Unix: un s.o. orientato allo sviluppo software

Caratteristiche principali di Unix:

- concepito apposta per lo sviluppo software
- multiutente e interattivo
- portabile su diverse architetture hardware
- standardizzato (portabilità), ma personalizzabile

Visione utente
UNIX= shell + comandi

Visione programmatore
UNIX= chiamate di sistema + strumenti di programmazione C

Visione sistemista
Unix = nucleo + servizi e privilegi

Alcuni punti chiave di Unix:

Unix = processi (multitasking) + file system condiviso (multiutente)

Interfaccia utente uniforme (shell)

Processi dinamici e comunicanti

File system gerarchico con sistema di diritti di accesso

I dispositivi I/O sono file speciali

Le funzioni di sistema sono una libreria del linguaggio C

Aspetti non legati a Unix, ma oggi tipici:

Gestione a finestre con X Windows
Gestione servizi di rete su TCP/IP

Storia di Unix

1969	Unix originale	Laboratori Bell
	Versioni Preliminari	
1972	Unix versione 4	
1975	Unix versione 6	
1977	Unix versione 7	AT&T

NASCITA DI UNIX™ e STANDARDIZZAZIONE

1978	XENIX MICROSOFT		UNIX AT&TBERKELEY
UNIX			
1980		System III	4.0 BSD
1981	Release 3		4.1 BSD
1984	Release 5	System V	4.2 BSD
1985			4.3 BSD

1990 UNIX System V vers. 4
1992 Standard Posix

Altre versioni famose:
ULTRIX (Digital Equipment), AIX (IBM), HPUX (Hewlett-Packard)

1987,1997 MINIX (Andy Tanenbaum)
1992 Linux (Linus Torvalds)

Una sessione con Unix

```
LOGIN: root
PASSWORD: ********
    (messaggio del giorno)

$> ls
FILE1  FILE2  FILE3  POSTA

$> ls -la
-rwxr--r-- root    .login 13.00  24-5
-rwxrwxr-- root    FILE1  11.22  12-4
-rwxr--rwx andrea  FILE2  13.00  04-1
-rwxr----x root    FILE3  13.00  22-5
drwxr--r-x root    POSTA  13.00  21-3

$> cd POSTA ; ls
LETTERA1

$> cd ..
$> cc FILE2 > PROG2
compilation ok

$> ls
FILE1  FILE2  FILE3  POSTA  PROG2

$> PROG2 < FILE1 &
$> ps
1234 PROG2
1236 PS
PROG2 terminated

$> logout
```

Alcuni comandi di Unix

La shell può attivare comandi, che possono essere interni o esterni
I comandi hanno zero o più argomenti, e possono essere modificati da opzioni (caratteri preceduti dal simbolo "-")

• esistono più shell (con diversi comandi **interni**)

• i comandi **esterni** attivano processi

• i comandi esterni sono memorizzati in file eseguibili (spesso contenuti in `/bin` oppure `/local/bin` o `/usr/bin`)

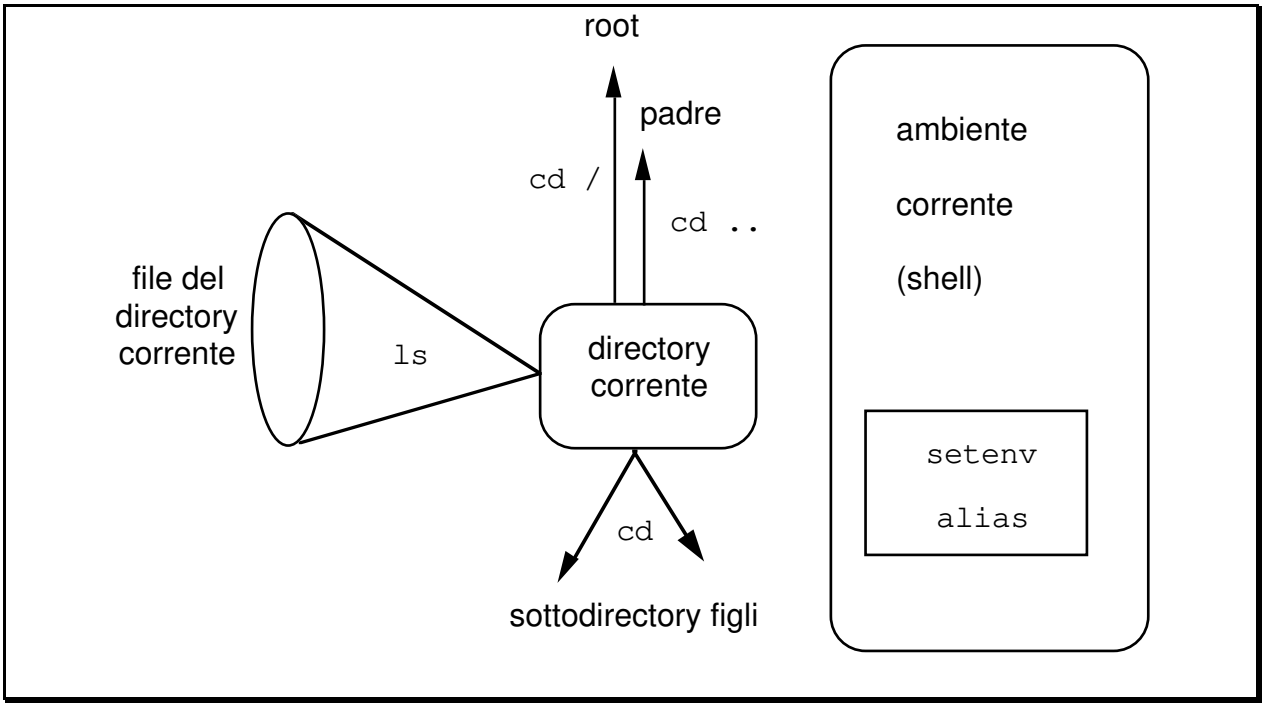
Comandi di introduzione

`login` entra nella shell dell'utente (richiede la password)
`logout` disattiva il terminale (alternativa: `exit`)
`date` stampa data e ora
`passwd` cambia la password dell'utente (richiede la vecchia)
`who` mostra gli utenti collegati
`whoami` stampa il nome dell'utente collegato
`learn` corso interattivo di istruzione

Comandi di uso generale

`cd` *nome_dir*	cambia il directory attivo
`ls` *nome_dir*	elenca il contenuto del directory
`mkdir` *nome*	crea un directory
`ps`	mostra i processi attivi
`pwd`	stampa il nome assoluto del directory corrente
`rm` *file*	cancella un file
`mv` *file1 dir*	trasferisce un file
`cp` *file1 file2*	copia un file
`chmod` *modi file*	modifica i diritti di accesso di un file
`more` *file*	visualizza un file
`rmdir` *dir*	cancella un directory (solo se è vuoto)
`cat` *file1*	direziona l'argomento sullo standard output
`man` *comando*	richiama il manuale in linea

Piccola sinossi dell'interazione con Unix



L'utente può creare o accedere directory e file

L'utente accede il file system via la shell, che è un processo che rappresenta l'utente all'interno del sistema

Fondamentalmente l'utente può fare due cose:

- attivare o uccidere processi (foreground, background)
- creare, leggere, scrivere, chiudere, cancellare file

I servizi principali

Manipolazione di testi
- editor `vi, emacs`
- linguaggi di manipolazione: `sed`, `perl`

L'ambiente C
- Il compilatore (il linker, l'ottimizzatore)
- Il debugger simbolico
- `make`, SCCS

Formattazione di testi
- `roff, nroff, troff`
- `TeX`

Archiviazione e codificazione
- `tar, shar, compress, uucode`

Servizi di rete
- `mail (elm,pine)`
- `finger`
- `telnet`
- `talk (IRC)`
- `readnews`
- `ftp`
- `www (mosaic,netscape)`

Il manuale di Unix

Esiste un formato standard del manuale
(che è in linea: comando `man`)

Le parentesi quadre [] indicano entità opzionali
I puntini (...) indicano che l'elemento può essere più volte
I qualificatori in neretto vanno usati esattamente come sono scritti

Le parole ed i simboli stampati normalmente sono argomenti e devono essere sostituiti da nomi di file, argomenti numerici ecc.

Comando: `ls` – mostra il contenuto di un directory
Sintassi: `ls [-adfgilrst]` *`nome`*
Qualificatori:

	–a	Mostra tutti gli elementi, anche . e ..
	–d	Non mostra il contenuto dei directory
	–f	Mostra gli argomenti come directory non ordinati
	–g	Restituisce l'id del gruppo
	–i	Stampa il numero di i-nodo nella prima colonna
	–l	Dettagliato: modo, rif., propr., dimens. e data
	–r	Inverte l'ordine
	–s	Dà la dimensione in blocchi (compresi gli indiretti)
	–t	Ordina in base alla data, l'ultimo per primo

Esempi:
```
ls -l     # Mostra i file del directory di lavoro
ls -lis   # Come sopra, con i-nodi e dimensioni
```

Fa un elenco per ogni file argomento. Per ogni directory argomento, mostra il suo contenuto, a meno che non sia specificato -d. Quando non vi sono argomenti, mostra il contenuto del directory di lavoro.

Caratteristiche principali di Unix

Classi di utenti

Utenti normali (ripartiti in gruppi con intersezioni non vuote)
Superutente (username root, oppure utenti con privilegi)

File system gerarchico

Directory, file normali, file speciali
Protezione a 3 livelli: *proprietario*, *gruppo*, *tutti*

Processi

Creazione dinamica di processi asincroni (*background*)
Comunicazione tra processi via il file system

I/O uniforme

I dispositivi sono file (speciali)
Stessi meccanismi di comunicazione come per i processi

Interfaccia utente: la shell

Linguaggio di comandi
Personalizzabile

Classi di utenti

User
Ogni utente ha il diritto di utilizzare solo alcune risorse;
ogni utente ha il pieno controllo sulle sole risorse che possiede

Ad ogni utente è associato un nome (*username*) ed una *password*; ogni utente fa parte di almeno un *gruppo* di utenti.

Ogni utente possiede un directory nel quale può conservare i suoi file (eventualmente organizzati su altri sottodirectory)

Ogni utente può usare un certo insieme di comandi

Superuser
È il nome col quale in gergo Unix si indica il system manager

Può controllare ed usare tutte le risorse del sistema (ha massimo privilegio su ogni componente); può scavalcare qualsiasi protezione; può impersonare qualsiasi utente (ma non può conoscerne la passwd)

Ha come directory home la radice del file system

Può usare qualunque comando

Può spegnere il sistema in qualsiasi momento

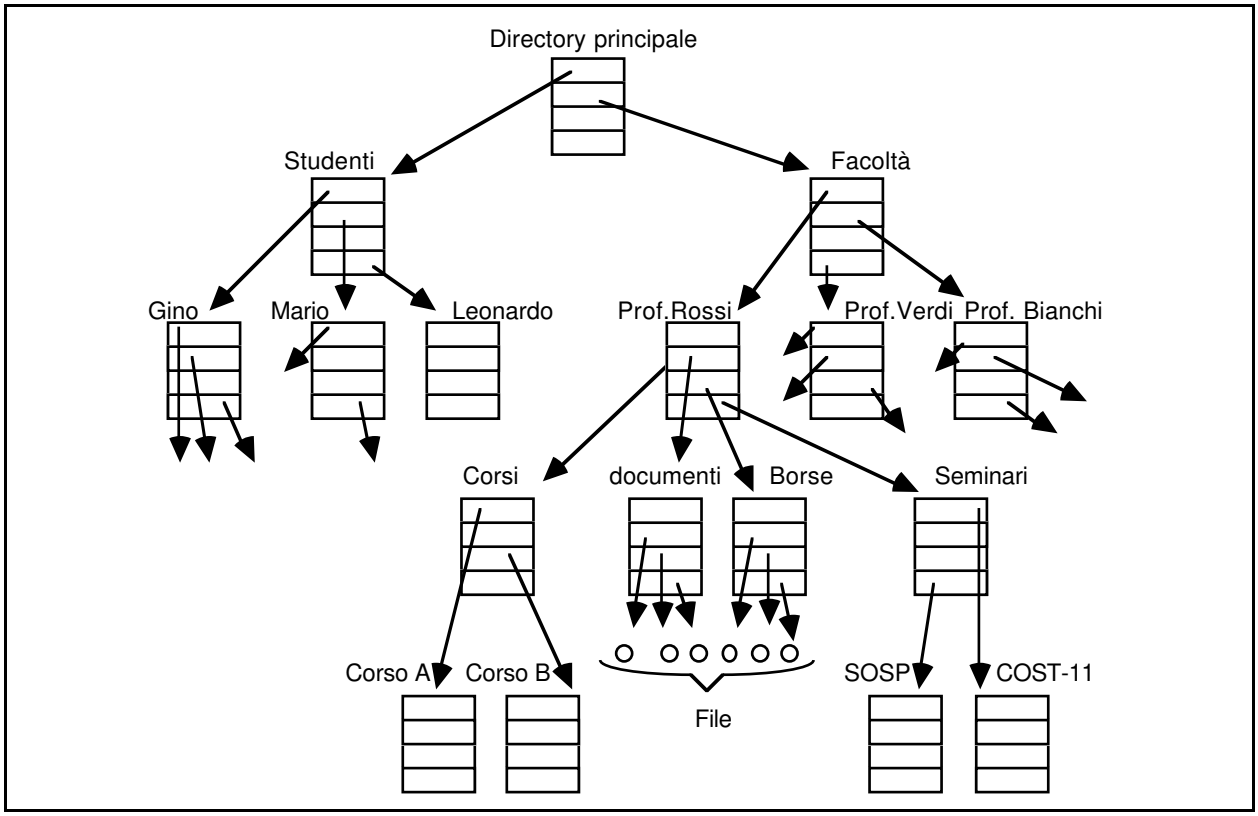
Il file system

Intuitivamente, un *file* è uno "spazio di archiviazione" con nome, in cui si memorizzano caratteri che rappresentano dati oppure programmi

L'insieme dei file, detto *file system*, ha una struttura ad albero (gerarchica); i nodi dell'albero si chiamano *directory*

Unix ammette 3 tipi di file:

Normale (sequenza di byte): una foglia dell'albero
Directory (insieme di file): un nodo interno dell'albero
File speciale: rappresenta un dispositivo di I/O



Nomi dei file

Nome *assoluto* di un file:
cammino necessario a raggiungere il file partendo dalla radice

`/`	nome della radice (root)
`/a`	nome di secondo livello (contenuto nella radice)
`/a/dev`	nome di terzo livello

Nome *relativo* di un file:
cammino necessario a raggiungere il file partendo dal directory corrente

Nota Bene 1:
In ogni momento la shell "vede" un solo directory (detto **corrente**); sono visibili solo i nomi in esso contenuti al livello successivo

Nota Bene 2:
Ogni directory (anche uno vuoto) contiene due nomi privilegiati:

`.`	nome del directory corrente
`..`	nome del directory genitore

Nota Bene 3:
Ogni directory è un file esso stesso

Struttura logica del file system

Il file system di un s.o. sotto Unix è organizzato più o meno sempre nello stesso modo

Il directory radice, detto `root`, contiene i seguenti sottodirectory:

/bin – contiene i più importanti programmi (eseguibili)
/dev – contiene i file speciali
/etc – contiene vari file e programmi di gestione del sistema
/lib – contiene le librerie
/tmp – usato per i file temporanei, sempre aperto a tutti
/user – il file system d'utente viene montato qui
/usr – altri dischi di sistema vengono montati qui

In */bin* troviamo i programmi più usati: *cat*, *cp* e *ls*, così come alcuni programmi, come *login* e *sh*, necessari per l'uso del sistema

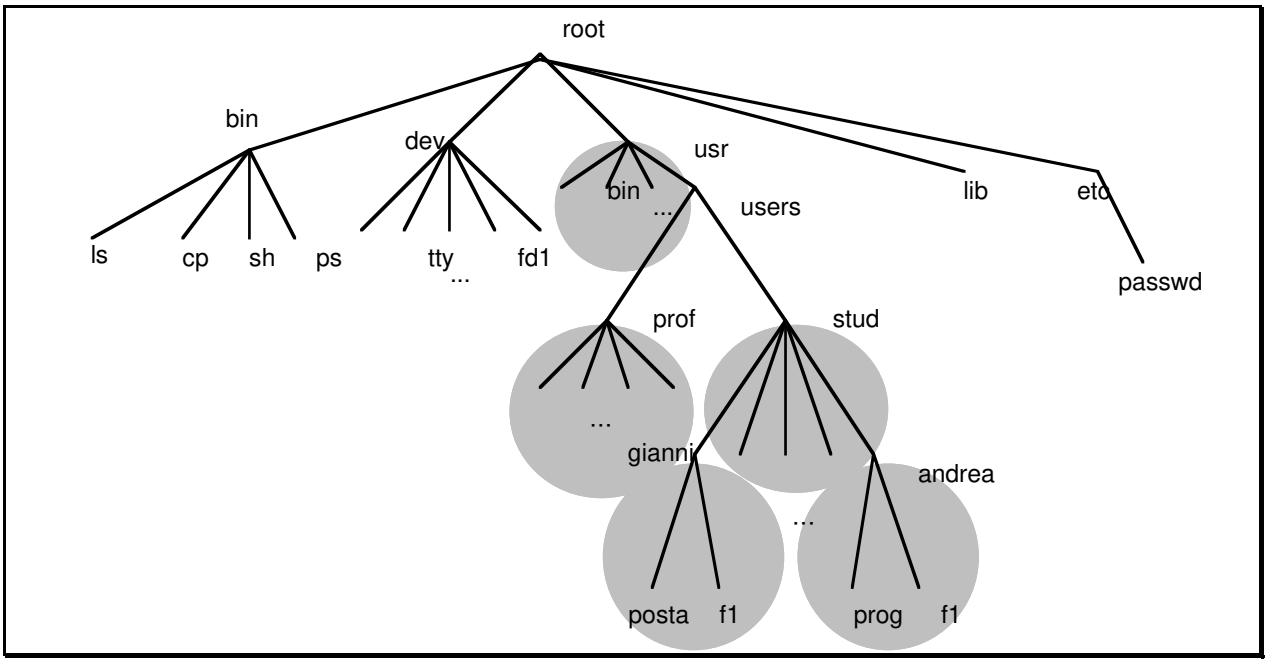
Il directory */dev* contiene i file speciali per i dispositivi di I/O:

/dev/ram – il disco RAM
/dev/mem – la memoria assoluta
/dev/kmem – la memoria del nucleo
/dev/null – il dispositivo nullo
/dev/fd0 – il dispositivo di dischetto 0
/dev/fd1 – il dispositivo di dischetto 1
/dev/hd0 – il dispositivo minore del disco rigido 0
/dev/hd1 – il dispositivo minore del disco rigido 1
/dev/lp – la stampante
/dev/tty – il terminale corrente
/dev/tty0 – il terminale 0 (console)

Struttura fisica del file system

Il comando df permette di vedere la struttura fisica del file sistem, memorizzato su un insieme di supporti (dischi)

```
Filesystem          kbytes    used   avail capacity
Mounted on
/dev/sd1a            14707   12445     791     94%
/
/dev/sd1g           453550  389932   18263     96%
/usr
/usr/spool/mail    1319319  479682  707705     40%
/var/spool/mail
gladio:/home/gladio 580734  296054  226606     57%
/home/gladio
gladio:/share       580734  506844   15816     97%
/share
gladio:/projects    580734  296054  226606     57%
/projects
gea:/home/gea       179529  125894   35682     78%
/home/gea
gea:/usr/projects   179529   79730   81846     49%
/home/gea/projects
apollo:/home/apollo 1319319  479682  707705     40%
/home/apollo
olimpo:/home/olimpo 197919  157592   20535     88%
/home/olimpo
olimpo:/usr/olimpo  388998  312445   37653     89%
/usr/olimpo
```


root
bin
dev
usr
lib
etc
ls
cp
sh
ps
tty
fd1
...
bin
...
users
passwd
prof
stud
...
gianni
andrea
...
posta
f1
prog
f1

Meccanismi di protezione dei file

Ad ogni file è associata una stringa che determina i diritti d'accesso (tale stringa fa parte del *descrittore* del file)

```
d     r    w     x     r    w    x     r    w    x
      proprietario    gruppo          altri
```

primo carattere: tipo del file (directory, speciale, normale, ecc.)

9 bit: tre classi (proprietario, gruppo del proprietario, altri utenti)

	file normale	directory
r	lettura	lettura del contenuto con ls
w	scrittura	inserimento e cancellazione di file
x	esecuzione	accesso con `cd`

proprietario	(u)	Chi ha creato il file
gruppo	(g)	Gruppi cui appartiene il prorietario
altri	(o)	Tutti gli altri utenti

Esempio:

```
$> ls -la
-rwxr--r-- root    .login 13.00  24-5
-rwxrwxr-- root    FILE1  11.22  12-4
-rwxr--rwx andrea  FILE2  13.00  04-1
-rwxr----x root    FILE3  13.00  22-5
drwxr--r-x root    POSTA  13.00  21-3
```

I diritti possono essere cambiati solo dal proprietario o dal super-utente (col comando `chmod`)

Esempio: `chmod go-rwx POSTA`
Esempio: `chmod 644 notizie`

File speciali

I file speciali rappresentano *dispositivi*, oppure
funzionano come *canali* di comunicazione tra processi (*pipe*, *sockets*)

Dispositivi

- il terminale console (tastiera + video)
- i dischi (e loro partizioni)
- il disco virtuale in RAM
- dispositivi di memoria in genere (nastri, altri tipi di dischi, ecc.)
- le stampanti
- altri terminali (tastiere+video)
- le porte di comunicazione (con la rete, con il modem)
- la memoria centrale

Ogni processo ha "sin dalla nascita"
un file di input, uno di output e uno di diagnostic

Standard Input	(di solito la tastiera)
Standard Output	(di solito il video)
Standard Diagnostic	(come lo standard output)

Ingresso e uscita possono essere rediretti

```
mail PAOLO < MESSAGGIO    (redirezione input)

ls -L > ELENCO            (redirezione output)

prog < DATI > RISULTATI   (redirezione di input e output)
```

Comunicazione tra processi

Il meccanismo principale per far collaborare più processi è il **pipe:**

Esempio1:

```
who | wc -l
```

`who` elenca gli utenti collegati
`wc` conta le parole (-l: opzione per contare le righe)

Esempio2:

```
ls -l /tmp | grep anna | sort +3nr | lpr
```

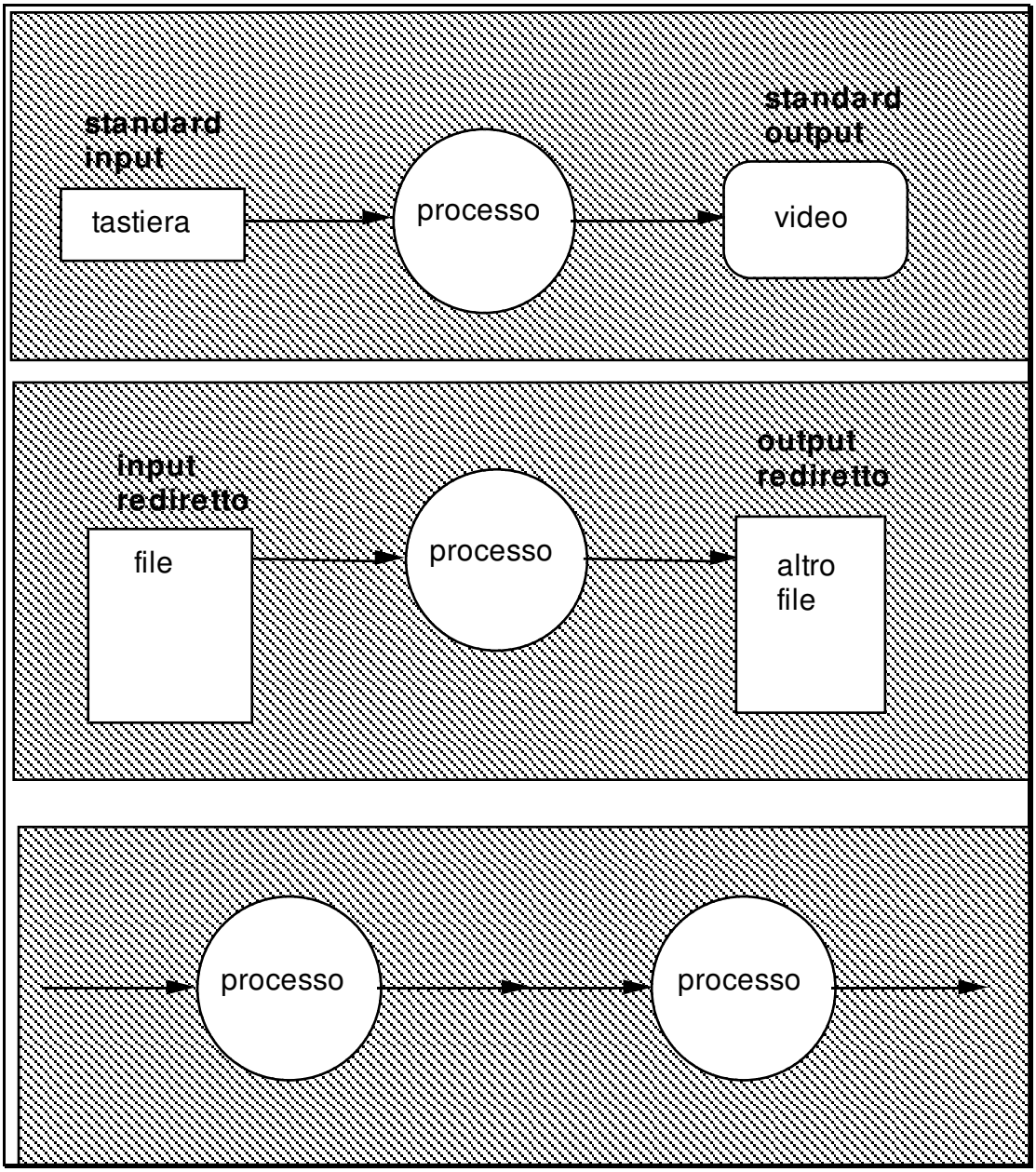
`ls` elenca il contenuto di un directory
`grep s` seleziona tutte le righe che contengono la stringa *s*
`sort` ordina un file (+3nr: in ordine inverso sul terzo campo)
`lpr` processo di stampa

La linea precedente equivale alla seguente sequenza di comandi:

```
ls -l /tmp > temp1
grep anna temp1 > temp2
sort +3nr temp2 > temp3
lpr temp3
rm temp [1 2 3]
```

Nota:
un pipe è un *file temporaneo* che un processo produttore scrive ed un processo consumatore legge

Comunicazione tra processi



Processi

Processo (visione dall'esterno)
Entità che esegue un programma: i processi hanno una "vita"
(sono degli "attori" che eseguono sequenze di istruzioni:
possono essere creati, fermati, uccisi a nostro piacimento)

Immagine (visione dall'interno)
Ambiente associato ad una elaborazione di un programma
(memoria + registri + file aperti + directory corrente)

I processi sono identificati da un identificatore (PID) unico entro il sistema (afssegnato dal s.o. al momento della macchina)

Ogni utente ha almeno un processo di **foreground** (di solito la shell); può attivare più processi di **background** (es: stampe, compilazioni, demoni, ecc.)

Comandi che agiscono su processi

`ps -xa` osserva i processi (ps = process status)

`kill` *`pid`* uccide il processo identificato da *pid*

Osservare i processi

Il comando `ps -ax` permette di vedere tutti i processi correntemente in esecuzione

```
  PID  TT  STAT      TIME COMMAND
    0  ?   D      0:26 swapper
    1  ?   S      0:18 init
    2  ?   D      0:00 pagedaemon
   92  ?   I      0:00 /usr/lib/sendmail -bd -om
   94  ?   I      0:02 dnet_spawner
  133  ?   S      0:05 /etc/cron
  261  co  I      0:00 -csh (csh)
17458  ?   S      0:00 -sh (csh)
17889  ?   I      0:00 mail11
17890  ?  I 0:00 -AA17890 di.unipi.it:RCPT To:<lov>
18126  ?   I      0:00 -sh (csh)
18128  ?   I      0:00 dlogin cnuce
18384  ?    D     0:00 send-mail -i -m aa@cs.yale.edu
  161  04  I      0:00 - std.4800 tty04 (getty)
18115  05  I      0:02 -csh (csh)
18349  05  S      0:02 /usr/users/stud/verdi/jove
17363  09  I      0:00 -csh (csh)
17634  d1  I      0:00 - D1200 ttyd1 (getty)
  167  11  I      0:00 - std.4800 tty11 (getty)
17247  12  I      0:00 - std.9600 tty12 (getty)
15242  14  I      0:02 -csh (csh)
16628  14  S      0:08 rlogin expert
  173  15  I      0:00 - std.4800 tty15 (getty)
  174  16  I      0:00 - std.4800 tty16 (getty)
```

La shell

La shell permette all'utente di accedere al sistema
(e lo rappresenta)

Nello stesso sistema possono convivere shell diverse
Es.: Bourne shell, C shell, Korn shell, tcsh
Le differenze tra le varie shell riguardano la sintassi e alcuni comandi che sono presenti in alcune e non in altre

La shell accetta comandi da tastiera oppure da file script

Esempio: (comando da tastiera)
```
$ ls -la
```

Esempio: (esecuzione di uno script)
```
$ sh script
```

Uno *script* (sinonimi: profile, programma di shell, file batch) è un file che contiene una sequenza di comandi di shell

Negli script si possono usare:

- Variabili (sia predefinite che d'utente)
- Strutture di controllo
- Sottoprogrammi
- Passaggio di Parametri
- Gestione di interruzioni

Tipici programmi di shell:
- `.login` (oppure `.profile`)
- `.logout`

Linguaggio di shell

Il linguaggio di Bourne shell permette la redirezione dell'I/O, i caratteri jolly, il lancio di processi in background, l'uso di procedure di comandi e molte altre caratteristiche

```
date                        # Comando regolare
sort < file                 # File è l'ingresso
sort < file1 > file2        # File1 è l'ingresso e file2 l'uscita
cc file.c 2 > error # standard error viene memorizzato in error
a.out > f 2 > &1   # Combina standard output e standard error
sort < file1 >> file2       # Appende l'uscita a file2
sort < file1 > file2 &      # Elaborazione in background
(ls -l; a.out) &   # Esegue 2 comandi in bg sequenzialmente
sort < file | wc            # Due processi in pipe
sort < f | uniq | wc        # Tre processi in pipe
ls -l *.c                   # Elenca i file che terminano con .c
ls -l [a-c]*                # Elenca i file che iniziano con a, b o c
ls -l ?           # Elenca i file il cui nome è lungo un carattere
ls \?          # Elenca i file il cui nome è un punto interrogativo
v = /usr/ast                # Imposta la variabile shell v
ls -l $v                    # Uso della variabile shell v
PS1 = 'ciao!'               # Cambia il prompt primario a ciao!
ls -l $HOME                 # Visualizza il directory principale
echo $PATH                  # Stampa il path di ricerca corrente
if … then … else … fi             # Istruzione condizionale
for … do … done             # Iterazione con eneumerazione
while … do … done           # Iterazione con condizione
```

Uso della “punteggiatura” di shell

/ separa i nodi di un cammino
(da solo, denota la radice del file system)

```
cd /usr/bin/mail
```

; separa più comandi sulla stessa linea

```
cd ..; ls -la
```

> redirezione dell’output

< redirezione dell’input

```
prog < dati > risultati
```

& il programma gira in background

```
prog < dati &
```

| pipe (comunicazione tra processi)

```
prog < dati | lpr
```

? jolly di carattere entro un identificatore

```
rm -i prog?
```

* jolly di stringa entro un identificatore

```
rm -i p*
```

- precede le opzioni che modificano un comando

Variabili di shell predefinite

L'ambiente di shell è un insieme di coppie <NOME = VALORE>;
tali coppie rappresentano le cosiddette *variabili predefinite*.

Tale ambiente definisce un insieme di parametri di elaborazione
che vengono ereditati dai processi figli
(cioè dai comandi che vengono invocati dalla shell)

`HOME`	cammino assoluto della *home directory* dell'utente
`PATH`	directory in cui cercare un comando non trovato in current
`CDPATH`	usato dal comando `cd` per cercare directory
`PS1`	*prompt* primario
`PS2`	*prompt* secondario
`LOGNAME`	*nome di login* dell'utente
`MAIL`	nome del directory utilizzato per la posta elettronica
`SHELL`	nome assoluto del programma di shell di default
`TERM`	tipo di terminale utilizzato
`PRINTER`	stampante di default

Esempio: un file .login

Il file `.login` contiene le definizioni preferite dall'utente, da attivare al moneto di inizializzazione della shell

```
set path=(. /usr/ucb /bin
       /usr/bin /usr/local/bin /usr/games)
noglob notify noclobber ignoreeof
mail=(/usr/spool/mail/user)
set prompt = VAX:
setenv PRINTER "lp1"
setenv TERM "vt100"
setenv BIM_PROLOG_DIR
/usr/local/bin/Bprolog/Bprolog
stty newcrt erase
alias rm rm -i
alias mv mv -i
alias cp cp -i
alias ls ls -F
alias cd 'cd \!:1;pwd;ls'
biff y
alias rec 'cd posta/Receive; mail'
unset noglob
msgs -f
set history=10
alias cd 'set old=$cwd; chdir \!*'
alias hi history
```

Esempio: uno script di shell

Dato un file chiamato `file1` contenente stringhe ordinate alfabeticamente, aggiornarlo copiando le sue stringhe in un altro file chiamato `file2`

Soluzione sbagliata: `cat file1 file2 > file1`

Soluzione corretta:

```
cat file1 file2 > file3
mv file1 oldfile1
sort file3 > file1
rm file3
```

Mediante un editor, scriviamo questi comandi in un file `aggiorna`
Per renderlo eseguibile (script) usiamo il comando `chmod`

```
chmod 755 aggiorna
```

Per eseguire lo script `aggiorna` usiamo il comando `csh:`

```
csh aggiorna
```

Rendiamo il programma di uso più generale mediante variabili:

```
cat $1 $2 > temp1
mv $1 old
sort temp1 > $1
rm temp1
```

Invocazione: `aggiorna f1 f2`