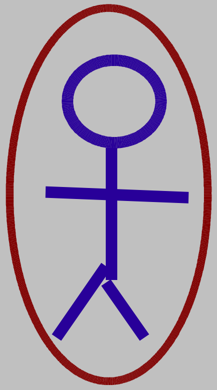
(not a mascot)



Operating System (without a name) (so far)

Benji, Jaap & Ting

Traditional “timesharing” OS

- Protect processes from each other
- Usual trick:
  - Virtualization (pretend we have >1 machine)
- Problems:
  - Context switches are slow
  - Copying buffers is slow
  - Too little security (big rooms)
  - Too much security (big walls)

What is the actual goal?

To protect <u>resources</u>!

```
char* vga = (char *) 0xb8000;
strcpy(“H e l l o “, vga);
```

- UNIX programmer says:
  - “Shouldn't have that address mapped.”
- ML/Lisp/have-your-pick programmer says:
  - `(char *)` is inherently unsafe

Loading user code in UNIX

- fork(2) creates a new memory space
- exec(2) loads an object file into memory
- Figure out where main() is and jump to it

Protection through virtualization

Loading user code in (unnamed)

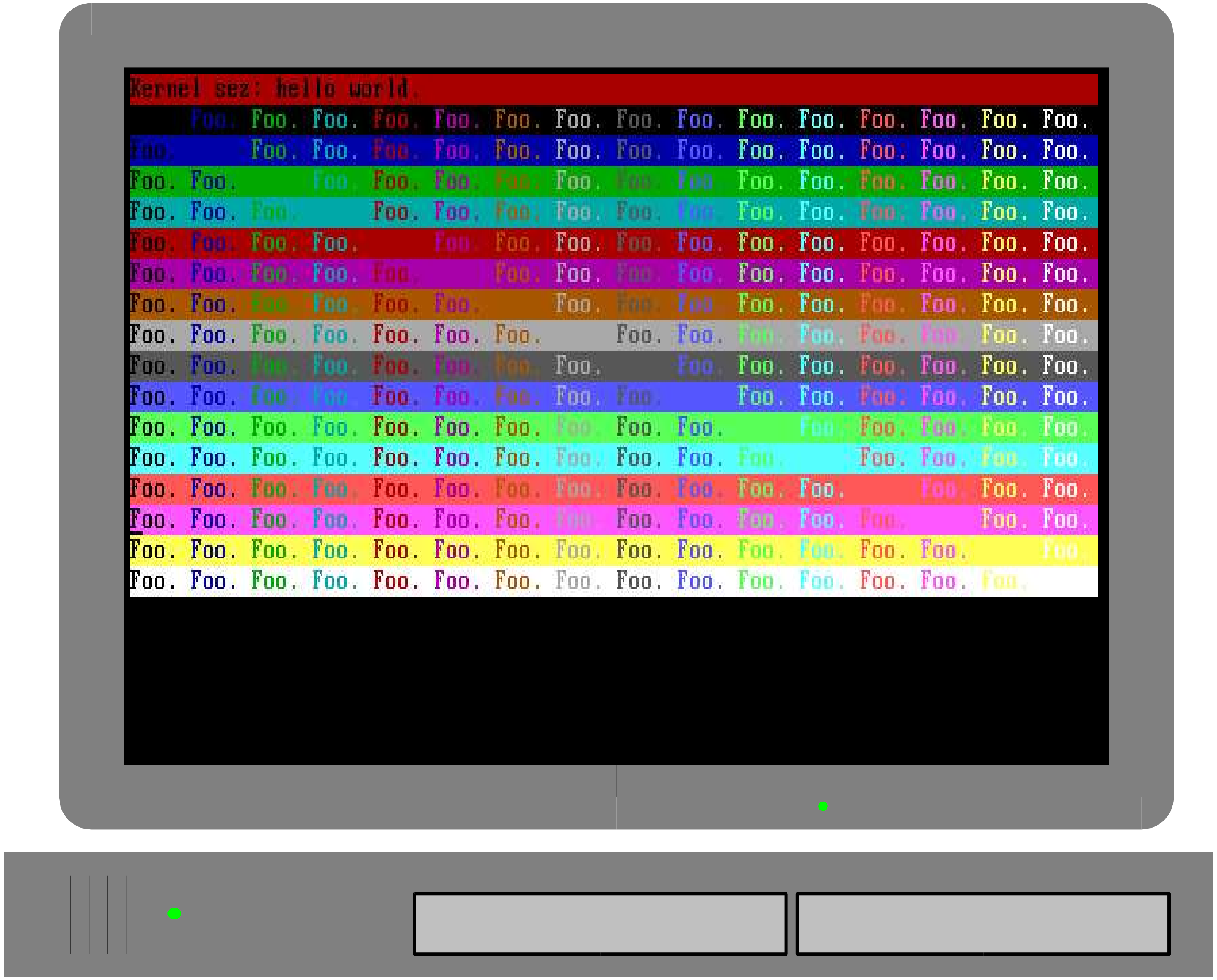
- Check that code came out of a type-safe compiler
- Do *not* generate a new memory map
- Load an object file into memory
- Figure out where main() is and jump to it

Protection without virtualization

Why does type safety help?

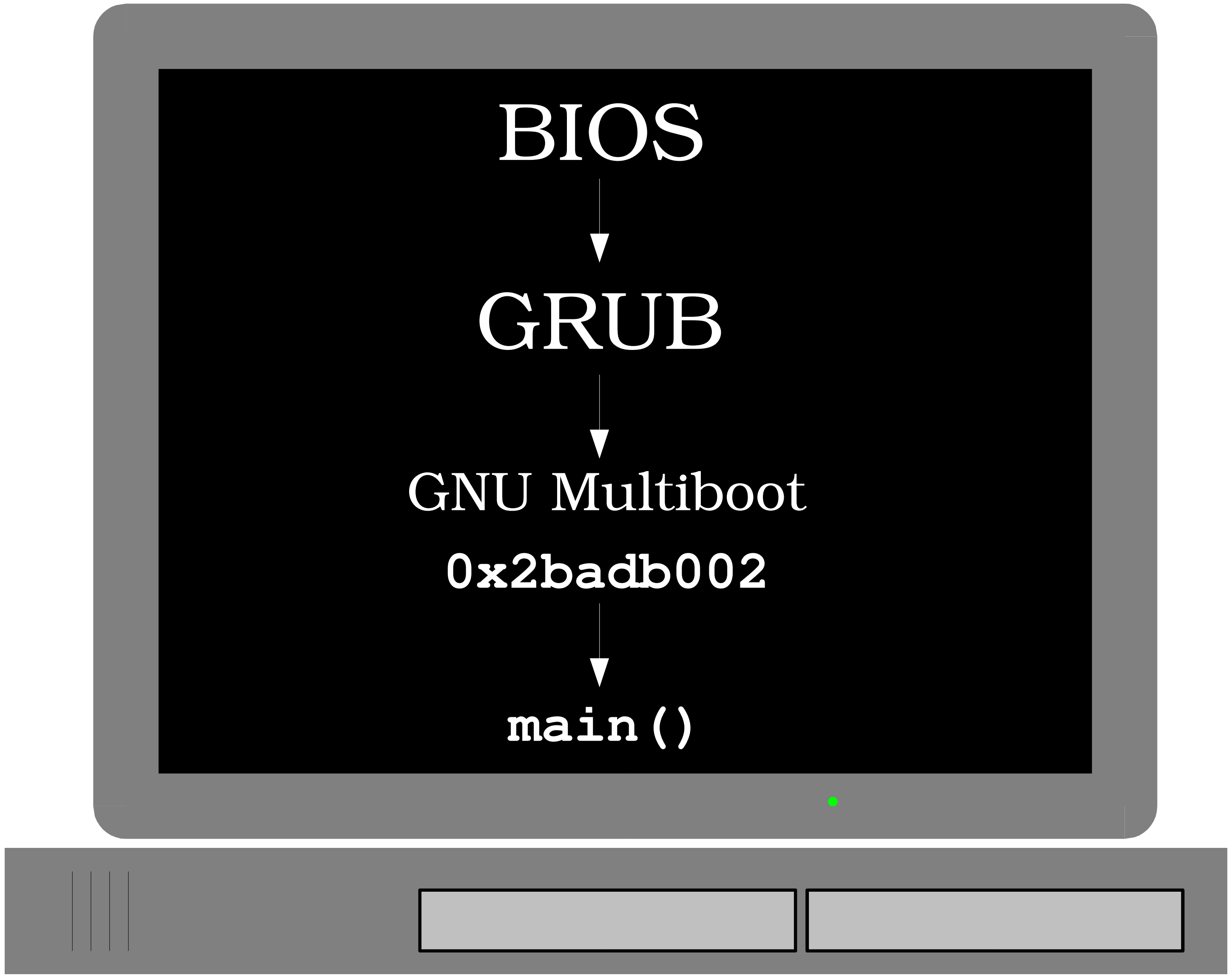
- *Theorem: "While this program is being executed, no memory outside of range R will be accessed, and control will not be transferred outside of range U."*
- *Proof:*
  - For a correctly typed program: trivial
    - Pointers don't appear out of thin air
  - For a C program: impossible
    - Pointers do appear out of thin air

(unnamed OS) so far....



```
Kernel sez: hello world.
     Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo.
Foo.      Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo.
Foo. Foo.      Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo.
Foo. Foo. Foo.      Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo.
Foo. Foo. Foo. Foo.      Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo.
Foo. Foo. Foo. Foo. Foo.      Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo.
Foo. Foo. Foo. Foo. Foo. Foo.      Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo.
Foo. Foo. Foo. Foo. Foo. Foo. Foo.      Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo.
Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo.      Foo. Foo. Foo. Foo. Foo. Foo. Foo.
Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo.      Foo. Foo. Foo. Foo. Foo. Foo.
Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo.      Foo. Foo. Foo. Foo. Foo.
Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo.      Foo. Foo. Foo. Foo.
Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo.      Foo. Foo. Foo.
Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo.      Foo. Foo.
Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo.      Foo.
Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo. Foo.
```

Booting made easy



Emulation made easy

```
qemu -m 16 -fda boot -fdb root
```

Next: basic device drivers

- VGA
- 8259 PIC (Interrupt controller)
- Keyboard
- Floppy disk
- Hard drive
- File system