操作系统原理

张竞慧

办公室：计算机楼366室
电邮：jhzhang@seu.edu.cn
主页：http://cse.seu.edu.cn/PersonalPage/zjh/
电话：025-52091017
Very Simple Questions about Operating System

- What Operating System are you familiar with?

- What is the best Operating System in your opinion? And for what features?
Ken Thompson & Dennis Ritchie
Linus Torvalds
OS Archaeology

- Multics → AT&T Unix → BSD Unix → Ultrix, SunOS, NetBSD,…

- Mach (micro-kernel) + BSD → NextStep → XNU → Apple OS X, iphone iOS

- Linux → Android OS
- Linux → RedHat, Ubuntu, Fedora, Debian, Suse,…

- CP/M → QDOS → MS-DOS → Windows 3.1 → NT → 95 → 98 → 2000 → XP → Vista → 7 → 8 → phone → …
An Motivating Example about OS: GUI and X Windows

- the X server receives input from a local keyboard and mouse and displays to a screen
- A web browser and a terminal emulator run on the user's workstation
- A terminal emulator runs on a remote computer but is controlled and monitored from the user's machine

We Have the Demo!
Another Motivating Example about OS: eCos/Linux usage in AMS-02

- The Embedded Configurable Operating System (eCos) is a free and open source real-time operating system

- eCos is intended for embedded systems and applications which need only one process with multiple threads

- It is designed to be customizable to precise application requirements of run-time performance and hardware needs like in AMS-02

We Have the Story!
AMS 在奋进号舱内
2011年5月19日5:15安装完毕，9:35开始接收数据
eCos usage in JMDC Computers
AMS Data Flow

马歇尔宇航中心 MSFC

数据中继卫星

国际空间站上的AMS

美国国旗

欧洲核子中心

东南大学
国家航空航天实验室

监控中心
AMS飞行状态监测
数据中心
数据分析
数据存储
data服务中心

监控中心
AMS飞行状态监测和控制
数据中心
数据重建
数据分发和协调

数据中心
蒙特卡洛数据生产
数据分析
数据存储
data服务
AMS Payload and Operation Control Center (CERN, Geneva)
Linux usage in SEU HPC Center
Course Goals

- This course is intended to introduce the fundamental concepts that modern operating systems use to manage the physical computer. Students will understand the core of the operating system, known as the kernel.

- Prerequisites: C/C++, Computer Organization
Course Textbooks

- 教材：操作系统概念（第七版 影印版） 西尔伯莎茨编著 高教社 2007
- 参考书：现代操作系统（第三版）Andrew S. Tanenbaum 机械工业出版社
Course Materials for Reference

- UC Berkeley OS Course
  - [https://inst.eecs.berkeley.edu/~cs162/sp15/](https://inst.eecs.berkeley.edu/~cs162/sp15/)

- Stanford OS Course
Course Requirements

- **Grading (考核方式)**
  - 闭卷考试
  - 成绩评定：由下述三部分组成 *(Percentage TBD)*
    - 期末考试成绩
    - 期中考试成绩
    - 平时成绩（作业、出勤等）

- **Course Schedule:** 64学时
  
  [1-16周] 周三(3-4) 教七-30A，周五(3-4) 教七-30A

- **Office Hours:**
  - Tuesday: 14:00~17:00
  - *(E-mail discussions are welcome: jhzhang@seu.edu.cn)*
## Course Outline (1/2)

<table>
<thead>
<tr>
<th>教学内容</th>
<th>教学要求</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>掌握操作系统的一般情况，了解本课程的学习方法</td>
</tr>
<tr>
<td>Operating system Structures</td>
<td>介绍操作系统的组成&lt;br&gt;使学生树立操作系统的整体概念</td>
</tr>
<tr>
<td>Process Management</td>
<td>操作系统的最重要的部分，要求学生牢固掌握进程概念，&lt;br&gt;能够借助进程概念编写并发程序</td>
</tr>
<tr>
<td>Threads</td>
<td>要求学生掌握创建和管理线程的方法&lt;br&gt;熟练使用多线程技术编程</td>
</tr>
<tr>
<td>CPU Scheduling</td>
<td>要求学生理解并能编写一个进程调度的程序</td>
</tr>
<tr>
<td>Process Synchronization</td>
<td>要求学生理解同步概念，掌握同步的编程方法</td>
</tr>
<tr>
<td>Deadlocks</td>
<td>要求学生理解死锁现象，了解预防、避免、检测、解除死锁的方法</td>
</tr>
</tbody>
</table>
## Course Outline (2/2)

<table>
<thead>
<tr>
<th>教学内容</th>
<th>教学要求</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Memory Management</strong></td>
<td>掌握固定分区, 可变分区, 段式, 页式存储管理原理与算法</td>
</tr>
<tr>
<td><strong>Virtual Memory</strong></td>
<td>虚拟存储是操作系统另一个重要概念。要求学生正确的理解虚存的概念。熟练掌握请求页式的地址变换过程以及常用的页面置换算法</td>
</tr>
<tr>
<td><strong>File Systems</strong></td>
<td>掌握文件系统的基本概念，熟练使用文件系统的操作方法</td>
</tr>
<tr>
<td><strong>I/O Systems</strong></td>
<td>掌握I/O系统的基本概念，理解设备驱动程序。弄清从用户请求到设备完成用户请求的全过程</td>
</tr>
<tr>
<td><strong>Mass-Storage Structure</strong></td>
<td>了解几种磁盘调度算法</td>
</tr>
</tbody>
</table>
Chapter 1 (Part 1)
Introduction to Computer System

张竞慧
办公室：计算机楼366室
电邮：jhzhang@seu.edu.cn
主页：http://cse.seu.edu.cn/PersonalPage/zjh/
电话：025-52091017
Computer System Organization

- A Computer’s Basic Elements
- Processor Registers
- Instruction Execution
- Interrupts
- The Memory Hierarchy
- Cache Memory
- I/O Communication Techniques
A Computer’s Basic Elements

- Processor
- Main Memory
- I/O Modules
- System Bus

Figure 1.1 Computer Components: Top-Level View

PC = Program counter
IR = Instruction register
MAR = Memory address register
MBR = Memory buffer register
I/O AR = Input/output address register
I/O BR = Input/output buffer register
Processor

- Controls operation
- Performs data processing
Main Memory

■ Volatile
  ◆ Data is typically lost when power is removed

■ Referred to as real memory or primary memory

■ Consists of a set of locations defined by sequentially numbers addresses
  ◆ Containing either data or instructions
I/O Modules

- Moves data between the computer and the external environment such as:
  - Storage (e.g. hard drive)
  - Communications equipment
  - Terminals

- Specified by an I/O Address Register
System Bus

- Communication among processors, main memory, and I/O modules
Computer System Organization

- Basic Elements
- Processor Registers
- Instruction Execution
- Interrupts
- The Memory Hierarchy
- Cache Memory
- I/O Communication Techniques
Processor Registers

- Faster and smaller than main memory
- User-visible registers
  - Enable programmer to minimize main memory references by optimizing register use
- Control and status registers
  - Used by processor to control operating of the processor
  - Used by privileged OS routines to control the execution of programs
Data and Address Registers

- Data
  - Often general purpose
  - But some restrictions may apply

- Address
  - Stack pointer
  - Index Register
  - Segment pointer
Control and Status Registers

- Program counter (PC)
  - Contains the address of an instruction to be fetched

- Instruction register (IR)
  - Contains the instruction most recently fetched

- Program status word (PSW)
  - Contains status information
Instruction Register

- Fetched instruction loaded into instruction register

- Categories of instruction
  - Processor-memory: e.g., MOV R0 4
  - processor-I/O: e.g., IN R0 4
  - Data processing: ADD R0 4
  - Control: TEST R0/ JMP 4
Computer System Organization

- Basic Elements
- Processor Registers
- Instruction Execution
- Interrupts
- The Memory Hierarchy
- Cache Memory
- I/O Communication Techniques
Instruction Execution

- A program consists of a set of instructions stored in memory

- Two steps
  - Processor reads (fetches) instructions from memory
  - Processor executes each instruction
Basic Instruction Cycle

Figure 1.2 Basic Instruction Cycle
Instruction Fetch and Execute

- The processor fetches the instruction from memory
- Program counter (PC) holds address of the instruction to be fetched next
  - PC is incremented after each fetch
Characteristics of a Hypothetical Machine

(a) Instruction format

(b) Integer format

Program counter (PC) = Address of instruction
Instruction register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers

0001 = Load AC from memory
0010 = Store AC to memory
0101 = Add to AC from memory

(d) Partial list of opcodes

Figure 1.3 Characteristics of a Hypothetical Machine
Example of Program Execution

### Fetch Stage

<table>
<thead>
<tr>
<th>Memory</th>
<th>CPU Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>300 PC 1940</td>
</tr>
<tr>
<td>301</td>
<td>5941 AC</td>
</tr>
<tr>
<td>302</td>
<td>2941 IR</td>
</tr>
<tr>
<td>940</td>
<td>0003</td>
</tr>
<tr>
<td>941</td>
<td>0002</td>
</tr>
</tbody>
</table>

**Step 1**

<table>
<thead>
<tr>
<th>Memory</th>
<th>CPU Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>301 PC 1940</td>
</tr>
<tr>
<td>301</td>
<td>5941 AC</td>
</tr>
<tr>
<td>302</td>
<td>2941 IR</td>
</tr>
<tr>
<td>940</td>
<td>0003</td>
</tr>
<tr>
<td>941</td>
<td>0002</td>
</tr>
</tbody>
</table>

**Step 2**

<table>
<thead>
<tr>
<th>Memory</th>
<th>CPU Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>302 PC 1940</td>
</tr>
<tr>
<td>301</td>
<td>5941 AC</td>
</tr>
<tr>
<td>302</td>
<td>2941 IR</td>
</tr>
<tr>
<td>940</td>
<td>0003</td>
</tr>
<tr>
<td>941</td>
<td>0002</td>
</tr>
</tbody>
</table>

**Step 3**

**Step 4**

<table>
<thead>
<tr>
<th>Memory</th>
<th>CPU Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>303 AC 1940</td>
</tr>
<tr>
<td>301</td>
<td>5941 AC</td>
</tr>
<tr>
<td>302</td>
<td>2941 IR</td>
</tr>
<tr>
<td>940</td>
<td>0003</td>
</tr>
<tr>
<td>941</td>
<td>0005</td>
</tr>
</tbody>
</table>

0001 = Load AC from memory  
0010 = Store AC to memory  
0101 = Add to AC from memory

**Step 5**

**Step 6**
Computer System Organization

- Basic Elements
- Processor Registers
- Instruction Execution
- Interrupts
- The Memory Hierarchy
- Cache Memory
- I/O Communication Techniques
Interrupts

- Interrupt the normal sequencing of the processor
- Provided to improve processor utilization
  - Most I/O devices are slower than the processor
  - Processor must pause to wait for device
Transfer of Control via Interrupts

Figure 1.6  Transfer of Control via Interrupts
Instruction Cycle with Interrupts
Simple Interrupt Processing

**Hardware**

- Device controller or other system hardware issues an interrupt
- Processor finishes execution of current instruction
- Processor signals acknowledgment of interrupt
- Processor pushes PSW and PC onto control stack
- Processor loads new PC value based on interrupt

**Software**

- Save remainder of process state information
- Process interrupt
- Restore process state information
- Restore old PSW and PC

Figure 1.10  Simple Interrupt Processing
Common Classes of Interrupts

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program</td>
<td>Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, and reference outside a user’s allowed memory space.</td>
</tr>
<tr>
<td>Timer</td>
<td>Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.</td>
</tr>
<tr>
<td>I/O</td>
<td>Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.</td>
</tr>
<tr>
<td>Hardware failure</td>
<td>Generated by a failure, such as power failure or memory parity error.</td>
</tr>
</tbody>
</table>
Changes in Memory and Registers for an Interrupt

(a) Interrupt occurs after instruction at location $N$

(b) Return from interrupt
Multiple Interrupts

- Suppose an interrupt occurs while another interrupt is being processed.
  - E.g. printing data being received via communications line.

- Two approaches:
  - Disable interrupts during interrupt processing
  - Use a priority scheme.
Sequential Interrupt Processing

(a) Sequential interrupt processing
Nested Interrupt Processing

(b) Nested interrupt processing
Computer System Organization

- Basic Elements
- Processor Registers
- Instruction Execution
- Interrupts
- The Memory Hierarchy
- Cache Memory
- I/O Communication Techniques
Memory Hierarchy

- Major constraints in memory
  - Amount
  - Speed
  - Expense

- Faster access time, greater cost per bit
- Greater capacity, smaller cost per bit
- Greater capacity, slower access speed
The Memory Hierarchy

- Going down the hierarchy
  - Decreasing cost per bit
  - Increasing capacity
  - Increasing access time
  - Decreasing frequency of access to the memory by the processor
Secondary Memory

- Auxiliary memory
- External
- Nonvolatile
- Used to store program and data files
IBM正要运送一台5MB的硬盘
(1965年)
Computer System Organization

- Basic Elements
- Processor Registers
- Instruction Execution
- Interrupts
- The Memory Hierarchy
- Cache Memory
- I/O Communication Techniques
Cache Memory

- Invisible to the OS
  - Interacts with other memory management hardware

- Processor must access memory at least once per instruction cycle
  - Processor speed faster than memory access speed

- Exploit the principle of locality with a small fast memory
Principal of Locality

- More details later but in short …
- Data which is required soon is often close to the current data
  - If data is referenced, then it’s neighbour might be needed soon.
  - e.g., Loop
Cache and Main Memory

Figure 1.16 Cache and Main Memory
Cache Principles

- Contains copy of a portion of main memory
- Processor first checks cache
  - If not found, block of memory read into cache
- Because of locality of reference, likely future memory references are in that block
Figure 1.17 Cache/Main-Memory Structure
Cache Read Operation

START

Receive address RA from CPU

Is block containing RA in cache?  Yes  No

RA - read address

Access main memory for block containing RA

Allocate cache slot for main memory block

Fetch RA word and deliver to CPU

DONE

Load main memory block into cache slot

Deliver RA word to CPU

Figure 1.18 Cache Read Operation
Cache Replacement Algorithm

- Chooses which block to replace when a new block is to be loaded into the cache.
- Ideally replacing a block that isn’t likely to be needed again
  - Impossible to guarantee
- Effective strategy is to replace a block that has been used less than others
  - Least Recently Used (LRU)
Computer System Organization

- Basic Elements
- Processor Registers
- Instruction Execution
- Interrupts
- The Memory Hierarchy
- Cache Memory
- I/O Communication Techniques
I/O Techniques

- When the processor encounters an instruction relating to I/O,
  - it executes that instruction by issuing a command to the appropriate I/O module.

- Three techniques are possible for I/O operations:
  - Programmed I/O
  - Interrupt-driven I/O
  - Direct memory access (DMA)
Programmed I/O

The I/O module performs the requested action

- Then sets the appropriate bits in the I/O status register
- But takes no further action to alert the processor.

As there are no interrupts, the processor must determine when the instruction is complete.
Programmed I/O Instruction Set

- Control
  - Used to activate and instruct device

- Status
  - Tests status conditions

- Transfer
  - Read/write between process register and device
Programmed I/O (1)

Steps in printing a string.

(a) User space

(b) Printed page

(c) Printed page

Kernel space

String to be printed

ABCD
EFGH

Next

ABCD
EFGH

AB
Programmed I/O (2)

```c
copy_from_user(buffer, p, count);
for (i = 0; i < count; i++) {
    while (*printer_status_reg != READY) ;
    *printer_data_register = p[i];
}
return_to_user();
/* p is the kernel buffer */
/* loop on every character */
/* loop until ready */
/* output one character */
```

Writing a string to the printer using programmed I/O.
Interrupt-Driven I/O

- Processor issues an I/O command to a module
  - and then goes on to do some other useful work.

- The I/O module will then interrupt the processor to request service when it is ready to exchange data with the processor.
Interrupt-Driven I/O

Writing a string to the printer using interrupt-driven I/O.

(a) Code executed at the time the print system call is made.

(b) Interrupt service procedure for the printer.

```c
copy_from_user(buffer, p, count);
enable_interrupts();
while (*printer_status_reg != READY) ;
*printer_data_register = p[0];
scheduler();

if (count == 0) {
    unblock_user();
} else {
    *printer_data_register = p[i];
    count = count - 1;
    i = i + 1;
}
acknowledge_interrupt();
return_from_interrupt();
```
Direct Memory Access

- Performed by a separate module on the system
- When needing to read/write processor issues a command to DMA module with:
  - Whether a read or write is requested
  - The address of the I/O device involved
  - The starting location in memory to read/write
  - The number of words to be read/written
Direct Memory Access

- I/O operation delegated to DMA module
- Processor only involved when beginning and ending transfer.
- Much more efficient.
I/O Using DMA

(a) Code executed when the print system call is made.
(b) Interrupt service procedure.

Printing a string using DMA.

```c
copy_from_user(buffer, p, count);
set_up_DMA_controller();
scheduler();
```

```c
acknowledge_interrupt();
unblock_user();
return_from_interrupt();
```