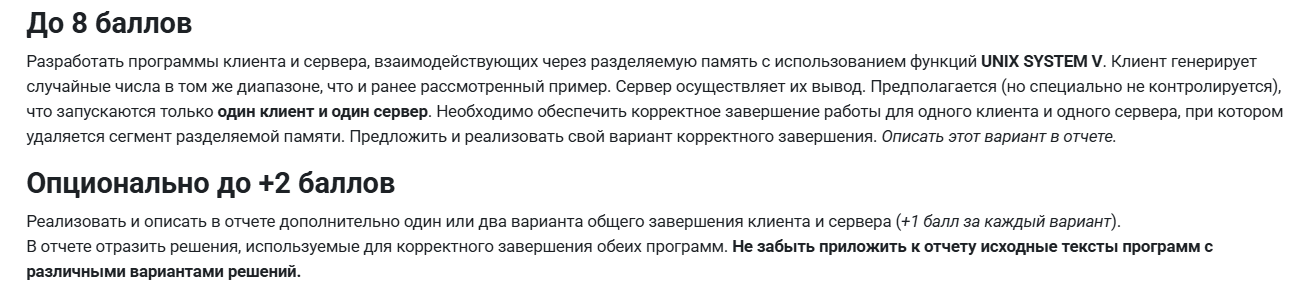
# Домашнее задание 6. Мусаев Умахан Рашидович.

Условие:



Работа выполнена с опциональной частью.

Отчет:

Основной вариант:

* server.cpp
* clienr.cpp

В этом варианте клиент и сервер используют разделяемую память для передачи данных и флага завершения. Клиент устанавливает флаг exit\_flag в разделяемой памяти при завершении, что позволяет серверу корректно завершить работу и удалить ресурсы.

server.cpp

#include <iostream>  
#include <cstdlib>  
#include <cstring>  
#include <unistd.h>  
#include <sys/ipc.h>  
#include <sys/shm.h>  
#include <sys/sem.h>  
#include <signal.h>  
  
#define SHM\_KEY 0x1234  
#define SEM\_KEY 0x5678  
  
struct SharedData {  
 int number;  
 int exit\_flag;  
};  
  
union semun {  
 int val;  
 struct semid\_ds \*buf;  
 unsigned short \*array;  
};  
  
int shm\_id;  
int sem\_id;  
SharedData \*shm\_ptr;  
  
void cleanup(int sig) {  
 std::cout << "\nCleaning up resources..." << std::endl;  
 shmdt(shm\_ptr);  
 shmctl(shm\_id, IPC\_RMID, nullptr);  
 semctl(sem\_id, 0, IPC\_RMID);  
 exit(EXIT\_SUCCESS);  
}  
  
void init\_semaphores(int sem\_id) {  
 union semun arg;  
 unsigned short values[2] = {1, 0};  
 arg.array = values;  
 if (semctl(sem\_id, 0, SETALL, arg) == -1) {  
 perror("semctl SETALL");  
 exit(EXIT\_FAILURE);  
 }  
}  
  
int main() {  
 shm\_id = shmget(SHM\_KEY, sizeof(SharedData), IPC\_CREAT | 0666);  
 if (shm\_id == -1) {  
 perror("shmget");  
 return EXIT\_FAILURE;  
 }  
  
 shm\_ptr = (SharedData\*)shmat(shm\_id, nullptr, 0);  
 if (shm\_ptr == (void\*)-1) {  
 perror("shmat");  
 shmctl(shm\_id, IPC\_RMID, nullptr);  
 return EXIT\_FAILURE;  
 }  
  
 shm\_ptr->exit\_flag = 0;  
  
 sem\_id = semget(SEM\_KEY, 2, IPC\_CREAT | 0666);  
 if (sem\_id == -1) {  
 perror("semget");  
 shmdt(shm\_ptr);  
 shmctl(shm\_id, IPC\_RMID, nullptr);  
 return EXIT\_FAILURE;  
 }  
  
 init\_semaphores(sem\_id);  
 signal(SIGINT, cleanup);  
  
 std::cout << "Server started. Waiting for client..." << std::endl;  
  
 while (true) {  
 struct sembuf sops = {1, -1, 0};  
 if (semop(sem\_id, &sops, 1) == -1) {  
 perror("semop server wait");  
 break;  
 }  
  
 if (shm\_ptr->exit\_flag) {  
 std::cout << "Client requested exit. Exiting..." << std::endl;  
 break;  
 }  
  
 std::cout << "Received number: " << shm\_ptr->number << std::endl;  
  
 sops = {0, 1, 0};  
 if (semop(sem\_id, &sops, 1) == -1) {  
 perror("semop client post");  
 break;  
 }  
 }  
  
 cleanup(0);  
 return EXIT\_SUCCESS;  
}

client.cpp

#include <iostream>  
#include <cstdlib>  
#include <cstring>  
#include <unistd.h>  
#include <sys/ipc.h>  
#include <sys/shm.h>  
#include <sys/sem.h>  
#include <signal.h>  
#include <ctime>  
  
#define SHM\_KEY 0x1234  
#define SEM\_KEY 0x5678  
  
struct SharedData {  
 int number;  
 int exit\_flag;  
};  
  
int shm\_id;  
int sem\_id;  
SharedData \*shm\_ptr;  
  
void cleanup(int sig) {  
 std::cout << "\nExiting client..." << std::endl;  
 shmdt(shm\_ptr);  
 exit(EXIT\_SUCCESS);  
}  
  
int main() {  
 shm\_id = shmget(SHM\_KEY, sizeof(SharedData), 0666);  
 if (shm\_id == -1) {  
 perror("shmget");  
 return EXIT\_FAILURE;  
 }  
  
 shm\_ptr = (SharedData\*)shmat(shm\_id, nullptr, 0);  
 if (shm\_ptr == (void\*)-1) {  
 perror("shmat");  
 return EXIT\_FAILURE;  
 }  
  
 sem\_id = semget(SEM\_KEY, 2, 0666);  
 if (sem\_id == -1) {  
 perror("semget");  
 shmdt(shm\_ptr);  
 return EXIT\_FAILURE;  
 }  
  
 signal(SIGINT, [](int sig) {  
 shm\_ptr->exit\_flag = 1;  
 struct sembuf sops = {1, 1, 0};  
 semop(sem\_id, &sops, 1);  
 cleanup(sig);  
 });  
  
 srand(time(nullptr));  
  
 while (true) {  
 int num = rand() % 100;  
  
 struct sembuf sops = {0, -1, 0};  
 if (semop(sem\_id, &sops, 1) == -1) {  
 perror("semop client wait");  
 break;  
 }  
  
 shm\_ptr->number = num;  
  
 sops = {1, 1, 0};  
 if (semop(sem\_id, &sops, 1) == -1) {  
 perror("semop server post");  
 break;  
 }  
  
 sleep(1);  
 }  
  
 shmdt(shm\_ptr);  
 return EXIT\_SUCCESS;  
}

Дополнительный вариант:

* client\_sigur1.cpp
* server\_sigusr1.cpp

В этом варианте клиент отправляет сигнал SIGUSR1 серверу при завершении. Сервер, получив сигнал, выполняет очистку ресурсов.

server\_sigusr1.cpp

#include <iostream>  
#include <cstdlib>  
#include <cstring>  
#include <unistd.h>  
#include <sys/ipc.h>  
#include <sys/shm.h>  
#include <sys/sem.h>  
#include <signal.h>  
  
#define SHM\_KEY 0x1234  
#define SEM\_KEY 0x5678  
  
struct SharedData {  
 int number;  
 int exit\_flag;  
 pid\_t server\_pid; // Добавлено поле для PID сервера  
};  
  
union semun {  
 int val;  
 struct semid\_ds \*buf;  
 unsigned short \*array;  
};  
  
int shm\_id;  
int sem\_id;  
SharedData \*shm\_ptr;  
  
void cleanup(int sig) {  
 std::cout << "\nCleaning up resources..." << std::endl;  
 shmdt(shm\_ptr);  
 shmctl(shm\_id, IPC\_RMID, nullptr);  
 semctl(sem\_id, 0, IPC\_RMID);  
 exit(EXIT\_SUCCESS);  
}  
  
void handle\_exit\_signal(int sig) {  
 std::cout << "\nReceived exit signal. Exiting..." << std::endl;  
 cleanup(sig);  
}  
  
void init\_semaphores(int sem\_id) {  
 union semun arg;  
 unsigned short values[2] = {1, 0};  
 arg.array = values;  
 if (semctl(sem\_id, 0, SETALL, arg) == -1) {  
 perror("semctl SETALL");  
 exit(EXIT\_FAILURE);  
 }  
}  
  
int main() {  
 shm\_id = shmget(SHM\_KEY, sizeof(SharedData), IPC\_CREAT | 0666);  
 if (shm\_id == -1) {  
 perror("shmget");  
 return EXIT\_FAILURE;  
 }  
  
 shm\_ptr = (SharedData\*)shmat(shm\_id, nullptr, 0);  
 if (shm\_ptr == (void\*)-1) {  
 perror("shmat");  
 shmctl(shm\_id, IPC\_RMID, nullptr);  
 return EXIT\_FAILURE;  
 }  
  
 shm\_ptr->exit\_flag = 0;  
 shm\_ptr->server\_pid = getpid(); // Записываем PID сервера  
  
 sem\_id = semget(SEM\_KEY, 2, IPC\_CREAT | 0666);  
 if (sem\_id == -1) {  
 perror("semget");  
 shmdt(shm\_ptr);  
 shmctl(shm\_id, IPC\_RMID, nullptr);  
 return EXIT\_FAILURE;  
 }  
  
 init\_semaphores(sem\_id);  
 signal(SIGINT, cleanup); // Обработчик Ctrl+C  
 signal(SIGUSR1, handle\_exit\_signal); // Обработчик SIGUSR1  
  
 std::cout << "Server started. PID: " << getpid() << ". Waiting for client..." << std::endl;  
  
 while (true) {  
 struct sembuf sops = {1, -1, 0};  
 if (semop(sem\_id, &sops, 1) == -1) {  
 perror("semop server wait");  
 break;  
 }  
  
 if (shm\_ptr->exit\_flag) {  
 std::cout << "Client requested exit. Exiting..." << std::endl;  
 break;  
 }  
  
 std::cout << "Received number: " << shm\_ptr->number << std::endl;  
  
 sops = {0, 1, 0};  
 if (semop(sem\_id, &sops, 1) == -1) {  
 perror("semop client post");  
 break;  
 }  
 }  
  
 cleanup(0);  
 return EXIT\_SUCCESS;  
}

client\_sigur1.cpp

#include <iostream>  
#include <cstdlib>  
#include <cstring>  
#include <unistd.h>  
#include <sys/ipc.h>  
#include <sys/shm.h>  
#include <sys/sem.h>  
#include <signal.h>  
#include <ctime>  
  
#define SHM\_KEY 0x1234  
#define SEM\_KEY 0x5678  
  
struct SharedData {  
 int number;  
 int exit\_flag;  
 pid\_t server\_pid; // Добавлено поле для PID сервера  
};  
  
int shm\_id;  
int sem\_id;  
SharedData \*shm\_ptr;  
  
void cleanup(int sig) {  
 std::cout << "\nExiting client..." << std::endl;  
 shmdt(shm\_ptr);  
 exit(EXIT\_SUCCESS);  
}  
  
int main() {  
 shm\_id = shmget(SHM\_KEY, sizeof(SharedData), 0666);  
 if (shm\_id == -1) {  
 perror("shmget");  
 return EXIT\_FAILURE;  
 }  
  
 shm\_ptr = (SharedData\*)shmat(shm\_id, nullptr, 0);  
 if (shm\_ptr == (void\*)-1) {  
 perror("shmat");  
 return EXIT\_FAILURE;  
 }  
  
 sem\_id = semget(SEM\_KEY, 2, 0666);  
 if (sem\_id == -1) {  
 perror("semget");  
 shmdt(shm\_ptr);  
 return EXIT\_FAILURE;  
 }  
  
 signal(SIGINT, [](int sig) {  
 pid\_t server\_pid = shm\_ptr->server\_pid;  
 std::cout << "\nSending SIGUSR1 to server (PID: " << server\_pid << ")..." << std::endl;  
 kill(server\_pid, SIGUSR1); // Отправляем SIGUSR1 серверу  
 cleanup(sig);  
 });  
  
 srand(time(nullptr));  
  
 while (true) {  
 int num = rand() % 100;  
  
 struct sembuf sops = {0, -1, 0};  
 if (semop(sem\_id, &sops, 1) == -1) {  
 perror("semop client wait");  
 break;  
 }  
  
 shm\_ptr->number = num;  
  
 sops = {1, 1, 0};  
 if (semop(sem\_id, &sops, 1) == -1) {  
 perror("semop server post");  
 break;  
 }  
  
 sleep(1);  
 }  
  
 shmdt(shm\_ptr);  
 return EXIT\_SUCCESS;  
}

Запуск программы:

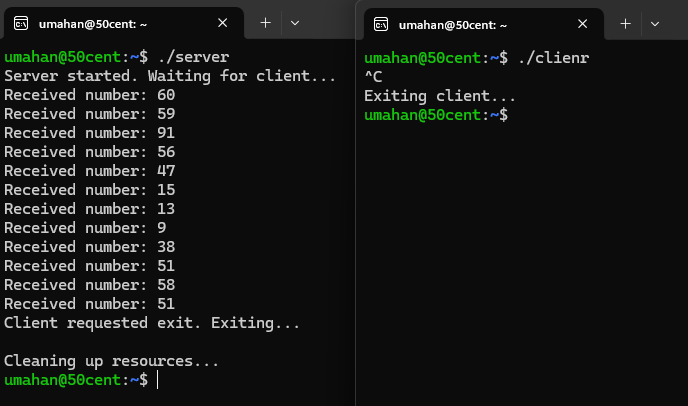
1. Сборка программы:

* g++ server.cpp -o server
* g++ clienr.cpp -o clienr
* g++ server\_sigusr1.cpp -o server\_sigusr1
* g++ client\_sigur1.cpp -o client\_sigur1

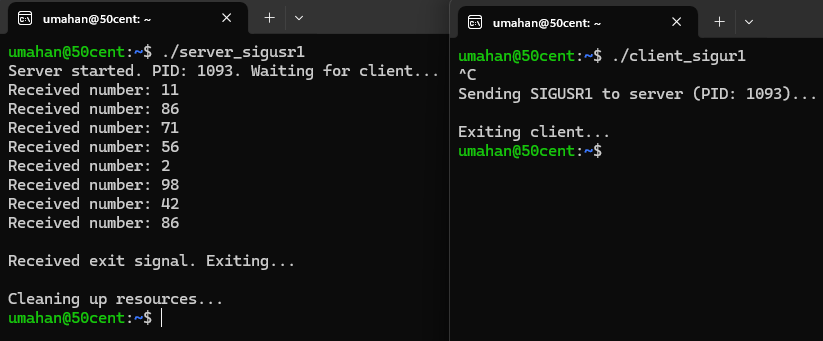
1. Запуск программы(нужно запускать на разных консолях)
   1. ./server или ./server\_sigusr1
   2. ./clienr ./client\_sigur1
2. Для остановки программы, в консоли клиента нужно прописать Ctrl+C.

Примеры работы программы:

Основной вариант:



Дополнительный вариант:



В архиве также содержится видеопример работы программы.