Operating Systems

Tutorial 2 & 16

Michael Tänzer

os-tut@nhng.de
http://os-tut.nhng.de

Calendar Week 45

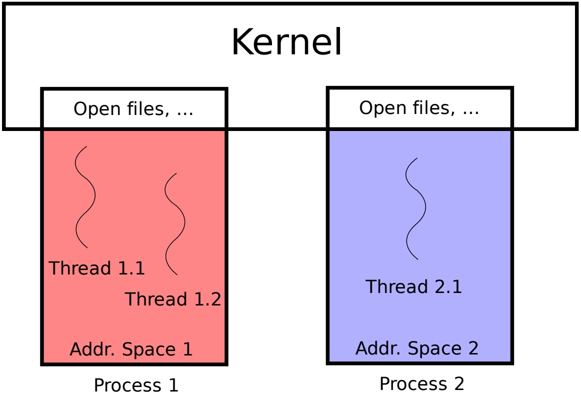
Outline

True or False

- If you are the leader of a supermarket minimizing personnel costs is a policy.
- `(0xFF - 1) == (0xFF & (~1))`
- Adding a new L0.5 cache built of faster memory than L1 will make the system faster in any case.

What are threads, processes and address spaces?

How are they related?



Difference between single- and multi-programming systems

What is the advantage of the latter?

- Single-programming: Only one application may run at a time
- Multi-programming: Multiple applications may run simultaneously
  - The CPU can do something useful while another programme waits for I/O
  - Only makes sense when doing interrupt driven I/O
- Multi-Programming $\neq$ Multi-Tasking
- But Multi-Tasking $\Rightarrow$ Multi-Programming

What is a Virtual Machine (VM)?

- Examples: JVM, VMWare, VirtualBox, XEN, ...
- A simulator (called VM monitor) which runs on machine A provides a simulation of machine B (called virtual machine).
- B doesn't need to be a 'real' machine (e. g. JVM)
- If $B \approx A$ many parts may be executed natively $\Rightarrow$ little overhead
- If B is as complex but very different from A simulation might be difficult $\Rightarrow$ large simulator, significant overhead

Why get functionality out of the kernel?

Which functionality?

- Less code (and bugs) in the kernel
- Isolation of components (no additional mechanisms required – the normal isolation used for user processes is enough)
- Everything that can be implemented outside the kernel without compromising security, protection or stability
- Sometimes the performance impact may be too big
- Examples:
  - Drivers for slow devices (parallel port, USB – libusb in Linux, low-level subsystem still in kernel though)
  - File systems (libFUSE – File system in USErspace, generic implementation still in kernel)

Compare systems based on a monolithic kernel with ones based on a μ-kernel

Strengths and weaknesses

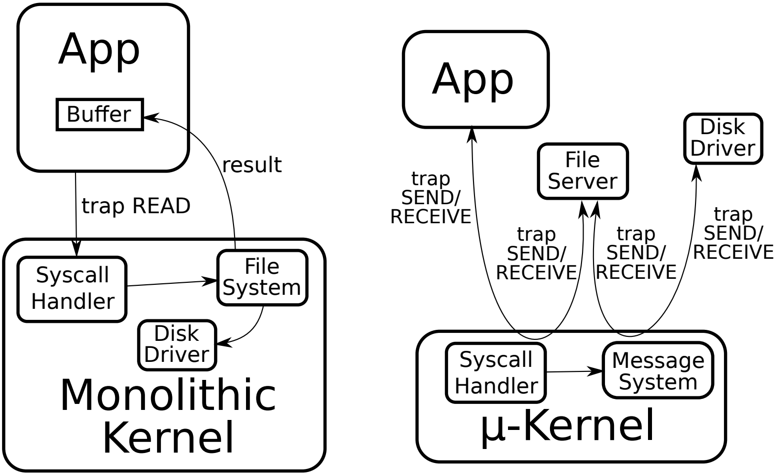
Monolithic Kernel:

- One large binary
- \+ Easy and fast service invocation through function calls
- – Complex interdependencies ⇒ difficult to extend
- – No isolation ⇒ a bug in one component can lead to corruption of the entire kernel

μ-kernel:

- Small kernel is host for servers running at user level
- \+ Each server offers a well-defined API ⇒ better structure
- \+ Malfunctions in one component can't affect others
- – Higher communication overhead

Overhead of an implementation of `read` on a monlithic opposed to a μ-kernel



Which events can lead to invocation of the kernel?

- Exceptions
- Interrupts
- System Calls

How is the `trap` instruction related to system calls?

- `trap` leads to a ‘software interrupt’ which causes the kernel to run
- It is used to implement system calls
- In principle you could also implement syscalls using exceptions

What problem exists when a syscall expects a pointer to a user buffer to write data to it?

Is reading also a problem?

- The user buffer could reach into kernel area and the kernel could overwrite its own data $\Rightarrow$ always validate user provided parameters
- The kernel might disclose secret information (e. g. write it to a file)

Questions & Comments

Any questions or comments?

The End

The End