Makefiles

Reading:

- King, chapter 15
- `man make`

Compilation without Make

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

```
#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 *itemk*, then file *label* is out of date.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:

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

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
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
-rw-------   1 dianeh       1416 Jan 19 17:13 prog.o
-rw-r--r--   1 dianeh        774 Jan 19 16:34 sub.c
-rw-------   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% make
'pgm' is up to date.

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 sub.c
'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
```


But how can I "see" my tabs?

Tabs and blanks look alike when you look at a file using programs like cat or an editor. Here is a corrected make file, with a tab instead of blanks before the g++ command:

```
% cat diane2
blah: fred barney
        g++ betty
```

One way to see that they're different is to use Unix's diff:

```
% diff diane1 diane2
2c2
<         g++ betty
---
>       g++ betty
```

But it's still hard to see what's in the file because blanks and tabs just look like "white space"

Use `od` to see your tabs and blanks

Unix's `od` will show you exactly what's in your file, byte by byte.

If you use it with the -b flag, it groups the digits in its output such that each (3-bit) group represents exactly one byte.

```
% od -b diane1
0000000 142 154 141 150 072 040 146 162 145 144 040 142 141 162 156 145
0000020 171 012 040 040 040 040 040 040 040 040 147 053 053 040 142 145
0000040 164 164 171 012
0000044
% od -b diane2
0000000 142 154 141 150 072 040 146 162 145 144 040 142 141 162 156 145
0000020 171 012 011 147 053 053 040 142 145 164 164 171 012
0000035
```


Output from `od -c` is easier to read

If you use od with the -c flag, it prints those bytes which represent characters (in our case that's all of them) as characters.

```
% od -c diane1
0000000   b   l   a   h   :       f   r   e   d       b   a   r   n   e
0000020   y  \n                                   g   +   +       b   e
0000040   t   t   y  \n
0000044
% od -c diane2
0000000   b   l   a   h   :       f   r   e   d       b   a   r   n   e
0000020   y  \n  \t   g   +   +       b   e   t   t   y  \n
0000035
```

Other pitfalls with make and tabs

If put in a blank before the tab, make will also freak out. Yuk!

But at least now you have some tools for figuring out the problem.