Makefiles

Timothy Daly

March 15, 2004

Abstract

Makefiles automate the process of building systems. In recent years there has been a standardization on the method of building software. The primary tool is "make" and we explain how to build Makefiles. We explain the expectations of the standard "configure", "make", "make install" steps. We briefly cover the Autoconf and Automake tools.

Contents

1 Weekly News

Brazil loves Linux - true [13]

> Brazil is the world's most enthusiastic exponent of open source software, a BBC report reveals. And, if the Latin-American love affair with Linux continues, a third of machines there may soon run non-MS OSes.
>
> Professor Arnando Mandel at Sao Paulo University summarises the appeal of Linux, which runs on both the university's servers: "The main thing is to be in control of your own data. In a democracy it's important that a free government shouldn't be subject to the whims of a company."

Latin America is rapidly becoming the land of open source. It is adopting Linux in several countries, has support in the government, and has a growing user population.

Australia-U.S. Trade Agreement Contains DMCA-like Provisions [12]

> femto writes "The text of the US-Australian Preferential Trade Agreement has been released. It has significant implications for Free Software and the Public Domain within Australia. Implications include extension of copyright terms (death to the Public Domain & Gutenberg Australia), software patents (death to Free Software) and the DMCA (death to fair use). It is not yet law. The Europeans have shown that software patents are not a done deal. Now is the time to write letters to members of the House of Representatives and the Senate. Join the EFA. Contact your local library. Sign up to the mailing list to organise opposition. Just make a noise during this year's federal election."

Copyrights and Patents have the potential to damage your ability to write and distribute open source. It is important, as I've stressed many times, to become aware of these issues. The media companies are working very hard on lobby efforts to make laws which limit what you can write. Worse yet is that there are international treaties that cause a bad law in one country to be forced into adoption by other countries. If you don't begin to pay attention to these issues and contact your representatives you'll find yourself facing a jail term for using a computer.

How To Hire Great Open Source Developers? [7]

An anonymous reader writes "This is the first article I've ever read specifically about hiring open source developers, and how to judge their ability not just to code but to work with others. It's reprinted over at ITMJ

[part of OSDN, as this site is] from a book by Martin Fink, the General Manager for HP's Linux Systems Division. Brings up a lot of good points, including how you need to make sure your open source people are developing things (on company time) that do the company some good, not just scratching their own itches. Fun quote: 'Discover what pseudonyms your candidate uses online. Look at the archives at SlashDot and other online locales. Does your candidate hide behind secret pseudonyms to trash other individuals? Is there passion without condemnation?'"

In the first class I told you that your reputation online **is** your resume. The net never forgets. Be active, be useful, and be careful what you write. It matters.

DVD Forum mandates Microsoft for HD disc standard [11]

There are two key issues here. First is the DRM issue. DRM limits your ability, not only to copy, but also to play DVDs you've already purchased. It is trivially easy to key the DVD to a particular player connected to a particular TV. So your DVD won't play anywhere else but in your living room. Second is the patent issue. Microsoft holds the patent on the DRM technology and can charge for its usage and limit it. This means that there cannot be a free HD-DVD player.

Game Development: Harder Than You Think" [10]

In the past 10 years games have gotten much more complex. This article talks about the issues that arise when code size grows by a factor of 100 or more. The article is of interest because game programming is an active, money-making area of software. Experienced game developers are in great demand. If you want to get involved find an online game and begin to contribute your time and attention.

Closing Windows: free Hebrew alternative for Microsoft system [9]

> Another blow for Microsoft: on the heels of a free alternative to its Office software, a no-charge alternative operating system to Windows is now flooding the country. Knoppix, a Hebrew operating system requiring no installation, is the first real threat to the Microsoft monopoly in Israel.

Isreal has begun the transition to open source software. One of the key driving reasons is the lack of support for Hebrew in Windows. Linux allowed people to develop a branch of the office software which supports the Hebrew language.

New Law in China to End Microsoft's Dominance [8]

> BEIJING, FEB 27: For years, China has been trying to end Microsoft Corp.'s monopoly on its computers. It has tried to

> develop its own operating system. It has appealed to the patriotism of consumers. Now, it is turning to the law.
>
> Officials say a new law will be announced by this summer requiring a minimum percentage of software purchased by the government be produced in China. That's crucial in a country where the government accounts for 25 percent of the $30 billion software market.
>
> No one is saying what that minimum will be – some say it may be as high as 70 percent – but one thing is certain: Linux will be the beneficiary.
>
> .....[snip].....
>
> China says it is merely trying to level the playing field for its own software companies.
>
> "If a software program is dominant for a long time, it's harmful for the development of the software industry," said Li Wuqiang of the Ministry of Science and Technology.

China realizes that it needs to develop a local IT industry. Almost all of its software comes from overseas which means that it is not developing native talent in this critical area. Further, at several hundred dollars (U.S.) per official copy China will develop a large trade deficit. It is China's best interest to develop a local talent pool. Such "social engineering laws" are the fastest way to ensure the local growth.

MPs call for Gov.uk to switch to open source (maybe) [6]

> Central government should replace Microsoft software with open source equivalents – if trials show the switch is practical...

This is really a pressure ploy against Microsoft prices. One city did a study and found that open source was the better alternative so Microsoft dropped their prices. Now the whole government is beginning to play the same pressure game. At least this can be viewed as more free (as in beer) advertising. The british have figured out that they can save money but they don't seem to understand the real benefit.

Freeing the Mind: Free Software and the Death of Proprietary Culture [5]

This article deals with the difference between Free and Proprietary software. It also details the difference between Free and Open Source software.

BBC Discusses PVR Software, Creative Archive Plans [4]

> Fidigit writes "You may have heard something about the BBC Internet Media Player - a computer-based PVR for the BBC's TV and radio content, 'only... available to UK broadband users',

> which'll use P2P to shuttle content around between downloaders. Now we hear the iMP content will distributed using DRM, using Microsoft's DRM technology, 'in a break with the BBC's long-standing support of Real.'" The previously mentioned BBC Creative Archive is also discussed - apparently its content "...will be downloaded using a similar application, but will not be restricted by DRM, enabling people to re-edit it, or use it to make other programmes" - the content "will not be the complete BBC archive", but an example given of the initial content is "nature programmes".

DRM (digital rights management) is technology which will enable companies to control your computer usage. It is a major community threat (like patents) because it will limit your ability to write free programs. You need to become very sensitive to the issues surround the introduction and use of DRM.

Rome Moving to Linux [3]

> fmstasi writes "La Repubblica, one of the main Italian newspapers, reports shortly about an interview (in Italian) with Mariella Gramaglia, Communication Councillor at the Municipality of Rome. They are planning to start soon trying Linux on the desktop: 'The first tests will concern e-mail, address book software and sharing systems', she says. The Councillor also says that motivations are political rather than economic: 'In the short term, the money saved on license will have to be spent on training'. It seems that there haven't been any reaction yet from Microsoft: 'At Microsoft they know how much we esteem them', she says; 'for example, they are sponsoring a campaign to spread the use of computers among the elderly. And we'll keep on cooperating with them on other projects'. Maybe Microsoft also appreciates that there is (yet) no project of migrating all the clients? The Municipality has about 9,500 clients, so an eventual migration project would be slightly smaller than the one taking place in Munich."

Another government has decided that Linux might be of some value. What is particularly interesting here is that Rome is talking about replacing the desktop software. This is a central struggle in the Microsoft vs Linux debate.

Judge bans DVD X Copy software [1]

> DVD duplication software maker 321 Studios has been given seven days to stop selling its controversial DVD X Copy family of utilities, a US federal judge has ruled.

> The ruling follows legal action brought against the developer by MGM Studios, Tristar Pictures, Columbia Pictures, Time Warner Entertainment, Disney, Universal City Studios and The Saul Zaentz Company last May. The movie companies alleged 321's product violated the Digital Millennium Copyright Act (DMCA).

Another blow against you by the media companies. Here they use the DMCA to keep you from making a copy of something you own by forcing a company to modify its product. It is pointless in the long run but you can see that media companies consider you to be a "consumer", not a person. The term "consumer" should be thought of as an insult.

Kazaa trial judge delays hearing [2]

> Kazaa owner Sharman Networks will have to wait until next week to hear if the Australian federal court will suspend legal action brought against it by the Aussie music industry.
>
> Judge Murray Rutledge Wilcox adjourned a hearing due to take place this past Friday in order to allow the parties to make further submissions this week, the peer-to-peer software company said.
>
> Music Industry Piracy Investigations (MIPI), an organisation sponsored by the Australian Recording Industry Association (ARIA), opened proceedings against Kazaa on 10 February. It claims the company is guilty of aiding and abetting copyright infringement.
>
> At the earlier hearing, Kazaa's lawyers asked Judge Wilcox to suspend the case pending the outcome of a similar hearing being held in the US Court of Appeals. The latter case pitches the US movie and music industries against P2P services Grokster and Morpheus.

This is equivalent to bringing suit against the phone company because people are saying things you don't like. They are asking for phones to be outlawed. In particular, the music industry doesn't want you to be able to share **any** files with anyone else because they **might** be music files. So P2P software should be outlawed. I find it hard to believe that anyone could try such a suit and impossible to believe that it could succeed. However, I feel sure that it will succeed because you just need to "follow the money".

EU backs tighter rules on piracy [15]

> The European Parliament has passed an anti-piracy law, covering everything from handbags to music downloads.
>
> Under the law, counterfeiters could face civil penalties, but proposals for criminal sanctions were dropped.

Before the vote, critics said the law was flawed as it applied the same penalties to both professional counterfeiters and consumers.

But a late amendment limited them to organised counterfeiters and not people downloading music at home.

The final vote on the EU Intellectual Property Rights Enforcement Directive took place in the European Parliament on 9 March. The directive was passed by 330 votes to 151.

The law was drawn up to target professional pirates, criminals and counterfeiters who make copies of goods such as football shirts or CDs.

During the debates, the directive was widened to cover any infringement of intellectual property.

The directive allows companies to raid homes, seize property and ask courts to freeze bank accounts to protect trademarks or intellectual property they believe are being abused or stolen.

Civil liberty and lobby groups feared that the music industry will also use the law to mount raids on the homes of people who swap songs via file-sharing systems such as Kazaa.

The Enforcement directive was compared to the controversial US Digital Millennium Copyright Act by Andreas Dietl, director of EU Affairs for the European Digital Rights (EDRi) lobby group.

The Recording Industry Association of America has used the DMCA to bring lawsuits against file-swappers in the US and EDRi fears the same could now happen in European countries.

The European law was shepherded through the European Parliament by MEP Janelly Fourtou, wife of Jean-Rene Fourtou who is boss of media giant Vivendi Universal.

But late amendments added to the law limited who intellectual property owners could take action against and what penalties they could apply.

One amendment said action should not be taken against consumers who download music "in good faith" for their own use.

Proposals to jail counterfeiters were also dropped from the act.

Lobbyists fear that the law could threaten press freedom in countries, such as Spain, which include confidential information in definitions of intellectual property.

In November, the EU copyright directive came into force in the UK which put many things people are used to doing with music, such as copying tracks to an MP3 player, fell into a legal grey area.

EU ministers are expected to sign off on the new rules against counterfeiting by the end of the week.

> Member states would then have 18 months to implement their own versions of the directive.

One of the most significant facts mentioned above was the comment that "The European law was shepherded through the European Parliament by MEP Janelly Fourtou, wife of Jean-Rene Fourtou who is boss of media giant Vivendi Universal." It is important to realize just how deeply the media is involved in this kind of legislation. Now that this has passed there are 18 months where each member state will write its own version of the law. You can be sure that the media people will be working to draft individual laws which are stronger than the minimum requirements.

2 Makefiles – The Basic Idea

Minimizing Work (only compile the changes) Automating Work (remember how to build the whole system)

2.1 What is a compile?

Compiling a single program involves several stages

$$preprocess- > compile- > link- > a.out$$

Compiling multiple programs involves several stages for each C file do:

$$preprocess- > compile- > file.o$$

finally do:

$$link- > a.out$$

2.2 Make without Makefile

2.2.1 Case 1: no makefile, build in steps

```
We create a new directory called pgm:
bash-2.05b# mkdir pgm
bash-2.05b# cd pgm
```

In this directory we create our first main program "mainprog.c":

```
#include <stdio.h>

int main(int argc, char* argv)
{ printf("I am the main program\n");
  printf("I depend on nothing\n");
}
```

The `make` program knows what a C file is and it knows how to compile a C file automatically. For example, we'll take a few simple steps (and missteps). First we just try "make":

```
bash-2.05b# make
make: *** No targets specified and no makefile found.  Stop.
```

Normally `make` will look for a file called Makefile or makefile. Neither was found so it complained. However, you can give command line arguments to `make` to request specific actions. Here we ask it to create `mainprog.c`

```
bash-2.05b# make mainprog.c
make: Nothing to be done for `mainprog.c'.
```

We asked `make` to do whatever it knows how to do to bring the file `mainprog.c` "up to date". This is a meaningless request since there is no program to change `mainprog.c`. We are the only ones who know how to change it. Lets try again.

This time we want the compiled form of `mainprog.c` which is the file `mainprog.o`. `make` knows how to produce a `.o` file from a `.c` file:

```
bash-2.05b# make mainprog.o
cc    -c -o mainprog.o mainprog.c
bash-2.05b# ls
mainprog.c  mainprog.o
```

You can see from the above that `make` called the C compiler with two options: `-c` which asks for only the compile step and `-o mainprog.o` which names the output file. Thus `make` did what we asked. It produced the `mainprog.o` file.

`make` also knows how to call the linker (thru the compiler). Here we ask for the executable program to be made from the `.o` file:

```
bash-2.05b# make mainprog
cc   mainprog.o   -o mainprog
bash-2.05b# ls
mainprog  mainprog.c  mainprog.o
```

This time `make` called the compiler with `mainprog.o` and asked `-o mainprog` for the output file to be called `mainprog`. In fact, we can now run it:

```
bash-2.05b# ./mainprog
I am the main program
I depend on nothing
```

2.2.2 Case 2: no makefile, build at once

Now we erase the generated files so we are back with only our source:

```
bash-2.05b# rm mainprog.o
bash-2.05b# rm mainprog
bash-2.05b# ls
mainprog.c
```

and we ask `make` to just create `mainprog` directly:

```
bash-2.05b# make mainprog
cc     mainprog.c   -o mainprog
bash-2.05b# ls
mainprog  mainprog.c
bash-2.05b# ./mainprog
I am the main program
I depend on nothing
```

`make` called the compiler with the source `mainprog.c` and asked for the output to be named `mainprog`.

Suppose we ask again:

```
bash-2.05b# make mainprog
make: 'mainprog' is up to date.
bash-2.05b# ls -l
total 16
-rwxr-xr-x    1 root     root        11593 Mar  9 05:28 mainprog
-rw-r--r--    1 root     root          125 Mar  9 05:07 mainprog.c
```

Since `mainprog` **exists and is newer** than `mainprog.c` there is nothing to do.

2.2.3 Case 3: no makefile, change the source code

Now we edit the source code so it says:

```
#include <stdio.h>

int main(int argc, char* argv)
{ printf("I am the main program\n");
  printf("I depend on nothing\n");
  printf("change 1 applied\n");
}
```

Notice that, after the edit we made, `mainprog` **exists but is older** than `mainprog.c`:

```
bash-2.05b# ls -l
total 20
-rwxr-xr-x    1 root     root        11593 Mar  9 05:28 mainprog
-rw-r--r--    1 root     root          157 Mar  9 05:34 mainprog.c
-rw-r--r--    1 root     root          125 Mar  9 05:07 mainprog.c~
```

So we ask to bring `mainprog` "up to date" again:

```
bash-2.05b# make mainprog
cc     mainprog.c   -o mainprog
bash-2.05b# ls -l
total 20
-rwxr-xr-x    1 root     root        11625 Mar  9 05:36 mainprog
-rw-r--r--    1 root     root          157 Mar  9 05:34 mainprog.c
-rw-r--r--    1 root     root          125 Mar  9 05:07 mainprog.c~
bash-2.05b# ./mainprog
I am the main program
I depend on nothing
change 1 applied
```

So if you are just using one C file you can use `make` to automatically call the compiler.

2.3 Using several files

Programs get larger but the number of changes between compiles tends to stay the same. So we spend most of our time compiling code that has *not* changed. This is one of several dozen good reasons to break the program into parts.

We are now going to change the `mainprog.c` file so it uses two other programs, `fun1` and `fun2`. The `fun1.c` program looks like:

```
void fun1()
{ printf{"fun1 called\n");
}
```

and the `fun2.c` looks like:

```
void fun2()
{ printf{"fun2 called\n");
}
```

and we change the main program to call them:

```
#include <stdio.h>
#include "fun1.c"
#include "fun2.c"

int main(int argc, char* argv)
{ printf("I am the main program\n");
  printf("I depend on fun1 and fun2\n");
  printf("change 1 applied\n");
  fun1();
  fun2();
}
```

and we try the simple approach again:

```
bash-2.05b# ls -l
total 28
-rw-r--r--    1 root     root           42 Mar  9 05:41 fun1.c
-rw-r--r--    1 root     root           42 Mar  9 05:41 fun2.c
-rwxr-xr-x    1 root     root        11625 Mar  9 05:36 mainprog
-rw-r--r--    1 root     root          213 Mar  9 05:45 mainprog.c
-rw-r--r--    1 root     root          177 Mar  9 05:45 mainprog.c~
make: *** No rule to make target `*.o'.  Stop.
bash-2.05b# make *.c
make: Nothing to be done for `fun1.c'.
make: Nothing to be done for `fun2.c'.
make: Nothing to be done for `mainprog.c'.
bash-2.05b# make *.o
make: *** No rule to make target `*.o'.  Stop.
```

Clearly `make` does not understand how to handle all of the files at once. But we've used a simple compiler trick to handle the problem. Notice that we actually `include` the C code from `fun1` and `fun2`. So all we need to do is build `mainprog` again:

```
bash-2.05b# ls -l
total 28
-rw-r--r--    1 root     root           36 Mar  9 06:02 fun1.c
-rw-r--r--    1 root     root           36 Mar  9 06:02 fun2.c
-rwxr-xr-x    1 root     root        11625 Mar  9 05:36 mainprog
-rw-r--r--    1 root     root          213 Mar  9 06:03 mainprog.c
-rw-r--r--    1 root     root          177 Mar  9 05:45 mainprog.c~
bash-2.05b# make mainprog
cc     mainprog.c   -o mainprog
bash-2.05b# ./mainprog
I am the main program
I depend on fun1 and fun2
change 1 applied
fun1 called
fun2 called
```

Of course we cheated. Nobody include C code into C code. They use `include` files. And they separately compile each file. So we rearrange things to follow convention. So the files now read:

`fun1.h`:

```
void fun1();
```

`fun1.c`:

```
#include <stdio.h>

void fun1()
{ printf("fun1 called\n");
}
```

`fun2.h`:

```
void fun2();
```

`fun2.c`:

```
#include <stdio.h>

void fun2()
{ printf("fun2 called\n");
}
```

`mainprog.c`:

```
#include <stdio.h>
#include "fun1.h"
#include "fun2.h"

int main(int argc, char* argv)
{ printf("I am the main program\n");
  printf("I depend on fun1 and fun2\n");
  printf("change 1 applied\n");
  fun1();
  fun2();
}
```

and now we try to construct `mainprog`:

```
make mainprog
cc     mainprog.c   -o mainprog
/tmp/ccApwlGj.o(.text+0x41): In function ‘main’:
: undefined reference to ‘fun1’
/tmp/ccApwlGj.o(.text+0x46): In function ‘main’:
: undefined reference to ‘fun2’
collect2: ld returned 1 exit status
make: *** [mainprog] Error 1
```

Two things happened. First, the `mainprog` program was removed:

```
bash-2.05b# ls -l
total 20
-rw-r--r--    1 root     root           61 Mar  9 06:07 fun1.c
-rw-r--r--    1 root     root           13 Mar  9 06:07 fun1.h
-rw-r--r--    1 root     root           61 Mar  9 06:08 fun2.c
-rw-r--r--    1 root     root           13 Mar  9 06:08 fun2.h
-rw-r--r--    1 root     root          213 Mar  9 06:13 mainprog.c
```

and second, `make` did not know that `mainprog.c` **depends** on compiling `fun1.c` and `fun2.c`.

2.4 Separate compiles

A Makefile is a set of rules for building programs. The rules are interpreted by `make`. The basic idea is very simple. When "programA depends on programB" we write a rule that looks like:

```
programA: programB
```

then we follow it with a set of shell commands that will "bring programA up to date". Each rule and its associated group of commands is called a "stanza". The command line in the stanza **MUST** start with a **TAB** character. This is the most common mistake when writing Makefiles.

Lets put together a simple Makefile for this project. Make will execute the first stanza in a Makefile by default. By convention this has the name "all". So our Makefile looks like:

```
all: mainprog

mainprog: mainprog.o fun1.o fun2.o
cc -o mainprog fun1.o fun2.o mainprog.o

mainprog.o: mainprog.c
cc -c mainprog.c

fun1.o: fun1.c
cc -c fun1.c

fun2.o: fun2.c
cc -c fun2.c
```

The above logic reads:

```
to bring all up to date        => bring mainprog up to date
to bring mainprog up to date   => bring mainprog.o fun1.o fun2.o up to date
to bring mainprog.o up to date => compile mainprog.c
to bring fun1.o up to date     => compile fun1.c
to bring fun2.o up to date     => compile fun2.c
to bring mainprog up to date   => link fun1.o fun2.o mainprog.o
```

So we start out with this set of files:

```
bash-2.05b# ls -l
total 36
-rw-r--r--    1 root     root           61 Mar  9 06:07 fun1.c
-rw-r--r--    1 root     root           13 Mar  9 06:07 fun1.h
-rw-r--r--    1 root     root           61 Mar  9 06:08 fun2.c
-rw-r--r--    1 root     root           13 Mar  9 06:08 fun2.h
-rwxr-xr-x    1 root     root        11897 Mar  9 18:28 mainprog
-rw-r--r--    1 root     root          219 Mar  9 18:30 mainprog.c
-rw-r--r--    1 root     root          208 Mar  9 18:30 Makefile
```

and we type `make`. We can see the compile steps take place in the order given above.

```
bash-2.05b# make
cc -c mainprog.c
cc -c fun1.c
cc -c fun2.c
cc -o mainprog fun1.o fun2.o mainprog.o
```

and we can see all of the files that got created.

```
bash-2.05b# ls -l
total 48
-rw-r--r--    1 root     root           61 Mar  9 06:07 fun1.c
-rw-r--r--    1 root     root           13 Mar  9 06:07 fun1.h
-rw-r--r--    1 root     root          788 Mar  9 18:34 fun1.o
-rw-r--r--    1 root     root           61 Mar  9 06:08 fun2.c
-rw-r--r--    1 root     root           13 Mar  9 06:08 fun2.h
-rw-r--r--    1 root     root          788 Mar  9 18:34 fun2.o
-rwxr-xr-x    1 root     root        11905 Mar  9 18:34 mainprog
-rw-r--r--    1 root     root          219 Mar  9 18:30 mainprog.c
-rw-r--r--    1 root     root          988 Mar  9 18:34 mainprog.o
-rw-r--r--    1 root     root          208 Mar  9 18:30 Makefile
```

and we can see that mainprog works.

```
bash-2.05b# ./mainprog
I am the main program
I depend on fun1 and fun2
change 1 applied
fun1 called
fun2 called
bash-2.05b#
```

Suppose we change fun2.c to output "fun2 invoked". It will look like:

```
#include <stdio.h>

void fun2()
{ printf("fun2 invoked\n");
}
```

Changing the file means that the fun2.c file is **newer** than fun2.o. `make` will follow instructions and bring fun2.o up to date by compiling fun2.c. However, `make` now knows that mainprog depends on fun2.o so it will relink mainprog. Notice that it does not need to compile fun1.c or mainprog.c as these files have not changed. So we see:

```
ls -l
total 52
-rw-r--r--    1 root     root           61 Mar  9 06:07 fun1.c
-rw-r--r--    1 root     root           13 Mar  9 06:07 fun1.h
-rw-r--r--    1 root     root          788 Mar  9 18:38 fun1.o
-rw-r--r--    1 root     root           62 Mar  9 18:43 fun2.c
-rw-r--r--    1 root     root           61 Mar  9 06:08 fun2.c~
-rw-r--r--    1 root     root           13 Mar  9 06:08 fun2.h
-rw-r--r--    1 root     root          788 Mar  9 18:38 fun2.o
```

```
-rwxr-xr-x    1 root     root        11905 Mar  9 18:38 mainprog
-rw-r--r--    1 root     root          219 Mar  9 18:30 mainprog.c
-rw-r--r--    1 root     root          988 Mar  9 18:38 mainprog.o
-rw-r--r--    1 root     root          208 Mar  9 18:30 Makefile
bash-2.05b# make
cc -c fun2.c
cc -o mainprog fun1.o fun2.o mainprog.o
bash-2.05b# ./mainprog
I am the main program
I depend on fun1 and fun2
change 1 applied
fun1 called
fun2 invoked
bash-2.05b# ls -l
total 52
-rw-r--r--    1 root     root           61 Mar  9 06:07 fun1.c
-rw-r--r--    1 root     root           13 Mar  9 06:07 fun1.h
-rw-r--r--    1 root     root          788 Mar  9 18:38 fun1.o
-rw-r--r--    1 root     root           62 Mar  9 18:43 fun2.c
-rw-r--r--    1 root     root           61 Mar  9 06:08 fun2.c~
-rw-r--r--    1 root     root           13 Mar  9 06:08 fun2.h
-rw-r--r--    1 root     root          788 Mar  9 18:43 fun2.o
-rwxr-xr-x    1 root     root        11905 Mar  9 18:43 mainprog
-rw-r--r--    1 root     root          219 Mar  9 18:30 mainprog.c
-rw-r--r--    1 root     root          988 Mar  9 18:38 mainprog.o
-rw-r--r--    1 root     root          208 Mar  9 18:30 Makefile
```

If we were to try `make` again without changing anything we get:

```
bash-2.05b# make
make: Nothing to be done for ‘all’.
```

So the basic idea is that you can write rules that tell `make` exactly how to construct your program.

References

[1] Smith, Tony,
"Judge bans DVD X Copy software",
"http://www.theregister.co.uk/content/54/35730.html"

[2] Smith, Tony,
"Kazaa trial judge delays hearing",
"http://www.theregister.co.uk/content/6/35732.html"

[3] slashdot.org,
"Rome Moving to Linux",
"http://slashdot.org", Feb 26, 2004 9:25AM

[4] slashdot.org,
"BBC Discusses PVR Software, Creative Archive Plans",
"http://slashdot.org", Feb 26, 2004 5:46AM

[5] Moglen, Eban,
"Freeing the Mind: Free Software and the Death of Proprietary Culture",
"http://moglen.law.columbia.edu/publications/maine-speech.html"

[6] The Register,
"MPs call for Gov.uk to switch to open source (maybe)",
"http://www.theregister.co.uk/content/4/35876.html"

[7] Fink, Martin,
"How to hire great open source developers",
"http://management.itmanagersjournal.com/management/04/02/27/2210254.shtml"

[8] China and Linux,
"New Law in China to End Microsoft's Dominance"
"http://www.financialexpress.com/fe_full_story.php?content_id=53682"

[9] Isreal and Linux,
"Closing Windows: free Hebrew alternative for Microsoft system"
"http://www.maarivintl.com/index.cfm?fuseaction=article&articleID=3382"

[10] Blow, Jonathan,
"Game Development: Harder Than You Think",
"http://www.acmqueue.com/modules.php?name=Contents&pa=showpage&pid=114"

[11] The Register,
"DVD Forum mandates Microsoft for HD disc standard",
"http://www.theregister.co.uk/content/54/35972.html"

[12] Slashdot,
"Australia-U.S. Trade Agreement Contains DMCA-like Provisions",
"http://www.dfat.gov.au/trade/negotiations/us_fta/text/index.html"

[13] Haines, Lester,
"Brazil loves Linux - true",
"www.theregister.co.uk/content/4/36050.html"

[14] Vaughn, G., Elliston, B., Tromey, T., Taylor, I.,
"Gnu Autoconf, Automake, and Libtool",
New Riders Publishing, ISBN 1-57870-190-2 (2000)

[15] BBC News,
"EU backs tighter rules on piracy",
"http://news.bbc.co.uk/2/hi/technology/3545839.stm"