, PROT_READ) = 0
mprotect(0x648a454d0000, 4096, PROT_READ) = 0
mprotect(0x7a85caaa5000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7a85caa5c000, 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", 1023)           = 11
openat(AT_FDCWD, "/dev/shm/shared_memory",
O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0666) = 3
openat(AT_FDCWD, "/dev/shm/shared_error_memory",
O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0666) = 4
ftruncate(3, 1024)                      = 0
ftruncate(4, 1024)                      = 0
mmap(NULL, 1024, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) =
0x7a85caaa4000
mmap(NULL, 1024, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) =
0x7a85caa6a000
openat(AT_FDCWD, "/dev/shm/sem.sem_parent_write", O_RDWR|O_NOFOLLOW) = -1
ENOENT (No such file or directory)
getrandom("\x55\x10\x7a\x40\x37\x1d\xba\xeb", 8, GRND_NONBLOCK) = 8
newfstatat(AT_FDCWD, "/dev/shm/sem.92lhpX", 0x7ffd80e633b0,
AT_SYMLINK_NOFOLLOW) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/dev/shm/sem.92lhpX", O_RDWR|O_CREAT|O_EXCL, 0666) = 5
write(5, "\0\0\0\0\0\0\0\200\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 32) = 32
mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 5, 0) = 0x7a85caa69000
link("/dev/shm/sem.92lhpX", "/dev/shm/sem.sem_parent_write") = 0
newfstatat(5, "", {st_mode=S_IFREG|0664, st_size=32, ...}, AT_EMPTY_PATH) = 0
getrandom("\x25\xd9\x28\x6e\x1b\x47\xd6\x2c", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x648a46c40000
brk(0x648a46c61000)                     = 0x648a46c61000
unlink("/dev/shm/sem.92lhpX")           = 0
close(5)                                = 0
openat(AT_FDCWD, "/dev/shm/sem.sem_child_read", O_RDWR|O_NOFOLLOW) = -1
ENOENT (No such file or directory)
getrandom("\xe6\x58\xb2\x32\x3b\xfb\xcc\x55", 8, GRND_NONBLOCK) = 8
newfstatat(AT_FDCWD, "/dev/shm/sem.iNVejk", 0x7ffd80e633b0,
AT_SYMLINK_NOFOLLOW) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/dev/shm/sem.iNVejk", O_RDWR|O_CREAT|O_EXCL, 0666) = 5
write(5, "\0\0\0\0\0\0\0\200\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 32) = 32
mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 5, 0) = 0x7a85caa68000
link("/dev/shm/sem.iNVejk", "/dev/shm/sem.sem_child_read") = 0
newfstatat(5, "", {st_mode=S_IFREG|0664, st_size=32, ...}, AT_EMPTY_PATH) = 0
unlink("/dev/shm/sem.iNVejk")           = 0
close(5)                                = 0
clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace: Process
57323 attached
, child_tidptr=0x7a85caa59a10) = 57323
[pid 57299] 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 57323] set_robust_list(0x7a85caa59a20, 24Введите строку (или 'exit' для выхода):
 <unfinished ...>
[pid 57299] <... write resumed>)        = 66
[pid 57323] <... set_robust_list resumed>) = 0
[pid 57299] read(0,  <unfinished ...>
[pid 57323] execve("./child", ["./child", "output.txt"], 0x7ffd80e64108 /* 71 vars */) = 0
[pid 57323] brk(NULL)                   = 0x6077f4448000
[pid 57323] arch_prctl(0x3001 /* ARCH_??? */, 0x7fffb5e2bb20) = -1 EINVAL (Invalid
argument)
```

```
[pid 57323] mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x70c23dfb4000
[pid 57323] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
[pid 57323] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
[pid 57323] newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=58575, ...},
AT_EMPTY_PATH) = 0
[pid 57323] mmap(NULL, 58575, PROT_READ, MAP_PRIVATE, 3, 0) = 0x70c23dfa5000
[pid 57323] close(3)                    = 0
[pid 57323] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6",
O_RDONLY|O_CLOEXEC) = 3
[pid 57323] 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"..., 832)
= 832
[pid 57323] 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
[pid 57323] 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
[pid 57323] 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
[pid 57323] newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...},
AT_EMPTY_PATH) = 0
[pid 57323] 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
[pid 57323] mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE,
3, 0) = 0x70c23dc00000
[pid 57323] mprotect(0x70c23dc28000, 2023424, PROT_NONE) = 0
[pid 57323] mmap(0x70c23dc28000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x70c23dc28000
[pid 57323] mmap(0x70c23ddbd000, 360448, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x70c23ddbd000
[pid 57323] mmap(0x70c23de16000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x70c23de16000
[pid 57323] mmap(0x70c23de1c000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x70c23de1c000
[pid 57323] close(3)                    = 0
[pid 57323] mmap(NULL, 12288, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x70c23dfa2000
[pid 57323] arch_prctl(ARCH_SET_FS, 0x70c23dfa2740) = 0
[pid 57323] set_tid_address(0x70c23dfa2a10) = 57323
[pid 57323] set_robust_list(0x70c23dfa2a20, 24) = 0
[pid 57323] rseq(0x70c23dfa30e0, 0x20, 0, 0x53053053) = 0
[pid 57323] mprotect(0x70c23de16000, 16384, PROT_READ) = 0
[pid 57323] mprotect(0x6077f3f58000, 4096, PROT_READ) = 0
[pid 57323] mprotect(0x70c23dfee000, 8192, PROT_READ) = 0
[pid 57323] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
[pid 57323] munmap(0x70c23dfa5000, 58575) = 0
[pid 57323] openat(AT_FDCWD, "/dev/shm/shared_memory",
O_RDWR|O_NOFOLLOW|O_CLOEXEC) = 3
[pid 57323] openat(AT_FDCWD, "/dev/shm/shared_error_memory",
O_RDWR|O_NOFOLLOW|O_CLOEXEC) = 4
[pid 57323] mmap(NULL, 1024, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) =
0x70c23dfed000
[pid 57323] mmap(NULL, 1024, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) =
0x70c23dfb3000
[pid 57323] openat(AT_FDCWD, "/dev/shm/sem.sem_parent_write",
O_RDWR|O_NOFOLLOW) = 5
```

```
[pid 57323] newfstatat(5, "", {st_mode=S_IFREG|0664, st_size=32, ...},
AT_EMPTY_PATH) = 0
[pid 57323] getrandom("\x8c\x11\xb2\xa0\x51\x49\xfb\x30", 8, GRND_NONBLOCK) = 8
[pid 57323] brk(NULL)                   = 0x6077f4448000
[pid 57323] brk(0x6077f4469000)         = 0x6077f4469000
[pid 57323] mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 5, 0) =
0x70c23dfb2000
[pid 57323] close(5)                    = 0
[pid 57323] openat(AT_FDCWD, "/dev/shm/sem.sem_child_read",
O_RDWR|O_NOFOLLOW) = 5
[pid 57323] newfstatat(5, "", {st_mode=S_IFREG|0664, st_size=32, ...},
AT_EMPTY_PATH) = 0
[pid 57323] mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 5, 0) =
0x70c23dfb1000
[pid 57323] close(5)                    = 0
[pid 57323] openat(AT_FDCWD, "output.txt", O_WRONLY|O_CREAT|O_TRUNC, 0644) =
5
[pid 57323] futex(0x70c23dfb2000,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 57299] <... read resumed>0x7ffd80e63bd0, 1024) = ? ERESTARTSYS (To be restarted
if SA_RESTART is set)
[pid 57323] <... futex resumed>)        = ? ERESTARTSYS (To be restarted if SA_RESTART
is set)
[pid 57299] --- SIGWINCH {si_signo=SIGWINCH, si_code=SI_KERNEL} ---
[pid 57323] --- SIGWINCH {si_signo=SIGWINCH, si_code=SI_KERNEL} ---
[pid 57299] read(0,  <unfinished ...>
[pid 57323] futex(0x70c23dfb2000,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANY
 <unfinished ...>
[pid 57299] <... read resumed>"\n", 1024) = 1
[pid 57299] futex(0x7a85caa69000, FUTEX_WAKE, 1) = 1
[pid 57323] <... futex resumed>)        = 0
[pid 57299] futex(0x7a85caa68000,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 57323] futex(0x70c23dfb1000, FUTEX_WAKE, 1 <unfinished ...>
[pid 57299] <... futex resumed>)        = -1 EAGAIN (Resource temporarily unavailable)
[pid 57323] <... futex resumed>)        = 0
[pid 57299] 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"..., 61 <unfinished ...>
[pid 57323] futex(0x70c23dfb2000,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANYСтрока не соответствует правилу!
 <unfinished ...>
[pid 57299] <... write resumed>)        = 61
[pid 57299] read(0, joker
"joker\n", 1024)    = 6
[pid 57299] futex(0x7a85caa69000, FUTEX_WAKE, 1) = 1
[pid 57323] <... futex resumed>)        = 0
[pid 57299] futex(0x7a85caa68000,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 57323] futex(0x70c23dfb1000, FUTEX_WAKE, 1 <unfinished ...>
[pid 57299] <... futex resumed>)        = -1 EAGAIN (Resource temporarily unavailable)
[pid 57323] <... futex resumed>)        = 0
```

```
[pid 57299] 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"..., 61 <unfinished ...>
[pid 57323] futex(0x70c23dfb2000,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANYСтрока не соответствует правилу!
 <unfinished ...>
[pid 57299] <... write resumed>)        = 61
[pid 57299] read(0, Queen;
"Queen;\n", 1024)   = 7
[pid 57299] futex(0x7a85caa69000, FUTEX_WAKE, 1) = 1
[pid 57323] <... futex resumed>)        = 0
[pid 57299] futex(0x7a85caa68000,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 57323] write(5, "Queen;", 6)       = 6
[pid 57323] write(5, "\n", 1)           = 1
[pid 57323] futex(0x70c23dfb1000, FUTEX_WAKE, 1) = 1
[pid 57299] <... futex resumed>)        = 0
[pid 57299] read(0,  <unfinished ...>
[pid 57323] futex(0x70c23dfb2000,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANY.
 <unfinished ...>
[pid 57299] <... read resumed>".\n", 1024) = 2
[pid 57299] futex(0x7a85caa69000, FUTEX_WAKE, 1) = 1
[pid 57323] <... futex resumed>)        = 0
[pid 57299] futex(0x7a85caa68000,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 57323] write(5, ".", 1)           = 1
[pid 57323] write(5, "\n", 1)          = 1
[pid 57323] futex(0x70c23dfb1000, FUTEX_WAKE, 1 <unfinished ...>
[pid 57299] <... futex resumed>)        = 0
[pid 57323] <... futex resumed>)        = 1
[pid 57299] read(0,  <unfinished ...>
[pid 57323] futex(0x70c23dfb2000,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANY;
 <unfinished ...>
[pid 57299] <... read resumed>";\n", 1024) = 2
[pid 57299] futex(0x7a85caa69000, FUTEX_WAKE, 1) = 1
[pid 57323] <... futex resumed>)        = 0
[pid 57299] futex(0x7a85caa68000,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 57323] write(5, ";", 1)           = 1
[pid 57323] write(5, "\n", 1)          = 1
[pid 57323] futex(0x70c23dfb1000, FUTEX_WAKE, 1 <unfinished ...>
[pid 57299] <... futex resumed>)        = 0
[pid 57323] <... futex resumed>)        = 1
[pid 57299] read(0,  <unfinished ...>
[pid 57323] futex(0x70c23dfb2000,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANYportal;;
 <unfinished ...>
[pid 57299] <... read resumed>"portal;;\n", 1024) = 9
[pid 57299] futex(0x7a85caa69000, FUTEX_WAKE, 1) = 1
```

```
[pid 57299] futex(0x7a85caa68000,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 57323] <... futex resumed>)        = 0
[pid 57323] write(5, "portal;;", 8)     = 8
[pid 57323] write(5, "\n", 1)           = 1
[pid 57323] futex(0x70c23dfb1000, FUTEX_WAKE, 1 <unfinished ...>
[pid 57299] <... futex resumed>)        = 0
[pid 57323] <... futex resumed>)        = 1
[pid 57299] read(0,  <unfinished ...>
[pid 57323] futex(0x70c23dfb2000,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANYexit
 <unfinished ...>
[pid 57299] <... read resumed>"exit\n", 1024) = 5
[pid 57299] futex(0x7a85caa69000, FUTEX_WAKE, 1) = 1
[pid 57323] <... futex resumed>)        = 0
[pid 57299] wait4(-1,  <unfinished ...>
[pid 57323] close(5)                    = 0
[pid 57323] munmap(0x70c23dfed000, 1024) = 0
[pid 57323] munmap(0x70c23dfb3000, 1024) = 0
[pid 57323] munmap(0x70c23dfb2000, 32)  = 0
[pid 57323] munmap(0x70c23dfb1000, 32)  = 0
[pid 57323] exit_group(0)               = ?
[pid 57323] +++ exited with 0 +++
<... wait4 resumed>NULL, 0, NULL)       = 57323
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=57323, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} ---
munmap(0x7a85caaa4000, 1024)            = 0
munmap(0x7a85caa6a000, 1024)            = 0
unlink("/dev/shm/shared_memory")        = 0
unlink("/dev/shm/shared_error_memory")  = 0
munmap(0x7a85caa69000, 32)              = 0
munmap(0x7a85caa68000, 32)              = 0
unlink("/dev/shm/sem.sem_parent_write") = 0
unlink("/dev/shm/sem.sem_child_read")   = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

В ходе выполнения лабораторной работы была составлена и отлажена программа на языке C,осуществляющая вызов дочернего процесса, переопределения стандартных ввода вывода для него и взаимодействие между дочерним и родительским процессами в операционной системе Linux с использованием общей памяти и семафоров. Была также написана программа, осуществляющая чтение данных из стандартного ввода, обработку строк и вывод информации об ошибках в стандартный вывод родительского файла.