МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО

ФАИС

Кафедра «Информатика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5
по дисциплине «**Операционные системы и среды**»

на тему: «**Знакомство со стандартной утилитой Gnu make для построения проектов в ОС Unix/Linux**»

Выполнил: студент гр. ИП-32
Прокопенко А. Р.
Принял: преподаватель
Процкая М. А.

Дата сдачи отчета: _____
Дата допуска к защите: _____
Дата защиты: _____

Гомель 2022

Цель: ознакомиться с техникой компиляции программ на языке программирования C (C++) в среде ОС семейства Unix/Linux, а также получить практические навыки использования утилиты GNU make для сборки проекта.

Задание

1. Ознакомиться с теоретическим материалом.
2. Воспользоваться утилитой make для автоматизации сборки проектов из лабораторной работы №4 «Инструментальные средства ращзработки Linux».
3. Создать make-файл с высоким уровнем автоматизированной обработки исходных файлов программы согласно следующим условиям:

- имя скомпилированной программы (выполняемый или бинарный файл), флаги компиляции и имена каталогов с исходными файлами и бинарными файлами (каталоги, src, bin, и т.п.) задаются с помощью переменных в makefile;
- зависимости исходных файлов на языке C (C++) и цели в make-файле должны формироваться динамически;
- наличие цели clean, удаляющей временные файлы;
- каталог проекта должен бьыть структурирован следующим образом:

src – каталог с исходными файлами;
bin – каталог с бинарными файлами (скомпилированными);
makefile.

- структура данных (STL) - forward list;
- функция обработки структурированных данных — определение максимального среди отрицательных элементов;
- начальная сортировка — обменом;
- конечная сортировка — быстрая.

Все функции размещаются в отдельных файлах.

4. Выполнить программу (скомпилировать, при необходимости отладить) для первого варианта сортировки данных.
5. Изменить тип сортировки и выполнить программу.
6. Показать, что при изменении одного исходного файла и последующем вызове make будут исполнены только необходимые команды компиляции (неизмененные файлы перекомпилированы не будут) и изменены атрибуты и/или размер объектных файлов (файлы с расширением o).

Выполнение

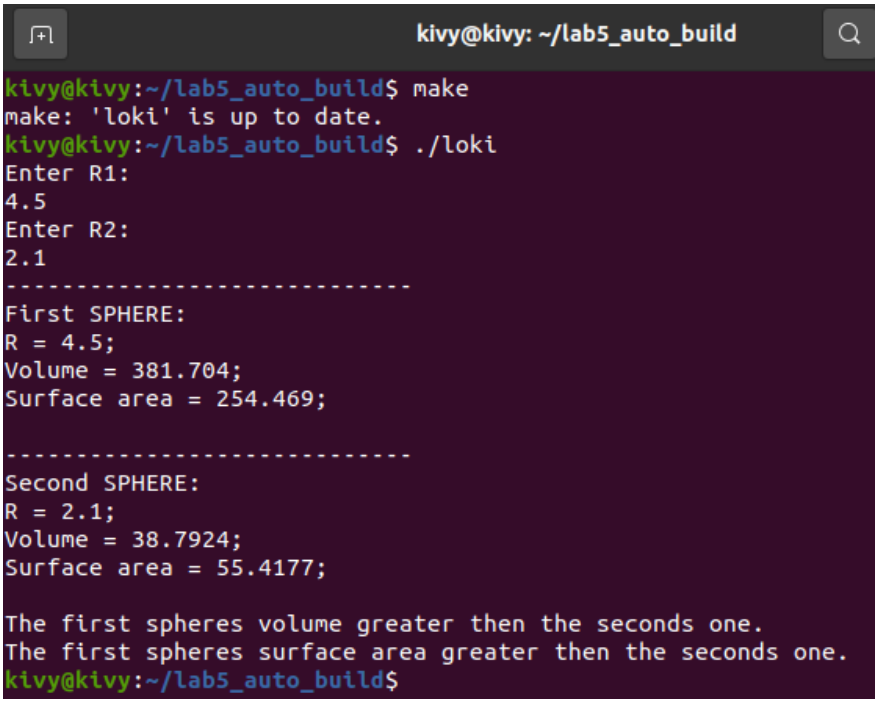
1. sudo apt get make
2. Переместим .cpp- и .h-файлы в каталог ~/lab5_auto_build/src и создадим каталог bin для скомпилированных .cpp-файлов.

Создадим makefile со следующим содержимым:

makefile:

```
loki: ./bin/main.o ./bin/calculate_volume.o ./bin/calculate_surface_area.o ./bin/get_radius.o ./bin/set_radius.o ./bin/sphere.o
	sudo g++ $^ -o $@
./bin/%.o: ./src/%.cpp
	sudo g++ -c $^ -o $@
```

Далее просто запустим в терминале находясь в каталоге ~/lab5_auto_build команду make.
Получим файл loki, который можно запустить командой ./loki.



Аналогичным образом можно было бы поступить со вторым (более легким) проектом из лабораторной работы №4.

3. **makefile**:

```
program_name:= run
compiled_files_catalog:= ./bin/
sourced_files_catalog:= ./src/
flags:= -o
obj_list:= $(compiled_files_catalog)main.o $(compiled_files_catalog)add_item.o $(compiled_files_catalog)change_value_by_id.o $(compiled_files_catalog)get_forwards_lst_size.o $(compiled_files_catalog)get_item_by_id.o $(compiled_files_catalog)hoara_sort.o $(compiled_files_catalog)my_fun.o $(compiled_files_catalog)remove_item_by_id.o $(compiled_files_catalog)show_flist.o $(compiled_files_catalog)sort_by_bubble.o
$(program_name): $(obj_list)
	sudo g++ $^ $(flags) $@
$(compiled_files_catalog)%.o: $(sourced_files_catalog)%.cpp
	sudo g++ -c $^ $(flags) $@
clean:
	sudo rm $(program_name)
	sudo rm $(compiled_files_catalog)*.o
```

functionality.h:

```
#pragma once

#include <forward_list>
#include <list>
#include <algorithm>
#include <stdlib.h>

using namespace std;
```

```
void show_flst(forward_list<int> numbers);
int get_forwards_lst_size(forward_list<int> numbers);
int get_item_by_id(forward_list<int> numbers, int id);
forward_list<int> change_value_by_id(forward_list<int> numbers, int id, int new_value);
forward_list<int> add_item(forward_list<int> numbers, int new_item);
forward_list<int> remove_item_by_id(forward_list<int> numbers, int id);
forward_list<int> sort_by_bubble(forward_list<int> numbers);
void hoara_sort(forward_list<int> &numbers, int left, int right);
void my_fun(forward_list<int> numbers);
```

sort_by_bubble.cpp:

```
#include "functionality.h"

forward_list<int> sort_by_bubble(forward_list<int> numbers)
{
	int size = get_forwards_lst_size(numbers);
	for (int i = 0; i < size; i++)
	{
		for (int j = 0; j < size - 1; j++)
		{
			int a = get_item_by_id(numbers, j);
			int b = get_item_by_id(numbers, j + 1);
			if (a > b)
			{
				numbers = change_value_by_id(numbers, j, b);
				numbers = change_value_by_id(numbers, j + 1, a);
			}
		}
	}
	return numbers;
}
```

remove_item_by_id.cpp:

```
#include "functionality.h"

forward_list<int> remove_item_by_id(forward_list<int> numbers, int id)
{
	int size = get_forwards_lst_size(numbers);
	if (id >= 0 && id < size)
	{
		auto current = numbers.begin();
		id--;
		while (id--)
			current++;
		numbers.erase_after(current);
	}
	return numbers;
}
```

add_item.cpp:

```
#include "functionality.h"

forward_list<int> add_item(forward_list<int> numbers, int new_item)
{
	numbers.push_front(new_item);
	return numbers;
}
```

change_value_by_id.cpp:

```
#include "functionality.h"

forward_list<int> change_value_by_id(forward_list<int> numbers, int id, int new_value)
{
	int size = get_forwards_lst_size(numbers);
	if (id >= 0 && id < size)
	{
		auto current = numbers.begin();
		while (id--)
			current++;
		*current = new_value;
	}
	return numbers;
}
```

get_item_by_id.cpp:

```
#include "functionality.h"

int get_item_by_id(forward_list<int> numbers, int id)
{
	// it is considered that id is an integer between 0 and size - 1...
	int size = get_forwards_lst_size(numbers);
	if (id >= 0 && id < size)
	{
		auto current = numbers.begin();

		while (id--)
			current++;
		return *current;
	}
	return -1;
}
```

get_forwards_lst_size.cpp:

```
#include "functionality.h"

int get_forwards_lst_size(forward_list<int> numbers)
{
```

```
	return distance(numbers.begin(), numbers.end());
}
```

hoara_sort.cpp:

```
#include "functionality.h"

void hoara_sort(forward_list<int> &numbers, int left, int right)
{
	int pivot;
	int l_hold = left;
	int r_hold = right;
	pivot = get_item_by_id(numbers, left);
	while(left < right)
	{
		while((get_item_by_id(numbers, right) >= pivot) && (left < right))
			right--;
		if (left != right)
		{
			int a = get_item_by_id(numbers, left);
			int b = get_item_by_id(numbers, right);
			numbers = change_value_by_id(numbers, left, b);
			left++;
		}
		while((get_item_by_id(numbers, left) <= pivot) && (left < right))
			left++;
		if (left != right)
		{
			int a = get_item_by_id(numbers, left);
			int b = get_item_by_id(numbers, right);
			numbers = change_value_by_id(numbers, right, a);
			right--;
		}
	}
	numbers = change_value_by_id(numbers, left, pivot);
	pivot = left;
	left = l_hold;
	right = r_hold;
	if(left < pivot)
		hoara_sort(numbers, left, pivot - 1);
	if(right > pivot)
		hoara_sort(numbers, pivot + 1, right);
}
```

my_fun.cpp:

```
#include "functionality.h"
#include <iostream>


void my_fun(forward_list<int> numbers)
{
```

```
	list<int> negative_elements;
	for (int n : numbers)
		if (n < 0)
			negative_elements.push_back(n);
	if (negative_elements.empty())
		cout << "No negative elements... " << endl;
	else
	{
		std::list<int>::const_iterator max_negative = max_element(negative_elements.begin(), negative_elements.end());
		cout << "Max negative is " << *max_negative << endl;
	}
}
```

show_flist.cpp:

```
#include "functionality.h"
#include <iostream>

void show_flst(forward_list<int> numbers)
{
	cout << "My FORWARD LIST: " << endl;
	auto current = numbers.begin();
   auto end = numbers.end();
   while (current != end)
   {
      std::cout << *current << endl;
      current++;
   }
}
```

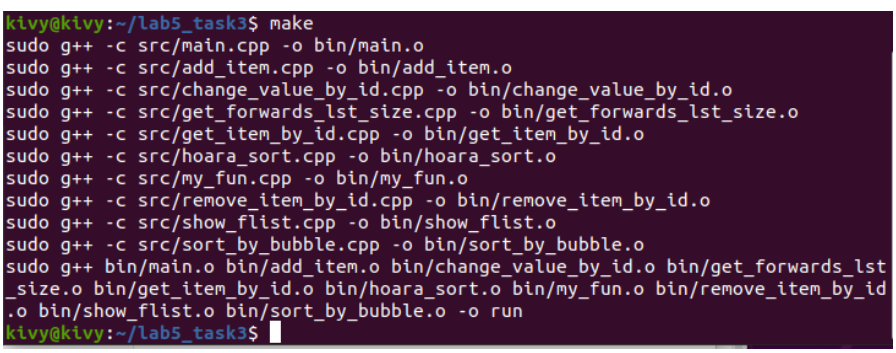
main.cpp:

```
#include <iostream>
#include "functionality.h"


int main()
{
	forward_list<int> numbers({1,100,1,6,5,4,3,2,1});
	bool flag = true;
	while (flag)
	{
		printf("\033c");
		cout << "1. Show elements." << endl;
		cout << "2. Add an element in front." << endl;
		cout << "3. Remove an element by id." << endl;
		cout << "4. Sort elements." << endl;
		cout << "5. Use my finction." << endl;
		cout << "6. Exit." << endl;
		int choice;
		cin >> choice;
```

```
            switch (choice)
            {
            case 1:
                show_flst(numbers);
                break;
            case 2:
                int elem;
                cout << "Enter new value: ";
                cin >> elem;
                numbers = add_item(numbers, elem);
                break;
            case 3:
                int id;
                cout << "Enter id: ";
                cin >> id;
                numbers = remove_item_by_id(numbers, id);
                break;
            case 4:
                //cout << "Sort by bubble ----> " << endl;
                //numbers = sort_by_bubble(numbers);
                cout << "Quick sort (Hoara) ----> " << endl;
                hoara_sort(numbers, 0, get_forwards_lst_size(numbers) - 1);
                break;
            case 5:
                my_fun(numbers);
                break;
            case 6:
                flag = false;
                break;
            default:
                cout << "Wrong data. Try again." << endl;
                break;
            }
            cout << "Press any key to continue..." << endl;
            char key;
            cin >> key;
        }
        return 0;
}
```

4. make



```
1. Show elements.
2. Add an element in front.
3. Remove an element by id.
4. Sort elements.
5. Use my finction.
6. Exit.
1
My FORWARD LIST:
1
100
1
6
5
4
3
2
1
Press any key to continue...
```

5, 6. Зафиксируем состояние main.cpp файла.

```
case 4:
    //cout << "Sort by bubble ----> " << endl;
    //numbers = sort by bubble(numbers);
    cout << "Quick sort (Hoara) ----> " << endl;
    hoara sort(numbers, 0, get forwards lst size(numbers) - 1);
    break;
```

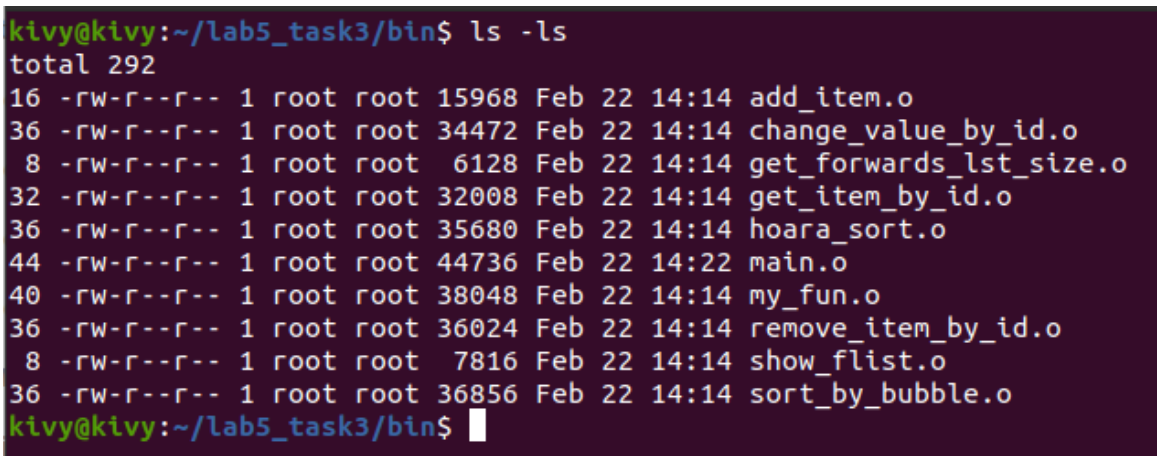
Зафиксируем размеры объектных файлов до пересборки:

```
kivy@kivy:~/lab5_task3/bin$ ls -ls
total 292
16 -rw-r--r-- 1 root root 15968 Feb 22 14:14 add_item.o
36 -rw-r--r-- 1 root root 34472 Feb 22 14:14 change_value_by_id.o
 8 -rw-r--r-- 1 root root  6128 Feb 22 14:14 get_forwards_lst_size.o
32 -rw-r--r-- 1 root root 32008 Feb 22 14:14 get_item_by_id.o
36 -rw-r--r-- 1 root root 35680 Feb 22 14:14 hoara_sort.o
44 -rw-r--r-- 1 root root 44776 Feb 22 14:14 main.o
40 -rw-r--r-- 1 root root 38048 Feb 22 14:14 my_fun.o
36 -rw-r--r-- 1 root root 36024 Feb 22 14:14 remove_item_by_id.o
 8 -rw-r--r-- 1 root root  7816 Feb 22 14:14 show_flist.o
36 -rw-r--r-- 1 root root 36856 Feb 22 14:14 sort_by_bubble.o
kivy@kivy:~/lab5_task3/bin$
```

Заменим вид сортировки в main.cpp файле и пересоберем проект.

```
case 4:
    cout << "Sort by bubble ----> " << endl;
    numbers = sort by bubble(numbers);
    //cout << "Quick sort (Hoara) ----> " << endl;
    //hoara sort(numbers, 0, get forwards lst size(numbers) - 1);
    break;
```

```
kivy@kivy:~/lab5_task3$ make
sudo g++ -c src/main.cpp -o bin/main.o
[sudo] password for kivy:
sudo g++ bin/main.o bin/add_item.o bin/change_value_by_id.o bin/get_forwards_lst
_size.o bin/get_item_by_id.o bin/hoara_sort.o bin/my_fun.o bin/remove_item_by_id
.o bin/show_flist.o bin/sort_by_bubble.o -o run
kivy@kivy:~/lab5_task3$
```



Файл main.o в двух случая весит по-разному (44776 и 44736), что говорит об его изменении. Также факт неизменности остальных файлов подтверждается записями во время сборки и неизменным весом.

Вывод: в процессе выполнения лабораторной работы я ознакомился с техникой компиляции программ на языке программирования C (C++) в среде ОС семейства Unix/Linux, а также получил практические навыки использования утилиты GNU make для сборки проекта.