SISTEMA OPERATIVO UNIX

FILE SYSTEM

- FILE COME **SEQUENZA DI BYTE**
  NON sono pensate organizzazioni logiche o accessi a record

- FILE SYSTEM **gerarchico**
  ALBERO di sottodirettori

- OMOGENEITÀ **dispositivi** e **file**
  TUTTO è file

FILE

astrazione unificante del **sistema operativo**

file ordinari

file direttori, accesso ad altri file

file speciali (dispositivi fisici), contenuti nel direttorio /dev

ORGANIZZAZIONE del FILE SYSTEM

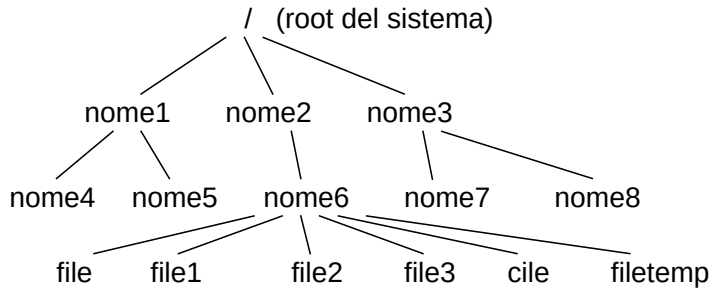
NOMI di file
- **ASSOLUTI**: dalla radice
  /nome2/nome6/file
- **RELATIVI**: dal direttorio corrente
  nome6/file

I nomi (e la sintassi in generale) sono **case-sensitive**

Si possono usare abbreviazioni nei nomi (**wild card**)
***** per una qualunque stringa
? per un qualunque carattere

Direttorio corrente identificato da **.**
Padre del direttorio corrente identificato da **..**

Ogni utente ha un direttorio a default
direttorio in cui l'utente si colloca all'ingresso nel sistema



file* ==> file, file1, file2, file3, filetemp
file? ==> file1, file2, file3
*ile ==> file, cile

Accesso al Sistema Operativo Unix
Ingresso (LOGIN)

Username: torroni
Password: **********

L'accesso viene verificato al login (autenticazione utente)

Uscita **(LOGOUT)**

L'uscita avviene solo al logout, tramite uno dei comandi:

- **exit**
- **logout**
- **CTRL-D**

Durante una sessione di lavoro, i comandi dati dall'utente sono interpretati da una **shell (interprete dei comandi)**

La shell non è unica: un sistema può metterne a disposizione varie (e altre possono essere aggiunte)

Ogni utente può **specificare quale shell desidera usare**

La shell associata a ogni utente è specificata all'interno del file /**etc/passwd**, che costituisce il "database" degli utenti registrati su quel sistema (con tutte le loro caratteristiche).

FORMATO DEL FILE /etc/passwd

utente:password:UID:GID:commento:direttorio:comando

utente ➔ nome che bisogna dare al login
password ➔ password che occorre dare al login **(memorizzata in forma cifrata)**

UID ➔ **User IDentifier** numero che identifica in modo univoco l'utente nel sistema

GID ➔ **Group IDentifier** numero che identifica il gruppo di cui fa parte l'utente

commento ➔ campo di commento

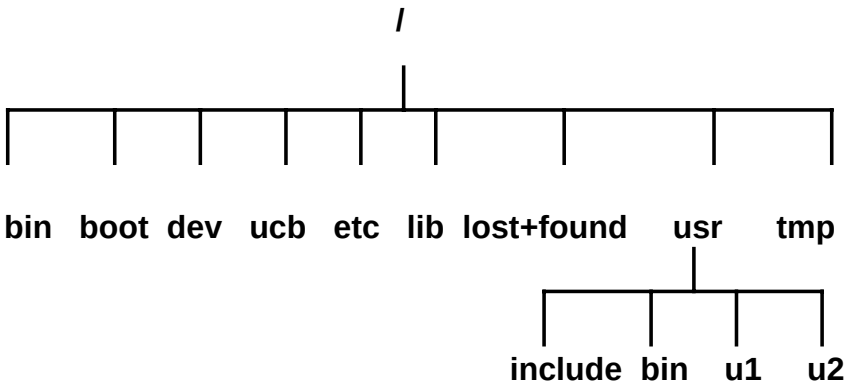
direttorio ➔ direttorio iniziale in cui si trova l'utente al login (home directory)

comando ➔ comando che viene eseguito automaticamente all'atto del login **(in genere è una shell)**

ESEMPIO:

root:.XPc4HKe0nPQA:0:1:Operator:/:/bin/sh
torroni:rTIW65BOuQ9ng:230:30:Paolo Torroni:/home/torroni:/bin/bash

STRUTTURA DEL FILE SYSTEM



bin comandi principali di sistema

dev i dispositivi

etc, file significativi per il sistema,
ad es. **passwd** che memorizza gli utenti autorizzati

lib, le librerie di sistema

tmp, file temporanei

usr, sottodirettorio per utenti

Esiste un utente, **root**, privilegiato rispetto agli altri:
il gestore del sistema (detto anche super-user) che sfugge alle regole di sicurezza

PROTEZIONE dei File

Molti utenti ➔ Necessità di regolare gli ***ACCESSI*** alle informazioni

Per un file, esistono 3 tipi di utilizzatori:
- il proprietario, **user**
- il gruppo del proprietario, **group**
- tutti gli altri utenti, **others**

Per ogni tipo di utilizzatore, si distinguono tre modi di accesso al file:
- **lettura (r)**
- **scrittura (w)**
- **esecuzione (x)**

Ogni utente è caratterizzato da:
- l'identificatore di utente (**UID, user ID**) e
- l'identificatore di gruppo (**GID, group GID**)

Ogni file è marcato con
- lo User-ID e il Group-ID **del proprietario**
- un insieme di 12 bit di protezione

12	*11*	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*
0	0	0	1	1	1	1	0	0	1	0	0
SUID	SGID	sticky	R	W	X	R	W	X	R	W	X
			User			**G**roup			**O**thers		

i primi 9 ==> triple di permessi

SIGNIFICATO DEI 12 BIT

12	*11*	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*
0	0	0	1	1	1	1	0	0	1	0	0
SUID	SGID	sticky	R	W	X	R	W	X	R	W	X
			User			**G**roup			**O**thers		

9 bit più a destra (bit 1-9) ➔ 3 triple di permessi:
- **lettura / scrittura / esecuzione**

rispettivamente per
- **il proprietario / quelli del suo gruppo / gli altri**

dodicesimo bit
SUID (Set-User-ID) (identificatore di utente effettivo)
Si applica a un file di **programma eseguibile**
Se vale 1, fa sì che *l'utente che sta eseguendo* quel programma venga considerato *il proprietario* di quel file (solo per la durata della esecuzione)

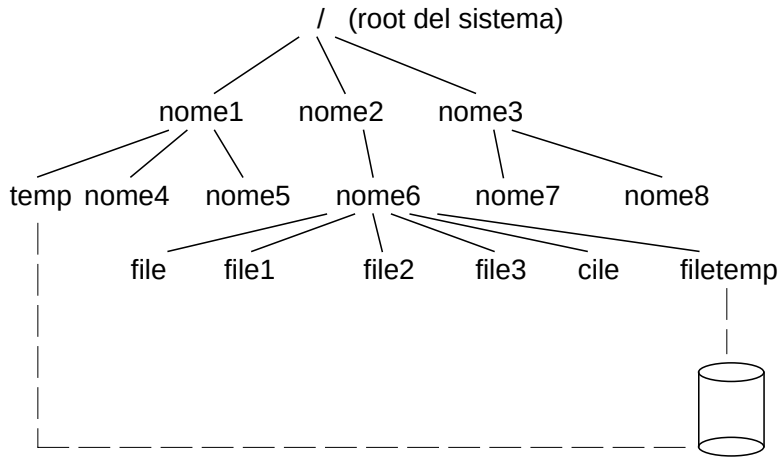
Questo bit è necessario per consentire, durante l'esecuzione di quel programma, operazioni di lettura/scrittura su file di sistema, che l'utente non avrebbe il diritto di leggere / modificare.

Es.: comando **passwd** per modificare la password di utente. Questa si trova (in forma cifrata) in un file di proprietà di **root**, quindi non potrebbe essere modificato da un altro utente. Solo il SUID lo rende possibile.

SGID bit: come SUID bit, per il gruppo
sticky bit (non più utilizzato: ignorato ad es. da Linux)
il sistema cerca di mantenere in memoria l'immagine del programma, anche se non è in esecuzione

Collegamenti (Link)

Le informazioni contenute in uno stesso file possono essere **visibili come file diversi**, tramite "riferimenti" (link) allo stesso file fisico.



Il sistema considera e tratta il tutto:
- se un file viene cancellato, le informazioni sono veramente eliminate solo se non ci sono altri link a esso
- Il link cambia i diritti? ➔ Meglio di no

Due tipi di link:
- link fisici (si collegano le strutture del file system)
- link simbolici (si collegano solo i nomi)

comando: ln [-s]

ln /nome2/nome6/filetemp /nome1/temp

Si vedano i link di ogni direttorio

SHELL ovvero
il **PROCESSORE COMANDI**

Lo shell interprete dei comandi:
esegue uno dopo l'altro i comandi forniti

loop forever
<**accetta** comando da console>
<**esegui** comando>
end loop;

Lo shell è un **processore comandi**, che **accetta comandi** da terminale o da un file comandi e li **esegue** fino alla fine del file
(si può usare <CTRL><D>)

loop forever
< LOGIN >
repeat
<accetta comando da console>
<esegui comando>
until <fine file>
< LOGOUT >
end loop

Maggiori capacità rispetto a un semplice esecutore di un comando alla volta

COMANDI RELATIVI AL FILE SYSTEM

Sintassi:

comando [**-opzioni**] [**argomenti**] <INVIO>

Un comando termina con un INVIO, oppure è separato con **;** da altri comandi nella stessa linea

CREAZIONE / GESTIONE DI DIRETTORI

mkdir	<nomedir>	*creazione di un nuovo direttorio*
rmdir	<nomedir>	*cancellazione di un direttorio*
cd	<nomedir>	*cambio di direttorio*
pwd		*stampa il direttorio corrente*
ls	[<nomedir>]	*visualizz. contenuto del direttorio*

TRATTAMENTO FILE

ln	<vecchionome> <nuovonome>	*link*
cp	<filesorgente> <filedestinazione>	*copia*
mv	<vecchionome> <nuovonome>	*rinom. / spost.*
rm	<nomefile>	*cancellazione*
cat	<nomefile>	*visualizzazione*

Esempi di comandi:

cd /nome2/nome6
cat file*
ls /nome1
rm *

PROTEZIONE e DIRITTI SUI FILE

Per variare i bit di protezione:

chmod [**u g o**] [**+ -**] [**rwx**] <nomefile>

I permessi possono essere concessi o negati dal
proprietario del file

Esempio di variazione dei bit di protezione:

chmod 0755 /usr/dir/file

0	0	0	|	1	1	1	|	1	0	1	| 1	0	1
SUID	SGID	sticky	|	R	W	X	|	R	W	X	| R	W	X
				User				**G**roup				**O**thers		

Ad esempio:
chmod u-w fileimportante

Altri comandi:
chown <nomeutente> <nomefile>
chgrp <nomegruppo> <nomefile>

Esempio, il comando ls per listare il contenuto di una directory

Varie opzioni:

ls -a [<nomedir>]
visualizza file "nascosti" (nome che inizia con '**.**')

ls -l [<nomedir>]
mostra tutte le informazioni per i file (tipo del file, permessi, numero link, proprietario...)

ls -la [<nomedir>]
è l'unione delle precedenti

ls -F
tutte le informazioni dei file visualizzando i file normali con il loro nome, gli eseguibili con nome e suffisso *****, i direttori con nome e suffisso ***/***

ls -d
i direttori sono visualizzati come i file, senza entrare nel contenuto e listarlo

ls -R <nomedir>
esame ricorsivo della gerarchia a partire da nomedir

ls -l [<nomedir>]
tutte le informazioni per i file, permessi, tipo del file, numero link, proprietario, ora modifica, nome

ls -i
lista gli i-number dei file con le altre informazioni

ls -r
lista i file in ordine opposto al normale ordine alfabetico

ls -t
lista i file in ordine di ultima modifica, dai più recenti, fino ai meno recenti

ALTRI COMANDI DI SISTEMA

- I *file* sono considerati insiemi di *linee*, fatte da *parole*
- Le *parole* sono sequenze di *caratteri*, separate da spazi

more <nomefile> *una pagina per volta*
sort <nomefile> *ordina alfabeticamente*
sort [**-r**] <nomefile> *(r = in ordine inverso)*
Molte opzioni:
- uscita su file opzione **-o** <nomefileout>

diff <file1> <file2> *mostra solo le righe diverse*
find <direttorio> **-name** <nomefile>
cerca nomefile *in* direttorio
wc [**-lwc**] [<nomefile>]
*conta le linee (opzione **l**), o le parole (opzione **w**), o i caratteri (opzione **c**) dello standard input o del file*

COMANDI di STATO del sistema

date *data e ora attuale*
time *cronometra l'esecuzione di un comando*
who *mostra gli utenti attualmente collegati*
ps *mostra i processi correnti nel sistema*
sleep *sospende il processo quando specificato*
man <comando> *mostra l'help (guida) del comando*

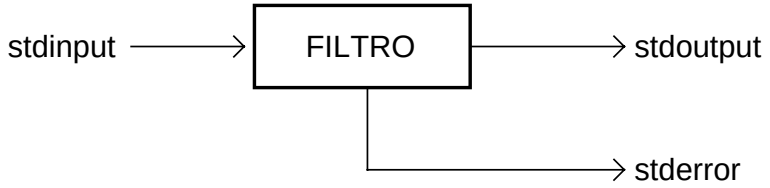
ESEMPIO: date; time; who; ps

RIDIREZIONE DELL'I/O

Tutti i comandi di UNIX sono **filtri**

Un **filtro** è un programma che riceve in ingresso da un input e produce risultati su output (uno o più)

standard input ➔ console
standard output ➔ console
standard error ➔ console



I filtri possono operare sull'input considerandolo a linee

more *stampa una pagina per volta*
sort *ordina alfabeticamente le linee*
grep <stringa> *cerca la* stringa *nel file*
rev *ribalta ogni singola linea*
tee <file> *copia l'input sia sull'output sia sul* file
head [-numerolinee] *filtra le prime linee del file*
tail [-numerolinee] *filtra le ultime linee del file*
awk *trasforma le stringhe di un file secondo certe regole*

- Ogni comando può essere ridiretto su un file diverso **senza cambiare il comando**
- **completa OMOGENEITÀ fra dispositivi e files**

ridirezione dell'input

<comando> **<** <fileinput>

ridirezione output

<comando> **>** <fileoutput> *riscrive il fileoutput*

<comando> **>>** <fileoutput> *appende a fileoutput*

Alcuni esempi di ridirezione:

ls -lga > file

ls > /tmp/file
cd /tmp
sort < file > filetemp
rev < filetemp > file
more < filetemp file
rm file filetemp

who > temp
wc -l temp

ps > file

RIDIREZIONE (estensioni)

Ogni comando trova aperti

stdin, stdout, stderr

RIDIREZIONE MULTIPLA

In caso di ridirezione, il file di ingresso è aperto in lettura, il file di output aperto in scrittura ***(cioè creato vuoto)***

comando **>** file1 **<** file2 **>** file3 **<** file 4 **>** file5
Il comando esegue con stdin da file4 e stdout su file5

EFFETTI COLLATERALI distruzione dei file file1 e file3

ALTRI FILE

In Bourne Shell è possibile anche:
- usare altri file in ridirezione, specificandoli alla invocazione del comando
- agganciare più output ad uno stesso file

comando **2>** file2 **3>** file3 **5>** file5
Si richiede la apertura del file2 con chiave 2, del file 3 con chiave 3, etc.

comando &> 2
L'uscita del comando viene forzata sullo std error

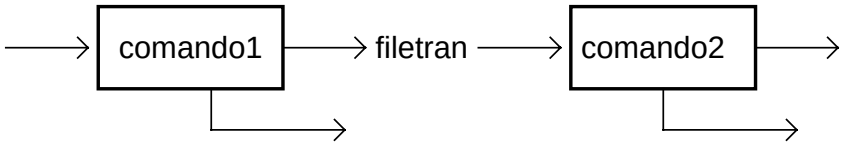
Si noti che la numerazione prosegue dai file standard:
stdin = 0, stdout = 1, stderr = 2, ...

PIPE

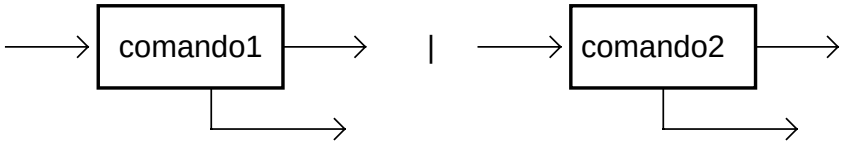
COLLEGAMENTO AUTOMATICO di comandi

La PIPE *(tubo)* **collega**
l'***output*** di un comando con l'***input*** del successivo

SCHEMA IMPLEMENTATIVO



<comando1> > <filetran> ; <comando2> < <filetran>



<comando1> **|** <comando2>

In **UNIX** ➔ PIPE come **COSTRUTTO PARALLELO**

le pipe sono svolte in modo parallelo
(ogni comando è mappato in un processo)

I comandi della pipe procedono in parallelo tra loro
➔ NON si creano file temporanei
Mentre un comando produce l'output, l'altro lo consuma

Esempi di piping:

who | wc -l *conta gli utenti collegati*

ls -R | more

rev < file1 | sort | rev | more