GNU/Linux Intermedio

–*Seconda lezione:*

–***FONDAMENTI DEI***

–***SISTEMI OPERATIVI***

Duro e soffice

- Ogni calcolatore è fatto da componenti il cui unico scopo è quello di eseguire istruzioni
  - Senza istruzioni il calcolatore è cosa morta
  - Il calcolatore ha una memoria in cui sono memorizzate le istruzioni di avvio (programma detto “bootstrap”)
  - Il processore ha un “Program Counter” che ,all'avvio , contiene l'indirizzo della memoria con il “bootstrap”
  - All'avvio il calcolatore esegue le istruzioni contenute nella memoria di “bootstrap” prendendo da dischi, nastri, rete, ecc. , il sistema operativo
- Compito del sistema operativo è eseguire gli ordini dei programmi scritti per quel calcolatore

Cos'è un Sistema Operativo?

- Al contrario di quello che i produttori dei sistemi operativi commerciali cercano di far credere, il sistema operativo NON E' l'insieme degli applicativi che compongono il sistema installato.
- L'editor di testo non fa parte, ad esempio, del sistema operativo, l'interfaccia grafica non fa parte del sistema operativo, la console non fa parte del sistema operativo.
- Il sistema operativo è quell'entità software (programma) che si occupa del diretto controllo e della gestione delle risorse hardware di un computer, e delle operazioni di base in qualche modo connesse.
- A seconda dei sistemi operativi, anche l'autenticazione degli utenti e dei processi può essere gestita dal sistema operativo.

Il kernel

- Tutte queste funzionalità, sono generalmente svolte da un unico applicativo, detto “kernel” (nucleo).
- Linux è un kernel
- Spesso nel concetto di Sistema Operativo, vengono inclusi anche una serie di programmi che ne consentono la gestione.
- Il concetto di Sistema Operativo è poi stato commercialmente ampliato dai produttori stessi, con il risultato di generare tanta ed inutile confusione.
- In sintesi, potremo dire che “Linux” è il Sistema Operativo secondo il concetto originale, mentre GNU/Linux è il “Sistema Operativo” in questa seconda, ampliata, accezione.

I processi

- Tutto il sistema operativo serve ad eseguire i processi
  - Quando li scriviamo, li memorizziamo, li compiliamo, sono programmi
  - Quando li mandiamo in esecuzione sono processi
- Mandare in esecuzione un programma significa
  - Copiarlo da disco in memoria centrale (RAM)
  - Predisporre delle tabelle nello scheduler (vedi più avanti)
  - Mettere l'indirizzo della prima istruzione nel “program counter” del processore in modo che il processore inizi ad eseguire le istruzioni previste dal programma
    - Questo viene fatto dallo scheduler

Il gestore di filesystem

- D'ora in poi, salvo dove diversamente riportato, parleremo di Sistema Operativo nella sua accezione originale, indicando cioè il solo kernel ed i programmi che ne consentono la gestione (ad esempio le utility per caricare e rimuovere moduli per quanto riguarda Linux).
- Un OS moderno, si compone di una serie di componenti piuttosto ben definiti:
  - Un gestore di filesystem
    - E' la parte di software che si occupa fisicamente di leggere e scrivere dati sul filesystem residente su una periferica di massa, a fronte di richieste.
    - Quando un processo (una singola istanza di un programma) vuole leggere o scrivere dati su

Gestore della memoria

- Un OS moderno, si compone di una serie di componenti piuttosto ben definiti:
  - Un gestore della memoria fisica
    - Che si occupa di gestire la RAM, caricare in memoria i processi, allocare le necessarie “pagine di memoria”
    - La memoria fisica (RAM) viene infatti divisa a livello “logico” in una serie di “pagine” di dimensione fissa (4K per i processori Intel)
    - Quando si vuole eseguire un programma, questo deve essere caricato in memoria. E' il gestore della memoria che “riserva” (alloca) una numero di pagine adatto a contenere l'eseguibile ed i dati necessari alla sua esecuzione nella memoria RAM.
    - Nel caso in cui fosse necessaria, in seguito,

Lo scheduler

- Un OS moderno, si compone di una serie di componenti piuttosto ben definiti:
  - Uno scheduler
    - E' quella parte del kernel che si occupa di decidere quale processo deve essere messo in esecuzione, ed eventualmente di metterne in esecuzione un'altro a fronte di una serie di valori ben definiti (vedremo).
    - Non dobbiamo infatti dimenticarci che il processore di un computer può eseguire una ed una sola istruzione alla volta, e questa istruzione appartiene, ovviamente, ad un unico programma.
    - Per poter dare l'impressione all'utente di poter eseguire piu processi nello stesso momento (sistema operativo “multitask”), è necessario

Gli stati

- La possibilità di eseguire più processi contemporaneamente è dovuta al fatto che ogni processo non impiega la CPU per tutto il tempo, dovendo spesso attendere altre componenti
  - Dati in input dall'utente
  - Operazioni su periferiche lente (dischi, rete)
- Quindi non sempre un processo è attivo, anzi
  - Un processo si disattiva e diventa “waiting” quando attende il completamento di un operazione lenta
  - Quando l'operazione è completata il processo ridiventa “ready” ovvero pronto per l'esecuzione
  - Solo un processo per volta è “running” ovvero in esecuzione

Tipologie di OS: Batch

- Come si decide quale dei tanti processi in esecuzione va effettivamente caricato in memoria ed eseguito?
- In risposta a questa domanda esistono una serie di tipologie differenti di sistemi operativi, che scelgono il processo (ed il momento in cui questo deve terminare la sua esecuzione) in maniera differente:
  - Batch
    - Ogni processo viene caricato in memoria, esegue, e termina. A quel punto il processo seguente nella “coda di batch” viene lanciato in esecuzione e cosi via. Non c'è esecuzione parallela (multitask)

Time-sharing collaborativo

- Quando un determinato processo arresta la propria esecuzione, l'esecuzione passa ad un altro processo.
- La scelta di quale processo mettere in esecuzione, è data da un'insieme di fattori:
  - i processi possono essere “ready” (pronti ad eseguire) o “waiting” (in attesa di qualche evento).
  - Se ci sono piu processi “ready” contemporaneamente, l'esecuzione viene passata in base ad una “priorità”. Il processo con “priorità piu altà” viene posto in esecuzione.
  - Se c'è un solo processo “ready”, viene eseguito quello.
  - Se non ce ne sono, viene eseguito il processo “idle” (che è sempre “ready” ma ha priorità piu bassa di tutti gli altri programmi).

Time-sharing preemptive

- Time-sharing preemptive
  - E' molto simile al “time-sharing collaborativo”, con la differenza che dopo un determinato tempo, anche se un processo non ha terminato il suo compito, viene sospeso.
  - Questo viene fatto perchè nel malaugurato caso in cui un processo non arresti mai autonomamente la sua esecuzione (ad esempio perchè è entrato in un “ciclo infinito”), tutto il sistema rimane bloccato in attesa che termini.
  - La scelta del nuovo processo da mettere in esecuzione, è uguale a quella del “time-sharing collaborativo”, ovvero basato su stato e priorità.
  - Linux è un sistema operativo che rientra in questa categoria

Real-time

- Real-Time
  - In alcuni ambiti particolari, è necessario che il tempo massimo che intercorre tra quando un evento si verifica e quando il processo che lo deve gestire lo prende in carico sia prestabilito.
  - Quanto questo tempo sia breve, non è importante. Può essere anche “2 milioni di anni”, ma deve essere garantito che entro quel tempo, l'evento venga gestito.
  - I sistemi operativi Real-Time sono proprio quei sistemi operativi che presentano questa caratteristica.
  - Un paio di esempi di usi di software real-time:
    - Telefonia, gestione audio/video
    - Automazione industriale

Context switching

- Nel momento in cui un processo viene sospeso, oltre al processo in se, sono presenti in RAM anche una gran quantità di dati (ad esempio le stringhe da visualizzare, o i dati vengono in quel momento elaborati) che sono necessari al funzionamento del processo stesso (gli indirizzi di memoria da usare, ad esempio)
- Ogni volta che si passa ad eseguire un nuovo processo quindi, non basta ricaricarlo in memoria. Bisogna anche ripristinare le identiche condizioni (nei registri del processore, la dove avviene la vera e propria esecuzione) che sussistevano quando il processo è stato sospeso.
- L'operazione di salvare questi dati, e ripristinarli quando il processo viene rimesso in esecuzione si chiama "context switch", "cambio di contesto"

Interrupt

- Per poter eseguire delle istruzioni allo scatenarsi di una condizione si utilizza il meccanismo degli interrupt
  - La periferica che richiede attenzione attiva un filo con cui chiede attenzione alla CPU
  - La CPU esegue un context switch limitato ed esegue il codice necessario a rispondere alla periferica
  - Sovente il risultato è il cambiamento dello stato di un processo che ha a che fare con la periferica e quindi può portare alla rischedulazione dei processi
- Un interrupt particolare è quello del timer che segnala la fine di un “tick temporale” e causa una rischedulazione dei processi nei sistemi operativi multitasking di tipo preemptive.

Interrupt

- L'interrupt del timer ha priorità più bassa degli altri
  - In modo che un azione “urgente” non venga spezzata da un cambio di processo
- Gli interrupt creano condizioni di “corsa critica” perché possono capitare in momenti inopportuni
  - Tutti i SO possono disabilitare gli interrupt quando necessario
    - Per tempi molto brevi (poche istruzioni)
    - In condizioni di “corsa critica”, ad esempio quando stanno eseguendo il context switch

Thread

- In alcuni casi, può risultare utile poter “suddividere” un singolo processo in una serie di azioni raggruppate in diversi “thread”
- Questo permette di sfruttare calcolatori multiprocessore dividendo il calcolo su più CPU
  - Senzai thread un processi finirebbe inevitabilmente in esecuzione su una singola CPU
- Questo consente anche di ottenere un “multi-tasking” su sistemi che non lo supportano
  - La Java Virtual Machine quando opera su un SO non multitasking
- Thread dello stesso processo condividono la memoria
  - switch veloce ma meno o nessun isolamento tra thread

Multi-tasking

- Uno dei vari problemi che il multi-tasking introduce, è quello della protezione dell'area di memoria associata ad un programma nei confronti degli altri programmi in esecuzione.
- Se infatti ogni processo potesse andare a “taroccare” la memoria che appartiene ad un altro processo, potrebbero insorgere una serie di problemi.
- Si introduce una parte integrante del kernel che si occupa di assegnare ogni singola pagina di memoria ad un singolo processo, ed a consentire solo a questo processo di accedervi: questo è compito dello scheduler.
- I processori moderni fanno questo “in hardware” usando un componente noto come Memory Management Unit.

Multi-utenza

- Se un Sistema Operativo è in grado di gestire piu processi contemporaneamente (multi-tasking), allora è facile che possa anche assegnare questi processi ad utenti diversi.
- Un sistema in grado di supportare utenti diversi contemporaneamente viene detto “multi-utente”
- Ogni utente ha un identificativo univoco, e ogni volta che lancia un nuovo processo, a questo viene associato questo identificativo.
- In questo modo piu utenti possono “loggarsi” contemporaneamente sul sistema e utilizzarlo.
- Si pone però il problema di distinguere i diversi utenti, limitare i comandi che questi possono impartire in modo che non possano arrecare danni agli altri utenti (o anche solo accedere i contenuti “privati” degli altri

Permessi sui file

- Per fare questo, nasce il concetto di “permessi” sui files (che vedremo piu approfonditamente nelle prossime lezioni)
- Viene inoltre ideato il meccanismo della diversificazione dei permessi degli utenti, con l'introduzione di un utente amministratore (“root” sui sistemi Linux) che può eseguire una serie di comandi che invece sono negati agli altri utenti “non privilegiati”.
- Ad esempio la configurazione del sistema, o lo spegnimento dello stesso, possono essere negati a tutti gli utenti salvo l'utente amministratore (o utenti a cui l'amministratore stesso abbia deciso di assegnare questi privilegi)
- Vedremo i concetti legati agli utenti piu avanti.

Tipi di processori

- CISC: Complex instruction set computer
  - Sono i “normali” processori che, generazione dopo generazione, crescono di prestazioni e complessità
- RISC: Reduced instruction set computer
  - Sono processori “semplici ma veloci” progettati quando ci si è accorti che i compliatori ed i sistemi operativi non riuscivano a stare dietro allo sciluppo dei CISC, le cui nuove istruzioni non riuscivano ad essere utilizzate
  - Hanno meno istruzioni, hanno istruzioni più semplici, ma sono comunque complessi: hanno più registri interni, una pipeline che permette di reperire in memoria l'istruzione<seguente mentre stanno ancora eseguendo quella attuale ...

Tipi di processori

- Anche se spesso tendiamo a dimenticarcelo, non esistono solo le piattaforme “PC” classiche (basate su processori della famiglia Intel o compatibili, come i processori AMD).
- Il 75% dei processori per applicazioni “embedded” esistenti al mondo, è di tipo ARM. Cellulari, palmari, calcolatrici avanzate e cosi via, infatti, pur non necessitando risorse particolarmente abbondanti, possono facilmente essere utilizzati con processori di questa famiglia di RISC a 32 bit, perchè caratterizzate da consumi estremamente ridotti in rapporto alle prestazioni.

PowerPC

- Anche nel mondo dei computer come li intendiamo comunemente, esistono piattaforme diverse:
  - PPC (PowerPC)
    - Si tratta della principale piattaforma RISC per sistemi computer, e per anni ha avuto in Apple la propria bandiera.
    - Nata nel 1991 da un'alleanza tra Apple, IBM e Motorola
    - Nonostante la recente migrazione di Apple verso i processori Intel, continuano ad esistere.
    - Lo scorso anno, IBM ha infatti deciso di non proseguire con un ulteriore sviluppo della piattaforma, volendosi concentrare sui nuovi processori CELL, espressamente pensati per le consoles .
    - Lo sviluppo è ora in mano a Power.org (di cui la

SPARC

- Anche nel mondo dei computer come li intendiamo comunemente, esistono piattaforme diverse:
  - SPARC
    - Piattaforma RISC sviluppata da SUN Microsystems
    - Si tratta di un'architettura aperta e non proprietaria, al punto che ne esiste un'implementazione “open source” completa, che si chiama LEON.
    - Il maggior sistema operativo che gira su questa piattaforma è Solaris, di SUN, recentemente rilasciato come progetto opensource con il nome di OpenSolaris.
    - Come per la piattaforma PowerPC, Linux e i 3 principali sistemi operativi liberi BSD (OpenBSD, FreeBSD e NetBSD) offrono supporto

Intel e AMD

- Anche nel mondo dei computer come li intendiamo comunemente, esistono piattaforme diverse:
  - INTEL
    - La principale piattaforma CISC, prende il nome dal produttore storico di processori di questa famiglia, la Intel, la quale ha iniziato la produzione di processori di questa architettura con la serie 80*, in seguito con la famiglia dei Pentium e Celeron (versione economica), ed oggi con la nuova famiglia dei processori Core (prima Core Solo e Core Duo, poi Core 2 Duo)
  - Principale concorrente di Intel nella produzione di processori in questa piattaforma è la AMD, con le famiglie K*, Athlon, Athlon64, Turion, Sempron (economico), Opteron (dual core), Athlon64X2 (dual core, il cui rilascio è stato recentemente rimandato

Architetture e piattaforme

- Anche nel mondo dei computer come li intendiamo comunemente, esistono piattaforme diverse:
  - INTEL
    - Con l'introduzione delle tecnologie a 64 bit, dei 2 e 4 core (previsti per il prossimo anno), ed il passaggio da parte anche di Apple su questa piattaforma, si può dire che oggi la lotta nel campo dei personal computer consumer avviene quasi esclusivamente su questa architettura.
    - Principali sistemi operativi disponibili per architettura Intel sono senza dubbio la famiglia dei sistemi operativi Microsoft Windows, il sistema operativo opensource Linux, il nuovo MacOS X di Apple ed i sistemi operativi liberi BSD che abbiamo già citato in