Parte 2°

Struttura interna del sistema LINUX

4. ASPETTI GENERALI DEL SISTEMA OPERATIVO LINUX

La funzione generale svolta da un Sistema Operativo può essere definita come la gestione dell'Hardware orientata a rendere più semplice il suo impiego; la semplificazione dell'impiego dell'Hardware avviene realizzando delle **macchine virtuali**[1], più semplici da utilizzare rispetto all'Hardware, dette **processi**. Gli utilizzatori di tali macchine sono i **programmi applicativi**. Un processo è quindi una macchina virtuale che esegue un programma; per questo motivo in certi casi si fa riferimento in maniera intercambiabile al processo oppure al programma in esecuzione da parte del processo. I due concetti sono però distinti; tra l'altro, è noto che l'esecuzione di un servizio di exec permette di sostituire il programma in esecuziona da parte di un processo con un altro.

Durante la sua esecuzione il programma può richiedere al processo anche particolari funzioni dette **servizi di sistema** (**system services**).

Alcuni programmi applicativi sono a loro volta orientati a permettere all'utilizzatore umano un modo facile di interagire col sistema, primi fra tutti gli **interpreti di comandi** (o **Shell**) e le **interfaccie grafiche** (**GUI**). E' importante tenere presente che sia le Shell che le GUI sono fondamentalmente dei particolari tipi di programmi applicativi, quindi le loro funzionalità dettagliate, pur essendo considerate spesso le funzionalità del SO, non sono altro che un modo particolare di utilizzo delle vere e proprie funzionalità del sistema, che sono definite dai servizi di sistema (figura 4.1).

Nel realizzare i processi il Sistema Operativo deve gestire le peculiarità dei diversi tipi di Hardware esistenti e deve adattarsi alle evoluzioni dell'Hardware che si verificano nel corso del tempo.

In base a queste premesse è logico analizzare le funzioni svolte da un SO analizzando i servizi messi a disposizione dei programmi applicativi da un lato e analizzando i meccanismi adottati dal SO per gestire la varietà dell'Hardware e la sua evoluzione dall'altro.

[1] le macchine create dal Software sono chiamate spesso macchine virtuali

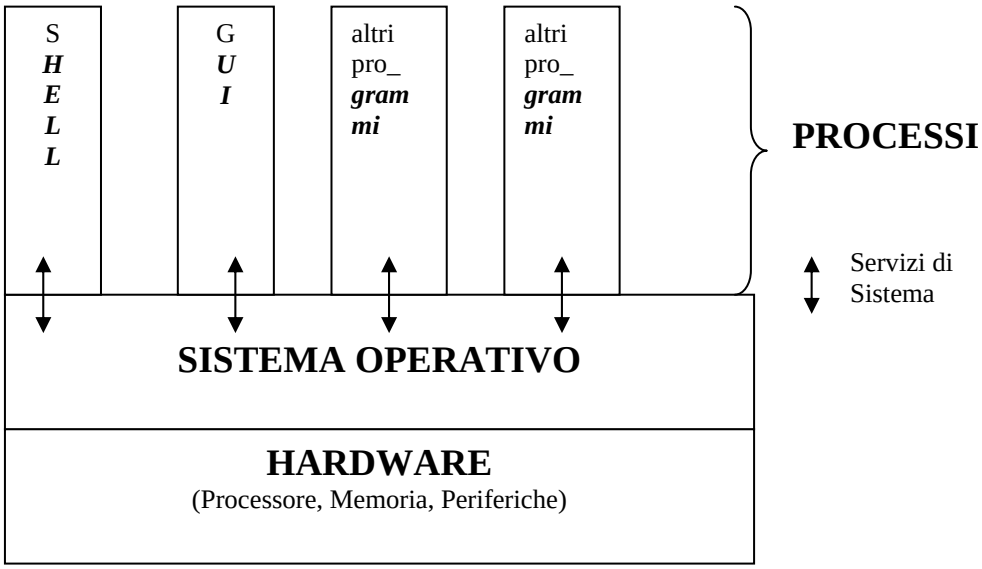

Figura 4.1 – Hardware, Sistema Operativo e Processi

Processi, Scheduler e Priorità

La funzione più fondamentale svolta da un sistema operativo **multiprogrammato** è la **virtualizzazione** dell'esistenza di diversi processi che operano in parallelo; per svolgere questa funzione il sistema operativo deve far eseguire all'unico processore i programmi dei diversi processi in alternanza.

In particolare, LINUX è un sistema **time-sharing**, che fa eseguire un programma per un intervallo di tempo predeterminato, detto **quanto**, quindi ne sospende l'esecuzione (**preemption**). In questo modo il SO tenta di garantire che il processore esegua i programmi in maniera equa, senza essere monopolizzato da un unico programma.

L'esecuzione di un programma può essere sospesa per due motivi:

- perchè il processo è arrivato ad un punto in cui decide autonomamente di porsi in attesa, ad esempio perchè deve aspettare l'arrivo di un dato dall'esterno, oppure

- perchè il processo viene sospeso a causa del completamento del suo quanto di tempo.

Il componente del SO che decide quale processo mandare in esecuzione è detto Scheduler. Il comportamento dello Scheduler è orientato a garantire le seguenti condizioni:

- che i processi più importanti vengano eseguiti prima dei processi meno importanti
- che i processi di pari importanza vengano eseguiti in maniera equa; in particolare ciò significa che nessun processo dovrebbe attendere il prorpio turno di esecuzione per un tempo molto superiore agli altri.

Scheduling e tipi di processi

Non tutti i processi hanno la stessa importanza. Ad un primo livello possiamo distinguere i processi nelle seguenti categorie:

1. Processi **Real-time (in senso stretto).** Questi processi devono soddisfare dei vincoli di tempo estremamente stringenti e devono quindi essere schedulati con estrema rapidità, ***garantendo in ogni caso di non superare un preciso vincolo di ritardo massimo***; un esempio di processo di questo tipo è il controllo di un aereoplano o di un impianto. ***LINUX NON è adatto a supportare processi di questo tipo***, perchè non garantisce tempi certi per la preemption. Esistono varianti di LINUX specializzate per il Real Time che non trattiamo.
2. Processi in **semi-Real-time** (soft Real time). Questi processi, pur richiedendo una relativa rapidità di risposta, non richiedono la garanzia assoluta di non superare un certo ritardo massimo. Un esempio di questo tipo è la scrittura di un CD o la esecuzione di un file Audio, che deve essere svolta producendo i dati con una certa continuità; se talvolta questa continuità non viene garantita si perde un CD, evento accettabile (a differenza di una caduta di aereo o di un'esplosione di impianto). Tuttavia, il processo che scrive un CD deve avere un certo livello di precedenza che gli permetta in generale di produrre i dati in tempo quando servono. ***In***

LINUX è possibile identificare questi processi e assegnargli un'alta priorità. Questa priorità è detta statica, vale da 1 a 99 e non cambia mai

3. Processi **Normali**. Sono tutti gli altri processi, ai quali si vuole garantire una equa distribuzione del tempo di CPU. Questi processi possono avere diversi comportamenti:
    - **processi I/O bound** sono i processi che si sospendono spesso perchè hanno bisogno di I/O (ad esempio, un Text editor)
    - **processi CPU bound** sono invece i processi che tendono ad usare molto la CPU, perchè non si autosospendono (ad esempio, un Compilatore)

In assenza di meccanismi correttivi i processi I/O bound tenderebbero a funzionare molto male in un sistema che si limitasse a sospendere i processi in base alle due regole citata al paragrafo precedente (autosospensione e completamento di un quanto di tempo).

Ad esempio, si considerino un Editor di testo e un Compilatore. Ambedue sono processi normali. Tuttavia, dato che l'Editor è molto interattivo (I/O bound) tenderebbe ad eseguire per poco tempo e poi a sospendersi, mentre il compilatore (CPU bound) ogni volta che va in esecuzione tenderebbe ad eseguire per l'intero quanto di tempo.

LINUX non riconosce a priori i processi I/O bound da quelli CPU bound, ma utilizza un meccanismo di **priorità dinamica** che favorisce implicitamente i processi I/O bound, perchè riduce la priorità dei processi in base al tempo di esecuzione che hanno già utilizzato e valuta la possibilità di sospendere un processo non solo quando è scaduto il suo quanto di tempo, ma anche quando un processo di priorità più elevata diventa pronto per eseguire. I dettagli di questo meccanismo sono spiegati più avanti, perchè richiedono conoscenze più approfondite.

Servizi ai programmi applicativi

Ovviamente, la virtualizzazione dei processi da parte del sistema operativo implica la realizzazione dei servizi fondamentali di creazione dei processi e di esecuzione dei programmi, in particolare i servizi fork, exec, wait e exit. Altri servizi collegati a questi riguardano la comunicazione tra diversi processi, ma questi servizi non vengono trattati in questo testo.

Oltre alla realizzazione dei processi e dei relativi servizi, il Sistema Operativo fornisce altri tipi di servizi, in particolare i **servizi di accesso ai file** (open, close, read, write, lseek, ecc…) tramite i quali i programmi applicativi possono non solo accedere ai file normali, ma, grazie alla nozione di **file speciale**, possono accedere anche alle periferiche. Ad esempio, la stampa su una stampante o l'interazione con il terminale sono visti come operazioni su file speciali associati alla stampante e al terminale.

Il sistema operativo come gestore di risorse

Le caratteristiche del sistema operativo possono anche essere osservate da un punto di vista diverso rispetto a quello delle funzionalità messe a disposizione dei programmi applicativi, cioè dal punto di vista del sistema operativo come gestore di risorse. Da questo punto di vista si mette in evidenza il fatto che il sistema operativo deve gestire le risorse fisiche disponibili (Hardware) in maniera efficiente nel fornire le funzionalità descritte sopra.

Le risorse che il sistema operativo deve gestire sono fondamentalmente le seguenti:

- il **processore** (o **CPU**), che deve essere assegnato all'esecuzione dei diversi processi e del sistema operativo stesso
- la **memoria**, che deve contenere i programmi (codice e dati) eseguiti nei diversi processi e il sistema operativo stesso
- le **periferiche**, che devono essere gestite in funzione delle richieste dei diversi processi

Un aspetto di fondamentale importanza nel comprendere le caratteristiche di un sistema operativo è l'esigenza del sistema operativo di adattarsi nel tempo all'evoluzione delle tecnologie Hardware, in particolare delle periferiche. La vita di un sistema operativo infatti è lunga (se il sistema è ben progettato); in particolare, il sistema UNIX, di cui LINUX rappresenta l'ultima evoluzione, risale al 1970. Le periferiche vengono invece riprogettate praticamente in continuazione da una moltitudine di produttori diversi, quindi è fondamentale che il sistema operativo possa essere rispecializzato continuamente per la gestione di nuove periferiche.

Questo obiettivo è conseguito da LINUX tramite due accorgimenti:

1. esiste un modo ben definito per inserire nel sistema il software di gestione di una nuova periferica, detto **gestore di periferica** (**device driver**);
2. l'accesso alle periferiche avviene assimilandole a dei file, in modo che un programma non debba conoscere i dettagli realizzativi della periferica stessa; ad esempio, dato che un programma scrive su una stampante richiedendo dei servizi di scrittura su un file speciale associato alla stampante, se la stampante viene sostituita con un diverso modello il programma non ne risente (ovviamente, è il diverso gestore (driver) della stampante che si occupa di gestire le caratteristiche della nuova stampante).

Il seguito di questo testo è strutturato, coerentemente con le caratteristiche del sistema oeprativo, nel modo seguente: nel prossimo capitolo analizzeremo le caratteristiche principali delle macchine che il sistema operativo deve gestire; nel capitolo 8 vedremo gli aspetti più fondamentali del funzionamento di LINUX, legati alla realizzazione dei processi e all'interazione tra le diverse componenti fisiche delle macchine. La parte del sistema operativo che svolge queste funzioni fondamentali è chiamata **nucleo** del sistema operativo (**kernel**). Nel capitolo successivo analizzeremo la **gestione della memoria**, che costituisce una risorsa fondamentale del sistema. Infine, negli ultimi capitoli vedremo i principi di funzionamento del **file system**, che gestisce tutti i file e fornisce tutti i relativi servizi di sistema ai programmi applicativi, e i principi di realizzazione dei **gestori di periferica**.