CS3841 Design of Operating Systems
OS Structure

- Objectives
  - Construct source code which performs a system call.
  - Explain the concept of a trap.
  - List some examples of System calls.
  - Draw a diagram showing the structure of a Modern *NIX System
  - Explain the concept of a loadable module in Linux
  - Draw a picture showing the relationship between Linux kernel components.

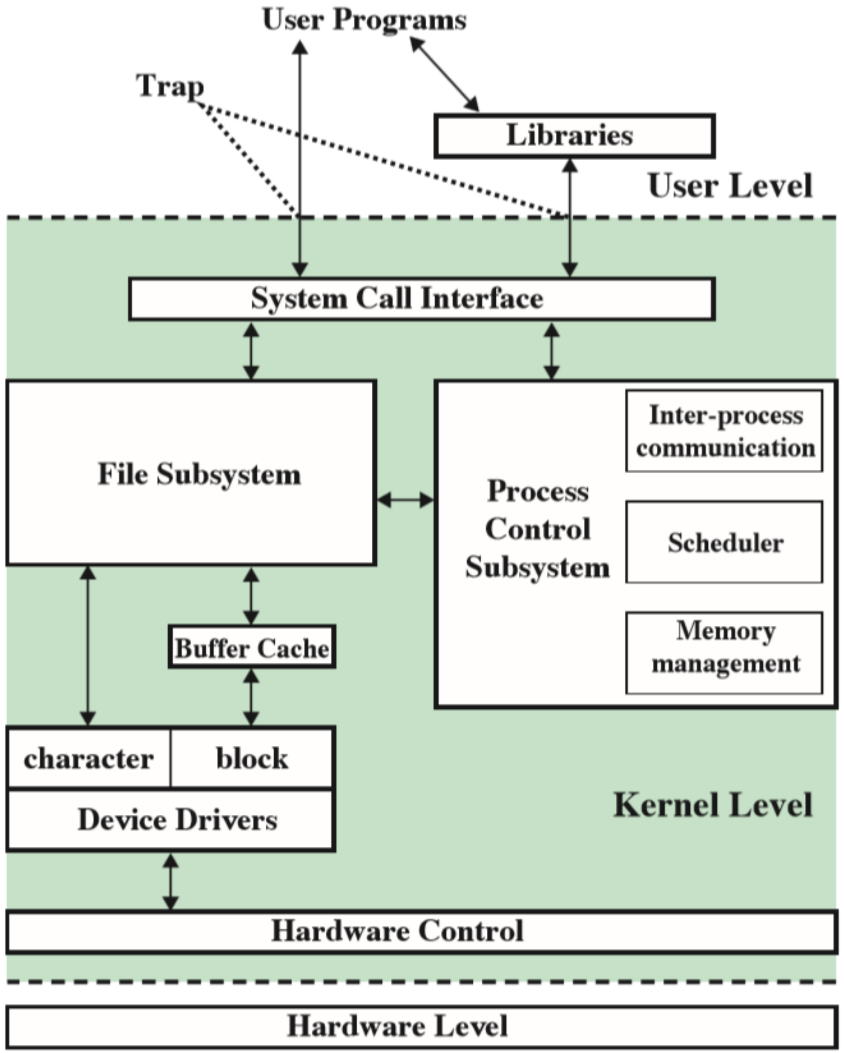
Operating System vs Kernel

- Operating System
  - A piece of software that provides services to applications
- Kernel
  - A piece of software that “bridges” hardware and software
  - Figurative sense of "core or central part of anything“ (https://www.etymonline.com/word/kernel)
- Questions
  - Is a kernel an operating system?
  - Is there more to an operating system than just the kernel?
  - Can an operating system have more than one kernel?
  - Does the kernel run on its own?
  - How do we create a kernel?

Dual Mode Operation

- Modern Operating Systems use at least two modes of operation
  - User mode
    - A restricted mode of operation which only allows certain instructions to be executed by the program
    - Prevents errant processes from crashing the system
  - Kernel Mode
    - Also referred to as supervisor mode, system mode, or privileged mode
    - Allows the system full access to the microprocessor
    - Intended to be used only by the operating system

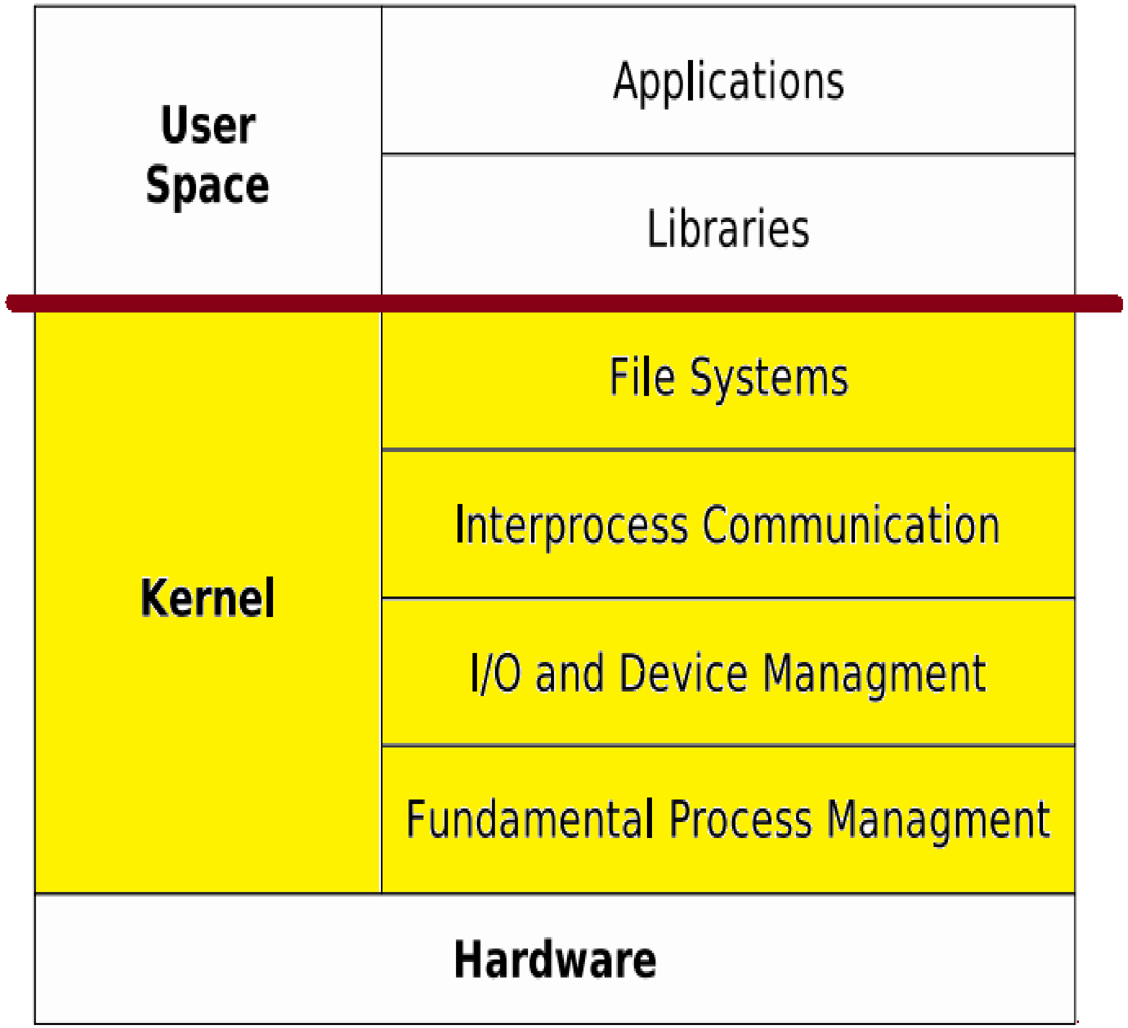
Traditional *NIX – Monolithic Kernel



- Few components
  - User programs
  - Kernel
  - Hardware
- Advantages
  - Single point of control – All services in single address space
- Disadvantages
  - Single point of failure
  - Updates require reload of system

Operating Systems: Internals and Design Principles, 9th Edition
William Stallings

Microkernel Structure



User Space

Applications

Libraries

Kernel

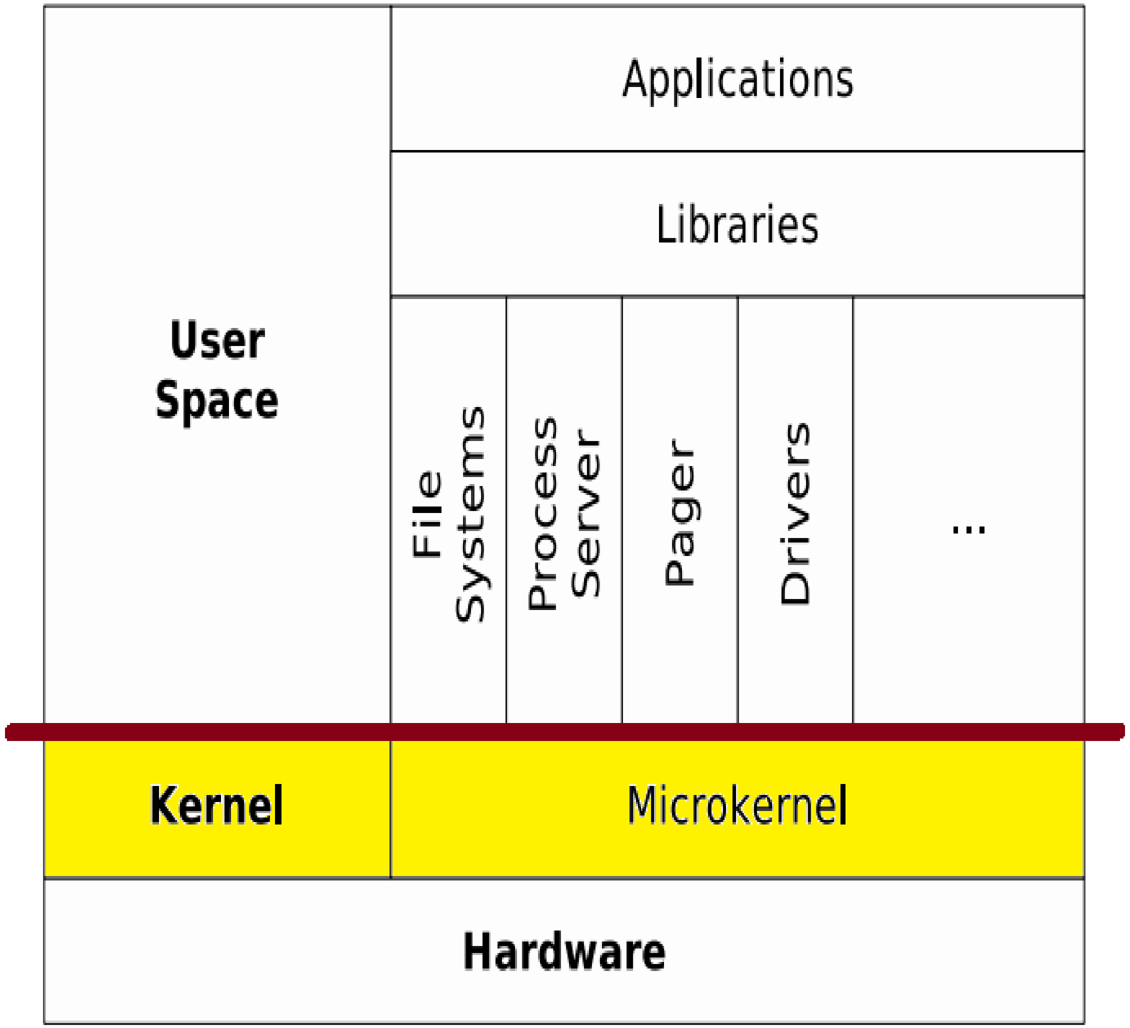
File Systems

Interprocess Communication

I/O and Device Managment

Fundamental Process Managment

Hardware



User Space

Applications

Libraries

File Systems

Process Server

Pager

Drivers

...

Kernel

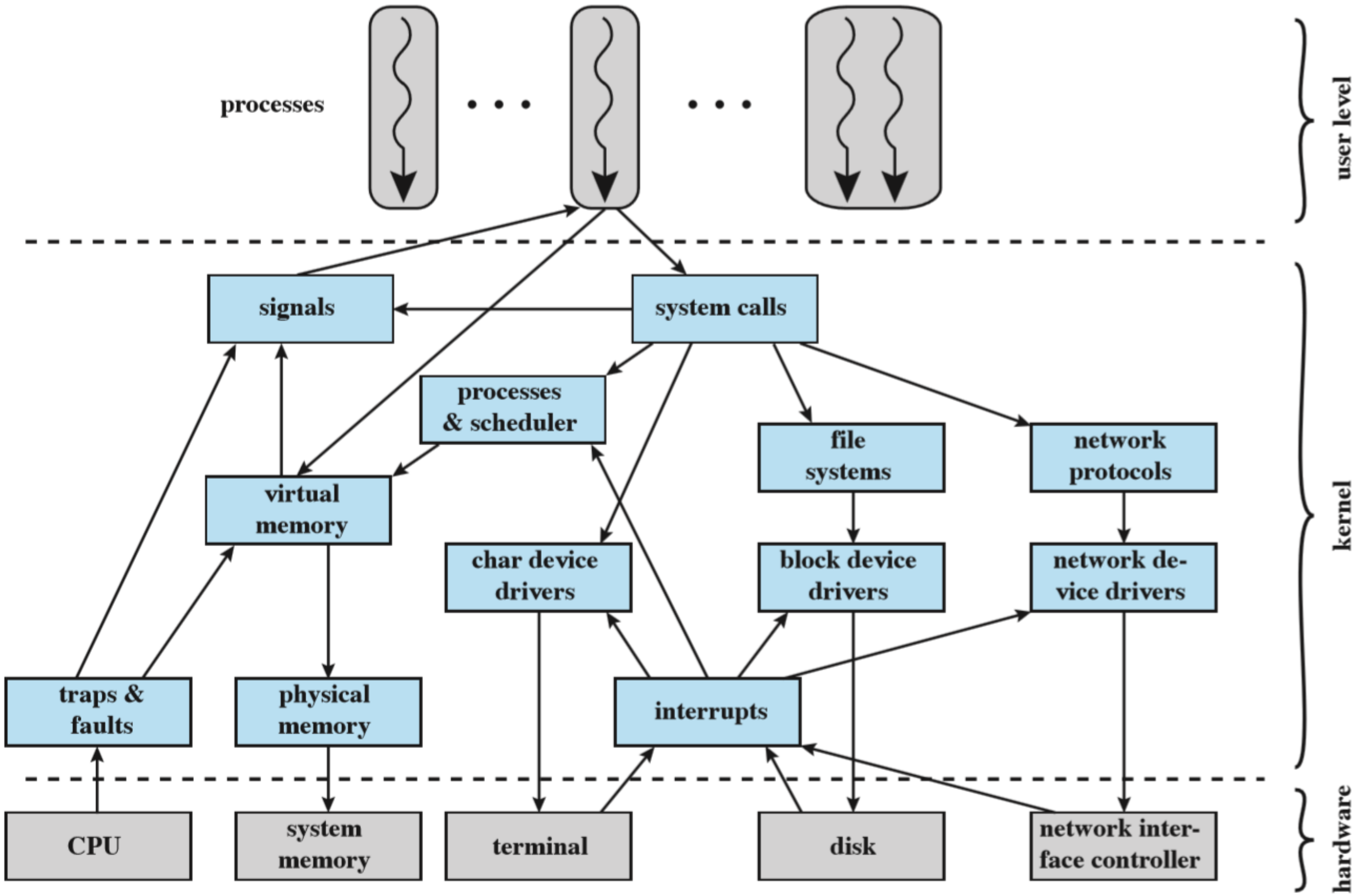
Microkernel

Hardware

Microkernels

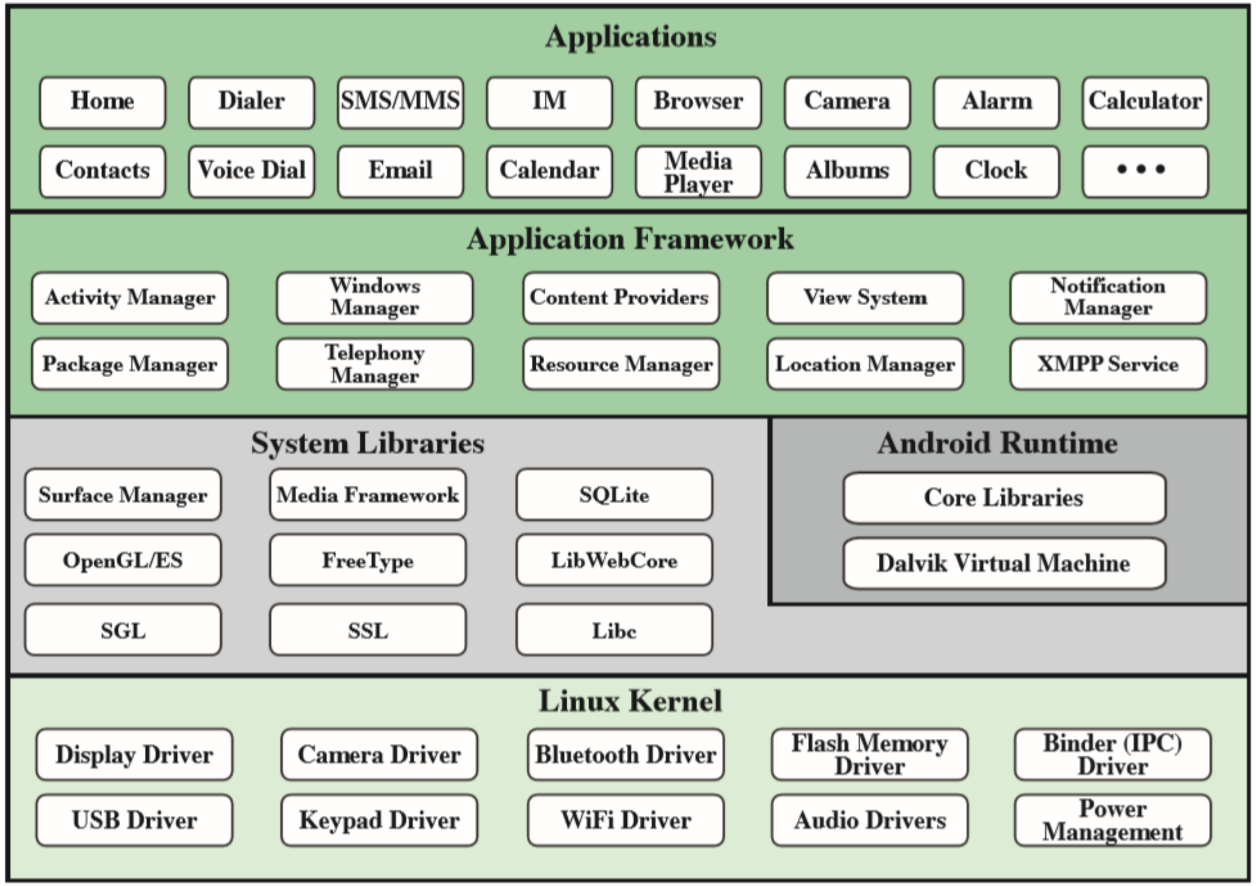
- Remove all but “essential components” from the kernel
- Bulk of responsibilities is in user space
- Communication through message passing
- Advantages
  - Smaller kernel
  - More robust - User space components can be updates/restarted easily
- Disadvantages
  - Message passing overhead
  - Additional system calls needed

Linux Kernel



Operating Systems: Internals and Design Principles, 9th Edition William Stallings

Android



Applications

Home, Dialer, SMS/MMS, IM, Browser, Camera, Alarm, Calculator

Contacts, Voice Dial, Email, Calendar, Media Player, Albums, Clock, • • •

Application Framework

Activity Manager, Windows Manager, Content Providers, View System, Notification Manager

Package Manager, Telephony Manager, Resource Manager, Location Manager, XMPP Service

System Libraries

Surface Manager, Media Framework, SQLite

OpenGL/ES, FreeType, LibWebCore

SGL, SSL, Libc

Android Runtime

Core Libraries

Dalvik Virtual Machine

Linux Kernel

Display Driver, Camera Driver, Bluetooth Driver, Flash Memory Driver, Binder (IPC) Driver

USB Driver, Keypad Driver, WiFi Driver, Audio Drivers, Power Management

Implementation:

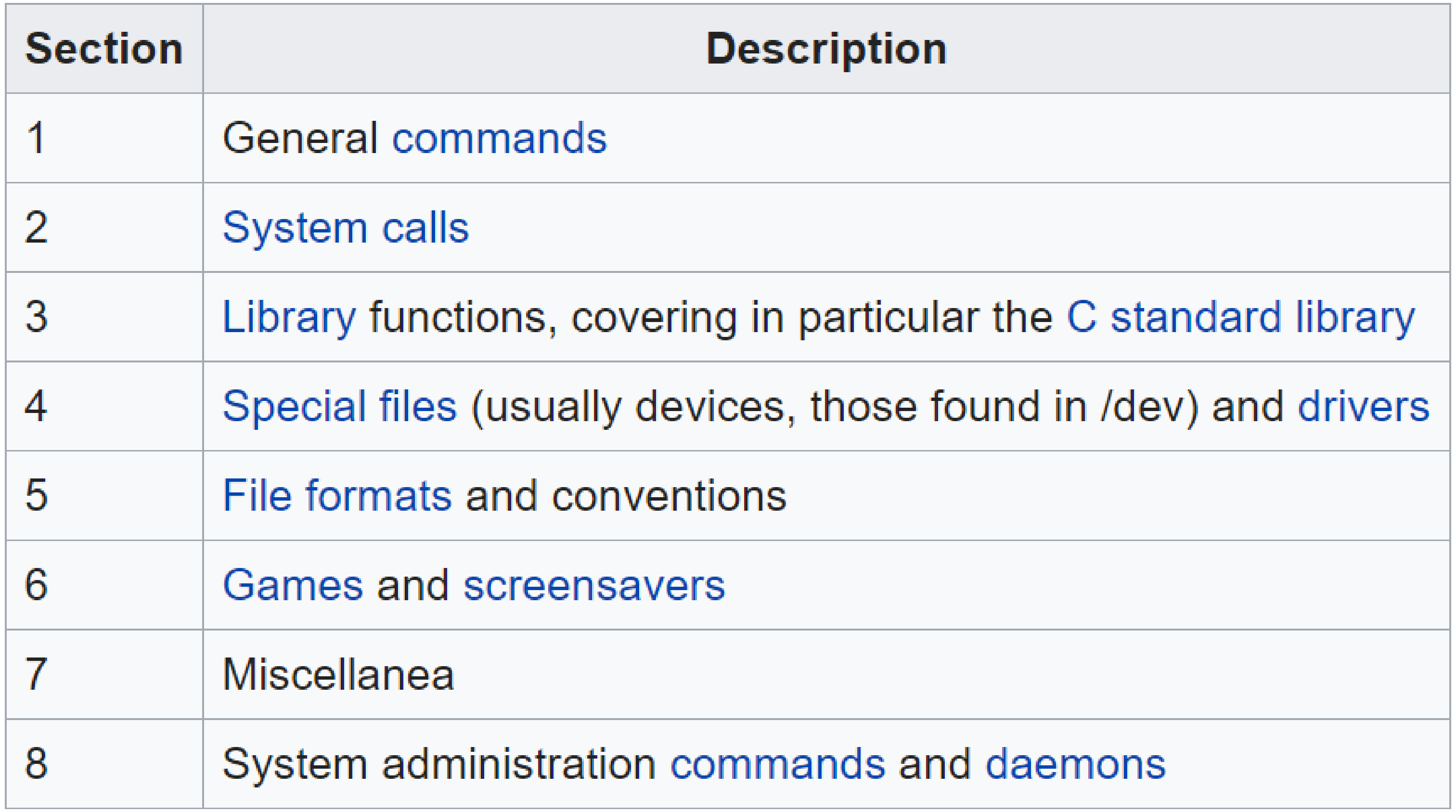
🟩 Applications, Application Framework: Java

⬜ ⬛ System Libraries, Android Runtime: C and C++

⬜ Linux Kernel: C

Getting help in Linux

- man
  - Manual pages
- apropos
  - Man page search



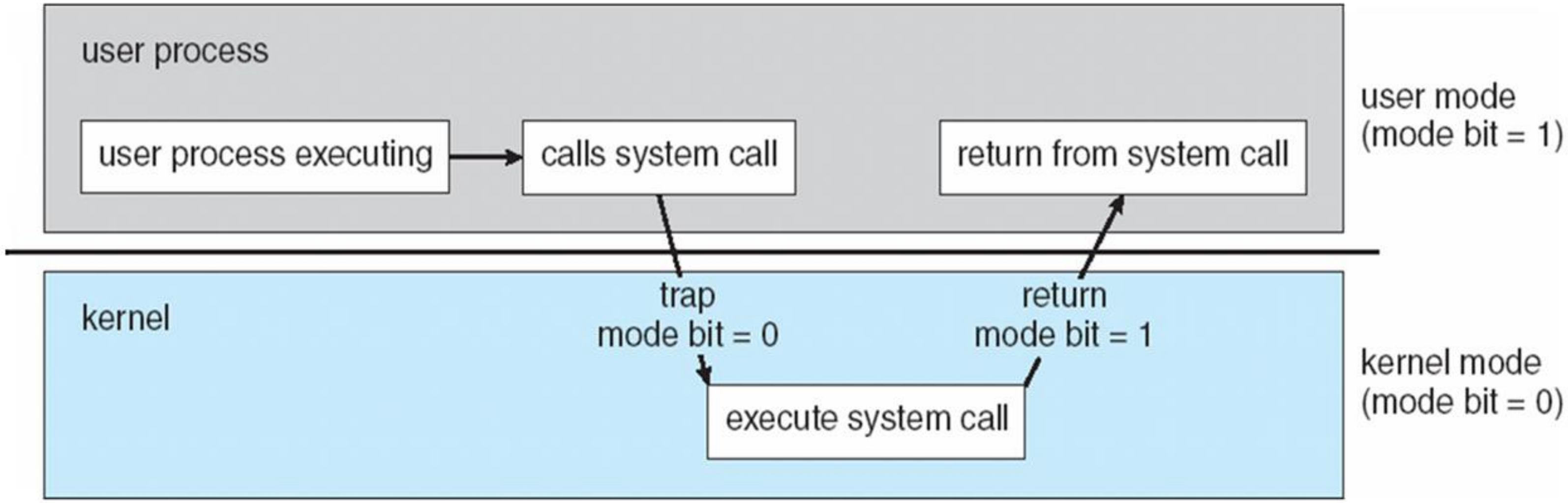
Section	Description
1	General commands
2	System calls
3	Library functions, covering in particular the C standard library
4	Special files (usually devices, those found in /dev) and drivers
5	File formats and conventions
6	Games and screensavers
7	Miscellanea
8	System administration commands and daemons

System Calls

- System Calls provide a set of “functions” for applications to use operating system services
  - – OS specific
  - – Portable Operating System Interface (POSIX)
    - C or C++ library interface
- Typically involve some “trap” to the operating system

System Calls

- Trap, System Call, Supervisor Call: user mode -> kernel mode
  - – Transfers control from user program to kernel function
  - – Sets mode from user to kernel



Abraham Silberschatz, Peter B. Galvin, and Greg Gagne. 2013. Operating System Concepts Essentials (2nd ed.). Wiley Publishing.

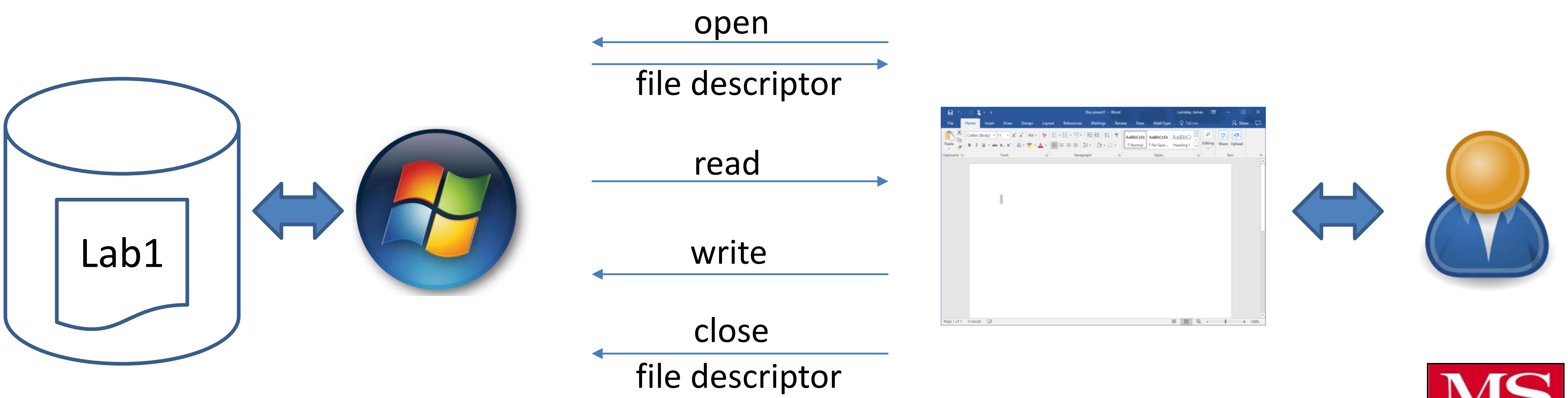
Why do we need system calls?

- Isolation and protection
- Kernel is running in privileged mode
  - User process is not
- Can processes share anything?
  - We will see this later as a method of inter-process communication
- Can processes share information with the kernel?
- In addition to sharing information, we also want kernel to take actions, perhaps immediately

System Calls Example - File Input/Output

– What’s a file? Abstract representation of data on “disk”

– How do we access a file? open, read/write, close



Lab1

open

file descriptor

read

write

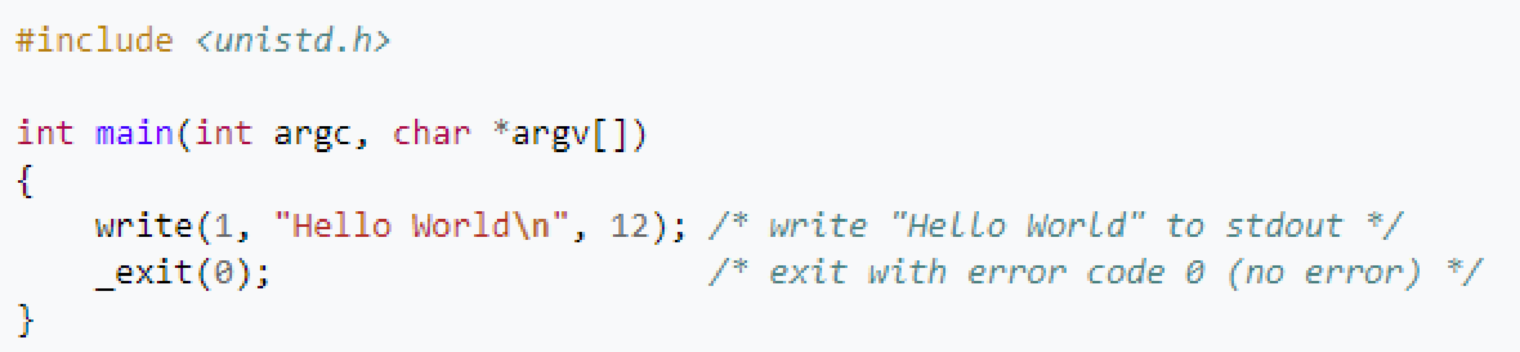
close

file descriptor

System Call Table

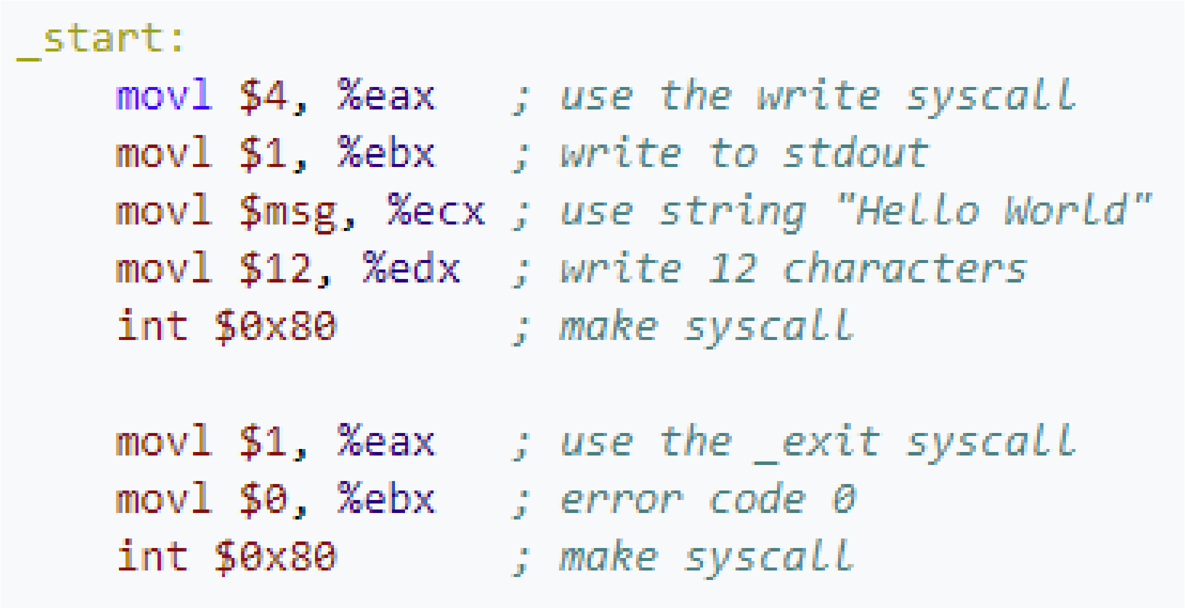
- System calls are invoked by number
- Kernel finds code to process the system call by indexing in a table
- Linux system call table:
  - 32 bit - https://chromium.googlesource.com/chromiumos/docs/+/master/constants/syscalls.md#x86-32_bit
  - 64 bit - https://chromium.googlesource.com/chromiumos/docs/+/master/constants/syscalls.md#x86_64-64_bit
- Windows system call table:
  - 32 bit - https://j00ru.vexillium.org/syscalls/nt/32/
  - 64 bit - https://j00ru.vexillium.org/syscalls/nt/64/

System Calls Example - Hello World



```
#include <unistd.h>

int main(int argc, char *argv[])
{
    write(1, "Hello World\n", 12); /* write "Hello World" to stdout */
    _exit(0);                      /* exit with error code 0 (no error) */
}
```



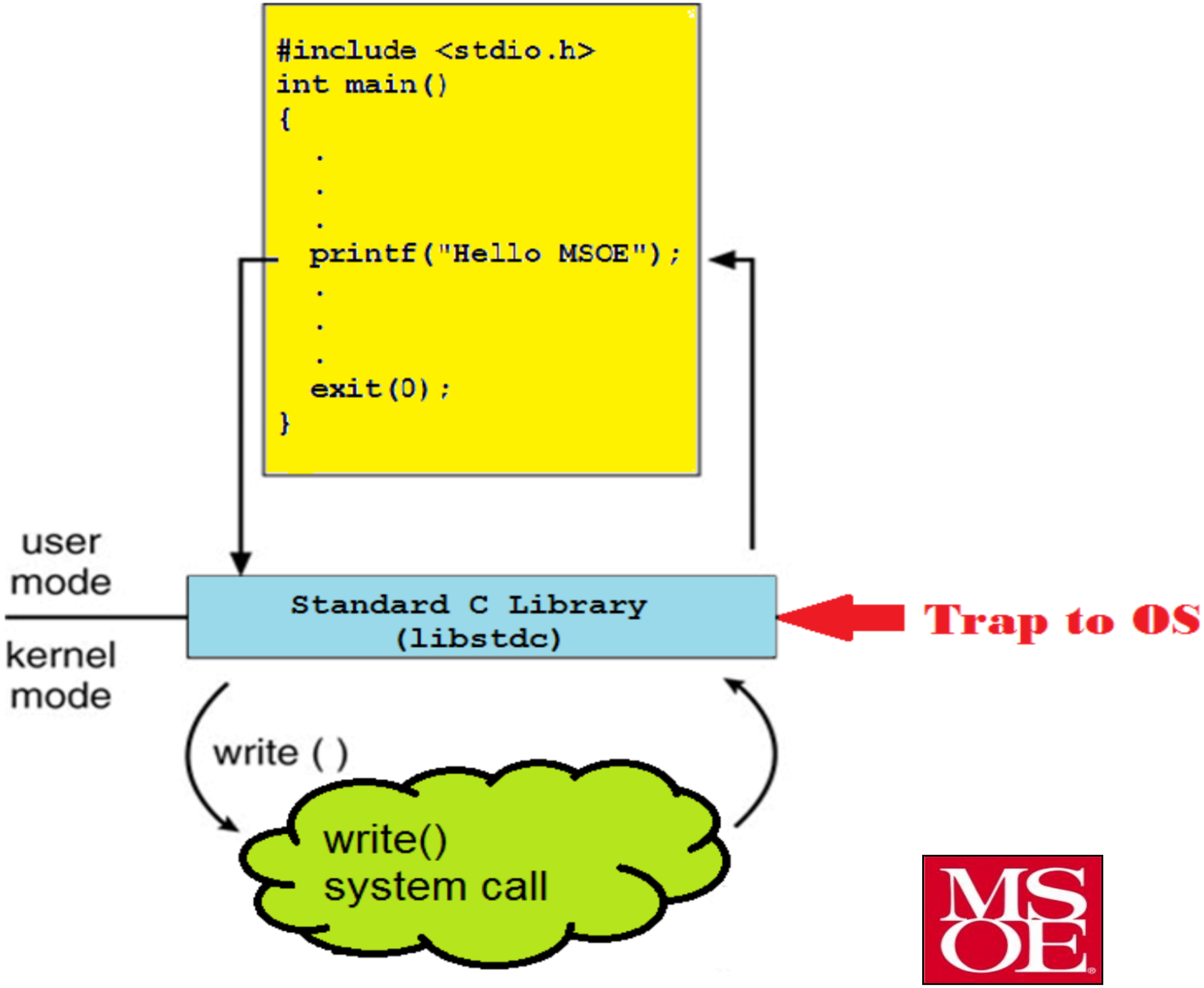
```
_start:
    movl $4, %eax   ; use the write syscall
    movl $1, %ebx   ; write to stdout
    movl $msg, %ecx ; use string "Hello World"
    movl $12, %edx  ; write 12 characters
    int $0x80       ; make syscall

    movl $1, %eax   ; use the _exit syscall
    movl $0, %ebx   ; error code 0
    int $0x80       ; make syscall
```

https://en.wikibooks.org/wiki/X86_Assembly/Interfacing_with_Linux

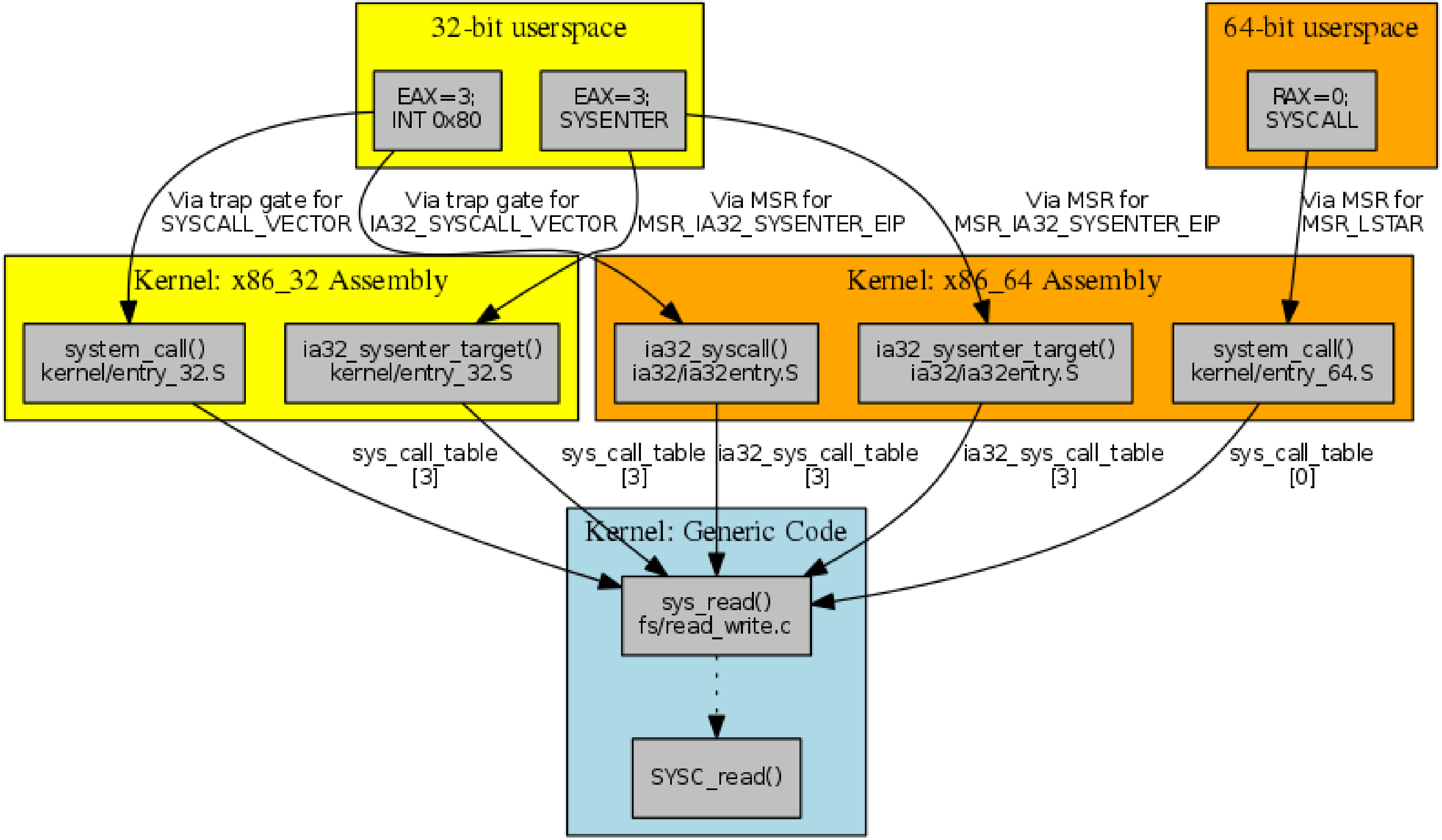
How do we invoke a system call?

- Can a system call be a function call?
- Software interrupt vs SYSENTER vs SYSCALL
- Most system calls are wrapped with user-callable functions available via the standard library
  - – Linux - libc / glibc
  - – Windows – NativeAPI (ntdll.dll)



```
#include <stdio.h>
int main()
{
   .
   .
   .
   printf("Hello MSOE");
   .
   .
   .
   exit(0);
}
```

Linux System Calls



System Calls - Questions

- How do we pass data to a system call?
- How many system calls do we need?
- What should the system calls do?
- What process executes a system call?