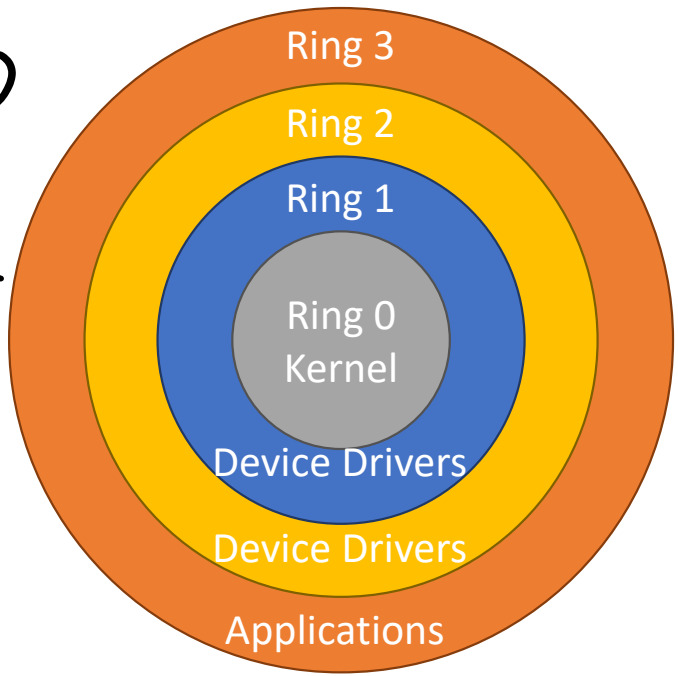
Protected Mode

- Most modern CPUs support protected mode
- x86 CPUs support three rings with different privileges
  - Ring 0: OS kernel
  - ~~Ring 1, 2: device drivers~~
  - Ring 3: userland
- Most OSes only use rings 0 and 3
- Privileged instructions?
  - https://sites.google.com/site/masumzh/articles/x86-architecture-basics/x86-architecture-basics



Borrowed from Ji-Yong Shin, CS5600 21spring

master ▾ linux / arch / x86 / entry / syscalls / syscall_64.tbl

Go to file ···

kvaneesh mm/mempolicy: wire up syscall set_mempolicy_home_node ... Latest commit 21b084f on Jan 14, 2022 History

22 contributors +10

419 lines (418 sloc) | 14.5 KB Raw Blame

```
1   #
2   # 64-bit system call numbers and entry vectors
3   #
4   # The format is:
5   # <number> <abi> <name> <entry point>
6   #
7   # The __x64_sys_*() stubs are created on-the-fly for sys_*() system calls
8   #
9   # The abi is "common", "64" or "x32" for this file.
10  #
11  0       common  read            sys_read
12  1       common  write           sys_write
13  2       common  open            sys_open
14  3       common  close           sys_close
15  4       common  stat            sys_newstat
16  5       common  fstat           sys_newfstat
17  6       common  lstat           sys_newlstat
18  7       common  poll            sys_poll
19  8       common  lseek           sys_lseek
20  9       common  mmap            sys_mmap
21  10      common  mprotect        sys_mprotect
```

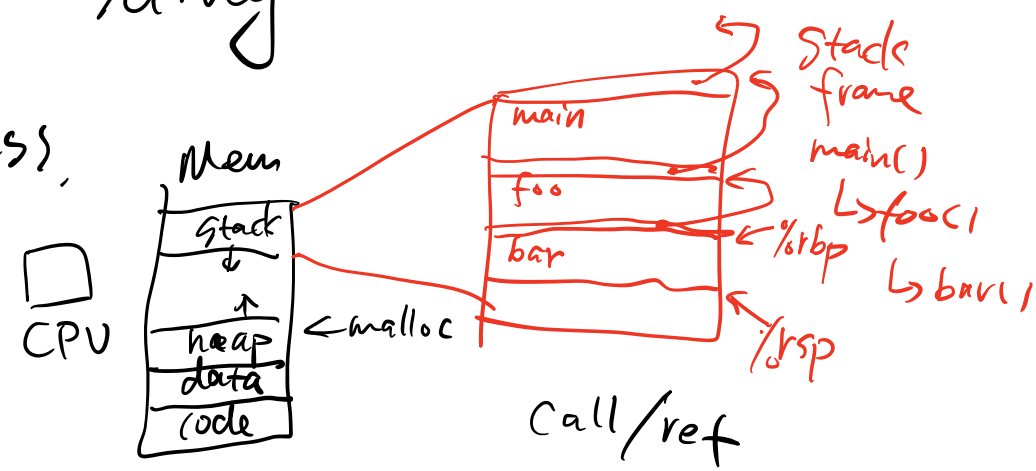
> 400

Week 3.a
CS5600
1/23 2023
https://naizhengtan.github.io/23spring/

1. Last time
2. syscall
3. process/OS control transfers
4. Process birth
5. Shell crash course

Admin Survey

Process



Calling convention

- args? → 6 register (%rdi, %rsi...)
- ret? → %rax
- Caller → callee

call-preserved / clobbered REGISTERS

Q: %rsp? %rax?

JIT

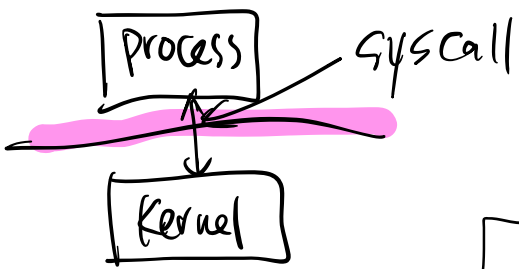
file descriptor

- here are some example system calls:

```
int fd = open(const char* path, int flags)
write(fd, const void *, size_t)
read(fd, void *, size_t)
```

"/tmp/xyz"

$ man 2 <syscalls>




```
open( . . . );
printf(. . . );
```

syscall

libc.

different

① Calling convention

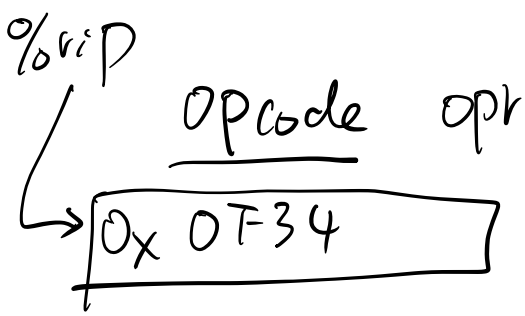
syscall ⇒ all reg preserved except %rax

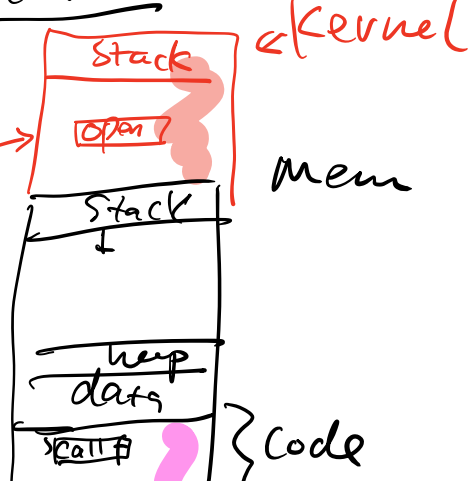
"frap"

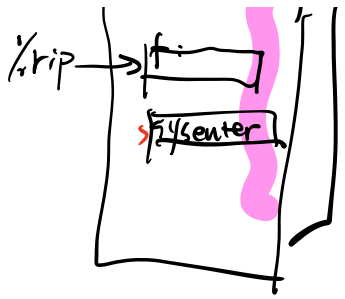
return val

② func call ⇒ Call foo

↳ sysenter





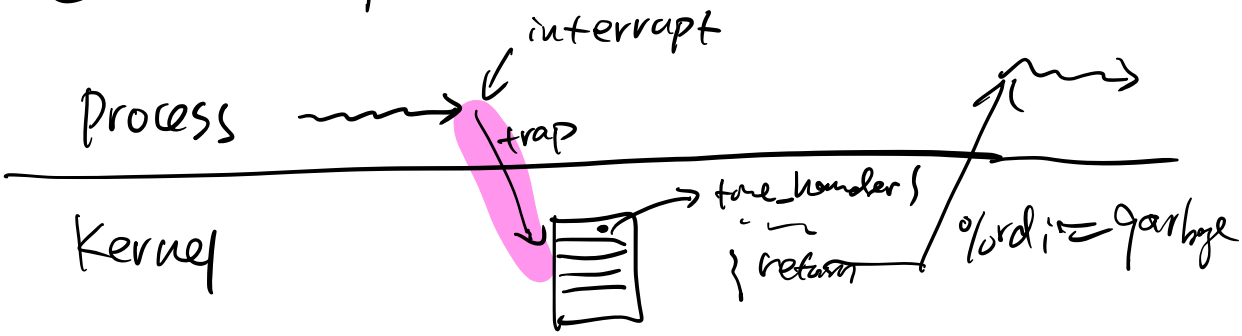


OS ↔ process

① syscall

② interrupt

timer



③ exception

```
x/0;
int *p = 0xdeadbeef;
*p = 1;
```

- page fault (VM)
- JVM:

```
if (rarely happens) {
  A
} else {
  B
}
```

⇒ access ptr: B

exception handler: