GNU make

Что это?

GNU Make - это версия программы make распространяемая Фондом Свободного Программного Обеспечения (Free Software Foundation - FSF) в рамках проекта GNU (www.gnu.org).

Для чего?

Обычно make используется для автоматической сборки проектов на C/C++, но круг возможностей make на этом далеко не заканчивается

Пример

Пример

- Допустим мы решили написать какую-то программу.
- Пусть это будет некий текстовый редактор.
- Рассмотрим структуру файлов этого проекта:

Файлы

```
$ ls
editor.cpp    editor.h      main.cpp      main.h
text_line.cpp text_line.h
```

Собираем

Для того, чтобы собрать этот проект, например, под g++, нам понадобится написать в консоли:

```
$ g++ main.cpp editor.cpp text_line.cpp
```

Усложним

- Представим, что наш проект вырос, и в нем уже по меньшей мере 50 файлов.
- Проект разбит на модули и подмодули
- В проекте уже сложная древовидная структура файлов
- Проект использует сторонние библиотеки, которые нужно подключать
- Нам требуется разделять сборки debug и release

make

make призван помочь нам в наших рутинных задачах по обслуживанию разработки и сборки проекта

Чтобы собрать проект уже по имеющемуся Makefile достаточно написать

```
$ make
```

в консоли, и мы получим сборку проекта по умолчанию

Makefile

Makefile

- Специальный файл, который описывает работу make в своей директории
- Содержит набор **целей** сборки
- Фактически является скриптовой программой на специфическом языке программирования

Цель сборки

- Целью сборки обычно является файл
- Название цели сборки совпадает с названием файла, который будет собран по этой цели
- Бывают цели, которые не предполагают создание файла (их имена помещают в специальный список **.PHONY**)

Цель сборки

Синтаксически это выглядит вот так:

```
<имя_цели>: <зависимость> <зависимость> ...
    <команда_сборки>
    <команда_сборки>
    ...
```

В качестве отступа для <команда_сборки> используется символ табуляции

Makefile

Пример Makefile для нашего проекта

```
iEdit: main.o editor.o text_line.o
    g++ main.o editor.o text_line.o -o iEdit

main.o: main.h editor.h text_line.h main.cpp
    g++ -c main.cpp

editor.o: editor.h text_line.h editor.cpp
    g++ -c editor.cpp

text_line.o: text_line.h text_line.cpp
    g++ -c text_line.cpp
```

Зависимости

Зависимости позволяют ускорить процесс сборки. Если ни один файл из списка зависимостей текущей цели не менялся, то цель не пересобирается

Переменные

Как и во многих языках программирования в make есть переменные

Упростим с их помощью наш Makefile

Makefile

```
obj = main.o editor.o text_line.o

iEdit: $(obj)
    g++ $(obj) -o iEdit

main.o: main.h editor.h text_line.h main.cpp
    g++ -c main.cpp

editor.o: editor.h text_line.h editor.cpp
    g++ -c editor.cpp

text_line.o: text_line.h text_line.cpp
    g++ -c text_line.cpp
```

Другие возможности язка

Другие синтаксические возможности языка make (такие как функции и циклы) очень похожи на аналогичные конструкции в других языках, поэтому предлагается ознакомиться с ними самостоятельно

Makefile

Изучив хорошо синтаксис мы сможем написать что-то такое

```
iEdit: main.o editor.o text_line.o
    gcc $^ -o $@

%.o: %.cpp
    gcc -c $<

main.o: main.h editor.h text_line.h main.cpp
editor.o: editor.h text_line.h editor.cpp
text_line.o: text_line.h text_line.cpp
```

Clear

make clear

Иногда требуется перекомпилировать проект полностью, а для этого нужно удалить все объектные файлы ***.o**

Для этого можно напиать цель clear, которая описывает процесс удаления всех созданных файлов

Задача

Переписать Makefile в соответствии с новой структурой проекта:

```
main.cpp
main.h

Editor
|- editor.cpp
 \ editor.h

TextLine
 |- text_line.cpp
  \ text_line.h
```