

# CSCI 3411: Operating Systems

# Administrative Fun

- Professor Gabriel Parmer (aka. *Gabe*)  
[gparmer@gwu.edu](mailto:gparmer@gwu.edu), Phillips 720E
- Class: Tompkins 201, Tues/Thurs 12:45-2:00
- Lab (required attendance): Tompkins 211, Thurs  
2:10-3:45 or 6:10-8:00
  - *Please move to the earlier lab!!!*

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

# Administrative Fun II

- Format of Class
  - Lecture
    - Concepts
    - Written HW/tests
  - Lab – TA: James Tyler ([jrt@gwu.edu](mailto:jrt@gwu.edu)) – 313 Staughton
    - Implementation: C, Linux
    - Programming assignments & final project
- Book(s)
- Grading – hw, exams, **participation**
- Academic Honesty

# Administrative Fun III

- Office hours options?
  - TR – 10-12
  - T – 2-4
  - R – 4-6
  - W – 1-3, 4-5:30
  
  - T – 3-5
  - W – 3-5
  - R – 4-6
  - F – 2-4

# “High-level”



Cars

Google

[Advanced search](#)  
[Language tools](#)

Google Search

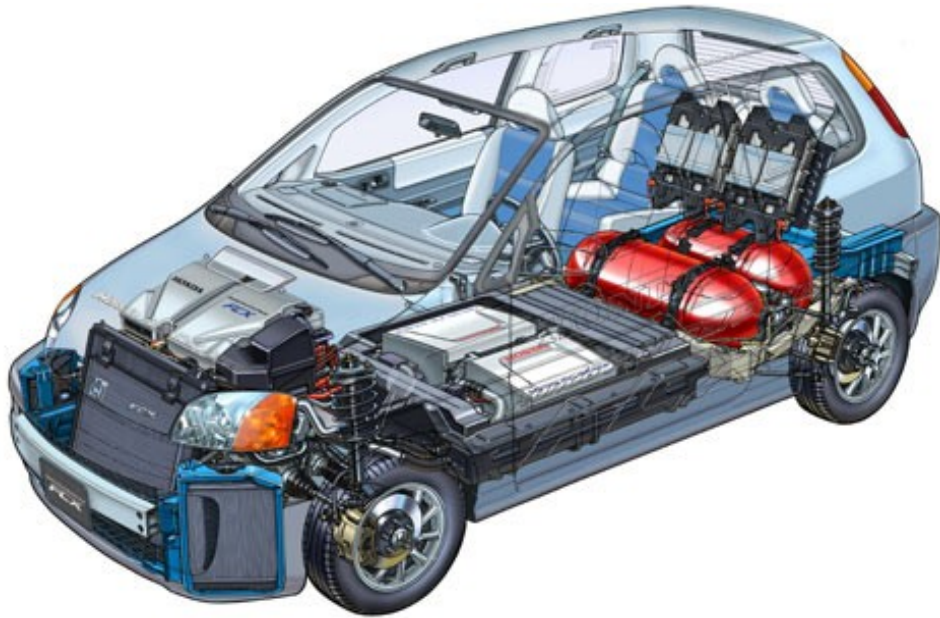
I'm Feeling Lucky

[Advertising Programs](#) [Business Solutions](#) [About Google](#)

© 2011 - [Privacy](#)

Computers

# ...details...



Cars

```
/*
 * 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);

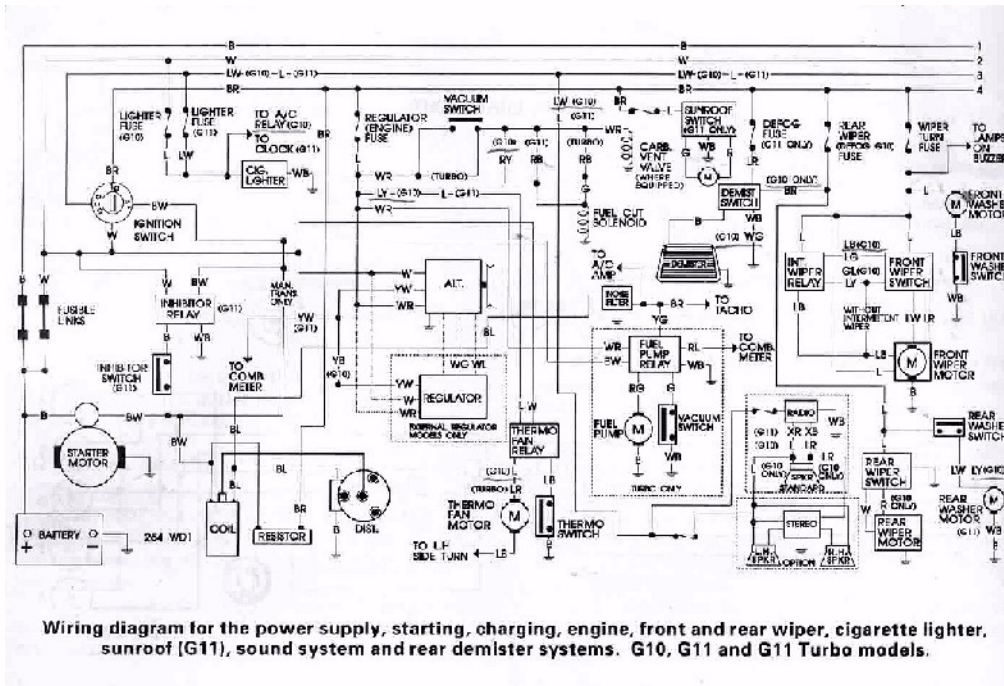
if (unlikely(flags)) {
    thd = switch_thread_slowpath(curr, flags, curr_spd, rthd_id, da, &ret_code,
                                &curr_sched_flags, &thd_sched_flags);
    /* If we should return immediately back to this
     * thread, and its registers have been changed,
     * return without setting the return value */
    if (ret_code == CDS_SCHED_RET_SUCCESS && thd == curr) goto ret;
    if (thd == curr) goto_err(ret_err, "sloooow\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);
```

Computers

# ...“low-level”



Cars

```

000000004006b0 <_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
    
```

Computers

# What is an Operating System!?





# What is an OS: Where is it?

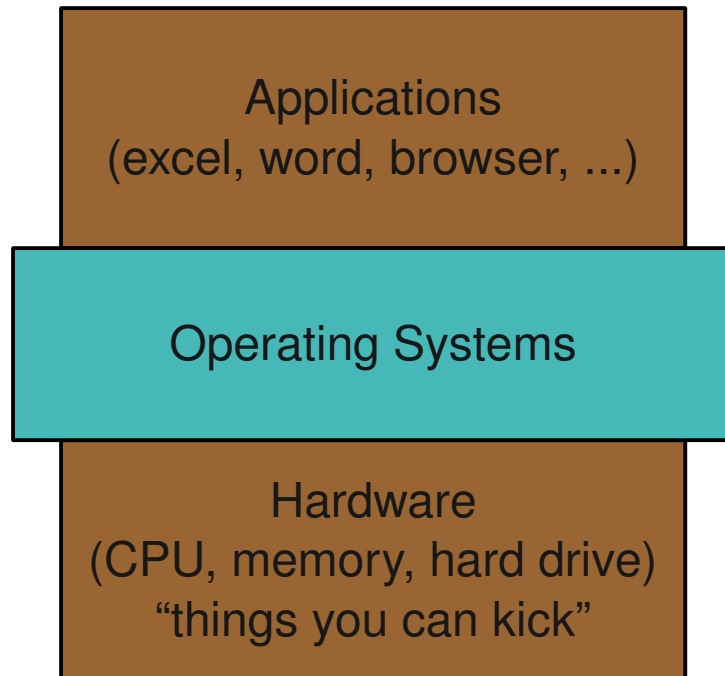
Applications  
(excel, word, browser, ...)

Operating Systems

Hardware  
(CPU, memory, hard drive)  
“things you can kick”

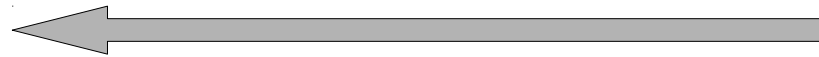


# What is an OS: Where is it?



COURTESY: KFC

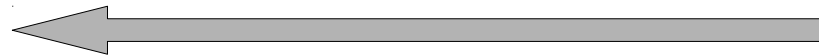
# What is an OS: Analogy



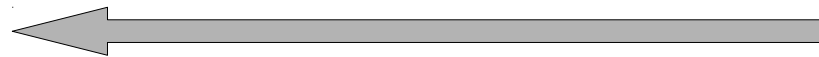
You!



Customer<sub>1</sub>



Customer<sub>2</sub>



Customer<sub>n</sub>

# What is an OS: Analogy



You!

Customer<sub>1</sub>

Customer<sub>2</sub>

Customer<sub>n</sub>



# What is an OS: Analogy

Hardware



Operating System



Applications

You!

Customer<sub>1</sub>

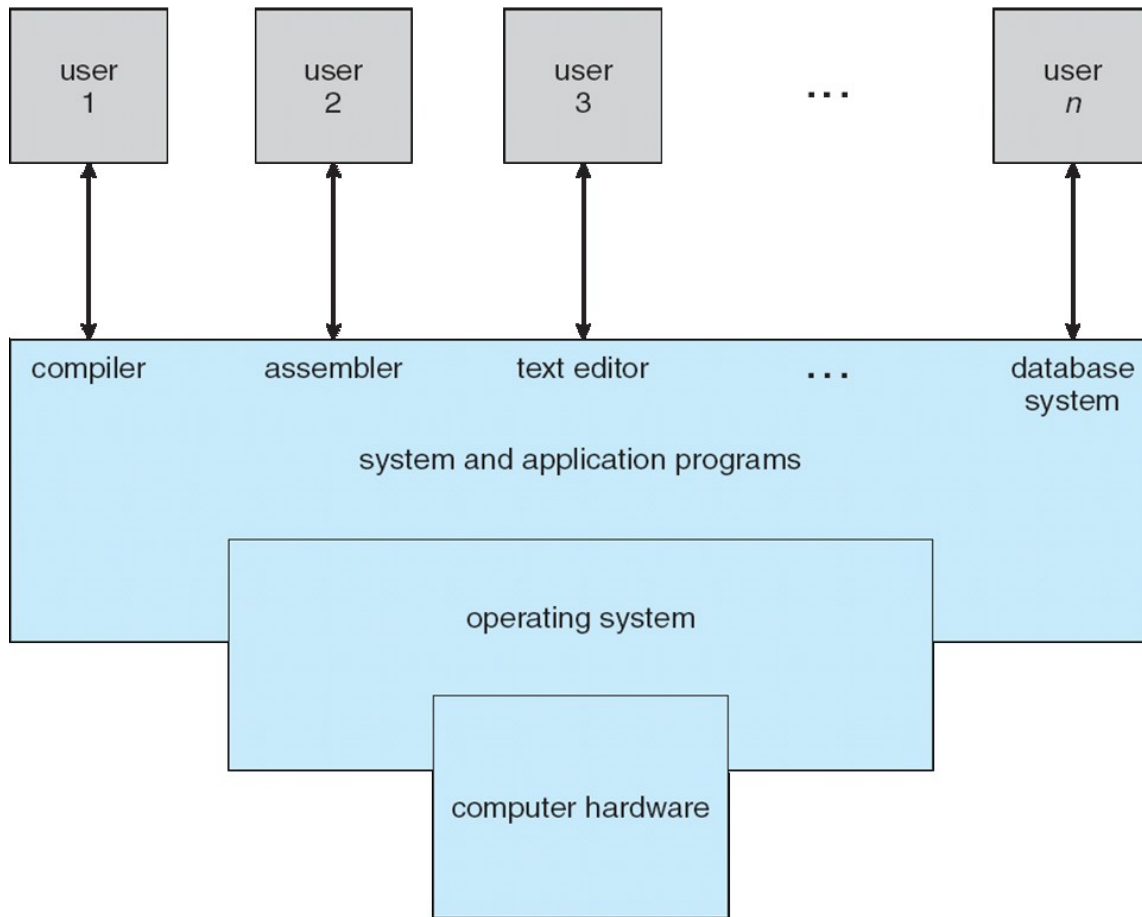
Customer<sub>2</sub>

Customer<sub>n</sub>

# 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
  - Each application can pretend like it is using the entire computer!
- Allow users to translate intentions into actions

# OS as Abstraction: System Layers



AN x64 PROCESSOR IS SCREAMING ALONG AT BILLIONS OF CYCLES PER SECOND TO RUN THE XNU KERNEL, WHICH IS FRANTICALLY WORKING THROUGH ALL THE POSIX-SPECIFIED ABSTRACTION TO CREATE THE DARWIN SYSTEM UNDERLYING OS X, WHICH IN TURN IS STRAINING ITSELF TO RUN FIREFOX AND ITS GECKO RENDERER, WHICH CREATES A FLASH OBJECT WHICH RENDERS DOZENS OF VIDEO FRAMES EVERY SECOND

BECAUSE I WANTED TO SEE A CAT JUMP INTO A BOX AND FALL OVER.

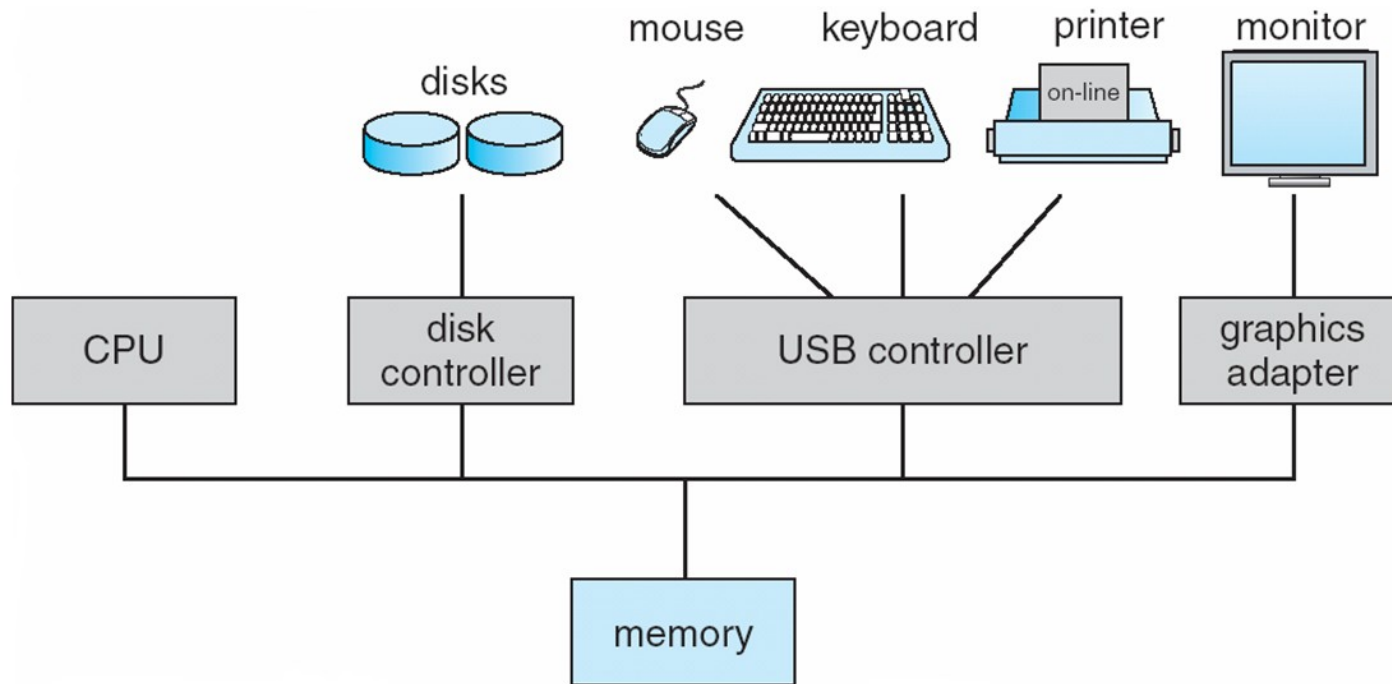


I AM A GOD.



# Computers as Distributed Systems

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



# OS as Hardware Manager

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

# History, or How did we get where we are now?

- Bare metal
  - Life cycle:
    - Boot up
    - Run a single application
    - Output result
    - Power down
- *OS support for these systems???*

# History: Batch Systems

- Goal: Maximize amount work done for multiple users
- Applications run one after the other
  - One application at a time!
- Application uses all computer resources
- *OS support for batch systems???*

# History: Batch Multiprogramming

- Multiple applications in memory
- When one waits for I/O, another executes
  - Better utilization of CPU
- *OS support for these systems???*

# History: Timesharing Systems

- Interactive use of computers
  - Responsiveness matters
  - Expect system to respond to keyboard input immediately
- Several users/applications can share computer
  - Share CPU, Disk, Memory...
- *OS support???*

# Batch vs. Timesharing: Fight!

- Which is more efficient? Which gets more work done?

# Batch vs. Timesharing: Fight!

- Which is more efficient? Which gets more work done?
  - Computer work: instructions processed per second
  - Human work: perform operations user requires



# History: PCs, Servers, Mobile

- Iterations on timesharing systems
- PCs
  - Less emphasis initially on protection (argh!)
- Servers
  - Throughput oriented
- Mobile
  - Power consumption

# iPhone vs. Android

- Which paradigm does each fall into?

# iPhone vs. Android

- Which paradigm does each fall into?
- iPhone: single user application running at any point in time
  - Back to the 70s
- Android: multiple applications concurrently execute
  - What happens when memory runs out?

# Fundamental OS Concepts

- Abstraction
- Resource management (CPU, RAM, devices)
- Concurrency
- Parallelism
- Protection/Security
- Performance
  - Kernel doesn't **do** useful work, enables it

# Course Objectives

- Explore core ideas in Operating Systems in two ways:
  - 1) understanding the concepts behind resource management, abstraction, and hardware interface
  - 2) practical coding experience with a real OS to understand the subtleties and challenges of systems

# Why should you care about OSes!?

- Glad you asked...
- Fundamentally: Understand what's going on under the hood
  - "In theory, there is no difference between theory and practice. But, in practice, there is." - Jan L. A. van de Snepscheut/Yogi Berra
- The world runs on systems
  - Microsoft, VMWare (Operating systems, virtual machines)
  - Google, Yahoo (distributed systems)
  - Boeing, NASA, BMW (embedded/distributed systems)
  - Financial firms (have to spend stimulus money on something!)
- The world is concurrent!
- Industry feedback