CSCI 6907: Special Topics in Operating Systems

Your Prof: Gabe

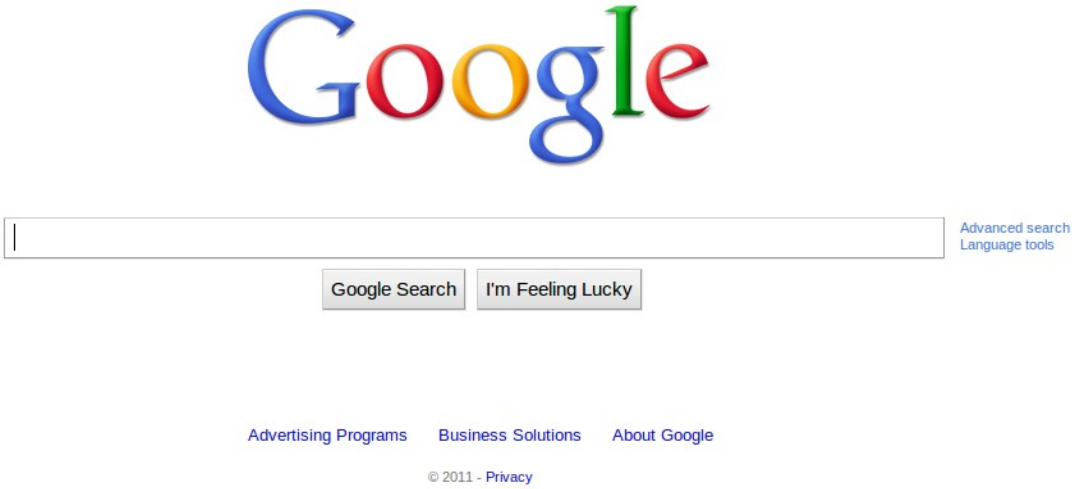
Acknowledgements: Some slide material derived from Silberschatz, et al.

Today

- Syllabus and class plan
- Piazza
- Homework – Due Jan 31st
  - This assignment **will take 2 weeks, at least**
  - *start now – tonight!*
- Structure – presentations/reading/project
  - will place a sign up sheet online soon – see piazza
- Project
- Project

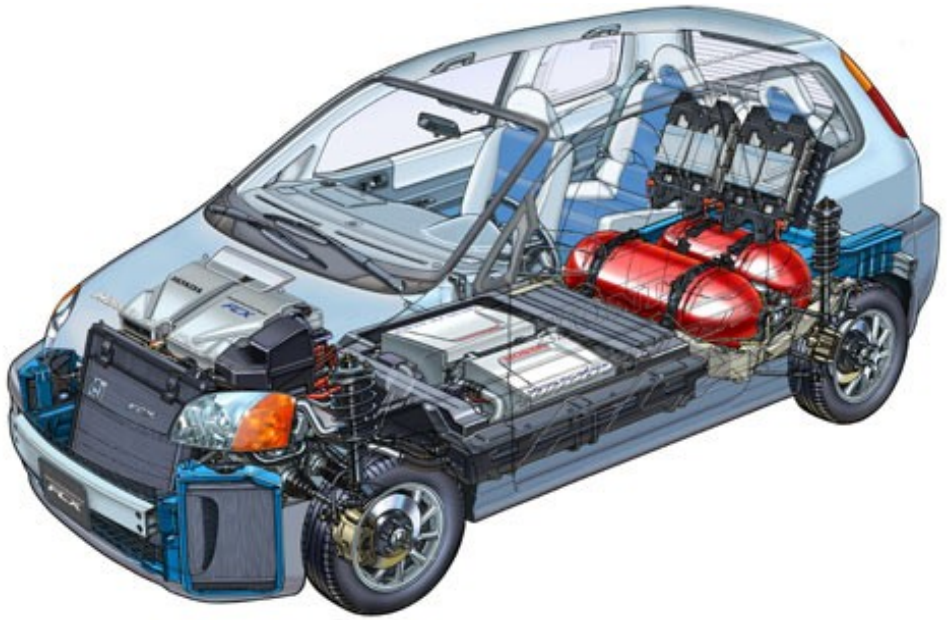
“High-level”

Cars



Computers

...details...

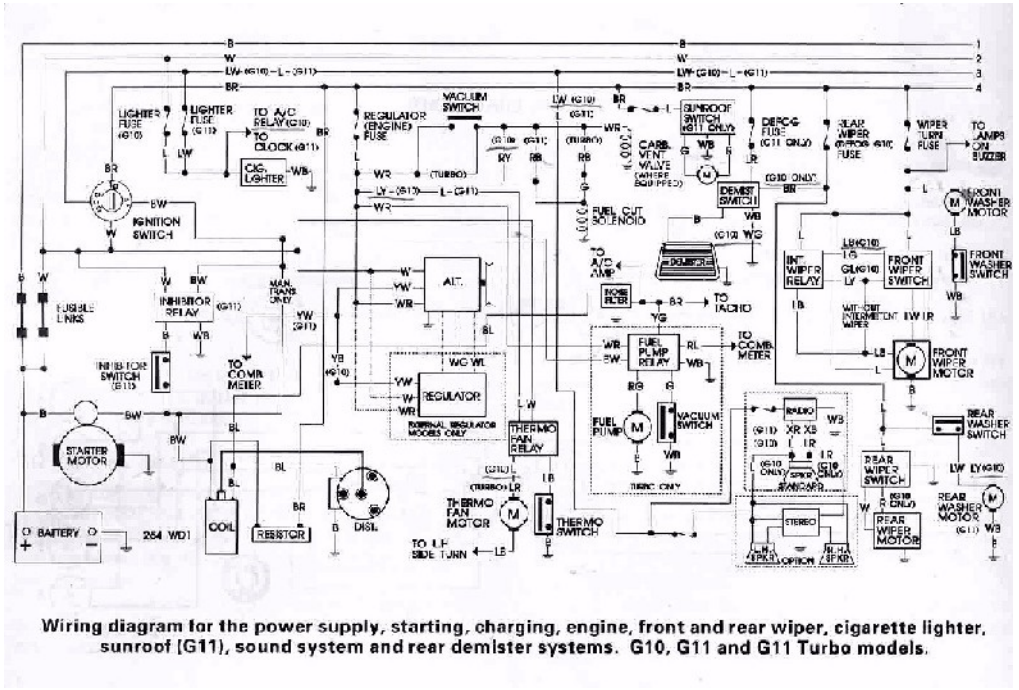


```
/*
 * So far all flags should be taken in the context of the
 * actual invoking thread (they effect the thread switching
 * _from_ rather than the thread to switch _to_) in which case
 * we would want to use the sched_page flags.
 */
flags = rflags;
switch_thread_update_flags(da, &flags);

if (unlikely(flags)) {
        thd = switch_thread_slowpath(curr, flags, curr_spd, rthd_id, da, &ret_c
                                     &curr_sched_flags, &thd_sched_flags);
        /* If we should return immediately back to this
         * thread, and its registers have been changed,
         * return without setting the return value */
        if (ret_code == COS_SCHED_RET_SUCCESS && thd == curr) goto ret;
        if (thd == curr) goto_err(ret_err, "slooooow\n");
} else {
        next_thd = switch_thread_parse_data_area(da, &ret_code);
        if (unlikely(0 == next_thd)) goto_err(ret_err, "data_area\n");

        thd = switch_thread_get_target(next_thd, curr, curr_spd, &ret_code);
        if (unlikely(NULL == thd)) goto_err(ret_err, "get target");
}

/* If a thread is involved in a scheduling decision, we should
 * assume that any preemption chains that existed aren't valid
 * anymore. */
break_preemption_chain(curr);
```

Cars

Computers

...“low-level”



Wiring diagram for the power supply, starting, charging, engine, front and rear wiper, cigarette lighter, sunroof (G11), sound system and rear demister systems. G10, G11 and G11 Turbo models.

```
0000000000400b0 <__libc_csu_init>:
  4006b0:       48 89 6c 24 d8          mov    %rbp,-0x28(%rsp)
  4006b5:       4c 89 64 24 e0          mov    %r12,-0x20(%rsp)
  4006ba:       48 8d 2d 53 07 20 00    lea    0x200753(%rip),%rbp        # 600e14 <__init_array_end>
  4006c1:       4c 8d 25 4c 07 20 00    lea    0x20074c(%rip),%r12        # 600e14 <__init_array_end>
  4006c8:       4c 89 6c 24 e8          mov    %r13,-0x18(%rsp)
  4006cd:       4c 89 74 24 f0          mov    %r14,-0x10(%rsp)
  4006d2:       4c 89 7c 24 f8          mov    %r15,-0x8(%rsp)
  4006d7:       48 89 5c 24 d0          mov    %rbx,-0x30(%rsp)
  4006dc:       48 83 ec 38             sub    $0x38,%rsp
  4006e0:       4c 29 e5                sub    %r12,%rbp
  4006e3:       41 89 fd                mov    %edi,%r13d
  4006e6:       49 89 f6                mov    %rsi,%r14
  4006e9:       48 c1 fd 03             sar    $0x3,%rbp
  4006ed:       49 89 d7                mov    %rdx,%r15
  4006f0:       e8 33 fd ff ff          callq  400428 <_init>
  4006f5:       48 85 ed                test   %rbp,%rbp
  4006f8:       74 1c                   je     400716 <__libc_csu_init+0x66>
  4006fa:       31 db                   xor    %ebx,%ebx
  4006fc:       0f 1f 40 00             nopl   0x0(%rax)
  400700:       4c 89 fa                mov    %r15,%rdx
  400703:       4c 89 f6                mov    %r14,%rsi
  400706:       44 89 ef                mov    %r13d,%edi
  400709:       41 ff 14 dc             callq  *(%r12,%rbx,8)
  40070d:       48 83 c3 01             add    $0x1,%rbx
  400711:       48 39 eb                cmp    %rbp,%rbx
  400714:       72 ea                   jb     400700 <__libc_csu_init+0x50>
  400716:       48 8b 5c 24 08          mov    0x8(%rsp),%rbx
  40071b:       48 8b 6c 24 10          mov    0x10(%rsp),%rbp
  400720:       4c 8b 64 24 18          mov    0x18(%rsp),%r12
  400725:       4c 8b 6c 24 20          mov    0x20(%rsp),%r13
  40072a:       4c 8b 74 24 28          mov    0x28(%rsp),%r14
  40072f:       4c 8b 7c 24 30          mov    0x30(%rsp),%r15
  400734:       48 83 c4 38             add    $0x38,%rsp
  400738:       c3                      retq
  400739:       90                      nop
  40073a:       90                      nop
  40073b:       90                      nop
  40073c:       90                      nop
  40073d:       90                      nop
  40073e:       90                      nop
  40073f:       90                      nop
```

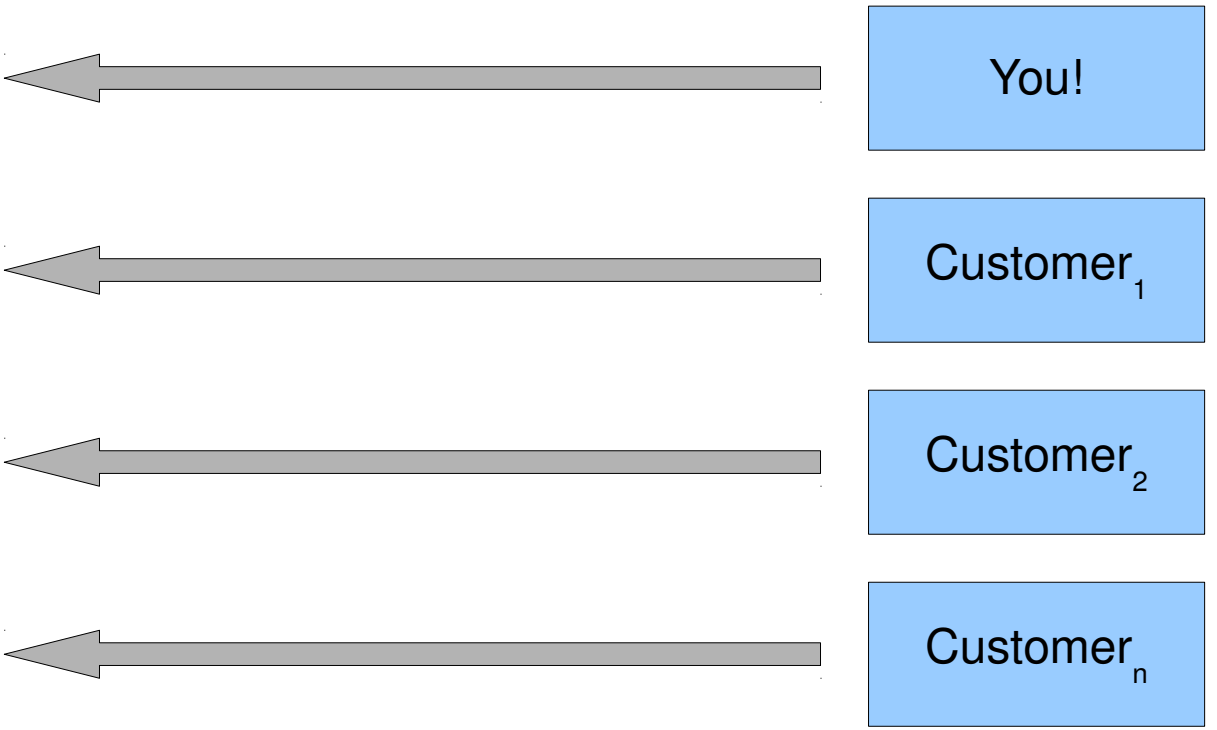
Cars

Computers

What is an Operating System!?

?

What is an OS: Analogy



What is an OS: Analogy

You!

Customer$_1$

Customer$_2$

Customer$_n$

What is an OS: Analogy

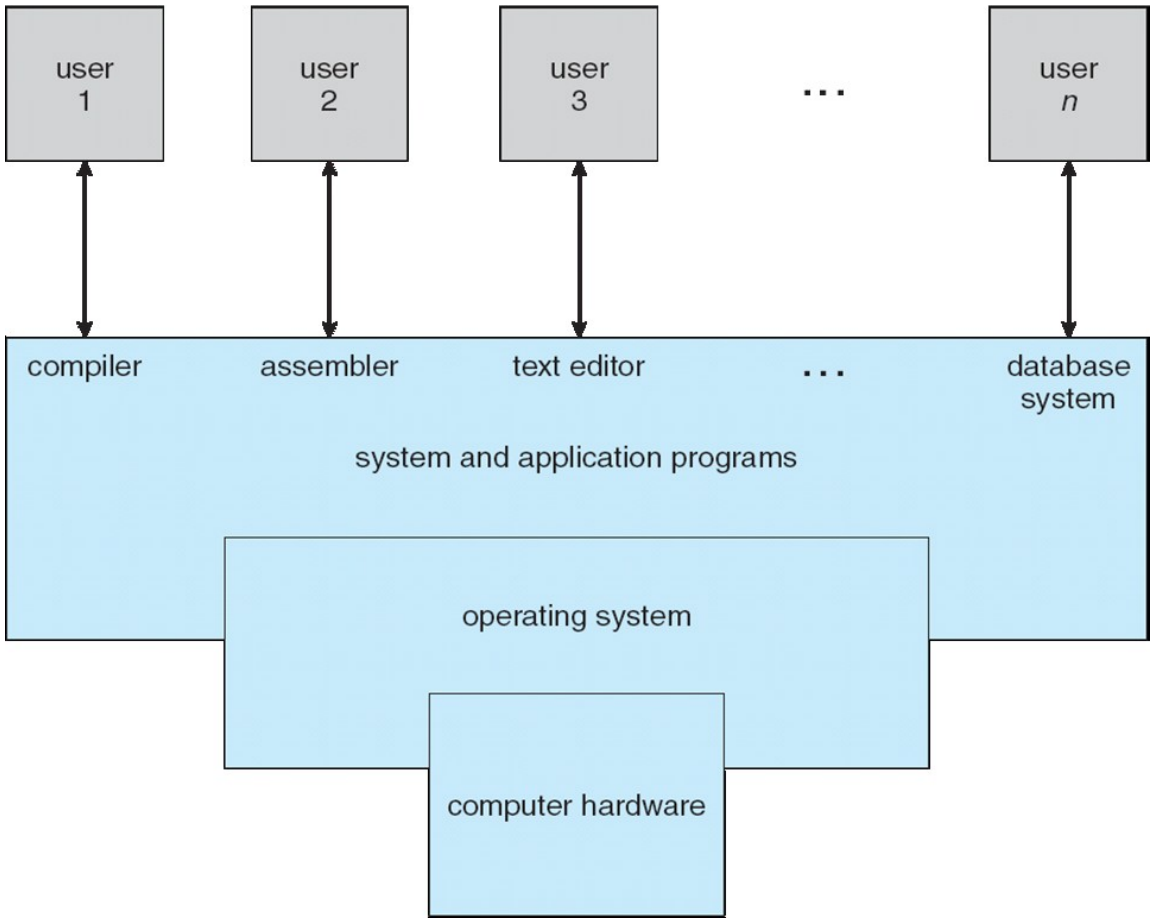
Hardware

Operating System

Applications

- You!
- Customer$_1$
- Customer$_2$
- Customer$_n$

OS as Abstraction: System Layers



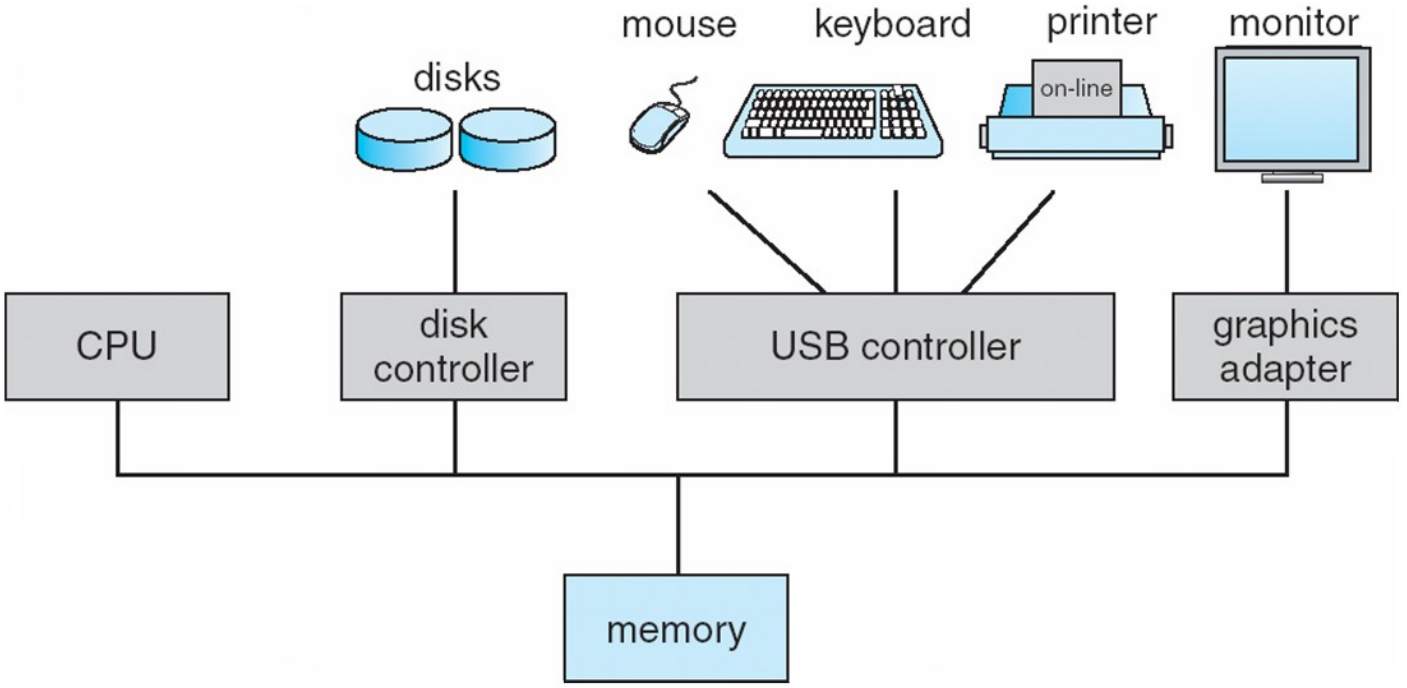
Operating System as Abstraction

- *"The effective exploitation of his powers of abstraction must be regarded as one of the most vital activities of a competent programmer."* - Edsger W. Dijkstra

- Provides abstractions for resources (memory, CPU, disk) and controls application execution
- Provide environment for application execution
- Protection and Isolation
- OS as the provider of system *abstractions*

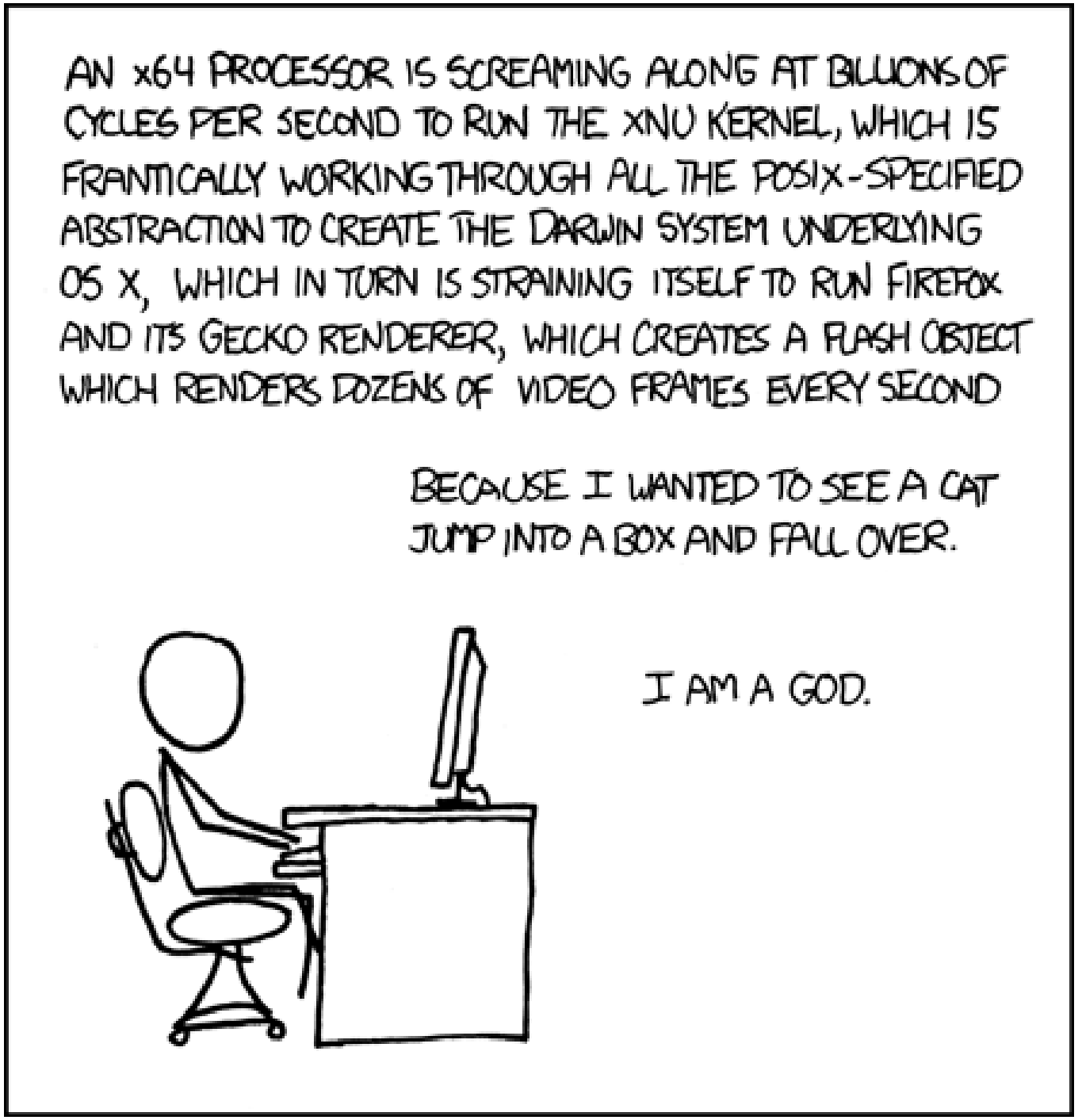
Computers as Distributed Systems

"Hardware: The parts of a computer system that can be kicked."
- Jeff Pesis



OS as Resource Manager

- Control a diverse set of hardware
  - Processors
  - Memory
  - Disks
  - Networking cards
  - Video cards
- Coordinates these hardware resources amongst user programs
  - Want to use resources efficiently!
- OS as a *resource manager/multiplexer*



Source: xkcd.com

What is Operating Systems Research!?

?

OS Research

- Systematic investigation of novel, interesting, and useful techniques to
  - Better utilize resources
    - System performance, predictability
  - Provide new programming abstractions
    - Enable systems to do things they weren't previously able
      - Recover from faults
      - Tolerate malicious users
    - More easily do the same thing
      - RPC

Why
Operating Systems Research!?

?

Worse is Better

- Background
  - Lisp: functional programming language
    - Automatic memory management – garbage collection
    - **Simple** semantics
      - lambda, if, numbers, booleans, application (function call), cons
      - ...yet amazingly powerful
  - vs. C
    - Manual memory management (malloc, free)
    - **No** semantics – *anything* can happen...and does!
- Which won? Why?

Worse is Better

- Read Richard Gabriel's piece, then Linus' email
- Write on back:
  - What is Richard's main thesis?
  - Do you agree? Why?
  - What are three examples of worse is better?
  - Any examples of better is better?

- What does this have to do with OS research?

This class...again

- Presenting papers
  - An acquired skill...takes diligent practice
  - If you are worried about this, spend more time on it!
- Reading papers
  - An acquired skill...takes diligent practice
  - Every week...so you will practice this!
- Semester-long project!

The Nucleus

- An example of “better”
- Practice reading a research paper
  - Read up to section 5
- Critically Reading a Research Paper:
  - What is the central point/contributions?
    - 1-3 sentences
  - Three most significant issues/problems/limitations with the approach?
    - Key: *critically* read the paper
  - Three questions you have about the approach?
    - Often derived from the paper's motivation, design, or evaluation

What/Why

Operating Systems Research!?

How
Operating Systems Research!?

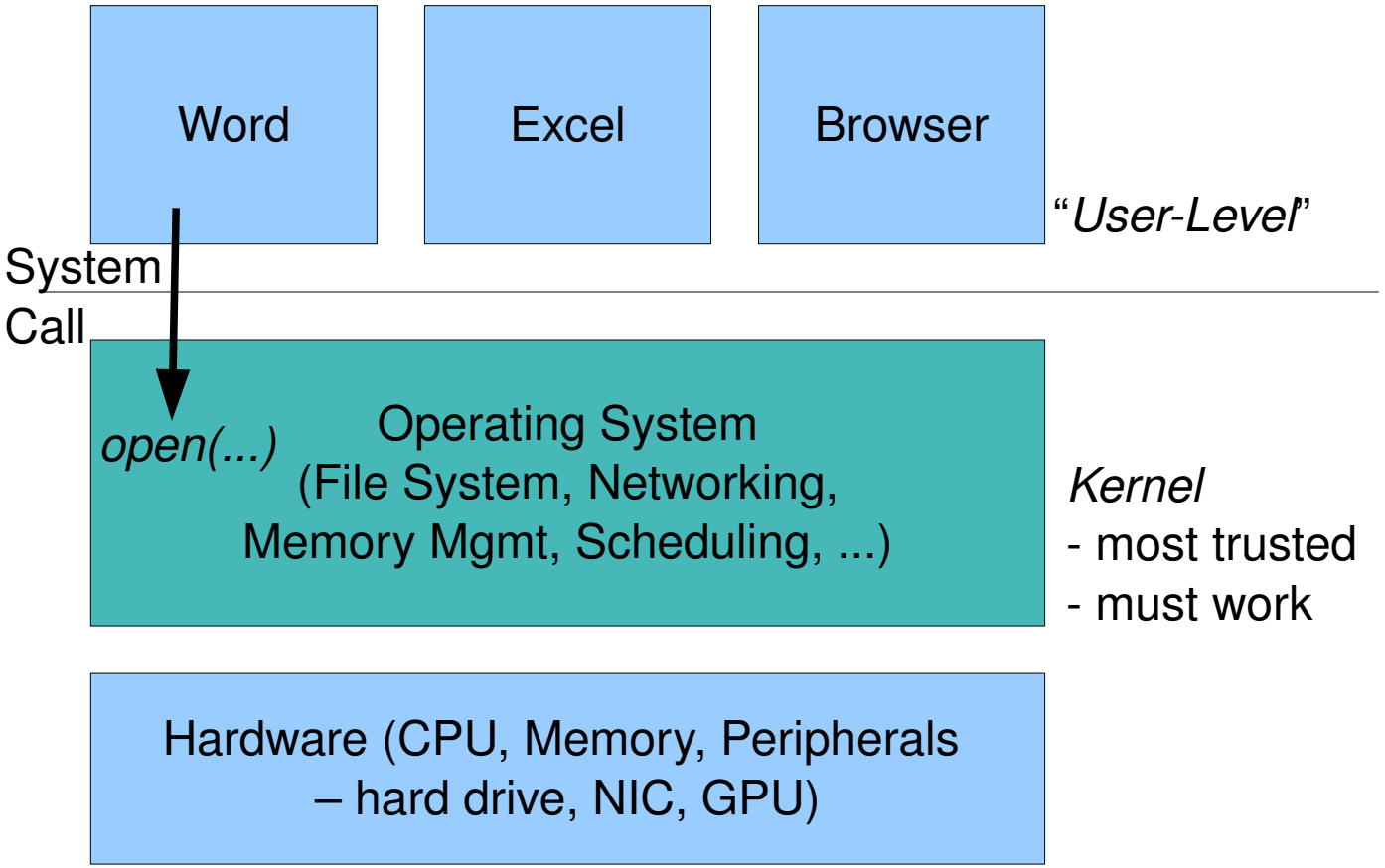
Welcome to Special Topics in OS

System Structure

- *System Structure* – How different parts of software
  1) Are separated from each other (*Why?)*
  2) Communicate
- How does a system use
  - dual mode
  - v*irtual address spaces*

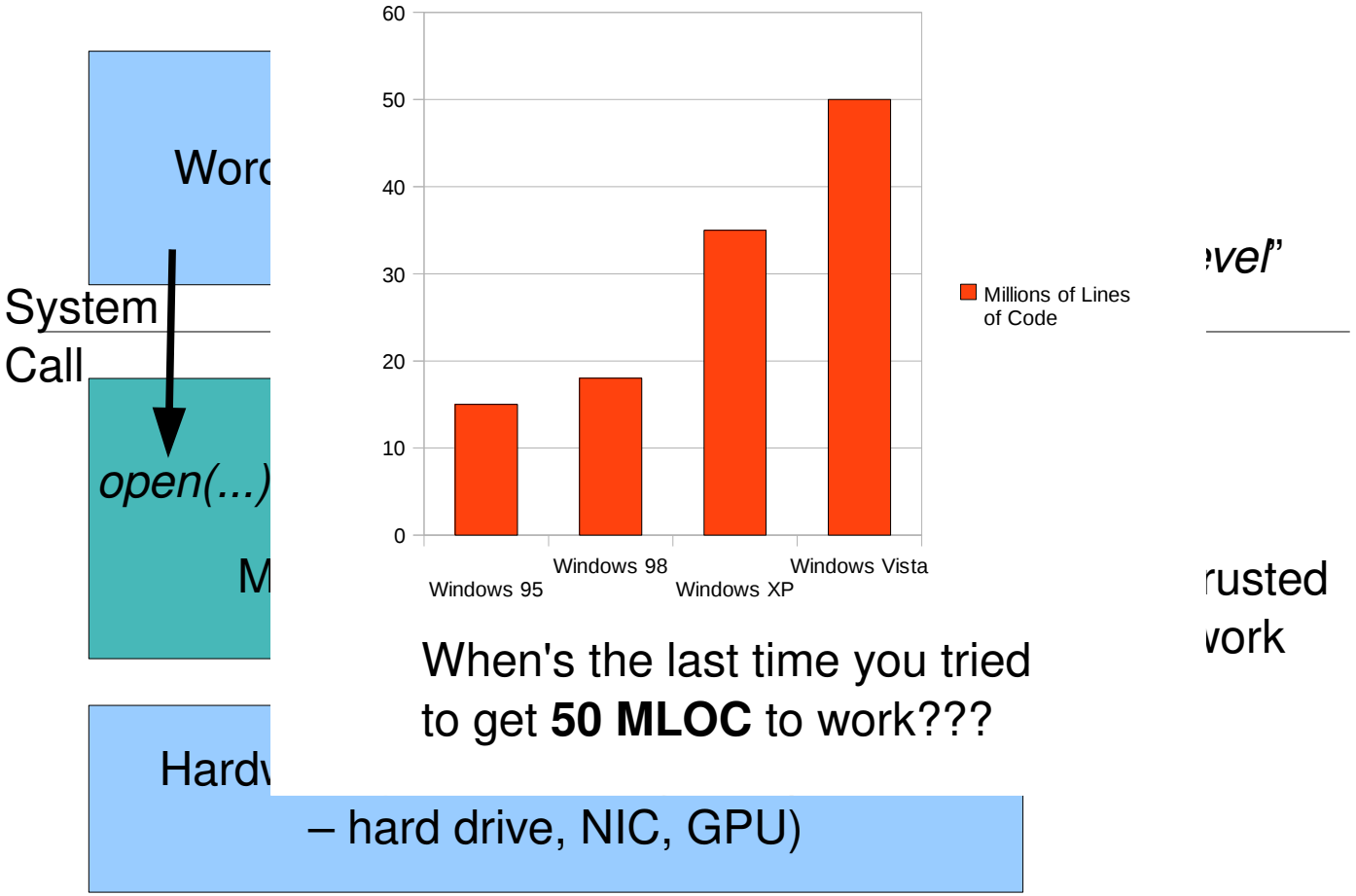
Monolithic System Structure

- Includes Unix/Windows/OSX



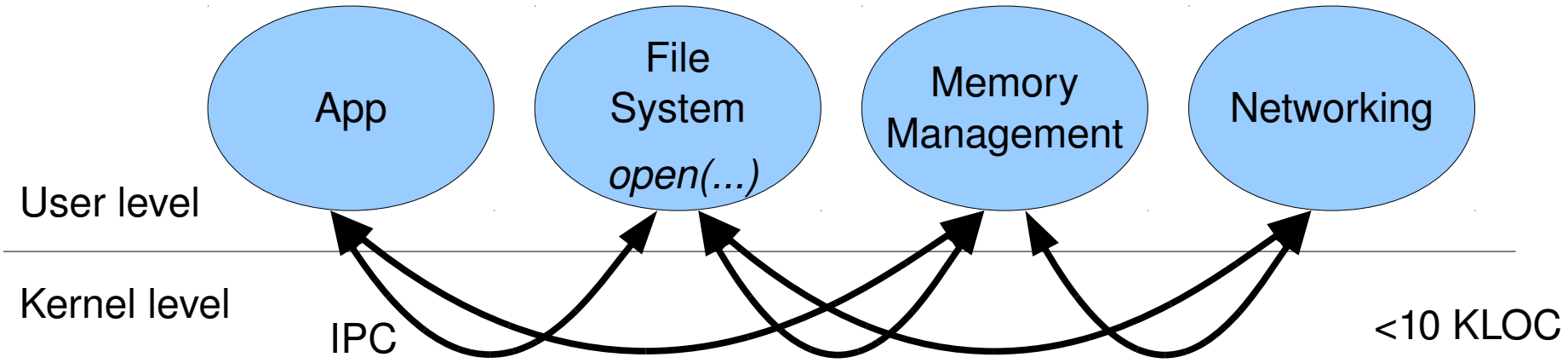
Monolithic System Structure

- Includes Unix/Windows/OSX



When's the last time you tried to get **50 MLOC** to work???

Microkernel System Structure



- Moves functionality from the kernel to "*user*" space
- Communication takes place between user *servers* using inter-process communication (IPC)
- Benefits:
  - Easier to add functionality
  - More reliable (less code is running in kernel mode)
  - More secure
- Detriments: performance! (why?)