CMSC421: Principles of Operating Systems

Nilanjan Banerjee

Assistant Professor, University of Maryland
Baltimore County
nilanb@umbc.edu
http://www.csee.umbc.edu/~nilanb/teaching/421/

Administrivia

- Course webpage: http://www.csee.umbc.edu/~nilanb/teaching/421
  - Instructor: Nilanjan Banerjee
  - Class time: MW 5:30 – 6:45 pm, IT 233
  - Email: nilanb@umbc.edu, room # 362
  - http://www.csee.umbc.edu/~nilanb/
  - Office hours: MW: 2:00 ---3:00 pm, room 362

- Teaching Assistant for this course
  - Lawrence Sebald
    - Email: lsebald1@umbc.edu
    - Office hours: Mon(11AM-1PM), Tu(9:30AM-11:30AM), ITE240
  - Milind Patil
    - Email: milind1@umbc.edu
    - Office hours: Tu(11AM-1:00PM), Th(1PM-3PM), ITE240

Teaching Style

- Highly interactive
  - Incentivize questions and discussions in class ☺
- Live coding in class
  - You are welcome to bring your laptops with toolkits installed
  - Writing userland code in class (mostly) and some kernel code
- Kernel code surfing
  - Will be using http://lxr.linux.no/linux/
  - Labeled, annotated linux source code

Focus is on implementation and hands-on experience

Course webpage

http://www.csee.umbc.edu/~nilanb/teaching/421/

will be moved to a central place

Lets take a look at the webpage

Blackboard page is a replica

All homeworks and projects would be submitted using blackboard

What is the grading distribution?

- Homeworks (10%, 3-4+1)
  - Theoretical concepts
  - Some small programming components
- Projects (40%)
  - 3 project
  - Involve writing linux kernel code
- Midterm (20%)
- Final (25%)
- End of class discussions (5%)

What is end of class discussions!

- Puzzles from Microsoft, Google, Intel, etc. interviews
  - Mathematical puzzles.
  - Algorithmic puzzles

- Programming puzzles
  - Bit hacks!
  - Weird stuff on C

- How would be grading done
  - 0 or 1 --- 0 (if you do not attend class or do not attempt the problem at all) , 1 (if you make a plausible attempt at the problem– need not be best solution)

Late Policy and Plagiarism

- Very strict policy on plagiarism
  - You can discuss questions but you are *NOT* supposed to see or replicate each others answers or source code

- There is no late policy for homeworks or projects!
  - extra-ordinary circumstances you will have to get permission from the instructor
  - We might extend deadlines if need be.

Lets get started with some demonstrations

What does a OS really consist of?

distribution

operating system

kernel

flavor of an OS

e.g. Redhat
Ubuntu

user-space

bunch of stuff
-- User interface
-- system management tools
-- developer tools
-- compilers/interpreters
-- assemblers/linkers

kernel-space

-- kernel
-- core OS

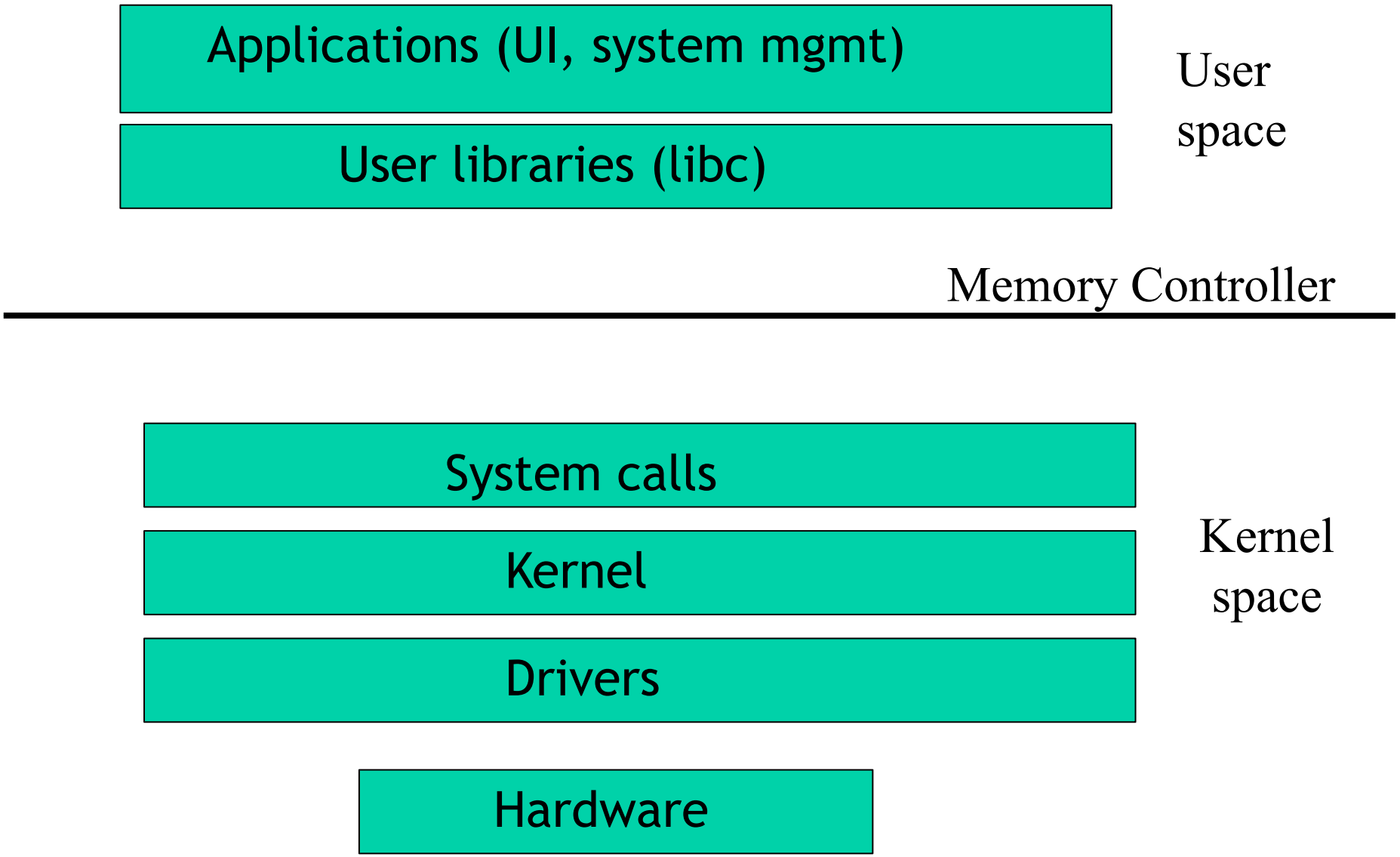
resource mgmt
-- process mgmt
-- CPU scheduling
-- memory mgmt
-- storage mgmt
-- I/O subsystem

Where do drivers reside?

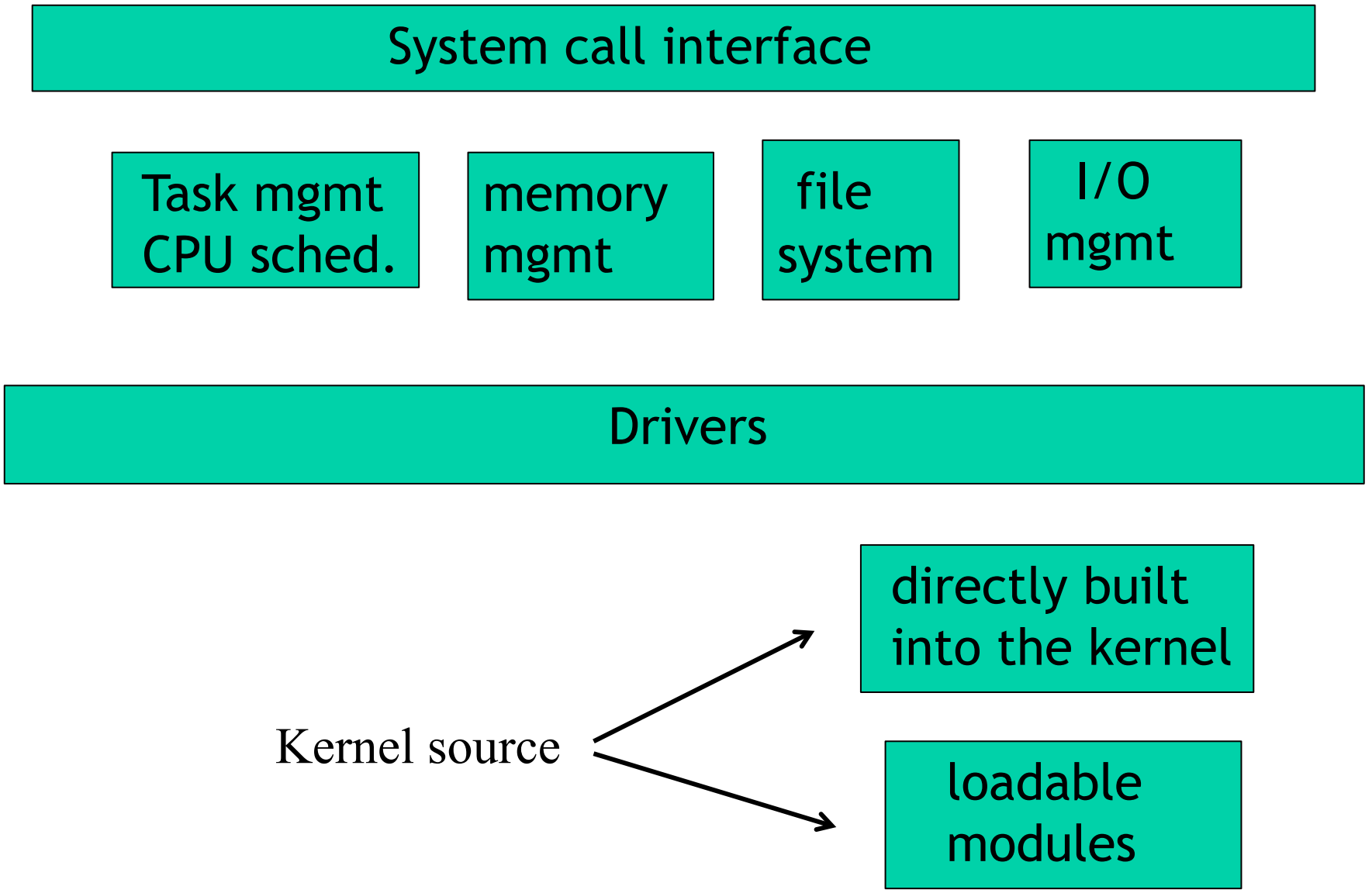
What does an OS provide us?

- Hardware abstractions
  - Applications do not have to deal with nits and grits of hardware
  - Applications cannot access hardware directly
    - Use OS to access disk drives, Network cards, CPU, memory (some parts)
- Multi-tasking/multiprocessing
  - Resource allocations
    - One CPU, one disk, one memory
    - Allocate resources to different applications
  - Protection
    - Sandbox applications & applications and the OS
    - Usually done through the Memory controller (virtual memory) (First microprocessor to support MC?)

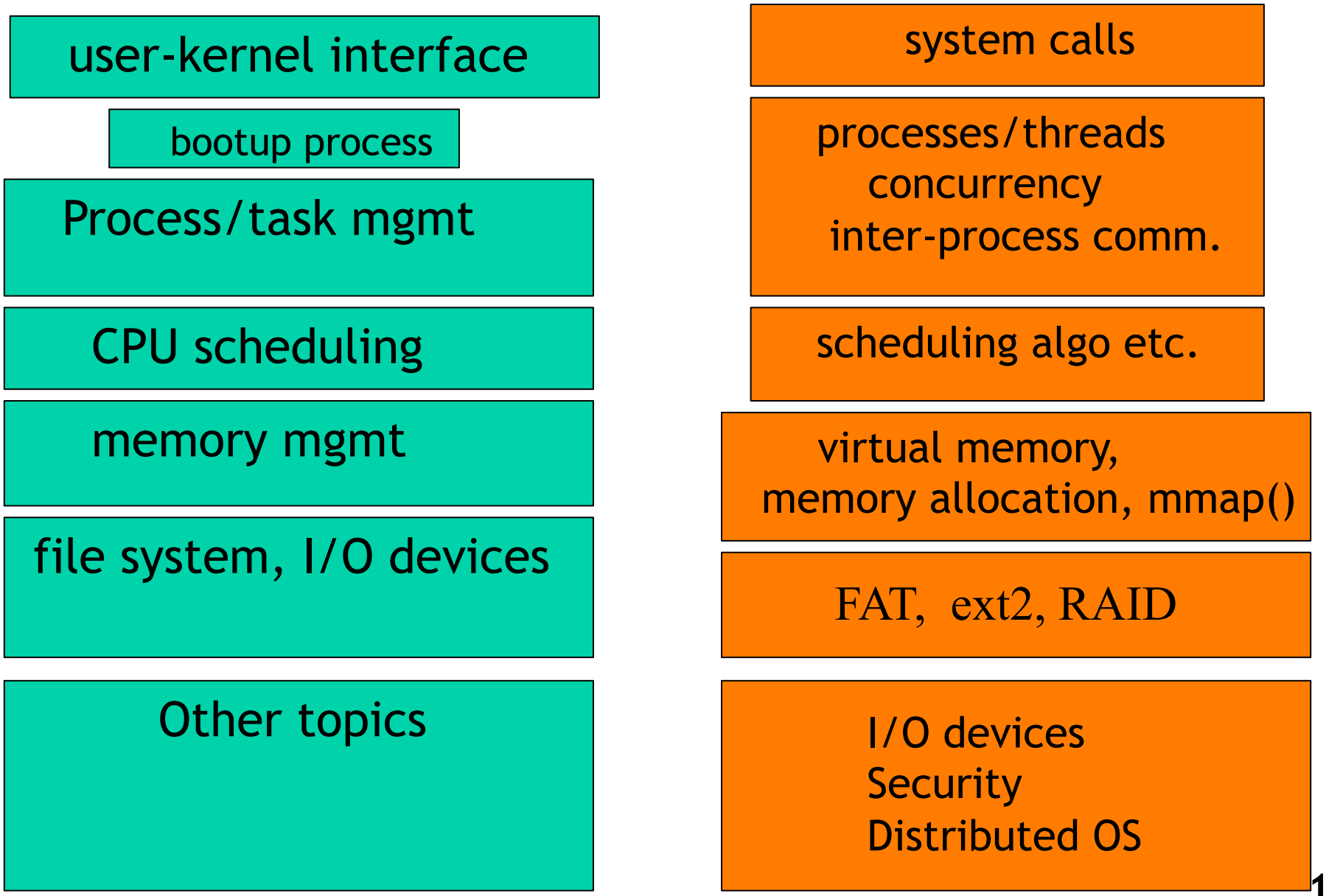
What does an Operating System look like?



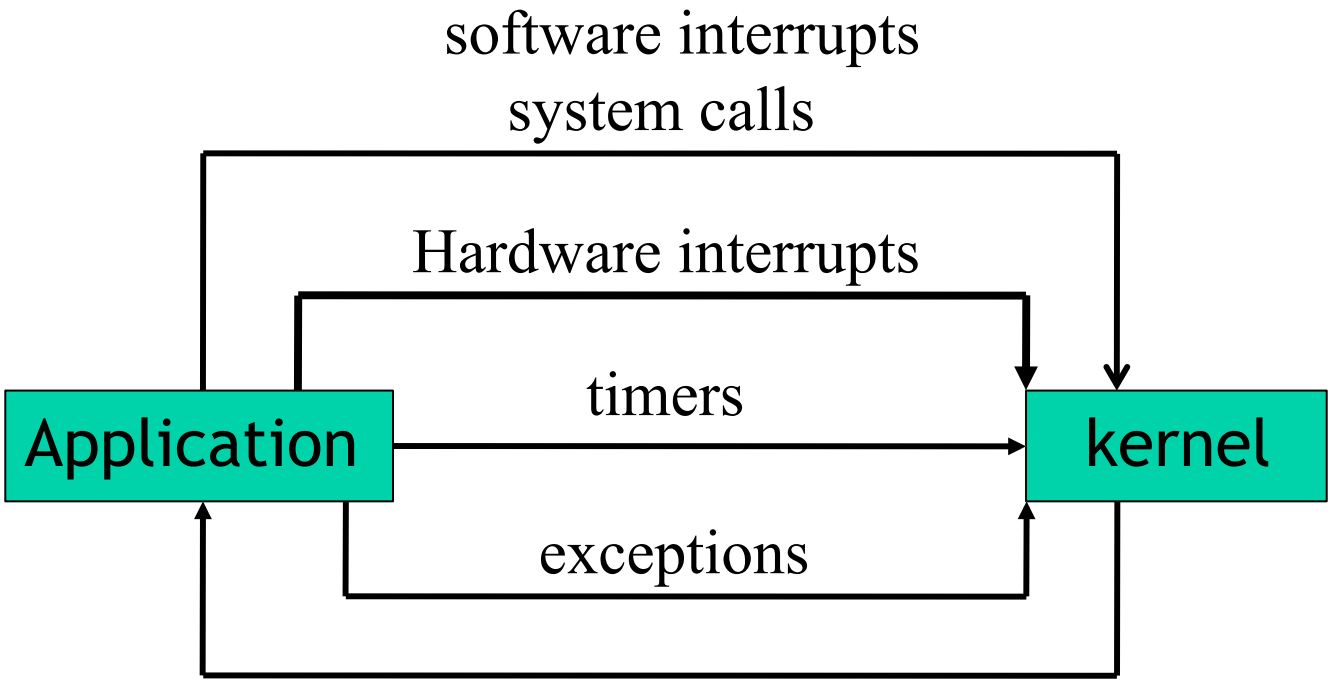
Magnify the kernel



Topics to be covered



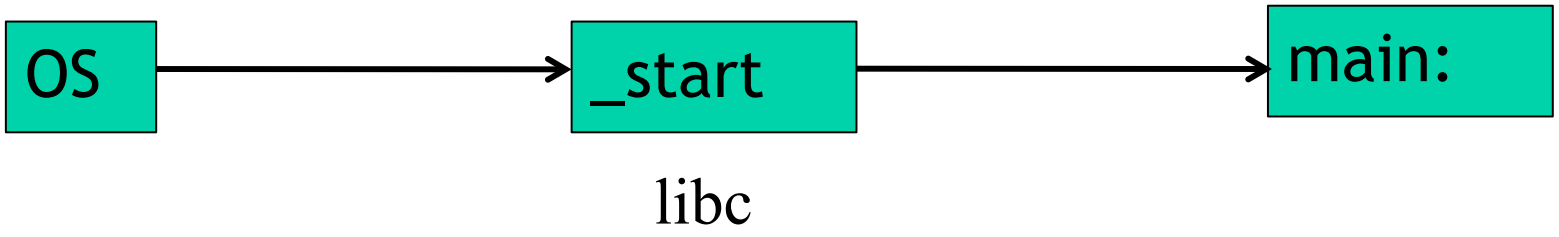
Kernel-userspace interaction

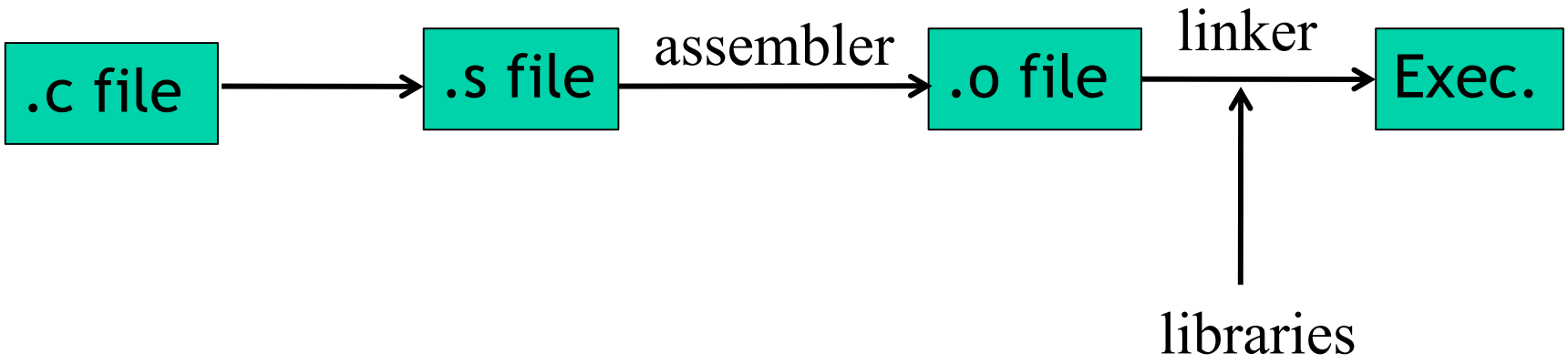


Timers/return from sys call

A closer look at system calls

Lets take an example



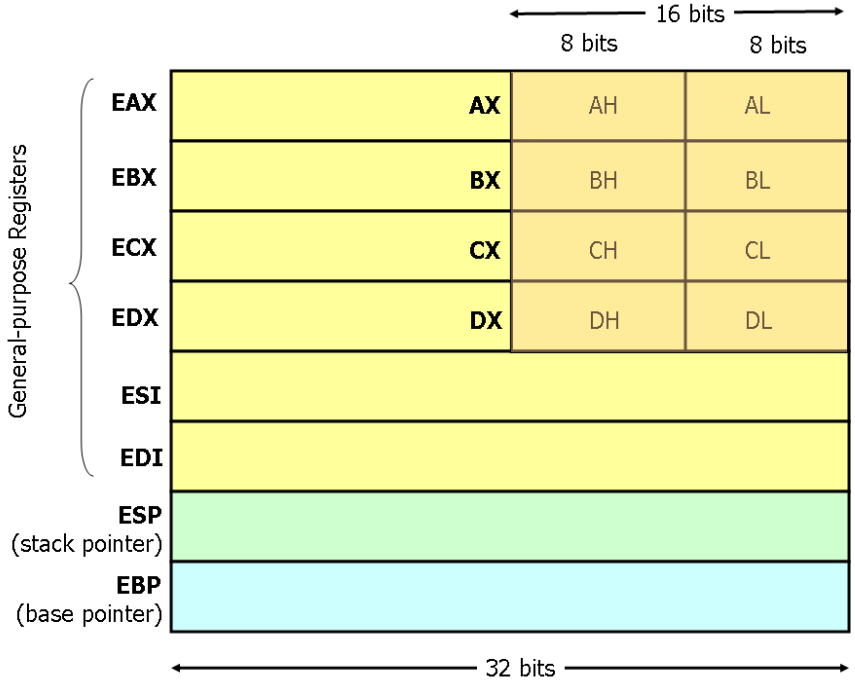


Let take a look at some x86 assembly

mov $1, %eax

mov $25, %ebx

int $0x80 ← **Execute interrupt # 128**
In the interrupt vector table



↓

Software interrupt

↓

Jump to an address in the kernel
where the syscall table is stored
And execute syscall # stored in %eax
args for syscall in registers
[ebx, ecx, edx, esi, edi]

Next class

- System calls in detail
  - What really happens during a system call
- Kernel Code surfing
  - Intro to the linux kernel
  - How the context switch happens to the kernel
- Writing System calls in the kernel
  - Write a simple system call
  - Things to know about system calls in the kernel
    - Concurrency issues
    - Copy data to and from the kernel

An in-class discussion (pointers in C)