Windows Research Kernel Source overview & project

9 October 2006
Singapore

Dave Probert, Ph.D.
Architect, Windows Kernel Group
Windows Core Operating Systems Division
Microsoft Corporation

Windows Research Kernel (WRK): Goals

- Make it easier for faculty and students to compare and contrast Windows to other operating systems
- Enable students to study source, and modify and build projects
- Provide better support for research & publications based on Windows internals
- Encourage more OS textbook and university-oriented internals books on Windows kernel
- Simplify licensing

UNIX/NT starting-points

	UNIX (early 1970s)	NT (late 1980s)
target hardware environment	16-bit, Kbytes, uniproc, swapping	32-bit, Mbytes, multiproc, virtual memory
style, influences	rich kernel, multics	rich kernel, VMS, multics/unix
namespace root	**file system**	**object manager**
app refs on kernel structures	**file descriptors, pids, IPC numbers, /proc**	**unified handle mechanisms, object mgmt, naming, refs**
devices	**file system**	**object manager**
processes	**integrated: fork/exec**	**assembled as components**
synchronization	**select, special APIs**	**integrated with handles**
memory mgmt	segment/object based	**separation of mapping/object**
I/O model	**synchronous, vtable**	**asynchronous, layered**

NTOS Kernel Sources

Based on Windows Server 2003 SP1 and Windows x64 NTOS

- Processes, threads, LPC, VM, scheduler, object manager, I/O manager, synchronization, worker threads, kernel memory manager, ...
  - most everything in NTOS except plug-and-play, power-management, and specialized code such as the driver verifier, splash screen, branding, timebomb, etc.
  - non-kernel (drivers, file systems, networking) code is from the DDK and IFSKIT
- Simplified in a few places, cleaned up comments, improved spelling
- Non-source is encapsulated in a binary library
- Build and set up utilities and tools
- Tools for tracing, performance monitoring, logging, debugging, etc
- Packaged with
  - WDK subset and documentation for working with drivers
  - VirtualPC environment
  - Kernel regression tests
  - Original specs & design docs for NT, CRK w/ source references
- Over 800K lines of kernel source

WRK licensing

Improvements over current MSR UR license:

- Students can use in classroom environment

License type:

- Non commercial, academic use only; allow derivative works for non-commercial purpose

Eligibility criteria:

- Available to faculty and students in colleges/universities

Usage scenarios:

- View, copy, reproduce, distribute within the institution
- Modify for teaching and experimentation purposes
- Produce teaching and research publications including relevant snippets of source
  - Can use in textbooks and academic publications, and community forums
  - Have to perpetuate MS copyright notices
- Share derivatives within academic community

NT Kernel Design Workbook

NT OS/2 Design Workbook: Core OS		
File	**Title**	**Author(s)**
dwintro	**NT OS/2 Design Workbook Introduction**	Lou Perazzoli
ke	**NT OS/2 Kernel Specification**	David N. Cutler, Bryan M. Willman
alerts	**NT OS/2 Alerts Design Note**	David N. Cutler
apc	**NT OS/2 APC Design Note**	David N. Cutler
ob	**NT OS/2 Object Management Specification**	Steven R. Wood
proc	**NT OS/2 Process Structure**	Mark Lucovsky
suspend	**NT OS/2 Suspend/Resume Design Note**	David N. Cutler
attproc	**NT OS/2 Attach Process Design Note**	David N. Cutler
vm	**NT OS/2 Virtual Memory Specification**	Lou Perazzoli
vmdesign	**NT OS/2 Memory Management Design Note**	Lou Perazzoli
io	**NT OS/2 I/O System Specification**	Darryl E. Havens
irp	**NT OS/2 IRP Language Definition**	Gary D. Kimura
namepipe	**NT OS/2 Named Pipe Specification**	David Cutler & Gary Kimura
mailslot	**NT OS/2 Mailslot Specification**	Manny Weiser
rsm	**Windows NT Session Management and Control**	Mark Lucovsky
fsdesign	**NT OS/2 File System Design Note**	Gary D. Kimura
fsrtl	**NT OS/2 File System Support Routines Specification**	Gary D. Kimura

NT Kernel Design Workbook

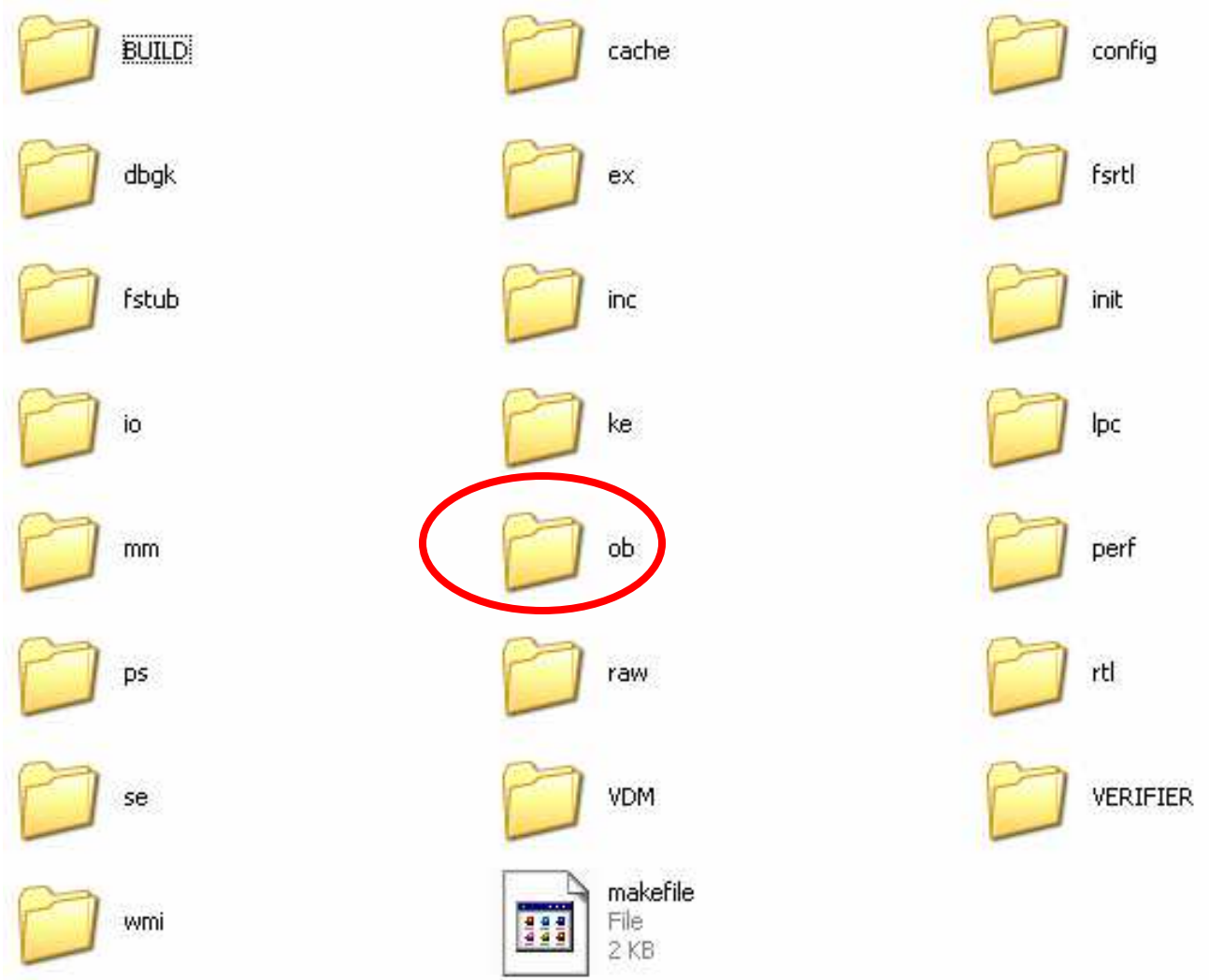
NT OS/2 Design Workbook: Core OS		
File	**Title**	**Author(s)**
sem	**NT OS/2 Event - Semaphore Specification**	Lou Perazzoli
argument	**NT OS/2 Argument Validation Specification**	David N. Cutler
timer	**NT OS/2 Timer Specification**	David N. Cutler
coding	**NT OS/2 Coding Conventions**	Mark Lucovsky, Helen Custer
ulibcode	**NT Utilities Coding Conventions**	David J. Gilman
exceptn	**NT OS/2 Exception Handling Specification**	David N. Cutler
os2	**OS/2 Emulation Subsystem Specification**	Steven R. Wood
status	**NT OS/2 Status Code Specification**	Darryl E. Havens
ntdesrtl	**NT OS/2 Subsystem Design Rational**	Mark H. Lucovsky
resource	**NT OS/2 Shared Resource Specification**	Gary D. Kimura
execsupp	**NT OS/2 Executive Support Routines Specification**	David Treadwell
support	**NT OS/2 Interlocked Support Routines Specification**	David N. Cutler
dd	**Windows NT Driver Model Specification**	Darryl E. Havens
oplock	**NT OS/2 Opportunistic Locking Design Note**	Darryl Havens, et al
memio	**NT OS/2 Memory Management Guide for I/O**	Lou Perazzoli
time	**NT OS/2 Time Conversion Specification**	Gary D. Kimura
mutant	**NT OS/2 Mutant Specification**	David N. Cutler

NT Kernel Design Workbook

NT OS/2 Design Workbook: Core OS		
File	**Title**	**Author(s)**
prefix	**NT OS/2 Prefix Table Specification**	Gary D. Kimura
startup	**NT OS/2 System Startup Design Note**	Mark Lucovsky
dbg	**NT OS/2 Debug Architecture**	Mark Lucovsky
coff	**NT OS/2 Linker/Librarian/Image Format Spec**	Michael J. O'Leary
cache	**NT OS/2 Caching Design Note**	Tom Miller
ntutil	**NT OS/2 Utility Design Specification**	Steven D. Rowe
implan	**NT OS/2 Product Description and Implementation Plan**	David N. Cutler
basecont	**NT OS Base Product Contents**	Lou Perazzoli

The WRK sources

%WRK%\base\ntos



Object Manager lookups

ObpLookupObjectName(Name,Context)

- Search a directory for specified object name
- Use ObpLookupDirectoryEntry() on Directories
- Otherwise call object-specific ParseProcedure
  - Implements symbolic links (SymbolicLink type)
  - Implements file systems (DeviceObject type)

IopParseDevice

(DeviceObject, Context, RemainingName)

- Call SeAccessCheck()
- If (!*RemainingName) directDeviceOpen = TRUE
- For file opens, get Volume from DeviceObject
- Update references on Volume and DeviceObject
- Construct an I/O Request Packet (IRP)
- FileObject = ObCreateObject(IoFileObjectType)
- Initialize FileObject
- Initiate I/O via IoCallDriver(VolumeDevice, IRP)
- Wait for I/O to signal FileObject->Event
- Return the FileObject to caller

ntos\ob\

ntos\io\iomgr\

Setup CMD Variables

WRK Lab

0a. set wap= D:\WAP
0b. set sing= %wap%\SingaporeWorkshop-October2006
0c. set wapcd= %sing%\WAP-July2006
0d. set adds= %sing%\WAP-Additions-October2006
0e. set cdwrk= %wapcd%\WindowsResearchKernel-WRK
0f. set cdspec= %cdwrk%\NTDesignWorkbook
0g. set cdsrc= %cdwrk%\WRK-v1.2
0h. set WRK= %wap%\WRK
0i. set dbgpipe= \\.\pipe\debug
0j. set targetmachine= SingaporeWRKxx
0k. set windbgargs= -k com:pipe,port=%dbgpipe%,resets=0,reconnect
0l. set arch= x86
0m. set hals= %cdsrc%\WS03SP1HALS\%arch%
0n. set dbg= %wap%\Debuggers
0o. set wrkexe= %WRK%\base\ntos\BUILD\EXE
0p. set _NT_SYMBOL_PATH= %WRKEXE%
0q. set cdcrk= %wapcd%\CurriculumResourceKit-CRK
0r. *path %dbg%;%wrk%\tools\%arch%;%path%*

<Open CMD window, RUN **D:\WAP\setvars.bat XX** where XX= **??**>

WRK Lab

Installing VirtualPC and Windows Server 2003

1. *Copied CD onto %wap%*
2. *Installed VirtualPC from: %cdwrk%\VirtualPC2004SP1\FullInstall*
3. *Inserted Windows Server 2003 SP1 disk into CD drive (or use ISO)*
4. *Ran VirtualPC and created virtual machine and hard disk for WS03 in %wap%\VirtualMachines (name machine %targetmachine%)*
5. *Booted from the Windows Server 2003 CD*
6. *Set COM1: %debugpipe% in VPC, resized screen to 800x600*
7. *Installed the VM additions, rebooted VPC*
8. *Installed debugger (at %dbg%):*

   cd /D "%cdcrk%\CRKTools\Debugging Tools"
   dbg_x86_6.6.03.5.exe

WRK Lab

Setup Windbg Debugger

8a. Open debugger help: **%dbg%\Debugger.chm**

8b. Start debugger: **windbg %windbgargs%**

<won't break-in until VirtualPC kernel booted with /DEBUG switch>

Setup VirtualPC shared folder

8c. In VPC Settings:

Add Shared Folder: W: %WRKEXE%

Click box to save across reboots

WRK Lab

Building the WRK kernel

9a. xcopy /crehkdq %cdsrc% %WRK%\
9b. xcopy %hals%\halacpim*.dll %WRKEXE%\
9c. xcopy %adds%*.dll %WRK%\tools\x86\
9d. cd %WRK%\base\ntos
9e. nmake -nologo %arch%=

*Will produce kernel files in **%WRKEXE%\wrkx86.****

NOTE: see instructions in %cdwrk%\README.txt

WRK Lab

Testing the WRK kernel

In CMD window on VPC:

10a. xcopy W:wrkx86.exe C:\Windows\System32\

10b. xcopy W:halacpim.dll C:\Windows\System32\

halacpim.dll is VirtualPC-compatible MP hal

because wrkx86 is built Multi-Processor not Uni-Processor

10c. notepad C:\boot.ini [may need to: attrib –r –h C:\boot.ini]

add: multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="test“

/kernel=wrkx86.exe /hal=halacpim.dll /debug /debugport=com1

10d. Reboot %targetmachine% and specify ‘test’ kernel from menu

10e. Break in with debugger, version will be *3800 MP*

WRK Lab

Fair-share scheduling

<approach suggested by Marty Humphrey at University of Virginia>

Adjust Quantum length based on how many threads are in the process

Kernel changes:

base\ntos\inc\ps.h – use bits in ETHREAD for 'scheduling bucket'

base\ntos\ke\ki.h – mod KxQueueReadyThread & KiSelectReadyThread & KiFindReadyThread

base\ntos\ke\thredsup.c – mod KiDeferredReadyThread

base\ntos\ke\balmgr.c – for completeness

base\ntos\ps\psquery.c – change to NtSetInformationThread to join a scheduling bucket

public\sdk\inc\ntpsapi.h – new definition for NtSetInformationThread

WRK Lab

Fair-share scheduling - algorithm

x = original quantum (from Thread->QuantumReset or just Thread->Quantum)

N = number of groups (i.e. on the ready queue)

n[i] = number of threads on the ready queue in group i

M = sum(i=1,N) N[i] => total number of threads in groups and on the ready queue

y[i] = new quantum for queued threads in group I = (M * x / n[i]) / N

note: sum(i=1,N) y[i] = M * x and: if N == M == n[i] == 1, then y[i] == x
and if N = k then M = N * k and so n[i] = M / N implies y[i] = (N/M) * (1/N) * M = x

WRK Lab

Fair-share scheduling - algorithm

if a thread from bucket i gets queued

```
if (n[i]++ == 0) N++
M++
```

if a thread from bucket i gets dequeued

```
if (--n[i] == 0) N--
M--
```

Implementation note: only put a thread into a bucket from within the thread itself since if it is on the ready queue our bookkeeping will be wrong

WRK Lab

Fair-share scheduling – code changes

```
// Scheduling Bucket globals
struct {
    ULONG ReadyGroups;                              // N
    ULONG ReadyThreads;                             // M
    ULONG Buckets[MAXIMUM_SCHEDULE_BUCKETS];        // n[i]
} BucketScheduling;

if (eThread->BucketsEnabled) {
     BucketScheduling.ReadyThreads++;
     if (BucketScheduling.Buckets
                    [eThread->ScheduleBucket]++ == 0)
         BucketScheduling.ReadyGroups++;
}
```

WRK Lab

Fair-share scheduling – code changes

```
// Adjust quantum if in a scheduling bucket
eThread = CONTAINING_RECORD(Thread, ETHREAD, Tcb);
if (eThread->BucketsEnabled) {
  ULONG bucket = eThread->ScheduleBucket;
  ULONG newquantum;

  newquantum = Thread->QuantumReset *
      BucketScheduling.ReadyThreads /
      (BucketScheduling.ReadyGroups *
      BucketScheduling.Buckets[bucket]);
  if (newquantum > 127)
    newquantum = 127;
  else if (newquantum < THREAD_QUANTUM)
    newquantum = THREAD_QUANTUM;
```

WRK Lab

Fair-share scheduling – code changes

```
    Thread->Quantum = (SCHAR)newquantum;

    BucketScheduling.ReadyThreads--;
    if (--BucketScheduling.Buckets[bucket] == 0)
      BucketScheduling.ReadyGroups--;
}
```

WRK Lab

Project: modify scheduler algorithm

Fair-share scheduling

<install project files>

copy WRKProject\usermode\exei386.exe V:\FairShare*

Logon to VirtualPC, cd to C:\FairShare and run:

spawnthreads -t 100 4 8 16 32

process	nthreads	cpu	%	**normal**	fair
0	4	6.74	**6.87%**	**6.67%**	25.00%
1	8	13.53	**13.79%**	**13.33%**	25.00%
2	16	26.08	**26.58%**	**26.67%**	25.00%
3	32	51.78	**52.77%**	**53.33%**	25.00%
	60	98.13	100.00%		

WRK Lab

Fair-share scheduling – result

spawnthreads ***-s*** *-t 100 4 8 16 32*

process	*nthreads*	*cpu*	*%*	*normal*	***fair***
0	*4*	*28.87*	***29.05%***	*6.67%*	***25.00%***
1	*8*	*19.27*	***19.39%***	*13.33%*	***25.00%***
2	*16*	*17.61*	***17.72%***	*26.67%*	***25.00%***
3	*32*	*33.63*	***33.84%***	*53.33%*	***25.00%***
	60	*99.38*	*100.00%*		

WRK Lab

Fair-share scheduling – result

spawnthreads [-s] -t 100 4 8 16 32

process	nthreads	%	[-s]%	normal	fair
0	4	6.87	29.05%	6.67%	25.00%
1	8	13.79	19.39%	13.33%	25.00%
2	16	26.58	17.72%	26.67%	25.00%
3	32	52.77	33.84%	53.33%	25.00%
	60				

Exercise for the class: Why not more exact?

spawnthreads -t 100 4 8 16 32 128 256 512 99 113 270 4 8 32 8 4 99 270

process	nthreads	%	% [-s]	normal	fair
0	4	**1.03**	5.16%	**0.21%**	5.88%
1	8	**1.96**	4.86%	**0.43%**	5.88%
2	16	**3.62**	4.61%	**0.86%**	5.88%
3	32	**6.66**	5.17%	**1.72%**	5.88%
4	128	**9.36**	6.35%	**6.87%**	5.88%
5	256	**9.77**	6.85%	**13.74%**	5.88%
6	512	**9.11**	7.15%	**27.48%**	5.88%
7	99	**8.82**	6.72%	**5.31%**	5.88%
8	113	**9.08**	6.83%	**6.07%**	5.88%
9	270	**10.75**	7.10%	**14.49%**	5.88%
10	4	**0.97**	6.83%	**0.21%**	5.88%
11	8	**1.83**	4.94%	**0.43%**	5.88%
12	32	**5.87**	4.57%	**1.72%**	5.88%
13	8	**1.81**	4.97%	**0.43%**	5.88%
14	4	**0.98**	5.70%	**0.21%**	5.88%
15	99	**9.75**	6.04%	**5.31%**	5.88%
16	270	**8.63**	6.15%	**14.49%**	5.88%
	1863				

WRK Lab - Demonstrations

Debugging with Windbg

Perfmon

WRK Futures

- *Data structure modification guide*
- *More internals documentation*
- *Community project resources*
- *More source (community-driven)*
- *Phoenix-based instrumentation*

Questions & Discussion