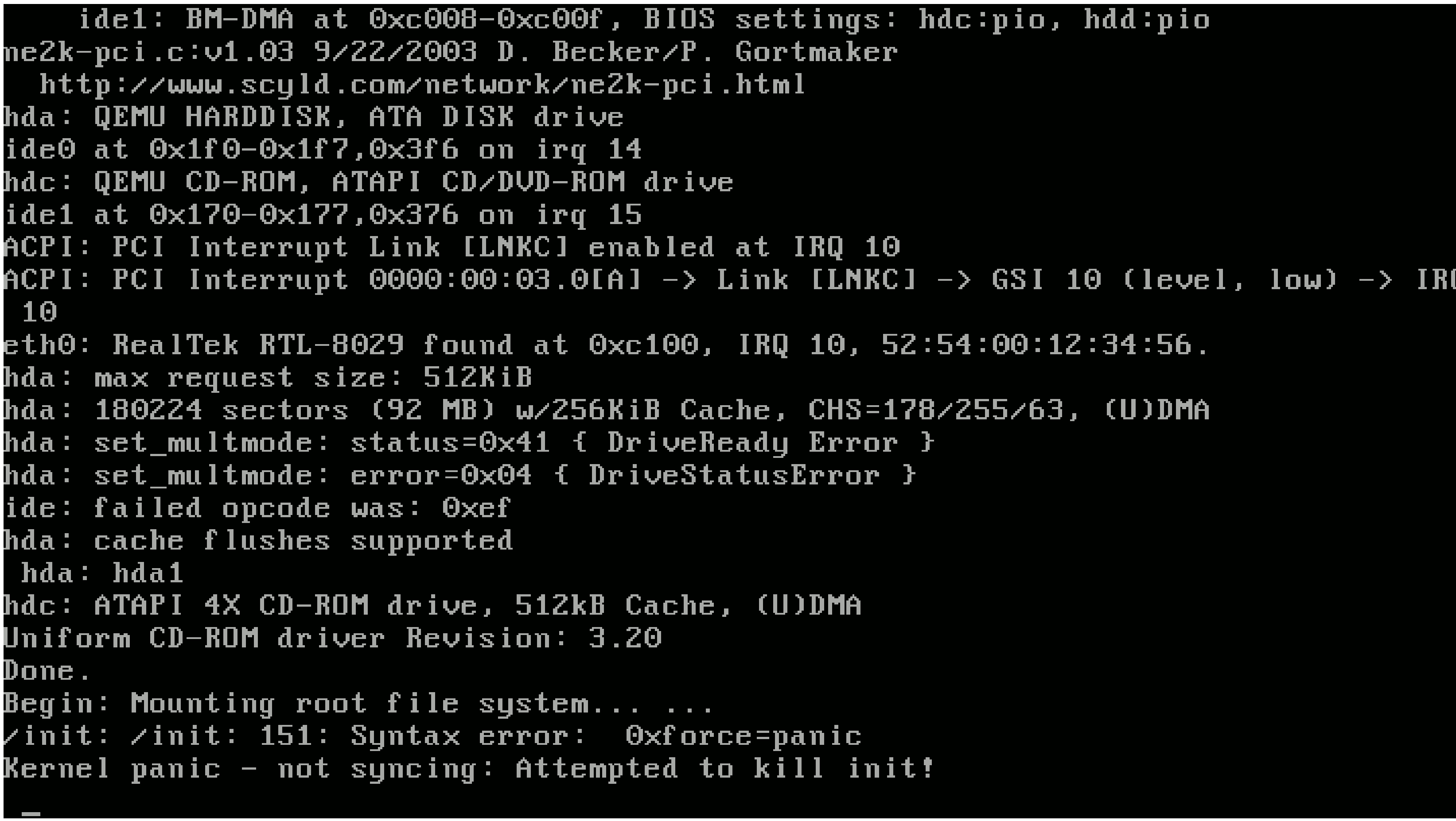
COMP3052.SEC Computer Security

Session 07: Reference Monitors



Acknowledgements

- Some of the materials we use this semester may come directly from previous teachers of this module, and other sources ...
- Thank you to (amongst others):
  - Michel Valstar, Milena Radenkovic, Mike Pound, Dave Towey, ...

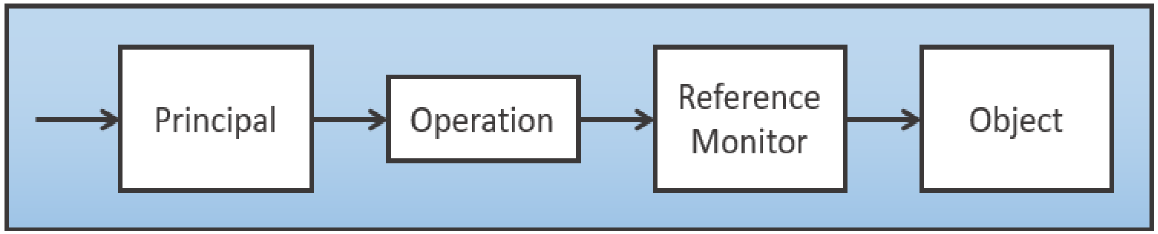
This Session

- Reference Monitors
- Operating System Integrity
- Privilege Elevation
- Memory Protection
- Page Tables

Concepts

- The Reference Monitor
  - An abstract concept
- Security Kernel
  - The implementation of a reference monitor
- Trusted Computing Base (TCB)
  - Kernel + other protection measures

Reference Monitor



"An access control concept that refers to an abstract machine that mediates all access to objects by subjects"

- Must be tamper proof / resistant
- **<u>Must always be invoked</u>** when access to an object is required
- Must be small enough to be verifiable / subject to analysis to ensure correctness

Security Kernel

"The hardware, firmware and software elements of a TCB that implement the reference monitor"

- Mediates all access
- Must be protected from modification
- Must be verifiably correct
- Usually in the bottom layers of a system

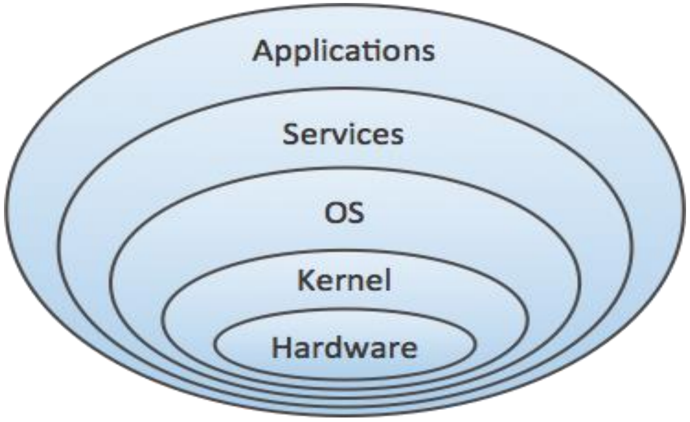
Trusted Computing Base

"The totality of protection mechanisms within a computer system responsible for enforcing a security policy"

- One or more components
- Enforce a unified security policy over a product or system
- Correct enforcement depends on components within as well as input from administrators

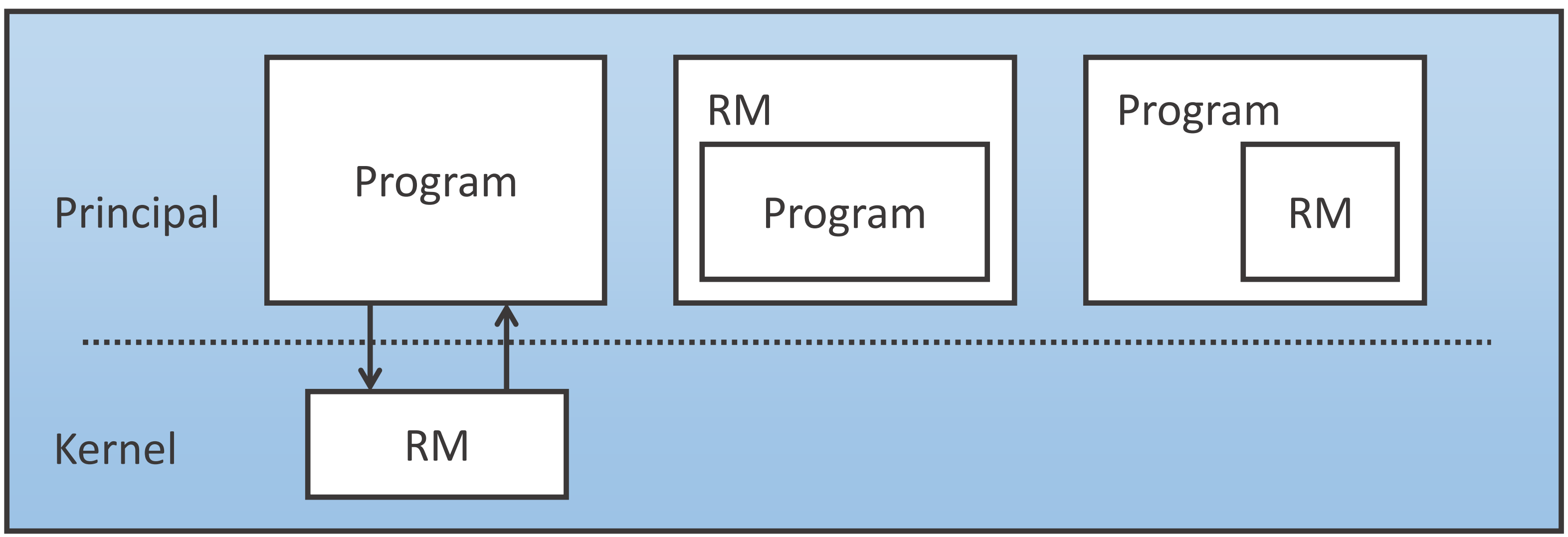
Placement

- Can be placed anywhere within a system
  - Hardware – Dedicated registers for defining privileges
  - Operating system kernel – E.g. Virtual Machine Hypervisor
  - Operating system – Windows security reference monitor
  - Services Layer – JVM, .NET
  - Application Layer – Firewalls



Placement

- Reference monitors could be placed in a variety of locations relative to the program being run



Lower Is Better

- Using a reference monitor or other security features at a lower level means:
  - We can assure a higher degree of security
  - Usually simple structures to implement
  - Reduced performance overheads
  - Fewer layer below attack possibilities
- However:
  - Access control decisions are far remote from applications

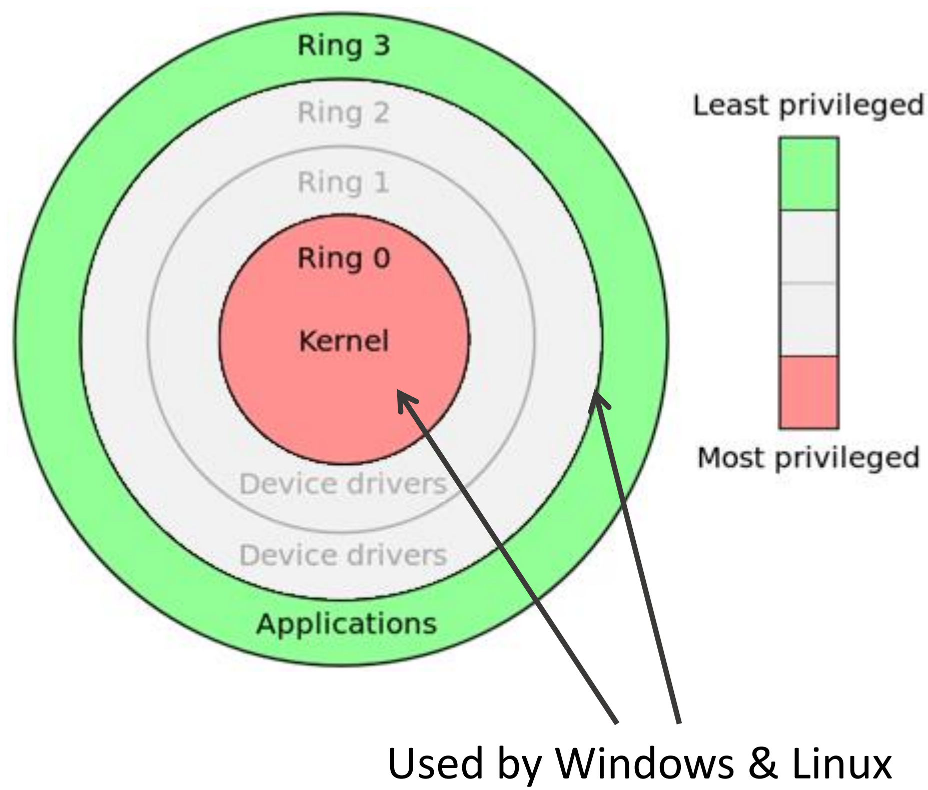
OS Integrity

- The operating system:
  - Arbitrates access requests
  - Is itself a resource that must be accessed
- This is a conflict, we want to use the OS but not mess with it

  "Users must not be able to modify the operating system"
- Modes of operation
  - Defines which actions are permitted in which mode, e.g. system calls, machine instructions, I/O
- Controlled Invocation
  - Allows us to execute privileged instructions safely, before returning to user code

Modes of Operation

- Distinguish between computations done on behalf of:
  - The OS
  - The user
- A status flag within the CPU allows the OS to operate in different modes



Controlled Invocation

- Many functions are held at kernel level, but are quite reasonably called from within user level code
    - Network and File IO
    - Memory allocation
    - Halting the CPU (at shutdown only!)

Protection Fault

- We need a mechanism to transfer between kernel mode (ring 0) and user mode (ring 3)

The Key Point

We don’t actually perform privileged operations: we ask the OS to perform them for us

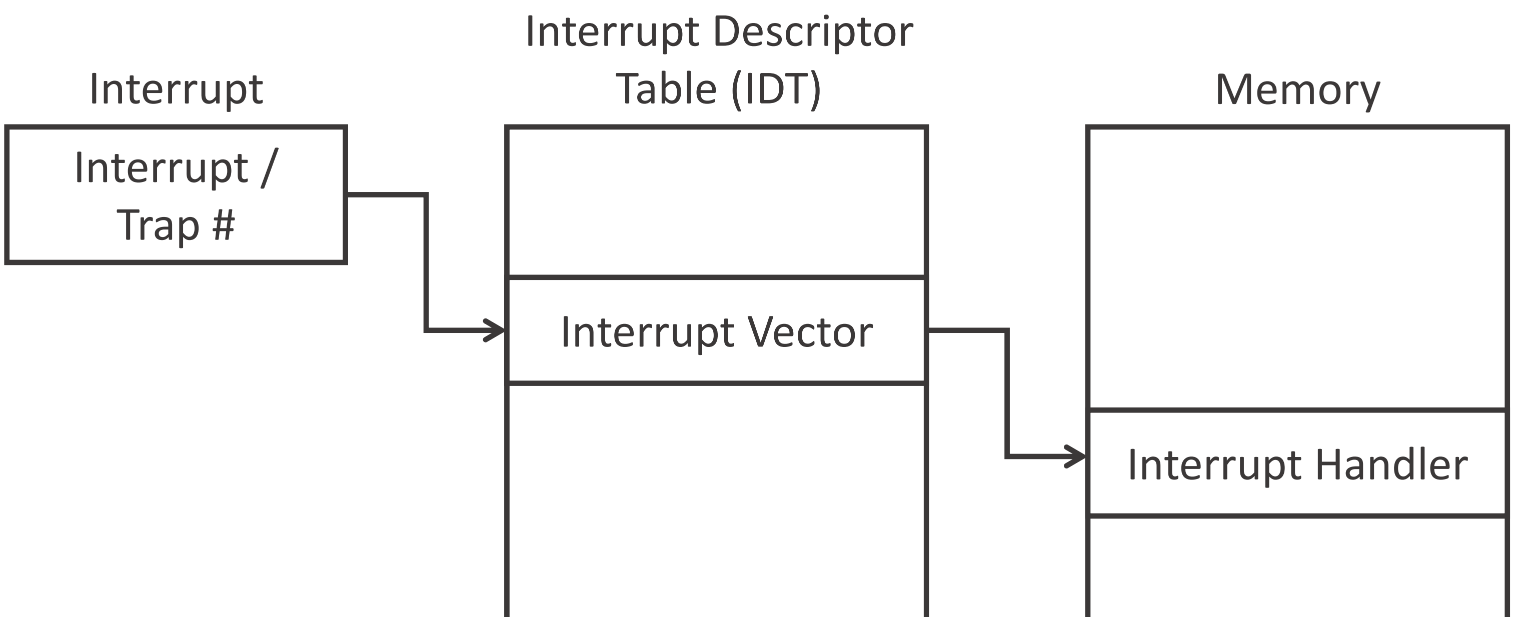
The OS can refuse to do it!

Controlled Invocation: Interrupts

- Exceptions / Interrupts / Traps
  - Called various things, for now we'll just use "Interrupt"
  - In many ways is the hardware equivalent to a software exception
- Handled by an interrupt handler which resolves the issue and returns to the original code

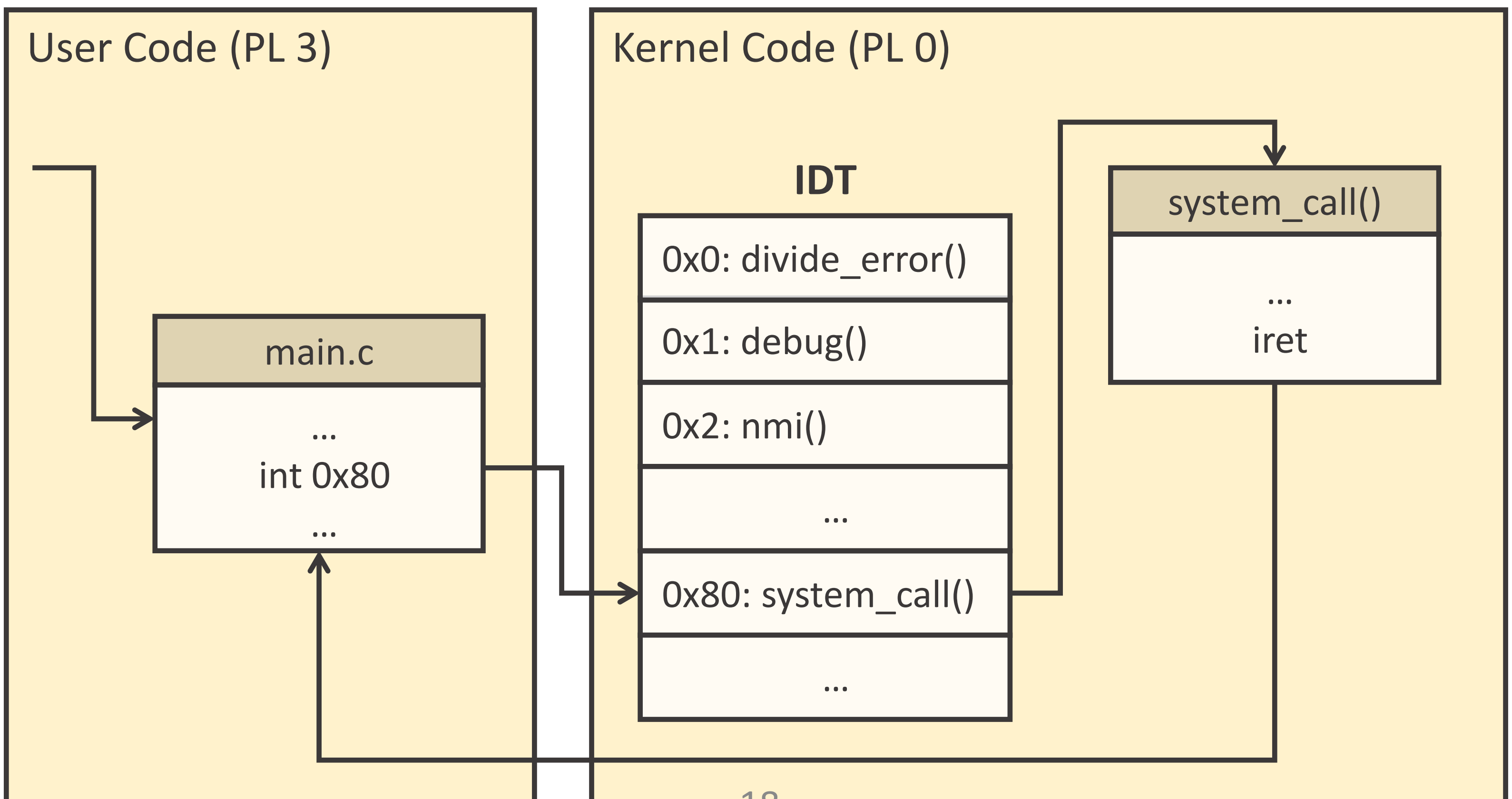
Processing an Interrupt

- Given an interrupt, the CPU will switch execution to the location given in an interrupt descriptor table (IDT)



Privilege Elevation in x86-Linux

- Linux initialises its IDT to handle syscalls at vector 0x80



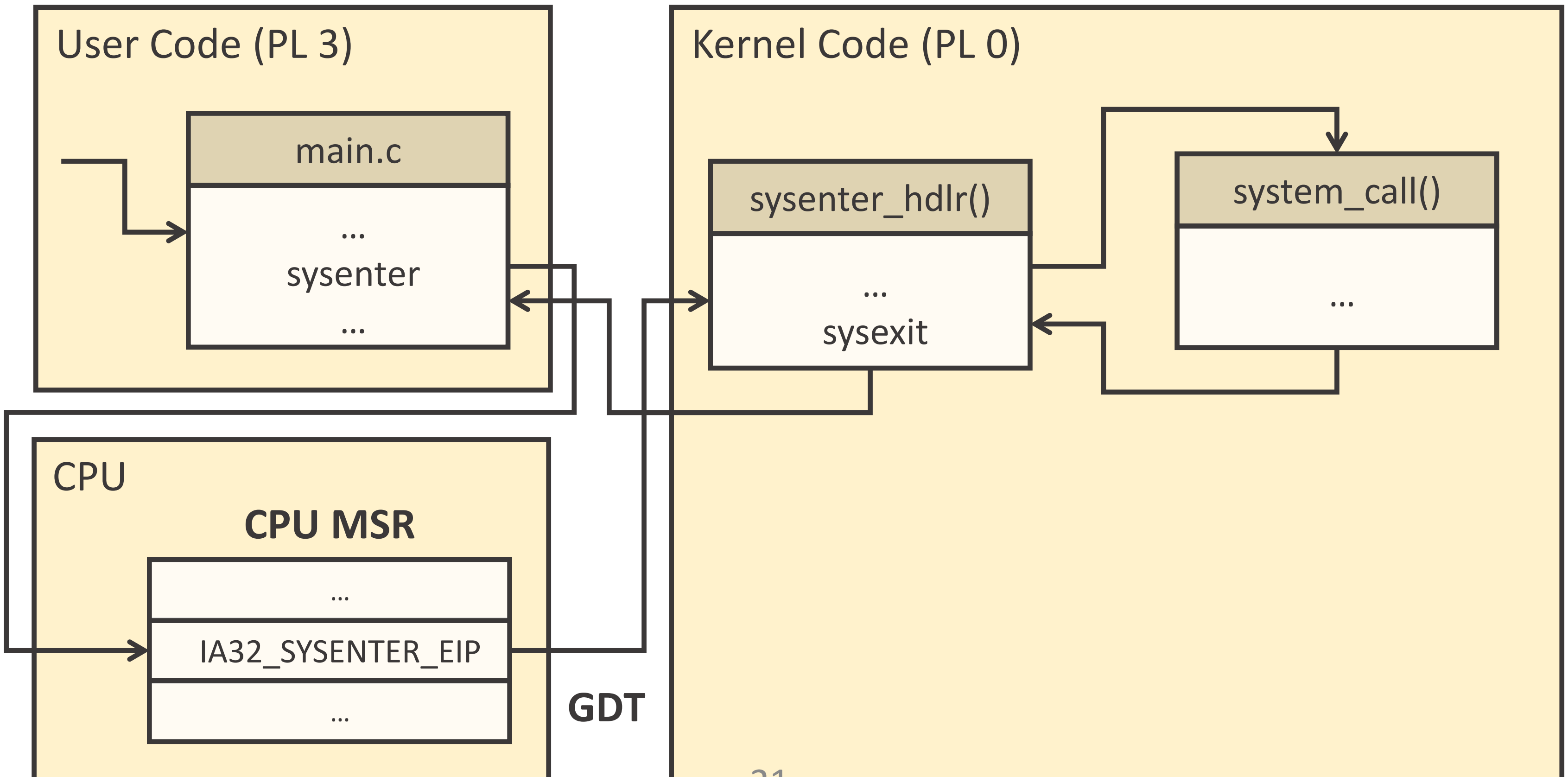
Interrupt Example

```c
#include <unistd.h>

int main(int argc, char *argv[])
{
   write(1, “Hello!", 6);
   _exit(0);
}
```

```
mov    $4, %eax
mov    $1, %ebx
mov    $msg, %ecx
mov    $6, %edx
int    $0x80

mov    $1, %eax
mov    $0, %ebx
int    $0x80
```

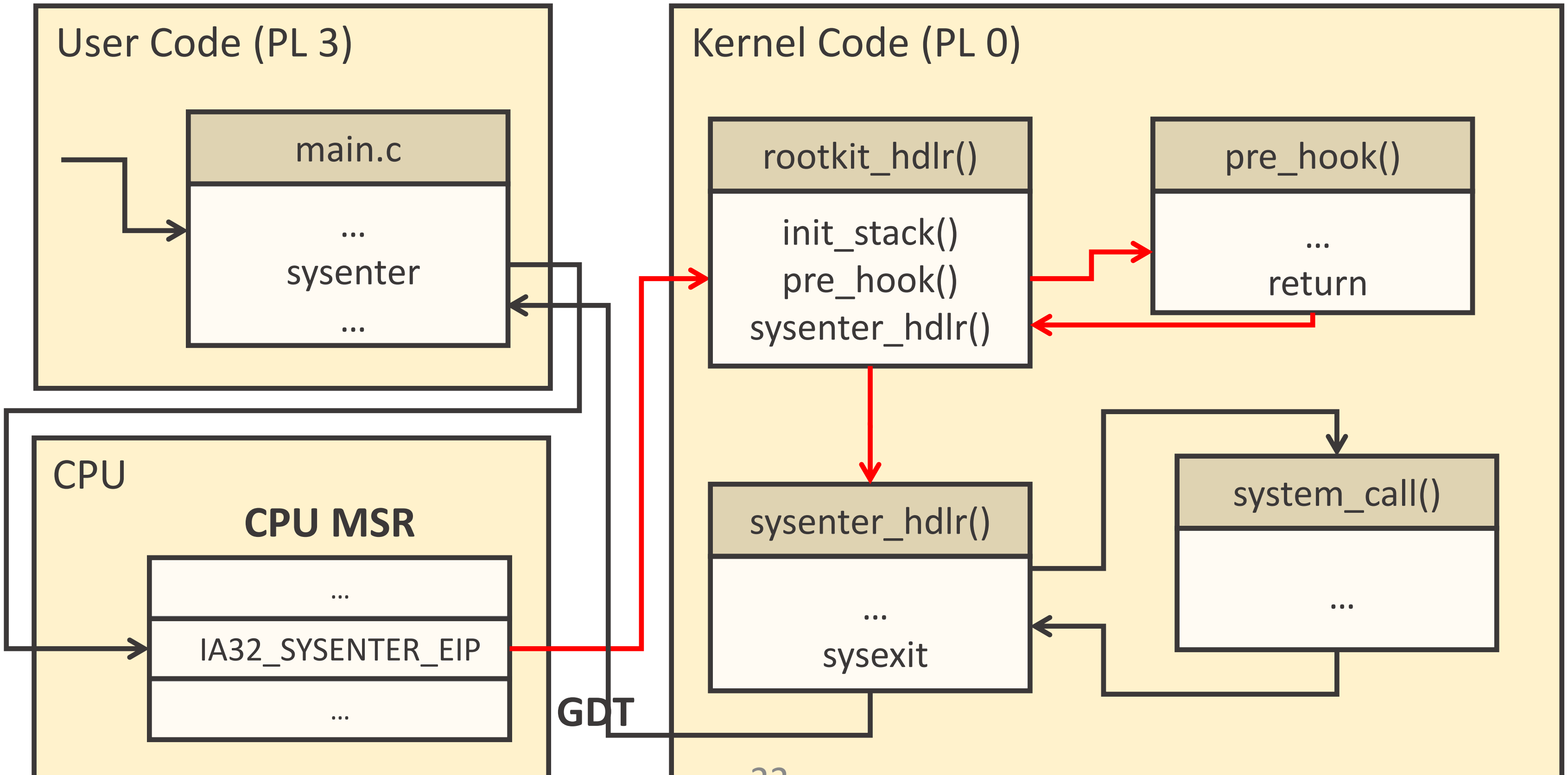
Modern Kernels

- Intel introduced the sysenter and sysexit operations with the Pentium II – much less overhead



Patching the Kernel

- If you can run custom PL 0 code (compromised driver?), you can insert your own handler – **Rootkit**



Processes and Threads

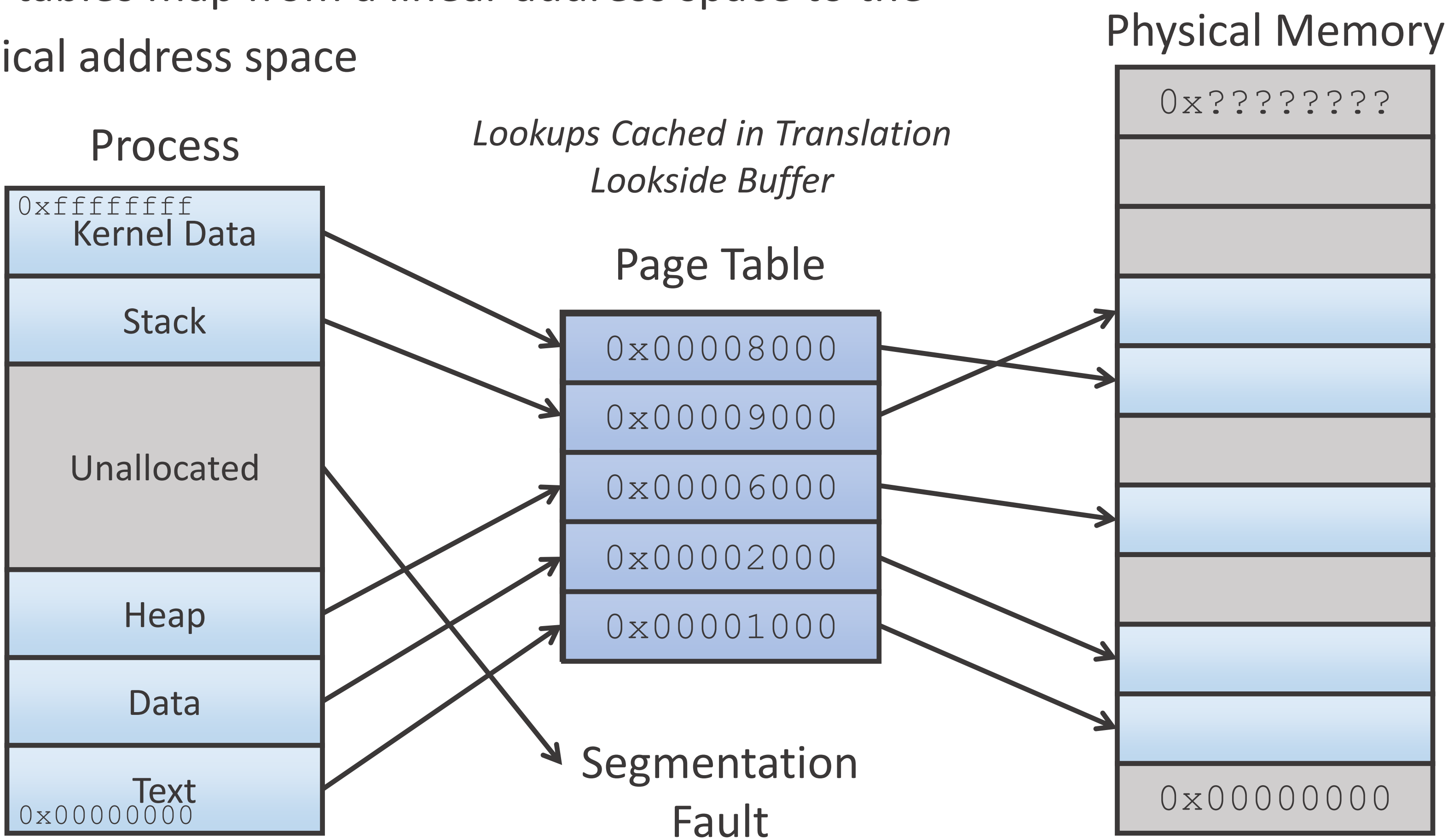
- A process is a program being executed
- Important unit of control:
  - Exists in its own address space
  - Communicates with other processes via the OS
  - Separation for security
- A Thread is a strand of execution within a process
  - Share a common address space

Memory Protection

- Segmentation – divides data into logical units
  - Good for security
  - Challenging memory management
  - Not used much in modern OSs
- Paging – divides memory into pages of equal size
  - Efficient memory management
  - Less good for access control
  - Extremely common in modern OSs

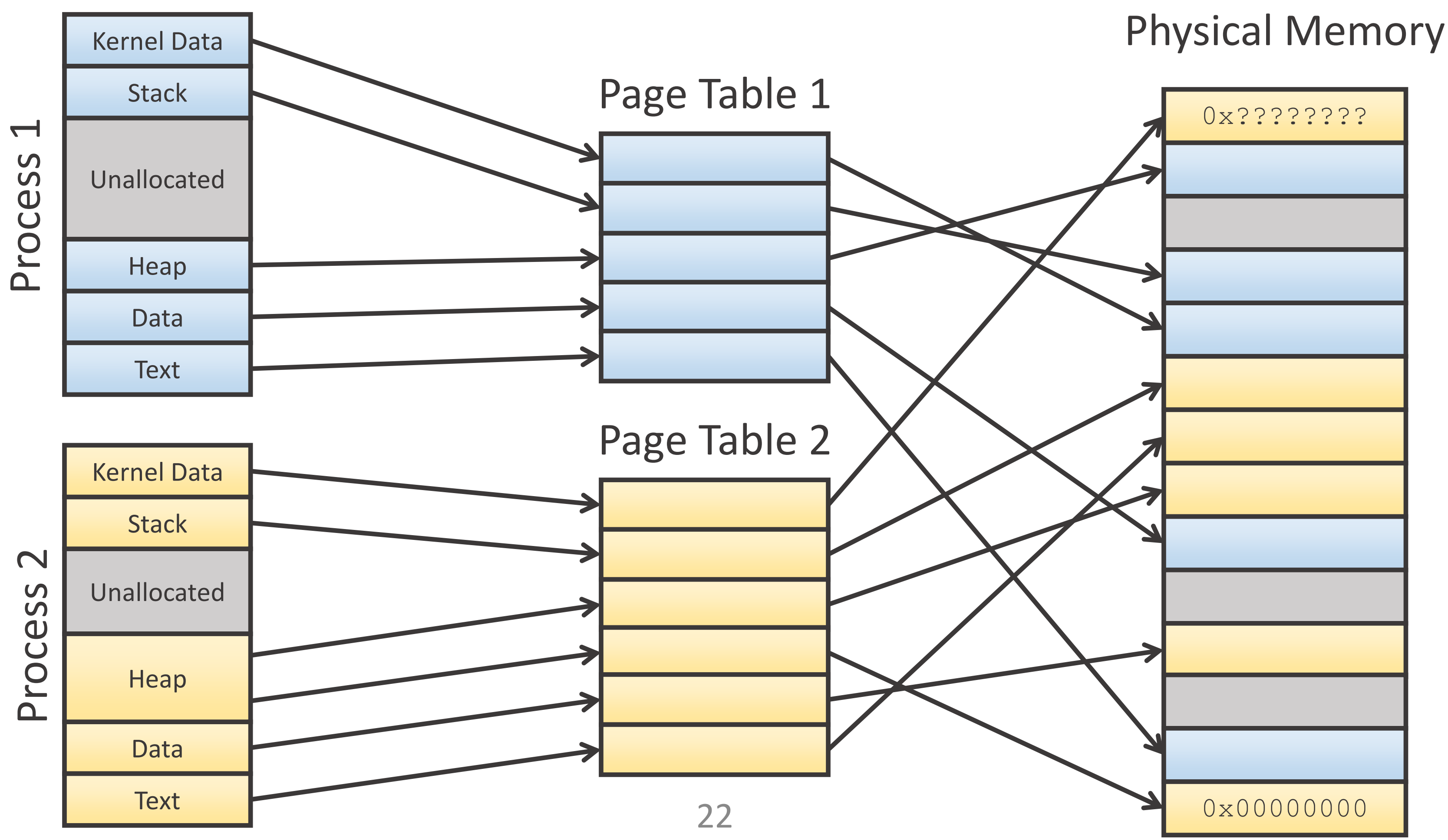
Page Tables

- All processes see an individual linear address space
- Page tables map from a linear address space to the physical address space



Page Tables

- Processes are separated by the page system



Paging

- Page Tables have a valid / invalid bit
  - Valid pages have page numbers allocated to the currently executing process
  - Invalid pages are either non-existent (not in the page table) or are in the page table but belong to other processes
- Memory access to an invalid page results in a segmentation / page fault or bus error
  - Trap causes context switch to kernel
  - Kernel sends SIGSEGV or SIGBUS to process
    - Usual behaviour is for process to end

Protecting Memory

- OS Integrity – preserved by separation of users and kernel space
- Separation of users
  - File management – logical memory object
  - Memory management – physical memory object

Summary

- Reference Monitors
- Operating System Integrity
- Privilege Elevation
- Memory Protection
- Page Tables

Gollman
Chapter 6