

7 - Caso di Studio Windows 8

Sommario

- Introduzione
- Storia
- Obbiettivi
- Architettura e Programmazione di Windows
- Meccanismi di gestione del sistema
 - Registry e gestore degli oggetti
 - Interrupt Request Levels (IRQLs)
 - Asynchronous Procedure Calls (APCs), Deferred Procedure Calls (DPCs)
- Processi e Thread
 - System Threads
 - Organizzazione
 - Scheduling dei Thread
- Gestione della memoria
 - Organizzazione e allocazione della memoria
 - Sostituzione di pagine
- File Systems
 - File System Drivers, NTFS

S. Balsamo – Università Ca' Foscari Venezia – SO.7.0

0

Obbiettivi

- Storia dei S.O. da MS-DOS a Windows
- Architettura di Windows
- Sottosistemi di Windows
- Gestione dei processi, thread, memoria in Windows 8
- File system NT in Windows 8
- Il sottosistema di I/O
- Rete e multiprocessing
- Modello di sicurezza

S. Balsamo – Università Ca' Foscari Venezia – SO.7.1

1

Introduzione

- Introduzione a Windows
 - S.O. proprietario Microsoft
 - S.O. per server, commerciale e per utenti privati
 - S.O. desktop, tablet, smartphone
 - S.O. per infrastruttura cloud computing Azure
- molte edizioni, alcuni sono diversi s.o.
 - Windows XP Home Edition desktop
 - Windows XP Professional sicurezza
 - Windows XP Tablet PC Edition supporto wireless
 - Windows XP Media Center Edition multimedia
 - Windows XP 64-Bit Edition applicazioni per molti dati
 - Windows XP 64-Bit Edition for 64-Bit Extended Systems
 - Windows 6 Vista, Windows 7 (2009), 8 (2012-2013), 10 (2016)
 - Windows Server 2008, 2012, 2016

S. Balsamo – Università Ca' Foscari Venezia – SO.7.2

2

Storia

- Quattro fasi
 - ① MS-DOS – anni '80
 - ② Windows basato su MS-DOS – anni '90
 - ③ Windows basato su NT – anni 2000
 - ④ Modern Windows – anni 2010
 - 1976 Bill Gates (studente di Harvard) e Paul Allen (programmatore a Honeywell) fondano Microsoft
 - Linguaggio BASIC – accordo per PC IBM – dal s.o. CP/M (Digital) il s.o.:
- (1) 1981 MS-DOS 1.0 (Microsoft disk operating system)
 - Indirizzamento a 16-bit
 - Modalità reale, utente singolo
 - Un solo processo per volta
 - Accesso diretto alla memoria principale
- (2) Windows basato su MS-DOS - 1985 Windows 1.0
 - Primo S.O. Microsoft GUI
 - GUI già sviluppata da Xerox negli anni '70, commercializzata dal 1981 e in Macintosh Apple nel 1983-84
 - Introduce la modalità protetta per DOS, ma sempre accesso diretto alla memoria

S. Balsamo – Università Ca' Foscari Venezia – SO.7.3

3

Storia

- (2) 1990 Windows 3.1 e Windows per Workgroups 3.1
 - Eliminata la modalità reale, introdotta la modalità 'potenziata' (*enhanced*)
 - Aggiunto il **supporto di rete** (LAN)
 - Tentativi di miglioramento della stabilità, limiti alla compatibilità a ritroso
- (3) 1993 Windows NT 3.1
 - S.O. con nuova tecnologia (*New Technology*)
 - Il processore di riferimento era Intel 860 detto N10
 - Analogie con sistema VMS di DEC (*Digital Equipment Corporation*)
 - Creata una nuova linea **aziendale**
 - Centrato su **sicurezza** e **stabilità**
 - **NTFS** (*New Technology File System*)
 - Eseguito in uno spazio protetto
 - Eliminato l'accesso diretto alla memoria
 - Alcune applicazioni multimediali meno efficienti
 - Indirizzamento a **32 bit**
 - Interfacce: estensioni a 32 bit delle API di Windows, chiamate **Win32**
 - Versioni successive con maggior portabilità per architetture MIPS e PowerPC

S. Balsamo – Università Ca' Foscari Venezia – SO.7.4

4

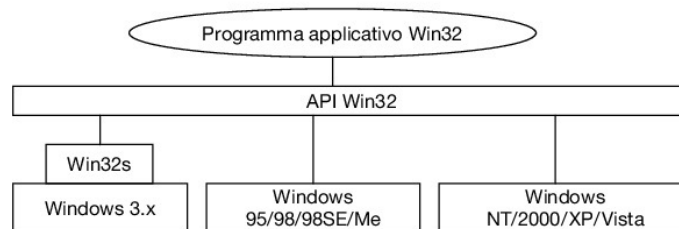
Storia

- (2) 1995 Windows 95
 - Linea **consumer**
 - Indirizzamento a 32 bit
 - Accesso diretto alla memoria principale
 - Introduzione di **DirectX**
 - Simula l'accesso diretto all'hardware tramite API
 - Multithreading, gestione dei processi
- 1998 Windows 98
 - Fornisce **Internet Explorer** nel S.O.
- 2000 Windows ME (*Millenium Edition*)
 - Ultima linea di S.O. desktop puramente consumer
 - Miglior supporto multimediale e di driver
 - Non si avvia in modalità DOS
- (3) 1996 Windows NT 4.0
 - Spostato il driver grafico nel nucleo
 - Maggior **sicurezza** e supporto alla **rete**
 - Interfaccia Windows 95

S. Balsamo – Università Ca' Foscari Venezia – SO.7.5

5

Storia



Compatibilità: uso delle API Win32 che permette ai programmi di essere eseguiti su diverse versioni di Windows

A. Tanenbaum – Modern Operating Systems

S. Balsamo – Università Ca' Foscari Venezia – SO.7.6

6

Storia

- (3) 2000 Windows 2000
 - L'ultima linea di S.O. desktop puramente **aziendale**
 - Nucleo NT 5.0
 - Plug and play
 - Migliore GUI e gestione dell'energia
 - Active Directory
 - **Database** di utenti, computer e servizi
 - **Sicurezza**, autenticazione
 - **Kerberos**
 - Consente firma singola sicura
- 2001 Windows XP
 - Unisce codici per **consumatore** e **aziende**
 - supporto a **64 bit**
 - **GUI rinnovata**

S. Balsamo – Università Ca' Foscari Venezia – SO.7.7

7

Storia

- Da Windows XP in avanti architettura 32/64 bit
- Versioni per Smartphone e Tablet
- Windows Vista (versione 6.) 2006
- Windows Server 2008
- Windows 7 (versione 6.1) 2009
 - migliori prestazioni, affidabilità e sicurezza
- (4) Windows 8 (versione 6.2) 2012
- Windows 8.1 (versione 6.3) 2013
 - migliore progettazione, modulare, codice meno ridondante
 - installazione su diversi dispositivi
- Windows 10 (versione 10.) 2016
 - integrazione fra sistema desktop e sistema per smartphone e tablet
- Windows 11 (versione 11.) 2021

S. Balsamo – Università Ca' Foscari Venezia – SO.7.8

8

Storia di Windows fino a Windows 11

Year	MS-DOS	MS-DOS based Windows	NT-based Windows	Modern Windows	Notes
1981	1.0				Initial release for IBM PC
1983	2.0				Support for PC/XT
1984	3.0				Support for PC/AT
1990		3.0			Ten million copies in 2 years
1991	5.0				Added memory management
1992		3.1			Ran only on 286 and later
1993			NT 3.1		Supported 32-bit x86, MIPS, and Alpha
1995	7.0	95	NT 3.51		MS-DOS embedded in Win 95 NT supports PowerPC
1996			NT 4.0		NT has Windows 95 look and feel
1998		98			
2000	8.0	Me	2000		Win Me was inferior to Win 98 NT supports IA-64
2001			XP		Replaced Win 98. NT supports x64
2006			Vista		Vista could not supplant XP
2009			7		Significantly improved upon Vista
2012				8	First Modern version, supports ARM
2013				8.1	Addressed complaints about Windows 8
2015–2020				10	Unified OS for multiple device form factors Rapid releases every 6 months Reached 1.3B devices
2021				11	Fresh new UI Broader application support Higher security baseline

A. Tanenbaum – Modern Operating Systems

S. Balsamo – Università Ca' Foscari Venezia – SO.7.9

9

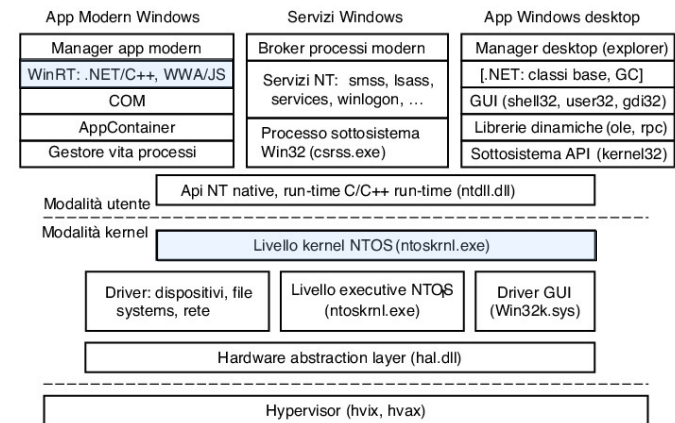
Architettura di sistema – livelli di programmazione

- Sottosistema Win32 – API NT native per le chiamate di sistema
 - Interfacce di programmazione implementate come servizi, chiamate da utente in modalità utente con RPC (*remote procedure call*)
- NTOS **nucleo** del S.O. NT che fornisce le tradizionali chiamate di sistema
- Le interfacce utente sono implementate dai sottosistemi che sono eseguiti sopra NTOS
- NT aveva tre personalizzazioni: OS/2 (*non in Windows XP*), POSIX (*non in Window 8.1*) e Win32
- Oggi tutte le applicazioni Windows sono scritte con API sopra Win32
- Es. API WinFX nel modello di programmazione .NET
- Modern Windows, da Window 8 ha introdotto anche nuove API WinRT singola applicazione alla volta a tutto schermo, interfaccia a tocco Modern Software Development Kit (MSDK) e usa comunque alcune API Win 32 incluse in MSDK

S. Balsamo – Università Ca' Foscari Venezia – SO.7.10

10

Architettura di sistema – livelli di programmazione



A. Tanenbaum – Modern Operating Systems

S. Balsamo – Università Ca' Foscari Venezia – SO.7.11

11

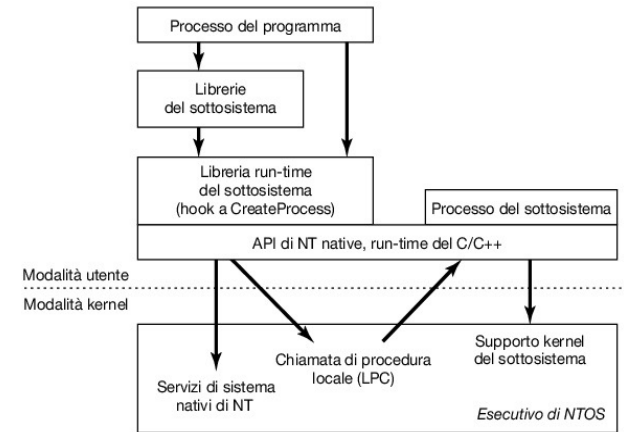
Architettura di sistema – livelli di programmazione

- Nei sistemi desktop Win32 le applicazioni sono installate da un programma che fa parte della stessa applicazione
- Le **applicazioni** moderne devono essere installate dal programma **AppStore** di Windows dallo Store on line di Micorsoft
 - Quando in esecuzione l'applicazione moderna è eseguita in una cosiddetta *sandbox* (chiamata **AppContainer**) che isola il codice per motivi di sicurezza
 - Un'applicazione accede alle risorse tramite Api WinRT che comunicano con **processi broker** che accedono ad altre risorse, e.g. file utente
- **Componenti dei sottosistemi NT**
 - **Processo** del sottosistema (un servizio), avviato da *smss.exe* (manager di sessione) a seguito della richiesta di *CreateProcess* in Win32
 - **Librerie** – funzioni di alto livello e funzioni di routine di comunicazione *stub*
 - **Hook** (agganci) a *CreateProcess* – quale sottosistema usare
 - Supporto nel **kernel**

S. Balsamo – Università Ca' Foscari Venezia – SO.7.12

12

Architettura di sistema componenti per costruire sottosistemi NT



A. Tanenbaum – Modern Operating Systems

S. Balsamo – Università Ca' Foscari Venezia – SO.7.13

13

Interfaccia di programmazione – applicazioni native NT

- **Chiamate di sistema** che Windows può eseguire, implementate nel livello Executive NTOS eseguito in modalità nucleo
- Le chiamate si riferiscono ad oggetti e restituiscono un **handle**, che possono essere usati e non possono di regola essere passati
- **Descrittore di sicurezza** degli oggetti per stabilire i diritti di accesso
- Gli oggetti in modalità nucleo hanno un nome, protezione, possono essere condivisi
- Alcuni tipi di **oggetti** utilizzati in modalità nucleo in Windows

Categoria dell'oggetto	Esempi
Sincronizzazione	Semafori, mutex, eventi, porte IPC, code di completamento dell'I/O
I/O	File, dispositivi, driver, timer
Programma	Job, processi, thread, sezioni, token
Win32 GUI	Desktop, callback delle applicazioni

S. Balsamo – Università Ca' Foscari Venezia – SO.7.14

14

Interfaccia di programmazione - oggetti

Esempio di **API nativi** che usano **handle espliciti** per gestire **oggetti** in modalità nucleo (es. processi, thread, porte IPC, sezioni – per oggetti mappati in memoria)

<code>NtCreateProcess(&ProcHandle, Access, SectionHandle, DebugPortHandle, ExceptPortHandle, ...)</code>
<code>NtCreateThread(&ThreadHandle, ProcHandle, Access, ThreadContext, CreateSuspended, ...)</code>
<code>NtAllocateVirtualMemory(ProcHandle, Addr, Size, Type, Protection, ...)</code>
<code>NtMapViewOfSection(SectHandle, ProcHandle, Addr, Size, Protection, ...)</code>
<code>NtReadVirtualMemory(ProcHandle, Addr, Size, ...)</code>
<code>NtWriteVirtualMemory(ProcHandle, Addr, Size, ...)</code>
<code>NtCreateFile(&FileHandle, FileNameDescriptor, Access, ...)</code>
<code>NtDuplicateObject(srcProcHandle, srcObjHandle, dstProcHandle, dstObjHandle, ...)</code>

Nota: in Unix si usano e accede agli **oggetti** con descrittori di file, PID, i-node in Windows tramite handle

funzionalità uniforme, implementazione unificata con il **gestore degli oggetti** centralizzato – sincronizzazione, sicurezza **spazio dei nomi** gerarchico ,

A. Tanenbaum – Modern Operating Systems

S. Balsamo – Università Ca' Foscari Venezia – SO.7.15

15

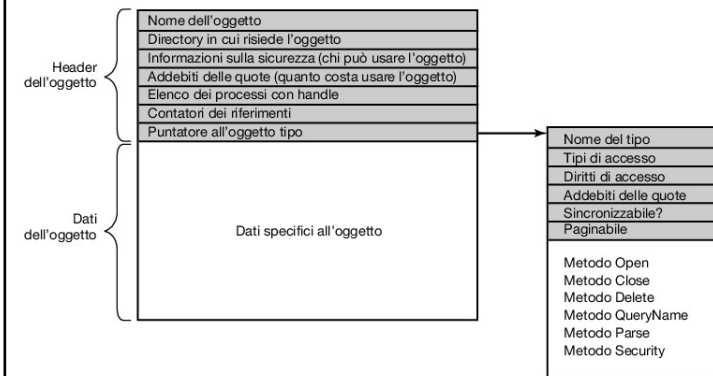
Gestore degli Oggetti

- **Oggetti** in Windows (*Object*)
 - Rappresentano **nomi** di una
 - **risorsa fisica** (es.: periferica) o
 - **risorsa logica** (es.: processo)
 - **N.B. NON** sono gli oggetti dei linguaggi di programmazione orientati ad oggetti
 - Gestiti dal **Gestore di Oggetti** (*Object Manager*)
 - Rappresentato da una **struttura dati** in memoria
 - Associati ad un **tipo**
 - Un'istanza di tipo di oggetto
 - Definisce gli **attributi** dell'oggetto
 - Definisce le **procedure** standard dell'oggetto (es.: open, close, delete)
 - Esempi di tipi di oggetti: processi, thread, pipe, file, dispositivi, ...

S. Balsamo – Università Ca' Foscari Venezia – SO.7.16

16

Gestore degli Oggetti



Struttura di un oggetto – gestito dal gestore di oggetti

A. Tanenbaum – Modern Operating Systems

S. Balsamo – Università Ca' Foscari Venezia – SO.7.17

17

Gestore degli Oggetti

- **Handles e Pointers**
 - **Pointer**
 - Puntatore referenziato per contare i riferimenti
 - **Handle**
 - Usati da processi in modalità utente e da componenti del nucleo
 - **Permette il controllo del sistema** su ciò che un thread può fare con l'oggetto (permessi)
 - Possono essere **duplicati** e **passati** ad altri processi
 - **Handle di kernel**: vi si può accedere dallo spazio di un processo, ma solo in modalità nucleo
 - Contenuti nella tabella di handle dei processi di sistema

S. Balsamo – Università Ca' Foscari Venezia – SO.7.18

18

Gestore degli Oggetti

- **Denominazione degli oggetti** (*Object naming*)
 - Un oggetto può essere **con o senza nome**
 - Oggetti con nome classificati nello spazio dei nomi dell'Object Manager
 - solo i threads del nucleo possono aprire un **handle** per un oggetto senza nome
 - **Spazio dei nomi gerarchico** (NT namespace) ed estensibile
 - Il gestore degli oggetti implementa **directory** e **link simbolici**
 - Routine di gestione **Parse**
 - Nomi permanenti, l'oggetto rimane finché non è cancellato, anche se non collegato
- **Eliminazione** di oggetti
 - Niente più **handle**: oggetti cancellati dallo spazio dei nomi
 - Non più **handle** e puntatori: oggetti cancellati dalla memoria

S. Balsamo – Università Ca' Foscari Venezia – SO.7.19

19

Interfaccia di programmazione – applicazioni native Win32

Chiamate di funzione Win32, sono chiamate **API Win32**

Implementate come procedure di libreria sopra alle chiamate di sistema native

Esempi di chiamate API Win32 e relative chiamate API NT native

Chiamata Win32	Chiamata API NT nativa
CreateProcess	NtCreateProcess
CreateThread	NtCreateThread
SuspendThread	NtSuspendThread
CreateSemaphore	NtCreateSemaphore
ReadFile	NtReadFile
DeleteFile	NtSetInformationFile
CreateFileMapping	NtCreateSection
VirtualAlloc	NtAllocateVirtualMemory
MapViewOfFile	NtMapViewOfSection
DuplicateHandle	NtDuplicateObject
CloseHandle	NtClose

A. Tanenbaum – Modern Operating Systems

S. Balamo – Università Ca' Foscari Venezia – SO.7.20

20

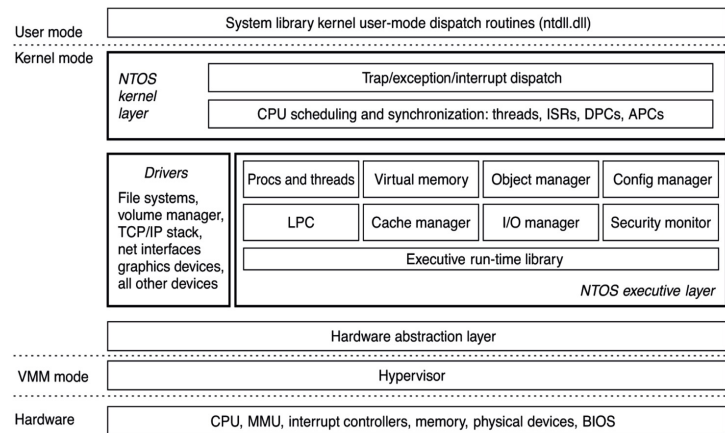
Architettura di sistema

- Progettazione del S.O.
 - Kernel NTOS** con due livelli
 - Esecutivo** – maggior parte dei servizi
 - Kernel – Scheduling** dei thread, **interrupt** dispatching
 - Meccanismi e servizi di base del sistema
 - Sincronizzazione dei thread,
 - Gestore delle **trap**, delle **interruzioni**
- Componenti
 - HAL (Hardware Abstraction Layer)**
 - Interagisce con l'hardware, gestisce i dispositivi sulla scheda madre
 - Astrae** dalle specifiche hardware che differiscono tra sistemi con la stessa architettura
 - Driver dei dispositivi**
 - Controllo dei dispositivi periferici e DMA
 - Hypervisor**
 - Livello più basso – **scheda processori virtuali** su processori fisici
 - Virtualizza hw e permette di eseguire in concorrenza diversi s.o. (m.v. chiamati partizioni)

S. Balamo – Università Ca' Foscari Venezia – SO.7.21

21

Architettura di sistema



A. Tanenbaum – Modern Operating Systems

S. Balamo – Università Ca' Foscari Venezia – SO.7.22

22

Architettura di sistema

- DLL (Dynamic Link Libraries)**
 - Librerie di collegamento dinamico
 - Moduli in modalità utente** che i processi possono collegare in modo dinamico
 - Forniscono **dati** e **funzioni** ai processi utente
 - Punti di ingresso per il nucleo per gestire eccezioni e APC: **chiamate di procedura asincrona**
- Servizi di sistema**
 - Processi speciali** eseguiti in modalità utente
 - Come i demoni in Linux: processi che vengono eseguiti in **background** costantemente
 - Es.: Task Scheduler, IPSec, Computer Browser (elenco di sistemi connessi alla rete locale), ecc

S. Balamo – Università Ca' Foscari Venezia – SO.7.23

23

Architettura di sistema

- **Livello kernel** (nel NTOS)
 - Sopra al livello executive
 - Astrazioni per la **gestione della CPU: astrazione del thread**
 - **Scheduling e sincronizzazione di thread**
 - Gestione di trap, interruzioni
 - Fornisce al livello più basso due meccanismi di sincronizzazione:
 - Oggetti di controllo – strutture dati per gestire la CPU
 - Oggetti dispatcher – oggetti esecutivi ordinari che usano una comune struttura dati per la sincronizzazione

S. Balsamo – Università Ca' Foscari Venezia – SO.7.24

24

Architettura di sistema

- **Livello Executive**
 - Sotto al livello kernel di NTOS
 - Scritto in C
 - Eseguito in modalità kernel
 - Indipendente dall'architettura (eccetto gestore della memoria)
 - **Componenti** (chiamati gestori) con strutture dati interne ed esterne
 - **Gestore di oggetti**
 - Gestore di I/O
 - Gestore dei **processi**
 - Gestore della **memoria**
 - **Memoria virtuale e paginazione su richiesta**
 - Gestore della **cache**
 - **Security** reference monitor
 - Comunicazione: LPC, notifica WNF (Window Notification Facility)
 - Gestore della configurazione: implementa il **registro di sistema**

S. Balsamo – Università Ca' Foscari Venezia – SO.7.25

25

Meccanismi di gestione del sistema

- Ambiente in cui vengono eseguite le componenti di *Windows*
 - Come sono conservati e recuperati i **dati (registry)**
 - **Oggetti**
 - **Priorità di interrupt**
 - **Interrupt software** e gestione delle priorità
 - **Thread** di sistema

S. Balsamo – Università Ca' Foscari Venezia – SO.7.26

26

Oggetti gestiti nell'Executive – gestore di oggetti

Tipo	Descrizione
Processo	Processo utente
Thread	Thread all'interno di un processo
Semaforo	Semaforo contatore usato per la sincronizzazione tra processi
Mutex	Semaforo binario usato per accedere a una regione critica
Evento	Oggetto di sincronizzazione con stato persistente (segnalato/non segnalato)
Porta ALPC	Meccanismo per il passaggio di messaggi tra processi
Timer	Oggetto che consente a un thread di dormire per un intervallo di tempo fissato
Coda	Oggetto usato per la notifica di completamento di I/O asincrono
File aperto	Oggetto associato a un file aperto
Token di accesso	Descrittore di sicurezza per un oggetto
Profilo	Struttura dati usata per profilare l'uso della CPU
Sezione	Oggetto usato per rappresentare file mappabili
Chiave	Chiave del registro, usata per attaccare il registro allo spazio dei nomi del gestore degli oggetti
Oggetto directory	Directory per raggruppare gli oggetti all'interno del gestore degli oggetti
Link simbolico	Si riferisce a un altro oggetto del gestore degli oggetti attraverso il path name
Dispositivo	Device object di I/O per un dispositivo fisico, bus, driver, o istanza del volume
Driver del dispositivo	Ciascun driver del dispositivo caricato ha il proprio oggetto

A. Tanenbaum – Modern Operating Systems

S. Balsamo – Università Ca' Foscari Venezia – SO.7.27

27

Registro di sistema - Registry

- **File speciale - database centrale** che memorizza i **dati di configurazione**
- accessibili a tutti i processi in modalità nucleo
 - dati di **utente** (es. impostazioni desktop)
 - dati di **sistema** (es. associazioni dei file)
 - dati **hardware** (es. driver dei dispositivi collegati)
 - dati per l'**applicazione** (es. voci di menù usate, preferenze)
- **Struttura** logica memorizzata come
 - **Albero** i cui nodi sono le chiavi
 - Sottochiavi e valori, nome e dato
 - Chiavi predefinite (es.: HKEY_LOCAL_MACHINE radice dei dati di configurazione dell'elaboratore locale)
 - Utilizzando il Registro di sistema
 - I thread navigano la struttura dell'albero passando dalle chiavi alle sottochiavi
 - GUI per utenti: `regedit` per esplorare le directory (chiavi) e dati (valori)

S. Balsamo – Università Ca' Foscari Venezia – SO.7.28

28

Registro di sistema

- Windows amministratore del registro
 - **Gestore delle configurazioni** (*Configuration Manager*) componente executive
 - I dati memorizzati in insiemi di **hives** (alveari) – porzioni dell'albero del registro, volumi - , ciascun hive memorizzato in un file
nella directory `C:\Windows\system32\config\` del volume di boot
 - I programmatori di Win32 possono accedere al registro, con chiamate, esempi

Funzione delle API Win32	Descrizione
RegCreateKeyEx	Crea una nuova chiave di registro
RegDeleteKey	Cancella una chiave di registro
RegOpenKeyEx	Apri una chiave per ottenere un handle a essa
RegEnumKeyEx	Calcola le sottochiavi subordinate alla chiave dell'handle
RegQueryValueEx	Cerca tra i dati per trovare un valore all'interno di una chiave

S. Balsamo – Università Ca' Foscari Venezia – SO.7.29

29

Registro di sistema

Hive del registro di sistema di Windows

File hive	Nome montato	Uso
SYSTEM	HKLM TEM	Informazioni di configurazione del SO, usate dal kernel
HARDWARE	HKLM DWARE	Hive nella memoria che registra l'hardware trovato
BCD	HKLM BCD*	Base di dati della configurazione di boot
SAM	HKLM	Informazioni dell'account dell'utente locale
SECURITY	HKLM URITY	Account di lsass e altre informazioni sulla sicurezza
DEFAULT	HKEY_USERS.DEFAULT	Hive di default per nuovi utenti
NTUSER.DAT	HKEY_USERS<user id>	Hive specifico dell'utente, mantenuto nella directory home
SOFTWARE	HKLM TWARE	Classi applicative registrate da COM
COMPONENTS	HKLM NENTS	Manifesti e dipendenze per i componenti del sistema

A. Tanenbaum – Modern Operating Systems

S. Balsamo – Università Ca' Foscari Venezia – SO.7.30

30

Livelli di richiesta di Interrupt

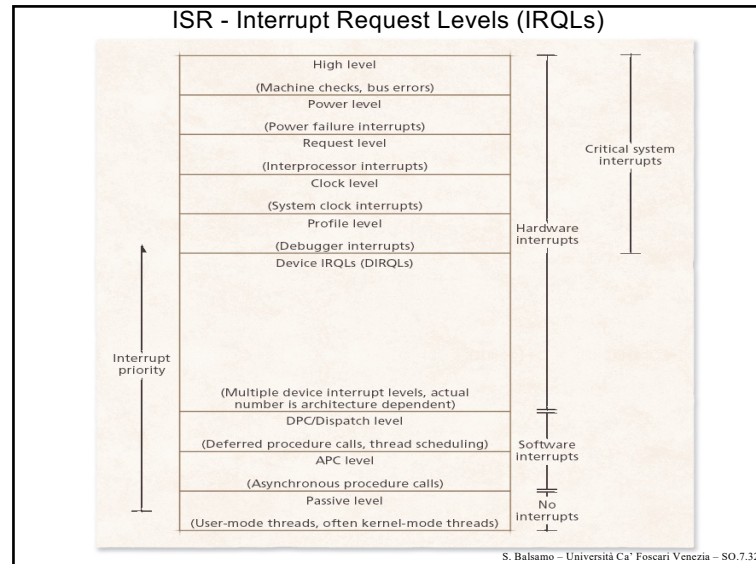
- **Interrupt Service Routine (ISR)**
 - Per la gestione delle interruzioni
 - Parametro di **priorità di interruzione**
 - Il processore viene sempre in esecuzione con riferimento ad un IRQ (Interrupt Request Level)
 - Il processore maschera l'interrupt con $IRQL \leq IRQL$ corrente
- **Livelli di priorità**
 - **Passivo: nessuna** interrupt in fase di elaborazione
 - APC: **chiamate di procedura asincrona** nel contesto di un thread
 - DPC/dispatch: **chiamate di procedura differite**, scheduling dei thread – nel contesto di una CPU
 - ISR di dispositivo: interrupt di periferica (dispositivo)
 - Interrupts di sistema **critici**: Profilo (debugger), Clock, Request (interprocessor), Power, High

No Interrupt

Interrupt
softwareInterrupt
Hardware

S. Balsamo – Università Ca' Foscari Venezia – SO.7.31

31



32

Thread di Sistema

- Utilizzato da **componenti in modalità nucleo** che devono eseguire azioni non in risposta a una **richiesta** dell'utente
 - gestore della **cache** per lo scaricamento delle pagine di cache modificate (*dirty*)
 - driver di periferica** che non si può soddisfare tutti gli interrupt a un IRQL elevato (es.: perché deve accedere ai dati paginabili)
- Due possibili approcci

1 - Thread del nucleo

- Creato da componenti in modalità nucleo
- In genere appartengono al **processo di sistema**
- In generale, si comportano e sono schedulati come thread utente
- Solitamente eseguiti a livello IRQL passivo

S. Balsamo – Università Ca' Foscari Venezia – SO.7.33

33

Thread di Sistema

2 - System worker thread

- Forniti dal microkernel all'inizio o creati in base alla quantità di richieste
- Dormienti finché un componente in modalità nucleo accoda una richiesta di lavoro
- 3 tipi (diverse priorità): delayed, critical, hypercritical

S. Balsamo – Università Ca' Foscari Venezia – SO.7.34

34

Gestione di Processi e Thread

- Processi**
 - Contesto di esecuzione** (spazio di indirizzamento virtuale, attributi,...)
 - Codice** del programma
 - Risorse**
 - Threads** associati
- Threads**: unità di esecuzione
- Processi e threads sono **oggetti**

S. Balsamo – Università Ca' Foscari Venezia – SO.7.35

35

Gestione di Processi e Thread

Nome	Descrizione	Note
Job	Raccolta di processi che condividono quote e limiti	Usato in AppContainers
Processo	Contenitore per le risorse	
Thread	Entità schedulata dal kernel	
Fibra	Thread leggero gestito interamente nello spazio utente	Usato raramente
Thread pool	Modello di programmazione orientato ai task	Costruito sui thread
Thread in modalità utente	Astrazione che consente lo scambio di thread in modalità utente	Estensione dei thread

A. Tanenbaum – Modern Operating Systems

S. Balsamo – Università Ca' Foscari Venezia – SO.7.36

36

Organizzazione di Processi e Thread

- Strutture dati che descrivono un **processo**
 - **Blocco di ambiente del Processo (PEB)**:
 - informazioni del **processo** disponibili ai thread utente
 - DLL (Librerie) collegati al processo, informazioni heap, ...
 - Memorizzato nello spazio del processo
 - **Blocco EPROCESS** (usato principalmente da **executive**):
 - Descrive un processo
 - ID del processo, token di accesso, tabella handle, etc.
 - Contiene un blocco KPROCESS
 - Punta a PEB
 - Memorizzato in una linked list
 - **Blocco KPROCESS** (info usate principalmente dal **kernel**):
 - informazioni di **scheduling**
 - informazioni di **sincronizzazione**

S. Balsamo – Università Ca' Foscari Venezia – SO.7.37

37

Organizzazione di Processi e Thread

- Strutture dati che descrivono un **thread**
 - **Thread environment block (TEB)**
 - informazioni del **thread** disponibili ai thread utente
 - Sezioni critiche che appartengono al thread, stack information, ...
 - Memorizzato nello spazio del processo al quale il thread appartiene
 - Memoria per il thread (Thread local storage)
 - Campi per Win32
 - **Blocco ETHREAD** (usato principalmente da **executive**)
 - Descrive un thread
 - ID del processo, richieste I/O sospese, indirizzo di inizio del thread
 - Contiene un blocco KTHREAD
 - Punta al blocco EPROCESS del processo del thread
 - **Blocco KTHREAD** (usato principalmente dal **kernel**)
 - Priorità di scheduling, stato del thread, ...
 - Punta al blocco TEB

S. Balsamo – Università Ca' Foscari Venezia – SO.7.38

38

Organizzazione di Processi e Thread

- I thread che appartengono ad un processo condividono lo **spazio di indirizzamento virtuale**
- **Memoria locale del Thread (TLS)**
 - Area dove i **threads** possono memorizzare **dati locali** (es.: per una DLL)
 - Un thread può allocare un indice TLS per il processo
 - L'indice TLS contiene uno *slot TLS* per ogni thread di un processo, puntatore alla locazione dei dati locali
 - Il processo può avere **molte indici** TLS per vari scopi (es. varie DDL e sottosistema ambiente)

S. Balsamo – Università Ca' Foscari Venezia – SO.7.39

39

Organizzazione di Processi e Thread

- Creazione e terminazione di processi
 - Creazione di processi
 - chiamata di funzione API
 - Processo genitore e figlio sono **indipendenti**, eccetto che
 - Possibile **ereditarietà** per attributi del processo figlio dal processo padre (es. *handle*, variabili di ambiente)
 - Un **processo** ha un **thread principale** che può crearne altri interni al processo
 - Terminazione di processi
 - Se tutti i thread del processo terminano
 - Un thread di un processo dirige la terminazione
 - L'utente che possiede il processo si disconnette
 - Terminazione di processi genitore e figlio sono indipendenti

S. Balsamo – Università Ca' Foscari Venezia – SO.7.40

40

Organizzazione di Processi e Thread

- Jobs
 - Gruppo di **processi** trattati insieme come una **unità**
 - **Gestisce le risorse** consumate da questi processi (es.: tempo di CPU, consumo di memoria, ...) può vincolare con token limitati
 - **Termina tutti** i processi in una sola volta
 - Ogni processo può far parte di un solo job
 - Attributi (priorità base, sicurezza, dimensione del *working set* e memoria virtuale,...) del job ereditati dal processo
- Fibers
 - Unità di esecuzione (come un **thread**), possono essere **schedulati** dal thread che li crea, non dal kernel
 - Crea allocando uno stack e struttura dati in modalità utente
 - Eseguiti nel contesto del thread
 - Il thread si deve convertire in una fibra per creare fibre
 - Minor overhead di cambio contesto fra fibre
 - Usato per il *porting*
 - memorizzazione locale di fibra (*Fiber local storage, FLS*), come TLS

S. Balsamo – Università Ca' Foscari Venezia – SO.7.41

41

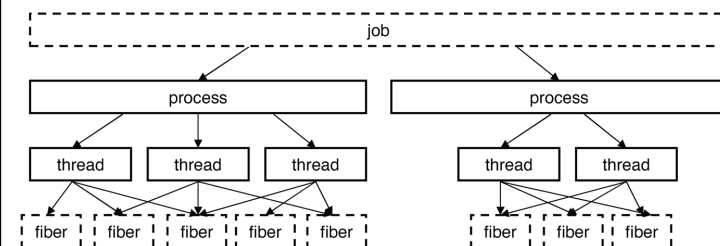
Organizzazione di Processi e Thread

- Thread pooling
 - Thread di lavoro che dormono in attesa di lavoro
 - Ogni **processo** riceve un **pool di thread**
 - Utile in determinate situazioni
 - Tipicamente per processi serventi che devono soddisfare le richieste dei clienti
 - I/O asincrono
 - Combinazione di molti thread che dormono la maggior parte del tempo
 - Overhead di memoria e meno controllo per il programmatore
 - Funzionalità thread pool di Win32
 - Se un thread si blocca su una risorsa, il thread non può essere riassegnato ad un altro

S. Balsamo – Università Ca' Foscari Venezia – SO.7.42

42

Organizzazione di Processi e Thread



43

43

Scheduling di Thread

- Per ogni thread visto a livello utente
 - un thread eseguito in modalità kernel
 - un thread eseguito in modalità utente
- **User-mode scheduling** funzione di attivazione dello scheduler
 - Per passare tra thread utente senza attivare il kernel
 - Funzionalità di basso livello da usare nelle librerie run-time e applicazioni
- API Win32 per la gestione dei thread, fra cui
 - SetPriorityClass - imposta la classe di priorità dei thread del processo
 - SetThreadPriority - imposta la classe di priorità relativa di un thread

S. Balsamo – Università Ca' Foscari Venezia – SO.7.44

44

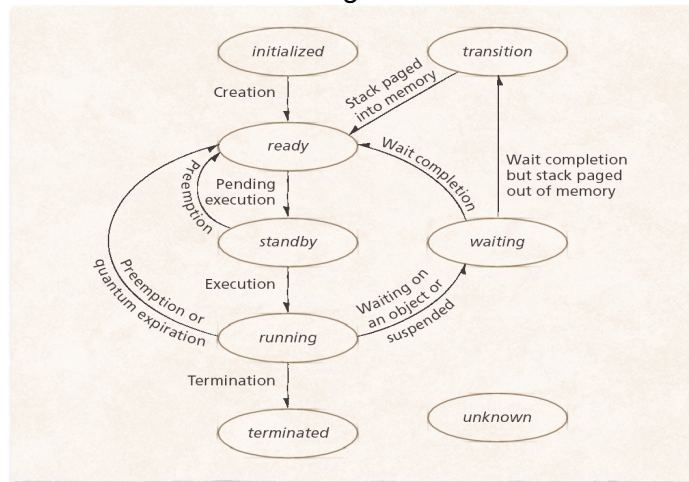
Scheduling di Thread

- **Stati** del Thread
 - *Initialized*
 - *Ready* (eseguibile)
 - *Standby* (eseguibile e schedulato)
 - *Running* (in esecuzione)
 - *Waiting*
 - *Transition* (lo stack è stato spostato dalla memoria)
 - *Terminated* (terminato ma non cancellato)
 - *Unknown*
- **Transizioni**

S. Balsamo – Università Ca' Foscari Venezia – SO.7.45

45

Scheduling di Thread



S. Balsamo – Università Ca' Foscari Venezia – SO.7.46

46

Scheduling di Thread

- Thread sono **schedulati** senza considerare i processi
- **32 livelli di priorità**
 - 0 priorità minima
 - 31 priorità massima
- **32 code ready**
- **Round-robin** disciplina di coda per code ready non vuote a partire dalla massima priorità

S. Balsamo – Università Ca' Foscari Venezia – SO.7.47

47

Scheduling di Thread

Mappe delle priorità dei thread Win32 nelle priorità di Windows

	Classi di priorità dei processi Win32					
	Real-time	Alta	Sopra il normale	Normale	Sotto il normale	Inattivo
Time critical	31	15	15	15	15	15
La più alta	26	15	12	10	8	5
Sopra il normale	25	14	11	9	7	5
Normale	24	13	10	8	6	4
Sotto il normale	23	12	9	7	5	3
La più bassa	22	11	8	6	4	2
Inattivo	16	1	1	1	1	1

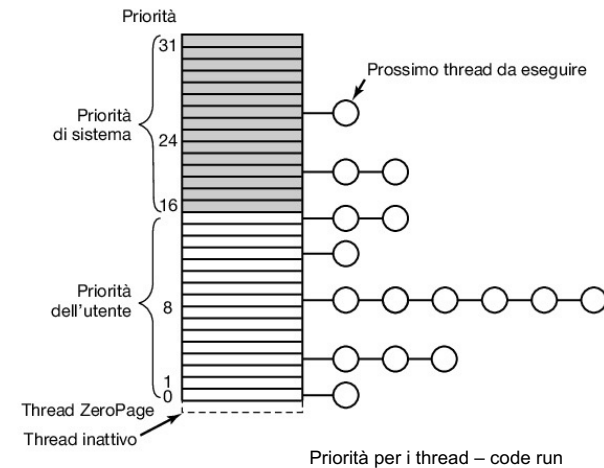
↓
Priorità dei thread Win32

A. Tanenbaum – Modern Operating Systems

S. Balsamo – Università Ca' Foscari Venezia – SO.7.48

48

Scheduling di Thread



A. Tanenbaum – Modern Operating Systems

S. Balsamo – Università Ca' Foscari Venezia – SO.7.49

49

Thread Scheduling

- **Prelazione**
 - Può accadere quando:
 - Cambia la priorità del thread
 - Il thread entra nello stato *ready*
 - Il thread esce nello stato *running*
 - I thread che subiscono il preilascio tornano in coda ready
 - Thread *real-time*: ottengono un nuovo quanto
 - Altri thread: non si riassegna il quanto

S. Balsamo – Università Ca' Foscari Venezia – SO.7.50

50

Thread Scheduling

- **Determinazione della priorità**
 - Thread *dinamici*: livelli di priorità 0-15
 - Thread *in tempo reale*: livelli di priorità 16-31 - statica
- **Processo**
 - **classe base di priorità**
 - Determina un **intervallo** limitato per le priorità di base dei suoi thread
 - Sei classi base
 - idle, below normal, normal, above normal, high, real-time
- **Thread**
 - **Livello base di priorità**
 - idle, lowest, below normal, normal, above normal, highest, real-time
 - Determina l'**esatta** priorità di base all'interno di una classe di priorità
 - Priorità di base = Classe base di priorità + livello base di priorità

S. Balsamo – Università Ca' Foscari Venezia – SO.7.51

51

Thread Scheduling

- Determinazione della **priorità**
 - Cambiamenti di priorità
 - Solo per thread **dinamici**
 - **Aumento** delle priorità
 - All'uscita dallo stato *wait*
 - Window riceve un input da utente
 - Non è stato eseguito da molto tempo
 - **Riduzione** priorità
 - Non può diminuire oltre la priorità di base
 - Decremento unitario se il thread viene eseguito per l'intero quanto
 - La priorità torna al livello precedente dopo un quanto nel caso in cui la priorità del thread è aumentata perché non era stato eseguito da molto tempo

S. Balsamo – Università Ca' Foscari Venezia – SO.7.52

52

Thread Scheduling

- Supporta anche una modalità di Scheduling interattiva per Terminal server
 - protocollo **Remote Desktop Protocol**
- Implementa anche un algoritmo di Scheduling Fair-share
 - **Dynamic Fair Share Scheduling**
 - gruppi con diversi vincoli di risorse assegnate, priorità del gruppo

S. Balsamo – Università Ca' Foscari Venezia – SO.7.53

53

Thread Scheduling

- Scheduling del **Multiprocessor**
 - **Maschera di affinità** insieme di processori dove poter eseguire il thread
 - **Ultimo processor** = processore sul quale il thread è stato eseguito l'ultima volta
 - **Processore ideale**
 - Massimizzare il parallelismo – threads in relazione, diversi processori ideali
 - Massimizzare cache hits - threads in relazione, stesso processore ideale
 - Dispatcher sceglie nella coda run non vuota con massima priorità basandosi sul tempo attesa, ultimo processore e processore ideale

S. Balsamo – Università Ca' Foscari Venezia – SO.7.54

54

Gestore della Memoria

- **Virtual memory manager** (VMM)
 - Componente **Executive** responsabile della gestione della memoria
- Meccanismi di ottimizzazione:
 - **Copy-on write**
 - **Allocazione lazy** (pigra)
 - Evita l'allocazione di memoria (pagine e righe della tabella delle pagine) fino a quando non è necessario
- **Prefetching**
 - Quando VMM fa una operazione di I/O, sposta le pagine dal disco alla memoria principale prima che siano necessarie
- **Pagefile**
 - Memorizza le **pagine** che non entrano nella memoria principale
 - Windows XP supporta fino a 16 file di pagefiles

S. Balsamo – Università Ca' Foscari Venezia – SO.7.55

55

Organizzazione della Memoria

- Spazio di indirizzamento virtuale a 32 bit
 - Windows a 64-Bit Edition dispone di spazi per indirizzi a 64 bit
 - 4 GB indirizzo spazio virtuale per processo
- spazio utente vs spazio di sistema
 - Un processo può accedere solo allo spazio utente
 - VMM memorizza le tabelle delle pagine e altri dati nello spazio di sistema
 - 2 GB di spazio utente, 2 GB di spazio di sistema (per componenti in modalità nucleo)
- pagine a **dimensione fissa** 4KB (in alcuni casi anche 2MB)

S. Balsamo – Università Ca' Foscari Venezia – SO.7.56

56

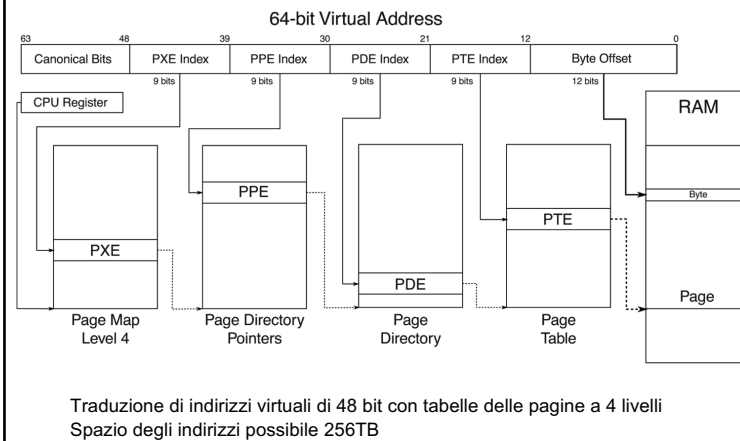
Organizzazione della Memoria

- **mapa della memoria** su due livelli
- Memorizza la posizione delle pagine in memoria e su disco
 - **Tabella di Directory di Pagina (PDE)**
 - Le **righe** di PDE indicano **tabella delle pagine**
 - Una tabella di directory di pagine **per processo**
 - Indicata nella posizione nel **registro** di directory di pagina
 - **Tabella delle pagine (PTE)**
 - Le righe della tabella pagine puntano ai page frame o locazione su disco
 - **Page frame**
 - Contiene una pagina di dati
- TLB (*Translation Look-aside Buffer*) accelera la conversione degli indirizzi

S. Balsamo – Università Ca' Foscari Venezia – SO.7.57

57

Organizzazione della Memoria

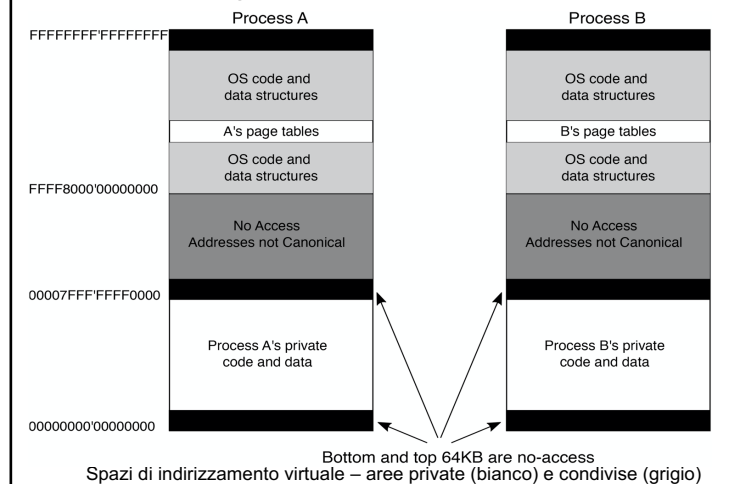


A. Tanenbaum – Modern Operating Systems

S. Balsamo – Università Ca' Foscari Venezia – SO.7.58

58

Organizzazione della Memoria



A. Tanenbaum – Modern Operating Systems

S. Balsamo – Università Ca' Foscari Venezia – SO.7.59

59

Organizzazione della Memoria

- PTE può puntare a dati in memoria principale o in un pagefile
 - 4 bit determinano quale dei 16 pagefile
 - 20 bit offset nel pagefile o frame in memoria
 - Indirizzano fino a 1.048.576 pagine virtuali
- PTE - 5 bit di protezione
 - Read
 - Write
 - Execute
 - Copy-on-write
 - Raise exception on access
- PTE - 4 bit di pagefile
 - Solo per pagine su disco
- PTE - 3 bit di stato
 - Solo per pagine in memoria
 - Valida/Modificata/Transizione

S. Balsamo – Università Ca' Foscari Venezia – SO.7.60

60

Organizzazione della Memoria

- Stati della pagina degli indirizzi virtuali
 - Non valida
 - non è mappata in alcun oggetto di sezione di memoria, se riferita si ha errore
 - Riservata
 - indirizzi che non saranno mai allocati dal gestore per altro
 - Committed
 - è mappata su una pagina virtuale

S. Balsamo – Università Ca' Foscari Venezia – SO.7.61

61

Organizzazione della Memoria

- Pagine grandi (larghe)
 - VMM tratta molte pagine come una pagina
 - Insieme di pagine contigue
 - Utile quando si usa lo stesso insieme ripetutamente
 - Un sola riga in TLB
- Restrizioni
 - Le pagine devono essere consecutive sia nella memoria virtuale e sia in memoria fisica
 - Le pagine grandi consentono sempre l'accesso in lettura e scrittura
 - Dimensione minima di solito 2MB
 - Dimensioni deve essere multiplo di un minimo (dipende dalla specifica piattaforma)

S. Balsamo – Università Ca' Foscari Venezia – SO.7.62

62

Organizzazione della Memoria

- Pagine Copy-on-write
 - I processi condividono i page frame in modo trasparente
 - Allocazione pigra
- Tabelle delle pagina prototipo
 - Abilita pagine copy-on-write
 - La PTE (tabella delle pagine) per una pagina copy-on-write punta alla tabella di pagina prototipo (PPTE)
 - Le righe della PPTE puntano ai page frame che contengono pagine copy-on-write
 - Risparmio di memoria
 - Un livello in più di indirizzamento

S. Balsamo – Università Ca' Foscari Venezia – SO.7.63

63

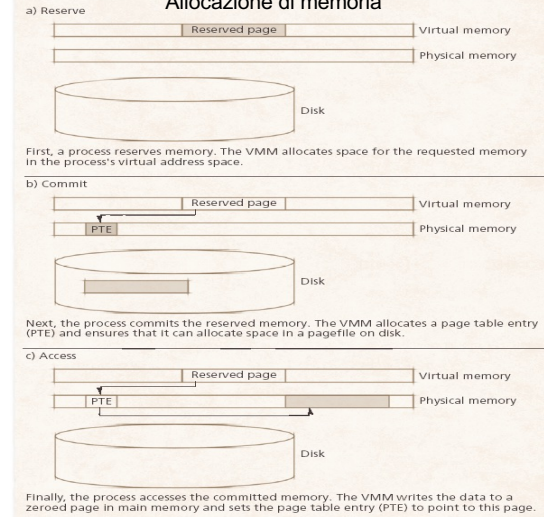
Allocazione di memoria

- 3 stadi di **allocazione di memoria**
- Per usare solo lo spazio necessario
 - **Reserve**
 - Un processo riserva lo spazio nel proprio spazio di indirizzamento virtuale
 - **Commit**
 - Un processo pronto a scrivere esegue una commit, VMM crea la tabella delle pagine e verifica la fattibilità nella memoria o pagefile
 - **Access**
 - Processo è pronto ad accedere alla memoria, VMM scrive i dati in una pagina azzerata e aggiorna la tabella delle pagine

S. Balsamo – Università Ca' Foscari Venezia – SO.7.64

64

Allocazione di memoria



S. Balsamo – Università Ca' Foscari Venezia – SO.7.65

65

Allocazione di memoria

- **Ottimizzazioni**
 - Vietate le richieste must-succeed (richieste imperative che venivano possibilmente soddisfatte, in versioni precedenti)
 - Regolazione dell'I/O (*throttling*)
 - se scarseggiano i page frame il sistema gestisce **una pagina** alla volta
 - il sistema rallenta e più tollerante ai guasti
- **Page frame database**
 - Traccia lo **stato** della memoria
 - **Liste delle pagine**, informazione di stato di ogni **page frame**
 - Otto stati, una lista per ogni stato (libere, azzerate,...)

S. Balsamo – Università Ca' Foscari Venezia – SO.7.66

66

Allocazione di memoria

Frame State	Definition
Valid	Page belongs to a process's working set and its PTE is set to valid.
Transition	Page is in the process of being transferred to or from disk.
Standby	Page has just been removed from a process's working set; its PTE is set to invalid and in transition.
Modified	Page has just been removed from a process's working set; it is not consistent with the on-disk version. The VMM must write this page to disk before freeing this page. The PTE of this page is set to invalid and in transition.
Modified No-Write	Page has just been removed from a process's working set; it is not consistent with the on-disk version. The VMM must write an entry to the log file before freeing this page. The PTE of this page is set to invalid and in transition.
Free	Page frame does not contain a valid page; however it might contain an invalid page that has no PTE and is not part of any working set.
Zeroed	Page frame is not part of any working set and all of its bits have been set to zero. For security reasons, only zeroed page frames are allocated to processes.
Bad	Page frame has generated a hardware error and should not be used.

Stati dei
page frame

S. Balsamo – Università Ca' Foscari Venezia – SO.7.67

67

Allocazione di memoria

- **Lista delle pagine**
 - Una lista pagina per ogni possibile stato di pagina in frame
 - Linked list
 - accesso rapido
- **Descrittore di indirizzi virtuali** (VAD, *Virtual Address Descriptor*)
 - Ogni VAD descrive l'intervallo di memoria virtuale allocata
 - Gestione dello spazio di indirizzamento dei processi
- **Cluster**
 - Spazio disco diviso in **gruppi di pagine** chiamate cluster, parti dello stesso file
 - Al massimo un file per cluster
 - Uso della località per allocare le pagine di un cluster
- **Paginazione Cluster a richiesta**
 - Windows XP precarica interi cluster quando il processo chiede una pagina

S. Balsamo – Università Ca' Foscari Venezia – SO.7.68

68

Allocazione di memoria

- **Prefetching**
 - Obiettivo: **ridurre il tempo** per caricare le applicazioni
 - Prefetcher Logico
 - Responsabile del prefetching
 - Registra tutti gli accessi di memoria durante
 - l'avvio di Windows
 - start up di applicazioni
 - prefetch delle prossime pagine richieste
 - di solito durante l'inizializzazione del dispositivo
 - Start up più veloce

S. Balsamo – Università Ca' Foscari Venezia – SO.7.69

69

Sostituzione di Pagina

- **Working set**
 - Insieme di pagine di un processo attualmente nella memoria principale
 - Le altre pagine del processo sono in un *pagefile*
- **Balance set manager**
 - Responsabile per la scelta del working set
 - Se necessario sposta pagine del working set nel pagefile per vari processi
- **Least Recently Used Localizzata**
 - Simile a LRU
 - Localizzato sul singolo processo
 - VMM assegna un intervallo [min,max] del working set ad ogni processo

S. Balsamo – Università Ca' Foscari Venezia – SO.7.70

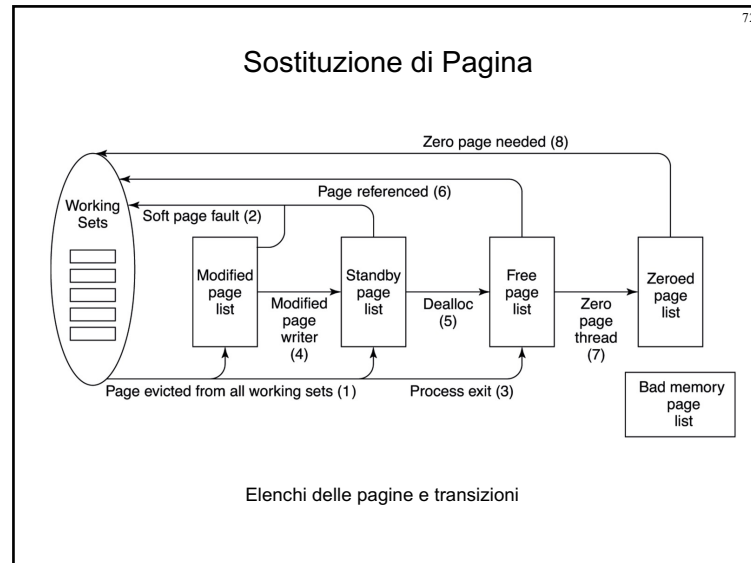
70

Sostituzione di Pagina

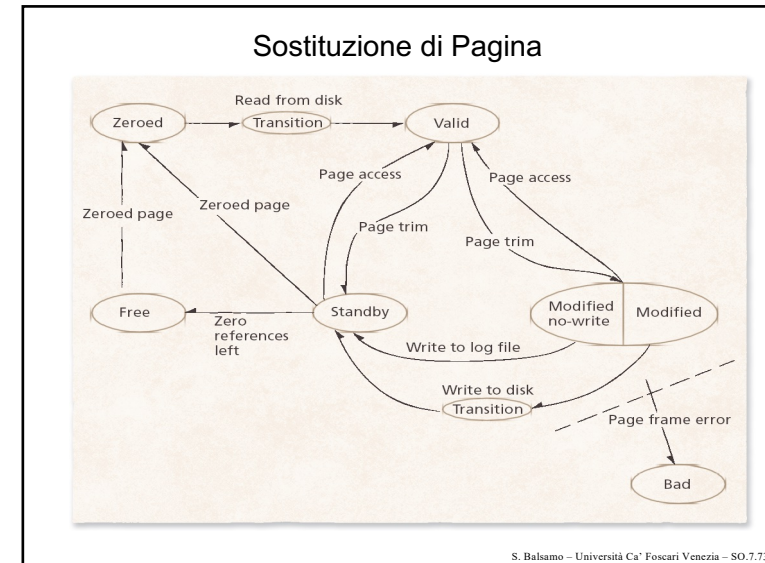
- **LRU Localizzata**
 - Sposta tutte le pagine sopra il massimo working set
 - Page *trimming* – **ridimensionamento** delle pagine
 - Mette le pagine eliminate in
 - Lista di **Pagine Standby** (in attesa)
 - Lista di **Pagine modificate**
 - Lista di **Pagine modificate e No-Write** (da non scrivere)
 - Se il processo chiede la pagina la considera nella lista delle pagine valide
 - Altrimenti la sposta nella lista delle pagine libere
 - Imposta il bit di stato a 'non valido' nella tabella delle pagine libere
 - Sposta le pagine nella lista di pagine azzerate
 - Alloca la pagina azzerata al processo che richiede nuova pagina

S. Balsamo – Università Ca' Foscari Venezia – SO.7.71

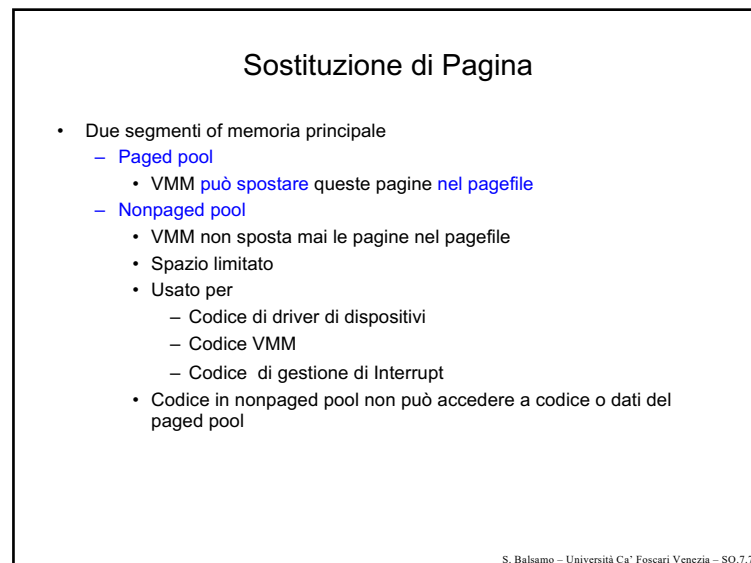
71



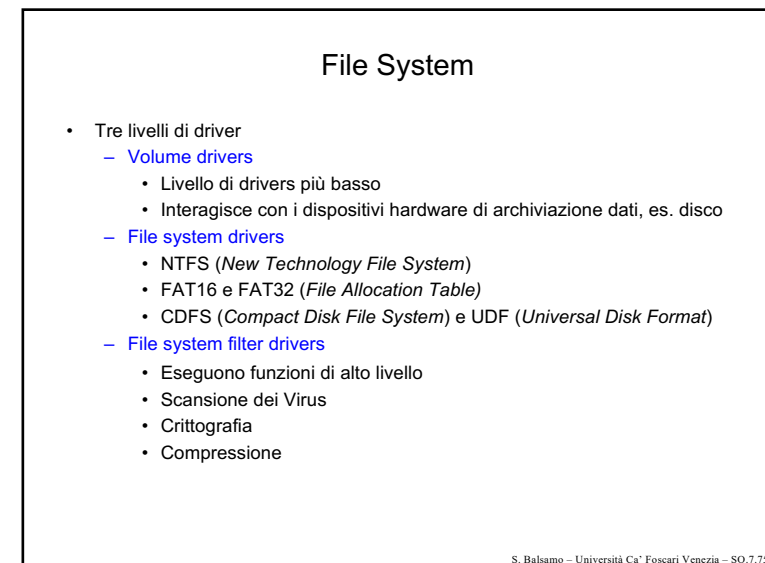
72



73



74



75

File System Drivers

- Implementazione di un file system
 - **Driver del file system locale**
 - Interagisce con i dispositivi hardware collegati al computer
 - Disco rigido
 - DVD
 - **Driver di sistema di file remoto**
 - Interagisce con driver del file system sul computer remoto
 - Utilizza protocolli di rete

S. Balsamo – Università Ca' Foscari Venezia – SO.7.76

76

File System Drivers

- Tipico flusso per una **richiesta di Disk I/O**
 - Thread utente invia una richiesta tramite API dei sottosistemi
 - Object manager passa il puntatore del file al driver del file system
 - Il driver di file system passa richiesta al driver del dispositivo
 - La richiesta alla fine raggiunge disco
 - che esegue la richiesta di I/O

S. Balsamo – Università Ca' Foscari Venezia – SO.7.77

77

NTFS

- File system nativo di Windows
- Più sicuro di FAT
- **Scala** bene a dischi di grandi dimensioni
 - dimensione dei cluster dipende dalla dimensione del disco
 - puntatori a file a 64 bit
 - Può indirizzare fino a 16 esabyte (miliardi di GB) di disco
- Flussi di dati multipli
- **Compressione e crittografia**
- Facile navigazione

S. Balsamo – Università Ca' Foscari Venezia – SO.7.78

78

NTFS

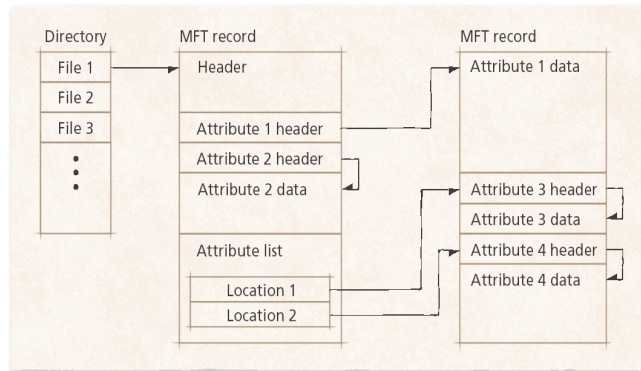
- **Master File Table (MFT)**
 - Ogni file ha una riga MFT
 - La riga MFT è formata da uno o più record
 - Tutte le informazioni dei file memorizzati come **attributi**
- Attributo **residente**
 - Memorizzato interamente dentro una riga MFT
- Attributo **non residente**
 - Intestazione memorizzata all'interno di una riga MFT
 - I dati memorizzati altrove sul disco
 - MFT registra la posizione degli attributi con tre numeri: VCN, LCN, Run

S. Balsamo – Università Ca' Foscari Venezia – SO.7.79

79

NTFS

Riga del Master File Table (MFT) per un file esempio



S. Balsamo – Università Ca' Foscari Venezia – SO.7.80

80

NTFS

- VCN (Virtual cluster number)
 - numero di cluster virtuale
- File diviso in cluster
- VCN indica il numero di cluster nel file
- LCN (Logical cluster number)
 - numero di cluster logico
- Disk diviso in cluster
- LCN indica su quale cluster del disco è posizionato l'attributo
- Run
 - Indica su quanti cluster il file si espande

S. Balsamo – Università Ca' Foscari Venezia – SO.7.81

81

NTFS

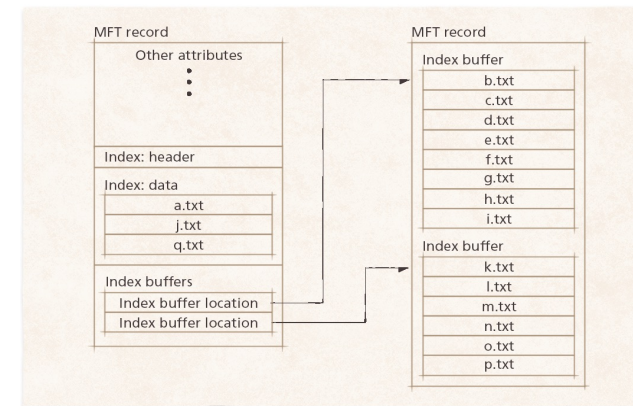
- Directories
 - Files su disk
 - Contiene un elenco alfabetico di file
 - Memorizzato come B-alberi
 - Ogni file in punti di directory per il suo record MFT

S. Balsamo – Università Ca' Foscari Venezia – SO.7.82

82

NTFS

I contenuti delle directory sono memorizzati in B-trees



S. Balsamo – Università Ca' Foscari Venezia – SO.7.83

83

NTFS

- Più righe della directory possono puntare al stesso file
- **Hard links**
 - Puntano al file corretto anche se quello originale è cancellato o spostato
 - NTFS tratta gli hardlinks come un pathname originale
- file_name
 - Attributo del file MFT
 - Uno per ogni pathname di file
- hard_link
 - Attributo del file MFT
 - Numero degli attributi del file_name
 - File cancellato solo quando sono stati cancellati tutti gli hard_link

S. Balsamo – Università Ca' Foscari Venezia – SO.7.84

84

NTFS

- **Data stream**
 - Il contenuto del file è memorizzato in uno o più data stream
 - Attributo di un file
- **Default data stream**
 - flusso di dati senza nome
 - contenuti primari di file
 - Testo del documento di word
 - Immagine di bitmap
- **Alternate data stream**
 - flusso di dati con nome
 - Utilizzato per memorizzare i metadati
 - Autore, sommario, copyright
 - Versioni di Back up

S. Balsamo – Università Ca' Foscari Venezia – SO.7.85

85

NTFS

- **Compressione di File**
 - Trasforma i file per ridurre lo spazio su disco
 - Algoritmo di compressione Lempel-Ziv
 - Trasparente
 - Le applicazioni accedono ai file usando API standard
 - Il sistema comprime e decompime i file
 - Le applicazioni ignorano se il file è compresso

S. Balsamo – Università Ca' Foscari Venezia – SO.7.86

86

NTFS

- **Unità di Compressione**
 - 16 cluster
 - NTFS comprime ogni unità di compressione individualmente
 - Se la versione compressa richiede ancora 16 cluster, l'unità di compressione è memorizzato non compresso
 - Abilita random I/O su file
 - NTFS cache unità di compressione a cui vi è stato accesso recente
 - Probabilità di accedere nuovamente
 - thread di scrittura pigro a bassa priorità comprime in unità cache e scrive su disco

S. Balsamo – Università Ca' Foscari Venezia – SO.7.87

87

NTFS

- Crittografia
 - Protegge i file da accessi illeciti
 - La crittografia eseguita in **unità di compressione**
 - chiavi
 - Crittografia con coppia di **chiavi pubblica/privata**
 - **chiave di recupero** data all'amministratore di sistema
 - Nel caso in cui l'utente dimentica la password
 - versioni crittografate di chiavi memorizzate su disco
 - chiavi di decifrazione memorizzate nel non-paged pool
 - operazioni di crittografia trasparenti all'utente

S. Balsamo – Università Ca' Foscari Venezia – SO.7.88

88

NTFS

- File **sparsi**
 - Esempi
 - Bitmap con grandi aree bianche
 - Matrici sparse
 - Memorizzare grandi serie di bit zero in liste di blocchi zero
 - Consente di risparmiare spazio
 - Minor overhead di compressione
 - Gli utenti possono specificare quali blocchi sono azzerati
 - NTFS rileva automaticamente unità di compressione azzerate
 - Crea elementi nella lista dei blocchi zero
-

S. Balsamo – Università Ca' Foscari Venezia – SO.7.89

89