Extensible OS

CS 256/456

Dept. of Computer Science, University of Rochester

Extending/Customizing the OS

- The given OS functionalities are insufficient or too rigid for some applications
  - sometimes you want to add things into the OS (filter out network packets from bad people) – extensibility
  - sometimes you don't want to use LRU memory management or you don't like the I/O prefetching size – customizability
- Does microkernel provide these features?
- Other ways to extend/customize the OS?

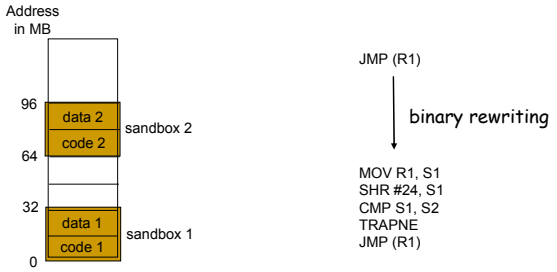
Downloading Code into Kernel

- Why downloading code into kernel?
  - extending the OS functionality
    - adding a new memory management policy
  - reduce kernel/user crossings, especially useful for small but frequent operations
    - packet filtering, selective event tracing, …
- Problem
  - How to protect the kernel from buggy/malicious user code?

Code Download Protection

- How to protect the kernel from buggy/malicious code downloading?
  - multiple address spaces in kernel
  - language and compiler support - type-safe language (disallow unsafe types, enforce array boundary checks)
  - sandboxing

Sandboxing



Memory divided into 16-MB sandboxes

- program can't jump out of its code sandbox
- program can't access data out of its data sandbox

Enforcement: runtime interpretation or binary rewriting

The Graybox Approach

- The OS is too complex
  - you don't want to mess around with it
  - adding an information/control layer to extend/customize the OS [Arpaci-Dusseau et al. SOSP2001/SOSP2003]
    - Information: tries to learn the OS with some basic knowledge
    - Control: tries to manipulate it through the given OS interface

Applications
Information/control
The OS

- It is called the gray-box approach because it is neither black-box nor transparent-box

Examples of Information/Control

Applications
Information/control
The OS

- Information
  - How do you learn what memory management policy is employed without looking at the code? Assume that you already know it must be one of several possibilities (e.g., LRU, MRU, or FIFO).
  - How do you learn the amount of free memory in system?
- Control
  - How to achieve FIFO on an OS that supports LRU memory management?

Configurable OS

- choosing parameter setting for each application
  - memory management policy
  - I/O prefetching size
  - many others …
- less ambitious than previous approaches – supporting customization, but not extension
- support multiple applications with distinct configuration settings
  - no new code in kernel
  - any challenge for this?

Virtual Machine

- Virtual machine can also be used to customize the OS for applications
  - one VM for each application
  - application run with its own customized OS
- Overhead
  - virtualization overhead
  - additional memory consumption
  - additional disk space consumption

Comparisons on OS Extension/Customization

- Approaches
  - Exokernel
  - Downloading code into the kernel
  - The graybox approach
  - Configurable OS
  - Virtual machine
- Comparison on
  - flexibility
  - how much changes in the original OS?
  - overall simplicity
  - overhead of each customization/extension