

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА  
(САМАРСКИЙ УНИВЕРСИТЕТ)»

Институт Информатики и кибернетики   
Кафедра Программных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**  
  
к лабораторной работе №2 по дисциплине «Технологии промышленного программирования»

Обучающийся группы 6232-020302D \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Портнов К.А.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Баландин А.В.

Самара 2023

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc147860313)

[2 Результаты работы 4](#_Toc147860314)

[ПРИЛОЖЕНИЕ А Листинг модуля M1 5](#_Toc147860315)

[ПРИЛОЖЕНИЕ B Листинг модуля M2 7](#_Toc147860316)

[ПРИЛОЖЕНИЕ С Листинг модуля M3 9](#_Toc147860317)

1. Постановка задачи

Разработать приложение, состоящее из трех взаимодействующих процессов. Требуется написать три программных модуля – М1, М2, М3. На базе модуля М1 из shell запускается стартовый процесс Р1(М1).

Процесс Р1 создает канал и, используя функцию семейства spawn\*(), запускает процесс Р2(М2), передавая ему в качестве параметра chid созданного канала, затем переходит в состояние приема сообщений по созданному каналу.

Процесс Р2 создает свой канал, и, используя функцию семейства spawn\*(), запускает процесс Р3(М3), передавая ему в качестве параметра chid созданного канала, затем устанавливает соединение с каналом процесса Р1 и переходит в состояние приема сообщений по своему каналу.

Процесс Р3 устанавливает соединение с каналом процесса Р2 и посылает ему запрос на получение pid процесса Р1 и chid его канала. Получив от Р2 ответ (pid и chid), устанавливает соединение с каналом процесса Р1 и посылает ему сообщение "Р3 send message to P1". После получения ответа посылает процессу Р2 сообщение "Р3 stop", затем, получив ответ, выводит на экран "Р3 ОК" и терминируется.

Процесс Р1, получив сообщение от процесса Р3, выдает его на терминал, затем посылает ответ и переходит в ожидание сообщений по своему каналу.

Процесс Р2, получив сообщение от процесса Р3, выводит его на экран и посылает ответ. Затем посылает процессу Р1 сообщение "Р2 stop", получает ответ, выводит на экран "Р2 ОК" и терминируется.

Процесс Р1, получив сообщение от процесса Р2 выдает его на терминал, посылает ответ, выводит на экран "Р1 ОК" и терминируется.

1. Результаты работы

Результаты работы представлены в виде вывода на консоль сообщений во время выполнения программы.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 1 – Результат выполнения программы в консоли

ПРИЛОЖЕНИЕ А  
Листинг модуля M1

**#include** <cstdlib>

**#include** <iostream>

**#include** <stdlib.h>

**#include** <sys/neutrino.h>

**#include** <unistd.h>

**#include** <process.h>

**#include** <string.h>

**#include** <unistd.h>

**#define** GetCurrentDir getcwd

**using** std::cout;

**using** std::endl;

// Отображение текущей директории

**void** **viewWorkDirectory**();

**int** **main**(**int** argc, **char** \*argv[]) {

cout << "P1: Запущен" << endl;

viewWorkDirectory();

**int** chid = **ChannelCreate**(\_NTO\_CHF\_SENDER\_LEN);

cout << "P1: Канал создан: chid = " << chid << endl;

**char** buffer[20];

**const** **char** \*chidStr = **itoa**(chid, buffer, 10);

**int** pidP2 = **spawnl**(P\_NOWAIT, "/home/host/lab1p2/x86/o/lab1p2", chidStr, NULL);

**if** (pidP2 < 0){

cout << "P1: Ошибка запуска процесса P2" << pidP2 << endl;

exit(EXIT\_FAILURE);

}

cout << "P1: pid процесса P2 - " << pidP2 << endl;

cout << "P1: Старт получения сообщений" << endl;

**int** countMsg = 0;

**while**(countMsg < 2){

**char** msg[200] = {};

\_msg\_info info;

**int** rcvid;

rcvid = **MsgReceive**(chid, msg, **sizeof**(msg), &info);

**if**(rcvid == -1){

cout << "P1: Ошибка MsgReceive" << endl;

}

**if** (info.pid == pidP2){

cout << "P1: Получено сообщение от P2 - " << msg << endl;

**MsgReply**(rcvid, NULL, msg, **sizeof**(msg));

} **else**{

cout << "P1: Получено сообщение от " << info.pid << " - " << msg << endl;

**MsgReply**(rcvid, NULL, msg, **sizeof**(msg));

}

countMsg++;

}

cout << "P1: Р1 ОК" << endl;

**return** EXIT\_SUCCESS;

}

// Отображение текущей директории

**void** **viewWorkDirectory**(){

**char** cCurrentPath[FILENAME\_MAX];

**if** (!GetCurrentDir(cCurrentPath, **sizeof**(cCurrentPath)))

exit(EXIT\_FAILURE);

cCurrentPath[**sizeof**(cCurrentPath) - 1] = '\0';

cout << "P1: текущая рабочая директория - " << cCurrentPath << endl;

}

ПРИЛОЖЕНИЕ B  
Листинг модуля M2

**#include** <cstdlib>

**#include** <iostream>

**#include** <stdlib.h>

**#include** <sys/neutrino.h>

**#include** <unistd.h>

**#include** <process.h>

**#include** <string.h>

**#include** <unistd.h>

**#define** GetCurrentDir getcwd

**using** std::cout;

**using** std::endl;

// Отображение текущей директории

**void** **viewWorkDirectory**();

**int** **main**(**int** argc, **char** \*argv[]) {

cout << "P2: Запущен" << endl;

viewWorkDirectory();

cout << "P2: Параметры: " << "argv[0]= " << argv[0] << endl;

**int** chid = **ChannelCreate**(\_NTO\_CHF\_SENDER\_LEN);

cout << "P2: Канал создан: chid = " << chid << endl;

**char** buffer[20];

**const** **char** \*chidStr = **itoa**(chid, buffer, 10);

**int** pidP3 = **spawnl**(P\_NOWAIT, "/home/host/lab1p3/x86/o/lab1p3", chidStr, NULL);

**if** (pidP3 < 0){

cout << "P2: Ошибка запуска процесса P3" << pidP3 << endl;

exit(EXIT\_FAILURE);

}

cout << "P2: pid процесса P3 - " << pidP3 << endl;

cout << "P2: Старт получения сообщений" << endl;

**int** countMsg = 0;

**while**(countMsg < 3){

**char** msg[200] = {};

\_msg\_info info;

**int** rcvid;

rcvid = **MsgReceive**(chid, msg, **sizeof**(msg), &info);

**if**(rcvid == -1){

cout << "P2: Ошибка MsgReceive" << endl;

}

**if** (info.pid == pidP3){

cout << "P2: Получено сообщение от P3 - " << msg << endl;

**if**(countMsg == 0){

**int** pidP1 = **getppid**();

**char** pidP1Buffer[50];

**char** **const** \*pidP3Str = **itoa**(pidP1, pidP1Buffer, 10);

MsgReply(rcvid, NULL, pidP3Str, strlen(pidP3Str));

}

**if**(countMsg == 1){

**char** \*pChidStr = argv[0];

**MsgReply**(rcvid, NULL, pChidStr, **sizeof**(pChidStr));

}

**if**(countMsg == 2){

**MsgReply**(rcvid, NULL, msg, **sizeof**(msg));

**int** pChid = atoi(argv[0]);

cout << "P2: установление соединения с каналом P1" << endl;

**int** coidP1 = **ConnectAttach**(0, **getppid**(), pChid, \_NTO\_SIDE\_CHANNEL, 0);

**if**(coidP1 == -1){

cout << "P2: Ошибка соединения с каналом P1" << endl;

exit(EXIT\_FAILURE);

}

**char** rmsg[200] = {};

cout << "Р2: Посылаю сообщение Р1" << endl;

**char** \*smsg = (**char** \*)"Р2 stop";

**if**(MsgSend(coidP1, smsg, strlen(smsg) + 1, rmsg, **sizeof**(rmsg)) == -1){

cout << "P2: Ошибка MsgSend при отправки в P1" << endl;

exit(EXIT\_FAILURE);

}

**if**(strlen(rmsg) > 0){

cout << "P2: Сервер P1 ответил" << endl;

}

}

countMsg ++;

}

}

cout << "P2: Р2 ОК" << endl;

**return** EXIT\_SUCCESS;

}

// Отображение текущей директории

**void** **viewWorkDirectory**(){

**char** cCurrentPath[FILENAME\_MAX];

**if** (!GetCurrentDir(cCurrentPath, **sizeof**(cCurrentPath)))

exit(EXIT\_FAILURE);

cCurrentPath[**sizeof**(cCurrentPath) - 1] = '\0';

cout << "P2: текущая рабочая директория - " << cCurrentPath << endl;

}

ПРИЛОЖЕНИЕ С  
Листинг модуля M3

**#include** <cstdlib>

**#include** <iostream>

**#include** <stdlib.h>

**#include** <sys/neutrino.h>

**#include** <unistd.h>

**#include** <process.h>

**#include** <string.h>

**#include** <unistd.h>

**#define** GetCurrentDir getcwd

**using** std::cout;

**using** std::endl;

// Отображение текущей директории

**void** **viewWorkDirectory**();

**int** **main**(**int** argc, **char** \*argv[]) {

cout << "P3: Запущен" << endl;

viewWorkDirectory();

cout << "P3: Параметры: " << "argv[0]= " << argv[0] << endl;

**int** pChid = atoi(argv[0]);

// Ответ от P2

**int** pidP1;

**int** pChidP1;

// Утсанавливаем соединение с P2 и отправляем сообщение для получения информации о P1 //

// Id канала для отправки сообщения

cout << "P3: установление соединения с каналом P2" << endl;

**int** coidP2 = **ConnectAttach**(0, **getppid**(), pChid, \_NTO\_SIDE\_CHANNEL, 0);

**if**(coidP2 == -1){

cout << "P3: Ошибка соединения с каналом P2" << endl;

exit(EXIT\_FAILURE);

}

**char** rmsg[200] = {};

// Послать сообщение

cout << "Р3: Посылаю сообщение Р2" << rmsg << endl;

**char** \*smsg1 = (**char** \*)"Дай мне pid процесса Р1";

**if**(MsgSend(coidP2, smsg1, strlen(smsg1) + 1, rmsg, **sizeof**(rmsg)) == -1){

cout << "P3: Ошибка MsgSend при отправки в P2" << endl;

exit(EXIT\_FAILURE);

}

**if**(strlen(rmsg) > 0){

cout << "P3: Сервер P2 ответил " << endl;

pidP1 = atoi(rmsg);

cout << "P3: pid процесса Р1: " << pidP1 << endl;

}

// Послать сообщение

cout << "Р3: Посылаю сообщение Р2" << endl;

**char** \*smsg2 = (**char** \*)"Дай мне chid канала Р1";

**if**(MsgSend(coidP2, smsg2, strlen(smsg2) + 1, rmsg, **sizeof**(rmsg)) == -1){

cout << "P3: Ошибка MsgSend при отправки в P2" << endl;

exit(EXIT\_FAILURE);

}

**if**(strlen(rmsg) > 0){

cout << "P3: Сервер P2 ответил " << rmsg << endl;

pChidP1 = atoi(rmsg);

cout << "P3: chid канала Р1: " << pChidP1 << endl;

}

// Утсанавливаем соединение с P2 и отправляем сообщение для получения информации о P1 //

// Утсанавливаем соединение с P1 и отправляем сообщение //

// Id канала для отправки сообщения

cout << "P3: установление соединения с каналом P1" << endl;

**int** coidP1 = **ConnectAttach**(0, pidP1, pChidP1, \_NTO\_SIDE\_CHANNEL, 0);

**if**(coidP1 == -1){

cout << "P3: Ошибка соединения с каналом P1" << endl;

exit(EXIT\_FAILURE);

}

// Буфер ответа

**char** rmsg2[200] = {};

cout << "Р3: Посылаю сообщение Р1" << endl;

**char** \*smsg3 = (**char** \*)"Р3 send message to P1";

**if**(MsgSend(coidP1, smsg3, strlen(smsg3) + 1, rmsg2, **sizeof**(rmsg2)) == -1){

cout << "P3: Ошибка MsgSend при отправки в P1" << endl;

exit(EXIT\_FAILURE);

}

**if**(strlen(rmsg2) > 0){

cout << "P3: Сервер P1 ответил - " << rmsg2 << endl;

}

// Утсанавливаем соединение с P1 и отправляем сообщение //

cout << "Р3: Посылаю сообщение Р2" << endl;

**char** \*smsg4 = (**char** \*)"Р3 stop";

**if**(MsgSend(coidP2, smsg4, strlen(smsg4) + 1, rmsg, **sizeof**(rmsg)) == -1){

cout << "P3: Ошибка MsgSend при отправки в P2" << endl;

exit(EXIT\_FAILURE);

}

**if**(strlen(rmsg) > 0){

cout << "P3: Сервер P2 ответил" << endl;

}

cout << "P3: Р3 ОК" << endl;

**return** EXIT\_SUCCESS;

}

// Отображение текущей директории

**void** **viewWorkDirectory**(){

**char** cCurrentPath[FILENAME\_MAX];

**if** (!GetCurrentDir(cCurrentPath, **sizeof**(cCurrentPath)))

exit(EXIT\_FAILURE);

cCurrentPath[**sizeof**(cCurrentPath) - 1] = '\0';

cout << "P3: текущая рабочая директория - " << cCurrentPath << endl;

}