Compiling without Make

Say we have a program broken into two files (prog.c and sub.c), and each begins with

```c
#include "incl.h"
```

We can compile the program in one line:

```
eddie% gcc prog.c sub.c
```

Or we can compile the parts separately:

```
eddie% gcc -c prog.c
eddie% gcc -c sub.c
eddie% gcc prog.o sub.o
```
There’s not much point to doing it separately, except when only parts of the program need to be recompiled.

E.g., say I've changed sub.c and nothing else. I don’t need to recompile prog.c.

```
  eddie% gcc -c sub.c
  eddie% gcc prog.o sub.o
```

This becomes really valuable when we have many files. But then it becomes hard to keep track of what needs to be recompiled.

This is what makefiles help us with.

### Makefiles

**General format:** A makefile consists of pairs of lines as follows:

```
  label: item1 ... itemk
  command
```

**Meaning:** To ensure that `label` is up to date:

1. Recursively ensure that `item1 ... itemk` are themselves up to date.

2. If file `label` is older than any of the files `item1 ... itemk`, then file `label` is out of date. Execute `command` to bring it up to date.

**Example:**

```
  prog.o: prog.c incl.h
    gcc -c prog.c
  prog.o: incl.h prog.o sub.o
    gcc -o prog prog.o incl.o
  sub.o: incl.h sub.c
    gcc -c sub.o
```

Running make

Script started on Sun Jan 19 17:13:22 1997

eddie% ls
README    makefile    prog.c    typescript
incl.h    pgm         sub.c

eddie% make
gcc -c prog.c
gcc -c sub.c
gcc -o pgm prog.o sub.o

eddie% pgm
hi
hi
there
there
there

eddie% vi prog.c
eddie% ls -l
total 15
-rw-r--r-- 1 dianeh 364 Sep 13 1994 README
-rw-r--r-- 1 dianeh 406 Sep 13 1994 incl.h
-rw-r--r-- 1 dianeh 477 Sep 13 1994 makefile
-rwx------- 1 dianeh 5844 Jan 19 17:13 pgm
-rw-r--r-- 1 dianeh 1247 Jan 19 17:14 prog.c
-rwx------- 1 dianeh 1416 Jan 19 17:13 prog.o
-rw-r--r-- 1 dianeh 774 Jan 19 16:34 sub.c
-rwx------- 1 dianeh 844 Jan 19 17:13 sub.o
-rw------- 1 dianeh 0 Jan 19 17:13 typescript

eddie% make
gcc -c prog.c
gcc -o pgm prog.o sub.o

eddie% vi incl.h

eddie% make
gcc -c prog.c
gcc -c sub.c
gcc -o pgm prog.o sub.o

eddie% make
‘pgm’ is up to date.

eddie% make
‘pgm’ is up to date.

eddie% make
‘sub.c’ is up to date.
Tabs are Crucial

You absolutely must put a tab before each compile command. If you use blanks instead, you will get an unhelpful message.

Example

Here’s a silly makefile. The white space before the g++ command is made of blanks.

```
% cat diane1
blah: fred barney
     g++ betty
```

Here’s what happens when I use it:

```
% make -f diane1
make: Fatal error in reader: diane1, line 3:
     Unexpected end of line seen
```