

[КАК СТАТЬ АВТОРОМ](#)[Собираем истории ко дню бэкапа](#)

BubaVV 18 окт 2012 в 18:52

## Makefile для самых маленьких

4 мин 723K

Программирование\*

[Тutorial](#)

Не очень строгий перевод материала [mrbook.org/tutorials/make](http://mrbook.org/tutorials/make) Мне в свое время очень не хватило подобной методички для понимания базовых вещей о make. Думаю, будет хоть кому-нибудь интересно. Хотя эта технология и отмирает, но все равно используется в очень многих проектах. ~~Кармы на хаб «Переводы» не хватило, как только появится возможность — добавлю и туда.~~ Добавил в Переводы. Если есть ошибки в оформлении, то прошу указать на них. Буду исправлять.

Статья будет интересная прежде всего изучающим программирование на C/C++ в UNIX-подобных системах от самых корней, без использования IDE.

Компилировать проект ручками — занятие весьма утомительное, особенно когда исходных файлов становится больше одного, и для каждого из них надо каждый раз набивать команды компиляции и линковки. Но не все так плохо. Сейчас мы будем учиться создавать и использовать Мейкфайлы. Makefile — это набор инструкций для программы make, которая помогает собирать программный проект буквально в одно касание.

Для практики понадобится создать микроскопический проект а-ля Hello World из четырех файлов в одном каталоге:

▸ [main.cpp](#)

▸ [hello.cpp](#)

▸ [factorial.cpp](#)

▸ [functions.h](#)

Все скопом можно скачать [отсюда](#)

Автор использовал язык C++, знать который совсем не обязательно, и компилятор g++ из gcc. Любой другой компилятор скорее всего тоже подойдет. Файлы слегка подправлены, чтобы собирались gcc 4.7.1

## Программа make

Если запустить

```
make
```

то программа попытается найти файл с именем по умолчанию `Makefile` в текущем каталоге и выполнить инструкции из него. Если в текущем каталоге есть несколько

мейкфайлов, то можно указать на нужный вот таким образом:

```
make -f MyMakefile
```

Есть еще множество других параметров, нам пока не нужных. О них можно узнать в ман-странице.

## Процесс сборки

Компилятор берет файлы с исходным кодом и получает из них объектные файлы.

Затем линковщик берет объектные файлы и получает из них исполняемый файл.

Сборка = компиляция + линковка.

## Компиляция руками

Самый простой способ собрать программу:

```
g++ main.cpp hello.cpp factorial.cpp -o hello
```

Каждый раз набирать такое неудобно, поэтому будем автоматизировать.

## Самый простой Мейкфайл

В нем должны быть такие части:

```
цель: зависимости  
[tab] команда
```

Для нашего примера мейкфайл будет выглядеть так:

```
all:  
    g++ main.cpp hello.cpp factorial.cpp -o hello
```

Обратите внимание, что строка с командой должна начинаться с табуляции!

Сохраните это под именем `Makefile-1` в каталоге с проектом и запустите сборку командой `make -f Makefile-1`

В первом примере цель называется `all`. Это цель по умолчанию для мейкфайла, которая будет выполняться, если никакая другая цель не указана явно. Также у этой цели в этом примере нет никаких зависимостей, так что `make` сразу приступает к выполнению нужной команды. А команда в свою очередь запускает компилятор.

### Использование зависимостей

Использовать несколько целей в одном мейкфайле полезно для больших проектов. Это связано с тем, что при изменении одного файла не понадобится пересобирать весь проект, а можно будет обойтись пересборкой только измененной части. Пример:

```
all: hello

hello: main.o factorial.o hello.o
    g++ main.o factorial.o hello.o -o hello

main.o: main.cpp
    g++ -c main.cpp

factorial.o: factorial.cpp
    g++ -c factorial.cpp

hello.o: hello.cpp
    g++ -c hello.cpp
```

```
clean:
    rm -rf *.o hello
```

Это надо сохранить под именем `Makefile-2` все в том же каталоге

Теперь у цели `all` есть только зависимость, но нет команды. В этом случае `make` при вызове последовательно выполнит все указанные в файле зависимости этой цели.

Еще добавилась новая цель `clean`. Она традиционно используется для быстрой очистки всех результатов сборки проекта. Очистка запускается так: `make -f Makefile-2 clean`

## Использование переменных и комментариев

Переменные широко используются в мейкфайлах. Например, это удобный способ учесть возможность того, что проект будут собирать другим компилятором или с другими опциями.

```
# Это комментарий, который говорит, что переменная CC указывает компилятор
CC=g++
#Это еще один комментарий. Он поясняет, что в переменной CFLAGS лежат флаг
CFLAGS=-c -Wall

all: hello

hello: main.o factorial.o hello.o
    $(CC) main.o factorial.o hello.o -o hello

main.o: main.cpp
```



```
$(CC) $(CFLAGS) factorial.cpp
```

```
hello.o: hello.cpp
```

```
$(CC) $(CFLAGS) hello.cpp
```

```
clean:
```

```
rm -rf *.o hello
```

## Это Makefile-3

Переменные — очень удобная штука. Для их использования надо просто присвоить им значение до момента их использования. После этого можно подставлять их значение в нужное место вот таким способом: `$(VAR)`

## Что делать дальше

После этого краткого инструктажа уже можно пробовать создавать простые мейкфайлы самостоятельно. Дальше надо читать серьезные учебники и руководства. Как финальный аккорд можно попробовать самостоятельно разобрать и осознать такой универсальный мейкфайл, который можно в два касания адаптировать под практически любой проект:

```
CC=g++
```

```
CFLAGS=-c -Wall
```

```
LDFLAGS=
```

```
SOURCES=main.cpp hello.cpp factorial.cpp
```

## ЧИТАЮТ СЕЙЧАС

За кем сейчас охотятся крупные работодатели в IT?

👁 12K

💬 21 +8

Президент дал поручение о производстве игровых приставок и консолей в России

👁 7.4K

💬 108 +108

Третий год борюсь с инфоцыганами. Теперь за мной следит наблюдательный совет

👁 6.6K

💬 10 +10

Microsoft выпустила крупное обновление под названием Moment 5 для Windows 11 для всех пользователей

👁 1.9K

💬 0

Samsung Pay перестанет работать в

```
OBJECTS=$(SOURCES:.cpp=.o)
EXECUTABLE=hello

all: $(SOURCES) $(EXECUTABLE)

$(EXECUTABLE): $(OBJECTS)
    $(CC) $(LDFLAGS) $(OBJECTS) -o $@

.cpp.o:
    $(CC) $(CFLAGS) $< -o $@
```

Makefile-4

Успехов!

Теги: makefile, перевод, мануал, начинающим

Хабы: Программирование

↑ 154 ↓  
Карма0  
Рейтинг

Вадим Марков @BubaVV

Пользователь



Подписаться

России

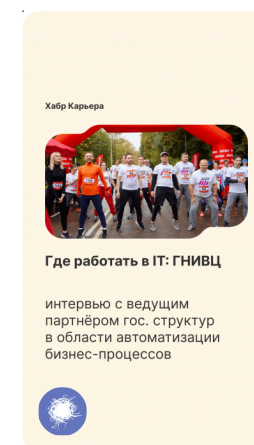
9.7K

38 +38

Как исправить раздвоение встреч в конференциях на базе Jitsi: опыт команды Телемоста

Интересно

ИСТОРИИ



Где работать в IT: ГНИВЦ



GitVerse: открой вселенную кода

БЛИЖАЙШИЕ СОБЫТИЯ

◦  **kekekeks** 18 окт 2012 в 18:56

Вот бы кто automake и autoconf разжевал.

↑ **+13** ↓ Ответить

◦  **danfe** 18 окт 2012 в 19:54 ^


I saw a book entitled «Die GNU Autotools» and I thought «My feelings exactly». Turns out the book was in German. [via]

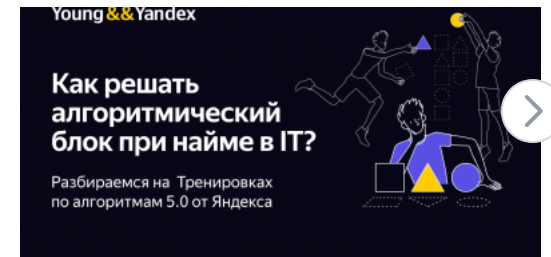
↑ **+26** ↓ Ответить

◦  **shock\_one** 18 окт 2012 в 20:00 ^



↑ **-13** ↓ Ответить

◦  **symbix** 18 окт 2012 в 16:26 ^



## Серия занятий «Тренировки по алгоритмам 5.0» от Яндекса

📅 1 марта – 19 апреля

🕒 19:00

📍 Онлайн

[Подробнее в календаре](#)



Если вы начинаете новый проект, а не вынуждены поддерживать существующее — обходите automake и autoconf за милую руку :) Есть множество альтернатив — cmake, scons итд — разработчиков которых хотя бы не хочется четвертовать.

 +3 Ответить  ...

o  **janatem** 19 окт 2012 в 00:44 ^

Нельзя ли поподробней, что такого ужасного в GNU Autotools? Я использовал их в нескольких проектах с самого начала и по большому счету ни разу не пожалел, хотя приходилось писать небольшие куски на m4, а также возникали определенные неудобства, когда хотелось странного.


 0 Ответить  ...

o  **symbix** 19 окт 2012 в 05:22 ^

m4 ужасен. divert — это вообще за гранью добра и зла.

Для небольших проектов, где ничего особенного не нужно, все выглядит неплохо, но... Попробуйте на досуге разобраться в autoconf-е php ;)

 0 Ответить  ...

o  **janatem** 19 окт 2012 в 11:13 ^

Я бы сказал, что m4 странен, а divert мне не понадобился (повезло наверно), поэтому я не знаю, что это такое. Сколь-нибудь нетривиальную макруху приходилось писать для Charm++ и для CUDA. В первом случае, насколько я помню, было всё гладко, а во втором осталось какое-то несовершенство, но терпимое. Правда, мои коллеги боялись даже заглядывать во все эти тексты и осторожно ковыряли только ат-мейкфайлы. Наверно, действительно, порог вхождения высок.

 0 Ответить  ...

o  **akalend** 18 окт 2012 в 16:56 ^

есть куча переведенных туториалов

 0 Ответить  ...

o  **sublimelion** 18 окт 2012 в 14:59

Если уж заикнулись про зависимости, рассказали бы про makedepend

 +4 Ответить  ...


o  **BubaVV** 18 окт 2012 в 15:23

[www.galassi.org/mark/mydocs/autoconf\\_tutorial\\_1.html](http://www.galassi.org/mark/mydocs/autoconf_tutorial_1.html)

[www.freesoftwaremagazine.com/books/autotools\\_a\\_guide\\_to\\_autoconf\\_automake\\_libtool](http://www.freesoftwaremagazine.com/books/autotools_a_guide_to_autoconf_automake_libtool)

Нужно что-то типа реферата по этим материалам?

 0 Ответить  ...

o  **Disasm** 18 окт 2012 в 16:00 ^

Нет конечно, вы что

 +5 Ответить  ...

o  **BubaVV** 18 окт 2012 в 16:17 ^

Неправильно выразился. На основе толстых талмудов сделать компактный  
легкоусвояемый туториал вроде вот этого. Или первый комментатор использовал

sarcasm mode on?

0 Ответить

Disasm 18 окт 2012 в 16:19

1. Это сделать полезно и нужно
2. Такие вопросы надоели уже

0 Ответить

BubaVV 18 окт 2012 в 16:26

Вопросы про реферат или про сарказм? Я еще атмосферу ресурса не очень улавливаю, поэтому спрашиваю глупости

0 Ответить

Ikart 18 окт 2012 в 17:43

На ресурсе часто можно встретить вопросы из разряда «А нужна ли статья про..» и, по моим наблюдениям, на такие вопросы всегда отвечают, что да, нужна, делайте. Зачем тогда такие бесполезные вопросы? Если можете сделать хорошую статью о чем-то, то просто сделайте.

+3 Ответить

vanxant 18 окт 2012 в 16:03

Для цели clean все же лучше указывать точку перед o (rm -rf \*.o)

+5 Ответить

- ◉  **BubaVV** 18 окт 2012 в 16:15 ^


Исправил в посте и в архиве проекта

◆ +1 Ответить 📖 ⋮

- ◉  **nerudo** 18 окт 2012 в 17:38 ^

А для цели fun просто писать \* без ".o"

◆ +5 Ответить 📖 ⋮

- ◉  **vanxant** 19 окт 2012 в 23:59 ^

Скорее для цели mazo :)

◆ 0 Ответить 📖 ⋮

- ◉  **burjui** 18 окт 2012 в 17:56

Вместо

```
main.o: main.cpp
    g++ -c main.cpp

factorial.o: factorial.cpp
    g++ -c factorial.cpp

hello.o: hello.cpp
    g++ -c hello.cpp
```

Лучше писать:

```
.SUFFIXES: .cpp .o



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

`$@` — имя .o-файла

`$<` — имя .cpp-файла

Такое правило будет автоматом работать для всех .o-файлов, указанных в качестве зависимостей цели.

 **+8** Ответить  

○  **Sap\_ru** 18 окт 2012 в 22:46 

Ровно до тех пор, пока не появится необходимость подключить проекту объектный модуль. Соответственно, для этого модуля не будет .cpp-файла. Более того, может появиться необходимость подключить один единственный файл .c и make тоже поломается.

 **-2** Ответить  

○  **burjui** 18 окт 2012 в 23:48 

Вы даже не проверяли то, о чём говорите.

Ровно до тех пор, пока не появится необходимость подключить проекту объектный модуль. Соответственно, для этого модуля не будет .сpp-файла.

Только что создал файл `x.c`, вручную скомпилировал в `z.o`, и добавил `z.o` в список целей сборки одного моего личного проекта — собирается нормально, никаких проблем у *make* не возникло со сборкой. Я даже успешно вызвал из кода на `D` функцию из `z.o`.

Более того, может появится необходимость подключить один единственный файл `.c` и *make* тоже ломается.

Опять же, никаких проблем:

```
.d.o:
    $(D_COMPILER) $(D_COMPILER_FLAGS) -c -of$@ $<

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

Убираю из списка целей `z.o`, добавляю `x.o` — всё прекрасно собирается.

◆ +1 Ответить

○  **gribozavr** 18 окт 2012 в 18:16



Если изменятся заголовочные файлы, ваш мейкфайл ничего не пересоберёт.

◆ +2 Ответить

○  **Bas1l** 18 окт 2012 в 19:16

Я очень порекомендую вам `premake` (среди тулзов `stake`, `qmake`, `premake` он мне понравился больше всего). Хотя последняя версия вышла 16 ноября 2010, так что, возможно, он не очень живой.

◆ -1 Ответить

○  **niisan** 18 окт 2012 в 19:35 

*программа попытается найти файл с именем по умолчанию `makefile`*

Это юникс, здесь важен регистр букв. `make` пытается найти файл с именем по умолчанию **Makefile**

◆ +5 Ответить

○  **BubaVV** 18 окт 2012 в 22:36 

позорно протупил. Конечно же так правильно. Исправляю

◆ 0 Ответить

○  **abusalimov** 18 окт 2012 в 23:26 

Если речь идет о GNU Make, то по умолчанию проверяются три файла, по порядку:  
**GNUmakefile -> makefile -> Makefile.**


Другое дело, что в документации *рекомендуется* использовать именно **Makefile**, чтобы в листинге директории видеть его сверху. Пруф.

◆ +2 Ответить

-  **KOLYUNYA** 18 окт 2012 в 23:53


До сих пор пользовался скриптом на баше, чтобы компилировать большое количество файлов. Спасибо за перевод, давно хотел изучить этот вопрос.

◆ 0 Ответить  ...

-  **iss** 19 окт 2012 в 03:04

Еще бы показали, как одним make-файлом собирать проекты написанные частично на С, частично на C++.

◆ 0 Ответить  ...

-  **gscdlr** 23 мая 2022 в 12:33 ^

g++ для .cpp

gcc для .c

отдельно собираем объектные .o для .cpp-кода через g++, и отдельно объектные сишек. Потом собираем общий elf a.out.

Поправьте, если что не так, я еще плохо шарю.

◆ 0 Ответить  ...

-  **Delsian** 19 окт 2012 в 11:05

как\_нарисовать\_сову.jpg

◆ -2 Ответить  ...

-  **NekR** 22 окт 2012 в 16:20



```
.cpp.o:  
$(CC) $(CFLAGS) $< -o $@
```

Makefile-4

Успехов!

Пока в комментариях не разживали, что такое \$< и \$@ ничего не было понятно. Да и после того всё придельно ясно не стало. Зачем вообще было упоминать эти токены, если они не рассказали?

♦ -1 Ответить

timurrrr 14 ноя 2012 в 21:49

В первом примере цель называется all. Это цель по умолчанию для мейкфайла, которая будет выполняться, если никакая другая цель не указана явно.

Тут злостно нарушена причинно-следственная связь!

На самом деле, «по умолчанию» цель выбирается не «all», а просто первая цель в Makefile'е.

Называться-то она может вообще как угодно:

```
firsttarget:  
    echo "The first one is the default"  
  
all:  
    echo "All is the default"
```

Проверяем:

```
$ make
echo "The first one is the default"
The first one is the default
```

Понимаю, что это перевод, но своя голова-то тоже не бывает лишней :-)

Из той же серии:

```
all:
    g++ main.cpp hello.cpp factorial.cpp -o hello
```

— правилом хорошего тона вроде как считается называть цель так же, как называется результат выполнения команд (hello).

Могу поверить, что стилистическая ошибка в последнем примере обусловлена непониманием, описанным выше.

 +2 Ответить  ...

Вы можете оставлять комментарии только к свежим публикациям

## Публикации

ЛУЧШИЕ ЗА СУТКИ   ПОХОЖИЕ

**ViktorSergeev** 23 часа назад

## Подключаемся к BBS через Amstrad NC100 из 1992 года

4 мин

2.4K

**+33**

12



12

**+12****valisak** 5 часов назад

## Может ли во Вселенной не быть тёмной материи? 5 фактов, которые нельзя отрицать

Средний

8 мин

3.9K

**+29**

8



9

**+9**

Обзор

**Martynov\_M** 18 часов назад

## Основание кулера выпуклое?

Простой

4 мин

6.1K

**+28**

15



20

**+20**

Recovery Mode

**bodyawm** 6 часов назад

## Исходников нет, но мы не сдадимся: как и зачем я портировал более старый Android, чем стоял «с завода»?

 Средний  13 мин  1.6K

Ретроспектива

 +20  3  11 +11



devops\_ht 22 часа назад

## ClickHouse как бэкенд для Prometheus

 Средний  8 мин  3.6K

Тutorial

 +20  48  0



ru\_vds 1 час назад

## Актуально ли сегодня ООП?

 Средний  11 мин  1.4K

Мнение

Перевод

 +17  7  3 +3



ankirill 6 часов назад

## За кем сейчас охотятся крупные работодатели в IT?

 Простой  3 мин  12K

 +17  22  21 +8



YuriPanchul 6 часов назад

## Макфол ответил на вопрос про санкции в микроэлектронике

 Простой  3 мин  2.8K

Репортаж

 +16  5  11 +11



Firemoon 3 часа назад

## Фабрика должна расти: настраиваем игровой кластер Factorio

 7 мин  1.8K

 +15  8  9 +9



arheo\_pterix 15 часов назад

## Гармония танцующих линий

 Простой  9 мин  2K

 +14  20  26 +26

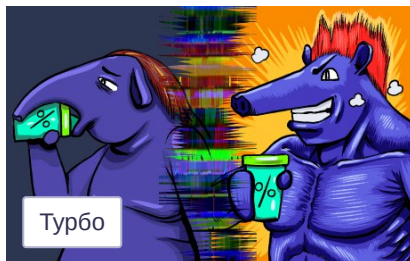
## Как исправить раздвоение встреч в конференциях на базе Jitsi: опыт команды Телемоста

[Интересно](#)[Показать еще](#)

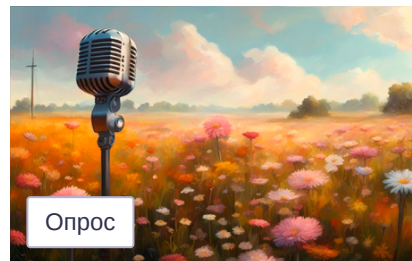
## МИНУТОЧКУ ВНИМАНИЯ



Планируй своё время и его хватит на IT-ивенты из Календаря



Как бессонница в час ночной, меняет промокодище облик твой



Ивент моей мечты: опрос среди айтишников

## КУРСЫ

 Python-разработчик с нуля


3 апреля 2024 · Нетология


 IT-профессия с нуля: курс с выбором специализации

8 апреля 2024 · Нетология

 Старт в программировании

12 апреля 2024 · Нетология

 Go-разработчик  
30 марта 2024 · Бруноям

 Графический дизайнер: расширенный курс  
1 апреля 2024 · Нетология  
Больше курсов на Хабр Карьере

Ваш аккаунт

Профиль  
Трекер  
Диалоги  
Настройки  
ППА

Разделы

Статьи  
Новости  
Хабы  
Компании  
Авторы  
Песочница

Информация

Устройство сайта  
Для авторов  
Для компаний  
Документы  
Соглашение  
Конфиденциальность

Услуги

Корпоративный блог  
Медийная реклама  
Нативные проекты  
Образовательные программы  
Стартапам

