Министерство Образования Республики Беларусь

Учреждение образования

«Гомельский государственный технический университет

имени П.О.Сухого»

Кафедра «Информатика»

**Лабораторная работа № 5**

по дисциплине: **«Операционные Системы и Среды»**

**Знакомство со стандартной утилитой**

**Gnu mаке для построения проектов в ОС Unix/Linux**

Выполнил студент

группы ИП-31

*Казимов А.Р.*

Проверил старший преподаватель

*Самовендюк Н.В.*

Гомель 2019 г.

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

**Практическая часть:**

1. Ознакомиться с теоретическим материалом.

2. Воспользоваться утилитой make для автоматизации сборки проектов из лабораторной работы №4 «Инструментальные средства разработки 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 С (С++) и цели в make-файле должны формироваться динамически;

• наличие цели clean, удалающей временные файлы;

• каталог проекта должен быть структурирован следующим образом:

src — каталог с исходными файлами;

bin — каталог с бинарными файлами (скомпилированными);

makejile.

|  |  |
| --- | --- |
| 10 | Forward\_List |
| Определение максимального среди отрицательных элементов |
| Пузырек 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.

5. 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 .о).

**Код программы:**

**main.cpp:**

#include <iostream>

#include <forward\_list>

#include <iterator>

#include "searchMaxInForwardList.h"

using namespace std;

int main() {

setlocale(LC\_ALL, "Russian");

forward\_list<int> numbers = { 1, -10, 2, 3,-15, 5,-1, -3, 4, -3, 5 };

print(numbers);

int max = getMaxValue(numbers);

if (max == INT32\_MIN) {

cout << "Нет отрицательных чисел" << endl;

}

else {

cout << "Максимальное число из отрицательных чисел = " << max << endl;

}

sort(&numbers, 0, size(numbers) - 1);

print(numbers);

system("pause");

return 0;

}

**searchMaxInForwardList.h:**

#include <forward\_list>

#include <iostream>

using namespace std;

int getMaxValue(forward\_list<int> fl);

void print(forward\_list<int> list);

void swap(forward\_list<int>::iterator first, forward\_list<int>::iterator second);

int size(forward\_list<int> list);

void sort(forward\_list<int> \*list, int left, int right);

int get(forward\_list<int> list, int index);

void swap(forward\_list<int> \*list, int x, int y);

**searchMaxInForwardList.cpp:**

#include "searchMaxInForwardList.h"

int getMaxValue(forward\_list<int> fl) {

int max = INT32\_MIN;

for (int n : fl)

if (n < 0 && n > max) {

max = n;

}

return max;

}

**bubleSort.cpp:**

#include "searchMaxInForwardList.h"

void sort(forward\_list<int> \*list, int left, int right) // сортировка пузырьком

{

cout << "Buble" << endl;

int temp = 0; // временная переменная для хранения элемента массива

bool exit = false; // болевая переменная для выхода из цикла, если массив отсортирован

int i = 0;

while (!exit) // пока массив не отсортирован

{

exit = true;

auto begin = list->begin();

auto end = list->end();

while (begin != end) {

auto next = begin;

next++;

if (next !=end && \*begin > \*next) // сравниваем два соседних элемента

{

// выполняем перестановку элементов массива

swap(begin, next);

exit = false; // на очередной итерации была произведена перестановка элементов

}

i++;

begin++;

}

}

}

**quickSort.cpp:**

#include "searchMaxInForwardList.h"

void sort(forward\_list<int> \*list, int left, int right) {

int i = left, j = right;

int tmp;

int pivot = get(\*list,(left + right) / 2);

/\* partition \*/

while (i <= j) {

while (get(\*list,i) < pivot)

i++;

while (get(\*list, j) > pivot)

j--;

if (i <= j) {

swap(list, i, j);

i++;

j--;

}

};

/\* recursion \*/

if (left < j)

sort(list, left, j);

if (i < right)

sort(list, i, right);

}

**auxiliaryFunction.cpp:**

#include "searchMaxInForwardList.h"

void print(forward\_list<int> list) {

auto begin = list.begin();

auto end = list.end();

while (begin != end)

{

cout << \*begin << " ";

begin++;

}

cout << endl;

}

int size(forward\_list<int> list) {

int count = 0;

auto begin = list.begin();

auto end = list.end();

while (begin != end)

{

begin++;

count++;

}

return count;

}

void swap(forward\_list<int>::iterator first, forward\_list<int>::iterator second) {

if (first != second) {

int temp = \*first;

\*first = \*second;

\*second = temp;

}

}

int get(forward\_list<int> list, int index) {

int i = 0;

auto begin = list.begin();

auto end = list.end();

while (begin != end && i != index)

{

begin++;

i++;

}

return \*begin;

}

void swap(forward\_list<int> \*list, int x, int y) {

auto begin = list->begin();

auto end = list->end();

forward\_list<int>::iterator first = begin;

forward\_list<int>::iterator second = begin;

int i = 0;

while (begin != end)

{

if (i == x) {

first = begin;

}

if (i == y) {

second = begin;

}

begin++;

i++;

}

if (first != second) {

int temp = \*first;

\*first = \*second;

\*second = temp;

}

}

**MakeFile:**

path\_to\_src:=~/linux\_lab\_5/src

path\_to\_bin:=~/linux\_lab\_5/bin/

path\_to\_base:=~/linux\_lab\_5

name\_program:=task

main\_function: create\_o\_files create\_program\_with\_buble\_sort

create\_o\_files: main.o auxiliaryFunction.o searchMaxInForwardList.o bubleSort.o

create\_program\_with\_buble\_sort:

g++ ${path\_to\_bin}main.o ${path\_to\_bin}auxiliaryFunction.o ${path\_to\_bin}searchMaxInForwardList.o ${path\_to\_bin}bubleSort.o -o ${name\_program} -lm

sudo mv ${name\_program} ${path\_to\_bin}

create\_program\_with\_quick\_sort:

g++ ${path\_to\_bin}main.o ${path\_to\_bin}auxiliaryFunction.o ${path\_to\_bin}searchMaxInForwardList.o ${path\_to\_bin}quickSort.o -o ${name\_program} -lm

sudo mv ${name\_program} ${path\_to\_bin}

change\_sort\_quick\_sort: delete\_buble\_sort quickSort.o create\_program\_with\_quick\_sort

change\_sort\_buble\_sort: delete\_quick\_sort bubleSort.o create\_program\_with\_buble\_sort

delete\_buble\_sort:

sudo rm ${path\_to\_bin}bubleSort.o

delete\_quick\_sort:

sudo rm ${path\_to\_bin}quickSort.o

main.o:

g++ -c ${path\_to\_src}/main.cpp

sudo mv main.o ${path\_to\_bin}

auxiliaryFunction.o:

g++ -c ${path\_to\_src}/auxiliaryFunction.cpp

sudo mv auxiliaryFunction.o ${path\_to\_bin}

searchMaxInForwardList.o:

g++ -c ${path\_to\_src}/searchMaxInForwardList.cpp

sudo mv searchMaxInForwardList.o ${path\_to\_bin}

bubleSort.o:

g++ -c ${path\_to\_src}/bubleSort.cpp

sudo mv bubleSort.o ${path\_to\_bin}

quickSort.o:

g++ -c ${path\_to\_src}/quickSort.cpp

sudo mv quickSort.o ${path\_to\_bin}

clean:

sudo rm ${path\_to\_bin}\*.o

sudo rm ${path\_to\_bin}task

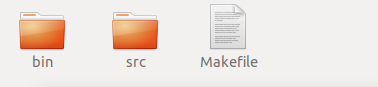


Рисунок 1 – Структура проекта

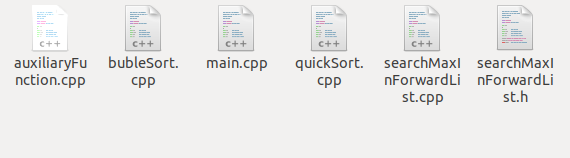


Рисунок 2 – Структура папки src

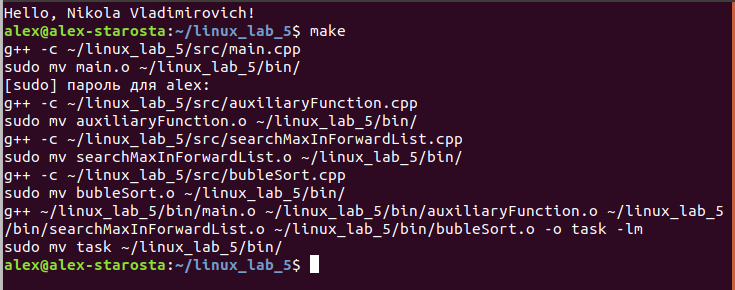


Рисунок 3 – Команда make

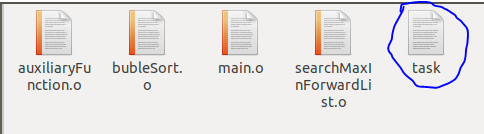


Рисунок 4 – Структура папки bin после команды make

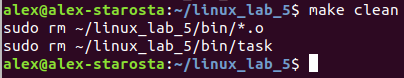


Рисунок 5 – Выполнение цели clean

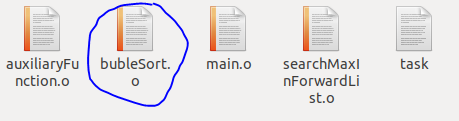


Рисунок 6 – Стандартная сортировка

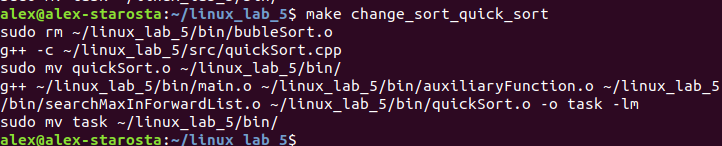


Рисунок 7 – Замена сортировки

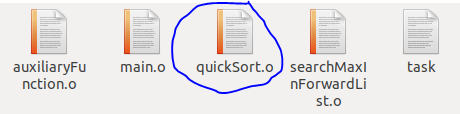


Рисунок 8 – Структура папки bin после замены

**Вывод:** изучил инструментальные средства разработки Linux: компиляторы сс/g++/gcc.