CS490 Windows Internals

Kenny Q. Zhu
Dept. of Computer Science
Shanghai Jiao Tong University
Kenny Zhu

Research Interests:

Programming Languages
Data processing
Concurrent programming

Database & Data mining
Information extraction
Knowledge discovery

Degrees: National University of Singapore (NUS)
Postdoc: Princeton University
Experiences: Microsoft Redmond, USA
Microsoft Research Asia
Faculty at SJTU since 2009
Administrative Info (I)

- **Lecturer:**
  - Kenny Zhu, SEIEE #03-524, kzhu@cs.sjtu.edu.cn
  - Office hours: by appointment or after class

- **Teaching Assistant:**
  - Jack Sun, SEIEE #03-341, jacksunwei@gmail.com
  - Office hours: Monday 4-6 PM

- **Textbook:** Windows Internals (5th ed.) By Mark E. Russinovich, et al.

- **Course Web Page (definitive source!):**
  http://www.cs.sjtu.edu.cn/~kzhu/cs490/
Administrative Info (II)

- **All-English Course:** everything in English!
- **Lectures:**
  - Course material + lab demo + quizzes
  - Lecture slides on course web site after class
- **Assignments and Labs:**
  - Released (usually) on Fridays
  - Assignments due on the following Friday
  - Submit *hard copies* to class or Jack’s office SEIEE #03-342
  - Late submission: -30% of full score for each additional day
- **Group Projects:**
  - 2 persons per group
Administrative Info (III)

- 3-credit course

Modes of Assessment:

- Quizzes: 20%
- Assignments: 40%
- 2 Group Projects: 40%

Email Jack the names and contact of your group by this weekend!
Course Overview

- Concepts and Tools
- Windows Structuring (Architecture)
- Core system mechanisms
- Concurrency and Windows Traps (interrupts/exceptions)
- Windows Synchronization
- Processes and thread, Scheduling
- Memory management
- I/O management
- File system
- Windows security (Pending)
Copyright Notice
© 2000-2005 David A. Solomon and Mark Russinovich

Much of the materials in this course are part of the *Windows Operating System Internals Curriculum Development Kit*, developed by David A. Solomon and Mark E. Russinovich with Andreas Polze

Microsoft has licensed these materials from David Solomon Expert Seminars, Inc. for distribution to academic organizations solely for use in academic environments (and not for commercial use)
Roadmap For This Lecture

- Overview of Operating Systems
- Main Concepts in Operating Systems
- Structure of Operating Systems
- Micro-kernels
- History of Windows
Operating Systems Concepts

- **OS is intermediary between user and computer hardware**
  - Ease of use (for the user)
  - Efficiency (for the hardware)
- **Layered architectures**

<table>
<thead>
<tr>
<th>Banking system</th>
<th>Airline reservation</th>
<th>Web browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compilers</td>
<td>Editors</td>
<td>Command interpreter</td>
</tr>
<tr>
<td>Operating system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Machine language</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Microprogramming</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical devices</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

User

Application programs

System programs

Hardware
History of operating systems

Batch processing

The elements of the basic IBM 1401 system are the 1401 Processing Unit, 1402 Card Read-Punch, and 1403 Printer.

Punching cards programming

Multiprocessing

Multiprocessing

Memory partitions

Job 3
Job 2
Job 1
OS
The Evolution of Operating System Functionality

- **Batch Job Processing**
  - Linkage of library routines to programs
  - Management of files, I/O devices, secondary storage

- **Multiprogramming**
  - Resource management and sharing for multiple programs: spooling
  - Quasi-simultaneous program execution
  - Single user

- **Multiuser/Timesharing Systems**
  - Management of multiple simultaneous users interconnected via terminals
  - Fair resource management: CPU scheduling, spooling, mutual exclusion

- **Real-Time Systems (process control systems)**
  - Management of time-critical processes
  - High requirements with respect to reliability and availability
Tasks of an Operating System

- Processor management - Scheduling
  - Fairness
  - Non-blocking behavior
  - Priorities
- Memory management
  - Virtual versus physical memory, memory hierarchy
  - Protection of competing/concurrent programs
- Storage management – File system
  - Access to external storage media
- Device management
  - Hiding of hardware dependencies
  - Management of concurrent accesses
- Batch processing
  - Definition of an execution order; throughput maximization
Kernel- and User Mode Programs

Typical functionality implemented in either mode:

Kernel Space:
- Privileged mode
- Strict assumptions about reliability/security of code
- Memory resident
  - CPU-, memory-, Input/Output management
  - Multiprocessor management, diagnosis, test
  - Parts of file system and of the networking interface

User Space:
- More flexible
- Simpler maintenance and debugging
  - Compiler, assembler, interpreter, linker/loader
  - File system management, telecommunication, network management
  - Editors, spreadsheets, user applications
## Layered Model of Operating System Concepts

<table>
<thead>
<tr>
<th>nr</th>
<th>name</th>
<th>typical objects</th>
<th>typical operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Integrated circuits</td>
<td>register, gate, bus</td>
<td>Nand, Nor, Exor</td>
</tr>
<tr>
<td>2</td>
<td>Machine language</td>
<td>instruction counter, ALU</td>
<td>Add, Move, Load, Store</td>
</tr>
<tr>
<td>3</td>
<td>Subroutine linkage</td>
<td>procedure block</td>
<td>Stack Call, JSR, RTS</td>
</tr>
<tr>
<td>4</td>
<td>Interrupts</td>
<td>interrupt handlers</td>
<td>Bus error, Reset</td>
</tr>
<tr>
<td>5</td>
<td>Simple processes</td>
<td>process, semaphore</td>
<td>wait, ready, execute</td>
</tr>
<tr>
<td>6</td>
<td>Local memory</td>
<td>data block, I/O channel</td>
<td>read, write, open, close</td>
</tr>
<tr>
<td>7</td>
<td>Virtual model</td>
<td>page, frame</td>
<td>read, write, swap</td>
</tr>
<tr>
<td>8</td>
<td>Process communication</td>
<td>channel (pipe), message</td>
<td>read, write, open</td>
</tr>
<tr>
<td>9</td>
<td>File management</td>
<td>files</td>
<td>read, write, open, copy</td>
</tr>
<tr>
<td>10</td>
<td>Device management</td>
<td>ext.memory, terminals</td>
<td>read, write</td>
</tr>
<tr>
<td>11</td>
<td>I/O data streams</td>
<td>data streams</td>
<td>open, close, read, write</td>
</tr>
<tr>
<td>12</td>
<td>User processes</td>
<td>user processes</td>
<td>login, logout, fork</td>
</tr>
<tr>
<td>13</td>
<td>Directory management</td>
<td>internal tables</td>
<td>create, delete, modify</td>
</tr>
<tr>
<td>14</td>
<td>Graphical user interface</td>
<td>window, menu, icon</td>
<td>OS system calls</td>
</tr>
</tbody>
</table>
## Layered Model of Operating System Concepts

<table>
<thead>
<tr>
<th>nr</th>
<th>name</th>
<th>typical objects</th>
<th>typical operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Integrated circuits</td>
<td>register, gate, bus</td>
<td>Nand, Nor, Exor</td>
</tr>
<tr>
<td>2</td>
<td>Machine language</td>
<td>instruction counter, ALU</td>
<td>Add, Move, Load, Store</td>
</tr>
<tr>
<td>3</td>
<td>Subroutine linkage</td>
<td>procedure block</td>
<td>Stack Call, JSR, RTS</td>
</tr>
<tr>
<td>4</td>
<td><strong>Interrupts</strong></td>
<td><strong>interrupt handlers</strong></td>
<td><strong>Bus error, Reset</strong></td>
</tr>
<tr>
<td>5</td>
<td>Simple processes</td>
<td>process, semaphore</td>
<td>wait, ready, execute</td>
</tr>
<tr>
<td>6</td>
<td><strong>Local memory</strong></td>
<td>data block, I/O channel</td>
<td>read, write, open, close</td>
</tr>
<tr>
<td>7</td>
<td><strong>Virtual model</strong></td>
<td>page, frame</td>
<td>read, write, swap</td>
</tr>
<tr>
<td>8</td>
<td><strong>Process communication</strong></td>
<td>channel (pipe), message</td>
<td>read, write, open</td>
</tr>
<tr>
<td>9</td>
<td><strong>File management</strong></td>
<td>files</td>
<td>read, write, open, copy</td>
</tr>
<tr>
<td>10</td>
<td>Device management</td>
<td>ext.memory, terminals</td>
<td>read, write</td>
</tr>
<tr>
<td>11</td>
<td>I/O data streams</td>
<td>data streams</td>
<td>open, close, read, write</td>
</tr>
<tr>
<td>12</td>
<td><strong>User processes</strong></td>
<td>user processes</td>
<td>login, logout, fork</td>
</tr>
<tr>
<td>13</td>
<td><strong>Directory management</strong></td>
<td>internal tables</td>
<td>create, delete, modify</td>
</tr>
<tr>
<td>14</td>
<td>Graphical user interface</td>
<td>window, menu, icon</td>
<td>OS system calls</td>
</tr>
</tbody>
</table>
OS acts as Extension of Hardware

- System view: layered model of OS
  - Implementation details on one layer are hidden from higher layers
- Same machine, different operating systems:
  - IBM PC: DOS, Linux, NeXTSTEP, Windows, SCO Unix
  - DEC VAX: VMS, Ultrix-32, 4.3 BSD UNIX
- Same OS, different machines: UNIX
  - PC (XENIX 286, APPLE A/UX)
  - Mac (OS X)
  - CRAY-Y/MP (UNICOS - AT&T Sys V)
  - IBM 360/370 (Amdahl UNIX UTS/580, IBM UNIX AIX/ESA)
  - Windows NT, XP, 2000, 2003, ...
    - Intel x86, Alpha, PowerPC, MIPS, Itanium
Operating Systems Evolution
(more at http://www.levnez.com)
Main Concepts: processes

- Processes is runtime context of a program
- Process table, core image
- Command interpreter, shell
- Child processes

- Scheduling, signals
- User identification, group identification
Main Concepts: Files

- Files, directories, root
- Path, working directory
- Protection, rwx bits
- File descriptor, handle
- Special files, I/O devices
- Block I/O, character I/O
- Standard input/output/error
- Pipes
Main concepts: system calls

- User programs access operating system services via system calls
- Parameter transmission via trap, register, stack
  \[ \text{count=\text{read(file, buffer, nbytes);} } \]
- 5 general classes of system calls:
  - Process control
  - File manipulation
  - Device manipulation
  - Information maintenance
  - Communications
Main concepts: shell

- Command interpreter
- Displays prompt, implements input/output redirection
- Background processes, job control, pseudo terminals

$ date

$ date >file

$ sort <file1 >file2

$ cat file1 file2 file3 > /dev/lp1

$ make all >log 2>&1 &

Note: 0: stdin, 1: stdout, 2: stderr
Monolithic systems

- Functionally *distinguishable* aspect not architecturally separate components, but all *interwoven*.
- Unstructured
- Supervisor/System call changes from user mode into kernel mode
Layered OS

Each layer is given access only to lower-level interfaces
Microkernel OS (Client/server OS)

Kernel implements:

- Scheduling
- Memory Management
- Interprocess communication (IPC)

User-mode servers
Is Windows a microkernel-based OS?
- No – not using the academic definition (OS components and drivers run in their own private address spaces, layered on a primitive microkernel)
- All kernel components live in a common shared address space
  - Therefore no protection between OS and drivers

Why not pure microkernel?
- Performance – separate address spaces would mean context switching to call basic OS services
- Most other commercial OSs (Unix, Linux, VMS etc.) have the same design

But it does have some attributes of a microkernel OS
- OS personalities running in user space as separate processes
- Kernel-mode components don't reach into one another’s data structures
  - Use formal interfaces to pass parameters and access and/or modify data structures
- Therefore the term “modified microkernel”
Mach Microkernel OS
Extended Memory Management

Paging handled by user-space server

Port: comm. endpoint, network-wide
Mach Microkernel OS
Distributed Shared Memory System

- Access remote memories, port access rights – ACL
- Kernel C attempts to write to mem of A and B.

Kernel C attempts to write to mem of A and B.

Kernel A

Kernel B

Pager

1. memory_object_data_request
   access=VM_PROT_WRITE

2. memory_object_lock_request
   should_flush=TRUE

3. memory_object_lock_completed

4. memory_object_data_provided
   lock_value=VM_PROT_NONE
Windows NT Origins

Design began in late 1988/early 1989 after Dave Cutler and a handful of Digital employees started at Microsoft

- Dave Cutler—legend in the operating system world
  - Project leader for Digital’s VMS (Virtual Memory System)
- Internally, Windows NT has many similarities to Digital’s VMS (scheduling, memory management, I/O and driver model)
- VMS+1=WNT just a coincidence

Original goal was replacement for OS/2

- Later goal changed to be the replacement for Windows 3.0
Window NT Origins (Cont’d)

The name “Windows NT” was chosen because

- NT stands for New Technology
  - But at a high level, the architecture and user interface are not really that “new”
    (as compared to most 32-bit OS’s)

- Also, the i860 RISC CPU NT was originally targeted at was code named N-Ten

Interesting book on the early years of NT:

- Show-stopper!: The Breakneck Race to Create Windows NT and the Next Generation at Microsoft
  - By G. Pascal Zachary, ISBN: 0029356717


- Where there are specific differences, these will be noted
VMS and Windows - a bird’s-eye view on architectures

Layered design for VAX/VMS operating system

<table>
<thead>
<tr>
<th>Level 1</th>
<th>Level 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layered Products (Apps)</td>
<td>Program Development Tools</td>
</tr>
<tr>
<td>Utilities</td>
<td>Support Libraries</td>
</tr>
<tr>
<td>Command Language Interpreter (CLI)</td>
<td>Supervisor</td>
</tr>
<tr>
<td>Supervisor</td>
<td></td>
</tr>
<tr>
<td>Record Management Service (RMS)</td>
<td>Executive</td>
</tr>
<tr>
<td>Executive</td>
<td></td>
</tr>
<tr>
<td>System services</td>
<td>Kernel</td>
</tr>
<tr>
<td>Kernel</td>
<td></td>
</tr>
<tr>
<td>Memory Management</td>
<td>I/O Subsystem</td>
</tr>
<tr>
<td>I/O Subsystem</td>
<td>Process and time management</td>
</tr>
<tr>
<td>Process and time management</td>
<td></td>
</tr>
<tr>
<td>System-wide data structures</td>
<td></td>
</tr>
<tr>
<td>Platform-Adaptation Layer (PAL) - Alpha</td>
<td></td>
</tr>
</tbody>
</table>

Windows high-level architecture
Release History

Although product name has varied, internally, each version identified by a “build number”

- Internal identification - increments each time NT is built from source (5-6 times a week)

Interesting timeline:

<table>
<thead>
<tr>
<th>Build#</th>
<th>Version</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>297</td>
<td>PDC developer release</td>
<td>Jul 1992</td>
</tr>
<tr>
<td>511</td>
<td>NT 3.1</td>
<td>Jul 1993</td>
</tr>
<tr>
<td>807</td>
<td>NT 3.5</td>
<td>Sep 1994</td>
</tr>
<tr>
<td>1057</td>
<td>NT 3.51</td>
<td>May 1995</td>
</tr>
<tr>
<td>1381</td>
<td>NT 4.0</td>
<td>Jul 1996</td>
</tr>
<tr>
<td>2195</td>
<td>Windows 2000 (NT 5.0)</td>
<td>Dec 1999</td>
</tr>
<tr>
<td>2600</td>
<td>Windows XP (NT 5.1)</td>
<td>Aug 2001</td>
</tr>
<tr>
<td>3790</td>
<td>Windows Server 2003 (NT 5.2)</td>
<td>Mar 2003</td>
</tr>
<tr>
<td>6000</td>
<td>Windows Vista (Longhorn)</td>
<td>Nov 2006</td>
</tr>
<tr>
<td>6001</td>
<td>Windows Server 2008</td>
<td>Feb 2008</td>
</tr>
<tr>
<td>7600</td>
<td>Windows 7</td>
<td>Jul 2009</td>
</tr>
</tbody>
</table>
Windows and Unix kernels are based on foundations developed in the mid-1970s.

(see http://www.levenez.com for diagrams showing history of Windows & Unix)
Further Reading

Dennis M. Ritchie, The Evolution of the Unix Time-sharing System,

David Donald Miller, OpenVMS Operating System Concepts,
  History of Digital Operating Systems (from pp. 447)

G. Pascal Zachary, Show Stopper! The Breakneck Race to Create Windows NT and the Next Generation at Microsoft,

Mark E. Russinovich, et al.,
  Windows Internals,
  Windows Versions (pp. 1)
Feedback?