Предисловие

Мейкфайлы это специализированные файлы с инструкциями для сборки исполняемого файла из множества исходников. Они очень упрощаю жизнь при работе с проектами, где модулей становится >1.

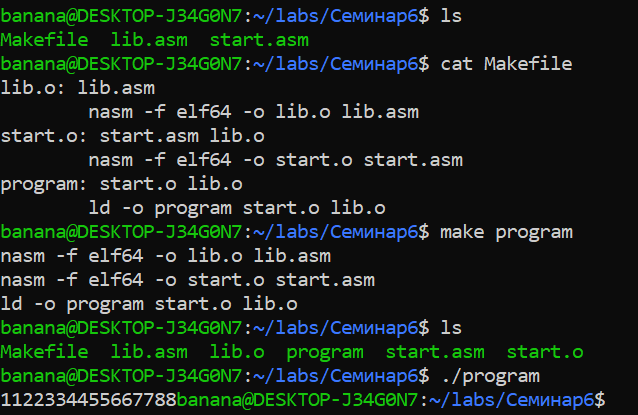


**файлы lib.asm start.asm Makefile**

сборкой исполняемого файла у нас занимается цель *program*

т.е. для сборки одноименного исполняемого файла нужно выполнить команду

*make program*

****

****

**rule (правило):**

*Конструкция, говорящая, когда и как необходимо изменить файл.*

Вот пример правила из предыдущего пункта.



Когда? В первый строчке мы говорим, что, когда изменяется файл start.asm, также необходимо внести изменения в файл start.o.

Как? Во второй строчке мы говорим, что изменить файл start.o необходимо методом перекомпиляции start.asm

Важно, что, отвечая на вопрос *как,* мы используем отступ.

**target (цель):**

Файл, который необходимо изменить. В примере это start.o

**Prerequisites (предпосылки / зависимости):**

Условия, необходимые для изменения цели. В примере это start.asm (а вернее факт изменения start.asm)

итого, правило имеет следующий общий вид:

*цель: зависимости*

*команды*



**phony target** или фальшивая цель - по сути та же цель из предыдущего пункта, но не ассоциирующаяся с именем файла. Используются они для избегания конфликтов имен файлов, либо для ускорения процесса сборки.

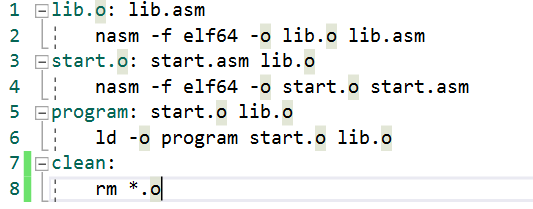
Например, мы можем создать цель clean, которая будет служить для удаления всех объектных файлов в каталоге.

*clean:*

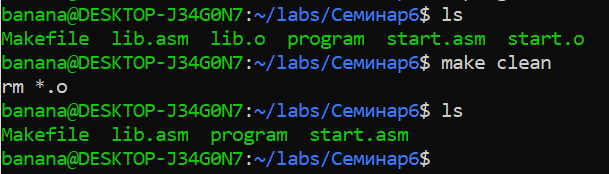
*rm \*.o*

мы можем исполнить это правило с помощью команды *make clean*

в данный момент Makefile имеет вид:

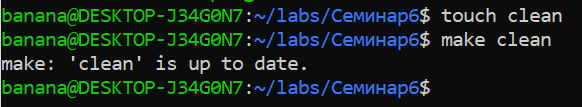


выполняем:

**

Может возникнуть такая ситуация, когда созданная нами phony target имеет имя, как и какой-либо файл в директории, тогда правило не отработает.

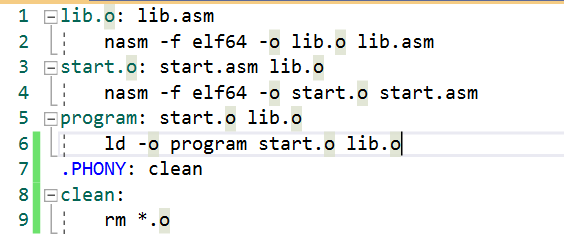
создадим файл clean и убедимся:



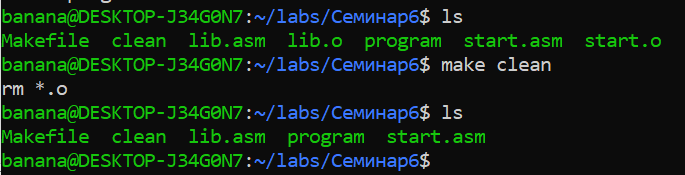
Обойти эту проблему можно, добавив в Makefile конструкцию вида:

.PHONY: phony\_target\_name

в случае с clean:

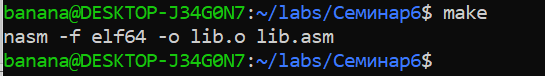


проверим, что все работает:

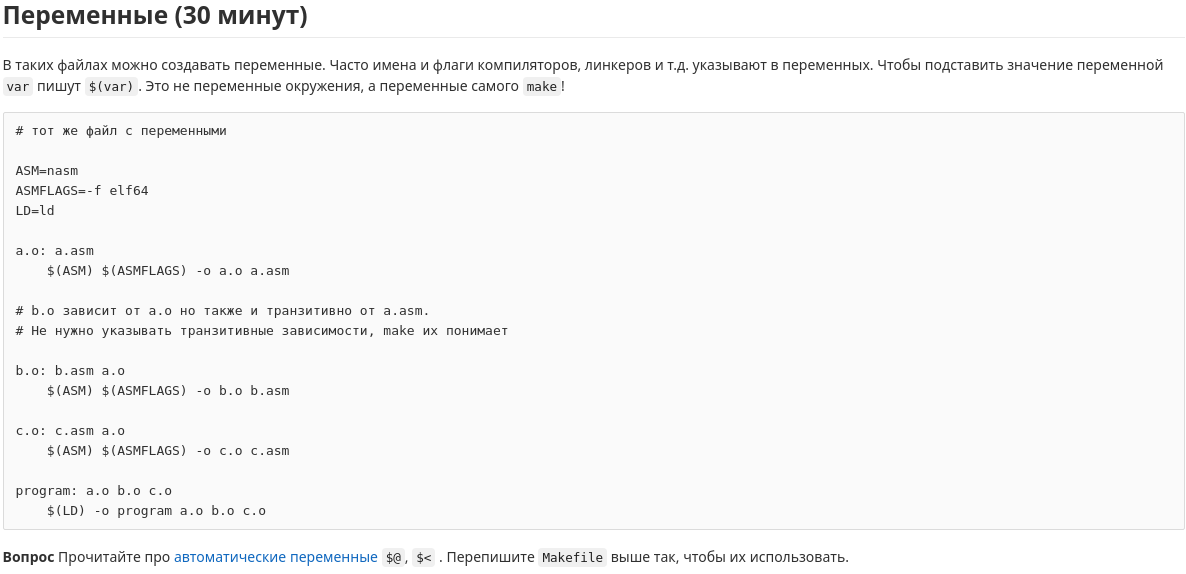
**



Если не указывать цель явно т.е. запустить make без аргумента, по умолчанию будет выбрано правило, стоящее первым в мейкфайле:

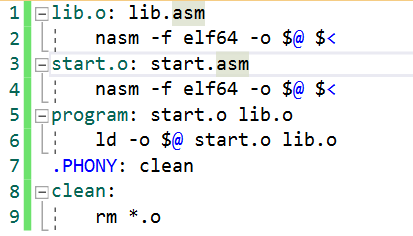


\*помним, что у нас как раз первым стояло правило сборки lib.o\*



автоматическая переменная $@ используется для подстановки цели правила (например lib.o или start.o в первом и втором правилах или program в третьем)

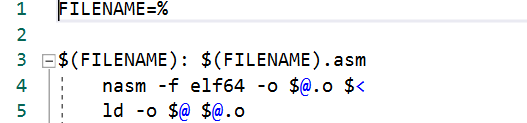
автоматическая переменная $< используется для подстановки зависимости цели (только одной!) (например lib.asm или start.asm в первом и втором правиле)





Мы можем использовать шаблоны с помощью специального символа %

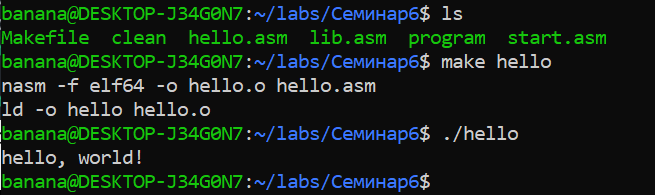
Он похож на регулярное выражение .+ (любой набор символов в количестве одного или более)



так как мы не знаем, как будет называться файл .asm, используем для него шаблон $(FILENAME).asm, что эквивалентно %.asm.

целью является файл с именем $(FILENAME), т.е. с именем файла-исходника, но без расширения .asm.

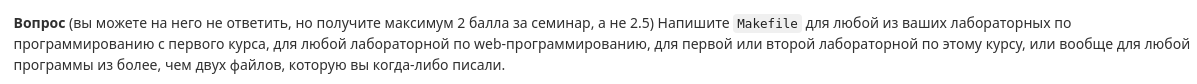
далее используем автоматические переменные для подстановки нужных имен:



теперь мы можем собрать исполняемый файл для любого имя\_файла.asm

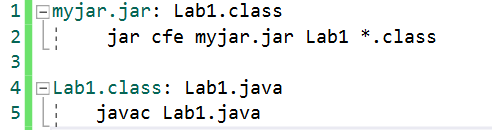
для этого нужно лишь написать make имя\_файла

(тут мы для примера написали прогу hello.asm, которая просто выводит “hello,world!”)



Соберем с помощью make первую лабу по проге, а точнее соберем исполняемый jar архив

Makefile будет иметь такой вид



выполнив make myjar.jar, получим готовый jar-ник

запустим его с помощью java -jar myjar.jar и убедимся, что все работает

