#include

Dave Eckhardt
de0u@andrew.cmu.edu

Synchronization

10/10/03	Friday	Checkpoint 1 due
10/13/03	Monday	HW1 due
10/15/03	Wednesday	Mid-term exam (pending)
10/17/03	Friday	Mid-semester break
10/20/03	Monday	Mid-term grades due

Outline

- #ifndef DSFLK_FSFDDS_FSDFDS
- What ***should*** go here, anyway?

What's _STDIO_H_ anyway?

```
#ifndef _STDIO_H_
#define _STDIO_H_

typedef struct FILE {
  ...
} ...;

#endif /* _STDIO_H_ */
```

Archaeology

- C is old
- C doesn't have modules
- C has ***files***
  - Compilers sort of know some file types: .c, .s
  - Compilers ***don't*** really know about .h
    - Auxiliary “pre-processor” brain hides them
- People use ***conventions*** to get module-like C
  - These conventions evolved slowly

The “.h Responsibility” Dilemma

- Assume: “stdio module”
- Assume: “network stack module”
  - (Trust us, it's modular!)
- Both need to know
  - What's a `size_t` on this machine, anyway?
  - #include <sys/types.h>

Nested Responsibility

- Program 1:
  - #include <stdio.h>
- Program 2:
  - #include <netinet/tcp_var.h>
- Assume
  - Program 1, 2 don't need sys/types.h themselves
- Solution 1
  - stdio.h and netinet/tcp_var.h each include sys/types.h

Too Much

- Program 3:
  - #include <stdio.h>
  - #include <netinet/tcp_var.h>
- Problem
  - Now we get ***two*** copies sys/types.h
  - Lots of whining about redefinitions
  - Maybe compilation fails

Passing the Buck

- Blame the user!
- Solution 2
  - Require ***main program*** to #include <sys/types.h>
- Problem
  - Annoying for user
  - Modules' needs ***change over time***
    - Didn't you know? Since last night xxx needs yyy...

Solution: Idempotent .h files

- .h responsibility
  - Activate only once
  - No matter how many times included
  - Choose string “unlikely to be used elsewhere”

```
#ifndef _STDIO_H_
#define _STDIO_H_
...
#endif /* _STDIO_H_ */
```

What ***Belongs*** In a .h?

- Types
- Exported interface routines (“public methods”)
- Constants
- Macros (when appropriate)
- Data items exported by module
  - Try to avoid this
  - Same reason as other languages: data != semantics
- ***No code!***

But What About...?

- Real modules have multiple .c files
  - Who declares ***internal*** data structures?
    - (Internally, we agree on semantics)
  - Who declares ***internal*** functions?
- Not “the” .h file
  - We ***don't want*** to publish internal details
- Maybe a “.i” file?
  - Help?

Use the ***Other*** .h File!

- stdio.h
  - Included by module clients
  - Included by module parts
- stdio_private.h
  - Included only by module parts
  - Ideally, not available to user's prying eyes
- *_private.h should be idempotent, too

Summary

- #ifndef DSFLK_FSFDDS_FSDFDS
  - Well, use a better string
  - Used to make .h files idempotent
- What ***should*** go here, anyway?
  - There are two “here”'s here
    - foo.h: public interface, available to public
    - foo_private.h: internal communication, maybe unpublished