CSCI 331:
Introduction to Computer Security

Lecture 5: C wrap-up

Instructor: Dan Barowy

Williams

Announcements

- CS Colloquium this **Friday, Sept 22 @ 2:35pm in Wege Auditorium (TCL 123)**

Your classmates

What I Did Last Summer, Industry Edition

Short presentations by your fellow CS students about internship experiences in industry. CS Colloquium credit awarded for attendance.

Topics

Office hours:
Thursday 4-5:30pm (**TBL 301**)
Friday 12:30-1:30pm (**TBL 301**)

Reading discussion (Davis)

Libraries and headers

Bugs in C programs

Public key encryption

Your to-dos

1. Lab 1 **due Sunday 9/24** by 10:00PM.
2. Read *Why Stolen Password Databases Are a Problem* **for Thu, 9/28**.
   1. Please take notes.
3. Project part 1 due **Sunday, 10/1**.
4. Sign up for *What I Did Last Summer*

Reading discussion

Makefiles

Makefiles

A `Makefile` is a **specification** used by the `make` tool to **automate** the compilation of programs.

Rationale

Programmers build software **frequently**.

Lazy
(don’t want to retype)

Impatient
(don’t want to wait for gcc)

Insight

An entire project does not need to rebuilt every time.

a.c

```
int foo (int a) {
  return a + 1;
}
```

b.c

```
int bar (int b) {
  return b - 1;
}
```

depends on

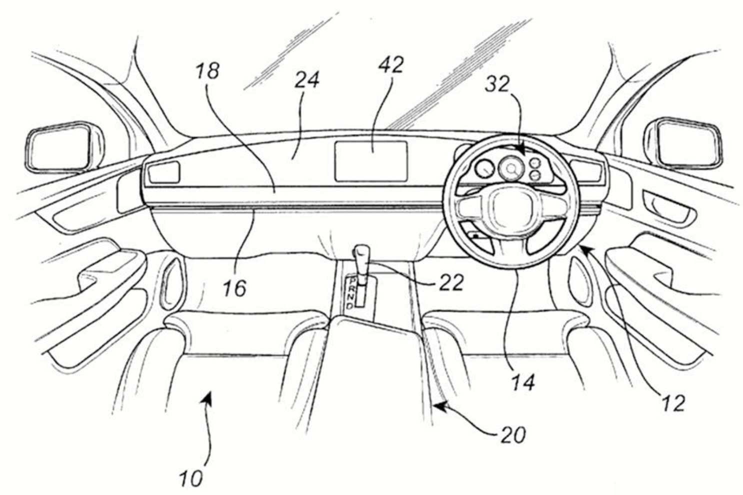
depends on

c.c

```
#include <stdio.h>
#include "a.h"
#include "b.h"

int main() {
  int c = bar(foo(1));
  printf("c = %d\n", c);
  return 0;
}
```

What are .h files?



What are .h files?

A .h file provides **interface** information so that a compiler can **separately compile** sources.

a.c

```
int foo (int a) {
  return a + 1;
}
```

a.h

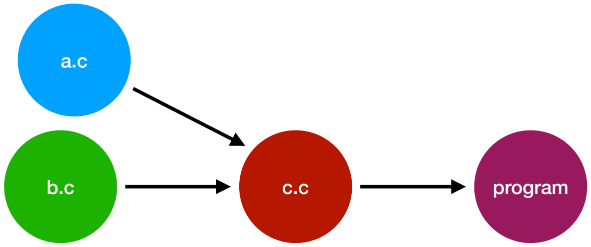
```
int foo (int a);
```

Should we put .h files in our Makefile?

Ask yourself: “if a file changes, should I rebuild?”

Answer: yes! If an interface changes, we should recompile.

A Makefile encodes dependencies



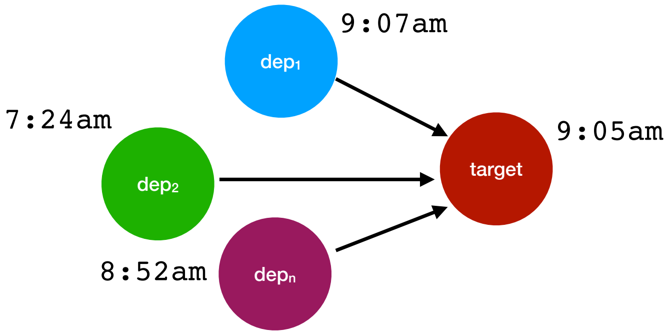
```
$ gcc a.c b.c c.c -o program
```

Small catch: make can only avoid rebuilding if there is a **thing** that it can avoid producing.

program: $\mathtt{dep}_1$ $\mathtt{dep}_2$ … $\mathtt{dep}_n$

    gcc -o target $\mathtt{dep}_1$ $\mathtt{dep}_2$ … $\mathtt{dep}_n$

Incremental rebuild



```
$ make target
```

1. make checks to see that no dep is newer than target, *recusively*
2. if so, it rebuilds the current target.

Activity

```
login0: console.o database.o login.c
  gcc -o login0 console.o database.o login.c

console.o: console.c console.h
  gcc -c console.c

database.o: database.c database.h
  gcc -c database.c
```

1. Draw the dependence graph for this `Makefile`.
2. Assume that the project is built with `make login0`, `database.h` is then updated, and then the user types `make login0` again. What commands are run?

Libraries: static vs shared

Libraries: static vs shared

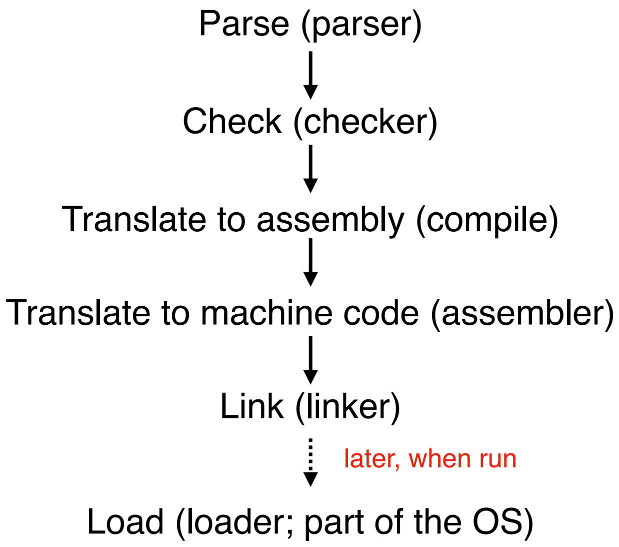
Static libraries are copied into program.

Shared libraries leave a “forwarding address”.

Static library:	`library.o`
Shared library:	`library.so`

Shared libraries must be linked with the `-l<libraryname>` linker flag for `gcc`.

What a compiler does



Parse (parser)

Check (checker)

Translate to assembly (compile)

Translate to machine code (assembler)

Link (linker)

later, when run

Load (loader; part of the OS)

Public keys