Лабораторная работа № 4

Сборка проекта из командной строки

Цель работы

Познакомиться с этапами подготовки исполняемого кода, утилитой make, получить навык компиляции и сборки проекта проекта, состоящего из нескольких модулей, с помощью командной строки, научиться создавать простейшие варианты Makefile.

Задание

1. Получить вариант задачи у преподавателя. Проанализировать задачу, разбить ее на составные части, определить ее структуру (количество функций, их название).
2. В локальной копии репозитория создать папку для хранения проекта (условное имя - myprj), включить ее под управление svn. Перейти в папку myprj.
3. Написать код программы в текстовом редакторе (Vim, Gedit, Mousepad и т. д.) и сохранить в папке проекта (условное название файла main.c), включить под управление svn. Сохранить проект в репозитории (ревизия № 1).
4. Выполнить обработку файла main.c препроцессором, результат обработки сохранить в файле с произвольным именем. Проанализировать полученный текст.
5. 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.
6. Просмотреть статус репозитория, выполнить сохранение в репозитории только файла с исходным кодом (ревизия № 2).
7. Выполнить пошаговую подготовку исполняемого файла (mypr1) с помощью компилятора gcc. Просмотреть содержимое папки myprj, объяснить назначение каждого файла. Запустить и протестировать программу. Удалить результаты компиляции.
8. Написать Makefile, целью которого будет получение исполняемого файла mypr2 из main.c, и сохранить его в папке myprj. Включить Makefile под управление svn.
9. Получить исполняемый файл mypr2 с помощью утилиты make. Просмотреть содержимое папки myprj, статус репозитория, выполнить и протестировать программу.
10. Изменить свойства папки myprj так, чтобы игнорировались файлы типа *.o.

Проверить статус репозитория.

11. Отредактировать Makefile: добавить новую цель «Удаление из каталога myprj объектных модулей». Выполнить удаление объектных файлов с помощью утилиты make. Проверить содержимое каталога myproj. Сохранить в репозитории main.c и Makefile (ревизия № 3).
12. 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.
13. 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.
14. Просмотреть статус репозитория. Сохранить результаты в репозитории (ревизия № 4).
15. Продемонстрировать преподавателю файлы каталога myprj, ревизии, относящиеся к каталогу myprj, файлам main.c, Makefile.

Справочный материал

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

№	Действие	Программа, выполняющая действие	Результат
1	Обработка исходного текста программы препроцессором	Препроцесор cpp	Файл (текстовый) *.i или любой иной тип
2	Компиляция	Компилятрор 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:

cpp my.c result.i

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

   gcc my.c

   Результат: исполняемый файл a.out.

   gcc -o myprj my.c

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

4. Пошаговая подготовка исполняемого файла

- Компиляция

  gcc -c my.c

  Ключ -c означает отказ от автоматической компоновки. Результат — 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)

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
clear:
	rm *.o
```

Вопросы

1. Что такое препроцессор?
2. Какие способы компиляции программ вы знаете?
3. Как вызвать компилятор из командной строки?
4. Для чего используется утилита make?