**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**

# ОТЧЕТ

по лабораторной работе

## Запуск и организация взаимодействия параллельных процессов

Вариант № 3

Студент: Лазарев М.Ю.

Группа: 6232-020402D

Преподаватель: Баландин А.В.

Самара 2025

# 1 Задание

Цель работы - изучить и практически освоить функции операционной системы QNX для запуска параллельных процессов и организации межпроцессного взаимодействия с помощью механизма обмена сообщениями.

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 процессы P2(M2) и P3(M3), передавая им в качестве параметра chid созданного канала, затем переходит в ожидание сообщений по своему каналу.

Процесс P2 создает свой канал, устанавливает соединение с каналом процесса P1, отправляет ему сообщение о chid своего канала и переходит в состояние приема сообщений по созданному каналу.

Процесс P3 создает свой канал, устанавливает соединение с каналом процесса P1, отправляет ему сообщение о chid своего канала и переходит в состояние приема сообщений по созданному каналу.

Процесс P1, приняв сообщение от процесса P2 или P3 о chid канала, устанавливает соединение и посылает по нему - "P1 send message to P?" (? – номер соответствующего процесса), принимает ответ и выдает его на терминал. После такого взаимодействия с P2 и P3 процесс P1 терминируется.

Процесс P?, получив сообщение от P1, выдает его на экран терминала, посылает ответ "P? OK" процессу P1 и терминируется.

2 Результаты работы

```
// Процесс P1
#include <cstdlib>
#include <iostream>
#include <stdlib.h>
#include <sys/neutrino.h>
#include <unistd.h>
#include <process.h>
#include <string.h>
#include <spawn.h>
#include <sys/wait.h>
int main(int argc, char *argv[]) {
    std::cout << "P1: Запущен..." << std::endl;
    int chid;
    chid = ChannelCreate(0); // создание канала
    char buffer[20];
    const char *chidStr = itoa(chid, buffer, 10); // преобразовать число в строку
    // запуск процессов в режиме NOWAIT (асинхронно)
    int pidP2 = spawnl(P_NOWAIT, "/home/M2", "/home/M2", chidStr, NULL);
    if (pidP2 < 0) {
        std::cout << "P1: Ошибка запуска процесса P2" << std::endl;
        exit(EXIT_FAILURE);
    }
    int pidP3 = spawnl(P_NOWAIT, "/home/M3", "/home/M3", chidStr, NULL);
    if (pidP3 < 0) {
        std::cout << "P1: Ошибка запуска процесса P3" << std::endl;
        exit(EXIT_FAILURE);
    }
```

```
int countMsg = 0; // сколько сообщений обработано
while(countMsg < 2) {
    char msg[200];      // буфер приема сообщения
    _msg_info info;     // информация о сообщении
    int rcvid;          // id сообщения
    rcvid = MsgReceive(chid, msg, sizeof(msg), &info);
    if(rcvid == -1) {
        std::cout << "P1: Ошибка MsgReceive" << std::endl;
        continue;
    }
    // обработка сообщения от P2
    if (info.pid == pidP2) {
        int ChidP2 = atoi(msg);
        strcpy(msg, "сообщение обработано");
        MsgReply(rcvid, NULL, msg, sizeof(msg));
        int coid = ConnectAttach(0, pidP2, ChidP2, _NTO_SIDE_CHANNEL, 0);
        if(coid == -1) {
            std::cout << "P1: ошибка соединения с P2" << std::endl;
            break;
        }
        char rmsg[200];
        char *smsg = (char *)"P1 отправил сообщение P2";
        if(MsgSend(coid, smsg, strlen(smsg)+1, rmsg, sizeof(rmsg)) == -1) {
            std::cout << "P1: Ошибка MsgSend -> P2" << std::endl;
            break;
        }
        std::cout << "P1: Ответ от P2: " << rmsg << std::endl;
        countMsg++;
```

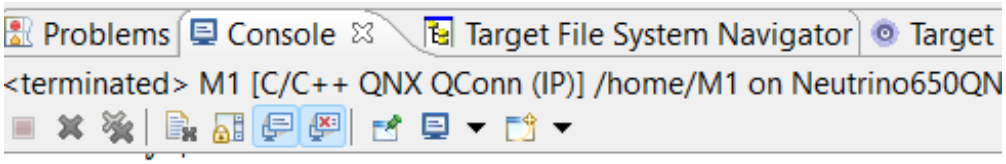
```
    }
    // обработка сообщения от P3
    if (info.pid == pidP3) {
        int ChidP3 = atoi(msg);
        strcpy(msg, "сообщение обработано");
        MsgReply(rcvid, NULL, msg, sizeof(msg));
        int coid = ConnectAttach(0, pidP3, ChidP3, _NTO_SIDE_CHANNEL, 0);
        if(coid == -1) {
            std::cout << "P1: ошибка соединения с P3" << std::endl;
            break;
        }
        char rmsg[200];
        char *smsg = (char *)"P1 отправил сообщение P3";
        if(MsgSend(coid, smsg, strlen(smsg)+1, rmsg, sizeof(rmsg)) == -1) {
            std::cout << "P1: Ошибка MsgSend -> P3" << std::endl;
            break;
        }
        std::cout << "P1: Ответ от P3: " << rmsg << std::endl;
        countMsg++;
    }
}
// дождаться завершения P2 и P3
int status;
waitpid(pidP2, &status, 0);
std::cout << "P1: P2 завершился" << std::endl;
waitpid(pidP3, &status, 0);
std::cout << "P1: P3 завершился" << std::endl;
```

```
    std::cout << "P1: завершился" << std::endl;
    return EXIT_SUCCESS;
}

//Процесс P2
#include <cstdlib>
#include <iostream>
#include <stdlib.h>
#include <sys/neutrino.h>
#include <unistd.h>
#include <process.h>
#include <string.h>
int main(int argc, char *argv[]) {
        std::cout << "P2: запущен..." << std::endl;
        int chid; // id канала
        chid = ChannelCreate(0); // создание канала для приёма сообщений от P1
        char buffer[20];
        const char *chidStr = itoa(chid, buffer, 10); // конвертировать в строку
        int pChid; // id канала родительского процесса
        pChid = atoi(argv[1]); // преобразовать в int
        int coid; // id канала для отправки сообщения
        std::cout <<"P2: установление соединения с каналом"<< std::endl;
        coid = ConnectAttach(0, getppid(), pChid, _NTO_SIDE_CHANNEL, 0); // установка
соединения с каналом P1
        if(coid == -1){
                std::cout <<"Ошибка соединения с каналом"<< std::endl;
           exit(EXIT_FAILURE);
        }
        char rmsg[200]; // буфер ответа
```

```
        // послать сообщение о своём chid
        std::cout <<"P2: Посылаю сообщение"<< std::endl;
        if(MsgSend(coid, chidStr, strlen(chidStr)+1, rmsg, sizeof(rmsg)) == -1){
        std::cout <<"P2: Ошибка MsgSend"<< std::endl;
                exit(EXIT_FAILURE);
        }
        if(strlen(rmsg) > 0) std::cout <<"P2: Получен ответ от P1: " << rmsg << std::endl;
        int rcvid;         // ссылка на нить P1
        _msg_info info;    // информация о сообщении
        char msg[200];     // буфер приёма сообщения
        rcvid = MsgReceive(chid, msg, sizeof(msg), &info);  // получить сообщение
        if(rcvid == -1) std::cout <<"P2: Ошибка MsgReceive"<< std::endl;
        else {
                std::cout <<"P2: Получено сообщение: "<< msg << std::endl;
                strcpy(msg, "P2 OK");
                MsgReply(rcvid, NULL, msg, sizeof(msg)); // посылка ответа клиенту
}
std::cout << "P2: Завершается" << std::endl;
        return EXIT_SUCCESS;
}

//Процесс P3
#include <cstdlib>
#include <iostream>
#include <stdlib.h>
#include <sys/neutrino.h>
#include <unistd.h>
#include <process.h>
#include <string.h>
int main(int argc, char *argv[]) {
```

```
	std::cout << "P3: запущен..." << std::endl;
	int chid; // id канала
	chid = ChannelCreate(0); // создание канала для приёма сообщений от P1
	char buffer[20];
	const char *chidStr = itoa(chid, buffer, 10); // конвертировать в строку
	int pChid; // id канала родительского процесса
	pChid = atoi(argv[1]); // преобразовать в int
	int coid; // id канала для отправки сообщения
	std::cout <<"P3: установление соединения с каналом"<< std::endl;
	coid = ConnectAttach(0, getppid(), pChid, _NTO_SIDE_CHANNEL, 0); // установка соединения с каналом P1
	if(coid == -1){
		std::cout <<"Ошибка соединения с каналом"<< std::endl;
	   exit(EXIT_FAILURE);
	}
	char rmsg[200]; // буфер ответа
	// послать сообщение о своём chid
	std::cout <<"P3: Посылаю сообщение"<< std::endl;
	if(MsgSend(coid, chidStr, strlen(chidStr)+1, rmsg, sizeof(rmsg)) == -1){
	std::cout <<"P3: Ошибка MsgSend"<< std::endl;
		exit(EXIT_FAILURE);
	}
	if(strlen(rmsg) > 0) std::cout <<"P3: Получен ответ от P1: " << rmsg << std::endl;
	int rcvid;		// ссылка на нить P1
	_msg_info info;	// информация о сообщении
	char msg[200];	// буфер приёма сообщения
	rcvid = MsgReceive(chid, msg, sizeof(msg), &info);  // Получить сообщение
	if(rcvid == -1) std::cout <<"P3: Ошибка MsgReceive"<< std::endl;
	else {
		std::cout <<"P3: Получено сообщение: "<< msg << std::endl;
		strcpy(msg, "P3 OK");
		MsgReply(rcvid, NULL, msg, sizeof(msg)); // посылка ответа клиенту
	}
	std::cout << "P3: Завершается" << std::endl;
```

```
    return EXIT_SUCCESS;
}
```




```
P2: установление соединения с каналом
P2: Посылаю сообщение
P2: Получен ответ от P1: сообщение обработано
P2: Получено сообщение: P1 отправил сообщение P2
P2: Завершается
P1: Ответ от P2: P2 OK
P3: запущен...
P3: установление соединения с каналом
P3: Посылаю сообщение
P3: Получен ответ от P1: сообщение обработано
P3: Получено сообщение: P1 отправил сообщение P3
P3: Завершается
P1: Ответ от P3: P3 OK
P1: P2 завершился
P1: P3 завершился
P1: завершился
```