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-209БВ-24

Студент: Котик М. Н.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 20.11.24

Москва, 2024

Постановка задачи

Вариант 11.

Реализовать цепочку процессов: Parent → Child1 → Child2 → Parent с использованием механизма разделяемой памяти (shared memory) и семафоров для синхронизации. Каждый процесс выполняет преобразование данных:

Child1: преобразует все символы в верхний регистр

Child2: заменяет пробелы на символ '_'

Программа должна принимать строки от пользователя, передавать их по цепочке и выводить результат.

Общий метод и алгоритм решения

В данной лабораторной работе используется модель взаимодействия процессов через разделяемую память (memory-mapped files) вместо каналов (pipes), что позволяет более эффективно обмениваться данными между процессами. Для синхронизации используются именованные семафоры.

Использованные системные вызовы:

fork() / clone() – создание дочерних процессов

openat() – открытие файлов разделяемой памяти в /dev/shm/

ftruncate() – установка размера файла разделяемой памяти

mmap() с флагом MAP_SHARED – отображение файла в адресное пространство процесса

sem_open() / futex() – работа с семафорами для синхронизации

waitpid() – ожидание завершения дочерних процессов

unlink() – удаление файлов разделяемой памяти и семафоров

Алгоритм работы:

Родительский процесс (Parent) создает три файла разделяемой памяти в /dev/shm/ и три семафора для синхронизации

Создаются два дочерних процесса:

Child1: читает данные из shm_parent_child1, преобразует в верхний регистр, записывает в shm_child1_child2

Child2: читает данные из shm_child1_child2, заменяет пробелы на '_', записывает в shm_child2_parent

Родительский процесс принимает ввод от пользователя, записывает в разделяемую память и активирует Child1 через семафор

Данные последовательно проходят через Child1 и Child2

Родительский процесс получает результат из разделяемой памяти и выводит его

При завершении все ресурсы освобождаются

Код программы

parent.c

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

#define BUFFER_SIZE 1024
#define SHM_SIZE 4096

typedef struct
{
    char data[BUFFER_SIZE];
    int exit_flag;
} shared_data;

void print_error(const char *msg)
{
    perror(msg);
    exit(EXIT_FAILURE);
}

int main()
{
    printf("=== Lab 3. Variant 11 ===\n");
    printf("Chain: Parent -> Child1 -> Child2 -> Parent (using Memory-Mapped Files)\n\n");

    int shm_fd1 = shm_open("/shm_parent_child1", O_CREAT | O_RDWR, 0666);
    int shm_fd2 = shm_open("/shm_child1_child2", O_CREAT | O_RDWR, 0666);
    int shm_fd3 = shm_open("/shm_child2_parent", O_CREAT | O_RDWR, 0666);

    if (shm_fd1 == -1 || shm_fd2 == -1 || shm_fd3 == -1)
    {
        print_error("shm_open");
    }

    if (ftruncate(shm_fd1, SHM_SIZE) == -1 ||
        ftruncate(shm_fd2, SHM_SIZE) == -1 ||
        ftruncate(shm_fd3, SHM_SIZE) == -1)
```

```
    {
        print_error("ftruncate");
    }

    shared_data *shm1 = mmap(NULL, SHM_SIZE, PROT_READ | PROT_WRITE,
MAP_SHARED, shm_fd1, 0);
    shared_data *shm2 = mmap(NULL, SHM_SIZE, PROT_READ | PROT_WRITE,
MAP_SHARED, shm_fd2, 0);
    shared_data *shm3 = mmap(NULL, SHM_SIZE, PROT_READ | PROT_WRITE,
MAP_SHARED, shm_fd3, 0);

    if (shm1 == MAP_FAILED || shm2 == MAP_FAILED || shm3 == MAP_FAILED)
    {
        print_error("mmap");
    }

    sem_t *sem_parent_child1 = sem_open("/sem_pc1", O_CREAT, 0666, 0);
    sem_t *sem_child1_child2 = sem_open("/sem_c1c2", O_CREAT, 0666, 0);
    sem_t *sem_child2_parent = sem_open("/sem_c2p", O_CREAT, 0666, 0);

    if (sem_parent_child1 == SEM_FAILED || sem_child1_child2 == SEM_FAILED ||
sem_child2_parent == SEM_FAILED)
    {
        print_error("sem_open");
    }

    memset(shm1, 0, SHM_SIZE);
    memset(shm2, 0, SHM_SIZE);
    memset(shm3, 0, SHM_SIZE);
    shm1->exit_flag = 0;
    shm2->exit_flag = 0;
    shm3->exit_flag = 0;

    printf("=== Creating Child1 ===\n");
    pid_t pid1 = fork();
    if (pid1 == -1)
    {
        print_error("fork Child1");
    }

    if (pid1 == 0)
    {
        close(shm_fd1);
        close(shm_fd2);
        close(shm_fd3);

        execl("./child1", "child1", NULL);
        print_error("execl Child1");
    }

    printf("Child1 created with PID: %d\n", pid1);
```

```
printf("=== Creating Child2 ===\n");
pid_t pid2 = fork();
if (pid2 == -1)
{
    print_error("fork Child2");
}

if (pid2 == 0)
{
    close(shm_fd1);
    close(shm_fd2);
    close(shm_fd3);

    execl("./child2", "child2", NULL);
    print_error("execl Child2");
}

printf("Child2 created with PID: %d\n", pid2);

close(shm_fd1);
close(shm_fd2);
close(shm_fd3);

printf("\n=== Starting data processing ===\n");
printf("Enter strings for processing (type 'exit' to quit):\n");

char buffer[BUFFER_SIZE];
int line_count = 0;

while (1)
{
    printf("> ");
    fflush(stdout);

    if (fgets(buffer, BUFFER_SIZE, stdin) == NULL)
    {
        break;
    }

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

    if (strcmp(buffer, "exit") == 0)
    {
        shm1->exit_flag = 1;
        sem_post(sem_parent_child1);
        break;
    }

    line_count++;
    printf("[Parent] Sending string #%d to Child1: '%s'\n", line_count, buffer);
```

```
        strncpy(shm1->data, buffer, BUFFER_SIZE - 1);
        shm1->data[BUFFER_SIZE - 1] = '\0';
        sem_post(sem_parent_child1);

        sem_wait(sem_child2_parent);

        printf("[Parent] Final result from Child2: '%s'\n\n", shm3->data);
    }

    printf("\n=== Shutting down ===\n");
    printf("Waiting for children to exit...\n");

    int status1, status2;
    waitpid(pid1, &status1, 0);
    waitpid(pid2, &status2, 0);

    munmap(shm1, SHM_SIZE);
    munmap(shm2, SHM_SIZE);
    munmap(shm3, SHM_SIZE);
    shm_unlink("/shm_parent_child1");
    shm_unlink("/shm_child1_child2");
    shm_unlink("/shm_child2_parent");

    sem_close(sem_parent_child1);
    sem_close(sem_child1_child2);
    sem_close(sem_child2_parent);
    sem_unlink("/sem_pc1");
    sem_unlink("/sem_c1c2");
    sem_unlink("/sem_c2p");

    printf("Child processes exited with codes: Child1=%d, Child2=%d\n",
           WEXITSTATUS(status1), WEXITSTATUS(status2));

    printf("Program finished successfully.\n");
    return 0;
}
```

child1.c

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

```
#define BUFFER_SIZE 1024
#define SHM_SIZE 4096

typedef struct
{
    char data[BUFFER_SIZE];
    int exit_flag;
} shared_data;

void print_error(const char *msg)
{
    perror(msg);
    exit(EXIT_FAILURE);
}

int main()
{
    int shm_fd1 = shm_open("/shm_parent_child1", O_RDWR, 0666);
    int shm_fd2 = shm_open("/shm_child1_child2", O_RDWR, 0666);

    if (shm_fd1 == -1 || shm_fd2 == -1)
    {
        print_error("shm_open");
    }

    shared_data *shm1 = mmap(NULL, SHM_SIZE, PROT_READ | PROT_WRITE,
MAP_SHARED, shm_fd1, 0);
    shared_data *shm2 = mmap(NULL, SHM_SIZE, PROT_READ | PROT_WRITE,
MAP_SHARED, shm_fd2, 0);

    if (shm1 == MAP_FAILED || shm2 == MAP_FAILED)
    {
        print_error("mmap");
    }

    sem_t *sem_parent_child1 = sem_open("/sem_pc1", 0);
    sem_t *sem_child1_child2 = sem_open("/sem_c1c2", 0);

    if (sem_parent_child1 == SEM_FAILED || sem_child1_child2 == SEM_FAILED)
    {
        print_error("sem_open");
    }

    close(shm_fd1);
    close(shm_fd2);

    while (1)
    {
        sem_wait(sem_parent_child1);

        if (shm1->exit_flag)
```

```
        {
            shm2->exit_flag = 1;
            sem_post(sem_child1_child2);
            break;
        }

        for (int i = 0; shm1->data[i] != '\0'; i++)
        {
            shm1->data[i] = toupper(shm1->data[i]);
        }

        strncpy(shm2->data, shm1->data, BUFFER_SIZE - 1);
        shm2->data[BUFFER_SIZE - 1] = '\0';
        sem_post(sem_child1_child2);
    }

    munmap(shm1, SHM_SIZE);
    munmap(shm2, SHM_SIZE);
    sem_close(sem_parent_child1);
    sem_close(sem_child1_child2);

    return 0;
}
```

child2.c

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

#define BUFFER_SIZE 1024
#define SHM_SIZE 4096

typedef struct
{
    char data[BUFFER_SIZE];
    int exit_flag;
} shared_data;

void print_error(const char *msg)
{
    perror(msg);
    exit(EXIT_FAILURE);
}
```

```
int main()
{
    int shm_fd2 = shm_open("/shm_child1_child2", O_RDWR, 0666);
    int shm_fd3 = shm_open("/shm_child2_parent", O_RDWR, 0666);

    if (shm_fd2 == -1 || shm_fd3 == -1)
    {
        print_error("shm_open");
    }

    shared_data *shm2 = mmap(NULL, SHM_SIZE, PROT_READ | PROT_WRITE,
MAP_SHARED, shm_fd2, 0);
    shared_data *shm3 = mmap(NULL, SHM_SIZE, PROT_READ | PROT_WRITE,
MAP_SHARED, shm_fd3, 0);

    if (shm2 == MAP_FAILED || shm3 == MAP_FAILED)
    {
        print_error("mmap");
    }

    sem_t *sem_child1_child2 = sem_open("/sem_c1c2", 0);
    sem_t *sem_child2_parent = sem_open("/sem_c2p", 0);

    if (sem_child1_child2 == SEM_FAILED || sem_child2_parent == SEM_FAILED)
    {
        print_error("sem_open");
    }

    close(shm_fd2);
    close(shm_fd3);

    while (1)
    {
        sem_wait(sem_child1_child2);

        if (shm2->exit_flag)
        {
            shm3->exit_flag = 1;
            sem_post(sem_child2_parent);
            break;
        }

        for (int i = 0; shm2->data[i] != '\0'; i++)
        {
            if (isspace(shm2->data[i]))
            {
                shm2->data[i] = '_';
            }
        }

        strncpy(shm3->data, shm2->data, BUFFER_SIZE - 1);
```

```
        shm3->data[BUFFER_SIZE - 1] = '\0';
        sem_post(sem_child2_parent);
    }

    munmap(shm2, SHM_SIZE);
    munmap(shm3, SHM_SIZE);
    sem_close(sem_child1_child2);
    sem_close(sem_child2_parent);

    return 0;
}
```

Протокол работы программы

Тестирование:

Компиляция

gcc -o parent parent.c -lrt -lpthread

gcc -o child1 child1.c -lrt -lpthread

gcc -o child2 child2.c -lrt -lpthread

Запуск

./parent

Вывод программы:

=== Lab 3. Variant 11 ===

Chain: Parent -> Child1 -> Child2 -> Parent (using Memory-Mapped Files)

=== Creating Child1 ===

Child1 created with PID: 10067

=== Creating Child2 ===

Child2 created with PID: 10068

=== Starting data processing ===

Enter strings for processing (type 'exit' to quit):

> mew hello world

[Parent] Sending string #1 to Child1: 'mew hello world'

[Parent] Final result from Child2: 'MEW_HELLO_WORLD'

> test message

[Parent] Sending string #2 to Child1: 'test message'

[Parent] Final result from Child2: 'TEST_MESSAGE'

> exit

=== Shutting down ===

Waiting for children to exit...

Child processes exited with codes: Child1=0, Child2=0

Program finished successfully.

Strace:

root@LAPTOP-TGRFAHQT:/mnt/c/Users/mkoti/os_laba1/lab3# strace -f ./parent

execve("./parent", ["./parent"], 0x7fff1d3424b8 /* 28 vars */) = 0

brk(NULL) = 0x5641281b3000

arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe9353d1d0) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f9e6c486000

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=20892, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 20892, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f9e6c480000

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"..., 48, 848) = 48

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O{\f\225\\=\201\327\312\301P\32$\230\266\235"..., 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) = 0x7f9e6c257000

mprotect(0x7f9e6c27f000, 2023424, PROT_NONE) = 0

mmap(0x7f9e6c27f000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f9e6c27f000

mmap(0x7f9e6c414000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7f9e6c414000

mmap(0x7f9e6c46d000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x7f9e6c46d000

mmap(0x7f9e6c473000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f9e6c473000

close(3) = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f9e6c254000

arch_prctl(ARCH_SET_FS, 0x7f9e6c254740) = 0

set_tid_address(0x7f9e6c254a10) = 10066

set_robust_list(0x7f9e6c254a20, 24) = 0

rseq(0x7f9e6c2550e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7f9e6c46d000, 16384, PROT_READ) = 0

mprotect(0x56411ab46000, 4096, PROT_READ) = 0

mprotect(0x7f9e6c4c0000, 8192, PROT_READ) = 0

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

munmap(0x7f9e6c480000, 20892) = 0

newfstatat(1, "", {st_mode=S_IFCHR|0600, st_rdev=makedev(0x88, 0x2), ...}, AT_EMPTY_PATH) = 0

getrandom("\xcc\x91\x4d\x0b\x15\x78\xb5\x28", 8, GRND_NONBLOCK) = 8

brk(NULL) = 0x5641281b3000

brk(0x5641281d4000) = 0x5641281d4000

write(1, "=== Lab 3. Variant 11 ===\n", 26=== Lab 3. Variant 11 ===

) = 26

write(1, "Chain: Parent -> Child1 -> Child"..., 72Chain: Parent -> Child1 -> Child2 -> Parent (using Memory-Mapped Files)

) = 72

write(1, "\n", 1

) = 1

openat(AT_FDCWD, "/dev/shm/shm_parent_child1", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0666) = 3

openat(AT_FDCWD, "/dev/shm/shm_child1_child2", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0666) = 4

openat(AT_FDCWD, "/dev/shm/shm_child2_parent", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0666) = 5

ftruncate(3, 4096) = 0

ftruncate(4, 4096) = 0

ftruncate(5, 4096) = 0

mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f9e6c4bf000

mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x7f9e6c485000

mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 5, 0) = 0x7f9e6c484000

openat(AT_FDCWD, "/dev/shm/sem.sem_pc1", O_RDWR|O_NOFOLLOW) = -1 ENOENT (No such file or directory)

getrandom("\x92\xb5\xdb\xe8\x00\x1c\x97\x63", 8, GRND_NONBLOCK) = 8

newfstatat(AT_FDCWD, "/dev/shm/sem.W5J1Fi", 0x7ffe9353cac0, AT_SYMLINK_NOFOLLOW) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/dev/shm/sem.W5J1Fi", O_RDWR|O_CREAT|O_EXCL, 0666) = 6

write(6, "\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, 6, 0) = 0x7f9e6c483000

link("/dev/shm/sem.W5J1Fi", "/dev/shm/sem.sem_pc1") = 0

newfstatat(6, "", {st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH) = 0

unlink("/dev/shm/sem.W5J1Fi") = 0

close(6) = 0

openat(AT_FDCWD, "/dev/shm/sem.sem_c1c2", O_RDWR|O_NOFOLLOW) = -1 ENOENT (No such file or directory)

getrandom("\xe3\x64\x13\x42\x52\x54\xa9\x8a", 8, GRND_NONBLOCK) = 8

newfstatat(AT_FDCWD, "/dev/shm/sem.hAI5hD", 0x7ffe9353cac0, AT_SYMLINK_NOFOLLOW) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/dev/shm/sem.hAI5hD", O_RDWR|O_CREAT|O_EXCL, 0666) = 6

write(6, "\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, 6, 0) = 0x7f9e6c482000

link("/dev/shm/sem.hAI5hD", "/dev/shm/sem.sem_c1c2") = 0

newfstatat(6, "", {st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH) = 0

unlink("/dev/shm/sem.hAI5hD") = 0

close(6) = 0

openat(AT_FDCWD, "/dev/shm/sem.sem_c2p", O_RDWR|O_NOFOLLOW) = -1 ENOENT (No such file or directory)

getrandom("\x79\xc7\x8c\x0a\xbe\xa3\x3e\x19", 8, GRND_NONBLOCK) = 8

newfstatat(AT_FDCWD, "/dev/shm/sem.5QPMA8", 0x7ffe9353cac0, AT_SYMLINK_NOFOLLOW) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/dev/shm/sem.5QPMA8", O_RDWR|O_CREAT|O_EXCL, 0666) = 6

write(6, "\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, 6, 0) = 0x7f9e6c481000

link("/dev/shm/sem.5QPMA8", "/dev/shm/sem.sem_c2p") = 0

newfstatat(6, "", {st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH) = 0

unlink("/dev/shm/sem.5QPMA8") = 0

close(6) = 0

write(1, "=== Creating Child1 ===\n", 24=== Creating Child1 ===

) = 24

clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace: Process 10067 attached

, child_tidptr=0x7f9e6c254a10) = 10067

[pid 10067] set_robust_list(0x7f9e6c254a20, 24) = 0

[pid 10066] write(1, "Child1 created with PID: 10067\n", 31 <unfinished ...>

Child1 created with PID: 10067

[pid 10067] close(3 <unfinished ...>

[pid 10066] <... write resumed>) = 31

[pid 10067] <... close resumed>) = 0

[pid 10066] write(1, "=== Creating Child2 ===\n", 24 <unfinished ...>

=== Creating Child2 ===

[pid 10067] close(4 <unfinished ...>

[pid 10066] <... write resumed>) = 24

[pid 10067] <... close resumed>) = 0

[pid 10066] clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD <unfinished ...>

[pid 10067] close(5) = 0

[pid 10067] execve("./child1", ["child1"], 0x7ffe9353d3a8 /* 28 vars */strace: Process 10068 attached

<unfinished ...>

[pid 10066] <... clone resumed>, child_tidptr=0x7f9e6c254a10) = 10068

[pid 10068] set_robust_list(0x7f9e6c254a20, 24 <unfinished ...>

[pid 10066] write(1, "Child2 created with PID: 10068\n", 31 <unfinished ...>

Child2 created with PID: 10068

[pid 10068] <... set_robust_list resumed>) = 0

[pid 10066] <... write resumed>) = 31

[pid 10066] close(3) = 0

[pid 10068] close(3 <unfinished ...>

[pid 10066] close(4 <unfinished ...>

[pid 10068] <... close resumed>) = 0

[pid 10066] <... close resumed>) = 0

[pid 10068] close(4 <unfinished ...>

[pid 10066] close(5 <unfinished ...>

[pid 10068] <... close resumed>) = 0

[pid 10066] <... close resumed>) = 0

[pid 10068] close(5 <unfinished ...>

[pid 10066] write(1, "\n", 1 <unfinished ...>

[pid 10068] <... close resumed>) = 0

[pid 10066] <... write resumed>) = 1

[pid 10068] execve("./child2", ["child2"], 0x7ffe9353d3a8 /* 28 vars */ <unfinished ...>

[pid 10066] write(1, "=== Starting data processing ==="..., 33=== Starting data processing ===

) = 33

[pid 10066] write(1, "Enter strings for processing (ty"..., 52Enter strings for processing (type 'exit' to quit):

) = 52

[pid 10066] write(1, "> ", 2>) = 2

[pid 10066] newfstatat(0, "", {st_mode=S_IFCHR|0600, st_rdev=makedev(0x88, 0x2), ...}, AT_EMPTY_PATH) = 0

[pid 10066] read(0, <unfinished ...>

[pid 10067] <... execve resumed>) = 0

[pid 10067] brk(NULL) = 0x56441dcb8000

[pid 10067] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffed886b7c0) = -1 EINVAL (Invalid argument)

[pid 10068] <... execve resumed>) = 0

[pid 10067] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 10068] brk(NULL <unfinished ...>

[pid 10067] <... mmap resumed>) = 0x7f2dcb9cd000

[pid 10068] <... brk resumed>) = 0x55f66a612000

[pid 10067] access("/etc/ld.so.preload", R_OK <unfinished ...>

[pid 10068] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffd8bc21010 <unfinished ...>

[pid 10067] <... access resumed>) = -1 ENOENT (No such file or directory)

[pid 10068] <... arch_prctl resumed>) = -1 EINVAL (Invalid argument)

[pid 10067] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>

[pid 10068] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 10067] <... openat resumed>) = 3

[pid 10068] <... mmap resumed>) = 0x7fd0f152d000

[pid 10067] newfstatat(3, "", <unfinished ...>

[pid 10068] access("/etc/ld.so.preload", R_OK <unfinished ...>

[pid 10067] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=20892, ...}, AT_EMPTY_PATH) = 0

[pid 10068] <... access resumed>) = -1 ENOENT (No such file or directory)

[pid 10067] mmap(NULL, 20892, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>

[pid 10068] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>

[pid 10067] <... mmap resumed>) = 0x7f2dcb9c7000

[pid 10068] <... openat resumed>) = 3

[pid 10067] close(3 <unfinished ...>

[pid 10068] newfstatat(3, "", <unfinished ...>

[pid 10067] <... close resumed>) = 0

[pid 10068] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=20892, ...}, AT_EMPTY_PATH) = 0

[pid 10067] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC <unfinished ...>

[pid 10068] mmap(NULL, 20892, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>

[pid 10067] <... openat resumed>) = 3

[pid 10068] <... mmap resumed>) = 0x7fd0f1527000

[pid 10067] read(3, <unfinished ...>

[pid 10068] close(3 <unfinished ...>

[pid 10067] <... read resumed>"\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

[pid 10068] <... close resumed>) = 0

[pid 10067] pread64(3, <unfinished ...>

[pid 10068] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC <unfinished ...>

[pid 10067] <... pread64 resumed>"\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 10068] <... openat resumed>) = 3

[pid 10067] pread64(3, <unfinished ...>

[pid 10068] read(3, <unfinished ...>

[pid 10067] <... pread64 resumed>"\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 10068] <... read resumed>"\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

[pid 10067] pread64(3, <unfinished ...>

[pid 10068] pread64(3, <unfinished ...>

[pid 10067] <... pread64 resumed>"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O{\f\225\\=\201\327\312\301P\32$\230\266\235"..., 68, 896) = 68

[pid 10068] <... pread64 resumed>"\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 10067] newfstatat(3, "", <unfinished ...>

[pid 10068] pread64(3, <unfinished ...>

[pid 10067] <... newfstatat resumed>{st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0

[pid 10068] <... pread64 resumed>"\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 10067] pread64(3, <unfinished ...>

[pid 10068] pread64(3, <unfinished ...>

[pid 10067] <... pread64 resumed>"\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 10068] <... pread64 resumed>"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O{\f\225\\=\201\327\312\301P\32$\230\266\235"..., 68, 896) = 68

[pid 10067] mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished ...>

[pid 10068] newfstatat(3, "", <unfinished ...>

[pid 10067] <... mmap resumed>) = 0x7f2dcb79e000

[pid 10068] <... newfstatat resumed>{st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0

[pid 10067] mprotect(0x7f2dcb7c6000, 2023424, PROT_NONE <unfinished ...>

[pid 10068] pread64(3, <unfinished ...>

[pid 10067] <... mprotect resumed>) = 0

[pid 10068] <... pread64 resumed>"\6\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 10067] mmap(0x7f2dcb7c6000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000 <unfinished ...>

[pid 10068] mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished ...>

[pid 10067] <... mmap resumed>) = 0x7f2dcb7c6000

[pid 10068] <... mmap resumed>) = 0x7fd0f12fe000

[pid 10067] mmap(0x7f2dcb95b000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000 <unfinished ...>

[pid 10068] mprotect(0x7fd0f1326000, 2023424, PROT_NONE <unfinished ...>

[pid 10067] <... mmap resumed>) = 0x7f2dcb95b000

[pid 10068] <... mprotect resumed>) = 0

[pid 10067] mmap(0x7f2dcb9b4000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000 <unfinished ...>

[pid 10068] mmap(0x7fd0f1326000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000 <unfinished ...>

[pid 10067] <... mmap resumed>) = 0x7f2dcb9b4000

[pid 10068] <... mmap resumed>) = 0x7fd0f1326000

[pid 10067] mmap(0x7f2dcb9ba000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 10068] mmap(0x7fd0f14bb000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000 <unfinished ...>

[pid 10067] <... mmap resumed>) = 0x7f2dcb9ba000

[pid 10068] <... mmap resumed>) = 0x7fd0f14bb000

[pid 10068] mmap(0x7fd0f1514000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000 <unfinished ...>

[pid 10067] close(3 <unfinished ...>

[pid 10068] <... mmap resumed>) = 0x7fd0f1514000

[pid 10067] <... close resumed>) = 0

[pid 10068] mmap(0x7fd0f151a000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 10067] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 10068] <... mmap resumed>) = 0x7fd0f151a000

[pid 10067] <... mmap resumed>) = 0x7f2dcb79b000

[pid 10068] close(3 <unfinished ...>

[pid 10067] arch_prctl(ARCH_SET_FS, 0x7f2dcb79b740 <unfinished ...>

[pid 10068] <... close resumed>) = 0

[pid 10067] <... arch_prctl resumed>) = 0

[pid 10068] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid 10067] set_tid_address(0x7f2dcb79ba10 <unfinished ...>

[pid 10068] <... mmap resumed>) = 0x7fd0f12fb000

[pid 10067] <... set_tid_address resumed>) = 10067

[pid 10068] arch_prctl(ARCH_SET_FS, 0x7fd0f12fb740 <unfinished ...>

[pid 10067] set_robust_list(0x7f2dcb79ba20, 24 <unfinished ...>

[pid 10068] <... arch_prctl resumed>) = 0

[pid 10067] <... set_robust_list resumed>) = 0

[pid 10068] set_tid_address(0x7fd0f12fba10 <unfinished ...>

[pid 10067] rseq(0x7f2dcb79c0e0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 10068] <... set_tid_address resumed>) = 10068

[pid 10067] <... rseq resumed>) = 0

[pid 10068] set_robust_list(0x7fd0f12fba20, 24) = 0

[pid 10067] mprotect(0x7f2dcb9b4000, 16384, PROT_READ <unfinished ...>

[pid 10068] rseq(0x7fd0f12fc0e0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 10067] <... mprotect resumed>) = 0

[pid 10068] <... rseq resumed>) = 0

[pid 10067] mprotect(0x564404b43000, 4096, PROT_READ) = 0

[pid 10068] mprotect(0x7fd0f1514000, 16384, PROT_READ <unfinished ...>

[pid 10067] mprotect(0x7f2dcba07000, 8192, PROT_READ <unfinished ...>

[pid 10068] <... mprotect resumed>) = 0

[pid 10067] <... mprotect resumed>) = 0

[pid 10068] mprotect(0x55f63fa0e000, 4096, PROT_READ <unfinished ...>

[pid 10067] prlimit64(0, RLIMIT_STACK, NULL, <unfinished ...>

[pid 10068] <... mprotect resumed>) = 0

[pid 10067] <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

[pid 10068] mprotect(0x7fd0f1567000, 8192, PROT_READ <unfinished ...>

[pid 10067] munmap(0x7f2dcb9c7000, 20892 <unfinished ...>

[pid 10068] <... mprotect resumed>) = 0

[pid 10067] <... munmap resumed>) = 0

[pid 10068] prlimit64(0, RLIMIT_STACK, NULL, <unfinished ...>

[pid 10067] openat(AT_FDCWD, "/dev/shm/shm_parent_child1", O_RDWR|O_NOFOLLOW|O_CLOEXEC <unfinished ...>

[pid 10068] <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

[pid 10067] <... openat resumed>) = 3

[pid 10068] munmap(0x7fd0f1527000, 20892 <unfinished ...>

[pid 10067] openat(AT_FDCWD, "/dev/shm/shm_child1_child2", O_RDWR|O_NOFOLLOW|O_CLOEXEC <unfinished ...>

[pid 10068] <... munmap resumed>) = 0

[pid 10067] <... openat resumed>) = 4

[pid 10068] openat(AT_FDCWD, "/dev/shm/shm_child1_child2", O_RDWR|O_NOFOLLOW|O_CLOEXEC <unfinished ...>

[pid 10067] mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0 <unfinished ...>

[pid 10068] <... openat resumed>) = 3

[pid 10067] <... mmap resumed>) = 0x7f2dcba06000

[pid 10068] openat(AT_FDCWD, "/dev/shm/shm_child2_parent", O_RDWR|O_NOFOLLOW|O_CLOEXEC <unfinished ...>

[pid 10067] mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0 <unfinished ...>

[pid 10068] <... openat resumed>) = 4

[pid 10067] <... mmap resumed>) = 0x7f2dcb9cc000

[pid 10068] mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0 <unfinished ...>

[pid 10067] openat(AT_FDCWD, "/dev/shm/sem.sem_pc1", O_RDWR|O_NOFOLLOW <unfinished ...>

[pid 10068] <... mmap resumed>) = 0x7fd0f1566000

[pid 10067] <... openat resumed>) = 5

[pid 10068] mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0 <unfinished ...>

[pid 10067] newfstatat(5, "", <unfinished ...>

[pid 10068] <... mmap resumed>) = 0x7fd0f152c000

[pid 10067] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH) = 0

[pid 10068] openat(AT_FDCWD, "/dev/shm/sem.sem_c1c2", O_RDWR|O_NOFOLLOW <unfinished ...>

[pid 10067] getrandom(<unfinished ...>

[pid 10068] <... openat resumed>) = 5

[pid 10068] newfstatat(5, "", <unfinished ...>

[pid 10067] <... getrandom resumed>"\xd1\xc2\x53\x2b\xa9\x13\x1b\xf3", 8, GRND_NONBLOCK) = 8

[pid 10068] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH) = 0

[pid 10067] brk(NULL <unfinished ...>

[pid 10068] getrandom(<unfinished ...>

[pid 10067] <... brk resumed>) = 0x56441dcb8000

[pid 10068] <... getrandom resumed>"\xf3\xb4\xae\x7f\xcf\x4a\xbe\xb4", 8, GRND_NONBLOCK) = 8

[pid 10067] brk(0x56441dcd9000 <unfinished ...>

[pid 10068] brk(NULL <unfinished ...>

[pid 10067] <... brk resumed>) = 0x56441dcd9000

[pid 10068] <... brk resumed>) = 0x55f66a612000

[pid 10067] mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 5, 0 <unfinished ...>

[pid 10068] brk(0x55f66a633000 <unfinished ...>

[pid 10067] <... mmap resumed>) = 0x7f2dcb9cb000

[pid 10068] <... brk resumed>) = 0x55f66a633000

[pid 10067] close(5 <unfinished ...>

[pid 10068] mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 5, 0) = 0x7fd0f152b000

[pid 10067] <... close resumed>) = 0

[pid 10068] close(5 <unfinished ...>

[pid 10067] openat(AT_FDCWD, "/dev/shm/sem.sem_c1c2", O_RDWR|O_NOFOLLOW <unfinished ...>

[pid 10068] <... close resumed>) = 0

[pid 10067] <... openat resumed>) = 5

[pid 10068] openat(AT_FDCWD, "/dev/shm/sem.sem_c2p", O_RDWR|O_NOFOLLOW <unfinished ...>

[pid 10067] newfstatat(5, "", <unfinished ...>

[pid 10068] <... openat resumed>) = 5

[pid 10067] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH) = 0

[pid 10068] newfstatat(5, "", <unfinished ...>

[pid 10067] mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 5, 0 <unfinished ...>

[pid 10068] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH) = 0

[pid 10067] <... mmap resumed>) = 0x7f2dcb9ca000

[pid 10068] mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 5, 0 <unfinished ...>

[pid 10067] close(5 <unfinished ...>

[pid 10068] <... mmap resumed>) = 0x7fd0f152a000

[pid 10067] <... close resumed>) = 0

[pid 10068] close(5 <unfinished ...>

[pid 10067] close(3 <unfinished ...>

[pid 10068] <... close resumed>) = 0

[pid 10067] <... close resumed>) = 0

[pid 10068] close(3 <unfinished ...>

[pid 10067] close(4 <unfinished ...>

[pid 10068] <... close resumed>) = 0

[pid 10067] <... close resumed>) = 0

[pid 10068] close(4 <unfinished ...>

[pid 10067] futex(0x7f2dcb9cb000,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>

[pid 10068] <... close resumed>) = 0

[pid 10068] futex(0x7fd0f152b000,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANYmew hello world

<unfinished ...>

[pid 10066] <... read resumed>"mew hello world\n", 1024) = 16

[pid 10066] write(1, "[Parent] Sending string #1 to Ch"..., 56[Parent] Sending string #1 to Child1: 'mew hello world'

) = 56

[pid 10066] futex(0x7f9e6c483000, FUTEX_WAKE, 1) = 1

[pid 10067] <... futex resumed>) = 0

[pid 10066] futex(0x7f9e6c481000,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>

[pid 10067] futex(0x7f2dcb9ca000, FUTEX_WAKE, 1 <unfinished ...>

[pid 10068] <... futex resumed>) = 0

[pid 10067] <... futex resumed>) = 1

[pid 10068] futex(0x7fd0f152a000, FUTEX_WAKE, 1 <unfinished ...>

[pid 10067] futex(0x7f2dcb9cb000,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>

[pid 10066] <... futex resumed>) = 0

[pid 10068] <... futex resumed>) = 1

[pid 10066] write(1, "[Parent] Final result from Child"..., 54 <unfinished ...>

[pid 10068] futex(0x7fd0f152b000,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANY[Parent] Final result from Child2:
'MEW_HELLO_WORLD'

<unfinished ...>

[pid 10066] <... write resumed>) = 54

[pid 10066] write(1, "> ", 2>) = 2

[pid 10066] read(0, mew mew

"mew mew\n", 1024) = 8

[pid 10066] write(1, "[Parent] Sending string #2 to Ch"..., 48[Parent] Sending string #2 to Child1: 'mew mew'

) = 48

[pid 10066] futex(0x7f9e6c483000, FUTEX_WAKE, 1) = 1

[pid 10067] <... futex resumed>) = 0

[pid 10066] futex(0x7f9e6c481000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>

[pid 10067] futex(0x7f2dcb9ca000, FUTEX_WAKE, 1 <unfinished ...>

[pid 10068] <... futex resumed>) = 0

[pid 10067] <... futex resumed>) = 1

[pid 10068] futex(0x7fd0f152a000, FUTEX_WAKE, 1 <unfinished ...>

[pid 10067] futex(0x7f2dcb9cb000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>

[pid 10066] <... futex resumed>) = 0

[pid 10068] <... futex resumed>) = 1

[pid 10066] write(1, "[Parent] Final result from Child"..., 46 <unfinished ...>

[pid 10068] futex(0x7fd0f152b000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY[Parent] Final result from Child2: 'MEW_MEW'

<unfinished ...>

[pid 10066] <... write resumed>) = 46

[pid 10066] write(1, "> ", 2>) = 2

[pid 10066] read(0, exit

"exit\n", 1024) = 5

[pid 10066] futex(0x7f9e6c483000, FUTEX_WAKE, 1) = 1

[pid 10067] <... futex resumed>) = 0

[pid 10066] write(1, "\n", 1 <unfinished ...>

[pid 10067] futex(0x7f2dcb9ca000, FUTEX_WAKE, 1

<unfinished ...>

[pid 10066] <... write resumed>) = 1

[pid 10067] <... futex resumed>) = 1

[pid 10066] write(1, "=== Shutting down ===\n", 22 <unfinished ...>

[pid 10068] <... futex resumed>) = 0

=== Shutting down ===

[pid 10066] <... write resumed>) = 22

[pid 10067] munmap(0x7f2dcba06000, 4096 <unfinished ...>

[pid 10066] write(1, "Waiting for children to exit...\n", 32 <unfinished ...>

Waiting for children to exit...

[pid 10068] munmap(0x7fd0f1566000, 4096 <unfinished ...>

[pid 10066] <... write resumed>) = 32

[pid 10067] <... munmap resumed>) = 0

[pid 10066] wait4(10067, <unfinished ...>

[pid 10068] <... munmap resumed>) = 0

[pid 10067] munmap(0x7f2dcb9cc000, 4096 <unfinished ...>

[pid 10068] munmap(0x7fd0f152c000, 4096 <unfinished ...>

[pid 10067] <... munmap resumed>) = 0

[pid 10068] <... munmap resumed>) = 0

[pid 10067] munmap(0x7f2dcb9cb000, 32 <unfinished ...>

[pid 10068] munmap(0x7fd0f152b000, 32 <unfinished ...>

[pid 10067] <... munmap resumed>) = 0

[pid 10068] <... munmap resumed>) = 0

[pid 10067] munmap(0x7f2dcb9ca000, 32 <unfinished ...>

[pid 10068] munmap(0x7fd0f152a000, 32 <unfinished ...>

[pid 10067] <... munmap resumed>) = 0

[pid 10068] <... munmap resumed>) = 0

[pid 10068] exit_group(0 <unfinished ...>

[pid 10067] exit_group(0 <unfinished ...>

[pid 10068] <... exit_group resumed>) = ?

[pid 10067] <... exit_group resumed>) = ?

[pid 10068] +++ exited with 0 +++

[pid 10067] +++ exited with 0 +++

<... wait4 resumed>[{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 10067

--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=10068, si_uid=0, si_status=0, si_utime=0, si_stime=0} ---

wait4(10068, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 10068

munmap(0x7f9e6c4bf000, 4096) = 0

munmap(0x7f9e6c485000, 4096) = 0

munmap(0x7f9e6c484000, 4096) = 0

unlink("/dev/shm/shm_parent_child1") = 0

unlink("/dev/shm/shm_child1_child2") = 0

unlink("/dev/shm/shm_child2_parent") = 0

munmap(0x7f9e6c483000, 32) = 0

munmap(0x7f9e6c482000, 32) = 0

munmap(0x7f9e6c481000, 32) = 0

unlink("/dev/shm/sem.sem_pc1") = 0

unlink("/dev/shm/sem.sem_c1c2") = 0

unlink("/dev/shm/sem.sem_c2p") = 0

write(1, "Child processes exited with code"..., 54Child processes exited with codes: Child1=0, Child2=0

) = 54

write(1, "Program finished successfully.\n", 31Program finished successfully.

) = 31

exit_group(0) = ?

+++ exited with 0 +++

root@LAPTOP-TGRFAHQT:/mnt/c/Users/mkoti/os_laba1/lab3#

1. СОЗДАНИЕ ПРОЦЕССОВ:

fork() / clone() - создание дочерних процессов Child1 и Child2

execve() - запуск программ child1 и child2 в дочерних процессах

2. РАЗДЕЛЯЕМАЯ ПАМЯТЬ:

openat() - создание файлов shared memory в /dev/shm/

ftruncate() - установка размера shared memory (4096 байт)

mmap() с флагом MAP_SHARED - отображение shared memory в адресное пространство

3. СИНХРОНИЗАЦИЯ:

futex(FUTEX_WAIT) - ожидание семафора (процесс спит)

futex(FUTEX_WAKE) - сигнал семафора (пробуждение процесса)

4. УПРАВЛЕНИЕ ПРОЦЕССАМИ:

wait4() / waitpid() - ожидание завершения дочерних процессов

exit_group() - завершение процесса

5. ОЧИСТКА РЕСУРСОВ:

munmap() - освобождение отображенной памяти

unlink() - удаление файлов shared memory и семафоров

6. ВВОД/ВЫВОД:

read() - чтение данных от пользователя

write() - вывод результатов на экран

= ?
+++ exited with 0 +++

Вывод

В ходе выполнения лабораторной работы №3 успешно реализована цепочка процессов с использованием разделяемой памяти и семафоров для синхронизации. Программа демонстрирует эффективный механизм межпроцессного взаимодействия (IPC), где данные передаются через memory-mapped файлы в директории /dev/shm/, а синхронизация обеспечивается именованными семафорами.

Основные сложности заключались в правильной настройке семафоров для избежания deadlock'ов и корректном освобождении ресурсов при завершении программы. Реализация показала, что shared memory предоставляет более быстрый способ обмена данными между процессами по сравнению с pipes из лабораторной работы №1.

Программа работает стабильно, корректно обрабатывает пользовательский ввод и освобождает все системные ресурсы при завершении.