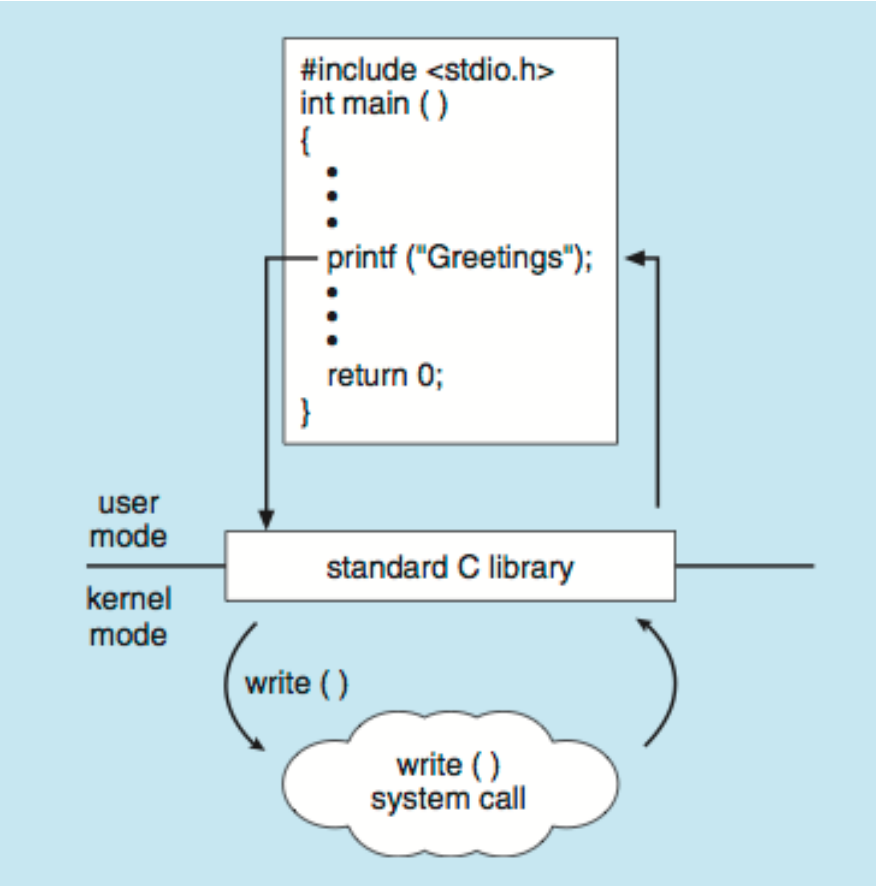
System calls

- User processes cannot perform privileged operations themselves
- Must request OS to do so on their behalf by issuing system calls
- System calls elevate privilege of user process
  - Must ensure kernel is not tricked into doing something a user process should not be doing
  - Must verify every single parameter!

Library vs. System Calls

- C program invoking printf() libc library call, which calls write() system call

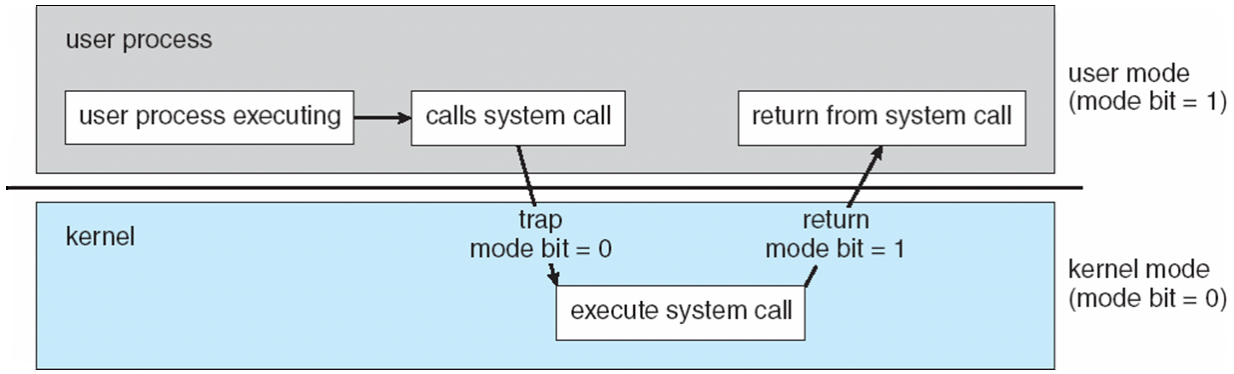


Examples of Windows and Unix System Calls

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

System Call Dispatch

- How should actual system call be invoked?
  - Program can't see kernel namespace



- Need hardware support to change privilege level
- Traps
  - Type of interrupt
  - Software interrupts and exceptions
  - Software interrupts initiated by programmer
  - Exceptions occur automatically

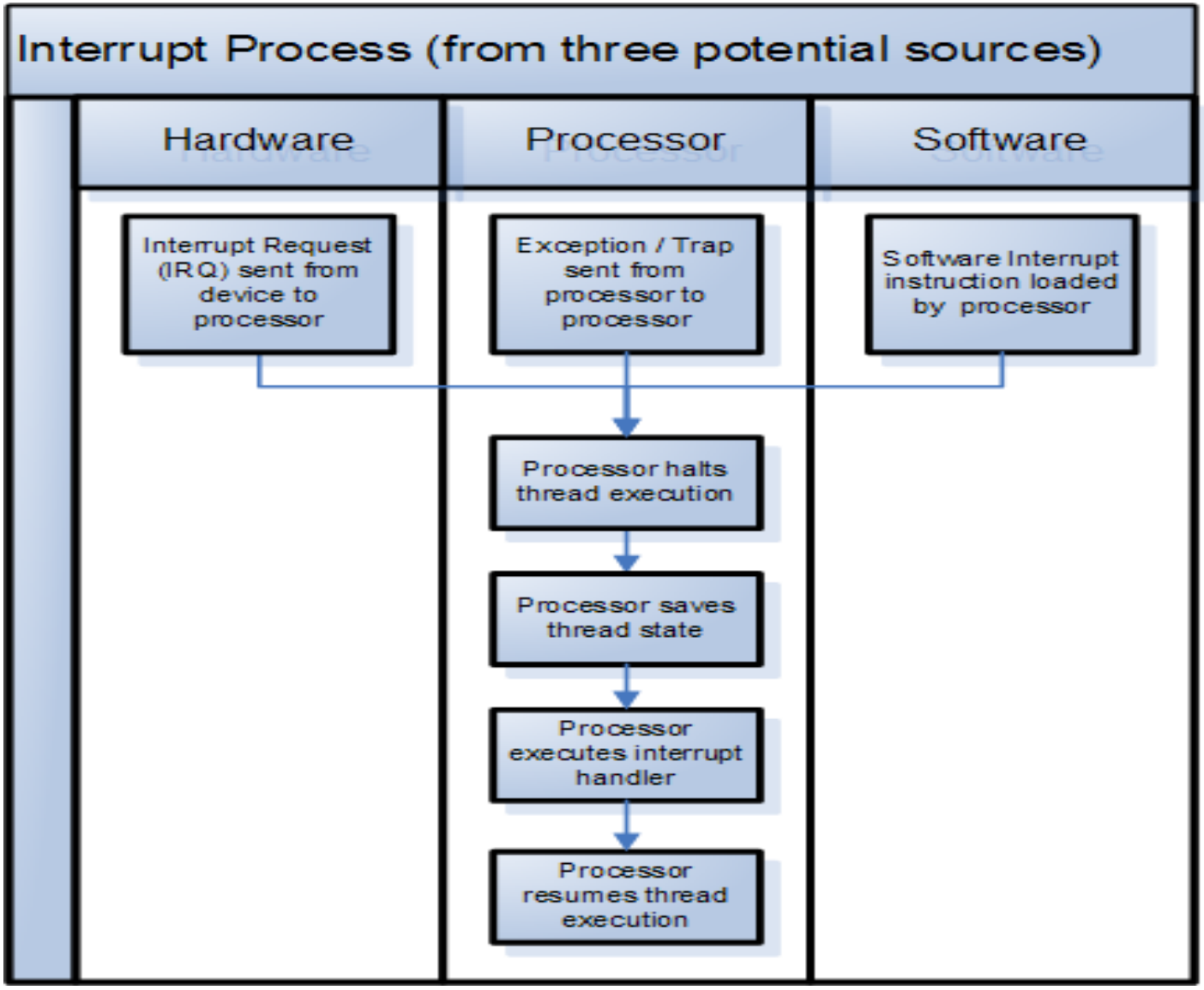
Traps, Interrupts, Exceptions

```
for(;;) {
   if (interrupt) {
      n = get interrupt number
      call interrupt handler n
   }
   fetch next instruction
   run next instruction
}
```

```
for(;;) {
   fetch next instruction
   run next instruction {
      if (instr == "int n")
        call interrupt handler n
   }
   if (error or interrupt) {
      n = get error or interrupt type
      call interrupt handler n
   }
}
```

- On x86, int n (n=0:255) calls interrupts n
- Some interrupts are privileged
- Can't be called by user mode
- Others aren't, e.g., syscalls
- Processor transitions to privileged mode when handling interrupt

Three kinds of interrupts

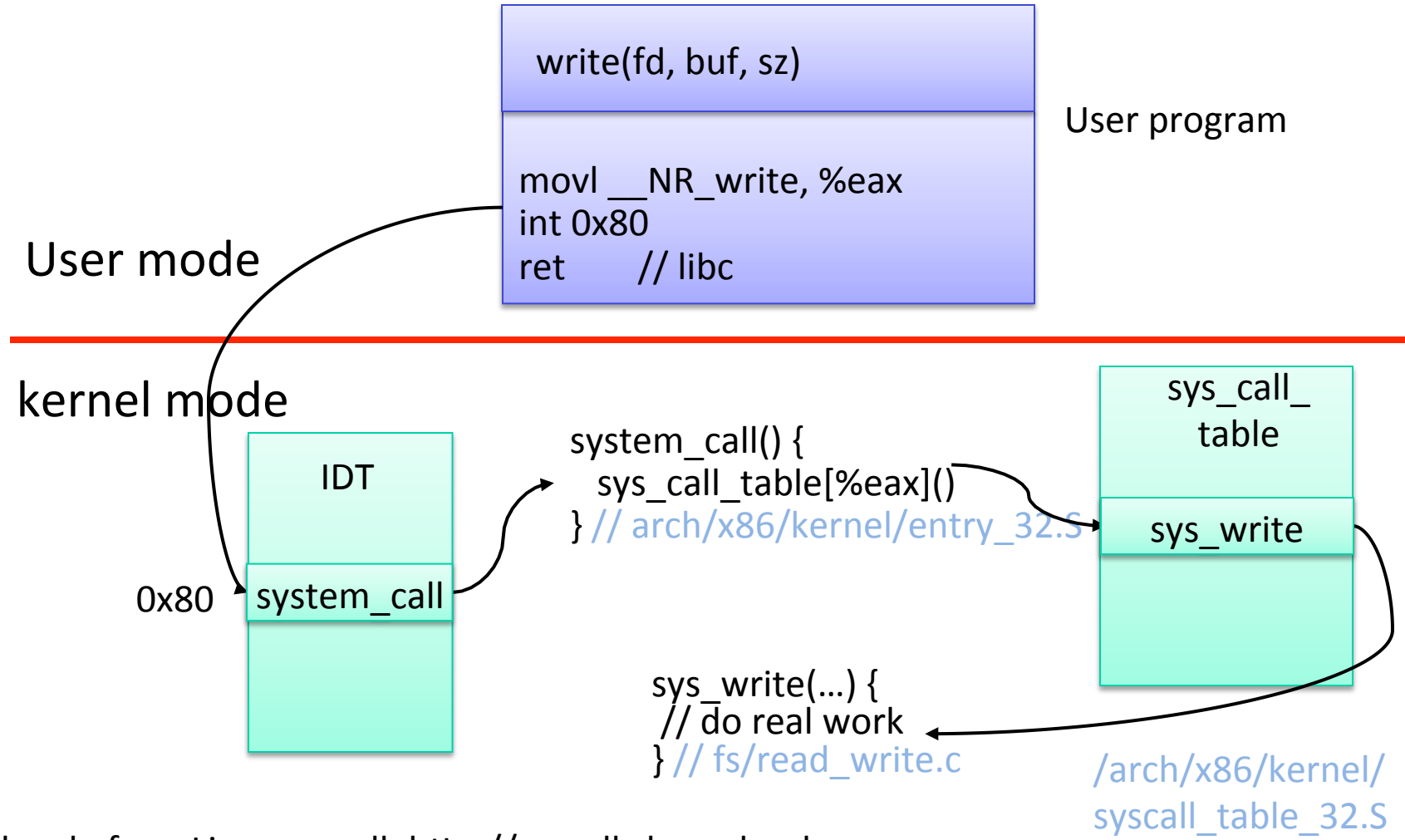


System call dispatch

1. Kernel assigns system call type a system call number
2. Kernel initializes system call table, mapping system call number to functions implementing the system call
   - Also called system call vector

3. User process sets up system call number and arguments
4. User process runs int X (on Linux, X=80h)
5. Hardware switches to kernel mode and invokes kernel's interrupt handler for X (interrupt dispatch)
6. Kernel looks up syscall table using system call number
7. Kernel invokes the corresponding function
8. Kernel returns by running iret (interrupt return)

Linux System Call Dispatch



To find code for a Linux syscall: http://syscalls.kernelgrok.com