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: Петров М.А.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 01.12.24

Москва, 2024

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:

1. shmget - используется для создания или получения доступа к сегменту разделяемой памяти.
2. fork - порождает дочерний процесс.
3. shmat - присоединяет сегмент разделяемой памяти к адресному пространству процесса.
4. strlen - вычисляет длину строки.
5. strcpy - копирует строку.
6. atof - преобразует строку в число с плавающей точкой.
7. fopen - открывает файл.
8. fprintf - записывает форматированные данные в файл.
9. fclose - закрывает файл.
10. memset - заполняет указанную область памяти заданным значением.
11. shmdt - отсоединяет сегмент разделяемой памяти от адресного пространства процесса.
12. shmctl - используется для управления сегментами разделяемой памяти.
13. wait - ожидает завершения дочернего процесса.

Программа создает сегмент разделяемой памяти, затем порождает дочерний процесс, который прикрепляет этот сегмент и ждет данные от родителя. Родительский процесс получает данные от пользователя и записывает их в разделяемую память. Дочерний процесс суммирует числа и сохраняет результат в файл. Обе программы завершают работу по команде "exit".

Код программы

parent.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/wait.h>

#define SHM_SIZE 1024 // размер общей памяти

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 shmid;
    char *shm_ptr;
```

```
    // Создаем сегмент общей памяти
    shmid = shmget(IPC_PRIVATE, SHM_SIZE, IPC_CREAT | 0666);
    if (shmid < 0) {
        perror("shmget");
        exit(EXIT_FAILURE);
    }

    // Порождаем дочерний процесс
    pid_t pid = fork();
    if (pid < 0) {
        perror("fork");
        exit(EXIT_FAILURE);
    }

    if (pid == 0) { // Дочерний процесс
        // Прикрепляем сегмент общей памяти
        shm_ptr = shmat(shmid, NULL, 0);
        if (shm_ptr == (char *)(-1)) {
            perror("shmat");
            exit(EXIT_FAILURE);
        }

        while (1) {
            // Ждем данные от родителя
            if (strlen(shm_ptr) > 0) {
                if (strcmp(shm_ptr, "exit") == 0) {
                    break; // Выход при получении команды exit
                }

                // Обработка чисел
                float sum = 0.0;
                char *token = strtok(shm_ptr, " ");
                while (token != NULL) {
                    sum += atof(token);
                    token = strtok(NULL, " ");
                }

                // Записываем результат в файл
                FILE *file = fopen(filename, "a");
                if (file != NULL) {
                    fprintf(file, "Сумма: %.2f\n", sum);
                    fclose(file);
                } else {
                    perror("fopen");
                }

                // Очищаем содержимое общей памяти
                memset(shm_ptr, 0, SHM_SIZE);
            }
        }
        // Отключаем сегмент общей памяти
        shmdt(shm_ptr);
        exit(EXIT_SUCCESS);
    } else { // Родительский процесс
        // Прикрепляем сегмент общей памяти
        shm_ptr = shmat(shmid, NULL, 0);
```

```
            if (shm_ptr == (char *)(-1)) {
                perror("shmat");
                exit(EXIT_FAILURE);
            }

            char input[SHM_SIZE];
            while (1) {
                printf("Введите числа (или 'exit' для выхода): ");
                fgets(input, SHM_SIZE, stdin);
                input[strcspn(input, "\n")] = 0; // Удаляем символ новой строки

                // Записываем данные в общую память
                strncpy(shm_ptr, input, SHM_SIZE);

                if (strcmp(input, "exit") == 0) {
                    break; // Выход при получении команды exit
                }
            }

            // Отключаем сегмент общей памяти и удаляем его
            shmdt(shm_ptr);
            shmctl(shmid, IPC_RMID, NULL);
            wait(NULL); // Ожидаем завершения дочернего процесса
            exit(EXIT_SUCCESS);
        }
    }
```

Child.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

#define BUFFER_SIZE 1024

int main() {
    float *shared_sum;
    key_t key = IPC_PRIVATE;
    int shm_id = shmget(key, sizeof(float), IPC_CREAT | 0666);

    // Присоединяем общую память
    shared_sum = (float*) shmat(shm_id, NULL, 0);
    if (shared_sum == (float*) -1) {
        perror("shmat");
        exit(EXIT_FAILURE);
    }

    char buffer[BUFFER_SIZE];
    float sum = 0.0;
```

```
    // Читаем команды от родительского процесса
    while (read(STDIN_FILENO, buffer, BUFFER_SIZE) > 0) {
        char *token = strtok(buffer, " ");
        while (token != NULL) {
            sum += atof(token);
            token = strtok(NULL, " ");
        }
    }

    // Сохраняем сумму в общей памяти
    *shared_sum = sum;

    // Закрываем память и завершаем дочерний процесс
    shmdt(shared_sum);
    return 0;
}
```

Протокол работы программы

Тестирование:

```
markvolkov@MacBook-Air-Mark-2 LAB3 % ./parent output.txt
Введите числа (или 'exit' для выхода): 30.3
Введите числа (или 'exit' для выхода): 30.3 45
Введите числа (или 'exit' для выхода):
Введите числа (или 'exit' для выхода): exit
```

Strace:

strace -f ./p output.txt

execve("./p", ["./p", "output.txt"], 0x7ffecb135060 /* 46 vars */) = 0

brk(NULL) = 0x608d1843e000

arch_prctl(0x3001 /* ARCH_??? */, 0x7ffceb6f3af0) = -1 EINVAL (Недопустимый аргумент)

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7705f2433000

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

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=58047, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 58047, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7705f2424000

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\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) = 0x7705f2000000

mprotect(0x7705f2028000, 2023424, PROT_NONE) = 0

mmap(0x7705f2028000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7705f2028000

mmap(0x7705f21bd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7705f21bd000

```
mmap(0x7705f2216000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x7705f2216000
mmap(0x7705f221c000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7705f221c000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7705f2421000
arch_prctl(ARCH_SET_FS, 0x7705f2421740) = 0
set_tid_address(0x7705f2421a10)         = 4202
set_robust_list(0x7705f2421a20, 24)     = 0
rseq(0x7705f24220e0, 0x20, 0, 0x53053053) = 0
mprotect(0x7705f2216000, 16384, PROT_READ) = 0
mprotect(0x608cd88c1000, 4096, PROT_READ) = 0
mprotect(0x7705f246d000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY})
= 0
munmap(0x7705f2424000, 58047)           = 0
shmget(IPC_PRIVATE, 1024, IPC_CREAT|0666) = 5
clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace: Process 4203 attached
, child_tidptr=0x7705f2421a10) = 4203
[pid  4202] shmat(5, NULL, 0 <unfinished ...>
[pid  4203] set_robust_list(0x7705f2421a20, 24) = 0
[pid  4202] <... shmat resumed>)        = 0x7705f246c000
[pid  4203] shmat(5, NULL, 0 <unfinished ...>
[pid  4202] newfstatat(1, "",  <unfinished ...>
[pid  4203] <... shmat resumed>)        = 0x7705f246c000
[pid  4202] <... newfstatat resumed>{st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x2), ...},
AT_EMPTY_PATH) = 0
[pid  4202] getrandom("\xba\x73\x62\x27\x41\x0a\x85\xaa", 8, GRND_NONBLOCK) = 8
[pid  4202] brk(NULL)                   = 0x608d1843e000
[pid  4202] brk(0x608d1845f000)         = 0x608d1845f000
[pid  4202] newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x2), ...},
AT_EMPTY_PATH) = 0
[pid  4202] write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\321\207\320\270\321\201\320\273\320\260 (\320\270\320\273\320"..., 63Введите числа (или 'exit' для
выхода): ) = 63
[pid  4202] read(0, 1223
"1223\n", 1024)     = 5
[pid  4202] write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\321\207\320\270\321\201\320\273\320\260 (\320\270\320\273\320"..., 63Введите числа (или 'exit' для
выхода): ) = 63
[pid  4202] read(0,  <unfinished ...>
[pid  4203] getrandom("\x4b\x80\x52\xcf\xec\x0f\x9d\x56", 8, GRND_NONBLOCK) = 8
[pid  4203] brk(NULL)                   = 0x608d1843e000
[pid  4203] brk(0x608d1845f000)         = 0x608d1845f000
[pid  4203] openat(AT_FDCWD, "output.txt", O_WRONLY|O_CREAT|O_APPEND, 0666) = 3
[pid  4203] lseek(3, 0, SEEK_END)       = 131
[pid  4203] newfstatat(3, "", {st_mode=S_IFREG|0664, st_size=131, ...}, AT_EMPTY_PATH) = 0
[pid  4203] write(3, "\320\241\321\203\320\274\320\274\320\260: 1223.00\n", 20) = 20
[pid  4203] close(3)                    = 0
exit
```

[pid 4202] <... read resumed>"exit\n", 1024) = 5
[pid 4202] **shmdt(0x7705f246c000)   = 0**
[pid 4202] shmctl(5, IPC_RMID, NULL)   = 0
[pid 4202] wait4(-1, <unfinished ...>
[pid 4203] **shmdt(0x7705f246c000)   = 0**
[pid 4203] exit_group(0)   = ?
[pid 4203] +++ exited with 0 +++
<... wait4 resumed>NULL, 0, NULL)   = 4203
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=4203, si_uid=1000, si_status=0, si_utime=39, si_stime=49} ---
exit_group(0)   = ?
+++ exited with 0 +++

Вывод

Эта программа демонстрирует использование механизма разделяемой памяти для коммуникации между родительским и дочерним процессами в Unix-подобных системах. Она позволяет эффективно обмениваться данными и выполнять их обработку в реальном времени, предоставляя простой способ взаимодействия процессов.