

Министерство науки и высшего образования  
Российской Федерации

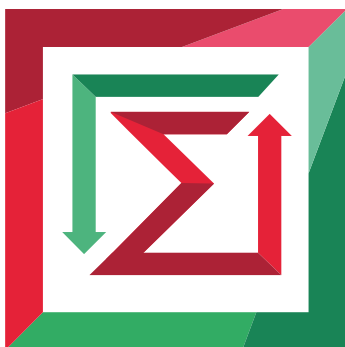
Федеральное государственное бюджетное образовательное  
учреждение высшего образования  
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»



**НГТУ  
НЭТИ**

Кафедра теоретической и прикладной информатики

Лабораторная работа № 5  
по дисциплине «Проектирование Систем Реального Времени»  
Синхронизация потоков



ФАКУЛЬТЕТ:	ПМИ
ГРУППА:	ПМИМ-01
СТУДЕНТЫ:	Ершов П. К. Гриценко И. Г.
БРИГАДА:	7
ПРЕПОДАВАТЕЛЬ:	Кобылянский В. Г.

Новосибирск  
2021

## 1. Цель работы

Цель работы - изучение основных принципов использования QNX Momentics IDE.

## 2. Задание на лабораторную

1. С помощью средств перспективы QNX System Information определить следующую информацию о целевой платформе:
  - архитектуру и частоту процессора;
  - общий объем оперативной памяти и объем занятой памяти;
  - версию операционной системы QNX;
  - количество запущенных процессов;
  - количество потоков у процесса **pipe** и их состояние.
2. Создать QNX C Project с кодом приложения согласно варианту. Передать проект под управление CVS.
3. Скомпилировать проект, исправить синтаксические ошибки, если такие имеются. Занести исправления в репозиторий с описанием изменений.
4. Создать конфигурацию для запуска отладки и выполнить пошаговую отладку приложения. Внесенные исправления также заносить в репозиторий с описанием.
5. Создать конфигурацию для обычного запуска без отладочной информации и выполнить запуск приложения. Сравнить размеры исполняемых файлов с отладочной информацией и без нее (пункт Properties в контекстном меню нужного файла в представлении Project Explorer).
6. Выполнить профилирование методами “Function Instrumentation profiling” и “Sampling and Call Count instrumentation profiling”. Из результатов получить время выполнения функций программы для обоих методов. Занести данные по времени в таблицу:

Функция	Затраченное время
main	
sorting_str	
print_str	

### 3. Ход работы.

#### 3.1. Определение информации о целевой платформе с помощью перспективы QNX System Information

- архитектура и частота процессора: x86, 3.186 Ghz

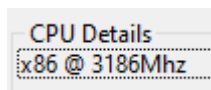


Рисунок 1.

- общий объем оперативной памяти и объем занятой памяти 2 ГВт

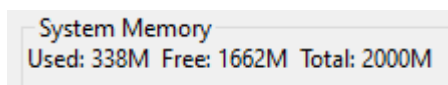


Рисунок 2.

- версия операционной системы QNX 6.5.9

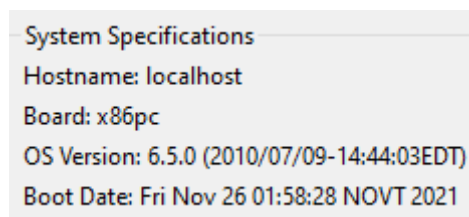
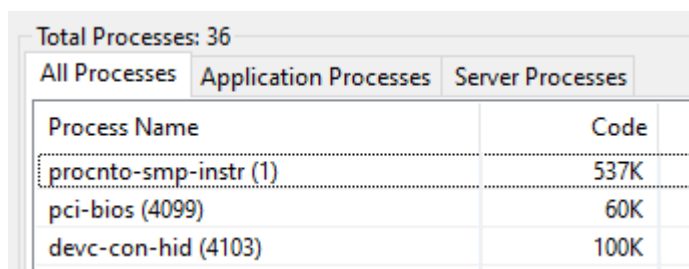


Рисунок 3.

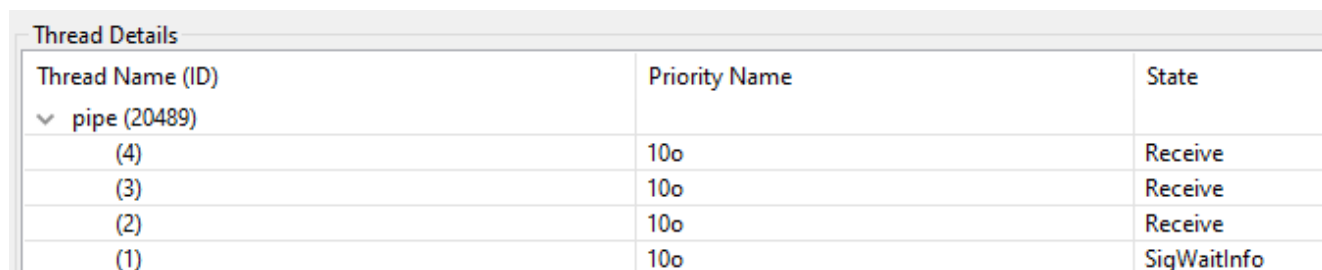
- количество запущенных процессов - 36



Total Processes: 36	
Process Name	Code
procnto-smp-instr (1)	537K
pci-bios (4099)	60K
devc-con-hid (4103)	100K

Рисунок 4.

- количество потоков у процесса **pipe** и их состояние



Thread Details		
Thread Name (ID)	Priority Name	State
▼ pipe (20489)		
(4)	10o	Receive
(3)	10o	Receive
(2)	10o	Receive
(1)	10o	SigWaitInfo

Рисунок 5.

- 3.2.** Создать QNX C Project с кодом приложения согласно варианту.  
Передать проект под управление CVS

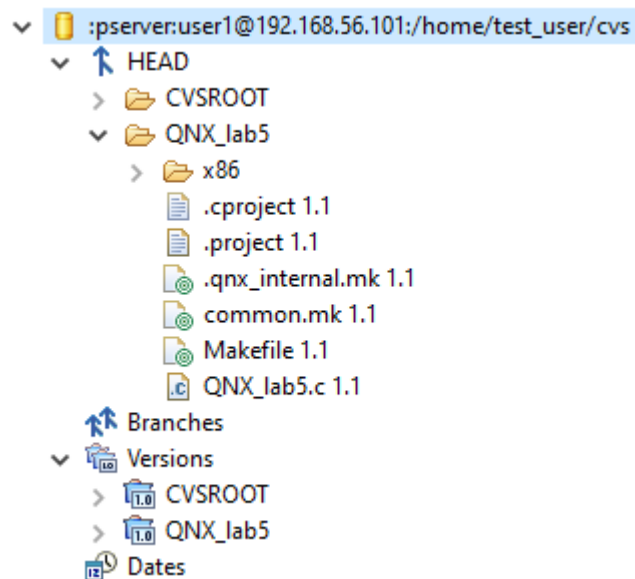


Рисунок 6. Проект создан и передан под управление CVS

- 3.3.** Скомпилировать проект, исправить синтаксические ошибки, если такие имеются. Занести исправления в репозиторий с описанием изменений.

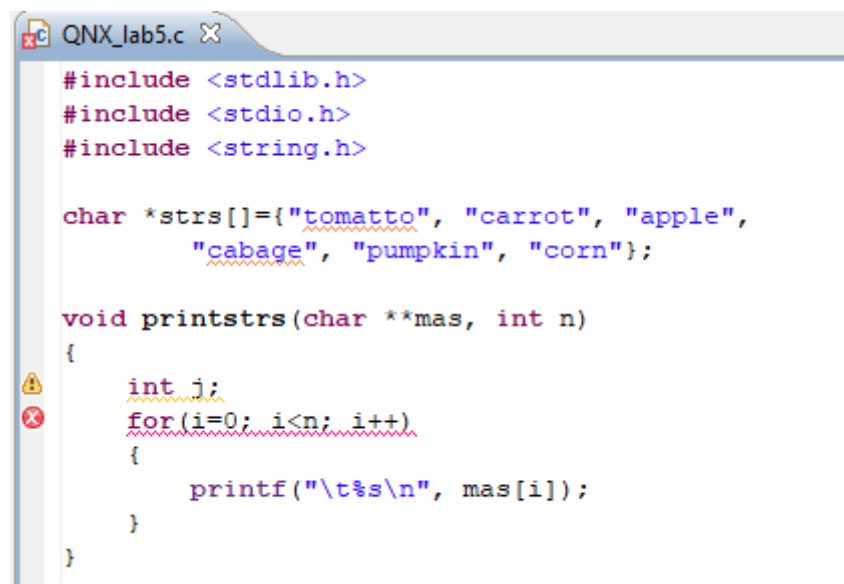


Рисунок 7. Ошибка в программе.

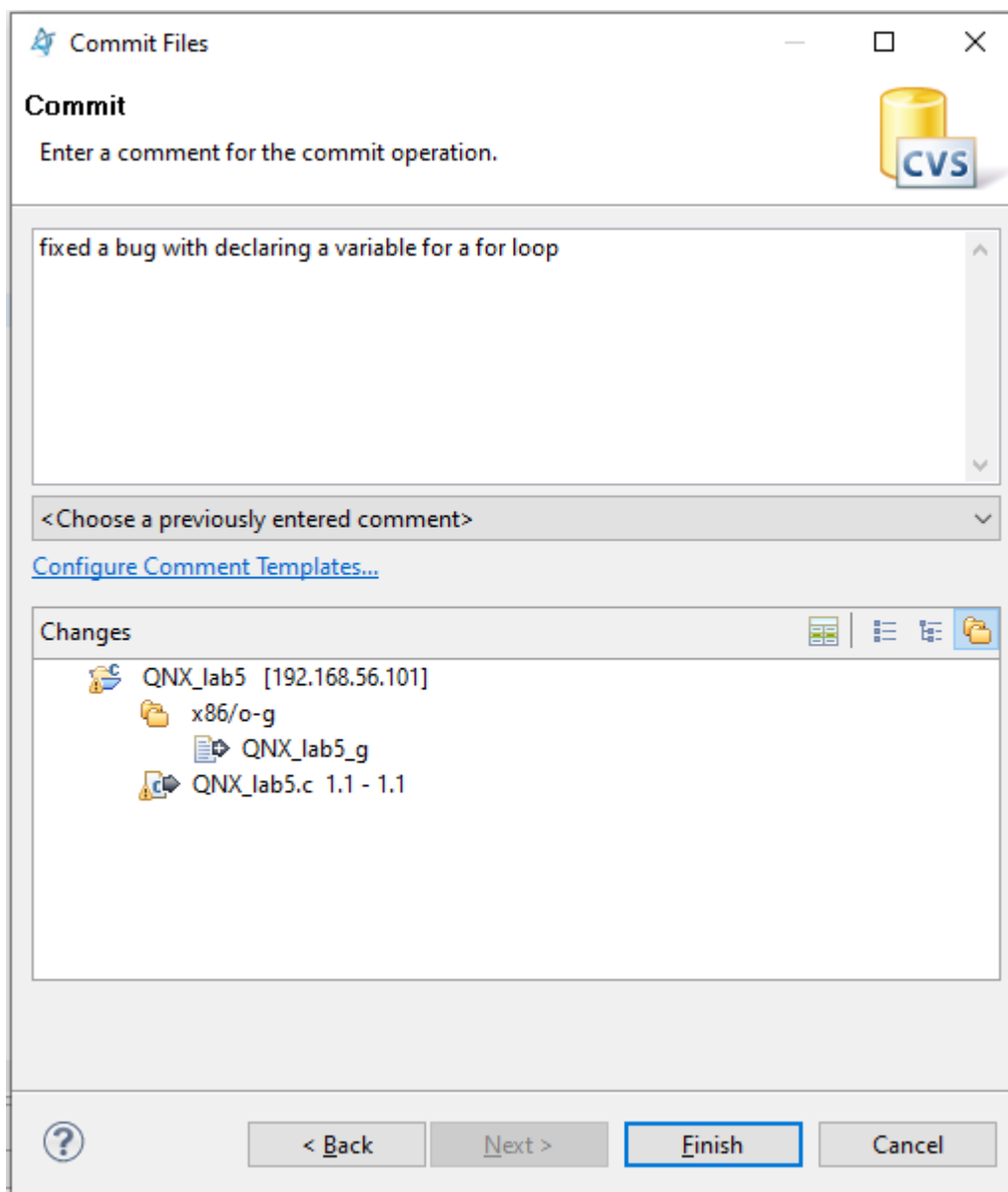


Рисунок 8. Создание коммита

### 3.4. Создать конфигурацию для запуска отладки и выполнить пошаговую отладку приложения. Внесенные исправления также заносить в репозиторий с описанием

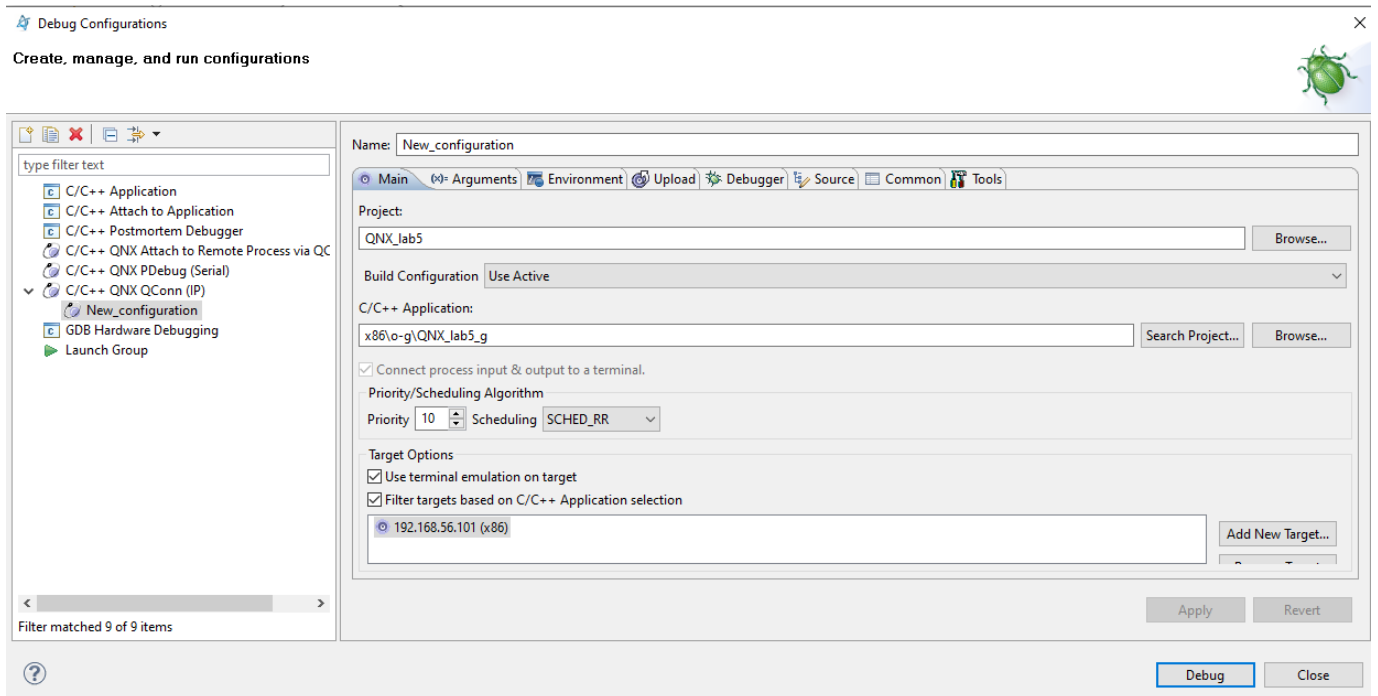


Рисунок 9. Конфигурация отладки

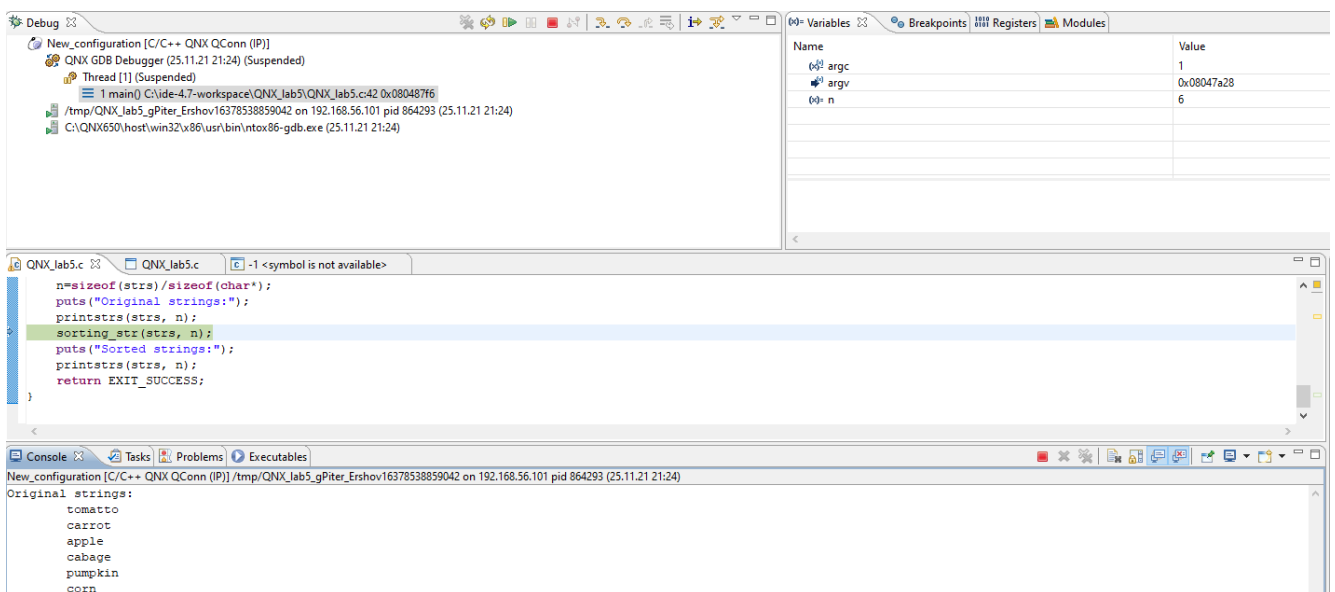


Рисунок 10. Окно отладки

**3.5. Создать конфигурацию для обычного запуска без отладочной информации и выполнить запуск приложения. Сравнить размеры исполняемых файлов с отладочной информацией и без нее (пункт Properties в контекстном меню нужного файла в представлении Project Explorer**

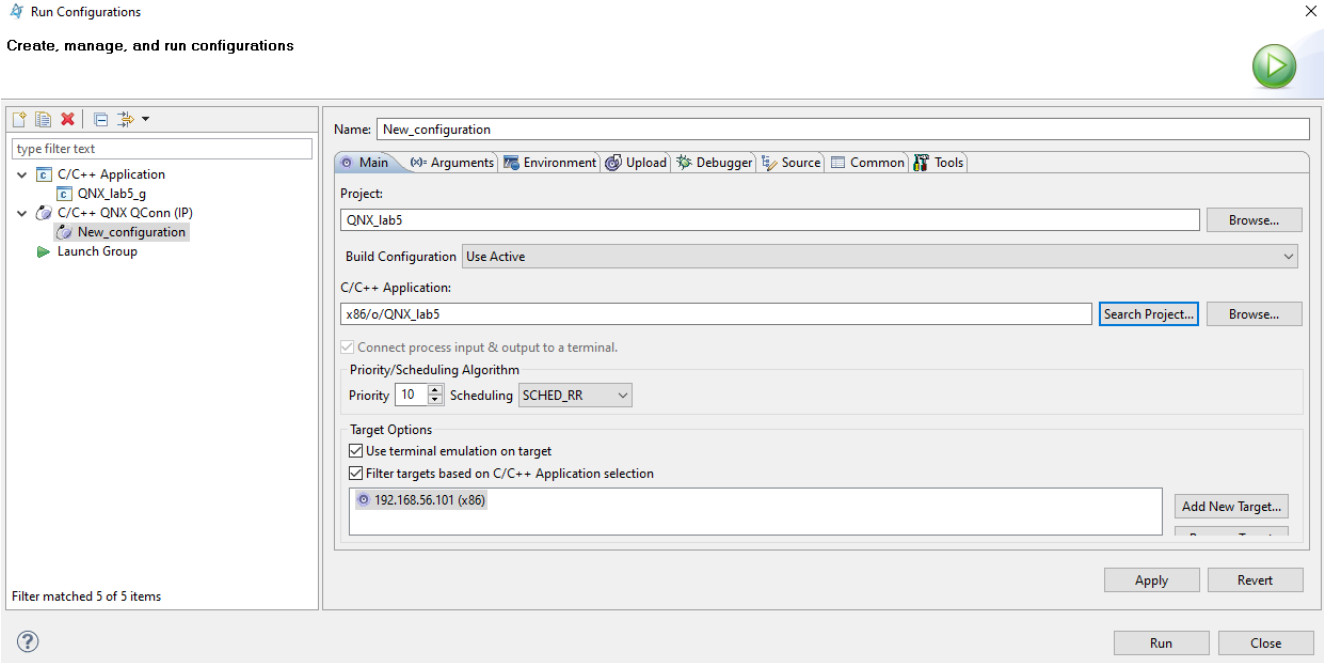


Рисунок 11. Конфигурация простого запуска (без отладки)

Resource	Resource
Path: /QNX_lab5/x86/o-g/QNX_lab5_g	Path: /QNX_lab5/x86/o/QNX_lab5
Type: File	Type: File
Location: C:\ide-4.7-workspace\QNX_lab5\x86\o-g\QNX_lab5_g	Location: C:\ide-4.7-workspace\QNX_lab5\x86\o\QNX_lab5
Size: 7 472 bytes	Size: 5 632 bytes

Рисунок 12. Сравнение размеров исполняемого файла отладки (слева) и обычного запуска (справа)

Как хорошо видно на рисунке 12, размер исполняемого файла отладки заметно больше.



**3.6.** Выполнить профилирование методами “Function Instrumentation profiling” и “Sampling and Call Count instrumentation profiling”. Из результатов получить время выполнения функций программы для обоих методов. Занести данные по времени в таблицу

Метод профилирования	Function Instrumentation profiling	Sampling and Call Count instrumentation profiling
Функция	Затраченное время	
main	0,068 мс	
sorting_str	0,001 мс	
print_str	0,987 мс, 1,621 мс	

## 4. Код программы

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

char *strs[]={"tomatto", "carrot", "apple", "cabage", "pumpkin", "corn"};

void printstrs(char **mas, int n)
{
    int i;
    for(i=0; i<n; i++)
        printf("\t%s\n", mas[i]);
}

void sorting_str(char **mas, int n)
{
    int i, j, min;
    char *temp;
    for(i=0; i<n-1; i++)
    {
        min=i;
        for(j=0; j<n; j++)
        {
            if(strcmp(mas[j], mas[min])<0)
                min=j;
        }
        temp=mas[i];
        mas[i]=mas[min];
        mas[min]=temp;
    }
}

int main(int argc, char *argv[])
{
    int n;
    n=sizeof(strs)/sizeof(char*);
    puts("Original strings:");
    printstrs(strs, n);
    sorting_str(strs, n);
    puts("Sorted strings:");
    printstrs(strs, n);
    return EXIT_SUCCESS;
}
```