LABORATORIO DI SISTEMI OPERATIVI

Cenni storici su UNIX e la shell; filesystem e primi comandi

fabio.rossi@unipg.it

075 5855005

Unix

- Unix è una *"famiglia"* di Sistemi operativi *multitasking* e *multiuser* che deriva dall'originale *AT&T Unix*
- Lo sviluppo di *AT&T Unix* iniziò nel '69 al *Bell Labs research center* ad opera di *Ken Thompson*, *Dennis Ritchie* ed altri

- Nel tempo si è cercato di standardizzare le varie release/varianti: POSIX, SUS, The OpenGroup, etc. (cfr anche qui)

Unix : concetti base

- *multitasking* e *multiuser*
- File system gerarchico (unica *root*)
- Formato di memorizzazione dei file semplice (*plain text*)
- I *device* e diversi strumenti per l'*IPC* sono "trattati/mostrati" come file
- concepito per essere adatto a supportare un gruppo di sviluppatori che lavorino insieme
- Tante semplici utility che possono essere combinate insieme (pipe line) per svolgere compiti molto meno semplice

Linux

- *Linux* è un «discendente/derivato» di *Unix*
- E' una famiglia *open-source* di sistemi operativi *Unix-like*
- basato sul kernel di *Linux* rilasciato per la prima volta nel 1991 da *Linus Torvalds*
- Sarebbe corretto in realtà parlare di sistema operativo *GNU/Linux*
- *POSIX* compatibile (più o meno...) ma non *POSIX-certified*
- Per chi è interessato ad approfondirne la storia...

Linux

- Varie distribuzioni
- *Debian*, *Ubuntu*, *RedHat*, *Fedora*, *Mint*, etc.

- segnalo questa introduzione di *The Linux Documentation Project*

macOS

- Un «gruppo» di sistemi operativi *Apple*
- Sistema proprietario
- Da tempo sono *Unix-like*, ma soprattutto Unix certified (da *The Open Group*)
- Processori: inizialmente *PowerPC*, poi dal 2006 *Intel*, poi dal 2020 processori *64-bit ARM-based Apple M1*
- recentemente (giugno 2022) *Apple* ha presentato il chip *M2*, un'evoluzione di *M1*

Voi siete troppo giovani...

Da Geeksforgeeks.org

- **Terminal :**
  A terminal is a text input and output environment.

  A terminal window, also known as a terminal emulator, is a text-only window that emulates a console in a graphical user interface (GUI).

- **Console :**
  A console is a physical terminal; an instrument panel containing computer controls. A console is a type of terminal.

- **Shell :**
  Shell stands for the command-line interpreter. A shell is a program that processes commands and outputs the results. A shell is a layer that sits on top of the kernel: 1) It interprets and processes the commands entered by the user. Unlike users, the shell has access to the kernel. Users can only gain access to the kernel by using a shell and entering commands (i.e. running programs). System calls are used by programs to gain access to kernel functionality. The system API is made up of all system calls.

Unix: autenticazione

- UNIX è multiutente
- ogni utente ha una *login*, una *password* e una *home directory* dove salvare i propri file
- Esiste un utente «privilegiato» che è ***root*** (amministratore del sistema)
- L'utente gestisce i permessi sui suoi file/cartelle
- Gli utenti possono essere aggregati in gruppi per definire privilegi comuni sui file/cartelle

Cos'è una *shell*

- E' l'interfaccia tra il sistema operativo e l'utente. Permette di eseguire i programmi installati nel sistema ed offre anche funzionalità simili ad un linguaggio di programmazione per automatizzare le operazioni da eseguire nel sistema
- Linea di comando
  - Bash, csh, ksh, prompt di MS/DOS, etc.
- Grafiche (GUI)
  - GNOME, KDE, Explorer, etc.

BASH

Shell

Comandi di sistema o built-in e applicazioni dell’utente

System Calls

Sistema Operativo

BASH

- Un normale programma che gira in spazio utente
- Espone il cosiddetto ‘*prompt*’ e attende che l’utente inserisca una ‘*linea di comando*’
- Interpreta ed esegue la linea di comando
- Attende un altra linea di comando
- Termina col comando exit o EOF (Ctrl+d)

BASH: (breve storia da *Wikipedia*)

- *Bash is a Unix shell and command language written by Brian Fox for the GNU Project as a free software replacement for the Bourne shell.*

- *The shell's name is an acronym for Bourne Again Shell*

- *First released in 1989, it has been used as the default login shell for most Linux distributions.*

- *Bash also was the default shell in all versions of Apple macOS prior to 2019*

BASH: risorse

- Advanced Bash-Scripting Guide

  da *The Linux Documentation Project (tldp.org).* Di Mendel Cooper

- Bash Reference Manual

  da *GNU* (‘*GNU's Not Unix*’). *Gnu.org*

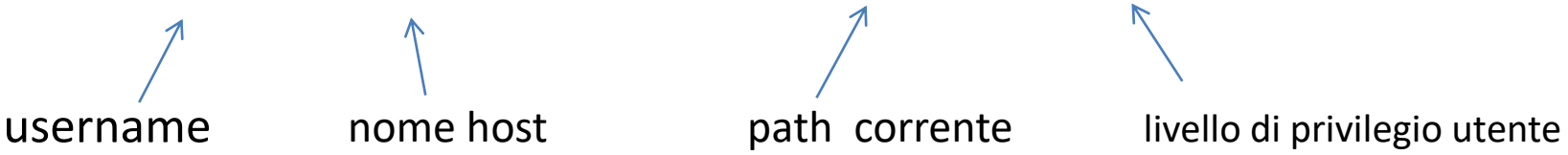
- *man bash* ... dalla *bash*...

- altre (buone) slide da *Internet* dall'Univ. di Pisa

BASH: prompt

- Il prompt è una stringa di caratteri con cui la shell segnala all'utente di essere in attesa di un comando
- Solitamente è:

fabio@fabiopc:/usr/lib/binfmt.d$



P.S.:
nome host: da non confondere con nome di dominio
path corrente: ogni applicazione ha in ogni istante una directory di default
livello di privilegio: '#' per root e '$' per gli altri

Perché mai usare una shell testuale?

- Ormai tutti i sistemi *Unix* hanno un'interfaccia grafica: perché usare i comandi in linea?

- Alcuni motivi:
  - per capire come funziona 'sotto'
  - per avere il massimo controllo sui comandi eseguiti e poter sfruttare tutte le opzioni a disposizione
  - perché vogliamo poter configurare le distribuzioni
  - perché i comandi *Unix* sono stati progettati per questo tipo di interfacce
  - perché una volta imparati sono più veloci e flessibili
  - perché sono veloci ed accessibili da remoto
  - perché sono sicuro di poter lavorare (una shell testuale l'avrò sempre, quella grafica forse...)
  - potrei non aver accesso all'interfaccia grafica
  - perché potrei avere un *HW* obsoleto e poco performante
  - .....

Perché mai usare una shell testuale?

- Un esempio:
  - devo cercare dove è definita la variabile ***MAXINT*** in una directory che contiene 390 file ****.h*** (***/usr/include***)
  - soluzione possibile:

    ```
    grep MAXINT /usr/include/*.h
    ```

  - risultato:

    ```
    values.h:#define        MAXINT          INT_MAX
    ```

BASH: singoli comandi

- Due categorie di comandi: *esterni* ed *interni*
  - ***Esterni***: eseguibili registrati nei supporti di memorizzazione (dischi, DVD, USB pen, etc.). Comprendono programmi per: manipolare file e directory, gestire i processi, utility varie, etc.
  - ***Interni***: riconosciuti ed eseguiti direttamente dalla shell. Comprendono anche i costrutti della *Bash* per lo *scripting*. Il comando (interno) **`help`** ne fornisce la lista

BASH: variabile $PATH

- Quando diamo un comando (esterno) dove è memorizzato l'eseguibile corrispondente? Comando **`which`**
- E dove viene cercato nei filesystem? Variabile di ambiente ***`PATH`***
- Comando **`help variables`**
- Comandi **`env`**, **`printenv`**, **`set`**, etc.

BASH: singoli comandi

- Sintassi tipica:

  nome_comando <opzioni> <argomenti>

- Ogni *comando* può accettare *parametri* (*opzioni* e *argomenti*) e restituisce un *valore di ritorno* che informa sull'esito (di solito 0 indica «tutto ok»)
- Gli argomenti seguono il nome del comando separati da spazi
- *man <nome comando>* per la documentazione

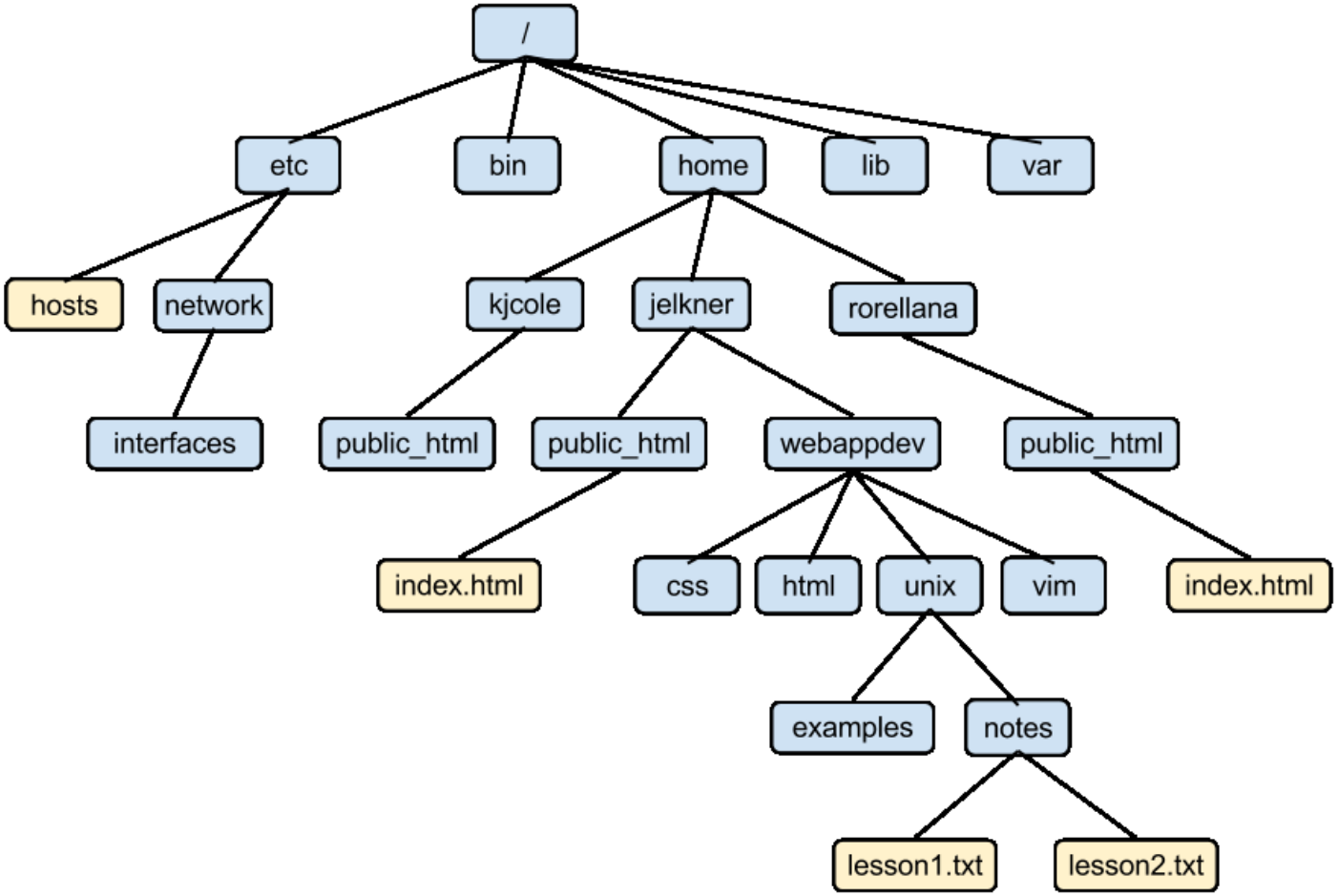
BASH: opzioni dei comandi

- Iniziano tipicamente col carattere '-'
- modificano la modalità di esecuzione del comando
- alcune non richiedono di essere seguite da un argomento (es.: *ls -l*). Altre richiedono un ulteriore argomento (es.: *tar –f file1.tar*)
- *long* e *short* format (es.: *ls -A equivalente a ls --almost-all*)
- Possono essere collassate insieme (es: *ls –ltra* equivale a *ls –l –t –r -a*)

BASH: non solo *man*

- ***man***: manuale in linea
- ***apropos***: lista i comandi che contengono la stringa passata come parametro nella descrizione sintetica
- ***whatis***: descrizione sintetica del comando passato come parametro
- ***locate***: ricerca i file aventi path contenente la stringa passata come parametro. Consulta un 'database' costruito con updatedb
- Generalmente ogni comando è in grado di fornire una descrizione
  sintetica delle sue funzionalità quando viene invocato con lo *switch* ***-h*** o ***--help***

Filesystem


/
etc
bin
home
lib
var
hosts
network
kjcole
jelkner
rorellana
interfaces
public_html
public_html
webappdev
public_html
index.html
css
html
unix
vim
index.html
examples
notes
lesson1.txt
lesson2.txt

Filesystem

- Il filesystem è una struttura ad albero al cui interno sono memorizzati (tutti) i dati

- E' organizzato in:
  - *File:*
  - *Directory (o cartelle):*
  - *Altro...*

- l'albero ha un'unica radice detta ***root***. E' la cartella «principale» che contiene tutte le altre (a differenza di *Windows* in cui ho più radici, i vari *drive* **C:**,**D**:, etc.). Il suo *path* è ***/***

- ciò significa che, anche se ho più dischi e partizioni, vedrò un unico albero dei dati

Filesystem: navigazione

- Ogni processo in ogni istante ha una ***directory corrente***
- ***pathname assoluto****:* ogni oggetto nel filesystem è univocamente determinato dal percorso che dalla root porta all'oggetto stesso. I pathname assoluti cominciano sempre con **/** (es.: */etc/hosts, /home/fabio/Desktop*)
- ***pathname relativo****:* percorso che porta all'oggetto a partire dalla directory corrente o comunque a partire da un certo punto del filesystem (es.: *log/apache, fabio/Desktop*)
- Nei pathname **.** indica la directory corrente
- **..** indica la directory «padre», cioè che contiene quella corrente

Filesystem: pathnames

Absolute:

/home/rorellana/public_html

/

/etc/network

Relative:

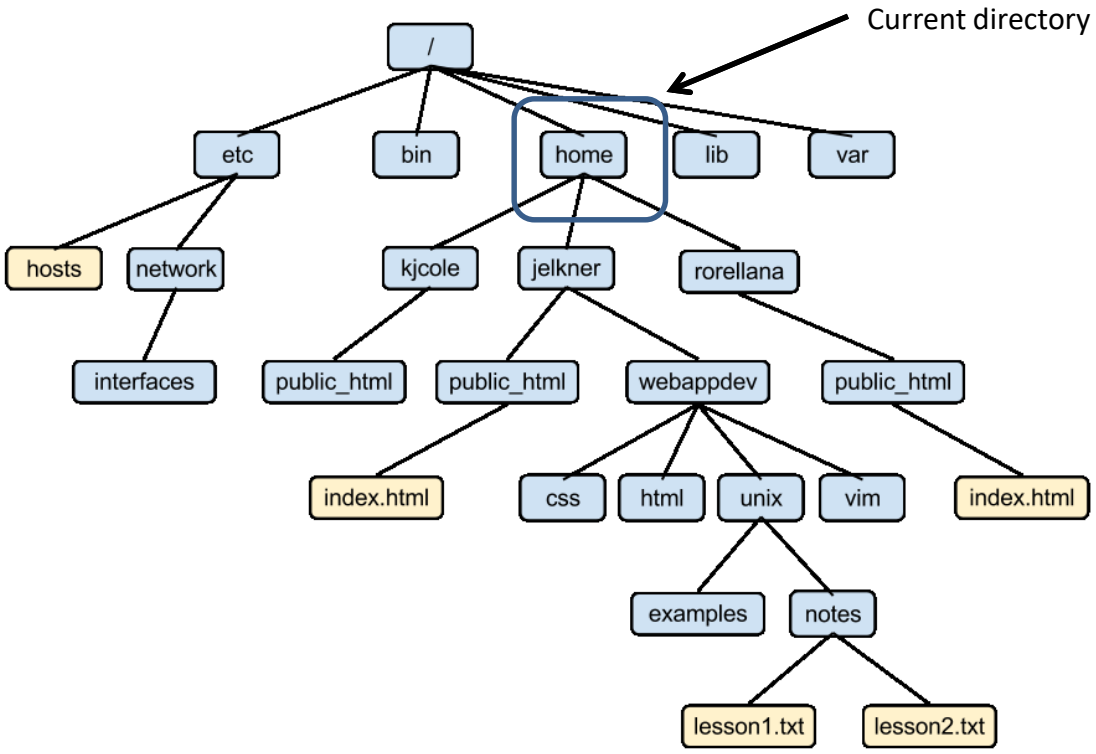
rorellana/public_html

kjcole/public_html

./jelkner

../bin

../etc/network



Filesystem: non solo file e cartelle

- Nel filesystem possono essere presenti file speciali chiamati ***link***. Sono puntatori/collegamenti ad altri oggetti del filesystem (analoghi ai *collegamenti* in *Windows*)

- Compiere un operazione su un link significa implicitamente compierla sull'oggetto cui punta

- In Unix sono presenti anche i ***special files*** o ***device files*** che «espongono» un dispositivo *HW* nel sistema come se fosse un file. Ciò fornisce un'interfaccia universale verso i dispositivi *HW* in modo che le system calls utilizzate per l'*I/O* sui file possano accedere allo stesso modo al *HW*

Filesystem: principali operazioni

- Listare (elencare) file e directory: ***ls***
  - **`ls`** fa vedere il contenuto della directory corrente
  - **`ls -l`** fa vedere il contenuto della directory corrente con informazioni in più
  - **`ls -l rata1`** elenca informazioni/contenuto per il file/directory **`rata1`**
  - **`ls -l lettera foto`** elenca informazioni/contenuto per i file/directory **`lettera`** e **`foto`**

Filesystem: principali operazioni

- Cambiare la directory corrente: ***cd***
  - **`cd /etc`** cambia la directory corrente in */etc*
  - in *Bash* è un comando interno (è la stessa shell che esegue il comando)

Filesystem: principali operazioni

- Copiare file: ***cp***
  - **`cp pippo pluto`** copia *pippo* in *pluto.*

    se *pluto* è una directory ed allora pippo viene copiato nella directory *pluto*. Se *pluto* esiste viene sovrascritto. Per copiare directory vedi punto successivo
  - **`cp -ar dir1 dir2`** copia ricorsivamente (opzione **`-r`**) *dir1* in *dir2*, cioè se *dir1* è una cartella ne copia tutto il contenuto ricorsivamente (tutto l'albero di cartelle /file sottostante). L'opzione **`-a`** serve per preservare gli attributi dei file/cartelle (per esempio la data di ultima modifica)

Filesystem: principali operazioni

- Spostare file o cartelle: ***mv***
  - **`mv pippo pluto`** sposta *pippo* in *pluto.*

    se *pluto* è una directory allora *pippo* viene spostato nella directory *pluto*. Se *pluto* esiste viene sovrascritto. Se *pippo* e *pluto* sono path che referenziano la stessa directory equivale a rinominare *pippo* in *pluto*

Filesystem: principali operazioni

- Creare cartelle: ***mkdir***
  - **`mkdir cartel1`**
- Rimuovere file: ***rm***
  - **`rm doc1`**
- Rimuovere cartelle (vuote): ***rmdir***
  - **`rmdir cartel1`** ma cartel1 deve essere vuota altrimenti il comando genera errore
- Rimuovere cartelle (non vuote): ***rm -r***
  - **`rm -r cartel1`** rimuove *cartel1* e, ricorsivamente, tutto il ramo di file/cartelle in essa contenuto

Filesystem: principali operazioni

- Creare collegamenti a file o directory: ***ln***
  - **`ln pippo link1`** crea il collegamento (***hard link***) *link1* a *pippo*. In breve: crea una copia (occupa spazio nel disco) di *pippo*, se il file originario viene cancellato *link1* rimane in alterato ed anche il suo contenuto
  - **`ln -s pippo link1`** crea il collegamento simbolico (***symbolic link***) *link1* a *pippo (o*ccupazione minimale nel disco*)*. In breve: link1 contiene una stringa che è il path di *pippo*. Se il file originario viene cancellato *link1* referenzia sempre lo stesso path ma diventa un collegamento non valido

BASH: filesystem

pwd	visualizza la directory corrente
cd (-)	cambia la directory corrente
ls (-lrtaRh)	elenca oggetti del filesystem
df (-hi)	elenca i file system montati
cat	visualizza il contenuto di un file
cp (-ar)	copia file(s)
mv	sposta file
rm (-r)	cancella file
rmdir	cancella directory

BASH: filesystem

ln (-s)	crea un collegamento (*link*)
realpath	visualizza il path assoluto risolto, per es.: anche per *link* o riferimenti relativi
file	ritorna il tipo di un file (testo, binario, pdf, etc.). Indipendentemente dall'estensione....
tail (-fn)	visualizza le ultime *n* linee di un file
head (-n)	visualizza le prime *n* linee di un file