Sistemi UNIX

Tullio Facchinetti

10 aprile 2024

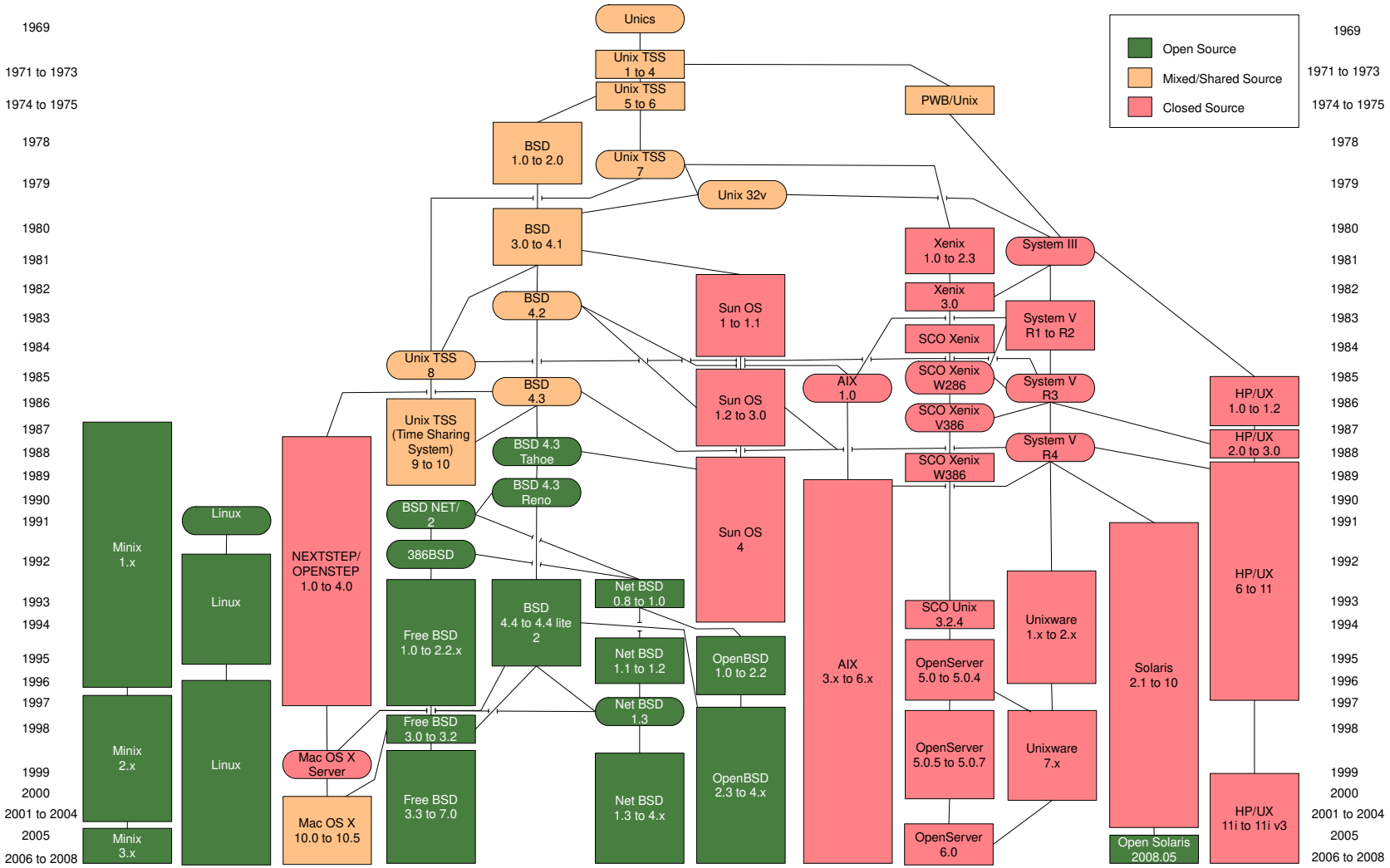
http://robot.unipv.it/toolleeo

UNIX (Linux) e il C

la storia del linguaggio C è strettamente legata a quella di UNIX – e oggigiorno di Linux

- negli anni '70 il C è nato per sviluppare programmi di sistema e driver per UNIX
- il kernel di Linux è implementato in C

Storia dei sistemi UNIX



Fonte: `https://www.cs.miami.edu/home/wuchtys/`

Autenticazione (login)

- l'**autenticazione**, o **login**, è necessaria per l'accesso al sistema
- permette il collegamento di più utenti, contemporaneamente o in momenti diversi
- è così possible individuare univocamente l'identità dell'utente
- permette di gestire i permessi di accesso a file e directory: ogni utente ha i propri
- storicamente questa informazione veniva usata per contabilizzare il tempo di utilizzo del calcolatore, che era una risorsa molto limitata, per farlo poi pagare

Username e password

- è basata sulla coppia username/password che devono essere forniti al sistema
- il processo di autenticazione inizia scrivendo il **nome utente**, detto anche **username**
- lo username viene assegnato dall'amministratore del sistema
- lo username identifica univocamente l'utente
- i caratteri della password sono tipicamente occultati, cioè non vengono visualizzati sul video
- questo evita che la password possa essere "rubata" da malintenzionati eventualmente appostati alle spalle dell'utente che sta effettuando l'autenticazione

Autenticazione (login): esempio

```
login: utente007
Password: pl67kf2
user007@europa:~$
```

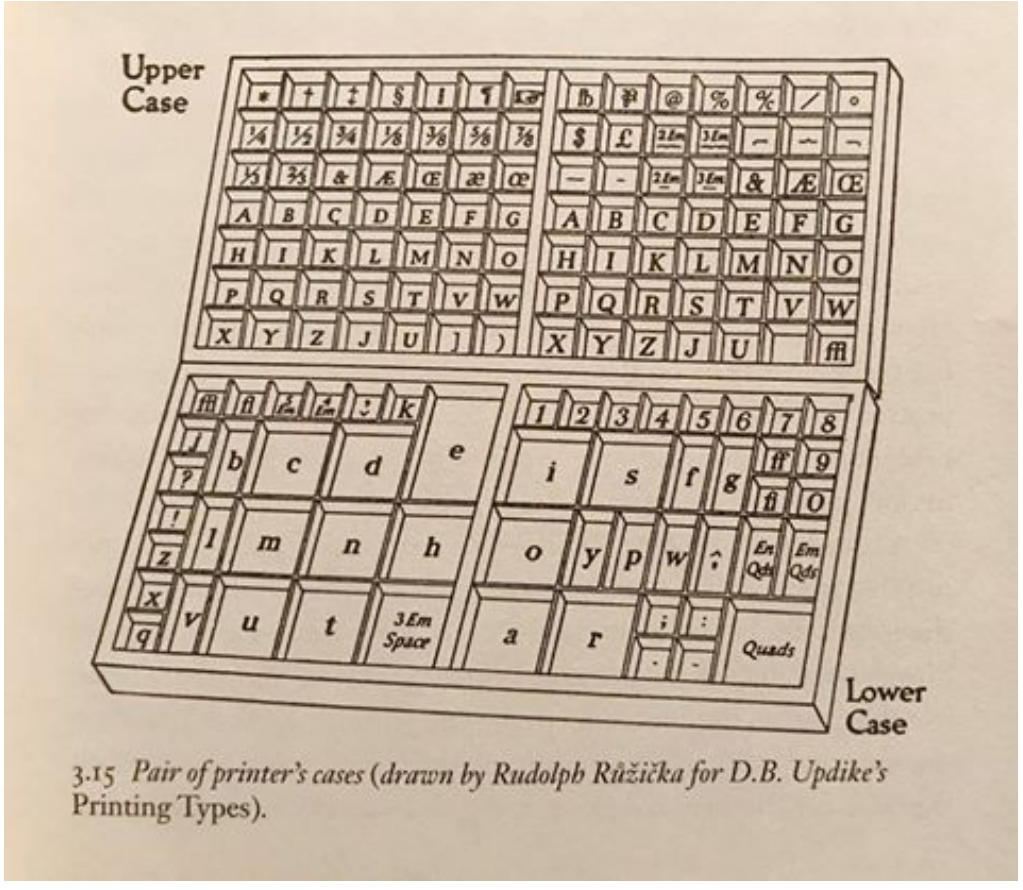
- la schermata di login può variare a seconda del sistema
- spesso si tratta di una schermata grafica con pulsanti, loghi e immagini
- nell'esempio la password viene mostrata per esigenze didattiche; normalmente i caratteri non sono visualizzati
- se l'accesso ha successo, viene presentato il **prompt**
- in questo caso, il prompt è formato dai caratteri `user007@europa:~$`

Case sensitivity di username e password

username e password sono case-sensitive

- il **case** di un carattere indica il fatto che esso sia maiuscolo o minuscolo
- un sistema si dice *case sensitive* se esso è sensibile al *case*, ovvero se viene fatta distinzione tra caratteri maiuscoli e minuscoli
- pertanto, per esempio, le parole `user007`, `USER007` e `User007` sono considerate tutte diverse tra loro

Origine del termine "case"



3.15 *Pair of printer's cases (drawn by Rudolph Růžička for D.B. Updike's* Printing Types).

Il *case* (*contenitore* o *cassa* in inglese) era la scatola che conteneva i punzoni per le prime macchine di stampa.

La parte alta della scatola, cioè l'*upper case*, conteneva i punzoni per le lettere maiuscole.

La parte bassa, ovvero il *lower case*, conteneva i punzoni per le lettere minuscole.

Autenticazione (login)

- gli errori nell'inserimento di username o password vengono opportunamente notificati
- la richiesta di login viene ripresentata finché l'autenticazione non ha successo

```
login: utente1
Password:
Login incorrect

login:
```

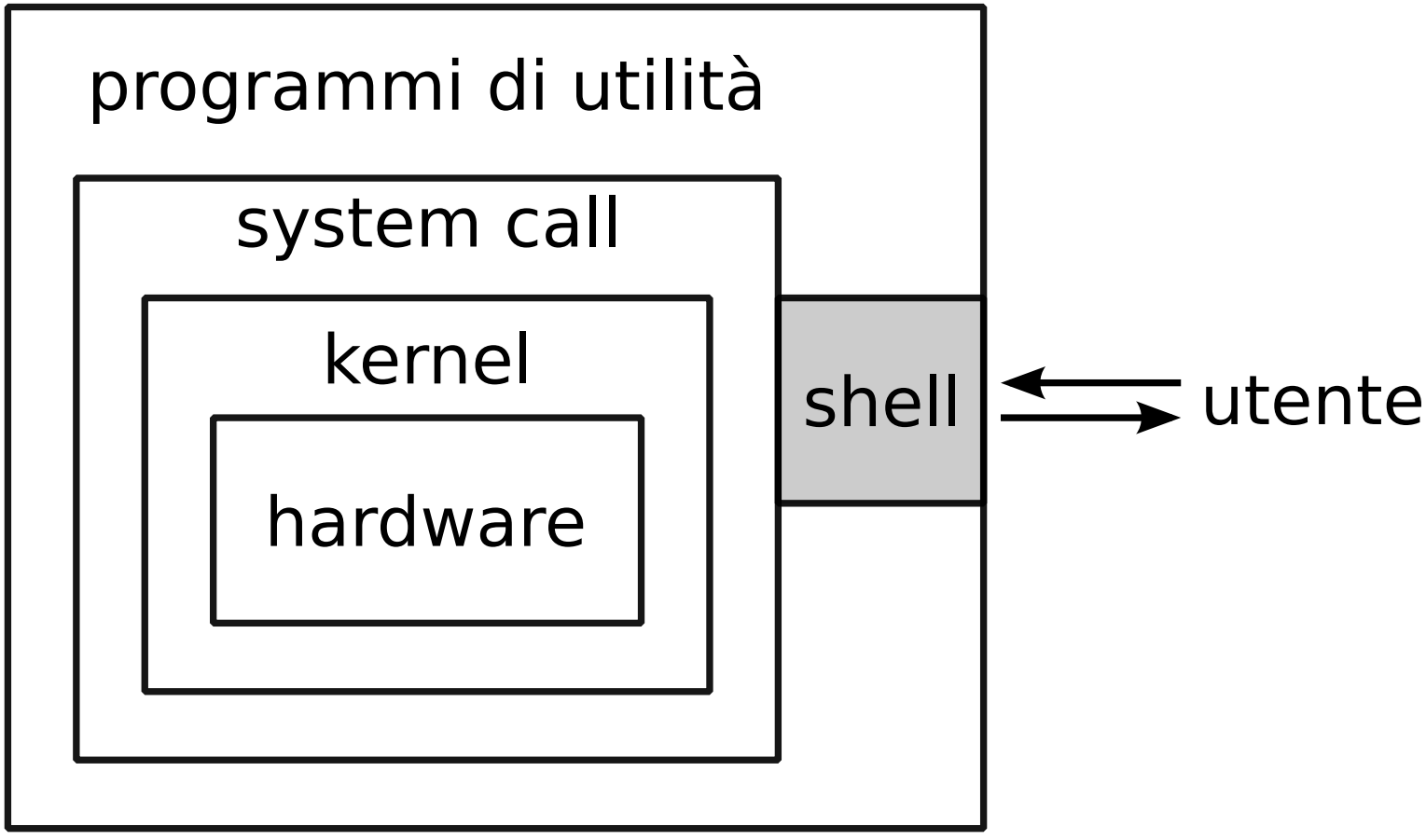
l'username `root` è associato ad un utente speciale con funzioni di amministrazione e gestione del sistema; molte funzioni sono accessibili soltanto a tale utente

L'interprete dei comandi: la shell

la **shell** è un programma che riceve i comandi, li intepreta, ed esegue le operazioni associate

- è così chiamata in quanto costituisce lo strato più esterno del sistema, cioè quello più vicino all'utente
- UNIX può infatti essere pensato come un insieme di strati logici posti tra la macchina e l'utente, il più esterno dei quali è appunto la shell

L'interprete dei comandi: la shell



Siamo nell'anno XXXX, che utilità ha la shell?

```
│       └── output
├── srt_sched
│   ├── docs
│   └── figs
├── srt_shared
│   └── figs
└── wireless_networks
    └── figs

63 directories
toolleeo@zbtool:slides$ cd -
/home/toolleeo/work/slides/fdi_unix
toolleeo@zbtool:fdi_unix$ ls
figs             Makefile     unix.aux        unix.inc.tex  unix.nav  unix.pdf  unix.tex  unix.vrb
generate_tree.sh  outfile.snX  unix_draft.pdf  unix.log      unix.out  unix.snm  unix.toc
toolleeo@zbtool:fdi_unix$ ll
total 1724
-rw-r--r-- 1 toolleeo toolleeo    278 Mar 10 15:48 generate_tree.sh
-rw-rw-r-- 1 toolleeo toolleeo      0 Mar 10 15:48 outfile.snX
-rw-r--r-- 1 toolleeo toolleeo     94 Mar 10 15:48 Makefile
-rw-r--r-- 1 toolleeo toolleeo    403 Mar 10 15:49 unix.tex
drwxr-xr-x 2 toolleeo toolleeo   4096 Mar 10 16:21 figs
-rw-rw-r-- 1 toolleeo toolleeo    525 Mar 10 16:41 unix.vrb
-rw-rw-r-- 1 toolleeo toolleeo    149 Mar 10 16:41 unix.toc
-rw-rw-r-- 1 toolleeo toolleeo      0 Mar 10 16:41 unix.snm
-rw-rw-r-- 1 toolleeo toolleeo    170 Mar 10 16:41 unix.out
-rw-rw-r-- 1 toolleeo toolleeo   8833 Mar 10 16:41 unix.nav
-rw-rw-r-- 1 toolleeo toolleeo  13070 Mar 10 16:41 unix.aux
-rw-rw-r-- 1 toolleeo toolleeo 778608 Mar 10 16:41 unix.pdf
-rw-rw-r-- 1 toolleeo toolleeo  79844 Mar 10 16:41 unix.log
-rw-rw-r-- 1 toolleeo toolleeo 778608 Mar 10 16:41 unix_draft.pdf
-rw-rw-r-- 1 toolleeo toolleeo  59733 Mar 10 16:41 unix.inc.tex
toolleeo@zbtool:fdi_unix$
 0  5d 16m  1 bash                                  94% | 16:44 | 10 Mar  toolleeo  zbtool
```

La shell, con la sua interfaccia a caratteri, può sembrare una reminiscenza degli albori dell'informatica

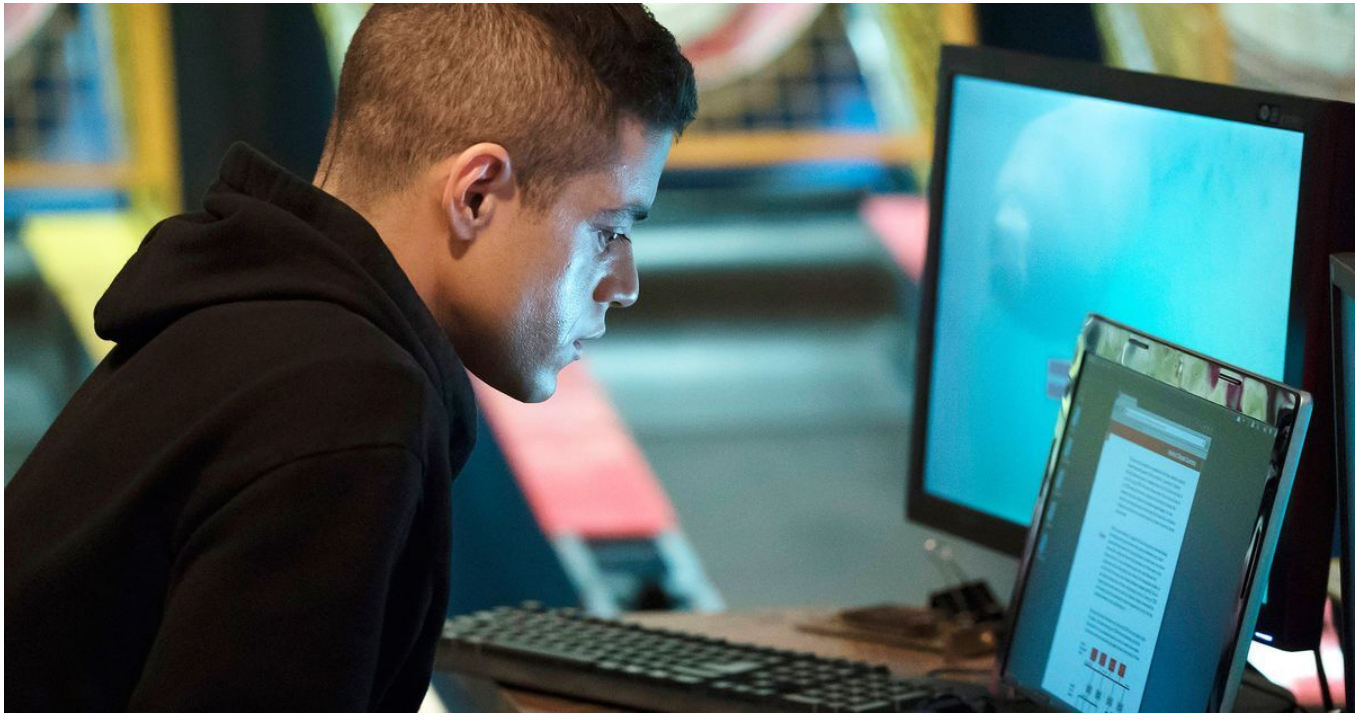
Siamo nell'anno XXXX, che utilità ha la shell?

- La shell è tuttora molto utilizzata in vari ambiti dell'informatica
- Essendo in uso da vari decenni, è particolarmente robusta e collaudata
- Rappresenta lo strumento più utilizzato per accedere a computer in remoto

Permette di automatizzare molte operazioni combinando i programmi e gli strumenti disponibili

Permette - e richiede! - una buona conoscenza del funzionamento del computer e degli strumenti necessari per utilizzarlo

Siamo nell'anno XXXX, che utilità ha la shell?



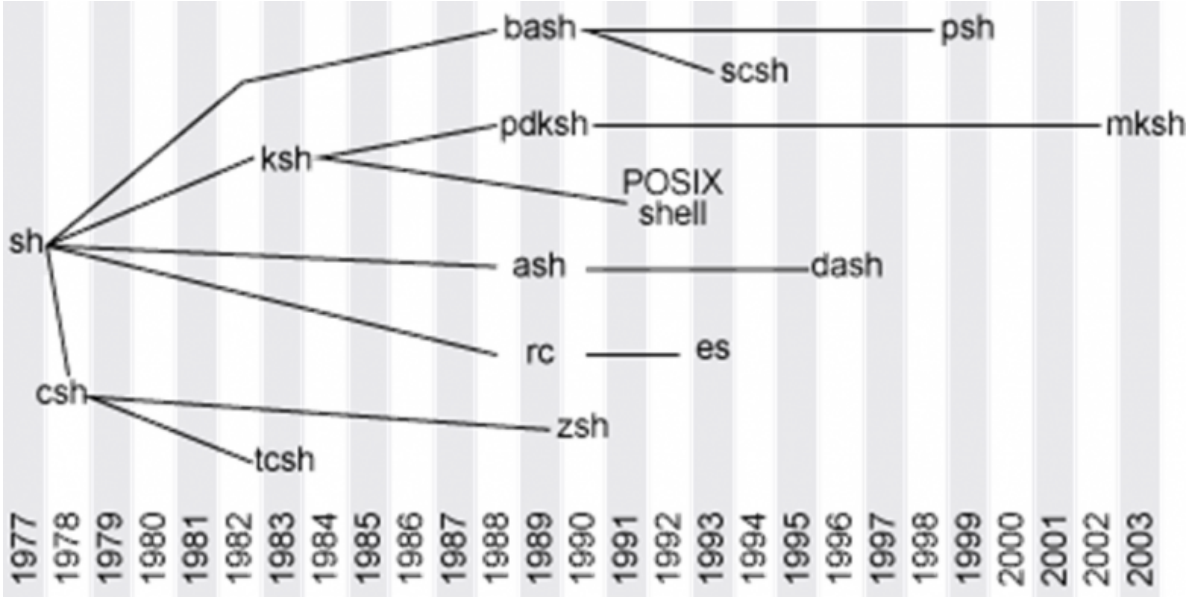
A qualcuno piace aggiungere: permette di impressionare gli amici e i conoscenti con le proprie abilità :-)

L'interprete dei comandi: la shell

- l'interazione tra l'utente e l'interprete avviene per mezzo della cosiddetta **linea di comando**
- l'utente inserisce i comandi sotto forma di **stringhe di testo**, ovvero sequenze di caratteri
- i caratteri inseriti dall'utente sono riprodotti sullo schermo accanto al prompt
- la riga di comando viene presa in considerazione dalla shell solo quando viene premuto il tasto Invio/Enter/Return
- fino ad allora, il suo contenuto si trova in un'area provvisoria detta **buffer di input**, e può essere corretto con il tasto di cancellazione

Vari tipi di shell

Esistono vari programmi che implementano la shell, tutti più o meno compatibili tra loro in termini di sintassi



- Oggigiorno esistono anche Fish, Xonsh, cosh e altre
- La shell più utilizzata è **Bash**

Il prompt

Il **prompt** è il segnale con cui la shell si mostra pronta a ricevere altro input dopo aver elaborato l'input precedente

Il prompt può essere personalizzato per mostrare informazioni quali lo username, il nome della macchina che si sta utilizzando (utile quando ci si collega a computer remoti), e la directory corrente.

Esempio:

```
user1@europa:~$
```

Il prompt

Il prompt può essere scomposto nei seguenti elementi:

```
user1   @   europa   :   ~   $
```

`user1` è lo username

`@` è un carattere convenzionale (si legge "at") che introduce il nome del computer al quale si è collegati

`europa` è il nome del computer in uso

`:` è un carattere convenzionale che introduce il percorso della directory corrente

$\sim$ è la directory corrente

`$` è un carattere convenzionale che delimita il prompt dall'input dell'utente

Il file system

- il **file system** è una struttura utilizzata per memorizzare e organizzare le informazioni in un sistema di calcolo
- i dati sono immagazzinati all'interno di **file**, che sono costituiti quindi dal blocco di informazioni che si intende memorizzare
- i file non contengono soltanto dati, ma possono contenere le istruzioni da eseguire corrispondenti ad un determinato programma (i cosiddetti programmi eseguibili)

I nomi dei file

Ad ogni file è associato un **nome**

Il nome può essere composto dai seguenti caratteri:

`A...Z a...z 0...9 _ - . ,`

- cambiando sistema operativo cambia anche l'insieme dei caratteri validi per un nome di file
- i caratteri riportati sono validi per pressoché ogni sistema operativo, e quindi in particolare anche per UNIX

I nomi dei file

I sistemi di tipo UNIX distinguono tra lettere maiuscole e minuscole (sono *case-sensitive*)

- alcuni esempi di nomi di file sono: `lezione`, `lezione.doc`, `LEZIONE.doc`, `Lezione.doc`, `lezione.old`, `file_mio`, `19.nov.92`
- questi nomi di file rappresentano tutti file diversi tra loro

Estensioni dei file

Il nome di un file è tipicamente costruito nella forma `nome.estensione`

- `nome` rappresenta il nome vero e proprio del file che ne identifica il contenuto
- `estensione` è una sequenza di caratteri che indica il tipo di dati contenuti del file
- nome ed estensione sono tipicamente separati dal punto

Estensioni dei file

- l'uso dell'estensione è non indispensabile
- è però utile per comprendere immediatamente la tipologia del contenuto del file

esempi

- le estensioni `.jpg`, `.gif`, `.png` o `.svg` contengono immagini in vari formati con caratteristiche diverse
- le estensioni `.c`, `.cpp`, `.java`, `.py` e `.m` sono utilizzate per i file sorgente di programmi scritti rispettivamente in linguaggio C, C++, Java, Python e Matlab
- le estensioni `.doc`, `.odt`, `.xls` e `.ppt` sono usate da programmi da ufficio

Estensioni dei file

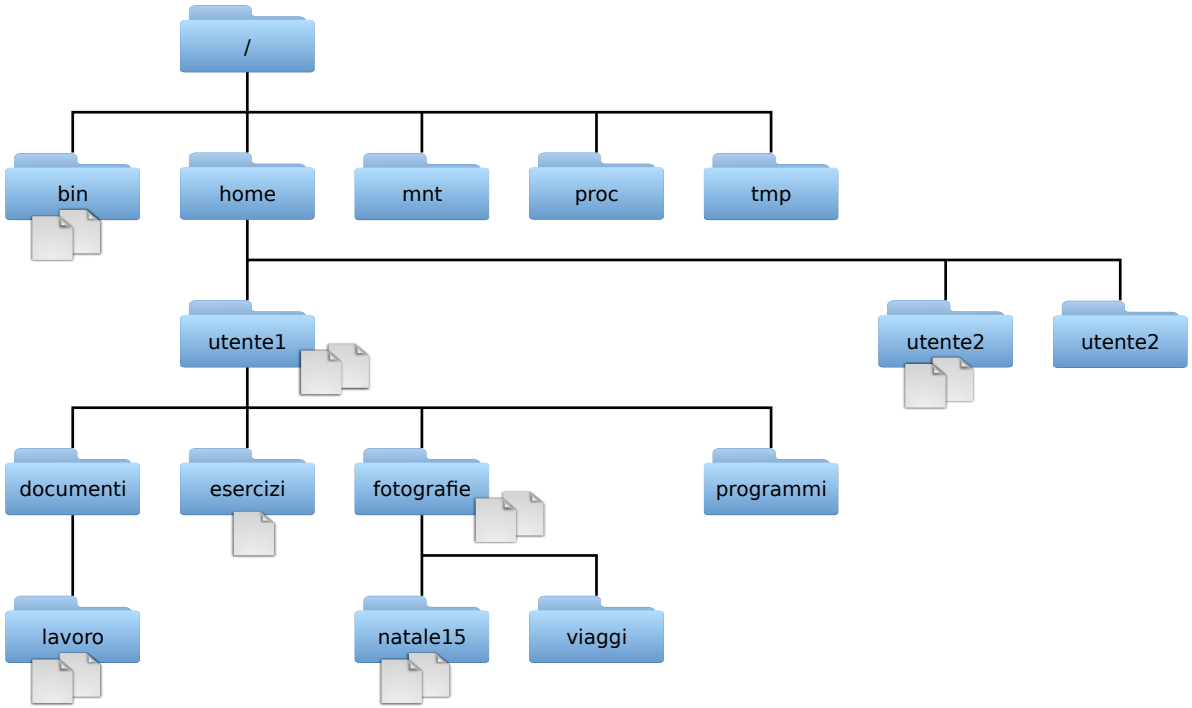
- esistono migliaia di estensioni diverse
- in alcune situazioni, la stessa estensione ha significati diversi, anche se questi sono casi molto particolari, poco comuni e soprattutto non interessano le più comuni estensioni di file
- due file indentici per quanto riguarda il contenuto potrebbero avere estensioni diverse

> l'uso non convenzionale delle estensioni ha il solo risultato di confondere le idee

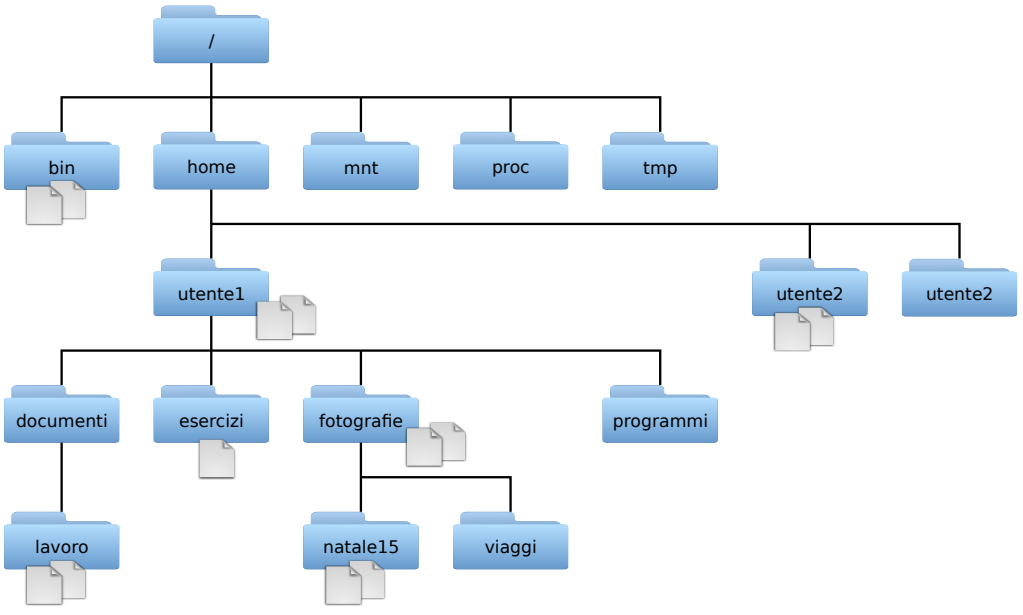
- il carattere punto non necessariamente viene soltanto utilizzato solo per introdurre l'estensione
- per esempio, il nome `archivio.tar.gz` è lecito e indica un file archiviato col comando `tar` di UNIX e compresso col programma `gzip`

Le directory

Le **directory** servono per organizzare in modo gerarchico i file, e costituiscono il file system



Le directory: esempio di albero delle directory



- ogni file ha una directory che lo contiene
- la struttura delle directory è quella di un **albero**
- ogni directory ha un solo genitore, in cui è contenuta e di cui si dice figlia

- la struttura delle directory si dice gerarchica perché la relazione genitore-figlio determina una gerarchia
- di norma una qualsiasi directory (es. `utente1`) si trova dentro un'altra directory

La directory radice

esiste una directory che non è contenuta in nessun'altra: si chiama `root` (radice dell'albero) e si indica con la barra /

- la barra / che indica la directory `root` si chiama **slash**
- ATTENZIONE: non è da confondersi con la barra \, la quale è chiamata *backslash*

La directory corrente

ad ogni istante è definita una directory corrente
o directory di lavoro

- la directory corrente è quella nella quale "ci si trova" in un determinato momento durante l'uso della macchina
- la directory corrente può essere cambiata in caso di necessità: ci si può spostare nell'albero delle directory

Percorsi o pathname

- directory o file diversi possono avere lo stesso nome, ma solo qualora siano contenuti in directory diverse
- pertanto, file e directory, intesi come elementi di un albero, non possono essere individuati univocamente soltanto dal nome

Esempio:

Il file `immagine01.jpg` può essere contenuto della directory `fotografie` oppure `documenti`.

Percorsi assoluti

Ad ogni file o directory è associato un **percorso**, o **pathname**

Il percorso e il nome del file identificano univocamente un file.

Ci sono due tipi di percorso:

- percorsi **assoluti**
- percorsi **relativi**

Percorsi assoluti

Si consideri il file o directory di nome `oggetto`

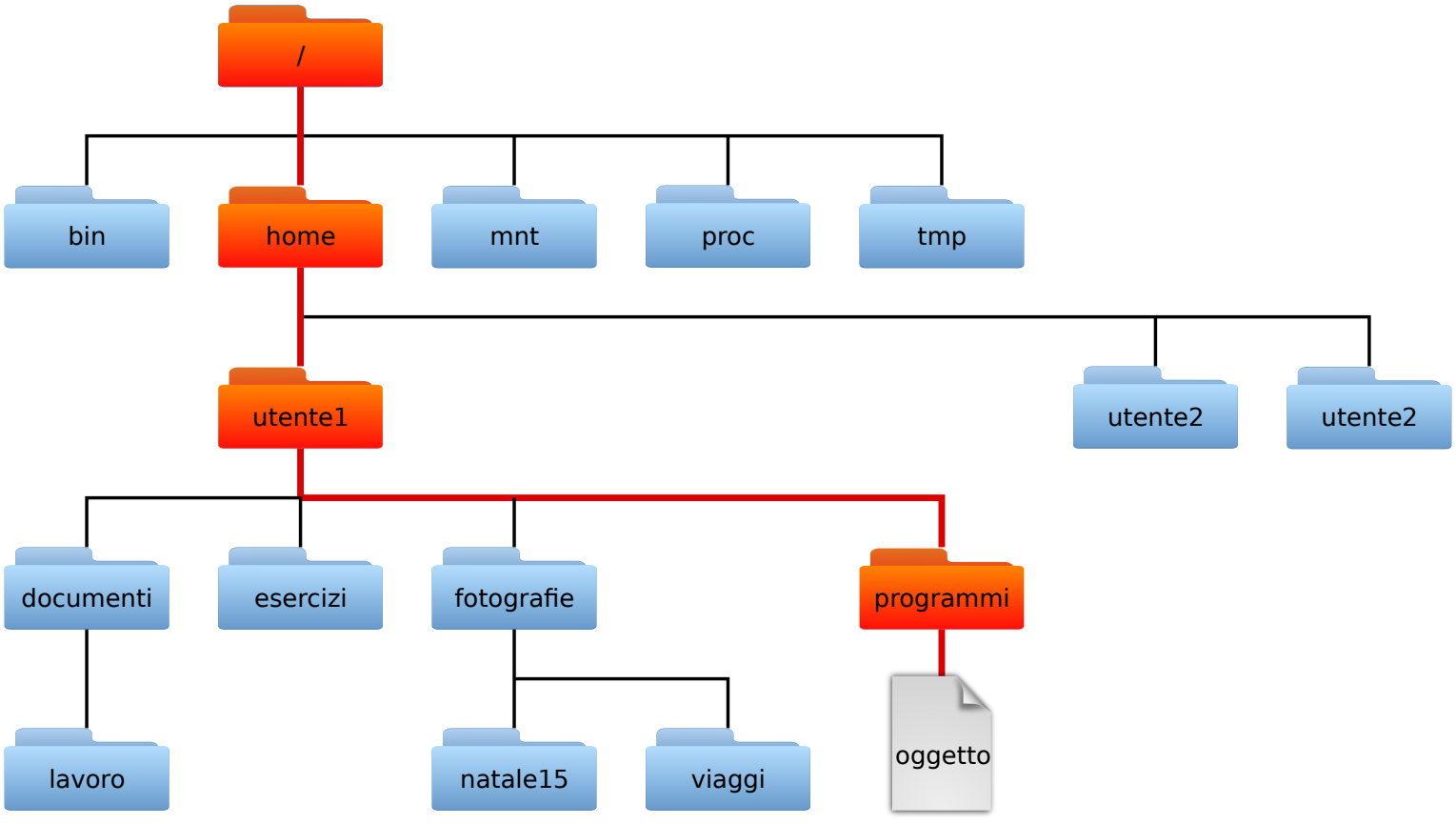
Percorso assoluto

1. localizza `oggetto` sull'albero rispetto alla directory root
2. è identificato dalle directory da / fino al file `oggetto` compreso, separati da /

Caratteristiche:

- dipende solo dalla posizione del file nel file-system
- è unico
- inizia sempre col carattere /

Esempio di percorso assoluto



Percorso assoluto: `/home/utente1/programmi/oggetto`

Percorsi relativi

Si consideri il file o directory di nome `oggetto`

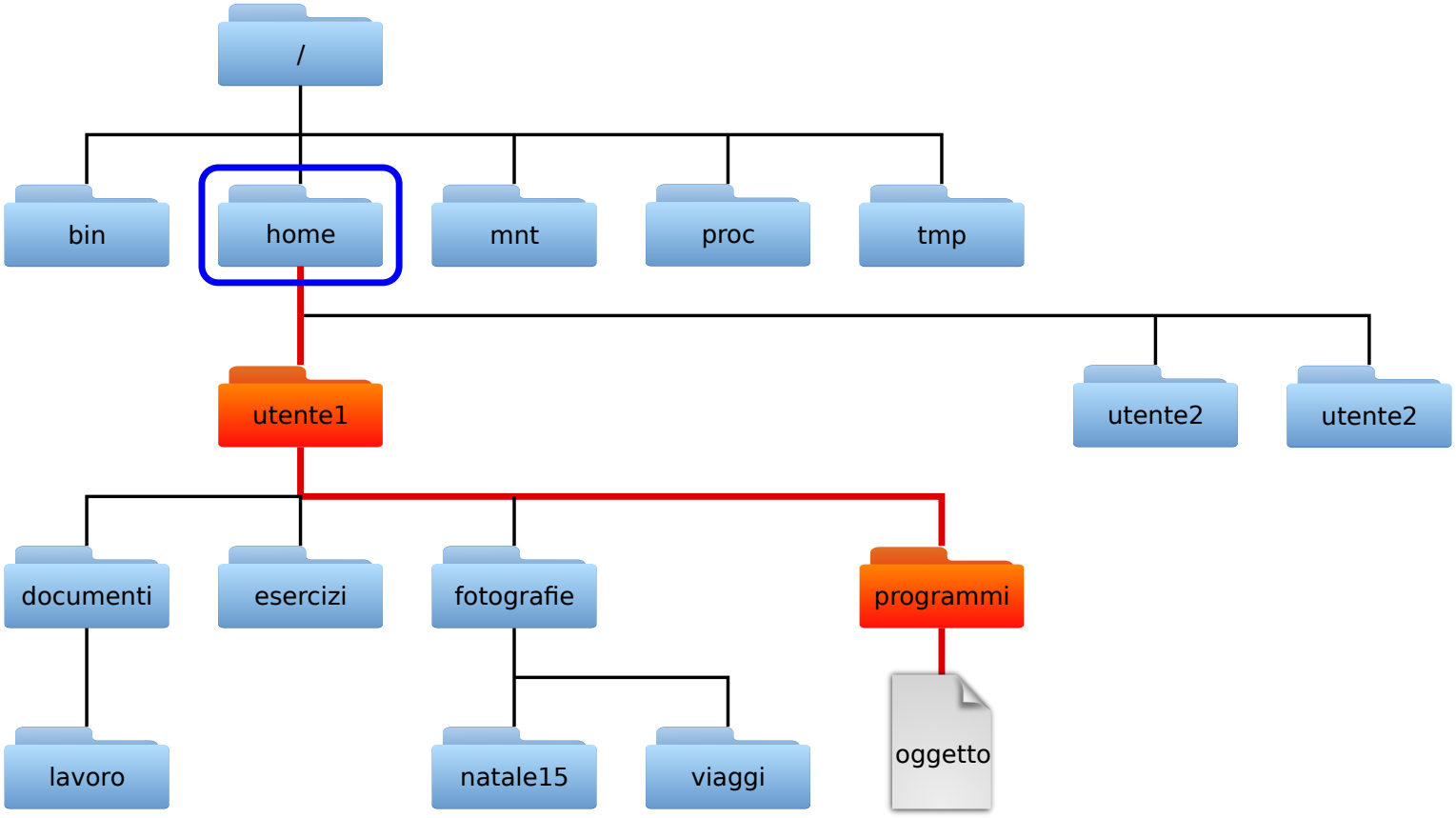
Percorso relativo

1. localizza `oggetto` rispetto alla directory corrente
2. è dato dalle directory a partire da quella corrente (esclusa) fino a `oggetto` compresa, separati da /

Caratteristiche:

- dipende dalla directory corrente
- il percorso non inizia mai col carattere /

Esempio di percorso relativo



percorso relativo: `utente1/programmi/oggetto`

directory corrente: `/home`

Percorsi relativi e assoluti

Alcune caratteristiche dei percorsi relativi e assoluti:

- un percorso assoluto inizia per /, altrimenti è relativo
- in ogni caso, un percorso relativo NON inizia con lo slash

Nella scrittura di un percorso, il carattere / ha due usi:

1. indica la directory `root` se posto all'inizio come primo carattere del pathname
2. funge da separatore di directory se posto in mezzo al pathname