From last time

A file system uses inodes which contain 8 block-numbers. These are for the first 7 blocks of the file and an indirect block, which just contains block-numbers for the remaining blocks in the file.
A block-number occupies 2 bytes. Each block is 4k bytes.

What is the maximum size of a file in bytes?

What is the maximum total size of directories and files in a single disk partition?

MANCHESTER 1824

The University of Manchester

COMP25111: Operating Systems

Lecture 17: Windows XP Case Study

John Gurd

School of Computer Science, University of Manchester

Autumn 2014

Overview & Learning Outcomes

Background

Components: layers & managers
- Scheduling
- Virtual Memory
- Input/Output
- File system

History & Motivation

Early Windows systems (3.1, 95, 98)
– demonstrated the usefulness of a GUI-based OS
– designed for “home use” (not secure or multiprocessing)
– initially 16-bit

Windows NT:
– designed as a commercial strength 32-bit system
– led to Windows 2000, XP, Vista, 7, ...

Goals

Portability: written in C and C++
($>$29M lines – Linux $<$2M + 1.5M for X windows)
– HAL (Hardware Abstraction Layer) = processor-dependent

Extensibility: layered architecture; executive in kernel mode.

Compatibility (Win16, Win32, POSIX, OS/2)
– DOS emulation in “Virtual DOS machine”

Performance, Scalability, Multiprocessor support

Reliability

International support (Unicode etc.)

Unifying Themes

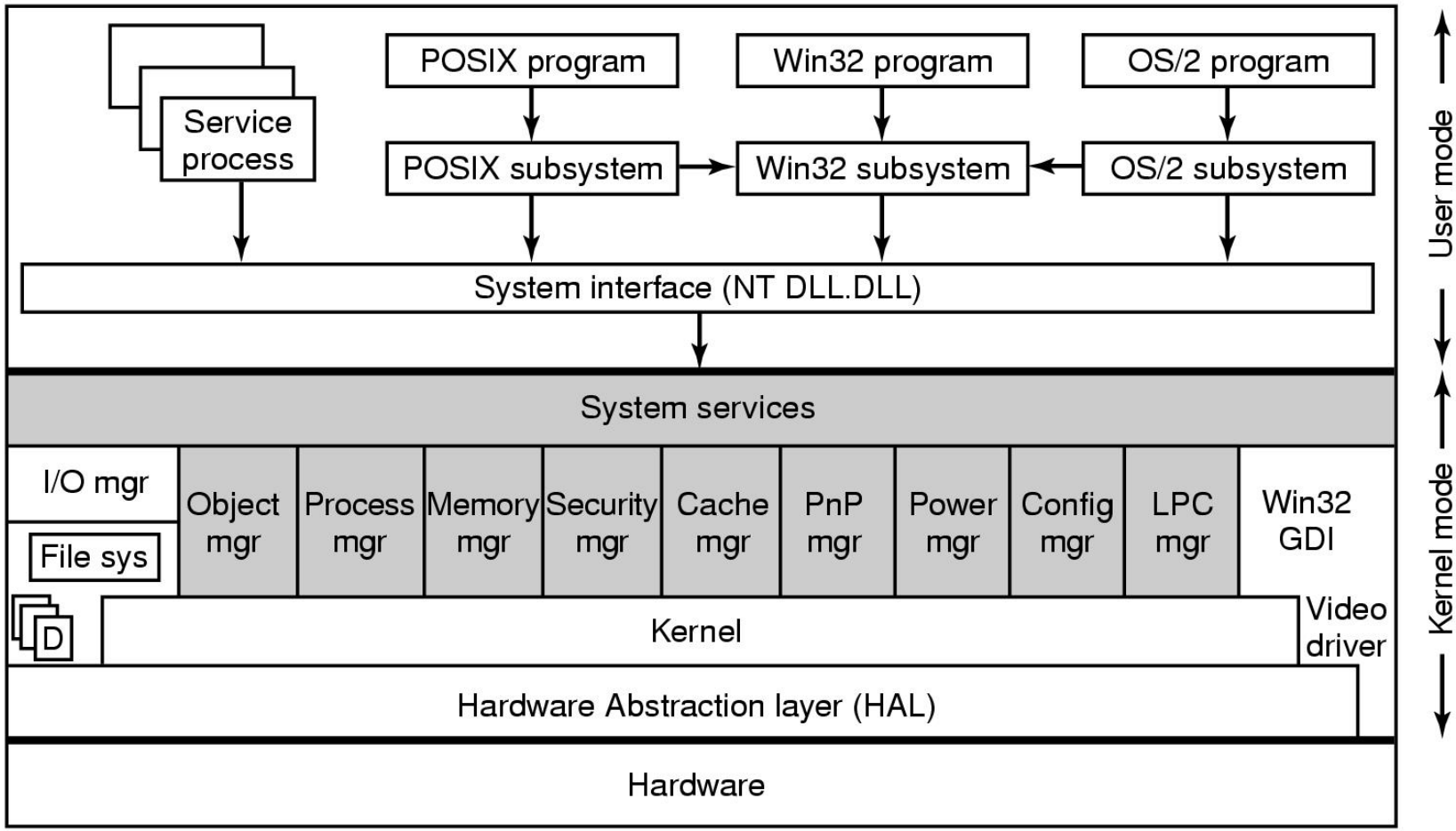
For non-programmers

Users (& programmers) see Windows as a single entity
– GUI built into OS
– Win32 API hides library/system split

“Object-Oriented”
– Handles
– methods

Registry

Structure (MOS fig 11.7)



Architecture – layered system of modules

Protected mode:

– HAL: e.g. read/write device registers, interrupt handling so independent of bus; timers/clock

– “kernel”: thread scheduling, interrupt handling, low-level processor synchronisation, recovery after power failure, etc.

– executive: managers; “system services” distribute system calls to managers

User mode:

– service processes

– environment subsystems (Win32, Posix, OS/2)

System services

System call interface not publicly available

Well defined Win32 API
– very large library of functions
– which hide system calls
– includes GUI

(Unix: minimal set of system calls, all public)

Object manager

"object-oriented" – uses objects ("handles") for all services & entities

Objects: process, thread, section of memory, open file, comms port, semaphore, timer, registry key, device, driver, ...

Searchable directory

Named objects can be shared,
Unnamed objects are private to creating process

Each object can have Access Control List (ACL)

Configuration manager – Registry

Hive = name of a registry database
e.g. HKEY_CURRENT_USER (HKCU)

Subkey/Key = sub/directory
e.g. `HKEY_CURRENT_USER\Control Panel\desktop`

entry = (name, type, data) i.e. value
e.g. “ScreenSaveTimeOut”=“60”
“ScreenSaveUsePassword”=dword:00000000

+ replaces many scattered configuration files
– hard to copy, share, edit, back-up
– heavily used, single point of failure

Integrity

Kernel – scheduling

Job – Process – Thread – Fiber
– process owns resources
– threads scheduled by OS (system calls)
– fibers scheduled by library (outside OS)

Also deals with
– pipes, sockets, rpc ...
– semaphores ...
– multi-core CPUs

Scheduling

Pre-emptive
Priorities (0:zero-pages, 1-15:user, 16-31:system)
Selectable time quantum (e.g. 20ms)

Dynamically change user priorities (initial-15 only):
– time quantum expired: –1
– boost priority of threads released from I/O (e.g.+6 keyboard)
– extend quantum of foreground application
– starvation prevention (temporarily boost priority)
  find threads that haven't run in a while

Cannot guarantee real-time deadlines

Memory Management

assumes H/W support; based on Intel PC memory architecture

2^{32} = 4GB address space; top half=OS, bottom=user
pages (e.g. 4kB); no segments, but 64kB boundaries
top & bottom 64kB unused

processes can share pages
– “copy on write”
– “position independent code”

version of LRU
keeps some pages free, so can always load new page
dynamically calculated max working set size per process
lots of heuristics and kludges

can memory-map files

Input/Output

Manager interrogates each slot of each bus
– at boot time (mostly)
– at any time (e.g. USB) – Plug&Play
device info $\rightarrow$ where to find driver

Drivers:
– I/O Request Packet (IRP)
– Object-based (list of methods):
init, add device, interrupt, fast I/O, DMA, abandon ...
– unplug
– configurable
– multi-processor safe

Monolithic or “stacked”

NTFS

"clusters" (blocks) e.g. 4KB

File = set of attributes (byte streams – max 2^{64}B)

Master File Table (MFT) = set of 1kb records
0-26: metadata (MFT, log, boot, rootdir, bitmap, ACL, ...)

MFT record = attributes, name, list of (block start & count)
(if needed: list of overflow records)

directory: data = list of (file-name, MFT record, ...)
(large directories use B+ trees)

Integrity: updates within transactions, logging

Compression & Encryption

Summary of key points

Background

Components: layers & managers
- Scheduling
- Virtual Memory
- Input/Output
- File system

Your Questions

Glossary

Hardware Abstraction Layer (HAL)
DLL
Win32 API
Handle
Registry
Hive, Key
Fiber
NTFS
Master File Table (MFT)

Reading

newer OSC/J: Ch 22

older OSC/J: Ch 21

MOS: Ch 11