

Sistemi Operativi

Il Sistema Operativo Windows

(parte 1)

Docente: Claudio E. Palazzi
cpalazzi@math.unipd.it

Crediti per queste slides al Prof. Tullio Vardanega

Genesi – 1

- **MS-DOS (MicroSoft Disk Operating System)**
 - Mono-utente in modalità *command line*
 - **Non multi-programmato**
 - Inizialmente ispirato a CP/M
 - 1981 : **1.0** (8 KB codice) → PC IBM 8088 (16 *bit*)
 - 1986 : **3.0** (36 KB) → PC IBM/AT (i286 @ 8 MHz, ≤ 16 MB)
- **Windows 1^a generazione**
 - Modalità GUI solo come rivestimento di MS-DOS
 - Interfaccia **copia** del 1° modello Macintosh di Apple
 - 1990 - 1993 : **3.0, 3.1, 3.11** → i386 (32 *bit*)

GUI

- **GUI** (*Graphical User Interface*)
 - Introdotto dal modello **Macintosh** di Apple il 24 gennaio 1984
 - Vedi <http://www.apple-history.com/lisa.html>
 - Basato sul paradigma **WIMP** (dispregiativo!)
 - Finestre (*windows*), icone (*icons*), menu e dispositivi di puntamento (*pointing*)
- Realizzabile
 - Sia come programma in spazio utente (GNU/Linux)
 - Che come parte del S/O (Windows)



Genesis – 2

- **Windows 2^a generazione**

- Vero e proprio S/O **multiprogrammato** ma sempre **mono-utente** con FS su modello FAT

- 1995 : Windows 95 (MS-DOS 7.0)

- 1998 : Windows 98 (MS-DOS 7.1)

- Nucleo a procedure Incapaci di più esecuzioni simultanee

- Ogni accesso a nucleo protetto da semaforo a mutua esclusione

- » Scarsissimi benefici di multiprogrammazione

- ¼ dello spazio di indirizzamento di processo (4 GB totali) condiviso R/W con gli altri processi; ¼ condiviso R/W con il nucleo

- » Scarsissima integrità dei dati critici

- 2000 : Windows ME (ancora MS-DOS)

Modeste
modifiche

Genesis – 3

- **Windows 3^a generazione**
 - Progetto **NT**: abbandono della base MS-DOS
 - Architettura a 16 *bit*
 - Enfasi su sicurezza e affidabilità
 - FS di nuova concezione (**ntfs**)
 - 1993 : Windows NT 3.1 → fiasco commerciale per la mancanza di programmi di utilità
 - 1996 : Windows NT 4.0 → reintroduzione di interfaccia e programmi Windows 95
 - Scritto in C e C++ per massima portabilità al costo di grande complessità (16 M linee di codice!)
 - Molto superiore a Windows 95/98 ma privo di supporto per *plug-and-play* gestione batterie e emulatore MS-DOS

Genesis – 4

- **Windows 3^a generazione** (segue)
 - **Architettura di NT 3.1 a *microkernel*** e modello *client-server*
 - La maggior parte dei servizi è incapsulata in processi di sistema eseguiti in modo utente e offerti ai processi applicativi tramite scambio messaggi
 - **Elevata portabilità**
 - Dipendenze *hardware* localizzate nel nucleo
 - **Bassa velocità**
 - Più costosa l'esecuzione in modo privilegiato
 - **Architettura di NT 4.0 a nucleo monolitico**
 - Servizi di sistema riposizionati entro il nucleo

Genesis – 5

- **Windows 3^a generazione** (segue)
 - 1999 : Windows 2000 (alias di **NT 5.0**)
 - Il S/O esegue in **modo nucleo**
 - Lo spazio di indirizzamento dei processi è interamente privato e distinto dal modo nucleo
 - Periferiche rimuovibili
 - *Plug-and-play*
 - Internazionalizzazione (configurabile per lingua nazionale)
 - Alcune migliorie a **ntfs**
 - MS-DOS completamente rimpiazzato da una *shell* di comandi che ne riproduce e estende le funzionalità
 - Enorme complessità: oltre 29 M linee di codice C[++]

Genesis – 6

- **Windows 3^a generazione** (segue)
 - 2001: Windows XP
 - Migliorie grafiche rispetto a Windows 98 da cui ci si vuole definitivamente staccare
 - Un successo
 - Non è previsto utilizzo come tipo server
 - 2007: Windows Vista
 - Windows Aero
 - Flop (lento e pesante)
 - Seppure più sicuro e con meno buffer overflow
 - Supporto WinXP esteso fino ad aprile 2014

Genesis – 7

– 2009: Windows 7

- A meno di 3 anni dall'uscita del predecessore
 - Corsa ai ripari per insuccesso Vista
- Miglioramenti in efficienza e interfaccia grafica
 - Supporto multitouch
- In alcune versioni include un emulatore di XP
 - Per assicurare assoluta compatibilità e facilitare transizione

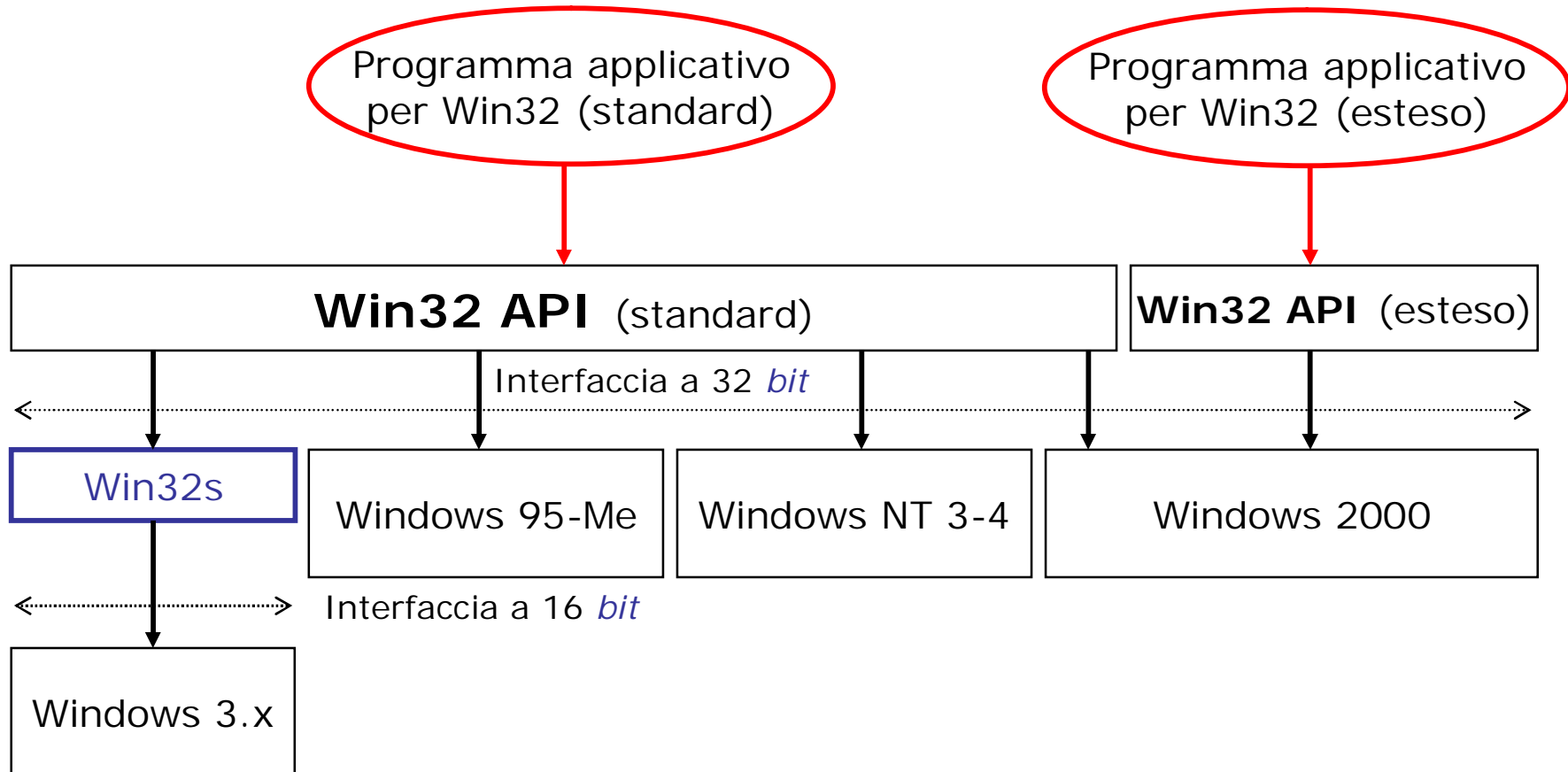
– 2012: Windows 8

- Integrazione PC, Smartphone, Tablet
- Windows Store per distribuzione app
- Interfaccia con Tiles
- Efficienza energetica (consumo batteria dispositivi mobili)

Interfaccia di programmazione – 1

- Basato su principio speculare a quello adottato da UNIX e GNU/Linux
 - Interfaccia di sistema **non** pubblica
 - Procedure di libreria pubblicate in **Win32 API** (***Application Programming Interface***) a uso del programmatore ma controllata da Microsoft
 - Alcune procedure includono chiamate di sistema
 - Altre svolgono servizi di utilità eseguiti interamente in modo utente
 - Nessun sforzo di evitare ridondanza o rigore gerarchico

Interfaccia di programmazione – 2



Informazioni di configurazione

- Tutte le informazioni vitali di configurazione del sistema sono raccolte in una specie di FS detto **registry** salvato su disco in *file* speciali detti **hives**
 - *Directory* → *key*
 - *File* → *entry* = {nome, tipo, dati}
- 6 *directory* principali con prefisso **HKEY_**
 - Per esempio: **HKEY_LOCAL_MACHINE** con *entry* descrittive dell'*hardware* e delle sue periferiche (**HARDWARE**) dei programmi installati (**SOFTWARE**) e con informazioni utili per l'inizializzazione (**SYSTEM**)

Informazioni di configurazione

Key	Description
HKEY_LOCAL_MACHINE HARDWARE SAM SECURITY SOFTWARE SYSTEM	Properties of the hardware and software Hardware description and mapping of hardware to drivers Security and account information for users System-wide security policies Generic information about installed application programs Information for booting the system
HKEY_USERS USER-AST-ID AppEvents Console Control Panel Environment Keyboard Layout Printers Software	Information about the users; one subkey per user User AST's profile Which sound to make when (incoming email/fax, error, etc.) Command prompt settings (colors, fonts, history, etc.) Desktop appearance, screensaver, mouse sensitivity, etc. Environment variables Which keyboard: 102-key US, AZERTY, Dvorak, etc. Information about installed printers User preferences for Microsoft and third party software
HKEY_PERFORMANCE_DATA	Hundreds of counters monitoring system performance
HKEY_CLASSES_ROOT	Link to HKEY_LOCAL_MACHINE\SOFTWARE\CLASSES
HKEY_CURRENT_CONFIG	Link to the current hardware profile
HKEY_CURRENT_USER	Link to the current user profile

Gestione dei processi – 1

- *Job* = {processi gestiti come singola unità con limiti risorse}
- Processo = possessore di risorse, con ≥ 1 *thread*

ID unico, 4 GB di spazio di indirizzamento (2 in modo utente e 2 in modo nucleo), inizialmente con singolo *thread*, simile al processo UNIX; **non** ha stato di avanzamento

- *Thread* = flusso di controllo gestito dal nucleo

Esegue per conto e nell'ambiente del processo (che **non** ha stato di avanzamento), con ID **localmente** unico, 2 *stack* (1 per modo)

- *Fiber* = suddivisione di *thread* ignota al nucleo

Esegue nell'ambiente del *thread* e viene gestita interamente a livello di servizi offerti dal sottosistema **Win32**

Gestione dei processi – 2

- I *thread* hanno vari modi per sincronizzarsi tra loro tramite **oggetti di ordinamento**
 - **Semafori binari** (**mutex**) o contatori
 - **Sezioni critiche** limitate allo spazio di indirizzamento del *thread* che le crea
 - Eventi (oggetti del kernel)
 - Thread attendono che si verifichino certi eventi
 - Manual-reset events (rilasci manuali)
 - Auto-reset events (al verificarsi dell'evento uno e uno solo viene rilasciato)

Gestione dei processi – 3

- I *thread* hanno vari modi per comunicare senza bisogno di sincronizzarsi
 - *Pipe* : canali bidirezionali come in UNIX e GNU/Linux a sequenza di *byte* **senza** struttura oppure per messaggi (sequenze **con** struttura)
 - *Mailslot* : canali unidirezionali anche su rete
 - *Socket* : come *pipe* ma per comunicazioni remote
 - **RPC** (**chiamata di procedura remota**) : per invocare procedure nello spazio di altri processi e riceverne il risultato localmente
 - **Condivisione di memoria** : usando (porzioni di) *file* mappati in memoria

Politica di ordinamento – 1

- **Ordinamento con prerilascio a priorità**
 - Effettuato da azioni **esplicite** del *thread* eseguite in modo nucleo → nessuna entità attiva dedicata di sistema
 - Thread si blocca ad un semaforo, I/O, etc.
 - Già in nucleo
 - Thread segnala un oggetto (es. fa up di semaforo)
 - Già in nucleo
 - Al completamento del proprio quanto di tempo
 - passa in modo nucleo tramite DPC per concludere l'*interrupt handler*
 - Causato da attività esterne eseguite nel contesto del *thread* corrente
 - Azioni di ordinamento programmate come **DPC** (*Deferred Procedure Call*) associate al trattamento di eventi asincroni
 - Completamento operazione I/O
 - Scadenza timer

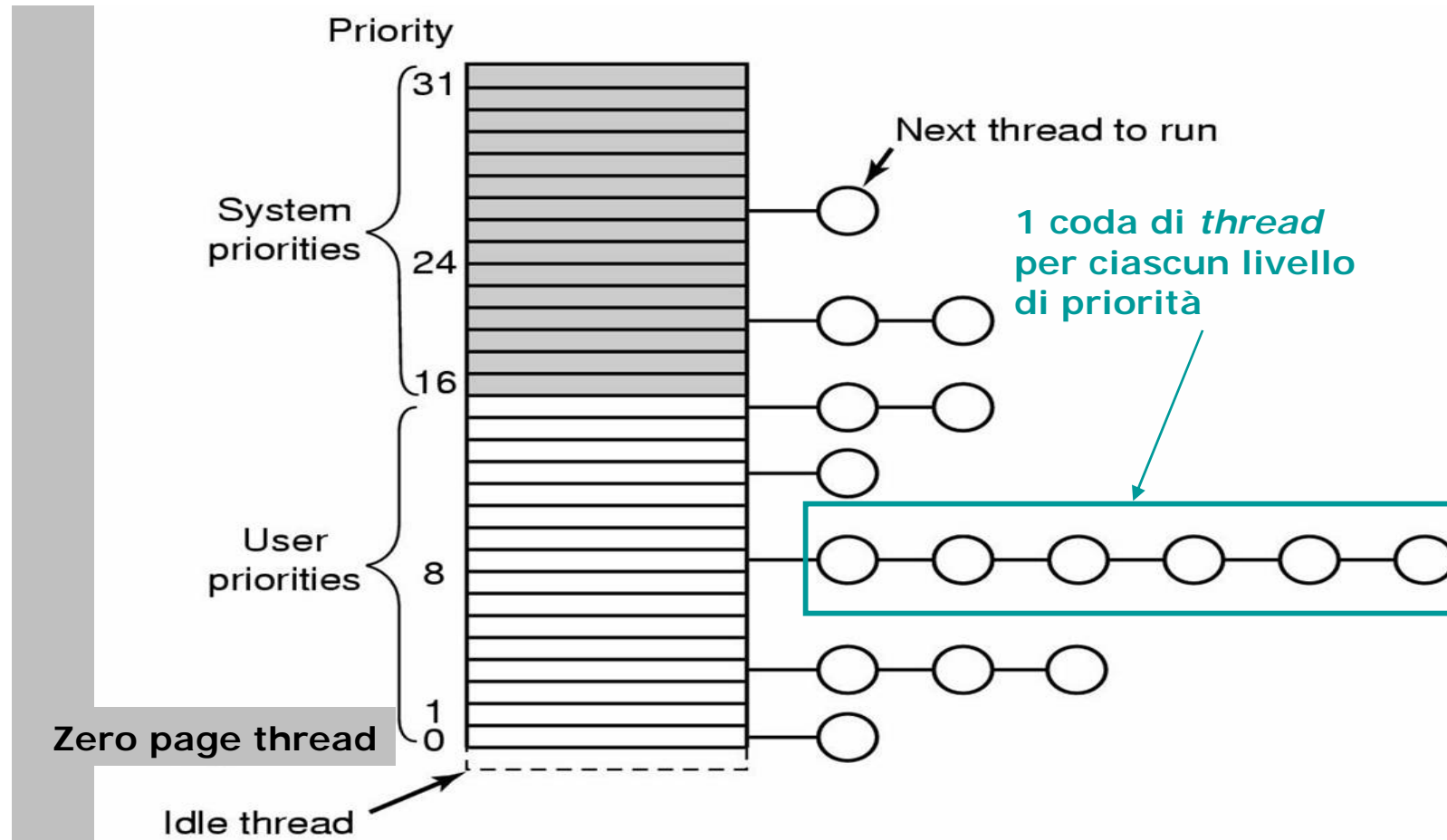
Politica di ordinamento – 2

- **6 classi di priorità per processo**
 - Realtime, high, above-normal, normal, below-normal, idle
- **7 classi di priorità per *thread***
 - Time-critical, highest, above-normal, normal, below-normal, lowest, idle
- **32 livelli di priorità (31 .. 0)**
 - Ciascuno associato a una coda di *thread* pronti
 - *Thread* non distinti per processo di appartenenza
 - 31 .. 16 priorità di sistema; 15 .. 0 priorità ordinarie
- Ricerca per priorità decrescente
- Selezione dalla testa della coda
- Priorità più elevate per processi I/O bound e interattivi

Politica di ordinamento – 3

		Win32 process class priorities					
Win32 thread priorities		Realtime	High	Above Normal	Normal	Below Normal	Idle
	Time critical	31	15	15	15	15	15
	Highest	26	15	12	10	8	6
	Above normal	25	14	11	9	7	5
	Normal	24	13	10	8	6	4
	Below normal	23	12	9	7	5	3
	Lowest	22	11	8	6	4	2
	Idle	16	1	1	1	1	1

Politica di ordinamento – 4



Politica di ordinamento – 5

- Ciascun *thread* ha una priorità **base** iniziale e una **corrente** che **varia** nel corso dell'esecuzione
 - Entro la fascia della classe di priorità del processo di appartenenza
- La priorità corrente si eleva quando il *thread*
 - Completa un'operazione di I/O
 - Per favorire maggior utilizzazione delle periferiche
 - Insieme a un ampliamento temporaneo della durata del quanto
 - Ottiene un semaforo o riceve un segnale d'evento
 - Per ridurre il tempo di attesa dei processi interattivi
- La priorità corrente decresce a ogni quanto consumato
 - Il quanto può essere più ampio se è riferito alla finestra in primo piano
- Usa una tecnica brutale per mitigare il problema di inversione di priorità
 - Un *thread* pronto non selezionato per un certo tempo riceve un incremento di priorità per 2 quanti