

GNU/Linux kernel space and user space

Autore: Stefano Pardini

<http://www.viareggiolinux.org>

Linux User Group: ACROS <http://www.lug-acros.org>

Come molti sapranno GNU/Linux nasce come sistema multitasking e multiutente, come tutti i sistemi Unix, questa caratteristica ne permette l'uso contemporaneo da parte di più utenti. Il sistema GNU/Linux ha una architettura tale per cui vi è una netta distinzione fra il *Kernel* che è il nucleo del sistema che gestisce le risorse hardware (quali CPU, memoria e periferiche) ed i *processi*, che in questo caso vanno dai comandi base di sistema alle applicazioni ed alle interfacce grafiche che gli utenti usano per interagire con la macchina. Per capire bene questa netta distinzione si fa riferimento ai concetti di *kernel space* e *user space*, rispettivamente all'ambiente in cui viene eseguito il kernel e all'ambiente in cui vengono eseguiti i processi. Di fatto solo il kernel accede direttamente alle risorse hardware del calcolatore mentre i processi vengono eseguiti in un ambiente virtuale, appunto l'*user space*. In questo ambiente virtuale i processi riescono ad accedere alle periferiche attraverso le cosiddette *system call* (chiamate di sistema), che vengono utilizzate attraverso l'interfaccia fornita dalla libreria di sistema (*GNU C library*).

Per capire meglio questo concetto si può dire che in pratica l'unico programma che viene eseguito veramente è il kernel stesso, che si occupa di costruire questo ambiente virtuale in cui girano gli altri programmi. Possiamo dire che una parte del kernel detta *scheduler*, si occuperà della realizzazione del multitasking con una corretta gestione del tempo di processore, una seconda parte detta VM (virtual memory), si occuperà di gestire l'uso della memoria disponibile facendo in modo che ogni processo sia indipendente e non possa accedere alla memoria di un altro processo e garantendo in caso di necessità lo *swap* (cioè lo spostamento delle pagine di memoria meno usate su un apposito spazio disco), mentre un'ultima parte detta *driver* farà in modo che i programmi possano accedere alle varie periferiche. Quest'ultima parte molto importante definisce il concetto per cui in un sistema quale GNU/Linux si ha a disposizione una interfaccia di accesso generica per qualsiasi dispositivo.

Questo è il motivo per cui in un sistema unix-like si usa dire che *everything is a file* (tutto è un file).

In questo contesto per capire bene la distinzione netta fra *user space* e *kernel space* si può prendere ad esempio l'avvio del sistema. Quando si accende il computer viene eseguito il programma BIOS che molti non conoscono e che si occupa di effettuare dei controlli interni per poi successivamente avviare il sistema. Il BIOS carica un piccolo programma detto *bootloader* che a sua volta carica un'immagine del kernel che solitamente risiede nel disco e viene così eseguita. Quando il kernel viene per così dire caricato, esso effettuerà dei controlli fra i quali uno molto importante che è quello di leggere la tabella delle partizioni del disco rigido e solo successivamente si occuperà di montare il file system dove risiede la directory radice. Infine farà partire il primo processo che per convenzione si chiama *init* che è il programma che a sua volta farà partire tutti gli altri processi che consentono di usare il sistema. Fra questi processi tanto per citarne uno abbiamo *login*, che si occupa di chiedere nome e password all'utente che si vuole collegare. Il processo *login* come tutti gli altri processi funziona attraverso una opportuna *system call* messa a disposizione appunto dal kernel, come abbiamo detto in precedenza.

Un sistema GNU/Linux per chi ne conosce un pò la storia si può considerare l'insieme del kernel più i programmi ed è per questo che si chiama appunto GNU/Linux e non solo Linux come troppo spesso ed erroneamente viene citato.

Per concludere possiamo dire che gran parte della stabilità di un sistema GNU/Linux deriva da queste eccezionali caratteristiche di solidità della architettura di base che a differenza di altri sistemi operativi garantisce prestazioni molto performanti ed una corretta gestione delle risorse hardware di un moderno calcolatore.

Ho scritto questo breve documento per cercare di dare una esaustiva spiegazione del concetto di architettura base di un sistema GNU/Linux, credo che ogni utilizzatore di questo potente e stabile sistema operativo dovrebbe avere queste nozioni fondamentali per capire alcune delle peculiarità di un sistema di questo tipo. Troppo spesso accade infatti di sentir dire semplicemente e in maniera riduttiva che GNU/Linux è stabile, veloce, flessibile eccetera ma altrettanto spesso non si è poi in grado di fornire una spiegazione a queste corrette affermazioni. Conoscere la base dell'architettura di GNU/Linux aiuta sicuramente a capire tante cose che diversamente resterebbero un pianeta sconosciuto.

Enjoy with Gnu/Linux...



Questo/a opera è pubblicata sotto una [Licenza Creative Commons](https://creativecommons.org/licenses/by-nc-sa/4.0/).