Makefile

Опции компиляции

-c - Компилировать или ассемблировать исходные файлы, но не линковать. Стадия ликовки просто не выполняется. Конечный вывод происходит в форме объектного файла для каждого исходного файла. По умолчанию, имя объектного файла делается из имени исходного файла заменой суффикса '.c', '.i', '.s', и.т.д. на '.o'.

-S - Остановиться после собственно компиляции; не ассемблировать. Вывод производится в форме файла с ассемблерным кодом для каждого не ассемблерного входного файла. По умолчанию, имя файла с ассемблерным кодом делается из имени исходного файла заменой суффикса '.c', '.i', и.т.д. на '.s'. Входные файлы, которые не требуют компиляции игнорируются.

-E - Остановиться после стадии препроцессирования; не запускать собственно компилятор. Вывод делается в форме препроцессированного исходного кода, который посылается на стандартный вывод. Входные файлы, которые не требуют препроцессирования игнорируются.

-o файл - Поместить вывод в файл 'файл'. Эта опция применяется вне зависимости от вида порождаемого файла, есть ли это выполнимый файл, объектный файл, ассемблерный файл или препроцессированный C код.

Задача

Написать программу состоящию из 5 файлов (main.c, file1.c, file2.c, file1.h, file2.h).

Задача

Скомпилировать программу

Задача

Создать файл Makefile и записать в нем зависимости исходных файлов.

make

make — утилита, автоматизирующая процесс преобразования файлов из одной формы в другую. Чаще всего это компиляция исходного кода в объектные файлы и последующая компоновка в исполняемые файлы или библиотеки.

Makefile

Makefile — это набор инструкций для программы make, состоящий из целей, зависимостей и команд.

Структура Makefile-а

Простой make-файл состоит из инструкций следующего вида:

```
ЦЕЛЬ ...        : ЗАВИСИМОСТЬ ...
    КОМАНДА
    ...
    ...
```

Компоненты Makefile-а

ЦЕЛЬ обычно представляет собой имя файла, генерируемого программой make; примерами целей являются исполняемые или объектные файлы. Цель может также быть именем выполняемого действия, как, например, 'clean'.

ЗАВИСИМОСТЬ - это файл, используемый как вход для порождения цели. Часто цель зависит от нескольких файлов.

КОМАНДА - это действие, которое выполняет make. Правило может иметь более, чем одну команду - каждую на своей собственной строке. *Важное замечание: вы должны начинать каждую строку, содержащую команды, с символа табуляции*.

Задача

Создать простой Makefile

Пример простого Makefile-а

```
main : main.o file1.o file2.o
        cc main.o file1.o file2.o
main.o : main.cpp file1.h file2.h
        cc -c main.c
file1.o : file1.c file1.h
        cc -c file1.c
file1.o : file2.c file2.h
        cc -c file2.c
```

make clean

clean -удаляет все файлы, которые обычно создаются в результате выполнения программы make.

Пример запуска:

```
make clean
```

Пример make clean

```
clean :
        rm -f *.o
```

Задача

Добавить clean в Makefile (должна удалять все объектные файлы и исполняемы файл)

Стандартные цели Makefile-а

all - порождает все цели верхнего уровня, которые упомянуты в make-файле.

clean - удаляет все файлы, которые обычно создаются в результате выполнения программы make.

install - копирует исполняемый файл в каталог, в котором пользователи обычно ищут команды, копирует все вспомогательные файлы, которые используются исполняемым файлом, в тот каталог, в котором он их ищет.

uninstall - выполняется удаление всех установленных файлов, созданных при исполнении цели *install.*

print - печатает список измененных исходных файлов.

tar - создает из исходных файлов tar-файл.

test - выполняет самотестирование программы, получаемой при помощи данного make-файла.

Флаги чтения

-f <имя файла>

--file=<имя файла>

--makefile=<имя файла>

Задача

Использовать флаг -f (скопировать Makefile в Makefile1)

Переменные

Переменные в make представляют собой именованные строки и определяются следующим образом:

```
<VAR_NAME> = <value string>
```

Существует негласное правило, согласно которому следует именовать переменные в верхнем регистре, например:

```
SRC = main.c file1.c file2.c
```

Так мы определили список исходных файлов. Для использования значения переменной ее следует разименовать при помощи конструкции $(<VAR_NAME>); например так:

```
cc -o main $(SRC)
```

Задание

Добавить в Makefile переменные TARGET и PREFIX.

TARGET — для определения имени целевой программы.

PREFIX — для определения пути установки программы в систему (/usr/local/bin)

Помимо сборки исполняемого файла использовать цели install, uninstall, clean.

Пример использования параметров

```
TARGET = hello
PREFIX = /usr/local/bin

all: $(TARGET)

clean:
	rm -rf $(TARGET) *.o
main.o: main.c
	gcc -c -o main.o main.c
hello.o: hello.c
	gcc -c -o hello.o hello.c
$(TARGET): main.o hello.o
	gcc -o $(TARGET) main.o hello.o
install:
	install $(TARGET) $(PREFIX)
uninstall:
	rm -rf $(PREFIX)/$(TARGET)
```

Пример универсального Makefile-а

```
TARGET = hello
PREFIX = /usr/local/bin
SRCS = main.c hello.c
OBJS = $(SRCS:.c=.o)

all: $(TARGET)
$(TARGET): $(OBJS)
	$(CC) -o $(TARGET) $(OBJS) $(CFLAGS)

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

clean:
	rm -rf $(TARGET) $(OBJS)
install:
	install $(TARGET) $(PREFIX)
uninstall:
	rm -rf $(PREFIX)/$(TARGET)
```

Список используемой литературы

http://rus-linux.net/nlib.php?name=/MyLDP/algol/gnu_make/gnu_make_3-79_russian_manual.html#SEC101

http://habrahabr.ru/post/211751/

Спасибо!