Extensible OS

CS 256/456

Dept. of Computer Science, University of Rochester

Microkernel

- Microkernel structure:
  - Moves functionalities from the kernel into "*user*" space.
- Benefits:
  - Modular design
  - More reliable/secure (less code is running in kernel mode)
- Disadvantage on performance:
  - Tend to have more frequent domain crossings.
- Two types of micro-kernels:
  - Running user-level OS in a trusted server – Mach
  - Running user-level OS within untrusted user processes – Exokernel
    - more flexibility, but problems on security and isolation

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 (record access traces for all disk I/Os) – 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?
- We will look at some other ways to extend/customize the OS.

Downloading Code into Kernel

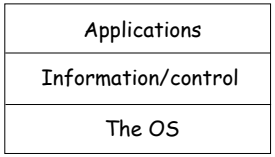
- Why downloading code into kernel?
  - extending the OS functionality
    - adding a new memory management policy
  - reduce kernel/user crossings, especially useful for small but frequent operations
    - system call interceptor, packet filtering, …
- Problem
  - How to protect the kernel from buggy/malicious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

Code Download Protection

- How to protect the kernel from buggy/malicious code downloading?
  - compiler support - type-safe language (disallow unsafe types, enforce array boundary checks)
  - sandboxing – binary rewriting
  - are we safe with those?

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



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

An Example: Information/Control on Memory Management

- Information
  - how do you learn what memory management policy is employed without looking at the code?
  - you already know it must one of several possibilities (LRU, MRU, FIFO, LFU, …)
- Control
  - how to achieve FIFO on an OS that supports LRU memory management?

Configurable OS

- choosing parameter setting for each application
  - I/O prefetching size
  - memory management policy
  - anything else?
- less ambitious than previous approaches – supporting customization, but not extension
- must support multiple applications with distinct configuration settings
  - any challenges 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

Comparisons on OS Extension/Customization

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