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-211Б-23

Студент: Соболин Т.С.

Преподаватель: Бахарев В.Д.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 13.01.25

Москва, 2025

# 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:

* shmget - используется для создания или получения доступа к сегменту разделяемой памяти.
* fork - порождает дочерний процесс.
* shmat - присоединяет сегмент разделяемой памяти к адресному пространству процесса.
* strlen - вычисляет длину строки.
* strcpy - копирует строку.
* atof - преобразует строку в число с плавающей точкой.
* fopen - открывает файл.
* fprintf - записывает форматированные данные в файл.
* fclose - закрывает файл.
* memset - заполняет указанную область памяти заданным значением.
* shmdt - отсоединяет сегмент разделяемой памяти от адресного пространства процесса.
* shmctl - используется для управления сегментами разделяемой памяти.

\* 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**

|  |
| --- |
| #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:**

kotlasboy@kotlasboy-Modern-15-B12M:~/Programming/Projects/OS/lab\_3$ ./a.out 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

kotlasboy@kotlasboy-Modern-15-B12M:~/Programming/Projects/OS/lab\_3$ strace -f ./a.out output.txt

execve("./a.out", ["./a.out", "output.txt"], 0x7ffd64de7470 /\* 60 vars \*/) = 0

brk(NULL) = 0x59b9f5716000

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7e764ace7000

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=68847, ...}) = 0

mmap(NULL, 68847, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7e764acd6000

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\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) = 0x7e764aa00000

mmap(0x7e764aa28000, 1605632, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7e764aa28000

mmap(0x7e764abb0000, 323584, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1b0000) = 0x7e764abb0000

mmap(0x7e764abff000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1fe000) = 0x7e764abff000

mmap(0x7e764ac05000, 52624, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7e764ac05000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7e764acd3000

arch\_prctl(ARCH\_SET\_FS, 0x7e764acd3740) = 0

set\_tid\_address(0x7e764acd3a10) = 65410

set\_robust\_list(0x7e764acd3a20, 24) = 0

hrseq(0x7e764acd4060, 0x20, 0, 0x53053053) = 0

mprotect(0x7e764abff000, 16384, PROT\_READ) = 0

mprotect(0x59b9e674c000, 4096, PROT\_READ) = 0

mprotect(0x7e764ad1f000, 8192, PROT\_READ) = 0

prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

munmap(0x7e764acd6000, 68847) = 0

shmget(IPC\_PRIVATE, 1024, IPC\_CREAT|0666) = 32789

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLDstrace: Process 65411 attached

, child\_tidptr=0x7e764acd3a10) = 65411

[pid 65410] shmat(32789, NULL, 0 <unfinished ...>

[pid 65411] set\_robust\_list(0x7e764acd3a20, 24) = 0

[pid 65410] <... shmat resumed>) = 0x7e764ace6000

[pid 65410] 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"..., 63 <unfinished ...>

Введите числа (или 'exit' для выхода): [pid 65411] shmat(32789, NULL, 0 <unfinished ...>

[pid 65410] <... write resumed>) = 63

[pid 65410] read(0, <unfinished ...>

[pid 65411] <... shmat resumed>) = 0x7e764ace6000

30.3

[pid 65410] <... read resumed>"30.3\n", 1024) = 5

[pid 65410] 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"..., 63 <unfinished ...>

[pid 65411] openat(AT\_FDCWD, "output.txt", O\_WRONLY|O\_CREAT|O\_APPEND, 0666Введите числа (или 'exit' для выхода): <unfinished ...>

[pid 65410] <... write resumed>) = 63

[pid 65411] <... openat resumed>) = 3

[pid 65410] read(0, <unfinished ...>

[pid 65411] write(3, "\320\241\321\203\320\274\320\274\320\260: 30.30\n", 18) = 18

[pid 65411] close(3) = 0

30.3 45

[pid 65410] <... read resumed>"30.3 45\n", 1024) = 8

[pid 65410] 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"..., 63 <unfinished ...>

[pid 65411] openat(AT\_FDCWD, "output.txt", O\_WRONLY|O\_CREAT|O\_APPEND, 0666Введите числа (или 'exit' для выхода): <unfinished ...>

[pid 65410] <... write resumed>) = 63

[pid 65411] <... openat resumed>) = 3

[pid 65410] read(0, <unfinished ...>

[pid 65411] write(3, "\320\241\321\203\320\274\320\274\320\260: 75.30\n", 18) = 18

[pid 65411] close(3) = 0

[pid 65410] <... read resumed>"\n", 1024) = 1

[pid 65410] 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 65410] read(0, exit

"exit\n", 1024) = 5

[pid 65410] shmdt(0x7e764ace6000 <unfinished ...>

[pid 65411] shmdt(0x7e764ace6000 <unfinished ...>

[pid 65410] <... shmdt resumed>) = 0

[pid 65410] shmctl(32789, IPC\_RMID, NULL <unfinished ...>

[pid 65411] <... shmdt resumed>) = 0

[pid 65410] <... shmctl resumed>) = 0

[pid 65411] exit\_group(0 <unfinished ...>

[pid 65410] wait4(-1, <unfinished ...>

[pid 65411] <... exit\_group resumed>) = ?

[pid 65411] +++ exited with 0 +++

<... wait4 resumed>NULL, 0, NULL) = 65411

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=65411, si\_uid=1000, si\_status=0, si\_utime=1441 /\* 14.41 s \*/, si\_stime=0} ---

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.**