# SEMINAR 5

# SYSCALLS

## AND A BIT OF X86 ASSEMBLY

# PREVIOUSLY

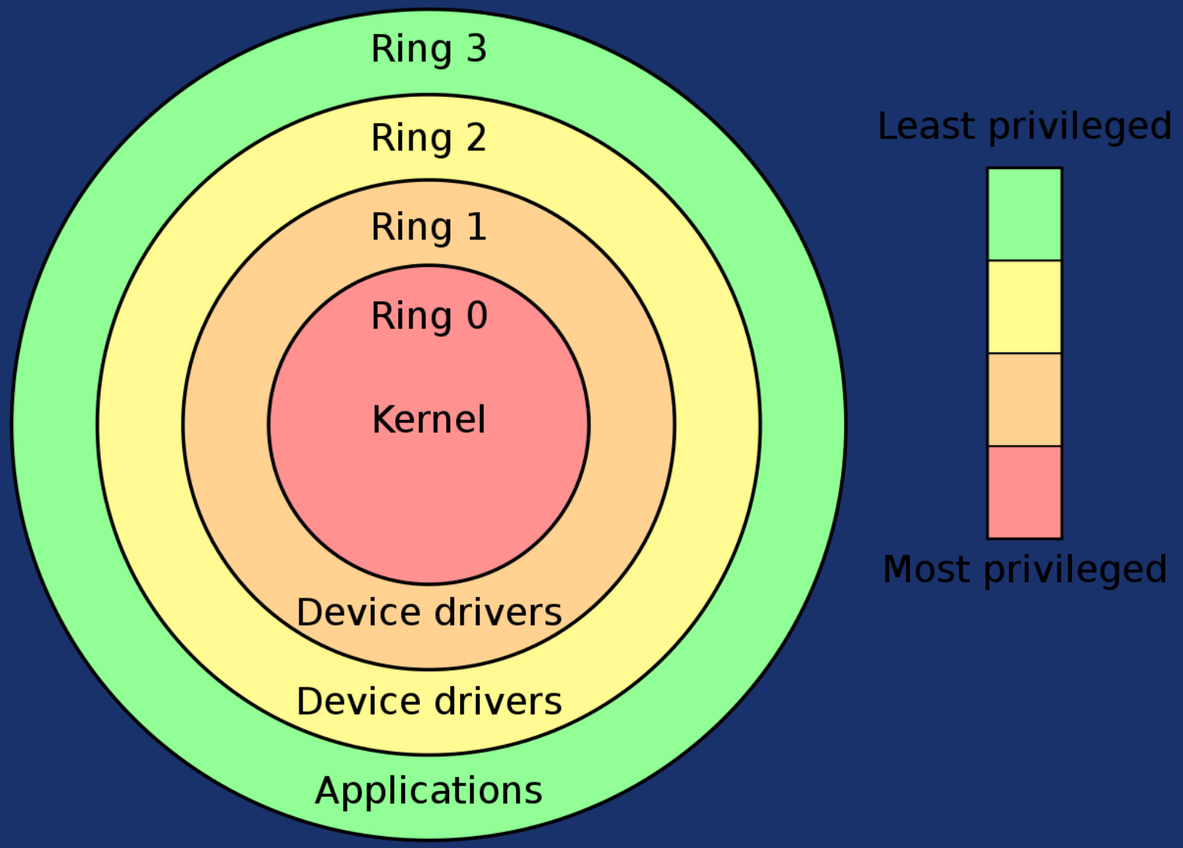
WE USED A LOW-LEVEL LANGUAGE BUT RELIED ON GLIBC TO MANAGE EFFECTS

# DEMO

## BARE METAL

# WONDERFUL (ALBEIT UNSAFE) DOS

# PRIVILEGE RINGS



# HOW SYSTEM CALL IS HANDLED

- ENABLE PROTECTION OF USERSPACE CODE
- REPLACE USER STACK WITH KERNEL STACK
- SAVE REGISTERS ON STACK
- PROCESS CALL
- RESTORE REGISTERS AND STACK
- DISABLE PROTECTION
- EXIT SYSTEM CALL

# INT 80H

# OLDER MECHANISM THAT WORKS ONLY WITH 32-BIT ADDRESSES

# SYSENTER/SYSEXIT OMITS CERTAIN CHECKS TO WORK FASTER

HOWEVER

IT IS NOT

AMD-COMPATIBLE

# SYSCALL/SYSRET

INTRODUCED BY AMD

SIMILAR TO SYSENTER

BUT USABLE

# VSYSCALL

ALLOWS TO EXECUTE CERTAIN SYSCALLS IN USERSPACE

# VDSO

## (VIRTUAL DYNAMIC SHARED OBJECT)

## BETTER IMPLEMENTATION OF VSYSCALL

# PERFORMANCE COMPARISON

| IMPLEMENTATION | TIME W/O KPTI (NS) | TIME W/ KPTI (NS) |
|---|---|---|
| INT 80H | 317 | 498 |
| SYSENTER | 150 | 338 |
| SYSCALL | 103 | 278 |
| VSYSCALL (EMUL) | 496 | 692 |
| VSYSCALL (NATIVE) | 103 | 278 |
| VDSO | 37 | 37 |

# A FEW WORDS ABOUT X86 ARCHITECHTURE

- CISC
- 16 GENERAL-PURPOSE 64-BIT REGISTERS: RAX, RBX, RCX, RDX, RBP, RSI, RDI, RSP, R8-R15
- RSP IS STACK POINTER
- LINUX USES CALLING CONVENTION DEFINED IN SYSTEM V AMD64 ABI

# SYSTEM V AMD64 ABI CALLING CONVENTION

- ARGUMENTS PASSED IN RDI, RSI, RDX, RCX, R8, R9, REST ON STACK
- RETURN VALUE IN RAX
- RBX, RSP, RBP, R12-R15 ARE CALLEE-SAVED
- REST ARE CALLER-SAVED

# LINUX SYSCALL CALLING CONVENTION

- RAX FOR SYSCALL NUMBER
- RDI. RSI. RDX. R10. R8. R9 FOR ARGUMENTS
- RAX. RDX FOR RETURN VALUE
- RCX AND R11 ARE DESTROYED

# USEFUL LINKS

- SYSCALL TABLE
- X86-64 INSTRUCTION SET