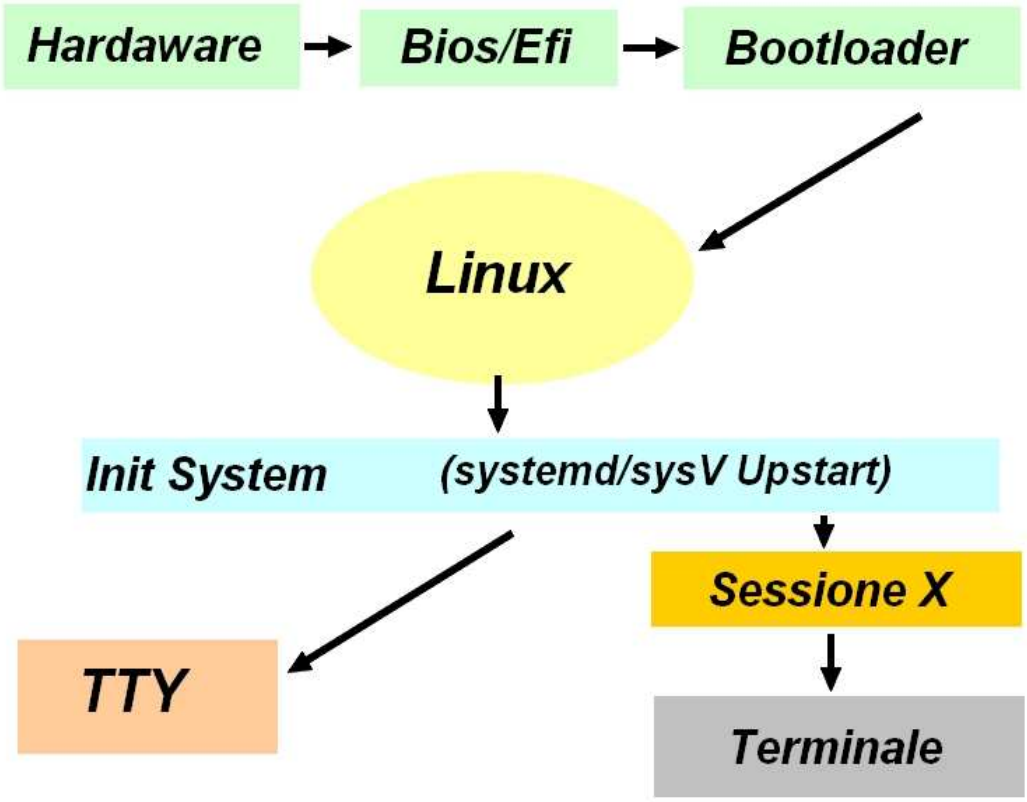
Cosa fa il sistemista? Lezione 1

→ Installazione/configurazione nuovi computer
→ gestione/manutenzione di molti computer
→ gestisce gli utenti di quei PC
→ progettazione sistemi/ infrastruttura di rete

Boot del sistema



→ **Inizializzazione hardware** che poi passa il controllo al…
→ **Bios** oppure **Efi** da li in poi prende il controllo il…
→ **Bootloader** (GRUB o LILO) che carica il sistema operativo…
→ **LINUX**

Tramite alcuni programmi che si chiamano **Init System** si avviano vari componenti che arrivano ad attivare due scelte: **TTY** che sono dei terminali “grezzi” oppure sessioni grafiche (**sessioni X**) e da li si può avviare il classico **terminale**.

Shell

→ ksh – BSD/UNIX
→ ash – si trova con busybox
→ bash – la shell di default nella maggior parte delle distro
→ zsh – shell compatibile con bash, che offre più feature
→ fish – user friendly shell.

Scripting

In questa “shell” possiamo scrivere script cioè file testuali eseguibili.

→ shell scripting (bash, zsh)
→ perl
→ python
→ ruby

Alcuni comandi (navigare nel file system)

CD (change directory) permette di spostarsi fra le varie cartelle del sistema

Con **CD ..** possiamo tornare in una cartella precedentemente aperta cartella superiore; la cartella punto **.** è la cartella corrente **..** è quella superiore.

Ci si sono due modi per indicare la cartella in cui vogliamo muoverci:
In modo assoluto partendo da una cartella radice **CD /**
E in modo relativo partendo da una cartella fino ad arrivare a quella finale.

Esempio 1

/corso >

/corso > cd .. per tornare alla cartella corso si scrive:

> cd ./corso e premendo invio

/corso > si ritorna nella cartella corso.

All’interno della cartella si possono creare altre cartelle con il comando

Mkdir **> mkdir corso2**

Per verificare (o individuare) se la cartella è stata creata usiamo il comando

ls (list) il quale mostra nella cartella corrente

/corso > ls

risultato: **corso2/** (lo slash si trova dopo il nome).

Per rimuovere la cartella (se non ci serve più) possiamo usare il comando:

rmdir piu il nome della cartella da cancellare (serve a rimuovere le directory vuote, se ci sono file all'interno non permette la cancellazione).

Si possono creare anche dei file all'interno delle cartelle con il comando

touch si crea un file vuoto

esempio: **touch** pippo

il comando **/corso > touch corso2/pippo2** crea il file pippo2 nella cartella corso2.

Si possono spostare i file da un una cartella all'altra (ma anche rinominare) Con il comando

mv (move)

/corso > mv corso2/ pippo2 . (cioè muovi corso2 il file pippo2 nella cartella corrente (ovvero corso).

Sempre con **mv** possiamo rinominare un file ad esmpio:

/corso > mv pippo2 pippo3

Per rimuovere un file si usa invece il comando **rm**

per rimuovere invece una cartella (non vuota) "con la forza" si usa il comando associato a dei parametri:

/corso > rm –r –f

-r indica di rimuovere in modo ricorsivo
-f indica di rimuovere la directory

(in modo ricorsivo significa che parto dalla cartella padre e rimuove tutte quelle figlio).

Il comando **cat** permette di visualizzare il contenuto di un file.

Creare uno script in bash shell

Ogni script inizia con i simboli **#!** Esempio:

#! /bin/bash

echo hello world
echo ciao mondo

per far capire alla shell che si tratta di uno script bisogna render eseguibile il file con il comando:

chmod +x esempio: **chmod +x ilmioscript**

per gli script è obbligatorio mettere il **./** per eseguire il file

./ilmioscript

Le variabili in bash iniziano con il segno del dollaro **$**

Variabili speciali

$1 è una variabile speciale (in sostanza il primo argomento passato allo script).

$@ è una variabile che contiene tutti gli argomenti separati da spazi.

I commenti in bash iniziano dopo il segno cancelletto **#**

questo è un commento….

L'istruzione **if** in bash per poter funzionare si deve scrivere correttamente rispettando gli spazi esempio:

if [[espressione]]; then

cat pippo2 #alla fine dello script si mette il comando **fi**

if verifica se l'istruzione è vera o falsa ed esegue il comando indicato all'interno del blocco.

if [[-e pippo2]]; then

cat pippo2

else

cat pippo

fi

l'istruzione **–e** all'interno dell'espressione verifica se ci sia il file dato come parametro (in questo caso pippo2).
Lo script infatti indica di visualizzare il file se il file pippo2 esiste;
altrimenti (con else) di visualizzare il file contenuto nel blocco di else cioè pippo.

Il comando **while** ha una sintassi simile al comando **if** esempio:

while [[espressione]]; do

done #per chiudere si usa la parola done

(fino a quando l'espressione si mantiene vera le istruzioni vengono eseguite)

In Linux esiste un comando **file** che identifica il tipo di file che stiamo eseguendo ad esempio:

file ilmioscript (premendo invio abbiamo il risultato)

Bourne- Again shell script, ASCII text executable

(identifica il tipo di file dalla prima riga dello script)

find è un altro comando utile che visualizza le cartelle e i file partendo da una cartella esempio:

find mario

il comando **grep** permette di eseguire espressioni regolari (che sono delle regole per far combaciare il testo scritto per verificare se coincide con quello che cerchiamo).

La più semplice espressione regolare è il testo stesso ad esempio:

/corso > cat luigi | grep blablabla ← grep cercherà il testo con blablabla

(Il simbolo **|** è una pipeline che passa tutto a grep)

Si può usare anche **/corso > cat luigi | grep -v blablabla** per invertire la ricerca; in pratica visualizza il resto del contenuto del file escludendo le lettere indicate (blablabla).

Il comando **screen** permette di avere più sessioni di shell aperte all'interno di una. Con **ctrl a w** (control + lettera a w) ci indica che abbiamo una sola finestra con **ctrl a c** si creare un'altra finestra; per cambiare da una finestra all'altra basta digitare **ctrl a (**numero della finestra) **0 , 1 , 2 , ecc…**

Il simbolo asterisco ***** indica la cartella corrente della finestra.

Per uscire dalla shell si usa il comando **exit**

Fine della prima lezione
28/11/2015

Cefalì Antonino