Лабораторная работа № U2

Программирование и сборка больших программных продуктов: утилита MAKE

В разных версиях/поставках систем Unix по-разному реализовано управление версиями установленного ПО (установка, обновление, контроль взаимосвязей и т.д.), в некоторых, эти вопросы автоматизированы с помощью т.н. менеджеров пакетов (apt в Ubuntu, rpm в RedHat), в ряде случаев установка ПО заключается в компилировании из исходных кодов «под данную систему».

Программа "Hello World" на C для Unix

Библиотеки C и компилятор C – обычно входят в поставку ОС Unix. C++ бывает что не входит в «дистрибутив».

Файлы с исходным кодом на C должны быть с расширением ***.c*** (для C++ д.б. .cpp)

На linux для компилирования необходимо выполнить команду:

gcc –o имя_исполн_файла имя_исходника.c

для компилирования программы на C++ необходимо вызывать компилятор G++ :

g++ -o имя_исполн_файла имя_исходника.cpp

Задание:

- написать программу "Hello world", откомпилировать и запустить

Примечание:

1) в printf чтобы на терминал текст выводился сразу нужно в конце шаблона строки указать \r\n
2) для запуска программы нужно набирать ./имя_исполн_файла

Сборка больших программных продуктов

При компилировании программ из исходных кодов часто необходимо указывать дополнительные опции для компилятора/линковщика. Кроме программный продукт может состоять из сотен и даже тысяч файлов с исходными кодами.

Для автоматизации процесса компилирования программ существует утилита make:

Для работы в каталоге с исходными кодами должен присутствовать файл с именем Makefile

Пример файла:

```
CC=gcc –m64
CCFLAGS=-O2 –Wall
OBJS=part1.o part2.o part3.o
```

```
all:  ${OBJS}
        $(CC)  $(CCFLAGS)  –o  prog  ${OBJS}
clean:
        rm  *.o  prog
```

где CC, CCFLAGS – опции компилирования (-m64 означает опцию для компилирования 64 разрядной программы, -O2 – опция оптимизации кода при компилировании, -Wall означает что любое предупреждение при компилировании (warning) будет интерпретироваться как ошибка и процесс будет прерван

OBJS – перечень объектных файлов (с расширениями .o), при этом необходимо учитывать что из файла A.c (с исходным кодом) получается файл A.o (автоматически, компилятором)

all: - это умолчательная опция команды make, вызывается либо “make all”, либо просто “make”
clean: - вызывается как “make clean”

Примечание: в строке “$(CC) $(CCFLAGS) –o prog ${OBJS}” в начале строки символ табуляции, а не пробелы.

В случае если программный продукт рассчитан на компилирование и работу под несколькими вариантами операционной системы Unix, например под Linux и FreeBSD, то перед компилированием, Makefile и некоторые исходные файлы, необходимо настроить под используемую операционную систему – задать константы определяющие используемые библиотеки, пути к нужным файлам и т.д.

Ручная настройка является достаточно трудоёмкой задачей, поэтому в современных программах используются специальное ПО для автоматизации этой задачи – например продукт cmake и/или скрипты Configure (широко применяются в ОС linux).

После того как это ПО отработает автоматически создаются Makefile и в программном коде настраиваются константы по которым компилятор будет подключать необходимые библиотеки и задействовать фрагменты исходного кода – конструкции #DEFINE для определения констант для компилятора, код внутри блоков, ограниченных #IFDEF ….. #ENDIF.

Задание:

1. Написать любую программу на C из одного файла C, например классическую “hello world”
2. Написать Makefile реализующий команды:
    a. Для компилирования (all)
    b. Для удаления старых объектных файлов (clean)
    c. Для установки программы (копирования исполняемого файла) в каталог /tmp (install)
3. Модифицировать программу разделив её на два файла C:
    a. Файл1: Функция печати “Hello world” (имя функции НЕ main)
    b. Файл2: Функция main, вызывающая функцию из файла1.
4. Модифицировать Makefile для сборки второй версии программы