SE 350: Operating Systems

Michael Noukhovitch

Winter 2015, University of Waterloo

Notes written from Thomas Reidemeister's lectures.

Contents

1 Introduction

1.1 Definitions

Operating Systems: a standardized abstraction from hardware that:

- manages resources
- provides set of services
- consumes resources

Instruction execution:

1. fetches instruction into IR
2. executes instruction

In reality, it is a bit more complicated: there is a pipeline, out of order execution...

1.2 Interrupts

1.2.1 Type of interrupts

- **program** result of instruction execution e.g. arithmetic overflow
- **timer** timer within process, allows OS to perform regular functions
- **I/O** generated by I/O controller
- **hardware failure** power failure or memory parity failure

1.2.2 How interrupts work

Hardware:

1. interrupt issued
2. processor finishes current instruction
3. acknowledge interrupt
4. push PSW and PC onto control stack
5. load new PC

Software:

6. save remainder of process state
7. interrupt
8. restore process state information
9. restore PSW and PC

1.3 Multiple interrupts

Two ways of handling an interrupt during an interrupt:

1.3.1 Sequential

Ignore any interrupts when you are in an interrupt and when done, check for interrupts that occurred.

1.3.2 Nested

If the second interrupt is of higher priority, recurse into it. Otherwise, wait until interrupt is finished.

1.4 Peripheral interrupts

Peripherals such as hard drives take a while to complete their action. As opposed to waiting and wasting that time, we use an interrupt to execute other instructions while our I/O process runs.

1.4.1 Programmed I/O

No interrupts occur, just wait until the I/O is complete

1.4.2 Interrupt-driven I/O

Processor interrupted when I/O is ready and all writes/reads are passed through CPU into memory. This is faster, since there is no waiting.

1.4.3 Direct memory access

Transfers a block of data directly into memory and interrupt is sent when process is complete. This is more efficient because data does not need to go through CPU. Not always available (e.g. external peripheral)

1.5 Memory hierarchy

Major constraints in memory:

- size
- speed
- cost

1.5.1 Hierarchy

Top: Inboard memory
Middle: Outboard storage
Bottom: Offline storage

1.5.2 Cache

Cache: small, fast memory, invisible to the OS, that speeds up accesses exploiting the principle of locality

2 Operating Systems Overview

2.1 Definition

Operating System a program that controls the execution of applications and is a standard interface between hardware and software

- **convenience**: need no knowledge of hardware
- **efficiency**: move optimization from devs to tools
- **ability to evolve**: can replace internals

Kernel: portion of the OS in main memory; a nucleus that contains frequently used functions **OS services**

- program development and execution
- access & control of access to I/O
- system access control
- error detection and response
- accounting

2.2 OS Innovations

2.2.1 Hardware Features

- **Memory protection**: do not allow memory containing monitor to be altered
- **Timer**: prevents a job from monopolizing system
- **Privileged instruction**: certain instructions can only be executed by the monitor (e.g. I/O)
- **Interrupts**

2.2.2 Modes of operation

To protect users from each other (and the kernel from user), we have two modes:

- **User mode**: not privileged
- **Kernel mode**: privileged and access to protected memory

2.2.3 Multiprogramming

When one job needs to wait for I/O, the processor can switch to another job. The timer is also used to switch processes and stop monopolization.

- maximize processor use
- use job control language

2.2.4 Time Sharing

Processor time is shared by multiple users

- minimize response time

2.3 Major Achievements

2.3.1 Processes

Process: a program in execution

- a program
- associated data
- execution content (needed by OS)

2.3.2 Memory Management

- process isolation
- automatic allocation
  - virtual memory: allows programmer to address memory without regard to physical addressing
  - paging: allows processes to be comprised of fixed-size blocks (pages)
- swap program code
- shared memory
- long term storage

2.3.3 Information Security

- availability: protecting system against interruption (downtime)
- confidentiality: authorizing data (chmod)
- data integrity: protect from modification
- authenticity: verifying identity of users

2.3.4 Scheduling and Resource Management

- fairness: give equal access to resources
- differential responsiveness: discriminate by class of jobs
- efficiency: maximize throughput

2.3.5 System Structure

The system as a hierarchical structure with each level relying on lower levels for its functions.

1. circuits: registers, gates, buffers
2. instructions: add, subtract, load, store
3. procedures: call stack, subroutine
4. interrupts
5. processes: suspend, wait, resume
6. local store: blocks of data, allocate
7. virtual memory: segments, pages
8. communications: pipes
9. file system: files
10. external devices
11. directories
12. user process
13. shell

2.4 Modern Operating System

Developments leading to modern operating systems:

- **Microkernel architecture**: only essentials to kernel, everything else in user space (e.g. QNX)
- **Multithreading**: process divided into concurrent threads
- **Symmetric multiprocessing**: multiple processors share main memory and I/O
- **Distributed OS**: illusion of a single main and secondary memory
- **Asymmetric multiprocessing**: one big processor controls many small ones
- **Object oriented design**: customize OS without disrupting system