Operating Systems
Mid-Semester Survey
Operating Systems

- Processor
- Registers
- L1 Cache
- L2 Cache
- Main Memory
- Disk
Operating Systems

• Unix
  • Berkeley Software Distribution (BSD)
  • macOS

• Linux distribution
  • Debian
  • Ubuntu
  • Red Hat

• Microsoft Windows
What is an Operating System

- Illusionist
  - Provide clean, easy-to-use abstractions of physical resources
What is an Operating System

- **Illusionist**
  - Provide clean, easy-to-use abstractions of physical resources

![Diagram of Operating System and Processes](image)
What is an Operating System

- Referee
  - Manage protection, isolation, and sharing of resources

### Diagram

- **Hardware**
  - Processor
  - Memory (L1$, L2$)
  - Disk

- **Operating System**
  - Process 1
    - Threads
    - Address Spaces
    - Files
    - Sockets
  - Process 2
    - Threads
    - Address Spaces
    - Files
    - Sockets

- **Compiled Program 1**
- **Compiled Program 2**
What is an Operating System

- Referee
  - Manage protection, isolation, and sharing of resources
What is an Operating System

• Referee
  • Manage protection, isolation, and sharing of resources
Context Switch

- Context switch: Switching from executing one program to another
- Allows multiple processes to run on the same processor
- The OS determines when to context switch
What happens on a context switch?

1. The OS takes control of the CPU from the current process
2. The OS saves the state of the current process
3. The OS loads the state of the next process
4. The OS hands over the CPU to the next process
What is an Operating System

- **Referee**
  - Manage protection, isolation, and sharing of resources

![Diagram showing components of an operating system and processes](image-url)
What is an Operating System

- Referee
  - Manage protection, isolation, and sharing of resources
Protection

- OS isolates processes from each other
- OS isolates itself from other processes
- … even though they are actually running on the same hardware!
Dual Mode Operation

- Hardware provides at least two modes:
  1. Kernel Mode (or “supervisor” mode)
  2. User Mode

- Certain operations are prohibited when running in user mode
  - interacting directly w/ hardware, writing to kernel memory

- OS mostly runs in user mode

- Switching between user mode and kernel mode
  - System calls, interrupts, exceptions
System Calls (syscall)

- Allows the program to request a service from the operating system

Examples

- creating and deleting files
- reading and writing files
- accessing external devices like a scanner
- (ecalls in RISC-V)

Similar to function calls except it’s executed by the kernel
System Calls

- Single instruction
- Series of Instructions

![Diagram of system calls]

- User mode
- Kernel mode
- User program
- System call instruction
- Execute system call
Interrupts vs Exceptions

- **Interrupts**
  - Caused by an event *external* to the current running program
  - Ex: Key press
  - Asynchronous to the current program
    - Does not need to be handled immediately, but should be handled soon

- **Exceptions**
  - Caused by an event *during* the execution of the current program
  - Ex: illegal instruction, divide by zero
  - Synchronous
    - Must be handled immediately
Exceptions

- PC address Exception
- Illegal Opcode
- Data address Exceptions
How to Handle Interrupts and Exceptions?

• Trap Handler: code that services the interrupt or exception
• From the program’s point of view, it must look like nothing happened
Traps

1. All instructions before the faulting instructions must complete
2. All instructions after to the faulting instruction must be flushed
3. The faulting instruction must be flushed
4. Execution of the trap handler begins
What does the Trap Handler Do?

1. Save the state of the current program
   - Save ALL of the registers

2. Determine what caused the exception or interrupt

3. Handle exception or interrupt

   - continue execution of the program
   - terminate the program

4. Restore the state of the program

5. Return control to the program

4. Terminate the program (free resources, etc)

5. Schedule a new program
Traps

The program continues at the instruction that caused the exception.

The program continues at the instruction that caused the exception.

\( i_a \) is the instruction that caused the exception.

Program

- \( i_{a-1} \)
- \( i_a \)
- \( i_{a+1} \)

Handler

- \( i_0 \)
- \( i_1 \)
- \( i_n \)

continue execution of the program

\( i_a \) is the instruction that caused the exception.
Traps

Program

\[ i_{a-1} \]
\[ \downarrow \]
\[ i_a \]
\[ \downarrow \]
\[ i_{a+1} \]

Handler

\[ i_0 \]
\[ \downarrow \]
\[ i_1 \]
\[ \downarrow \]
\[ i_n \]

Program

\[ i_0 \]
\[ \downarrow \]
\[ i_1 \]
\[ \downarrow \]
\[ i_n \]

continue execution of the program

terminate the program
Which path to choose?

• Continue execution of the program
  • Interrupts (most likely)
  • Certain memory exceptions (we’ll see more later)

• Terminate program
  • Illegal instruction
  • Certain illegal memory accesses
Program’s Point of View

• Almost like nothing ever happened
  • The program state did not change
• Maybe large gap in between one instruction and the next
• Caches may have been trashed
  • Because something else was using them
Kernel

- Core of the OS
- Manages resources
  - scheduling, memory, I/O
- Things not in the kernel
  - User interface
  - Networking
- Lots of variation between different operating systems
Shell

• A program that exposes the operating system’s services
What happens at Boot?

1. The BIOS (Basic Input/Output System) runs
   • Power-on-self-test (POST)
   • The BIOS finds and executes the bootloader

2. The bootloader loads in part of the operating system

3. The operating system initializes services, drivers, etc

4. Launch a process that waits for an input in a loop

Bootstrapping: A chain of stages, in which at each stage, a smaller, simpler program loads and then executes the larger, more complicated program of the next stage (Wikipedia)
How do you begin executing a program?

• Loader: responsible for loading programs into memory
  1. The loader loads program into memory
  2. The loader sets \texttt{argc} and \texttt{argv}
  3. The OS jumps to \texttt{main} and transfers control to the process
What is an Operating System?

• Provides *isolation* between running processes
  • Each program runs in its own world

• Provides *interaction* with the outside world
  • interact with devices like mouse, display, network
What is an Operating System

• Illusionist
  • Provide clean, easy-to-use abstractions of physical resources
    • Masks limitations
    • Higher level objects: files, sockets

• Referee
  • Manage protection, isolation, and sharing of resources
    • Resource allocation and communication
Coming up…

- Virtual memory
  - How the OS isolates processes
- I/O
  - How the OS communicates with the outside world