Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу
«Операционные системы»

Тема работы
“Межпроцессорное взаимодействие через memory-mapped files”

Студент: Тутаев Владимир Владимирович
Группа: М8О-201Б-23
Вариант: 18

Преподаватель: Миронов Евгений Сергеевич

Оценка: _____
Дата: _____
Подпись: _____

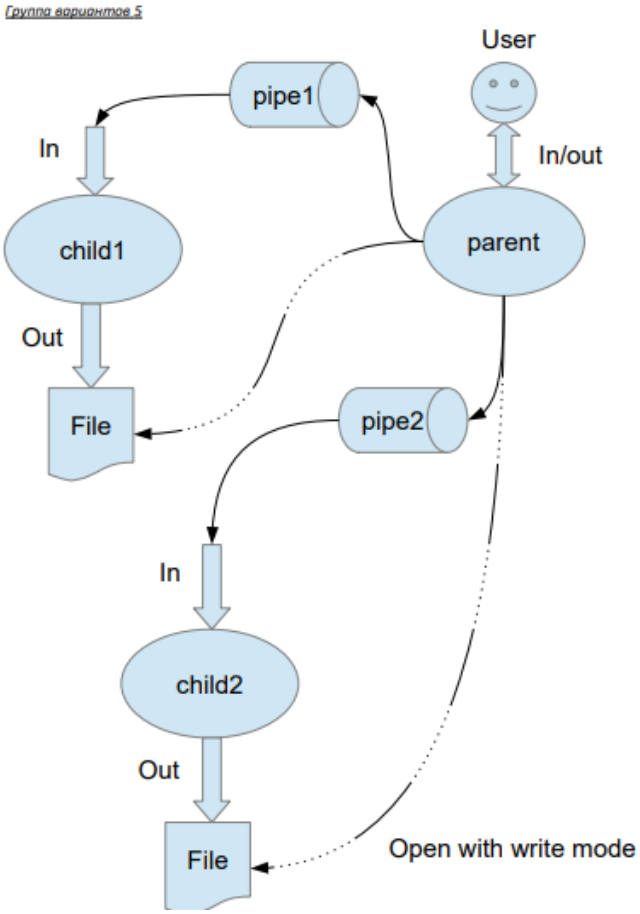
Москва, 2024

Репозиторий

https://github.com/Volan4ik/MAI_OS.git

Постановка задачи

Задача: реализовать программу, в которой родительский процесс создает два дочерних процесса. Межпроцессорное взаимодействие осуществляется посредством отображаемых файлов (memory-mapped files).



Вариант 18) Правило фильтрации: нечетные строки отправляются в pipe1, четные в pipe2. Дочерние процессы удаляют все гласные из строк.

Общие сведения

Родительский процесс создает два дочерних процесса. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись для child1. Аналогично для второй строки и процесса child2. Родительский и дочерний процесс должны быть представлены разными программами.

Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1 или в pipe2 в зависимости от правила фильтрации. Процесс child1 и child2 производят работу над строками. Процессы пишут результаты своей работы в стандартный вывод.

Исходный код в Приложении 1

Strace в Приложении 2

Выводы

В этой лабораторной работе я закрепил навыки взаимодействия с дочерними процессами, научился использовать семафоры, чтобы синхронизовать процессы. Узнал, как работать с файлами, которые отражены в память (mmap)

Приложение 1

child.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <ctype.h>
#include <signal.h>
#include <sys/mman.h>

#define SHARED_MEMORY_SIZE 1024

static volatile sig_atomic_t sigflag;
static sigset_t newmask, oldmask, zeromask;

void handle_sigusr1(int sig) {
    sigflag = 1;
    printf("Дочерний процесс получил SIGUSR1\n");
}

void WAIT_PARENT(void) {
    while (sigflag == 0)
        sigsuspend(&zeromask);
    sigflag = 0;
}

void TELL_PARENT(pid_t pid) {
    kill(pid, SIGUSR2);
}

void remove_vowels(char *str) {
    char *src = str, *dst = str;
    while (*src) {
        if (*src == 'a' || *src == 'e' || *src == 'i' || *src == 'o' || *src == 'u' ||
            *src == 'A' || *src == 'E' || *src == 'I' || *src == 'O' || *src == 'U') {
            src++;
        } else {
            *dst++ = *src++;
        }
    }
    *dst = '\0';
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
        fprintf(stderr, "Использование: %s <имя_файла>\n", argv[0]);
        return 1;
    }
```

```
    int fd = open("/tmp/shared_memory_file", O_RDWR, 0666);
    if (fd == -1) {
        perror("open");
        exit(EXIT_FAILURE);
    }

    void *shared_mem_child = mmap(NULL, SHARED_MEMORY_SIZE, PROT_READ | PROT_WRITE,
MAP_SHARED, fd, 0);
    if (shared_mem_child == MAP_FAILED) {
        perror("mmap");
        exit(EXIT_FAILURE);
    }

    close(fd);

    int file = open(argv[1], O_WRONLY | O_APPEND | O_CREAT, 0666);
    if (file == -1) {
        perror("open");
        exit(EXIT_FAILURE);
    }

    struct sigaction sa;
    sa.sa_handler = handle_sigusr1;
    sigemptyset(&sa.sa_mask);
    sa.sa_flags = 0;
    if (sigaction(SIGUSR1, &sa, NULL) == -1) {
        perror("sigaction");
        exit(EXIT_FAILURE);
    }

    pid_t parentPID = getppid();

    sigemptyset(&newmask);
    sigaddset(&newmask, SIGUSR1);
    sigemptyset(&zeromask);

    if (sigprocmask(SIG_BLOCK, &newmask, &oldmask) < 0) {
        perror("sigprocmask");
        exit(EXIT_FAILURE);
    }

    char str[SHARED_MEMORY_SIZE] = {0};

    while (1) {
        WAIT_PARENT();

        strncpy(str, (char *)shared_mem_child, SHARED_MEMORY_SIZE);

        if (strcmp(str, "quit") == 0) {
            TELL_PARENT(parentPID);
            break;
        }
```

```
        remove_vowels(str);
        write(file, str, strlen(str));
        write(file, "\n", 1);

        TELL_PARENT(parentPID);
    }

    if (sigprocmask(SIG_SETMASK, &oldmask, NULL) < 0) {
        perror("sigprocmask");
        return 1;
    }

    close(file);

    exit(EXIT_SUCCESS);
}
```

parent.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <signal.h>
#include <sys/mman.h>

#define SHARED_MEMORY_SIZE 1024

static volatile sig_atomic_t sigflag;
static sigset_t newmask, oldmask, zeromask;

void handle_sigusr2(int sig) {
    sigflag = 1;
    printf("Родительский процесс получил SIGUSR2\n");
}

void handle_child_ready(int sig) {
    sigflag = 1;
    printf("Родительский процесс получил SIGUSR1\n");
}

void WAIT_CHILD(void) {
    while (sigflag == 0)
        sigsuspend(&zeromask);
    sigflag = 0;
}

void TELL_CHILD(pid_t pid) {
    kill(pid, SIGUSR1);
}
```

```
int main() {
    struct sigaction sa;
    sa.sa_handler = handle_sigusr2;
    sigemptyset(&sa.sa_mask);
    sa.sa_flags = 0;
    if (sigaction(SIGUSR2, &sa, NULL) == -1) {
        perror("sigaction");
        exit(EXIT_FAILURE);
    }

    sa.sa_handler = handle_child_ready;
    if (sigaction(SIGUSR1, &sa, NULL) == -1) {
        perror("sigaction");
        exit(EXIT_FAILURE);
    }

    sigemptyset(&newmask);
    sigaddset(&newmask, SIGUSR1);
    sigemptyset(&zeromask);

    if (sigprocmask(SIG_BLOCK, &newmask, &oldmask) < 0) {
        perror("sigprocmask");
        exit(EXIT_FAILURE);
    }

    char filename1[1024], filename2[1024];
    printf("Enter filename for child1: ");
    fgets(filename1, sizeof(filename1), stdin);
    filename1[strcspn(filename1, "\n")] = '\0';

    printf("Enter filename for child2: ");
    fgets(filename2, sizeof(filename2), stdin);
    filename2[strcspn(filename2, "\n")] = '\0';

    int fd = open("/tmp/shared_memory_file", O_RDWR | O_CREAT, 0666);
    if (fd == -1) {
        perror("open");
        exit(EXIT_FAILURE);
    }

    if (ftruncate(fd, SHARED_MEMORY_SIZE) == -1) {
        perror("ftruncate");
        exit(EXIT_FAILURE);
    }

    void *shared_memory = mmap(NULL, SHARED_MEMORY_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED,
fd, 0);
    if (shared_memory == MAP_FAILED) {
        perror("mmap");
        exit(EXIT_FAILURE);
    }
```

```
close(fd);

pid_t pid1 = fork();
if (pid1 == -1) {
    perror("fork");
    exit(EXIT_FAILURE);
}

if (pid1 == 0) {
    kill(getppid(), SIGUSR1);

    char *args[] = {"./child", filename1, NULL};
    if (execv(args[0], args) == -1) {
        perror("execv error");
        return 1;
    }
} else {
    WAIT_CHILD();

    pid_t pid2 = fork();
    if (pid2 == -1) {
        perror("fork");
        exit(EXIT_FAILURE);
    }

    if (pid2 == 0) {
        kill(getppid(), SIGUSR1);

        char *args[] = {"./child", filename2, NULL};
        if (execv(args[0], args) == -1) {
            perror("execv error");
            return 1;
        }
    } else {
        WAIT_CHILD();

        char buffer[1024];
        int line_number = 0;

        while (1) {
            printf("Введите строку (для завершения введите 'quit'): ");
            if (fgets(buffer, sizeof(buffer), stdin) == NULL) {
                perror("fgets");
                exit(EXIT_FAILURE);
            }

            buffer[strcspn(buffer, "\n")] = '\0';

            snprintf((char *)shared_memory, SHARED_MEMORY_SIZE, "%s", buffer);

            if (line_number % 2 == 0) {
                TELL_CHILD(pid1);
            } else {
```

```
                TELL_CHILD(pid2);
            }

            if (strcmp(buffer, "quit") == 0)
                break;

            WAIT_CHILD();

            line_number++;
        }

        if (sigprocmask(SIG_SETMASK, &oldmask, NULL) < 0) {
            perror("sigprocmask");
            return 1;
        }

        munmap(shared_memory, SHARED_MEMORY_SIZE);
        wait(NULL);
        wait(NULL);
        exit(EXIT_SUCCESS);
    }
  }
}
```

Приложение 2

execve("./parent", ["./parent"], 0x7ffd37ebd7e0 /* 75 vars */) = 0

brk(NULL) = 0x63d9302b4000

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fd380efe000

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

fstat(3, {st_mode=S_IFREG|0644, st_size=87915, ...}) = 0

mmap(NULL, 87915, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fd380ee8000

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\0\220\243\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

fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 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, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fd380c00000

mmap(0x7fd380c28000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7fd380c28000

mmap(0x7fd380db0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x7fd380db0000

mmap(0x7fd380dff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x7fd380dff000

mmap(0x7fd380e05000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fd380e05000

close(3) = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fd380ee5000

arch_prctl(ARCH_SET_FS, 0x7fd380ee5740) = 0

set_tid_address(0x7fd380ee5a10) = 920540

set_robust_list(0x7fd380ee5a20, 24) = 0

rseq(0x7fd380ee6060, 0x20, 0, 0x53053053) = 0

mprotect(0x7fd380dff000, 16384, PROT_READ) = 0

mprotect(0x63d92fe24000, 4096, PROT_READ) = 0

mprotect(0x7fd380f36000, 8192, PROT_READ) = 0

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

munmap(0x7fd380ee8000, 87915) = 0

rt_sigaction(SIGUSR2, {sa_handler=0x63d92fe22429, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7fd380c45320}, NULL, 8) = 0

rt_sigaction(SIGUSR1, {sa_handler=0x63d92fe22451, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0x7fd380c45320}, NULL, 8) = 0

rt_sigprocmask(SIG_BLOCK, [USR1], [], 8) = 0

fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0

getrandom("\xdb\x10\xb2\x32\x90\xe9\x7b\x38", 8, GRND_NONBLOCK) = 8

brk(NULL) = 0x63d9302b4000

brk(0x63d9302d5000) = 0x63d9302d5000

fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0

write(1, "Enter filename for child1: ", 27) = 27

read(0, "tests1.txt\n", 1024) = 11

write(1, "Enter filename for child2: ", 27) = 27

read(0, "tests2.txt\n", 1024) = 11

openat(AT_FDCWD, "/tmp/shared_memory_file", O_RDWR|O_CREAT, 0666) = 3

ftruncate(3, 1024) = 0

mmap(NULL, 1024, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7fd380efd000

close(3) = 0

clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7fd380ee5a10) = 920603

rt_sigsuspend([], 8) = ? ERESTARTNOHAND (To be restarted if no handler)

--- SIGUSR1 {si_signo=SIGUSR1, si_code=SI_USER, si_pid=920603, si_uid=1000} ---

write(1, "\320\240\320\276\320\264\320\270\321\202\320\265\320\273\321\214\321\201\320\272\320\270\320\271\320\277\321\200\320\276\321"..., 63) = 63

rt_sigreturn({mask=[USR1]}) = -1 EINTR (Interrupted system call)

clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7fd380ee5a10) = 920604

rt_sigsuspend([], 8) = ? ERESTARTNOHAND (To be restarted if no handler)

--- SIGUSR1 {si_signo=SIGUSR1, si_code=SI_USER, si_pid=920604, si_uid=1000} ---

write(1, "\320\240\320\276\320\264\320\270\321\202\320\265\320\273\321\214\321\201\320\272\320\270\320\271\320\277\321\200\320\276\321"..., 63) = 63

rt_sigreturn({mask=[USR1]}) = -1 EINTR (Interrupted system call)

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\264\320"..., 81) = 81

read(0, "\321\207\321\221\321\202\320\275\320\276\320\265\n", 1024) = 13

kill(920603, SIGUSR1) = 0

rt_sigsuspend([], 8) = ? ERESTARTNOHAND (To be restarted if no handler)

--- SIGUSR2 {si_signo=SIGUSR2, si_code=SI_USER, si_pid=920603, si_uid=1000} ---

write(1, "\320\240\320\276\320\264\320\270\321\202\320\265\320\273\321\214\321\201\320\272\320\270\320\271\320\277\321\200\320\276\321"..., 63) = 63

```
rt_sigreturn({mask=[USR1]})            = -1 EINTR (Interrupted system call)

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\264\320"..., 81) = 81

read(0, "\320\275\320\265\321\207\320\265\321\202\320\275\320\276\320\265\320\265\n", 1024) = 19

kill(920604, SIGUSR1)                  = 0

rt_sigsuspend([], 8)                   = ? ERESTARTNOHAND (To be restarted if no handler)

--- SIGUSR2 {si_signo=SIGUSR2, si_code=SI_USER, si_pid=920604, si_uid=1000} ---

write(1, "\320\240\320\276\320\264\320\270\321\202\320\265\320\273\321\214\321\201\320\272\320\270\320\271
\320\277\321\200\320\276\321"..., 63) = 63

rt_sigreturn({mask=[USR1]})            = -1 EINTR (Interrupted system call)

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\264\320"..., 81) = 81

read(0, "quit\n", 1024)                = 5

kill(920603, SIGUSR1)                  = 0

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

--- SIGUSR2 {si_signo=SIGUSR2, si_code=SI_USER, si_pid=920603, si_uid=1000} ---

write(1, "\320\240\320\276\320\264\320\270\321\202\320\265\320\273\321\214\321\201\320\272\320\270\320\271
\320\277\321\200\320\276\321"..., 63) = 63

rt_sigreturn({mask=[]})                = 0

munmap(0x7fd380efd000, 1024)           = 0

--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=920603, si_uid=1000, si_status=0,
si_utime=0, si_stime=0} ---

wait4(-1, NULL, 0, NULL)               = 920603

wait4(-1, NULL, 0, NULL)               = ? ERESTARTSYS (To be restarted if SA_RESTART is set)

--- SIGINT {si_signo=SIGINT, si_code=SI_KERNEL} ---

+++ killed by SIGINT +++
```