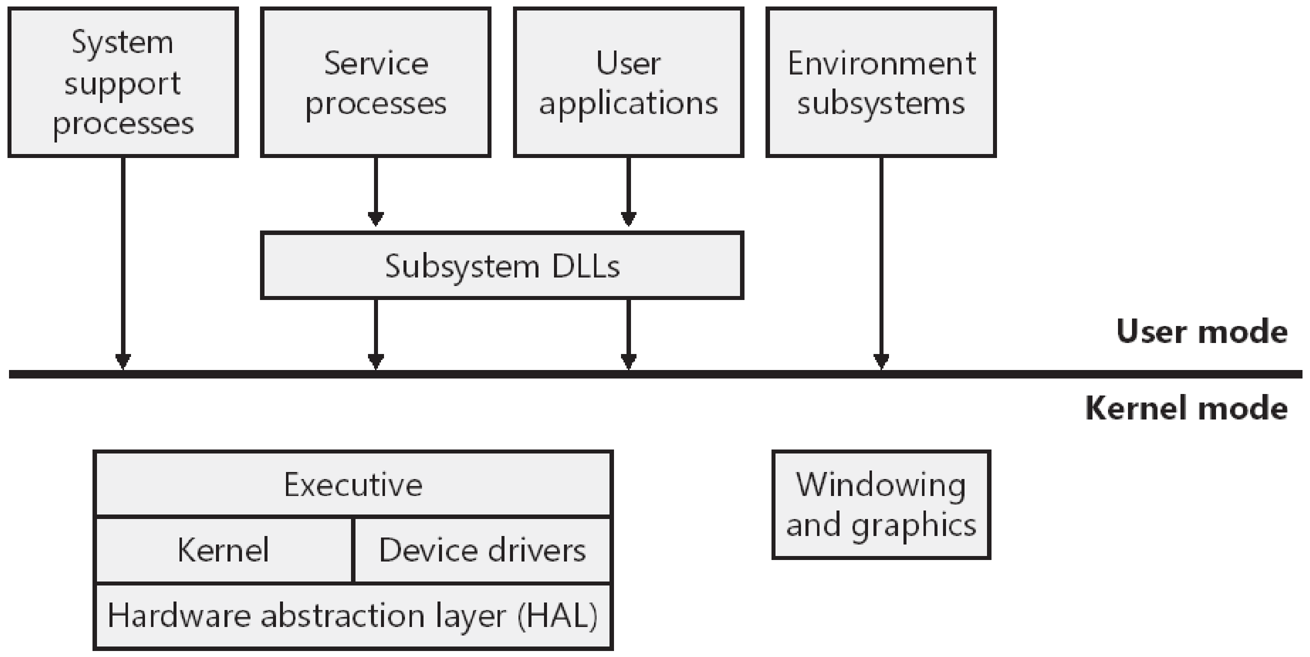
windows

maurizio pizzonia
roma tre university

references

M. Russinovich, D. A. Solomon
Windows® Internals: Including Windows Server 2008 and Windows Vista 5th ed.
Microsoft Press

architecture overview



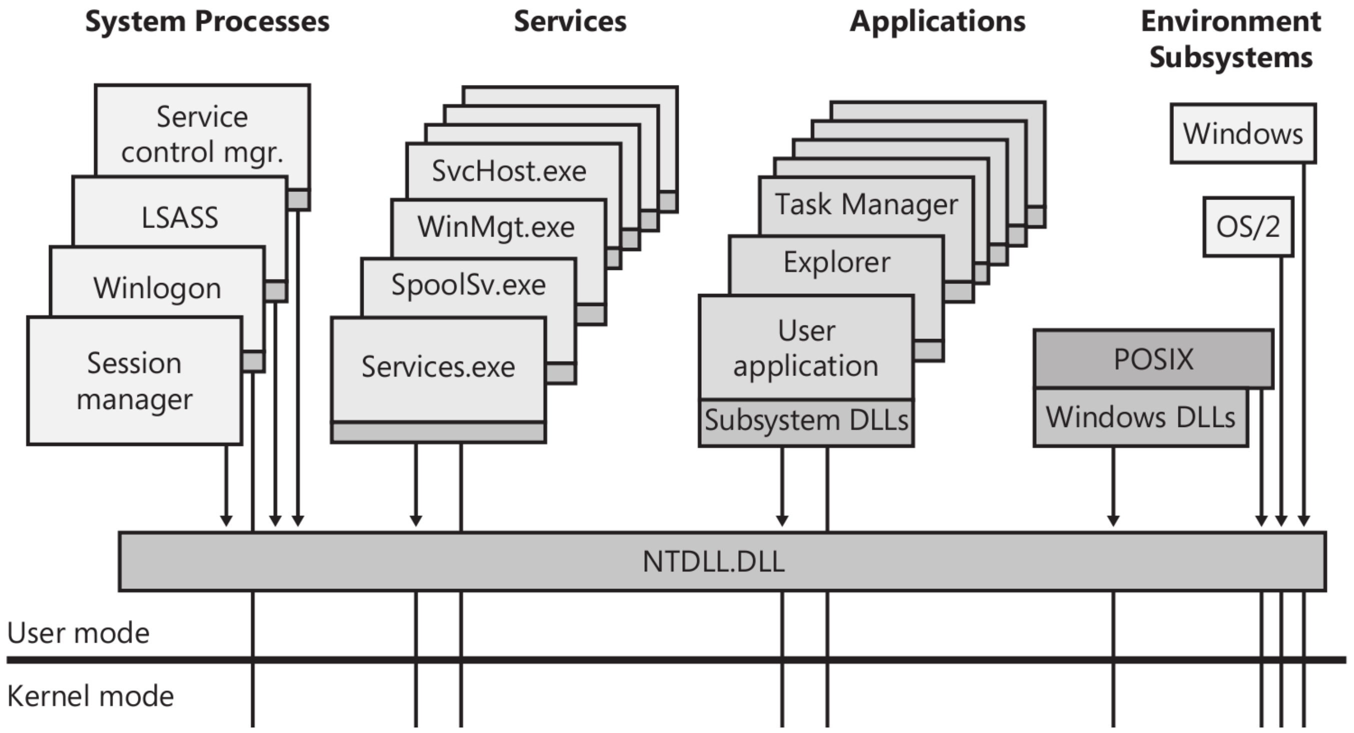
- kernel: overloaded word
  - according to MS it is only a part of what runs in kernel mode
- graphics is in kernel mode
- user applications and services never access syscalls directly
  - use "subsystems DLLs" that goes with "environment subsystems"
- several "system support processes"

windows processes

- processes “usually” form a tree
  - the parent is the creator of the process
- if the parent dies the info is not updated in the child
  - parent information is not reliable
- so... tree is only informative, Windows does not rely on that

- besides, Windows uses kernel threads for its own needs

architecture details: user mode



user processes and relationships with Windows

- syscalls are never directly performed by processes
  - syscalls are not documented
  - decoupling layers
    - Ntdll.dll (documented)
    - substystem DLLs (preferred way to ask Windows something)
- subystems: windows, posix, os/2
  - decouple user processes from underlying OS
    - e.g. allows “easy” porting of unix software
  - are DLLs + supporting process
    - supporting process: see “environment subsystem”
    - subsystem DLLs may call Ntdll.dll, interact with supporting process or just update a “local state”
- are subsystems really needed?
  - my impression is that subsystems are a “legacy” feature

windows subsystem

- a particular subsystem
- DLLs
  - kernel32.dll, Advapi32.dll, User32.dll Gdi32.dll
- environment subsystem process: **csrss.exe**
  - it should be always running (otherwise whole system crashes)
    - the only one, others run on-demand
  - in NT 3.51 it contained a lot of stuff (graphics)
  - now it is almost empty (console applications)
- kernel parts
  - graphics: win32k.sys
    - window manager (mouse, windows, graphic message handling, etc.)
    - graphic rendering (rendering of text, drawing, etc.)

typical system processes

- windows subsystem
  - csrss.exe
- session manager
  - smss.exe, winlogon.exe, winint.exe
- service control manager (scm)
  - services.exe, svchost.exe
- ...

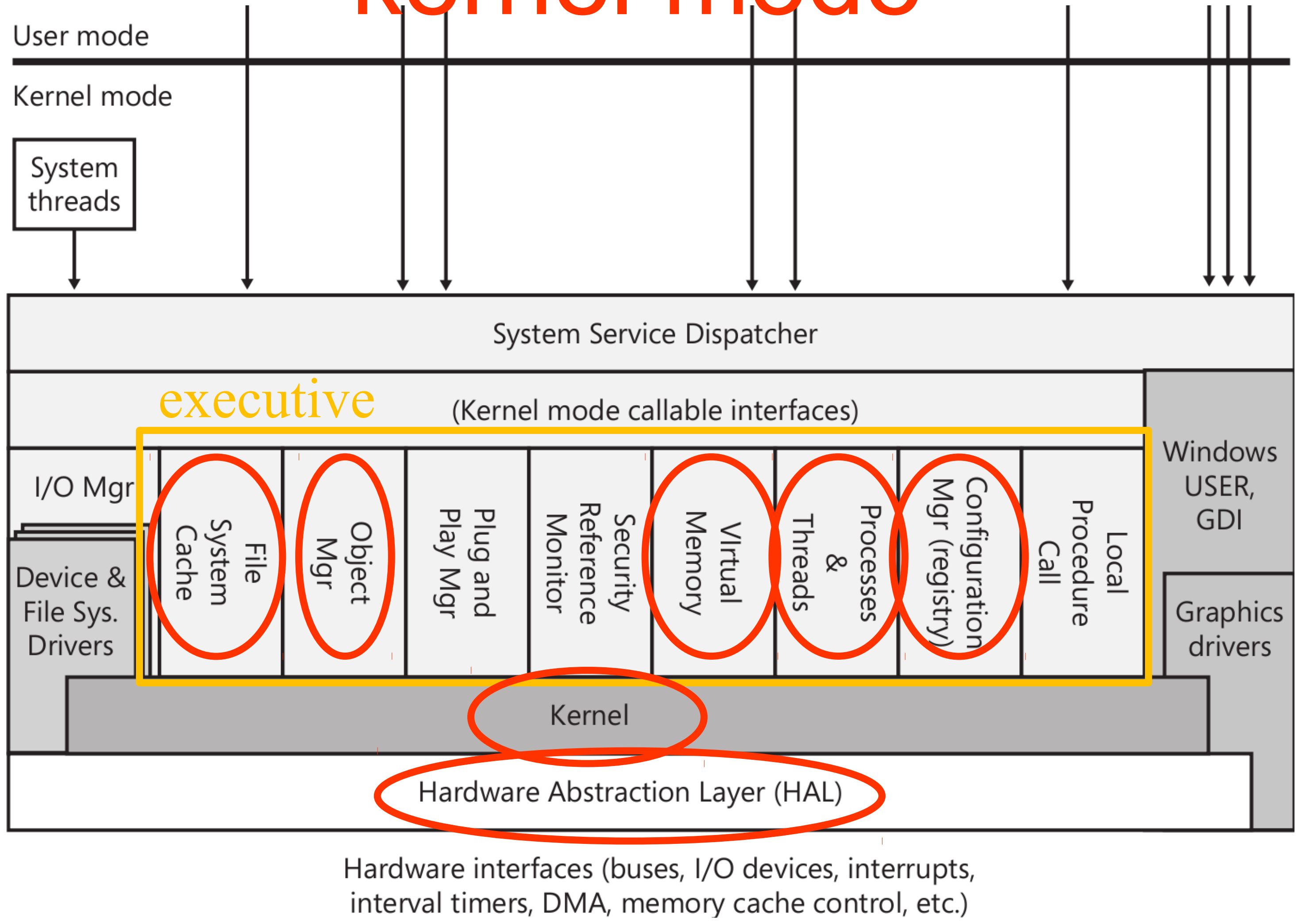
sessions manager and boot

- smss is the first process created at boot
  - smss is the only process to use syscalls directly since windows subsystem is not started yet!
  - it starts Autochk (filesystem check)
  - it starts wininit.exe
    - which starts csrss.exe
      - which loads win32k.sys (and the video switches to correct resolution)
    - and services.exe which starts all configured services
- smss also waits for new session requests, and for each session...
  - starts winlogon.exe (the password dialog box)
    - when a logon happens it (indirectly) starts explorer.exe
  - starts new csrss.exe
- to have more sessions you need “terminal server” and proper licensing

services

- managed by the service control manager (scm, services.exe)
  - scm is started by wininit.exe
- it is in charge of starting/stopping/pausing services
  - configured in the registry
  - edited by Control panel → Admin Tool → Services
- a **single process can host more services**
  - the standard generic host service: svchost.exe
  - in this case services hosted are implemented as DLLs
- such processes have **specific APIs** to interact with the control manager
  - e.g. notify correct start-up, pause, start a guest service, etc.
- services have three names
  - the **executable**, the name in the **registry**, the name shown by the **configuration** utilities
- e.g.
  - EventLog, TaskScheduler, Spooler, etc.

architecture details
kernel mode


User mode
Kernel mode
System threads
System Service Dispatcher
executive
(Kernel mode callable interfaces)
I/O Mgr
Device & File Sys. Drivers
File System Cache
Object Mgr
Plug and Play Mgr
Security Reference Monitor
Virtual Memory
Processes & Threads
Configuration Mgr (registry)
Local Procedure Call
Windows USER, GDI
Graphics drivers
Kernel
Hardware Abstraction Layer (HAL)
Hardware interfaces (buses, I/O devices, interrupts, interval timers, DMA, memory cache control, etc.)

architecture details kernel mode

- hal
  - handle motherboards differences
- kernel
  - basic threads and processes scheduler, synchronization, interrupt handling
  - no I/O
- executive
  - executive objects
  - memory management, real process/thread management, security, I/O, networking, inter-process communication,etc.

resources, objects, and handles

- any **resources** is view by a process as an **executive object**
  - e.g. an open file, a process, a session, etc.
- an executive object is stored in kernel space
- in user space, executive objects are represented by **handles**
  - processes use them through handles
- **object manager**
  - part of executive
  - keep a **process handle table** for each process
    - it contains handle that the process can use
- most API parameters are handles

executive objects types

type	Description
Process	A collection of executable threads along with virtual addressing and control information.
Thread	An entity containing code in execution, inside a process.
Job	A collection of processes.
File	An open file or an I/O device.
File mapping object	A region of memory mapped to a file.
Access token	The access rights for a process
Event	An object which encapsulates information to be notified to a processes of something.
Semaphore/Mutex	Objects which serialize access to other resources.
Timer	An objects which notifies processes at fixed intervals.
Key	A registry key.
Desktop	A logical display surface to contain GUI elements.
Clipboard	A temporary repository for other objects.
WindowStation	An object containing a group of Desktop objects, one Clipboard and other user objects.
Symbolic link	A reference to other objects, via which the referred object can be used.

object sharing

- objects can be shared among processes
- some are “anonymous”
- some are named
  - identified by a string
- there exists system of directory
  - string is a pathname
  - not persistent
    - it exists only in memory

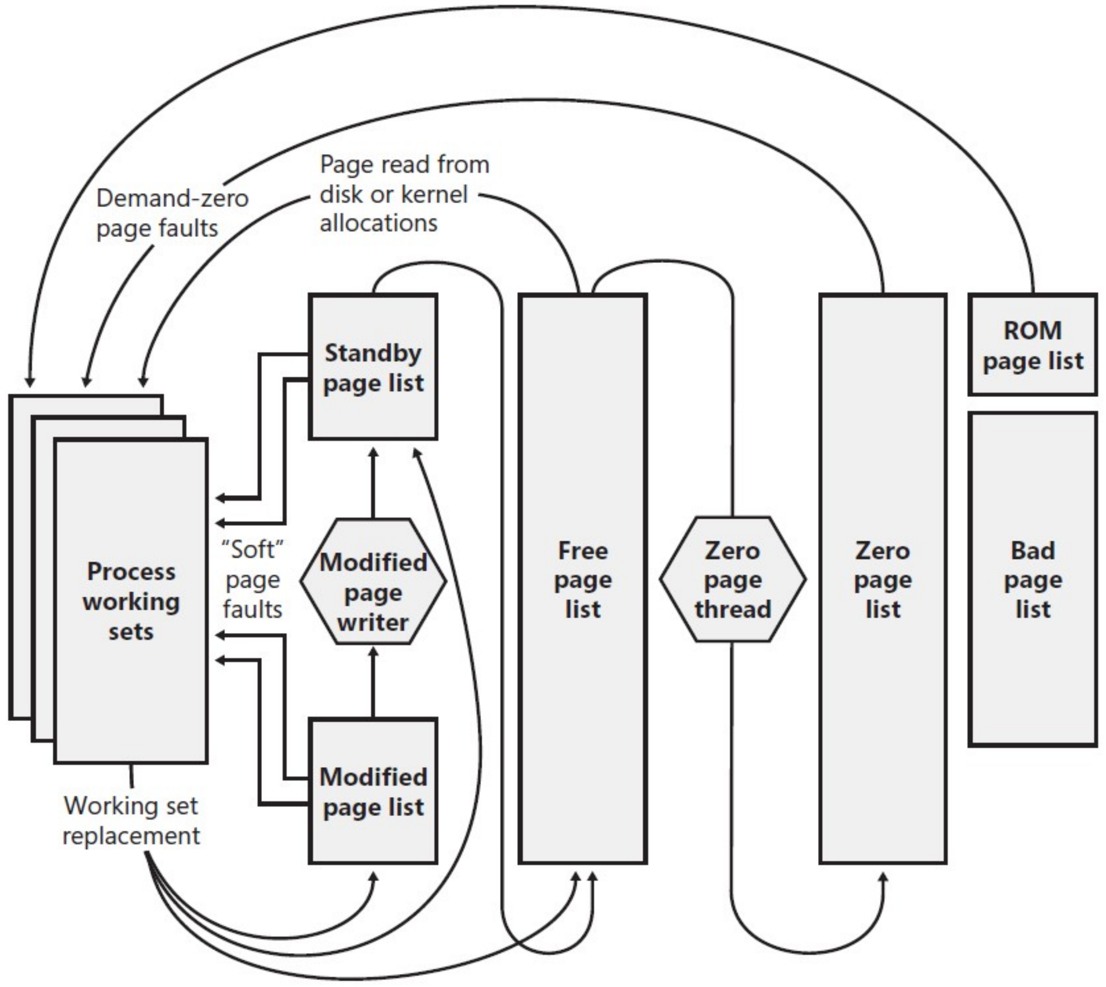
memory management

- process address space contains shared kernel space
- kernel space/user space
  - 32bit systems: 2GB/2GB (config. 1GB/3GB)
  - 64bit systems: 6TB/8TB
- virtual memory
- memory mapped files and disk cache
- process heap managed in kernel mode
- two kernel space heaps
  - one is not paged
  - one is paged
    - windows keeps a lot of data, it needs paging also in kernel space
- copy-on-write
  - posix environment uses it for implementing fork operations

memory management components

- balance set manager
  - decide resident set for processes
    - in MS terminology it is called “working set” (no global replacement policy!)
  - eviction strategy: aging
  - kernel thread, run once per second
  - also part of kernel space can be evicted
- page buffering (in MS terminology “stand by pages”)
  - two kernel thread for cleaning the pages
  - one kernel thread for zeroing the pages
    - new empty pages are always given a zeroed frame
- swap (page file)
  - one kernel thread to change its size
- disk cache
  - cache part of files using memory mapping

page frames states



- from russinovich, solomon

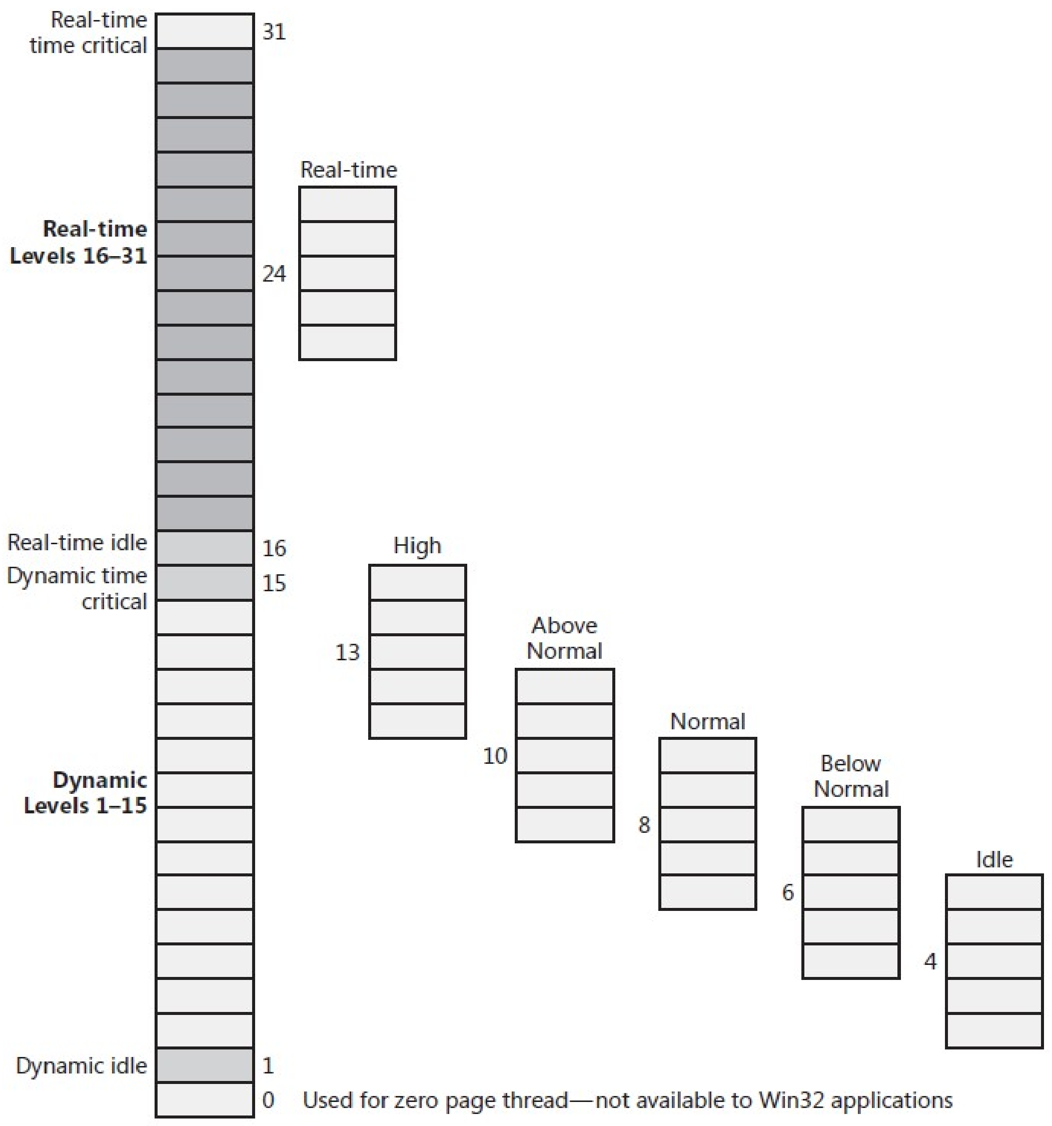
disk cache

- caches part of files
- two kinds of blocks
  - regular read/write
    - mapped on system address space
    - act as a middle layer between processes and the filesystem
  - memory mapped files
    - mapped on process address space
- size of cache changes along with system resident set
  - balance manager can change it dynamically

cpu scheduling

- internally: 32 priority level
  - 31-16: “real time”
  - 15-1: dynamic
  - 0: system (the zeroing page thread)
- each level has its queue
- preemption
- at user level: 5 priority levels (base priority)
  - high, above normal, normal, below normal, idle
  - each of them have an internal priority dynamically assigned within a range of 5 internal priority levels

priority levels



priority boost

- windows increases internal process priority after waiting for
  - i/o completion
  - synchronization events
  - user input from GUI
- after a long time in ready state without being scheduled
  - to avoid starvation

interactive processes

- system clock
  - interrupt every 10-15ms
- default quantum
  - windows xp, 2 clock intervals
  - servers, 12 clock intervals
    - less context switches (more efficient) but slower interactive response
- quantum boost for foreground processes (i.e. with focused window)
  - windows xp: 6 clock intervals

interactive processes

- quantum accounting
  - in units that are 1/3 of a clock interval
  - at each clock interrupt
    - running process has quantum decremented by 3 units
    - waiting processes have quantum decremented by 1 unit
    - check for quantum expired
- when quantum expires
  - put at the end of its queue (round robin)

registry

- it's like a filesystem for small “data element”
  - persistent: realized as a set of files (*hives*)
- structure
  - key/subkey = directory/subdirectory
  - value = file
    - typed: strings, numbers, arrays, symbolic links
- symbolic links
  - not persistent!
  - re-created after each boot

standard registry tree

- six roots
  - cannot be changed
  - named with abbreviations HK...
- three are “real”
  - **HKEY_USERS (HKU)**
  - **HKEY_CLASSES_ROOT (HKCR)**
  - **HKEY_LOCAL_MACHINE (HKLM)**
- others are not
  - HKEY_CURRENT_USER (HKCU)
    - link to something within HKU
  - HKEY_CURRENT_CONFIG (HKCC)
    - link to HKLM\SYSTEM\CurrentControlSet\Hardware Profiles\Current
  - HKEY_PERFORMANCE_DATA (HKPD)
    - performance data, created on-the-fly, it does not appear in regedit

registry content

- HKEY_USERS
  - preferences of each user
  - HKCU points to the user that is asking the request
- HKEY_CLASSES_ROOT
  - file associations and com object data
- HKEY_LOCAL_MACHINE
  - HARDWARE
    - hardware collected data at boot
  - SECURITY
    - security staff, e.g. user accounts and SAM (also linked under HKLM)
  - SOFTWARE
    - one subkey for each installed software, content depends on the sotware
  - SYSTEM
    - windows, system wide, configuration (needed at boot)