Отчет
Пирогов Даниил Игоревич
БПИ-245

Разработаны две независимые программы - клиент и сервер, взаимодействующие через разделяемую память стандарта POSIX. Клиент генерирует случайные числа в диапазоне 0-999, сервер осуществляет их вывод.

Задачи клиента:

1. Создать/открыть объект разделяемой памяти shm_open()
2. Установить размер памяти ftruncate()
3. Отобразить память в адресное пространство mmap()
4. В цикле сгенерировать случайные числа и записать в разделяемую память
5. Обновить флаг своей активности

Сервер:

1. Открыть существующий объект разделяемой памяти
2. Отобразить память в свое адресное пространство
3. В цикле проверить обновления данных от клиента
4. Вывести полученные числа на экран
5. Следить за активностью клиента

Механизм корректного завершения реализован на основе обработки сигнала SIGINT (Ctrl+C):

1. При получении сигнала процессы устанавливают флаги завершения
2. Выполняется очистка ресурсов через зарегистрированные функции atexit()
3. Клиент сбрасывает флаг своей активности в разделяемой памяти
4. Сервер удаляет объект разделяемой памяти при завершении
5. Оба процесса корректно освобождают память и закрывают дескрипторы

Исходный код

Компиляция и запуск

```
gcc -o shmem-client shmem-client.c -lrt
gcc -o shmem-server shmem-server.c -lrt
./shmem-server
./shmem-client
```

shmem-client.c

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

#define SHARED_OBJ_NAME "random-numbers-shmem"
#define MAX_RANDOM 1000

typedef struct {
   int number;
   int client_active;
   int server_active;
} shared_data_t;

shared_data_t *shared_data = NULL;
int shm_fd = -1;
volatile sig_atomic_t client_running = 1;

void client_sigint_handler(int sig) {
   printf("\nКлиент: Получен сигнал завершения\n");
   client_running = 0;
}

void cleanup() {
   if (shared_data != NULL) {
       shared_data->client_active = 0;
       munmap(shared_data, sizeof(shared_data_t));
   }
   if (shm_fd != -1) {
```

```
        close(shm_fd);
    }
    printf("Клиент: Завершение работы\n");
}

void setup_signal_handler() {
    struct sigaction sa;
    sa.sa_handler = client_sigint_handler;
    sigemptyset(&sa.sa_mask);
    sa.sa_flags = 0;

    if (sigaction(SIGINT, &sa, NULL) == -1) {
        perror("Ошибка настройки обработчика сигнала");
        exit(1);
    }
}

int main() {
    printf("Клиент: Запуск генератора случайных чисел\n");

    setup_signal_handler();
    atexit(cleanup);

    shm_fd = shm_open(SHARED_OBJ_NAME, O_CREAT | O_RDWR, 0666);
    if (shm_fd == -1) {
        perror("Клиент: Ошибка создания разделяемой памяти");
        exit(1);
    }

    if (ftruncate(shm_fd, sizeof(shared_data_t)) == -1) {
        perror("Клиент: Ошибка установки размера памяти");
        exit(1);
    }

    shared_data = mmap(NULL, sizeof(shared_data_t),
                       PROT_READ | PROT_WRITE, MAP_SHARED, shm_fd, 0);
    if (shared_data == MAP_FAILED) {
        perror("Клиент: Ошибка отображения памяти");
        exit(1);
    }

    shared_data->client_active = 1;
    shared_data->server_active = 0;
    shared_data->number = 0;

    printf("Клиент: Начало генерации чисел (0-%d)\n", MAX_RANDOM-1);
    printf("Клиент: Для завершения нажмите Ctrl+C\n");

    srand(time(NULL));

    while (client_running) {
        int random_num = rand() % MAX_RANDOM;
        shared_data->number = random_num;
        printf("Клиент: Сгенерировано число %d\n", random_num);
        sleep(1);
    }
```

```
    return 0;
}
```

shmem-server.c

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

#define SHARED_OBJ_NAME "random-numbers-shmem"
#define MAX_RANDOM 1000

typedef struct {
    int number;
    int client_active;
    int server_active;
} shared_data_t;

shared_data_t *shared_data = NULL;
int shm_fd = -1;
volatile sig_atomic_t server_running = 1;

void server_sigint_handler(int sig) {
    printf("\nСервер: Получен сигнал завершения\n");
    server_running = 0;
}

void cleanup() {
    if (shared_data != NULL) {
        shared_data->server_active = 0;
        munmap(shared_data, sizeof(shared_data_t));
    }
    if (shm_fd != -1) {
        close(shm_fd);
        shm_unlink(SHARED_OBJ_NAME);
        printf("Сервер: Разделяемая память удалена\n");
    }
    printf("Сервер: Завершение работы\n");
}
```

```
void setup_signal_handler() {
    struct sigaction sa;
    sa.sa_handler = server_sigint_handler;
    sigemptyset(&sa.sa_mask);
    sa.sa_flags = 0;

    if (sigaction(SIGINT, &sa, NULL) == -1) {
        perror("Ошибка настройки обработчика сигнала");
        exit(1);
    }
}

int main() {
    printf("Сервер: Запуск чтения случайных чисел\n");

    setup_signal_handler();
    atexit(cleanup);

    shm_fd = shm_open(SHARED_OBJ_NAME, O_RDWR, 0666);
    if (shm_fd == -1) {
        perror("Сервер: Ошибка открытия разделяемой памяти");
        exit(1);
    }

    shared_data = mmap(NULL, sizeof(shared_data_t),
                       PROT_READ | PROT_WRITE, MAP_SHARED, shm_fd, 0);
    if (shared_data == MAP_FAILED) {
        perror("Сервер: Ошибка отображения памяти");
        exit(1);
    }

    shared_data->server_active = 1;

    printf("Сервер: Ожидание чисел от клиента\n");
    printf("Сервер: Для завершения нажмите Ctrl+C\n");

    int last_number = -1;

    while (server_running) {
        if (shared_data->number != last_number) {
            last_number = shared_data->number;
            printf("Сервер: Получено число %d\n", last_number);
        }

        if (!shared_data->client_active) {
            printf("Сервер: Клиент завершил работу\n");
            break;
        }
```

```
        usleep(100000);
    }

    return 0;
}
```