



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  
Дисциплина \_\_\_\_\_ Технологии промышленного программирования

## ОТЧЁТ

к лабораторной работе

«Запуск и организация взаимодействия параллельных процессов»

Обучающийся группы 6231-020302D \_\_\_\_\_ Гижевская В.Д.

Преподаватель \_\_\_\_\_ Баландин А.В.

Самара 2023

## СОДЕРЖАНИЕ

1	Постановка задачи.....	3
2	Результаты работы .....	4
	ПРИЛОЖЕНИЕ А Листинг модуля М1 .....	5
	ПРИЛОЖЕНИЕ В Листинг модуля М2 .....	7
	ПРИЛОЖЕНИЕ С Листинг модуля М3 .....	9

## 1 Постановка задачи

Разработать приложение, состоящее из трех взаимодействующих процессов. Требуется написать три программных модуля – M1, M2, M3. На базе модуля M1 из shell запускается стартовый процесс P1(M1).

Процесс P1 создает канал и, используя функцию семейства `spawn*`(), запускает процесс P2(M2), передавая ему в качестве параметра `chid` созданного канала, затем переходит в состояние приема сообщений по созданному каналу.

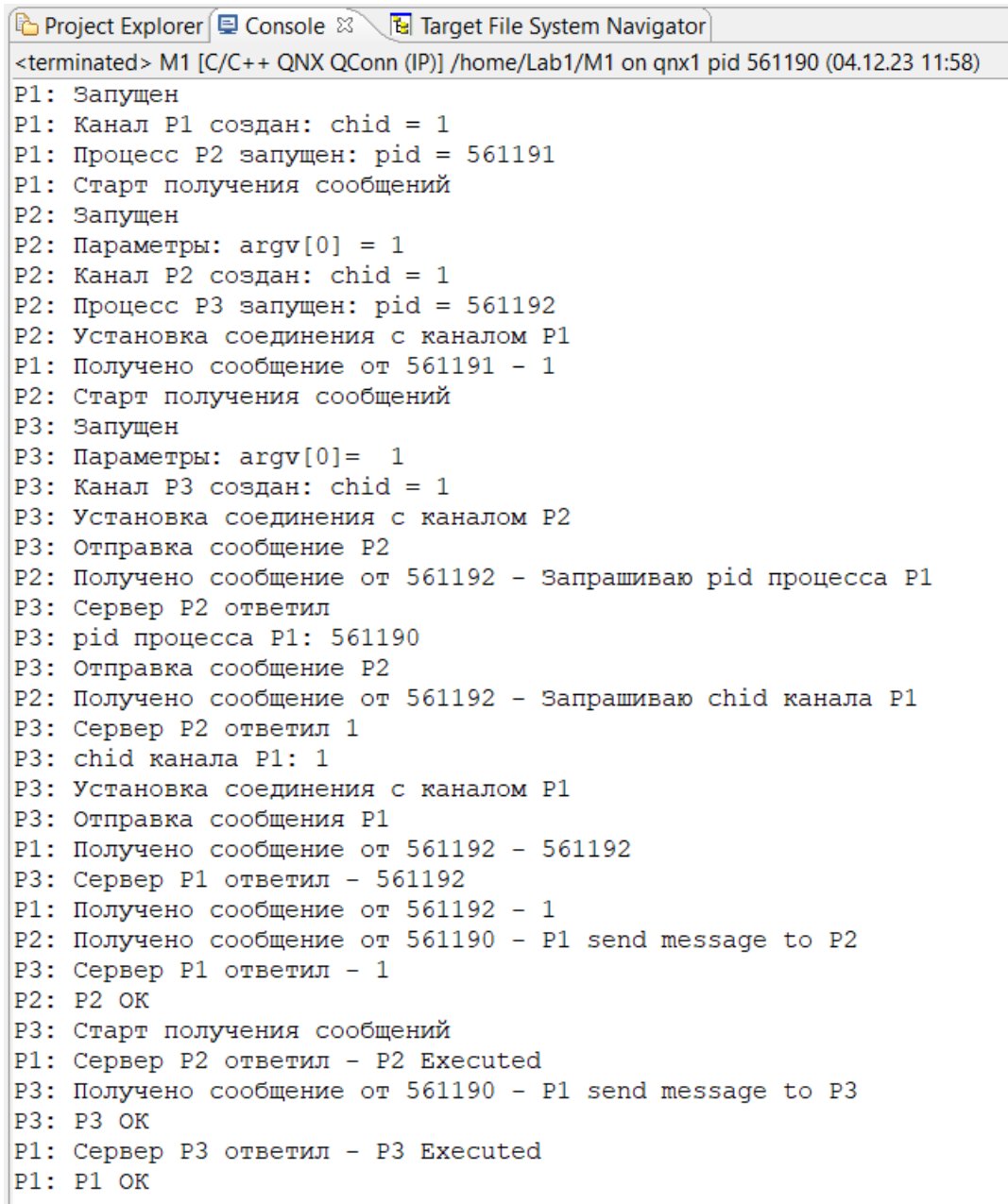
Процесс P2 создает свой канал, и, используя функцию семейства `spawn*`(), запускает процесс P3(M3), передавая ему в качестве параметра `chid` созданного канала, затем устанавливает соединение с каналом процесса P1, передает ему `chid` своего канала, затем переходит в состояние приема сообщений по своему каналу. Процесс P3 устанавливает соединение с каналом процесса P2 и посылает ему запрос на получение `pid` процесса P1 и `chid` его канала. Получив от P2 ответ (`pid` и `chid`), устанавливает соединение с каналом процесса P1 и посылает ему свой `pid` и `chid` своего канала, после чего переходит в состояние приёма сообщений по своему каналу.

Процесс P1 устанавливает соединение с каналом процесса P2 и передает ему сообщение "P1 send message to P2", получив ответ, выводит его на терминал, затем устанавливает соединение с каналом процесса P3 и передает ему сообщение "P1 send message to P3", получает ответ, выводит его на экран и завершается.

Процесс P2, получив сообщение от процесса P1, выводит его на терминал, посылает ответ "P2 executed " и завершается. Процесс P3, получив сообщение от процесса P1, выводит его на экран, посылает ответ "P3 executed " и завершается.

## 2 Результаты работы

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



```
Project Explorer Console Target File System Navigator
<terminated> M1 [C/C++ QNX QConn (IP)] /home/Lab1/M1 on qnx1 pid 561190 (04.12.23 11:58)
P1: Запущен
P1: Канал P1 создан: chid = 1
P1: Процесс P2 запущен: pid = 561191
P1: Старт получения сообщений
P2: Запущен
P2: Параметры: argv[0] = 1
P2: Канал P2 создан: chid = 1
P2: Процесс P3 запущен: pid = 561192
P2: Установка соединения с каналом P1
P1: Получено сообщение от 561191 - 1
P2: Старт получения сообщений
P3: Запущен
P3: Параметры: argv[0]= 1
P3: Канал P3 создан: chid = 1
P3: Установка соединения с каналом P2
P3: Отправка сообщение P2
P2: Получено сообщение от 561192 - Запрашиваю pid процесса P1
P3: Сервер P2 ответил
P3: pid процесса P1: 561190
P3: Отправка сообщение P2
P2: Получено сообщение от 561192 - Запрашиваю chid канала P1
P3: Сервер P2 ответил 1
P3: chid канала P1: 1
P3: Установка соединения с каналом P1
P3: Отправка сообщения P1
P1: Получено сообщение от 561192 - 561192
P3: Сервер P1 ответил - 561192
P1: Получено сообщение от 561192 - 1
P2: Получено сообщение от 561190 - P1 send message to P2
P3: Сервер P1 ответил - 1
P2: P2 OK
P3: Старт получения сообщений
P1: Сервер P2 ответил - P2 Executed
P3: Получено сообщение от 561190 - P1 send message to P3
P3: P3 OK
P1: Сервер P3 ответил - P3 Executed
P1: P1 OK
```

Рисунок 1 – Результат выполнения программы в консоли

## ПРИЛОЖЕНИЕ А

### Листинг модуля М1

```
#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

int main(int argc, char *argv[]) {
    std::cout << "P1: Запущен" << std::endl;

    int chid = ChannelCreate(_NTO_CHF_SENDER_LEN);
    std::cout << "P1: Канал P1 создан: chid = " << chid << std::endl;

    char buffer[20];
    const char *chidStr = itoa(chid, buffer, 10);

    int pidP2 = spawnl(P_NOWAIT, "/home/host/Lab1/M2/x86/o/M2", chidStr,
NULL);
    if (pidP2 < 0){
        std::cout << "P1: Ошибка запуска процесса P2" << pidP2 <<
std::endl;
        exit(EXIT_FAILURE);
    }
    std::cout << "P1: Процесс P2 запущен: pid = " << pidP2 << std::endl;

    int pidP3 = {};
    int chidP3 = {};

    std::cout << "P1: Старт получения сообщений" << std::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){
            std::cout << "P1: Ошибка MsgReceive" << std::endl;
        }

        std::cout << "P1: Получено сообщение от " << info.pid << " - " <<
msg << std::endl;
        if (countMsg == 0){
            MsgReply(rcvid, NULL, msg, sizeof(msg));
        }
        if (countMsg == 1) {
            pidP3 = atoi(msg);
            MsgReply(rcvid, NULL, msg, sizeof(msg));
        }
        if (countMsg == 2) {
            chidP3 = atoi(msg);
        }
    }
}
```

```

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

    countMsg++;
}

char rmsg[200] = {};

int coidP2 = ConnectAttach(0, pidP2, chid, _NTO_SIDE_CHANNEL, 0);
if(coidP2 == -1){
    std::cout << "P1: Ошибка соединения с каналом P2" << std::endl;
    exit(EXIT_FAILURE);
}

char *msg1 = (char *) "P1 send message to P2";

if(MsgSend(coidP2, msg1, strlen(msg1) + 1, rmsg, sizeof(rmsg)) == -1){
    std::cout << "P1: Ошибка MsgSend" << std::endl;
    exit(EXIT_FAILURE);
}

if(strlen(rmsg) > 0){
    std::cout << "P1: Сервер P2 ответил - " << rmsg << std::endl;
}

int coidP3 = ConnectAttach(0, pidP3, chidP3, _NTO_SIDE_CHANNEL, 0);
if(coidP3 == -1){
    std::cout << "P1: Ошибка соединения с каналом P3" << std::endl;
    exit(EXIT_FAILURE);
}

char *msg2 = (char *) "P1 send message to P3";

if(MsgSend(coidP3, msg2, strlen(msg2) + 1, rmsg, sizeof(rmsg)) == -1){
    std::cout << "P1: Ошибка MsgSend при отправке в P3" << std::endl;
    exit(EXIT_FAILURE);
}

if(strlen(rmsg) > 0){
    std::cout << "P1: Сервер P3 ответил - " << rmsg << std::endl;
}

std::cout << "P1: P1 OK" << std::endl;
return EXIT_SUCCESS;
}

```

## ПРИЛОЖЕНИЕ В

### 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

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

    std::cout << "P2: Запущен" << std::endl;

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

    int pChid = atoi(argv[0]);

    int chid = ChannelCreate(_NTO_CHF_SENDER_LEN);
    std::cout << "P2: Канал P2 создан: chid = " << chid << std::endl;

    char buffer[20];
    const char *chidStr = itoa(chid, buffer, 10);

    int pidP3 = spawnl(P_NOWAIT, "/home/host/Lab1/M3/x86/o/M3", chidStr,
NULL);
    if (pidP3 < 0){
        std::cout << "P2: Ошибка запуска процесса P3" << pidP3 <<
std::endl;
        exit(EXIT_FAILURE);
    }
    std::cout << "P2: Процесс P3 запущен: pid = " << pidP3 << std::endl;

    std::cout << "P2: Установка соединения с каналом P1" << std::endl;
    int coidP1 = ConnectAttach(0, getppid(), pChid, _NTO_SIDE_CHANNEL, 0);
    if(coidP1 == -1){
        std::cout << "P2: Ошибка соединения с каналом P1" << std::endl;
        exit(EXIT_FAILURE);
    }

    char rmsg1[200] = {};
    if(MsgSend(coidP1, chidStr, strlen(chidStr) + 1, rmsg1, sizeof(rmsg1))
== -1){
        std::cout << "P2: Ошибка MsgSend" << std::endl;
        exit(EXIT_FAILURE);
    }

    std::cout << "P2: Старт получения сообщений" << std::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){
            std::cout << "P2: Ошибка MsgReceive" << std::endl;
        }
        std::cout << "P2: Получено сообщение от " << info.pid << " - " <<
msg << std::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){
            char *msgEx = (char *) "P2 Executed";
            MsgReply(rcvid, NULL, msgEx, strlen(msgEx) + 1);
        }

        countMsg ++;
    }

    std::cout << "P2: P2 OK" << std::endl;
    return EXIT_SUCCESS;
}

```



## ПРИЛОЖЕНИЕ С

### Листинг модуля МЗ

```
#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

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

    std::cout << "P3: Запущен" << std::endl;

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

    int chid = ChannelCreate(_NTO_CHF_SENDER_LEN);
    std::cout << "P3: Канал P3 создан: chid = " << chid << std::endl;

    int pChid = atoi(argv[0]);

    int pidP1;
    int pChidP1;

    std::cout << "P3: Установка соединения с каналом P2" << std::endl;
    int coidP2 = ConnectAttach(0, getpid(), pChid, _NTO_SIDE_CHANNEL, 0);
    if(coidP2 == -1){
        std::cout << "P3: Ошибка соединения с каналом P2" << std::endl;
        exit(EXIT_FAILURE);
    }

    char rmsg[200] = {};

    std::cout << "P3: Отправка сообщение P2" << rmsg << std::endl;
    char *smsg1 = (char *)"Запрашиваю pid процесса P1";
    if(MsgSend(coidP2, smsg1, strlen(smsg1) + 1, rmsg, sizeof(rmsg)) == -1){
        std::cout << "P3: Ошибка MsgSend" << std::endl;
        exit(EXIT_FAILURE);
    }

    if(strlen(rmsg) > 0){
        std::cout << "P3: Сервер P2 ответил " << std::endl;
        pidP1 = atoi(rmsg);
        std::cout << "P3: pid процесса P1: " << pidP1 << std::endl;
    }

    std::cout << "P3: Отправка сообщение P2" << std::endl;
    char *smsg2 = (char *)"Запрашиваю chid канала P1";
    if(MsgSend(coidP2, smsg2, strlen(smsg2) + 1, rmsg, sizeof(rmsg)) == -1){
        std::cout << "P3: Ошибка MsgSend" << std::endl;
        exit(EXIT_FAILURE);
    }

    if(strlen(rmsg) > 0){
        std::cout << "P3: Сервер P2 ответил " << rmsg << std::endl;
        pChidP1 = atoi(rmsg);
    }
}
```

```

        std::cout << "P3: chid канала P1: " << pChidP1 << std::endl;
    }

    std::cout << "P3: Установка соединения с каналом P1" << std::endl;
    int coidP1 = ConnectAttach(0, pidP1, pChidP1, _NTO_SIDE_CHANNEL, 0);
    if(coidP1 == -1){
        std::cout << "P3: Ошибка соединения с каналом P1" << std::endl;
        exit(EXIT_FAILURE);
    }

    char rmsg2[200] = {};

    std::cout << "P3: Отправка сообщения P1" << std::endl;

    int pid = getpid();
    char pidBuffer[50];
    char const *pidStr = itoa(pid, pidBuffer, 10);

    char buffer[20];
    const char *chidStr = itoa(chid, buffer, 10);

    if(MsgSend(coidP1, pidStr, strlen(pidStr) + 1, rmsg2, sizeof(rmsg2)) ==
-1){
        std::cout << "P3: Ошибка MsgSend" << std::endl;
        exit(EXIT_FAILURE);
    }
    if(strlen(rmsg2) > 0){
        std::cout << "P3: Сервер P1 ответил - " << rmsg2 << std::endl;
    }

    if(MsgSend(coidP1, chidStr, strlen(chidStr) + 1, rmsg2, sizeof(rmsg2))
== -1){
        std::cout << "P3: Ошибка MsgSend" << std::endl;
        exit(EXIT_FAILURE);
    }
    if(strlen(rmsg2) > 0){
        std::cout << "P3: Сервер P1 ответил - " << rmsg2 << std::endl;
    }

    std::cout << "P3: Старт получения сообщений" << std::endl;
    int countMsg = 0;
    while(countMsg < 1){

        char msg[200] = {};
        _msg_info info;
        int rcvid;

        rcvid = MsgReceive(chid, msg, sizeof(msg), &info);
        if(rcvid == -1){
            std::cout << "P3: Ошибка MsgReceive" << std::endl;
        }

        char *msgEx = (char *) "P3 Executed";
        std::cout << "P3: Получено сообщение от " << info.pid << " - " <<
msg << std::endl;
        MsgReply(rcvid, NULL, msgEx, strlen(msgEx) + 1);

        countMsg++;
    }

    std::cout << "P3: P3 ОК" << std::endl;
    return EXIT_SUCCESS;
}

```