Министерство образования и науки Российской Федерации

Санкт-Петербургский государственный электротехнический университет

«ЛЭТИ»

Факультет компьютерных технологий и информатики

Кафедра автоматики и процессов управления

отчет

**по лабораторной работе №8**

**по дисциплине «ПСРВ»**

Тема: Разработка многопоточной программы управления роботом

по координатам линейных перемещений.

|  |  |  |
| --- | --- | --- |
| Студент гр. 3391 |  | Малина А. |
| Преподаватель |  | Дорогов А.Ю. |

Санкт-Петербург

## 2018

## Цель работы:

Освоение системных средств, порождения программных потоков и поддержки взаимодействия процессов через импульсы и сигналы.

## Задание 1. Взаимодействие процессов через импульсы

На основе программы **prog1** (см. Практическое занятие 7) разработать двухпоточное приложение **prog2** , в котором поток **main** реализует пользовательский интерфейс для управления эмулятором робота, а дочерний поток **Display** отображает текущее состояние линейных координат X,Y,Z.



Поток Display принимает и обрабатывает импульсы от датчиков перемещения робота манипулятора. Импульсы формируются при перемещении по любой координате на один шаг. Код импульса (поле **code**) несет информацию о координате:

1- координата **X**,

2- координат **Y**,

4- координата **Z**.

Значение импульса (поле **value**) содержит текущее значение координаты.

Эмулятор робота способен генерировать импульсы только после выполнения процедуры инициализации датчиков перемещения. Формат инициализирующих команд:

struct MESSAGE // пересылаемое сообщение send message structure

{

unsigned char type; // 0,1,2,3,4,5,6

unsigned int buf;// send data

};

* Инициализация датчика координаты **X**  type = 4, buf = номер канала,
* Инициализация датчика координаты **Y** type = 5, buf = номер канала,
* Инициализация датчика координаты **Z** type = 6, buf = номер канала.

При инициализации, эмулятор робота открывает соединения с каналом для каждого датчика.

Инициализация датчиков подтверждается сообщениями **Init X, Init Y, Init Z**.

**Ход работы:**

1) В среде Momentix создан проект PROG2

2) В функции main() ожидается команда ввода с клавиатуры.

3) В потоке Display создан канал с помощью системного вызова ChannelCreate(). Номер канала следует выводится на экране с помощью оператора printf().

chid = **ChannelCreate**(0);

**printf**("Создан канал %d\n ", chid);

4) В потоке Display, используя системные вызовы MsgSend() и полученный номер канала, инициализируются датчики перемещения робота по координатам X,Y,Z.

// Инициализация датчиков координат X,Y,Z

**for**(**int** i = 4; i < 7; i++)

{

msg.type = i;

msg.buf = chid;

**MsgSend**(coid, &msg, **sizeof**(msg), NULL, 0);

}

5) В потоке Display, в бесконечном цикле while организуется прием импульсов от эмулятора, используя системный вызов MsgReseivePulse().

**for**(;;){

**MsgReceivePulse**(chid, &pulse, **sizeof**(pulse), NULL);

...

}

6) В теле цикла, анализ импульсов выполняется с помощью конструкции switch/case.

**switch**(pulse.code)

{

**case** B\_X: // Если изменилась координата X.

x\_cnt = pulse.value.sival\_int;

changed = 1;

**break**;

**case** B\_Y: // Если изменилась координата Y.

y\_cnt = pulse.value.sival\_int;

changed = 1;

**break**;

**case** B\_Z: // Если изменилась координата Z.

z\_cnt = pulse.value.sival\_int;

changed = 1;

**break**;

**default**: // Если изменились другие координаты.

changed = 0;

**break**;

}

7) В теле цикла, вывод организован оператором printf()

**if**(changed)

{

// Вывод текущих координат на экран.

**printf**("Roby's coordinates:\r\n\tX=%d\r\n\tY=%d\r\n\tZ=%d\r\n",

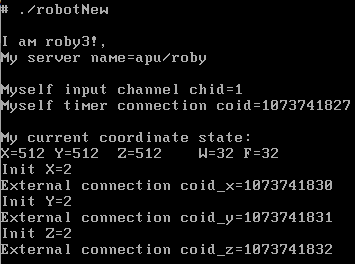
x\_cnt,

y\_cnt,

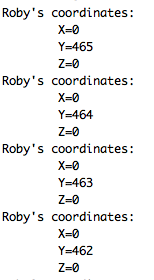
z\_cnt);

}

При запуске программы эмулятор выводит информацию о новых подключениях



Скриншот с работой программы (команда «движение по Y назад») приведен ниже.



**Задание 2. Взаимодействие процессов через сигналы**

Эмулятор робота способен генерировать сигналы только после выполнения процедуры инициализации датчиков перемещения значением buf=0. Для взаимодействия используется сигнал **SIGUSR1**. Сигнал формируется при изменении любой линейной координаты **X,Y,Z** на один шаг.

Код сигнала (поле **code** отрицательное число) несет информацию о координате:

-1- координата **X**,

-2- координат **Y**,

-4- координата **Z**.

Значение импульса (поле **value**) содержит текущее значение координаты.

**Ход работы:**

1) В среде Momentix создан проект PROG3

2) В потоке Display создано описание для сигнала SIGUSR1, **,** используя системный вызов sigaction() и сопутствующие сервисные вызовы sigemptyset(), sigaddset()

sigset\_t set; //Набор сигналов.

**sigemptyset**(&set); // Очистить набор сигналов set.

**sigaddset**(&set, SIGUSR1); // Добавить в набор set сигнал SIGUSR1.

3) Для полей структуры struct sigaction act установлены значения

act.sa\_mask = set; // Маска сигналов.

act.sa\_handler = &handler; // Указать обработчик сигналов.

/\*

Если в sa\_flags указан SA\_SIGINFO,

то sa\_sigaction (вместо sa\_handler) задаёт функцию обработки сигнала signum.

В первом аргументе функция принимает номер сигнала

\*/

act.sa\_flags = SA\_SIGINFO;

4) Для функции обработчика использована пустая функция следующего вида

**void** **handler**(**int** signo)

{

// Обработчик сигналов SIGUSR1.

}

5) В потоке Display, используя системные вызовы MsgSend(), инициализированы датчики перемещения робота по координатам X,Y,Z значением 0

// Инициализация датчиков перемещения по X, Y и Z робота roby

msg.buf = 0;

msg.type = 4;

**MsgSend**(coid, &msg, **sizeof**(msg), NULL, 0);

msg.type = 5;

**MsgSend**(coid, &msg, **sizeof**(msg), NULL, 0);

msg.type = 6;

**MsgSend**(coid, &msg, **sizeof**(msg), NULL, 0);

6) В теле цикла использован системный вызов SignalWaitinfo() для приема сигнала.

// Приём СИГНАЛА SIGUSR1 от roby

// Ожидать СИГНАЛ из набора set.

**sigwaitinfo**(&set, &info);

7) В теле цикла, анализ импульсов выполняйтся с помощью конструкции switch/case

// Анализ сигнала по коду

**switch**(info.si\_code)

{

**case** -B\_X:

x\_cnt = info.si\_value.sival\_int;

changed = 1;

**break**;

**case** -B\_Y:

y\_cnt = info.si\_value.sival\_int;

changed = 1;

**break**;

**case** -B\_Z:

z\_cnt = info.si\_value.sival\_int;

changed = 1;

**break**;

**default**:

changed = 0;

**printf**("\nDisplay thread: unknown signal code: %i", info.si\_code);

}

8) В теле цикла организуван вывод значений координат с помощью оператора printf()

**if**(changed)

{

// Вывод текущих координат на экран.

**printf**("Roby's coordinates:\r\n\tX=%d\r\n\tY=%d\r\n\tZ=%d\r\n",

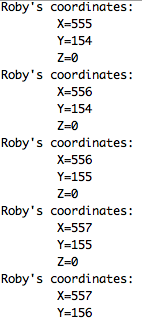
x\_cnt,

y\_cnt,

z\_cnt);

}

Скриншот работы программы (команда “движение по X вперед» “движение по Y вперед» приведен ниже

****

**Приложение**

**Листинг программы prog2.c**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <sys/iofunc.h>

#include <sys/dispatch.h>

#include <sys/neutrino.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include "roby.h"

#define ATTACH\_POINT "apu/roby"

#define COMMANDS\_MAX\_COUNT 10

// пересылаемое сообщение

struct MESSAGE

{

// message type

unsigned char type;

// send data

unsigned int buf;

};

int coid;

void \*Display() {

struct MESSAGE msg;

struct \_pulse pulse;

int chid, x\_cnt, y\_cnt, z\_cnt;

x\_cnt = y\_cnt = z\_cnt = 0;

int changed = 0;

chid = ChannelCreate(0);

printf("Создан канал %d\n ", chid);

// Инициализация датчиков координат X,Y,Z

for(int i = 4; i < 7; i++)

{

msg.type = i;

msg.buf = chid;

MsgSend(coid, &msg, sizeof(msg), NULL, 0);

}

for(;;)

{

// Приём испульсных сообщений по открытому каналу chid:

MsgReceivePulse(chid, &pulse, sizeof(pulse), NULL);

switch(pulse.code)

{

case B\_X: // Если изменилась координата X.

x\_cnt = pulse.value.sival\_int;

changed = 1;

break;

case B\_Y: // Если изменилась координата Y.

y\_cnt = pulse.value.sival\_int;

changed = 1;

break;

case B\_Z: // Если изменилась координата Z.

z\_cnt = pulse.value.sival\_int;

changed = 1;

break;

default: // Если изменились другие координаты.

changed = 0;

break;

}

if(changed)

{

// Вывод текущих координат на экран.

printf("Roby's coordinates:\r\n\tX=%d\r\n\tY=%d\r\n\tZ=%d\r\n",

x\_cnt,

y\_cnt,

z\_cnt);

}

}

}

int main()

{

unsigned char PA=0, PC=0; //Эти переменные хранят значения регистров PA и PC

struct MESSAGE msg; // буфер посылаемого сообщения

unsigned char rmsg; // Буфер ответного сообщения

char command, ch[3];

coid = name\_open(ATTACH\_POINT, 0);

if(coid == -1)

{

return EXIT\_FAILURE;

}

printf("apu/roby has coid = %d\n", coid);

pthread\_create(NULL, NULL, &Display, NULL);

do

{

printf("Enter command>\n");

scanf("%s", ch);

switch(ch[0])

{

case 'I':

// Двигаться в начальное положение по всем координатам

// и сбросить значения датчиков положений.

PA = 0; PC = 0; //Для этого сбросить в 0 переменные PA и PC.

command ='C'; //Подать команду записи в регистр PC роботу Roby.

break;

case '+': case '=':

command = 'c';

break;

case '!':

command = 'B';

break;

case 'Y':

scanf("%s", ch);

switch(ch[0])

{

case 'X':

PA ^= A\_X\_FORWARD;

command ='A';

break;

case 'x':

PA ^= A\_X\_BACK;

command ='A';

break;

case 'Y':

PA ^= A\_Y\_FORWARD;

command ='A';

break;

case 'y':

PA ^= A\_Y\_BACK;

command ='A';

break;

case 'Z':

PA ^= A\_Z\_FORWARD;

command ='A';

break;

case 'z':

PA ^= A\_Z\_BACK;

command ='A';

break;

case 'F':

PC ^= C\_F\_FORWARD;

command ='C';

break;

case 'f':

PC ^= C\_F\_BACK;

command ='C';

break;

case 'W':

PC ^= C\_W\_FORWARD;

command ='C';

break;

case 'w':

PC ^= C\_W\_BACK;

command ='C';

break;

case 'S':

PA ^= A\_S;

command ='A';

break;

case 'D':

PA ^= A\_D;

command ='A';

break;

}

}

switch(command)

{

case 'A':

msg.buf = PA;

msg.type = 0;

MsgSend(coid, &msg, sizeof(msg), NULL, 0);

break;

case 'B':

msg.type = 3;

MsgSend(coid, &msg, sizeof(msg), &rmsg, sizeof(rmsg));

printf("Port B: 0x%X\n", rmsg);

break;

case 'c':

msg.type = 2;

MsgSend(coid, &msg, sizeof(msg), &rmsg, sizeof(rmsg));

printf("Port C: 0x%X\n", rmsg);

break;

case 'C':

msg.type = 1;

msg.buf = PC;

MsgSend(coid, &msg, sizeof(msg), NULL, 0);

break;

default:

printf("default last\n");

break;

}

} while(1);

}

**Листинг программы prog3.c**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <sys/iofunc.h>

#include <sys/dispatch.h>

#include <sys/neutrino.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include "roby.h"

#define ATTACH\_POINT "apu/roby"

#define COMMANDS\_MAX\_COUNT 10

// пересылаемое сообщение

struct MESSAGE

{

// message type

unsigned char type;

// send data

unsigned int buf;

};

struct PULSE

{

unsigned int code;

unsigned int value;

};

int coid;

void handler(int signo)

{

// Обработчик сигналов SIGUSR1.

}

void\* Display()

{

struct sigaction act;

sigset\_t set; //Набор сигналов.

siginfo\_t info; //Информация о сигнале.

int changed = 0;

int x\_cnt, y\_cnt, z\_cnt;

x\_cnt = y\_cnt = z\_cnt = 0;

struct MESSAGE msg;

sigemptyset(&set); // Очистить набор сигналов set.

sigaddset(&set, SIGUSR1); // Добавить в набор set сигнал SIGUSR1.

act.sa\_mask = set; // Маска сигналов.

act.sa\_handler = &handler; // Указать обработчик сигналов.

/\*

Если в sa\_flags указан SA\_SIGINFO,

то sa\_sigaction (вместо sa\_handler) задаёт функцию обработки сигнала signum.

В первом аргументе функция принимает номер сигнала

\*/

act.sa\_flags = SA\_SIGINFO;

/\* Установить связь между получением

данным процессом сигнала SIGUSR1 и действием act,

которое предусматривает вызов обработчика сигналов handler()

\*/

sigaction(SIGUSR1, &act, NULL);

// Инициализация датчиков перемещения по X, Y и Z робота roby

msg.buf = 0;

msg.type = 4;

MsgSend(coid, &msg, sizeof(msg), NULL, 0);

msg.type = 5;

MsgSend(coid, &msg, sizeof(msg), NULL, 0);

msg.type = 6;

MsgSend(coid, &msg, sizeof(msg), NULL, 0);

while(1)

{

//sleep(1); //Вернуть управление потоку main (на некоторое время).

// Приём СИГНАЛА SIGUSR1 от roby

// Ожидать СИГНАЛ из набора set.

sigwaitinfo(&set, &info);

// Анализ сигнала по коду

switch(info.si\_code)

{

case -B\_X:

x\_cnt = info.si\_value.sival\_int;

changed = 1;

break;

case -B\_Y:

y\_cnt = info.si\_value.sival\_int;

changed = 1;

break;

case -B\_Z:

z\_cnt = info.si\_value.sival\_int;

changed = 1;

break;

default:

changed = 0;

printf("\nDisplay thread: unknown signal code: %i", info.si\_code);

}

if(changed)

{

// Вывод текущих координат на экран.

printf("Roby's coordinates:\r\n\tX=%d\r\n\tY=%d\r\n\tZ=%d\r\n",

x\_cnt,

y\_cnt,

z\_cnt);

}

}

// Завершить работу потока Display

pthread\_exit(NULL);

}

int main()

{

unsigned char PA = 0, PC = 0; //Эти переменные хранят значения регистров PA и PC

struct MESSAGE msg; // буфер посылаемого сообщения

unsigned char rmsg; // Буфер ответного сообщения

char command, ch[3];

coid = name\_open(ATTACH\_POINT, 0);

if(coid == -1)

{

return EXIT\_FAILURE;

}

printf("apu/roby has coid = %d\n", coid);

pthread\_create(NULL, NULL, &Display, NULL);

do

{

printf("Enter command>\n");

scanf("%s", ch);

switch(ch[0])

{

case 'I':

// Двигаться в начальное положение по всем координатам

// и сбросить значения датчиков положений.

PA = 0; PC = 0; // Для этого сбросить в 0 переменные PA и PC.

command ='C'; // Подать команду записи в регистр PC роботу Roby.

break;

case '+': case '=':

command = 'c';

break;

case '!':

command = 'B';

break;

case 'Y':

scanf("%s", ch);

switch(ch[0])

{

case 'X':

PA ^= A\_X\_FORWARD;

command ='A';

break;

case 'x':

PA ^= A\_X\_BACK;

command ='A';

break;

case 'Y':

PA ^= A\_Y\_FORWARD;

command ='A';

break;

case 'y':

PA ^= A\_Y\_BACK;

command ='A';

break;

case 'Z':

PA ^= A\_Z\_FORWARD;

command ='A';

break;

case 'z':

PA ^= A\_Z\_BACK;

command ='A';

break;

case 'F':

PC ^= C\_F\_FORWARD;

command ='C';

break;

case 'f':

PC ^= C\_F\_BACK;

command ='C';

break;

case 'W':

PC ^= C\_W\_FORWARD;

command ='C';

break;

case 'w':

PC ^= C\_W\_BACK;

command ='C';

break;

case 'S':

PA ^= A\_S;

command ='A';

break;

case 'D':

PA ^= A\_D;

command ='A';

break;

}

}

switch(command)

{

case 'A':

msg.buf = PA;

msg.type = 0;

MsgSend(coid, &msg, sizeof(msg), NULL, 0);

break;

case 'B':

msg.type = 3;

MsgSend(coid, &msg, sizeof(msg), &rmsg, sizeof(rmsg));

printf("Port B: 0x%X\n", rmsg);

break;

case 'c':

msg.type = 2;

MsgSend(coid, &msg, sizeof(msg), &rmsg, sizeof(rmsg));

printf("Port C: 0x%X\n", rmsg);

break;

case 'C':

msg.type = 1;

msg.buf = PC;

MsgSend(coid, &msg, sizeof(msg), NULL, 0);

break;

default:

printf("Unknown command\n");

break;

}

} while(1);

}

**Листинг заголовочного файла roby.h**

**#ifndef** ROBY\_H\_

**#define** ROBY\_H\_

**#define** A\_D 0x01

**#define** A\_S 0x02

**#define** A\_X\_FORWARD 0x04

**#define** A\_X\_BACK 0x08

**#define** A\_Z\_BACK 0x10

**#define** A\_Z\_FORWARD 0x20

**#define** A\_Y\_BACK 0x40

**#define** A\_Y\_FORWARD 0x80

**#define** B\_X 0x01

**#define** B\_Y 0x02

**#define** B\_Z 0x04

**#define** B\_W\_END 0x08

**#define** B\_W\_BEGIN 0x10

**#define** B\_Z\_BEGIN 0x20

**#define** B\_Y\_BEGIN 0x40

**#define** B\_X\_BEGIN 0x80

**#define** C\_F\_END 0x04

**#define** C\_F\_BEGIN 0x08

**#define** C\_W\_FORWARD 0x10

**#define** C\_W\_BACK 0x20

**#define** C\_F\_FORWARD 0x40

**#define** C\_F\_BACK 0x80

**#define** X\_MIN 0

**#define** X\_MAX 1024

**#define** Y\_MIN 0

**#define** Y\_MAX 1024

**#define** Z\_MIN 0

**#define** Z\_MAX 1024

**#define** W\_MIN 0

**#define** W\_MAX 100

**#define** F\_MIN 0

**#define** F\_MAX 50

**#define** S\_MIN 0

**#define** S\_MAX 1

**#define** D\_MIN 0

**#define** D\_MAX 1

**#define** W\_END 0x08

**#define** W\_BEGIN 0x10

**#define** F\_END 0x20

**#define** F\_BEGIN 0x40

**#endif** /\* ROBY\_H\_ \*/