

Extensions and Dynamic Linking/Binding

The oldest process on the oldest machine said to itself: *"I want to call some new routines, but the code wasn't even written when I started running"*

Can it be done? and how?



Extensible OSs

- What do we mean by extension?
- Is it something we want?
- Possible "extensions"
- Risks of extension
- Example systems
 - Libraries
 - Services
 - Extensible OSs
 - Spring
 - Exo-Kernel

CS 2510, cs.pitt.edu

Extensions

2



Extensions

- Adding additional abilities – I want more
 - Providing asynchronous messages over a synchronous message service
- Improving/Replacing existing services – I want different/better
 - Faster sort/search routines
 - Faster device drivers (screen, disk, network etc.)
- What does the OS provider want:
 - Allow for extensions, without compromising the operating system

CS 2510, cs.pitt.edu

Extensions

3



Basic Approaches to Extensions

- Modify the source code
 - Easy with open source, not so easy otherwise
 - Actually allows modifying the implementation of system calls
- For devices, does the OS provider write all drivers?
 - Common devices
 - New devices?
 - The OS must provide a mechanism for writing/installing device drivers
 - Review I/O introduction material for this
- Libraries
 - Static vs. Dynamic linking

CS 2510, cs.pitt.edu

Extensions

4



When do we extend code?

- Code
 - these times are general, not specifically OS code
- Possible Extension times
 - Static
 - Change the original source code
 - Link-time
 - Change the modules linked to form an executable image on disk
 - Load time
 - Change the modules that are used to form an executable image in memory
 - Dynamic
 - Change code while the process using them is actually running

CS 2510, cs.pitt.edu

Extensions

5

Architecture and Extension

- Microkernel architecture
 - Messages passed between client and server processes
 - Easier to support true dynamic extension
- As long as the message passing can support it, then changing the currently running server process is equivalent to dynamically changing the implementation of a system call
- Can we refer to system calls as dynamically linked code? What's the difference (hint: think switch)?

CS 2510, cs.pitt.edu

Extensions

6

Language and Extension

- Original Libraries (Multics '72)
 - No loader support, required trap to the kernel
- Libraries
 - Language independent
 - Binary interface
- Interpreted languages
 - Extensions can be applied to the virtual machine or the interpreter
 - Access to virtual machine = Kernel mode?

CS 2510, cs.pitt.edu

Extensions

7

Extension and Security

- Libraries
 - Extend user-space code with additional/different user-space code
 - Only risky if the executable is running with higher privileges than the user
- Plug-ins, ActiveX, and Javascript
 - Code running in user space provided by a third-party
 - Security risk is more than just malicious intent
 - Microsoft IE ActiveX control – signed by Microsoft, yet vulnerable to “buffer overflow” attack – cannot be remedied if a malicious website offers you a vulnerable version (it is authentic ... just broken)

CS 2510, cs.pitt.edu

Extensions

8

Extending the Kernel

- Traditional Micro Kernel
 - Extension through replacement of server processes
 - Have not eliminated context switch
 - You must make context switches very fast and efficient, they happen all the time now
- Micro-kernel architectures are easily extended, but inefficient if we have to route all dynamic linking through the kernel
 - Same problem was seen with the basically monolithic Multics system in the the early 70's. Dynamic linking originally had to pass through the kernel, but was dropped because this was considered inefficient.

CS 2510, cs.pitt.edu

Extensions

9

Extending the Kernel (cont'd)

- Spring
 - Objects accessed without regard to their physical location
 - Client-side stubs
 - Stubs are compiled separately and dynamically linked to any code that needs to use the objects they access
 - Micro-kernel, yet deliberately avoids entering kernel for dynamic linking ... How?
 - Address spaces are objects that can be operated upon to insert code, or to map segments so that they are shared
 - Actions on object interfaces are performed after being authenticated through an authentication agent.
 - Implemented within Spring's version of the standard C library "libc"
- This allows efficient dynamic linking in a micro-kernel OS ... but is it enough extensibility?

CS 2510, cs.pitt.edu

Extensions

10

Exo-Kernels and VM Architectures

- VM architectures attempted to multiplex the hardware
 - Each virtual machine saw its own hardware, and thought it was exclusive.
 - Extension similar to Java VM, but not traditionally a goal
- Exo-Kernels
 - MIT
 - Go beyond the multiplexing of the hardware, but looks at the operating system support
 - "making the kernel just another application library"
 - Arguably much more efficient than a non-extensible OS – allows for application customization of resource management

CS 2510, cs.pitt.edu

Extensions

11