Лабораторная работа № 5

Сборка проекта из командной строки

Цель работы: познакомиться с этапами подготовки исполняемого кода, утилитой make, научиться создавать простейшие варианты Makefile, получить навык компиляции и сборки в командной строке проекта, состоящего из нескольких модулей.

Задание

1. Получить вариант задачи для программирования у преподавателя. Проанализировать задачу, разбить ее на составные части, составить план решения задачи и способ проверки. Первый вариант решения может содержать лишь несколько строчек кода, например, ввод исходных данных и их вывод. В итоговом коде должно быть не менее двух функций, одна их которых — main(). Выполнение заданий сопровождается сохранением исходного кода проекта в репозитории. Метки (снимок N) указывают моменты выполнения работы, которые необходимо дополнительно зафиксировать в виде копий экрана и представить в отчёте.
2. Изучить справочный материал к данной работе. Он содержит достаточный объем материала для выполнения заданий.
3. Запустить программу-эмулятор терминала (или перейти в один из текстовых терминалов Linux и войти в систему).
4. В рабочей копии репозитория создать папку для хранения проекта (условное имя — myprj) и сохранить ее в репозитории. Перейти в папку myprj.
5. Написать код программы в текстовом редакторе (Vim, Gedit, Mousepad и т. д.) и сохранить файл в папке проекта (условное название файла main.c). Сохранить файл в репозитории (снимок 1).
6. Выполнить обработку файла main.c препроцессором, результат обработки сохранить в файле с произвольным именем. Проанализировать полученный

текст. Какую информацию содержит файл? Что исчезло из файла по сравнению с исходным текстом? Сравнить размеры исходного и полученного файлов. Просмотреть статус репозитория (снимок 2). Удалить полученный файл.

7. Выполнить компиляцию программы из командной строки без образования объектного модуля с помощью компилятора gcc. Найти в папке myprj исполняемый файл, запустить его на выполнение, выполнить тестирование программы.
8. Просмотреть статус репозитория, выполнить сохранение в репозитории только файла с исходным кодом (снимок 3).
9. Выполнить пошаговую подготовку исполняемого файла с именем mypr1 с помощью компилятора gcc. Просмотреть содержимое папки myprj, объяснить назначение каждого файла. Запустить и протестировать программу (снимок 4). Удалить результаты компиляции.
10. Написать Makefile, целью которого будет получение исполняемого файла mypr2 из main.c, и сохранить его в папке myprj. Включить Makefile под управление репозитория.
11. Получить исполняемый файл mypr2 с помощью утилиты make. Просмотреть содержимое папки myprj, статус репозитория, выполнить и протестировать mypr2.
12. Отредактировать Makefile: добавить новую цель «Удаление из каталога myprj объектных файлов». Выполнить удаление объектных файлов с помощью утилиты make. Проверить содержимое каталога myprj (снимок 5). Сохранить в репозитории main.c и Makefile.
13. Разбить исходный код программы на модули так, чтобы в главном модуле находилась функция main(), в модуле пользователя — все остальные функции. Файлы с исходным кодом модулей ввести по управление svn.
14. Внести изменения в Makefile, чтобы он обеспечивал сборку проекта, состоящего из нескольких модулей, цель — исполняемый файл mypr3.

Выполнить сборку проекта с помощью утилиты make, выполнить и протестировать программу.

15. Просмотреть статус репозитория (снимок 6). Сохранить результаты в репозитории (снимок 7).
16. Продемонстрировать преподавателю файлы каталога myprj, ревизии, относящиеся к каталогу myprj, файлам main.c, Makefile. При дистанционном выполнении работ следует представить подготовленный в соответствии с рекомендациями отчёт.
17. При дистанционном выполнении работ удалить с помощью Makefile результаты компиляции (снимок 8).

Справочный материал

1. Этапы получения исполняемого кода для программы на C/C++

№	Действие	Программа, выполняющая действие	Результат
1	Обработка исходного текста программы препроцессором	Препроцессор **cpp**	Файл (текстовый) *.i или любой иной тип
2	Компиляция	Компилятор языка C — **gcc** Компилятор языка C++ - **g++**	Файл (двоичный, объектный модуль) *.o *.obj
3	Компоновка (редактирование связей)	Компоновщик (линкер, редактор связей)	Исполняемый (двоичный) файл

2. Команды для компиляции, сборки и исполнения программ, используемые в командной строке.

1. Выполнение готовой программы:

./myprj

2. Обработка файла my.c препроцессором.

Стандартное название препроцессора — cpp. Обычно препроцессор вызывается автоматически перед компиляцией, и результаты его работы не

сохраняются в файл, но препроцессор также можно вызвать отдельно в командной строке, для этого надо указать имя исходного файла и имя файла с результатом обработки (*.i):

cpp my.c result.i

3. Компиляция файла my.c без образования объектного модуля (автоматическая компоновка):

   gcc my.c

   Результат: исполняемый файл получает имя по умолчанию a.out.

   gcc -o myprj my.c

   Ключ -o — отказ от стандартного имени исполняемого файла a.out. Имя исполняемого файла задается после ключа -o как параметр — myprj.

4. Пошаговая подготовка исполняемого файла

- Компиляция

  gcc -c my.c

  Ключ -c означает отказ от автоматической компоновки. Результат — объектный файл my.o.

- Компоновка

  gcc -o myprj my.o

  Компоновка с библиотекой математических функций языка C — libm:

  gcc -o myprj my.o -lm

  (минус эль,эм)

  Здесь ключ -l (строчная латинская буква эль) указывает, что для компоновки следует использовать библиотеку m, что соответствует математической библиотеке libm.

5. Пошаговая подготовка исполняемого кода для программы с модулем пользователя

   Исходный код находится в файлах main.c, modul.c, modul.h.

- Компиляция - каждый модуль компилируется отдельно:

  gcc -c main.c

gcc -c modul.c

В результате будут получены файлы main.o и modul.o

- Компоновка (исполняемый файл myproj):

gcc -o myproj main.o modul.o

6. Компиляция с использованием Makefile.

Утилита make — это программа автоматической сборки текста из нескольких файлов. В частности, make используется для автоматической сборки программ, написанных на C/C++, а также на других языках в *nix — системах.

Алгоритм автоматической сборки:

- Написать исходные (*.c,*.cc) и заголовочные (*.h) файлы.
- Подготовить make-файлы, содержащие сведения о проекте. По умолчанию make-файлу присваивается имя makefile, Makefile, GNUmakefile, но можно использовать нестандартное имя, которое указывается при вызове make после ключа -f.
- Вызвать утилиту make, которая собирает целевой проект на основании данных, полученных из make-файла.

make myprj

где myprj — имя целевого проекта, при сборке данные берутся из make-файла со стандартным именем или

make -f mymakef myprj

где mymakef — имя используемого make-файла.

7. Синтаксис make-файла

- Комментарии

#Текст комментария

- Объявления констант

Константы в make-файлах служат для подстановки.

- Целевые связки — устанавливают зависимости между различными частями программы и определяют действия, которые будут выполняться при сборке. В любом make-файле должна быть хотя бы одна целевая связка.

8. Целевая связка компонентов:
   - Имя цели — это может быть файл, после имени цели ставится двоеточие.
   - Список зависимостей — перечисляются через пробел имена файлов или промежуточных целей, если цель ни от чего не зависит, список пуст.
   - Инструкции — это команды, которые должны быть выполнены для достижения цели. Каждая инструкция пишется в новой строке и начинается с символа табуляции.
9. Примерный текст make-файла, выполняющего компиляцию и сборку целевого проекта myprj из main.c и модуля mod.c с заголовком mod.h, а также выполняющего удаление объектных файлов из каталога проекта.

```
#Пример make-файла
myprj: main.o mod.o
	gcc -o myprj main.o mod.o
main.o: main.c
	gcc -c main.c
mod.o: mod.c mod.h
	gcc -c mod.c
clean:
	rm *.o
```

Вопросы

1. Что такое препроцессор?
2. Какие способы компиляции программ вы знаете?
3. Как вызвать компилятор языка Си из командной строки?
4. Как вызвать компилятор языка C++ из командной строки?
5. Для чего используется утилита make?
6. Какое имя может иметь make-файл?
7. Какую структуру имеет make-файл?
8. Как с помощью make выполнить конкретную задачу, например, удаление объектных файлов?

Методические указания к дистанционному выполнению лабораторной работы № 4

Получить вариант задачи для программирования у преподавателя. Проанализировать задачу, разбить ее на составные части, составить план решения задачи. Первый вариант решения может содержать лишь несколько строчек кода, например, ввод исходных данных и их вывод. В итоговом коде должно быть не менее двух функций, одна их которых — main(). Выполнение заданий сопровождается сохранением исходного кода проекта в репозитории. Метки (снимок N) указывают моменты выполнения работы, которые необходимо дополнительно зафиксировать в виде копий экрана и представить в отчёте.

Ввод данных выполняется из файла, вывод — на экран.

Содержание отчета

1. Список участников команды, номер группы.
2. URL-адрес репозитория.
3. Список пользователей репозитория и распределение логинов между участниками. Количество логинов должно соответствовать количеству участников, и каждый должен выполнить хотя бы один коммит.
4. Данные для входа преподавателя (логин, пароль).
5. Операционные системы, используемые для выполнения заданий.
6. Копии экранов с пояснениями.
7. Итоговый текст программы.
8. Текст make-файла.
9. Список тестов программы: исходные данные — ожидаемый результат.
10. Копия экрана исполнения последней версии программы.

Подготовка к выполнению работы в ОС Windows

На базе установленного пакета Code::Blocks+компилятор MinGW.

1. Определите путь к компиляторам.

Например, путь может быть таким, если Code::Blocks устанавливался вместе с

компилятором:

C:\ProgramFiles(x86)\CodeBlocks\MinGW\bin.

2. Перейдите в этот каталог и найдите файлы:

gcc.exe – компилятор языка Си,

g++.exe компилятор языка C++,

mingw32-make.exe – утилита make.

3. Проверьте, что путь в каталог bin прописан в переменной окружения операционной системы path – запустите в режиме командной строки из любого пользовательского каталога какой-нибудь из этих исполняемых файлов. Если команда не будет найдена, то либо пишем полный путь при запуске компилятора и make, либо добавляем этот путь в переменную окружения path.

4. При выполнении лабораторной работы вместо команды make используйте mingw32-make.

На базе пакета компилятора MinGW

Если компилятор установлен отдельно, без Code::Blocks:

https://metanit.com/cpp/tutorial/1.2.php

На базе пакетов MSYS+MinGW (MSYS2+MinGW)

ttps://librebay.blogspot.com/2018/12/install-msys2-for-windows.html