

CS510 Languages and Low Level Programming: Portfolio submission, Topic 6

Due on May 13, 2016 at 11:59pm

Mark P. Jones Spring 2016

Konstantin Macarenco

Topic 6.

Describe the motivation, implementation, and application of microkernel abstractions for managing address spaces, threads, and interprocess communication (IPC).

Motivation

Idea behind microkernel is to provide “policy free” environment, with no assumptions about services it will be running. All functionality of a traditional kernel is moved out, and implemented as a set of independent services. Microkernel provides only minimum set of abstraction over hardware, such as

- Address Spacing
- Inter-process Communication (IPC)
- Thread Management
- Unique Identifiers

Systems that use this approach will implement only the functionality needs to operate, have smaller footprint. All nonessential for the kernel functions, like network support, are implemented as services. Each service has its own address space, communication with other services allowed only through Only kernel functionality is in privileged mode (ring 0).

Advantages

- Robustness, a service crash doesn't affect any other part of the system.
- Easier maintenance.
- Enforces modular structure.
- Small size of Trusted Computing Base.

Disadvantages

- Performance loss due to high overhead of services interaction.
- Larger memory footprint.
- Complicated process management.

Implementation

History

Mach

L3

L4

Implementation details

- Userspace perspective
 - Kernel Information Page (kernel version, host system info, address space layout info, system call entry points)
 - Predefined KIP address
 - Slow system call LOCK NOP -i illegal opcode exception -i kernel -i load KIP address to context registers -i return to user mode
 - and User Thread Control Block (One UTCB for each thread) in the address space Message registers (MRs), and thread control registers (TCRs)
 - all UTCBs are in UTCB area

64 message registers $MR_0 - MR_{63}$

Miscellaneous fields: error code, exception handler pager acceptor

UTCB address points to the middle of the UTCB

User processes can read/write data to any UTCB available

thread priority is stored in protected structure, any data read from UTCB cannot be trusted and must be validated by the kernel before use.

UTCB mappings created by kernel

Application