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-210Б-23

Студент: Тульчинский Г.С

Преподаватель: Бахарев В.Д. (ФИИТ)

Оценка: _____

Дата: 24 .12.24

Москва, 2024

Постановка задачи

Постановка задачи

Вариант 2.

Пользователь вводит команды вида: число число число<endline>. Далее эти числа передаются от родительского процесса в дочерний. Дочерний процесс считает их сумму и выводит её в файл. Числа имеют тип float. Количество чисел может быть произвольным.

Общий метод и алгоритм решения

Использованные системные вызовы:

- mmap – отображение файла в память
- fork – создание дочернего процесса
- execv – замена исполняемого кода
- sem_open – создание/подключение к семафору
- sem_post – поднятие семафора
- sem_wait – опускание семафора
- wait – ожидание завершения процесса
- kill – завершение процесса
- sem_unlink - уничтожает именованный семафор
- shm_open – открывает объект разделяемой памяти
- ftruncate - укорачивает файл до указанной длины

Код программы

parent.c

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

#define SHM_NAME "/shared_memory_example"
#define SEM_PARENT "/sem_parent"
#define SEM_CHILD "/sem_child"
#define BUF_SIZE 1024

int main() {
    char filename[BUF_SIZE];
    printf("Введите имя выходного файла: ");
    fgets(filename, BUF_SIZE, stdin);
    filename[strcspn(filename, "\n")] = '\0';
    int shm_fd = shm_open(SHM_NAME, O_CREAT | O_RDWR, 0666);
    if (shm_fd == -1) {
        perror("shm_open");
        exit(EXIT_FAILURE);
    }
    ftruncate(shm_fd, BUF_SIZE);
    char *shared_memory = mmap(0, BUF_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, shm_fd, 0);
    if (shared_memory == MAP_FAILED) {
        perror("mmap");
        exit(EXIT_FAILURE);
    }
    sem_t *sem_parent = sem_open(SEM_PARENT, O_CREAT, 0666, 0);
    sem_t *sem_child = sem_open(SEM_CHILD, O_CREAT, 0666, 0);
    if (sem_parent == SEM_FAILED || sem_child == SEM_FAILED) {
        perror("sem_open");
        exit(EXIT_FAILURE);
    }
```

```
    pid_t pid = fork();
    if (pid == -1) {
        perror("fork");
        exit(EXIT_FAILURE);
    }
    if (pid == 0) {
        execl("./child", "./child", filename, NULL);
        perror("execl");
        exit(EXIT_FAILURE);
    }
    while (1) {
        printf("Введите числа, разделенные пробелами (или 'exit' чтобы завершить): ");
        fgets(shared_memory, BUF_SIZE, stdin);
        shared_memory[strcspn(shared_memory, "\n")] = '\0';
        sem_post(sem_parent);
        if (strcmp(shared_memory, "exit") == 0) {
            break;
        }
        sem_wait(sem_child);
    }
    wait(NULL);
    munmap(shared_memory, BUF_SIZE);
    shm_unlink(SHM_NAME);
    sem_close(sem_parent);
    sem_close(sem_child);
    sem_unlink(SEM_PARENT);
    sem_unlink(SEM_CHILD);
    return 0;
}
```

child.c

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

#define SHM_NAME "/shared_memory_example"
#define SEM_PARENT "/sem_parent"
#define SEM_CHILD "/sem_child"
#define BUF_SIZE 1024

int main(int argc, char *argv[]) {
    if (argc != 2) {
        fprintf(stderr, "Использование: %s <имя файла>\n", argv[0]);
        exit(EXIT_FAILURE);
    }
    char *filename = argv[1];

    int shm_fd = shm_open(SHM_NAME, O_RDWR, 0666);
    if (shm_fd == -1) {
        perror("shm_open");
        exit(EXIT_FAILURE);
    }
    char *shared_memory = mmap(0, BUF_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, shm_fd, 0);
    if (shared_memory == MAP_FAILED) {
        perror("mmap");
        exit(EXIT_FAILURE);
    }
    sem_t *sem_parent = sem_open(SEM_PARENT, 0);
    sem_t *sem_child = sem_open(SEM_CHILD, 0);
    if (sem_parent == SEM_FAILED || sem_child == SEM_FAILED) {
        perror("sem_open");
        exit(EXIT_FAILURE);
    }
    while (1) {
```

```c
    while (1) {
        sem_wait(sem_parent);
        if (strcmp(shared_memory, "exit") == 0) {
            break;
        }
        char *token = strtok(shared_memory, " ");
        float sum = 0;
        while (token != NULL) {
            sum += atof(token);
            token = strtok(NULL, " ");
        }
        FILE *file = fopen(filename, "a");
        if (!file) {
            perror("fopen");
            exit(EXIT_FAILURE);
        }
        fprintf(file, "Сумма: %.2f\n", sum);
        fclose(file);
        sem_post(sem_child); // Уведомляем родителя, что данные обработаны
    }
    munmap(shared_memory, BUF_SIZE);
    sem_close(sem_parent);
    sem_close(sem_child);
    return 0;
}
```

```
execve("./lab", ["./lab", "output.txt"], 0x7ffe0b6df7e8 /* 35 vars */) = 0
brk(NULL)                               = 0x55f0fd9d4000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc692b7120) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f6daffd1000
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=16995, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 16995, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f6daffcc000
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\0300\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\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"..., 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) = 0x7f6dafda3000
mprotect(0x7f6dafdcb000, 2023424, PROT_NONE) = 0
mmap(0x7f6dafdcb000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000)
 = 0x7f6dafdcb000
mmap(0x7f6daff60000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7f6daff60000
mmap(0x7f6daffb9000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000)
 = 0x7f6daffb9000
mmap(0x7f6daffbf000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0)
 = 0x7f6daffbf000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f6dafda0000
arch_prctl(ARCH_SET_FS, 0x7f6dafda0740) = 0
set_tid_address(0x7f6dafda0a10)         = 16955
set_robust_list(0x7f6dafda0a20, 24)     = 0
rseq(0x7f6dafda10e0, 0x20, 0, 0x53053053) = 0
mprotect(0x7f6daffb9000, 16384, PROT_READ) = 0
mprotect(0x55f0c1d55000, 4096, PROT_READ) = 0
mprotect(0x7f6db000b000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7f6daffcc000, 16995)           = 0
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x4), ...}, AT_EMPTY_PATH) = 0
getrandom("\x89\xd8\x30\x3b\x0e\x0e\xf3\x3e", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x55f0fd9d4000
```

```
write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\270\320\274\321\217 \320\262\321\213\321\
205\320\276\320\264"..., 53Введите имя выходного файла: ) = 53
read(0, 1.txt
"1.txt\n", 99) = 6
unlink("/dev/shm/sem.semaphore") = -1 ENOENT (No such file or directory)
unlink("/dev/shm/shared_memory") = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/dev/shm/shared_memory",
O_RDWR|O_CREAT|O_EXCL|O_NOFOLLOW|O_CLOEXEC, 0666) = 3
ftruncate(3, 4096) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f54c162f000
getrandom("\x40\x3b\x32\x94\x84\x4b\x00\x06", 8, GRND_NONBLOCK) = 8
newfstatat(AT_FDCWD, "/dev/shm/sem.2psWY6", 0x7ffeeba08ba0, AT_SYMLINK_NOFOLLOW) = -1
ENOENT (No such file or directory)
openat(AT_FDCWD, "/dev/shm/sem.2psWY6", O_RDWR|O_CREAT|O_EXCL, 0666) = 4
write(4, "\0\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", 32) = 32
mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7f54c15f5000
link("/dev/shm/sem.2psWY6", "/dev/shm/sem.semaphore") = 0
newfstatat(4, "", {st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH) = 0
getrandom("\x1c\xe3\x53\xd4\xc3\xdb\x6b\xf0", 8, GRND_NONBLOCK) = 8
brk(NULL) = 0x555fed568000
brk(0x555fed589000) = 0x555fed589000
unlink("/dev/shm/sem.2psWY6") = 0
close(4) = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7f54c13c0a10) = 1671
read(0, 10 2 6
"10 2 6\n", 4096) = 7
futex(0x7f54c15f5000, FUTEX_WAKE, 1) = 1
read(0, 15 3 9
"15 3 9\n", 4096) = 7
futex(0x7f54c15f5000, FUTEX_WAKE, 1) = 1
read(0, 7 2 4
"7 2 4\n", 4096) = 6
futex(0x7f54c15f5000, FUTEX_WAKE, 1) = 1
read(0, "", 4096) = 0
futex(0x7f54c15f5000, FUTEX_WAKE, 1) = 1
Child finished
wait4(-1, NULL, 0, NULL) = 1671
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=1671, si_uid=1000, si_status=0,
si_utime=0, si_stime=0} ---
munmap(0x7f54c162f000, 4096) = 0
close(3) = 0
unlink("/dev/shm/shared_memory") = 0
munmap(0x7f54c15f5000, 32) = 0
unlink("/dev/shm/sem.semaphore") = 0
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH) = 0
write(1, "Parent finished\n", 16Parent finished) = 16 exit_group(0) = ?
+++ exited with 0 +++
```

Вывод

В ходе лабораторной работе я приобрел базовые навыки по работе с разделяемой памятью в си. Я научился создавать объект разделяемой памяти, записывать в него данные и читать их из него. Также я узнал о работе с семафорами, научился использовать их для синхронизации при работе с разделяемой памятью. Помимо этого, я узнал о файловых системах и памяти в целом.