Windows Kernel Internals II

Overview

*University of Tokyo – July 2004**

Dave Probert, Ph.D.

Advanced Operating Systems Group

Windows Core Operating Systems Division

Microsoft Corporation

Contributors

Neill Clift
Adrian Marinescu
Eric Li
Nar Ganapathy
Jake Oshins
Eric Traut
Dragos Sambotin
Arun Kishan
Brad Abrams

Brian Andrew
Ben Leis
Dan Lovinger

Course Overview

Four projects

- Writing kernel extensions for Windows
- Windows OS subsystems
- NTFS investigation using C#
- Monad (future)

Lecture topics

- Overview, kernel extensions, I/O drivers, WDF
- Object manager, LPC, Processes & Threads
- X86, VirtualPC, Advanced File Systems
- Longhorn, Monad, WinFX

Windows Overview

Current source base is Windows NT

- Foundation for NT4, Win2K, WinXP, WS03, Longhorn
- API is still Win32 – compatible with Win9x
- .NET Framework pushing out Win32 for Longhorn

Most applications written in VB or VC++ today

- Future is managed applications – marrying VB productivity with C++/Java richness => C#

Hot issues

- Trustworthy Computing
- Enable new computing experiences
- Create new business opportunities

Security Issues

Lots of legacy code now hooked to the internet

Most code written to work correctly under normal conditions

Security design issues are subtle, particularly w.r.t. DoS

Constantly evolving threats:

- Stack-buffer overruns, Heap overruns, Format string overruns
- One byte overruns, Integer overflows

Reliability Issues

- Reboots required to do just about anything
- Huge base of third party code, esp. drivers
- Hangs are hard to track and debug
- Patch management is tough
- Windows extension points poorly defined
  - Apps break other apps
  - Installation not idempotent
  - Uninstall doesn't return system to pre-install state
- Compatibility issues everytime OS changes
  - Apps test out the bugs in a particular OS environment
  - Apps hardcode OS version information
- Windows management particularly hard
  - Can't answer: what is the difference between 2 systems
  - Registry is too opaque and heavily abused
  - GUI-based management doesn't scale

Customer Experience

Establish tighter feedback loops

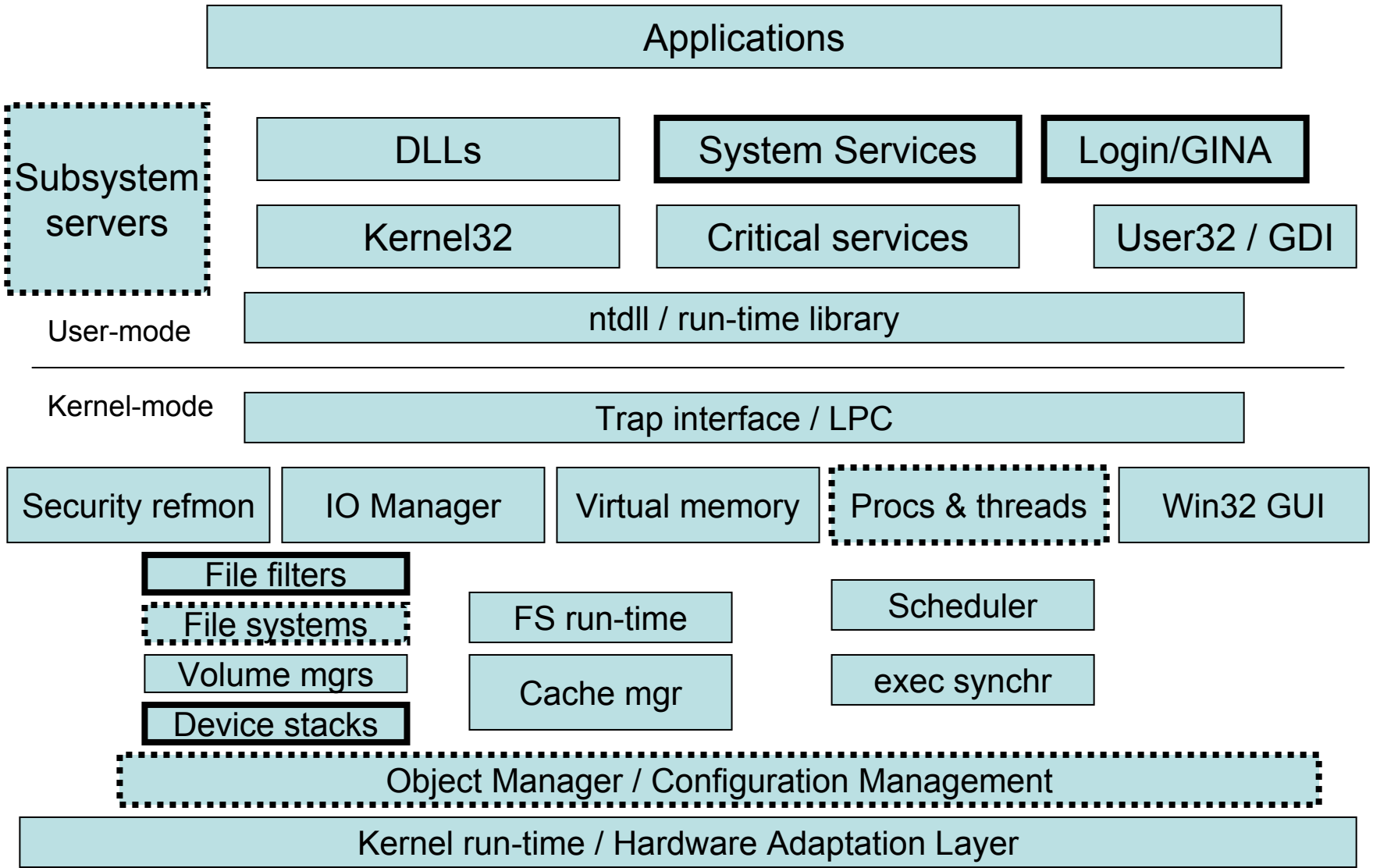
- WATSON – capture data on app crashes and hangs
- OCA – capture data on BSODs
- Windows Update and SUS – simplify patching of systems
- Enterprise tools – for deployment, event log analysis, helpdesk

Use collected data to

- prioritize fixes
- work with 3rd parties
- analyze common usage patterns
- improve future products

Feedback loops pioneered by Office

Windows Architecture



Windows Kernel Organization

Kernel-mode organized into

- NTOS (kernel-mode services)
  - Run-time Library, Scheduling, Executive services, object manager, services for I/O, memory, processes, …
- Hal (hardware-adaptation layer)
  - Insulates NTOS & drivers from hardware dependencies
  - Providers facilities, such as device access, timers, interrupt servicing, clocks, spinlocks
- Drivers
  - kernel extensions (primarily for device access)

Major Kernel Services

Process management

- Process/thread creation

Security reference monitor

- Access checks, token management

Memory manager

- Pagefaults, virtual address, physical frame, and pagefile management
- Services for sharing, copy-on-write, mapped files, GC support, large apps

Lightweight Procedure Call (LPC)

- Native transport for RPC and user-mode system services.

I/O manager (& plug-and-play & power)

- Maps user requests into IRP requests, configures/manages I/O devices, implements services for drivers

Cache manager

- Provides file-based caching for buffer file system I/O
- Built over the memory manager

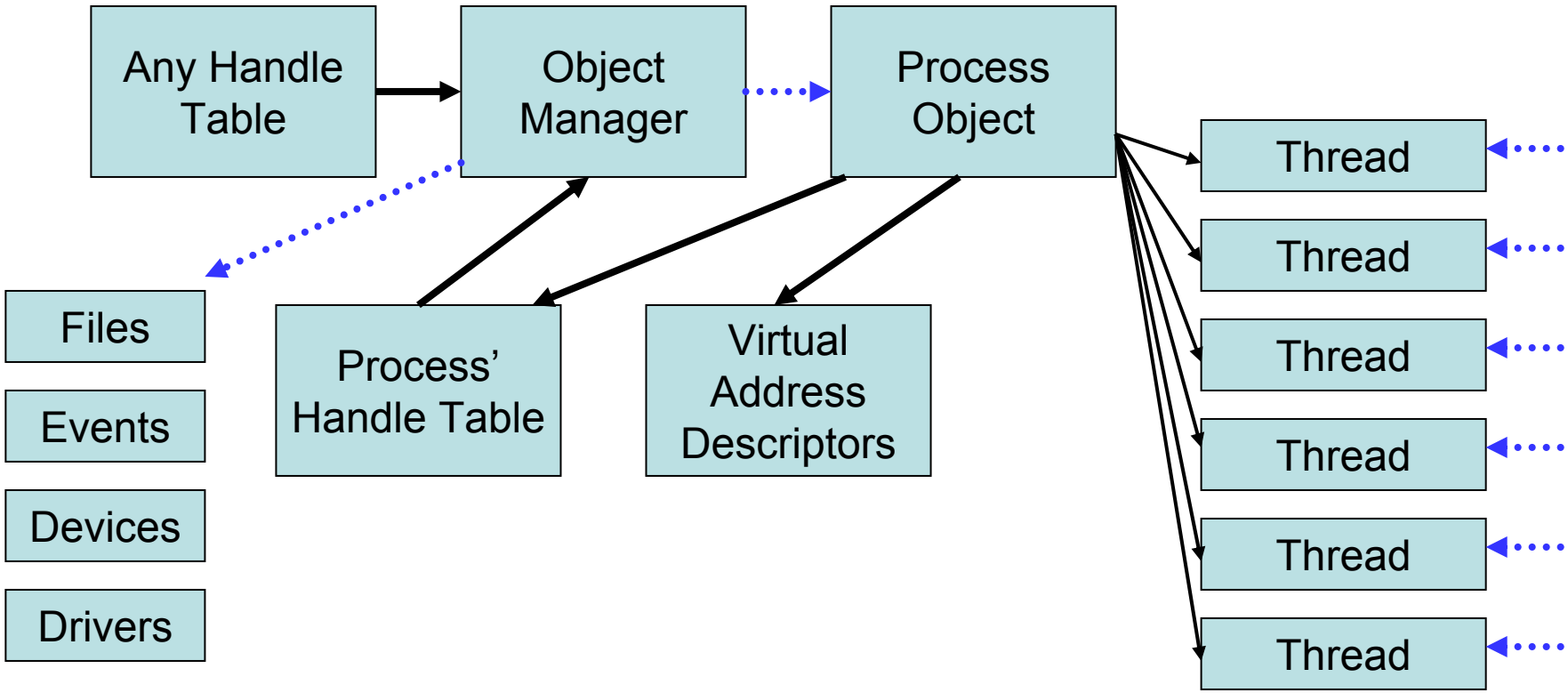
Scheduler (aka 'kernel')

- Schedules thread execution on each processor

CPU Scheduling & IRQLs

- Thread scheduling occurs at PASSIVE or APC level (IRQL < 2)
- APCs (Asynchronous Procedure Calls) deliver I/O completions, thread/process termination, etc (IRQL == 1)
  - Not a general mechanism like unix signals (user-mode code must explicitly block pending APC delivery)
- Interrupt Service Routines run at IRL > 2
- ISRs defer most processing to run at IRQL==2 (DISPATCH level)
- A pool of *worker threads* available for kernel components to run in a normal thread context when user-mode thread is unavailable or inappropriate
- Normal thread scheduling is round-robin among priority levels, with priority adjustments (except for fixed priority real-time threads)

Process/Thread structure



Process

Container for an address space and threads

Associated User-mode Process Environment Block (PEB)

Primary Access Token

Quota, Debug port, Handle Table etc

Unique process ID

Queued to the Job, global process list and Session list

MM structures like the WorkingSet, VAD tree, AWE etc

Thread

Fundamental schedulable entity in the system

Represented by ETHREAD that includes a KTHREAD

Queued to the process (both E and K thread)

IRP list

Impersonation Access Token

Unique thread ID

Associated User-mode Thread Environment Block (TEB)

User-mode stack

Kernel-mode stack

Processor Control Block (in KTHREAD) for cpu state when not running

Significant Windows Releases

- Windows NT 3.1
- **Windows 95**
- Windows 98/98se/ME
- Windows NT4
- **Windows 2000 (enterprise)**
- **WindowsXP (consumer)**
- Windows Server 2003
- Windows XP/SP2
- **“Longhorn”**

Longhorn

Longhorn: codename for next major release

- Most kernel improvements are clean-up, scalability, URT support, fundamentals
- Big bets:
  - WinFX – managed replacement for Win32
  - WinFS – new unified information model
  - Avalon – new GUI programming model
  - Indigo – new messaging infrastructure for services
  - Media – improve audio/video streaming
  - Management, reliability, security