Operating System support: Exceptional Control Flow Processes

Supporting shared, managed resources and interaction with the Outside.

Operating System
Manage and abstract hardware resources
- Simpler, consistent interface to varied hardware
- Share limited resources among multiple processes, users
- Protect co-resident processes and users from each other

A brief tour for 240:
- Exceptional control flow -- hardware support needed for OS functionality
- Process model -- illusion of single process running on simple system
- Virtual memory -- hardware support for process model

Control Flow

Exceptional control flow:
- Transfer control between processes and OS
- Handle I/O and virtual memory within the OS
- Deal with unexpected errors

Changed by:
- •
- •
- •
Exceptional Control Flow

Synchronous exceptions: caused by instruction
- **Traps**: like procedure call to OS
  - Intentional: transfer control to OS to perform some function
    - *system calls* (syscall), breakpoint traps, special instructions
  - Returns control to "next" instruction
- **Fauls**: unintentional, maybe recoverable
  - page faults, segment protection faults, divide by zero
  - Re-execute faulting ("current") instruction or abort process.
- **Aborts**: unintentional, unrecoverable
  - hardware failure detected, parity error, etc.

Asynchronous exceptions: caused outside program
- incoming I/O activity, reset button, timers

Exceptions

An **exception** is transfer of control to the operating system (OS) in response to some event.

**Traps**

An exception is transfer of control to the operating system (OS) in response to some event.

**How does the system know where to jump in the OS?**

Interrupt Vector

A jump table for exceptions

- Exception numbers
- Exception Table
  - code for exception handler 0
  - code for exception handler 1
  - code for exception handler 2
  - ...
Process Model

A process is an instance of a running program.

Process != program

Two key abstractions:

- Logical control flow
  - Each process seems to have exclusive use of the CPU
- Private virtual address space
  - Each process seems to have exclusive use of main memory

Why are these illusions important?

How are these illusions maintained?

- Interleaved execution (multi-tasking)
- Virtual memory

Concurrent Processes

Two processes run concurrently (are concurrent) if their executions overlap in time.

Otherwise, they are sequential.

Examples:

- Concurrent: A & B, A & C
- Sequential: B & C

User View of Concurrent Processes

Control flows for concurrent processes are physically disjoint in time.

CPU only executes instructions for one process at a time.

However, we can think of concurrent processes as executing in parallel.

On today’s machines, with multiple CPU cores, we get some true parallel process execution.

Context Switching

Processes are managed by OS code called the kernel.

- Not a separate process, runs as part of a user process.
- Control flow passes between processes via context switch.