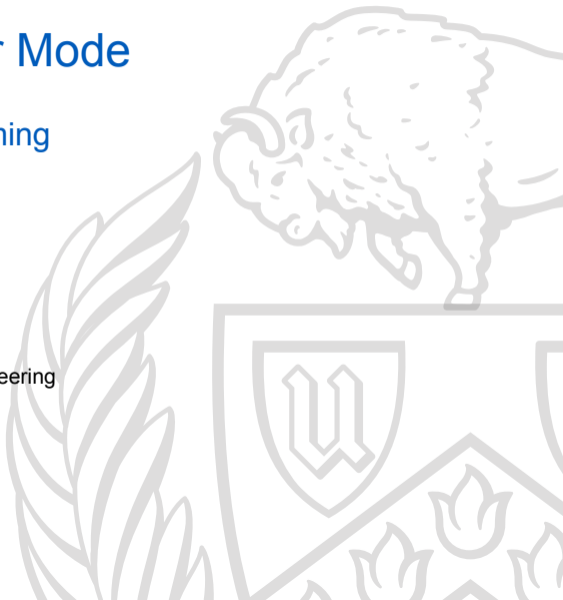


The Kernel and User Mode

CSE 220: Systems Programming

Ethan Blanton

Department of Computer Science and Engineering
University at Buffalo



The Operating System Kernel

We have talked about **the operating system** or **the kernel**.

The operating system **manages the hardware**.

On our systems, it also:

- Supports the **dedicated computer** model
- Provides **protection** against misbehaving programs

The kernel is **the code of the inner core of the OS**.

In some sense the OS and Kernel are **just programs**.

User Mode Programs

Our programs run in **user mode**.

User mode programs **appear** to run on a **dedicated computer**.

This means that **shared resources** must be managed for them.

User mode programs **ask the kernel** for access to shared resources.

If the user mode program has a dedicated computer ... **where is the kernel?**

Exceptions

Exceptions are another type of **control flow**.

Unlike *if*, *for*, *etc.*, they:

- Allow **non-local** (to another function or even program) transfer of control
- Can be **asynchronous** (triggered by an external event)

Exceptions may be caused by **hardware** or **software**.

The handling of exceptions requires both.

System Calls

A **system call** is a special kind of exception.

It allows a program to:

- “break out” of its **dedicated computer**, and
- **contact the kernel**

System calls are **synchronous** but **non-local** transfer of control.

The Kernel and Supervisor Mode

The kernel **does not have** a dedicated computer. ¶

The kernel has the **real computer!**

It runs in a special mode (often called **supervisor mode**).

It can:

- Access hardware **directly**
- Manipulate virtual memory mappings
- Modify process memory
- ...

Protection Domains

This supervisor mode is a different **protection domain**.

Protection domains are a **hardware capability**.

User programs run in **user mode**, the kernel in **supervisor mode**.

The hardware **enforces access restrictions** on user mode.

Some hardware has **more than two** protection domains.

Changing Protection Domains

Changing protection domains is a **supervisor mode operation**.

This programs from **breaking out** of user mode.

It also means there must be a **safe way** to switch!

We will see how **exceptions** provide a controlled mode change.

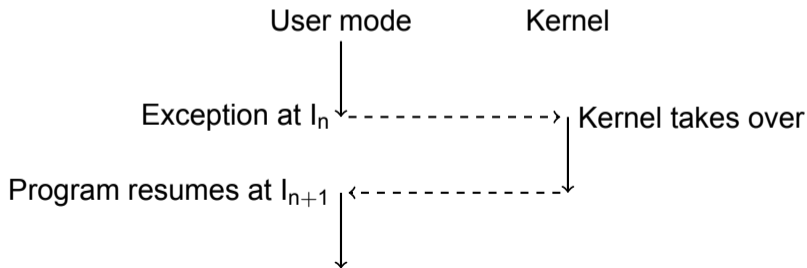
Changing protection domains can be **slow** and **expensive**.

Exception Flow

When an exception occurs, control passes to the kernel.

If control is already in the kernel, it changes location.

If control is in a user mode program, it switches contexts.



Types of Exceptions

There are four major types of exceptions:

- **Interrupts** are **asynchronous** notifications from hardware
- **Traps** are **synchronous** exceptions caused by software intentionally
- **Faults** are **synchronous** exceptions caused by software **due to potentially recoverable errors**
- **Aborts** are **synchronous** exceptions caused by **unrecoverable errors outside of software control**

We have only seen **faults** thus far (remember page faults?).

We are currently most interested in **traps**.

Interrupts

Interrupts are a way for hardware to signal the OS.

Examples:

- A network packet has arrived
- A clock has “ticked”
- A disk has completed a read

Interrupts are handled by the kernel.

Faults

We have already seen faults!

Segmentation fault (core dumped)

Faults are **program errors** that **may be recoverable**.

When a fault occurs, the kernel may:

- try to fix it
- notify the program

It may also **terminate the program** or **shut down**.

Fault Recovery

Some faults are not true errors:
e.g., **page faults** to bring in new pages.

Other faults may be **recoverable by the program**:

- Divide by zero
- Segmentation fault
- Bus error
- ...

Each of these **is an error**, but might not be **fatal**.

For example, a calculation might usefully return **some concrete value** if it reaches a divide-by-zero.

Aborts

Aborts are **relatively uninteresting** to us.

They represent some **unrecoverable error** that often ends in:

- Rebooting the computer
- Shutting down the computer
- Terminating some or all processes
- *etc.*

Aborts are handled **by the kernel**.

Traps

Traps are **software-generated exceptions**.
(They are sometimes called software interrupts.)

They are generated by **special instructions** run by a program.

Their critical feature is:

Trap handlers are run **by the kernel in supervisor mode**.

This means that a **user mode program** can **call into the kernel**.

This provides a **safe method** of changing protection domains.

System Calls

System calls are:

- traps
- used by user-mode programs
- to invoke kernel functions

Many platforms have a dedicated hardware instruction for this:

- ARM: `svc`
- x86-64: `syscall`

System Call Handling

When the system call instruction runs, the hardware:

- Switches to **supervisor mode**
- Invokes a specific kernel routine

When the kernel receives a system call, it:

- Identifies **what the program wants**
- Verifies the program **arguments**
- Authenticates the request
- Performs the operation (or indicates failure)

This allows the **kernel to decide** whether a program can **access something outside its “dedicated computer”**.

The Implications of the Trap

The user mode program **cannot decide** what kernel code runs.

This is:

- **controlled** by the hardware
- **configured** by the kernel

This is how modern operating systems **protect themselves** from malicious or buggy programs.

Invoking a System call

We have **invoked system calls!**

`open()`, `sbrk()`, `mmap()`, *etc.* are **system calls!**

We never used the `syscall` instruction.

The **C library** makes system calls **look like a C function.**

All functions in manual section 2 are system calls.

Overhead

System calls **are very slow**.

They can take tens to **hundreds of thousands** of clock cycles.

This is due to:

- Changing protection domains
- Validating arguments
- Adjusting memory mappings
- Cache effects
- ...

Programs should **make fewer system calls** when practical.

Summary

- Exceptions are special control flow
- Protection domains control access to hardware resources
- Exception handlers run in supervisor mode in the kernel
- Special trap exceptions can be used to implement system calls
- System calls allow user mode programs to request access to the kernel

References I

Required Readings

- [1] Randal E. Bryant and David R. O'Hallaron. *Computer Science: A Programmer's Perspective*. Third Edition. Chapter 8: Intro, 8.1, 8.2. Pearson, 2016.

License

Copyright 2019– 2022 Ethan Blanton, All Rights Reserved.
Copyright 2019 Karthik Dantu, All Rights Reserved.

Reproduction of this material without written consent of the author is prohibited.

To retrieve a copy of this material, or related materials, see <https://www.cse.buffalo.edu/~eblanton/>.