

Programovanie v operačných systémoch

01 - syscalls, IO

Jozef Šiška



Department of Applied Informatics
Comenius University in Bratislava

2020/2021

OS

HW intermezzo – interrupts

syscalls

HW intermezzo – IO

Input/Output

Filesystem

Von Neumann architecture

- ▶ data and program in the same memory

OS:

- ▶ Process management
- ▶ Resource management
 - ▶ Memory
 - ▶ HW
- ▶ kernel vs userspace processes
 - ▶ kernel run with highest privileges
 - ▶ userspace process need to ask the kernel to perform some operations
→ syscall

Interrupts

- ▶ hardware
- ▶ software
 - ▶ ... used to "invoke OS functions"
- ▶ interrupt vector table

Invoking services in kernel - syscall

We need to

- ▶ pass parameters to kernel
- ▶ actually switch to kernel "process" / thread of execution
- ▶ software interrupt (0x80 linux, 0x23 win?)
- ▶ special instruction (sysenter, syscall)

Syscall example

```
write(1,"ahoj",5);
```

```
000000000040099e <main>:
```

```
...
```

```
4009a2: ba 05 00 00 00      mov     $0x5,%edx
4009a7: be 84 77 48 00      mov     $0x487784,%esi
4009ac: bf 01 00 00 00      mov     $0x1,%edi
4009b1: e8 da 1a 03 00      callq   432490 <__libc_write>
```

```
...
```

```
0000000000432490 <__libc_write>:
```

```
432490: 83 3d 25 19 28 00 00  cmpl     $0x0,0x281925(%rip) #<__libc_multiple_threads
432497: 75 14                jne     4324ad <__write_nocancel+0x14>
```

```
0000000000432499 <__write_nocancel>:
```

```
432499: b8 01 00 00 00      mov     $0x1,%eax
43249e: 0f 05                syscall
4324a0: 48 3d 01 f0 ff ff    cmp     $0xffffffffffffffff001,%rax
4324a6: 0f 83 74 34 00 00    jae     435920 <__syscall_error>
4324ac: c3                  retq
```

```
...
```

- ▶ C functions for most calls
- ▶ `syscall` fallback - takes syscall number as argument
- ▶ `man syscalls` or `/usr/include/sys/syscall.h`
- ▶ return positive number on success (or just zero)
- ▶ negative number (-1) on errors
- ▶ real error code in global `errno` variable!
- ▶ not all POSIX calls map 1-1 to syscalls
- ▶ `opendir`, `readdir` vs `readdir`, `getdents`

- ▶ IO ports
- ▶ memory mapped
- ▶ DMA

Input/Output kernel interface

- ▶ device independence
- ▶ uniform naming
- ▶ error handling, access control
- ▶ buffering
- ▶ synchronous (blocking) / asynchronous access

Input/Output kernel interface

- ▶ device independence
- ▶ uniform naming
- ▶ error handling, access control
- ▶ buffering
- ▶ synchronous (blocking) / asynchronous access

- ▶ block devices
- ▶ character devices

Input/Output kernel interface

- ▶ device independence
- ▶ uniform naming
- ▶ error handling, access control
- ▶ buffering
- ▶ synchronous (blocking) / asynchronous access
- ▶ block devices
- ▶ character devices
- ▶ open, close, read, write, (seek, ioctl,...)
- ▶ file descriptor (handle)
- ▶ special device nodes in filesystem

- ▶ VFS (virtual filesystem)
- ▶ mounted "real" filesystems
- ▶ files
 - ▶ name, data, metadata
 - ▶ inode
 - ▶ open (creat), close, read, write, stat, ...
- ▶ directories ("folders")
 - ▶ list of entries (files, directories)
 - ▶ open, close, readdir, getdents, ...