# Министерство Образования Республики Беларусь

Учреждение образования

«Гомельский государственный технический университет

имени П.О.Сухого»

Кафедра: «Информатика»

по курсу: «Операционные системы и среды»

Лабораторная работа №5

«Знакомство со стандартной утилитой

Gnu mаке для построения проектов в ОС Unix/Linux»

Выполнил: студент группы ИП-31

Земчёнок Вадим Геннадьевич

Допуск к защите: Проверил: старший преподаватель

Дата защиты: Самовендюк Николай Владимирович

Гомель 2020 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.

**Задание 1.**

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.

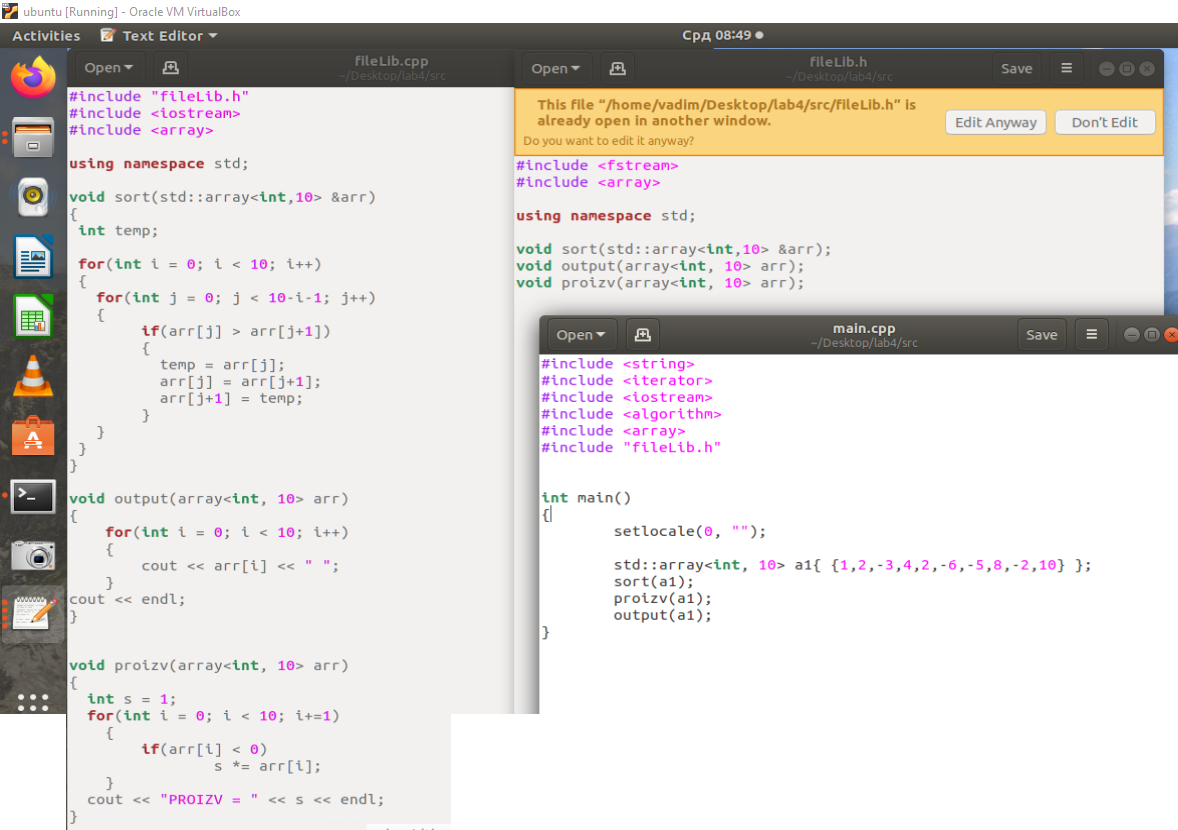
|  |  |  |  |
| --- | --- | --- | --- |
| Вариант | Структура данных (STL) | Функции обработки (таблица 2) | Тип сортировки  (таблица 3) |
| 7 | Array | Произведение отрицательных элементов на нечетных местах | начальная – обменом, конечная – выбором максимального элемента |

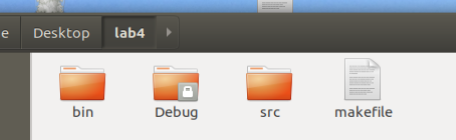
Все функции размещаются в отдельных файлах.

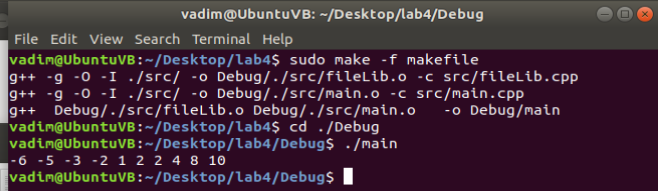
4. Выполнить программу (скомпилировать, при необходимости отладить) для первого варианта сортировки данных.

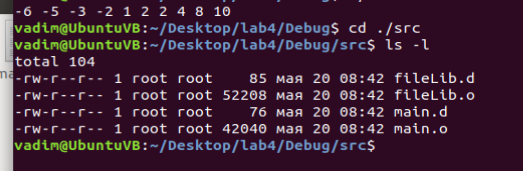
5. Изменить тип сортировки и выполнить программу.

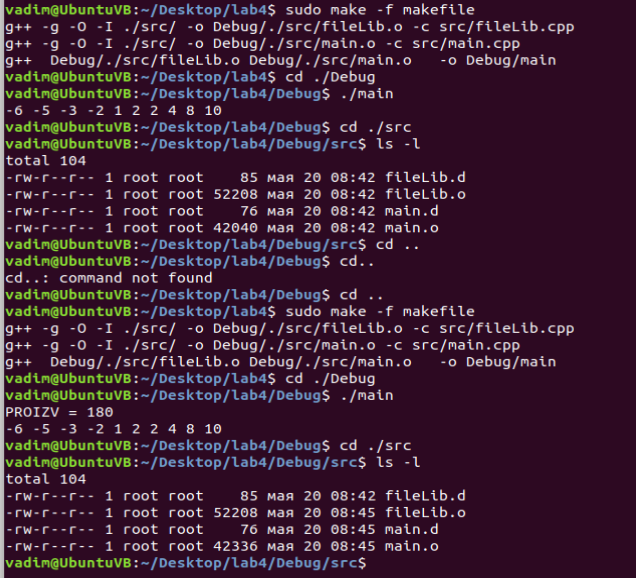
5. Показать, что при изменении одного исходного файла и последующем вызове make будут исполнены только необходимые команды компиляции (неизмененные файлы перекомпилированы не будут) и изменены атрибуты и/или размер объектных файлов (файлы с расширением .о).

****

****

****

****

****

**Листинг make файла**

#### #######################################################

#

# MAKEFILE FOR C/C++ PROJECT

# Author: swm8023 <swm8023@gmail.com>

# Date: 2014/01/30

#

#### #######################################################

.PHONY: all clean

all:

# annotation when release version

DEBUG := y

TARGET\_PROG := main

# project directory

DEBUG\_DIR := ./Debug

RELEASE\_DIR := ./Release

BIN\_DIR := $(if $(DEBUG), $(DEBUG\_DIR), $(RELEASE\_DIR))

# shell command

CC := gcc

CXX := g++

RM := rm -rf

MKDIR := mkdir -p

SED := sed

MV := mv

# init sources & objects & depends

sources\_all := $(shell find . -name "\*.c" -o -name "\*.cpp" -o -name "\*.h")

sources\_c := $(filter %.c, $(sources\_all))

sources\_cpp := $(filter %.cpp, $(sources\_all))

sources\_h := $(filter %.h, $(sources\_all))

objs := $(addprefix $(BIN\_DIR)/,$(strip $(sources\_cpp:.cpp=.o) $(sources\_c:.c=.o)))

deps := $(addprefix $(BIN\_DIR)/,$(strip $(sources\_cpp:.cpp=.d) $(sources\_c:.c=.d)))

# create directory

$(foreach dirname,$(sort $(dir $(sources\_c) $(sources\_cpp))),\

$(shell $(MKDIR) $(BIN\_DIR)/$(dirname)))

# complie & link variable

CFLAGS := $(if $(DEBUG),-g -O, -O2)

CFLAGS += $(addprefix -I ,$(sort $(dir $(sources\_h))))

CXXFLAGS = $(CFLAGS)

LDFLAGS :=

LOADLIBES += #-L/usr/include/mysql

LDLIBS += #-lpthread -lmysqlclient

# add vpath

vpath %.h $(sort $(dir $(sources\_h)))

vpath %.c $(sort $(dir $(sources\_c)))

vpath %.cpp $(sort $(dir $(sources\_cpp)))

# generate depend files

# actually generate after object generated, beacasue it only used when next make)

ifneq "$(MAKECMDGOALS)" "clean"

sinclude $(deps)

endif

# make-depend(depend-file,source-file,object-file,cc)

define make-depend

$(RM) $1; \

$4 $(CFLAGS) -MM $2 | \

$(SED) 's,\($(notdir $3)\): ,$3: ,' > $1.tmp; \

$(SED) -e 's/#.\*//' \

-e 's/^[^:]\*: \*//' \

-e 's/ \*\\$$//' \

-e '/^$$/ d' \

-e 's/$$/ :/' < $1.tmp >> $1.tmp; \

$(MV) $1.tmp $1;

endef

# rules to generate objects file

$(BIN\_DIR)/%.o: %.c

@$(call make-depend,$(patsubst %.o,%.d,$@),$<,$@,$(CC))

$(CC) $(CFLAGS) -o $@ -c $<

$(BIN\_DIR)/%.o: %.cpp

@$(call make-depend,$(patsubst %.o,%.d,$@),$<,$@,$(CXX))

$(CXX) $(CXXFLAGS) -o $@ -c $<

# add-target(target,objs,cc)

define add-target

REAL\_TARGET += $(BIN\_DIR)/$1

$(BIN\_DIR)/$1: $2

$3 $(LDFLAGS) $$^ $(LOADLIBES) $(LDLIBS) -o $$@

endef

# call add-target

$(foreach targ,$(TARGET\_PROG),$(eval $(call add-target,$(targ),$(objs),$(CXX))))

all: $(REAL\_TARGET) $(TARGET\_LIBS)

clean:

$(RM) $(BIN\_DIR)

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