


# quiz

Donnerstag, 16. März 2017 11:17

ETH zürich  [spcl.inf.ethz.ch](http://spcl.inf.ethz.ch)  
@spcl\_eth

## A Small Quiz

- **True or false (raise hand)**
  - A process has a virtual CPU  true
  - A thread has a virtual CPU  true
  - A thread has a private set of open files  false
  - A process is a resource container  true
  - A context switch can be caused by a thread  true, eg. syscall
  - When a process calls a blocking I/O, it is put into runnable state  false
  - A zombie is a dead process waiting for its parent  true
  - Simple user-level threads run efficiently on multiprocessors  false
  - A device can trigger a system call  false; syscalls triggered by processes
  - A device can trigger an upcall  true
  - Unix fork() starts a new program  false; duplicates program
  - Windows CreateProcess starts a new program  true
  - A buggy process can overwrite the stack of another process  false; programs have separate resource containers
  - User-level threads can context switch without a syscall  true
  - The scheduler always runs in a kernel thread  false; can run in context of calling thread

3

# Quiz

Montag, 6. März 2017 13:38

## True or false (raise hand)

- Mutual exclusion on a multicore can be achieved by disabling interrupts
- Test and set can be used to achieve mutual exclusion
- Test and set is more powerful than compare and swap
- The CPU retries load-linked/store conditional instructions after a conflict
- The best spinning time is 2x the context switch time
- Priority inheritance can prevent priority inversion
- The receiver never blocks in asynchronous IPC
- The sender blocks in synchronous IPC if the receiver is not ready
- A pipe file descriptor can be sent to a different process
- Pipes do not guarantee ordering
- Named pipes in Unix behave like files
- A process can catch all signals with handlers
- Signals always trigger actions at the signaled process
- One can implement a user-level tasking library using signals
- Signals of the same type are buffered in the kernel



# VL Quiz

Freitag, 10. März 2017 10:26

1. Base (relocation) and limit ... provide a full virtual address space (maybe)
2. Base and limit regs provide protection. true but maybe
3. segmentation provides a base and a limit for each segment. true
4. segmentation suffers from external fragmentation. true
5. segmentation allows libraries to share their code. true
6. " provides linear addressing. true (adds base and checks limit)
7. segment tables are setup for each process in the CPU (tru)
8. segmenting prevents internal fragmentation. (true)  
    cuz with pagetables, you always work with 4KB blocks. if you don't use that much, you have internal fragmentation.
9. Paging prevents internal fragmentation (false)
10. Protection information is stored at the physical frame (fals)  
    cuz two processes may have different views on the same physical page( read/writ)
11. Pages can be shared between processes (tru)
12. Same page may be writeable in process A and write protedted in proc B (tru)
13. same physical addr referenced through different address from  
    two differen processes: true  
    the same process: true
14. false

## **the many uses of address translation**

access same library from multiple processes

Process isolation

interprocess communication

<http://stackoverflow.com/questions/4856255/the-difference-between-fork-vfork-exec-and-clone>

## **How does it work**

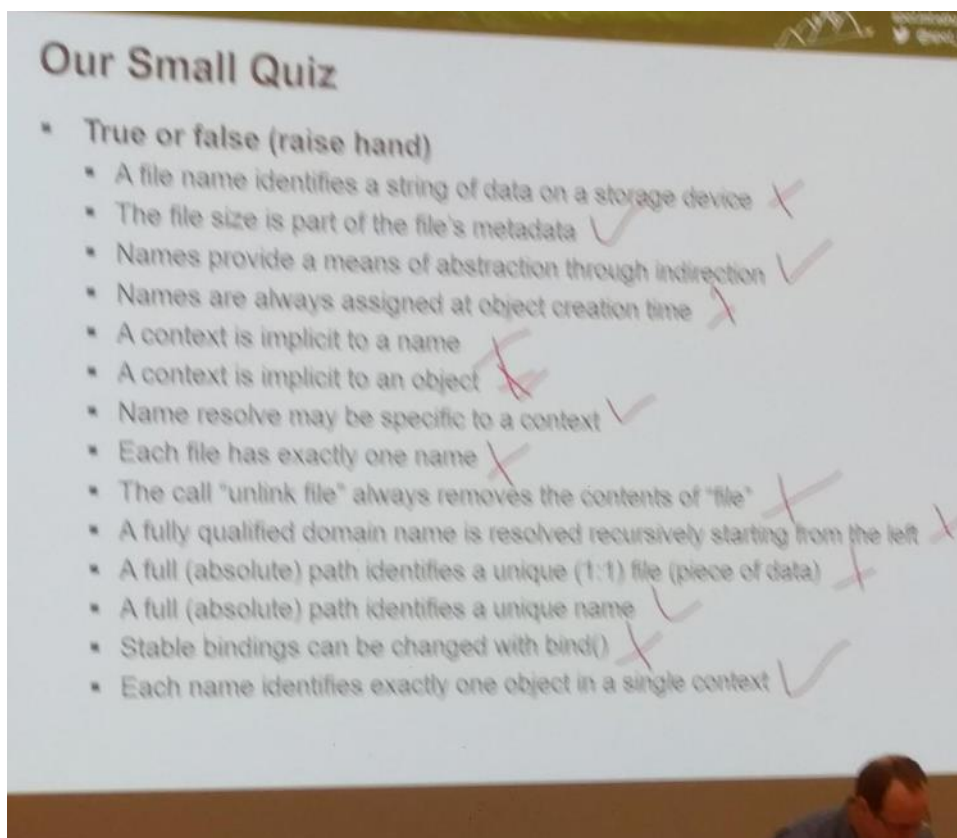
page fault handler duplicates page when read from child process

# Quiz

Freitag, 17. März 2017 10:25

## Our Small Quiz

- **True or false (raise hand)**
  - A file name identifies a string of data on a storage device
  - The file size is part of the file's metadata
  - Names provide a means of abstraction through indirection
  - Names are always assigned at object creation time
  - A context is implicit to a name
  - A context is implicit to an object
  - Name resolve may be specific to a context
  - Each file has exactly one name
  - The call "unlink file" always removes the contents of "file"
  - A fully qualified domain name is resolved recursively starting from the left
  - A full (absolute) path identifies a unique (1:1) file (piece of data)
  - A full (absolute) path identifies a unique name
  - Stable bindings can be changed with bind()
  - Each name identifies exactly one object in a single context



# Quiz

Montag, 20. März 2017 15:24

## Our small quiz

*My guess*

### ▪ True or false (raise hand)

- ✗ 1. Directories can never contain cycles ✗
- ✗ 2. Access control lists scale to large numbers of principals ✗/✓
- ✓ 3. Capabilities are stored with the principals and revocation can be complex ✓
- ✓ 4. POSIX (Unix) access control is scalable to large numbers of files ✓
- ✓ 5. Named pipes are just (special) files in Unix ✓
- ✗ 6. Memory mapping improves sequential file access ✗
- ✓ 7. Accessing different files on disk can have different speeds ✓
- ✗ 8. The FAT filesystem enables fast random access ✗
- ✓ 9. FFS enables fast random access for small files ✓
- ✗ 10. The minimum storage for a file in FFS is 8kB (4kB inode + block) ✓
- ✗ 11. Block groups in FFS are used to simplify the implementation ✗
- ✗ 12. Multiple hard links in FFS are stored in the same inode ✓
- ✓ 13. NTFS stores files that are contiguous on disk more efficiently than FFS ✓
- ✓ 14. The volume information in NTFS is a file in NTFS ✓

## Inode and file size in FFS

### ▪ Example:

- Inode is 1 block = 4,096 bytes
- Block addresses = 8 bytes
- Inode metadata = 512 bytes

### ▪ Hence:

- $(4,096 - 512) / 8 = 448$  block pointers
- $448 * 4,096 = 1,792$  kB max. file size

TODO: read about hard links in FS

# Quiz

Freitag, 24. März 2017 10:31

solution: Nnnnyynnnny

## True or false (raise hand)

- Open files are part of the process' address-space ✗
- Unified buffer caches improve the access times ✗
- A partition table can unify the view of multiple disks ✗
- Unix enables to bind arbitrary file systems to arbitrary locations ✓
- The virtual file system interface improves modularity of OS code ✓
- Programmed I/O is efficient for the CPUs ✗
- DMA enables devices to access virtual memory of processes ✗
- IOMMUs enable memory protection for devices ✓
- IOMMUs improve memory access performance ✗
- First level interrupt handlers process the whole request from the hardware ✗
- Software interrupts reduce the request processing latency ✗
- Deferred procedure calls execute second-level interrupt handlers ✓

## Mid-lecture mini-quiz

### ▪ Character or block device (raise hand)

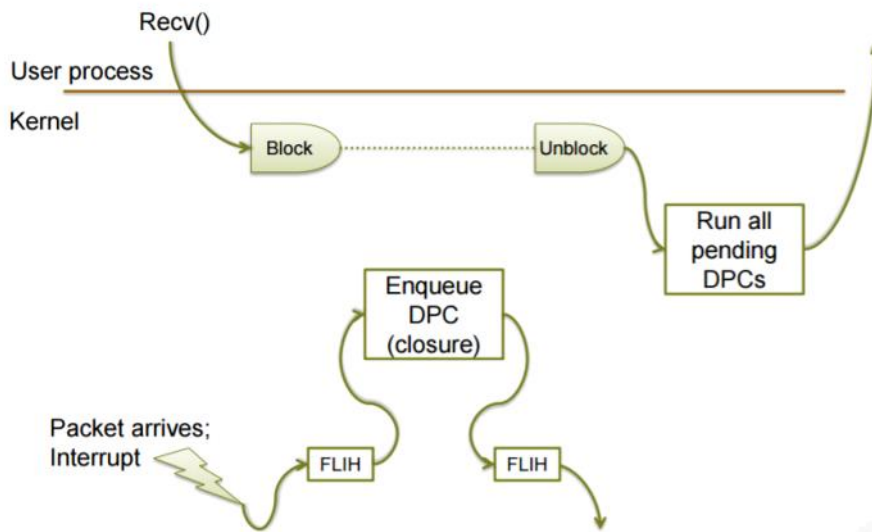
- Video card
- USB stick
- Microphone
- Screen (graphics adapter)
- Network drive

The purpose of a VFS is to allow client applications to access different types of concrete file systems in a uniform way. A VFS can, for example, be used to access [local](#) and network storage devices transparently without the client application noticing the difference. It can be used to bridge the differences in [Windows](#), [classic Mac OS/macOS](#) and [Unix](#) filesystems, so that applications can access files on local file systems of those types without having to know what type of file system they are accessing.

Aus <[https://en.wikipedia.org/wiki/Virtual\\_file\\_system](https://en.wikipedia.org/wiki/Virtual_file_system)>

A **Deferred Procedure Call (DPC)** is a [Microsoft Windows](#) operating system mechanism which allows high-priority tasks (e.g. an [interrupt handler](#)) to defer required but lower-priority tasks for later execution.

Aus <[https://en.wikipedia.org/wiki/Deferred\\_Procedure\\_Call](https://en.wikipedia.org/wiki/Deferred_Procedure_Call)>



FLIH = First level interrupt handler

## 1<sup>st</sup>-level Interrupt Handler (FLIH)

- Linux calls this the “top half”.
- In contrast to every other OS on the planet.

interrupt handlers are divided into two parts: the **First-Level Interrupt Handler (FLIH)** and the **Second-Level Interrupt Handlers (SLIH)**. FLIHs are also known as *hard interrupt handlers* or *fast interrupt handlers*, and SLIHs are also known as *slow/soft interrupt handlers*, or [Deferred Procedure Calls](#) in Windows. A FLIH implements at minimum platform-specific interrupt handling similar to *interrupt routines*. In response to an interrupt, there is a [context switch](#), and the code for the interrupt is loaded and executed. The job of a FLIH is to quickly service the interrupt, or to record platform-specific critical information which is only available at the time of the interrupt, and [schedule](#) the execution of a SLIH for further long-lived interrupt handling.<sup>1</sup>

Aus <[https://en.wikipedia.org/wiki/Interrupt\\_handler](https://en.wikipedia.org/wiki/Interrupt_handler)>

# Quiz

Montag, 27. März 2017 13:22

## Our small quiz

Guess  
actual

### ▪ True or false (raise hand)

- Spooling can be used to improve access times ~~X~~ ✓ ✓
- Buffering can cope with device speed mismatches ✓ ✓
- The Linux kernel identifies devices using a single number ~~X~~ ✓ ✓
- From userspace, devices in Linux are identified through files ✓ ✓
- Standard BSD sockets require two or more copies at the host ✓ ✓
- Network protocols are processed in the first level interrupt handler ✓ ✓
- The second level interrupt handler copies the packet data to userspace ✓ ✓
- Deferred procedure calls can be executed in any process context ✓ ✓
- Unix mbufs (and skbufs) enable protocol-independent processing ~~X~~ ? ✓
- Network I/O is not performance-critical ~~X~~ ✓
- NAPI's design aims to reduce the CPU load ✓ ✓
- NAPI uses polling to accelerate packet processing ✓ ✓
- TCP offload reduces the server CPU load ✓ ✓
- TCP offload can accelerate applications ~~X~~ ✓



# QUIZ

Freitag, 31. März 2017 10:22

## Our Small Quiz

### ▪ True or false (raise hand)

- Receiver side scaling randomizes on a per-packet basis X
- Virtual machines can be used to improve application performance X
- Virtual machines can be used to consolidate servers ✓
- A hypervisor implements functions similar to a normal OS ✓
- If a CPU is strictly virtualizable, then OS code execution causes nearly no overheads ✓
- x86 is not strictly virtualizable because some instructions fail when executed in ring 1 X
- x86 can be virtualized by binary rewriting ✓
- A virtualized host operating system can set the hardware PTBR X
- Paravirtualization does not require changes to the guest OS X
- A page fault with shadow page tables is faster than nested page tables ✓
- A page fault with writeable page tables is faster than shadow page tables X
- Shadow page tables are safer than writable page tables X
- Shadow page tables require paravirtualization X

they don't fail in ring 1, they just behave differently

The whole point of paravirtualisation is to change the guest OS

Nested page table need to lookup both the PhysicalToMachine and the other one

writeable and shadow are very similar.

safer... they should all be safe

//writeable page tables need paravirtualization