Sistemi Operativi Windows - Architettura di Sistema.

Il seguente materiale è di proprietà di Stefano Pinardi ed è coperto da copyright ne è consentito l'uso agli studenti per soli motivi di studio, novembre 2021.

Capitolo 1 - Sezione 1: Processi e thread

Capitolo 1 - Sezione 2: L'autenticazione nel quadro architetturale

Capitolo 2 - Utente e Dominio

Capitolo 1 – sezione 2

L’autenticazione nel quadro architetturale

1.39 Il quadro architetturale

Per poter soddisfare esigenze dettate esigenze enterprise i progettisti di Windows hanno sviluppato un sistema operativo che rispondesse alle seguenti caratteristiche:

- scalabile;
- portabile su altre piattaforme hardware;
- sicuro;
- stabile;
- compatibile con le vecchie applicazioni;
- performante.

Il risultato di queste design è un sistema operativo modulare la cui struttura è rappresentata in fig. 1.19 e in fig 1.0 Questa struttura a livelli e moduli consente una maggiore flessibilità dato che ogni singola componente può essere modificata o corretta senza dover intervenire sulle rimanenti. Ogni modulo è in grado di comunicare con le altre attraverso apposite chiamate di funzione, i Questo le rende indipendenti, o meglio modulari. E’ inoltre possibile realizzare versioni con kernel minimali, che occupano poca memoria .

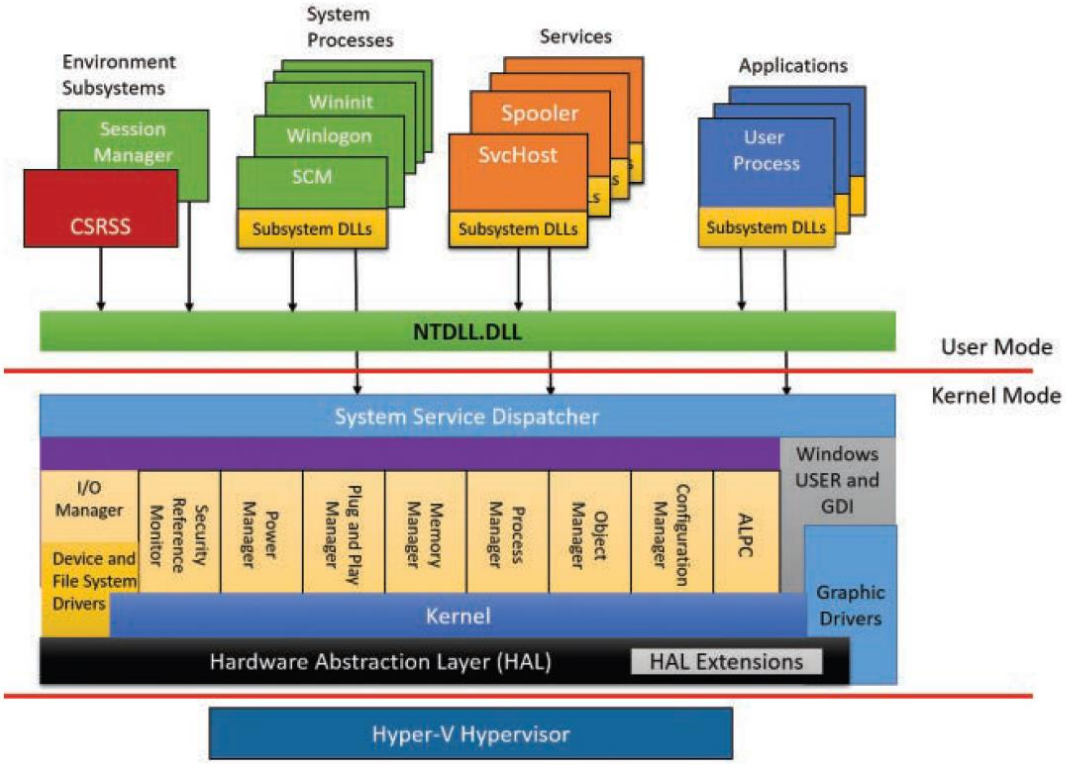

Fig. 1.19 Struttura del sistema operativo Windows (image of Microsoft Press © – do not copy)

1.40 Hardware Abstraction Layer (HAL)

Windows prevedeva sin dalla sua prima edizione (NT) la possibilità di essere un sistema operativo multi piattaforma; per realizzare questo obiettivo si è reso necessario implementare un meccanismo di astrazione rispetto alla macchina. Il modulo *HAL* (cfr. fig. 1.19) è la componente essenziale che realizza l'indipendenza dall'architettura hardware sottostante. Al suo interno sono implementate funzioni che permettono al sistema operativo di utilizzare le funzioni in hardware disponibili (ad esempio riconosce ed utilizza la motherboard). La sua implementazione è contenuta in una speciale libreria dinamica chiamata *HAL.dll.*

1.41 Kernel

Questo livello e quello successivo (Executive) formano in realtà un unico oggetto; questa suddivisione è puramente logica e serve a distinguere un insieme di funzioni semplici e primarie da alcune più complessi. In questo modulo sono state implementate alcune componenti basilari e fondamentali del sistema operativo come ad esempio lo scheduler (cfr. par. 1.19).

1.42 Executive

Questo livello, strettamente legato al precedente, contiene un insieme di componenti che svolgono varie funzioni. Tra loro vi sono il VMM, ovvero *Virtual Memory Manager, l'Object Manager, il Security Reference Monitor, il Process and thread Manager* e *l'I/O Manager*. Un altro compito di questo modulo è quello di rendere disponibili le API (Application programming interface) native del sistema operativo ai livelli sovrastanti, in particolare alle applicazioni usermode. Queste API, per dirla in termini di UNIX, sono le system call di Windows, termine che va preso un po' con le pinze data la profonda differenza tra le due architetture.

Il livello EXECUTIVE e quello KERNEL (precedentemente visto) risiedono entrambi in un unico file chiamato NTOSKRNL.exe ed insieme formano il cosiddetto kernel o nucleo base di servizi di sistema di Windows.

1.43 NTDLL.DLL

Questa libreria rende disponibili alle applicazioni utente le API native di Windows. All'interno di queste funzioni è implementato il meccanismo che permette l'invocazione di uno specifico componente dell'EXECUTIVE.

1.44 Subsystem

Gli architetti di Windows hanno previsto, già in fase di progettazione di NT, la possibilità di "simulare" la presenza del core di più sistemi operativi sulla base di un unico kernel. Quest'obiettivo è stato raggiunto grazie proprio ai subsytem il cui scopo è quello di simulare l'ambiente di esecuzione (le chiamate con le loro firme) come è atteso dagli applicativi usermode.

Come parte della struttura architetturale di Windows esistono almeno due subsystem con questo scopo:

- Win32 subsystem: csrss.exe;
- WSL subsytem: wslss.exe;

In passato sono esistiti (ora obsoletati)

- Interix subsytem;
- Posix subsystem;
- OS2 subsystem;.

I subsystem hanno il compito fondamentale di offrire (esporre programmaticamente) alle applicazioni le API native di Windows mascherandole come se fossero quelle proprie del sistema che *emulano*. In questo modo le applicazioni dell'utente utilizzano ma *indirettamente* le funzioni esportate **dall'unico kernel esistente**. Solamente le funzioni esportate da ogni subsystem sono ben documentate poiché si assume che i programmatori utilizzino queste funzioni **per implementare** i loro programmi senza invocare direttamente l'executive.

Tra i subsystem elencati precedentemente il predominante è il Win32 che viene direttamente attivato all'avvio della macchina, gli altri vengono caricati in memoria *on demand*, ovvero ogni qualvolta un'applicazione lo richieda espressamente. Per motivi di semplicità, ovvero per evitare di replicare in ogni subsystem codice che effettua le medesime funzioni, gli altri subsystem non implementano al loro interno *ogni* aspetto della gestione o accesso alle risorse della macchina, ma possono utilizzare le funzionalità già offerte dal subsystem Win32 che è già attivo.

Un'applicazione che deve funzionare all'interno di un determinato subsystem può utilizzare le librerie fornite da quest'ultimo ed è durante la fase di costruzione dell'applicazione che viene indicato a quale subsystem dovrà fare riferimento Per quanto riguarda il Win32 subsystem tra le sue librerie vi sono la KERNEL32.dll, la USER32.dll, la ADVAPI32.dll e la SHELL32.dll e le funzioni da queste esportate che

prendono il nome ufficalmente inteso di Win32 API. il subsystem OS/2 (IBM) non è più supportato, mentre quello Posix e Interix sono stato sostituito WSL (Subsystem Linux). Quest'ultimo permette di far girare applicativi Unix / Linux in modo più completo e funzionale rispetto al vecchio subsystem Posix e Interix.

1.45 Session Manager (SMSS.exe)

Esiste *un processo* privilegiato che accede *direttamente* alla libreria NTDLL.dll. Questo processo è il **Session Manager**, che è il primo processo che viene eseguito in modalità user mode non appena la macchina viene avviata. I suoi compiti sono di:

- effettuare una serie di operazioni di inizializzazione del sistema operativo;
- creare i vari processi subsystem:

  a) in fase di avvio della macchina (Win32);

  b) on demand ogni qualvolta vengano richiesti (WLS).

Dato che il Session Manager viene eseguito in modalità **user mode,** anche i subsystem girano in questo contesto di esecuzione[1]

> ***Nota:*** *nelle macchine di tipo server (ma non solo) può essere presente un servizio che permette a più utenti, in parallelo, di utilizzare, da una postazione remota (detta "terminale") le sue risorse (hardware e software) in modo interattivo. In questo scenario si parla di "multiutenza parallela" (come avveniva nei primi sistemi Unix e nei mainframe anni '80 e '90). Il servizio che si occupa di gestire questo tipo di accesso della multiutenza è il Terminal Service. Quando siamo in questa modalità il Session Manager crea un'istanza del subsystem Win32, ovvero del processo CSRSS.exe, ed un'istanza del processo Winlogon.exe,* ***per ogni utente*** *"loggato" sul server.*

[1] *i successivi processi saranno creati a partire da questo e gireranno anche essi in modalita user*

1.46 Processi con funzioni speciali

Esistono alcuni processi (fig. 1.0 e fig. 1.19) le cui funzioni sono specifiche:

- winlogon.exe;
- lsass.exe;
- services.exe;
- servizi.

Il *Winlogon.exe* è il processo creato all'avvio della macchina dal Session Manager, il cui scopo è quello di permettere ad un utente di inserire le credenziali di accesso (tipicamente uno username ed una password). E' il noto pannello di autenticazione che si offre anche quando fate accesso al vostro portatile Windows. Questo processo gestisce le operazioni di *logon, logoff, lock e unlock* di una macchina.

Il processo **Lsass.exe**, anche lui creato dal Session Manager, è responsabile dell'applicazione e della gestione del modello di security implementato in Windows. Uno degli elementi di questo modello riguarda autenticazione degli utenti che è obbligatoria e intrinseca al sistema operativo. In particolare, ogni volta che un utente effettua un'operazione di logon, il processo **Winlogon.exe** inoltra al processo **Lsass.exe** le credenziali fornite dall'utente, affinché si proceda a verificarne l'esistenza nel database di autenticazione (SAM o Active Directory) e la correttezza dei dati. Nel caso in cui tale verifica abbia un esito negativo viene semplicemente negata all'utente la possibilità di accedere alla macchina.

In caso positivo, il processo Lsass.exe autorizza l'utente a servirsi del computer e gli rilascia una particolare **"carta d'identità"** che indica tutto ciò che è autorizzato a fare nel sistema con quella identità.

Il processo **Services.exe** è responsabile della gestione di alcuni processi chiamati appunto servizi (il Web server, il Mail server , il DHCP, un DNS). Parliamo di servizi che devono essere attivati direttamente all'avvio della macchina oppure se ne viene richiesta l'esecuzione on demand (cfr. par. 1.39). Corrispondono ai *daemon* presenti negli ambienti UNIX.

Come ultimo elemento esistono i processi utente. Questi colloquiano con i processi precedentemente descritti e sfruttano le risorse messe loro a disposizione dal sistema operativo.

1.47 Servizi

Come appena detto nel par. 1.46 Windows - immediatamente dopo il boot della macchina - avvia automaticamente i servizi.

Questi lavorano in un proprio contesto di sicurezza specificato al momento della loro installazione, che può essere quello di un qualsiasi utente del dominio, o quello dell'utente “SYSTEM”.

> ***Nota**: SYSTEM non è un account che impersona un amministratore, è un account non gestibile dall'amministratore, dotato di privilegi sicurezza elevati.*

I servizi vengono gestiti dal processo denominato Service Control Manager (SERVICES.EXE), e avviato durante il *boot* della macchina; questo processo avvia automaticamente tutti i servizi elencati in un *database*.

La configurazione e la gestione dei servizi possono essere compiute solo per mezzo del **Service Control Manager (SCM)**. Con lo snap-in dei servizi (services.msc) è possibile modificare lo stato di un servizio, ad es avviare un servizio in modo manuale, o fermarlo; un utente con sufficienti privilegi può avviare “a mano” un servizio premendo il pulsante start o fermarlo. Un servizio può essere avviato automaticamente o fermato dal SCM anche su richiesta di un altro servizio (on demand) e per via programmativa.

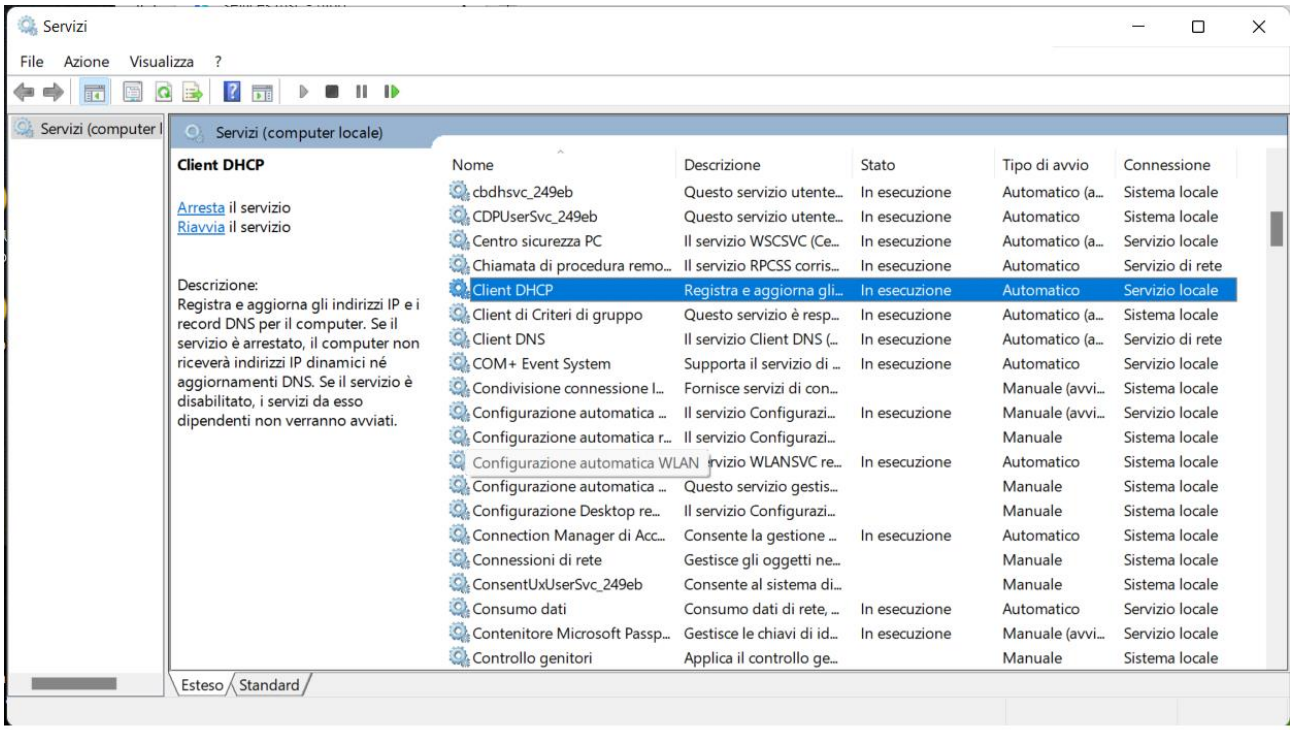

Fig. 1.23 I processi ed il loro contesto di sicurezza mostrati dal tool "services.msc"

A livello programmativo per poter comunicare con loro e per poterli gestire, i servizi hanno bisogno di essere strutturati secondo precise indicazioni. Le parti fondamentali di cui un servizio si compone e che non possono mancare sono tre:

- la funzione main;
- la funzione che contiene il codice del servizio vero e proprio;
- la funzione utilizzata per gestire i messaggi del SCM.

Un servizio, come ogni altro programma ha bisogno di un main() o di un WinMain().