GCC- набор трансляторов

GCC — GNU Compiler Collection

Языки, входящие в GCC:

C, C++	**- gcc, g++**
Objecive-C, Objecive-C++	**- gobjc, gobjc++**
Ada	**- gnat**
Java	**- gcj**
Go	**- gccgo**

Трансляция программ

ТРАНСЛЯЦИЯ ПРОГРАММЫ — это процесс преобразования исходного кода программы в исполняемый файл

Программы на языке С:

Исходный код:

- **Заголовочные файлы (файлы *.h)**
- **Описание функций (файлы *.c)**

Исполняемые файлы:

- **Формат elf.**

Процесс трансляции программы

ТРАНСЛЯЦИЯ

От исходного кода до исполняемого файла.

Дополнительные преобразования исполняемого файла

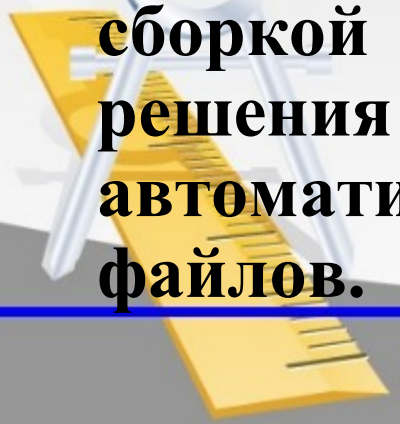
Дополнительные преобразования — не обязательны.

Автоматизация процесса трансляции

Утилита make

- **Утилита make автоматически определяет какие части большой программы должны быть перекомпилированы, и выполняет необходимые для этого действия.**

- **В примерах будут фигурировать программы на языке Си, однако, можно использовать make с любым языком программирования для которого имеется компилятор, работающий из командной строки. На самом деле, область применения make не ограничивается только сборкой программ. Вы можете использовать ее для решения любых задач, где одни файлы должны автоматически обновляться при изменении других файлов.**



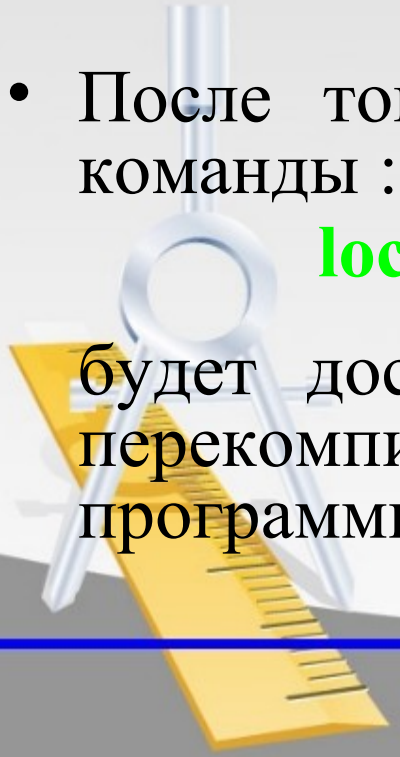
Автоматизация процесса трансляции

- Перед тем, как использовать make, вы должны создать так называемый **мэйк-файл (Makefile)**, который будет описывать зависимости между файлами вашей программы, и содержать команды для обновления этих файлов. Как правило, исполняемый файл программы зависит от объектных файлов, которые, в свою очередь, получаются в результате компиляции соответствующих файлов с исходными текстами.

- После того, как нужный make-файл создан, простой команды :

  localhost# make

  будет достаточно для выполнения всех необходимых перекомпиляций если какие-либо из исходных файлов программы были изменены.



Автоматизация процесса трансляции

- Используя информацию из мэйк-файла, и, зная время последней модификации файлов, утилита make решает, каких из файлов должны быть обновлены. Для каждого из этих файлов будут выполнены указанные в make-файле команды.

  При вызове make, в командной строке могут быть заданы параметры, указывающие, какие файлы следует перекомпилировать и каким образом это делать.
- Простой make-файл состоит из "**правил**" (**rules**) следующего вида:

цель*: *зависимость (или пререквизит)
Символ ТАБУЛЯЦИИ ***команда***

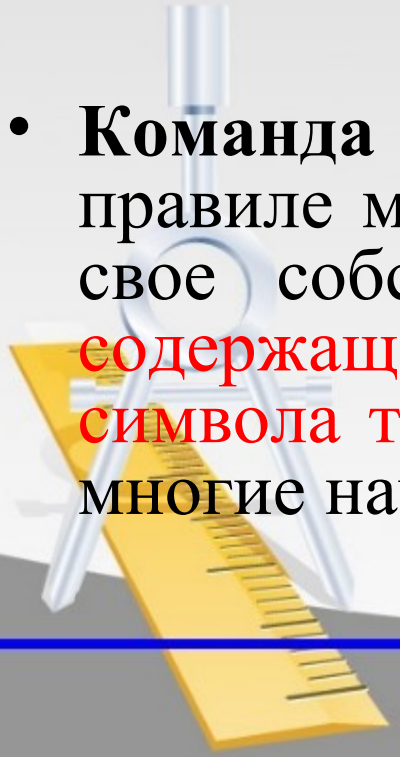
- Обычно, **цель** (**target**) представляет собой имя файла, который генерируется в процессе работы утилиты make.

Автоматизация процесса трансляции

- Примером могут служить объектные и исполняемый файлы собираемой программы. Цель также может быть именем некоторого действия, которое нужно выполнить (например, `**clean**' — очистить).

- **Пререквизит (зависимость)** - это цель или файл, который используется как исходные данные для порождения цели. Очень часто цель зависит сразу от нескольких файлов.

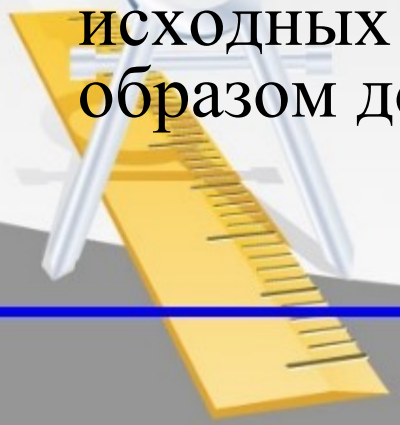
- **Команда** - это действие, выполняемое утилитой make. В правиле может содержаться несколько команд - каждая на свое собственной строке. **Важное замечание:** строки, содержащие команды обязательно должны начинаться с символа табуляции! Это - "грабли", на которые наступают многие начинающие пользователи.



Автоматизация процесса трансляции

- Обычно, команды находятся в правилах с пререквизитами и служат для создания файла-цели, если какой-нибудь из пререквизитов был модифицирован. Однако, правило, имеющее команды, не обязательно должно иметь пререквизиты. Например, правило с целью `**clean**' ("очистка"), содержащее команды удаления, может не иметь пререквизитов.

- **Правило (rule)** описывает, когда и каким образом следует обновлять файлы, указанные в нем в качестве цели. Для создания или обновления цели, make исполняет указанные в правиле команды, используя пререквизиты в качестве исходных данных. Правило также может описывать, каким образом должно выполняться некоторое действие.



Автоматизация процесса трансляции

- Помимо правил, make-файл может содержать и другие конструкции, однако, простой make-файл может состоять и из одних лишь правил.
- Вот пример простого make-файла, в котором описывается, что исполняемый файл **edit** зависит от объектного файла, который, в свою очередь, завфисит от соответствующих исходного файла и заголовочного файлов.

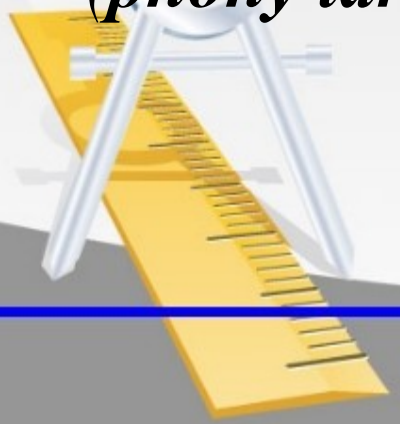
```
edit : main.o
	gcc -o edit main.o

main.o : main.c defs.h
	gcc -c main.c

clean :
	rm edit main.o
```

Символ ТАБУЛЯЦИИ

Автоматизация процесса трансляции

- Цель `**clean**' является не файлом, а именем действия. Поскольку, при обычной сборке программы это действие не требуется, цель `**clean**' не является пререквизитом какого-либо из правил. Следовательно, **make** не будет "трогать" это правило, пока вы специально об этом не попросите.

  Заметьте, что это правило не только не является пререквизитом, но и само не содержит каких-либо пререквизитов. Таким образом, единственное предназначение данного правила - выполнение указанных в нем команд. Цели, которые являются не файлами, а именами действий называются ***абстрактными целями (phony targets)***.



Автоматизация процесса трансляции

- Для повышения удобочитаемости, можно разбить длинные строки на две части с помощью символа обратной косой черты, за которым следует перевод строки. Для того, чтобы с помощью этого make-файла создать исполняемый файл `edit', наберите:

  localhost# make

- Для того, чтобы удалить исполняемый и объектные файлы из директории проекта, наберите:

  localhost# make clean

- В примере целями являются объектный файл `**main.o**', а также исполняемый файл `**edit**'. К пререквизитам относятся файлы`**main.c**' и `**defs.h**'. Каждый объектный файл является одновременно и целью и пререквизитом. Пример команд `**gcc -c main.c**'.