



*Corso di Laurea Triennale in Informatica
Università Degli Studi della Basilicata*

Sistemi Operativi

*Docente:
ugo.erra@unibas.it*

14° Lezione

Introduzione a Linux

VMWare Player 2.5.1

The screenshot shows the VMware Player 2.5.1 product page. The page features a navigation bar with links for Communities, Virtual Appliances, Store, and Support. Below the navigation bar, there are links for Solutions, Products, Technology, Services, Resources, Customers, Partners, and About Us. The main content area includes a 'Download' button, a description of the product, and a list of features. The right sidebar contains sections for Desktop Products, Related Products, Next Steps, and Player Resources.

vmware® Communities Virtual Appliances Store Support Worldwide Search

Solutions | Products | Technology | Services | Resources | Customers | Partners | About Us Downloads | Account | Contact Us

Home > Products > Desktop Products > VMware Player

VMware Player

Run Virtual Machines on Linux or Windows PCs for Free

[Download](#)

Run virtual machines on your Windows or Linux PC with VMware Player 2.5. This free desktop virtualization software application makes it easy to operate any virtual machine created by VMware Workstation, VMware Fusion, VMware Server or VMware ESX, as well as Microsoft Virtual Server virtual machines or Microsoft Virtual PC virtual machines. You can also use Player to evaluate one of the many virtual appliances available from the [VMware Virtual Appliance Marketplace](#).

- Run multiple operating systems simultaneously on a single PC.
- Experience the benefits of preconfigured products without any installation or configuration hassles
- Share data between host computer and virtual machine

Questions? 1-877-486-9273

- Email Sales
- Join Discussion

[Desktop Products](#)

- [VMware Player](#)

[Related Products](#)

- [VMware Workstation](#)

[Next Steps](#)

- [Download Player](#)
- [Download Browser Appliance](#)
- [Download Virtual Appliances](#)

[Player Resources](#)

- [Product Info](#)
- [Documentation](#)
- [Knowledge Base](#)
- [Community](#)
- [Self-Help Support](#)
- [Icon Usage & Guidelines](#)

[Overview](#) [Features](#) [FAQs](#)

Run multiple operating systems simultaneously on a single PC

Get more out of your existing computer hardware. Use the [free](#) VMware Player application to run multiple operating systems simultaneously on a single PC.

With VMware Player, you can use any virtual machine created by VMware Workstation, VMware Fusion, VMware Server or VMware ESX, as well as Microsoft Virtual Server virtual machines and Microsoft Virtual PC virtual machines. Import third party images including Symantec Backup Exec System Recovery (formerly called Live State Recovery) images, Norton Ghost 10 images, Norton Save & Restore images, StorageCraft ShadowProtect images, and Acronis True Image images to VMware Player compatible virtual machines. Use 32- and 64-bit Windows, Linux, NetWare, or

Trova: peter

Successivo Precedente Evidenzia Maiuscole/minuscole

<http://www.vmware.com/products/player/>

Prima di Linux



- Prima degli anni 80...
 - ▣ Microsoft DOS era il Sistema Operativo dominante
 - ▣ Apple MAC era un'ottima alternativa ma era costoso
 - ▣ UNIX era il migliore ma ancora più costoso
- Avere un sistema operativo basato su UNIX per PC non era possibile in quanto i sorgenti erano proprietari
 - ▣ Le modifiche al codice erano possibili solo pagando i costi di licenza

GNU Linux

- Nel 1984 Richard Stallman definì il concetto di software libero (free) attraverso quattro libertà
 - ▣ Libertà di eseguire il programma per qualsiasi scopo (chiamata "libertà 0")
 - ▣ Libertà di studiare il programma e modificarlo ("libertà 1")
 - ▣ Libertà di copiare il programma in modo da aiutare il prossimo ("libertà 2")
 - ▣ Libertà di migliorare il programma e di distribuirne pubblicamente i miglioramenti, in modo tale che tutta la comunità ne tragga beneficio ("libertà 3")



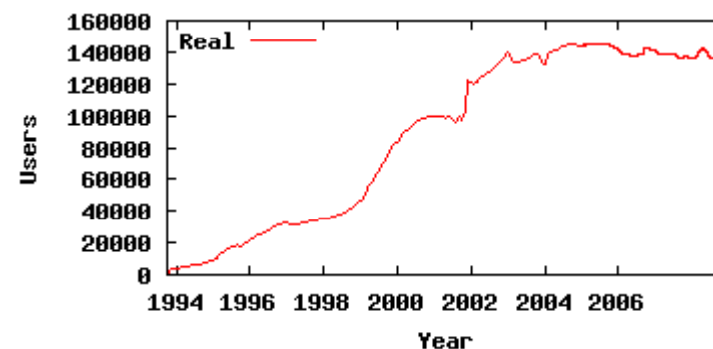
Linux

- Minix era una versione semplificata di Linux sviluppato per PC da Andrew Tanenbaum
 - ▣ Sviluppato per scopi didattici e non commerciali
- Nel 1991, Linus Torvalds, uno studente di informatica al secondo anno sviluppò una versione preliminare del kernel di Linux (versione 0.0.1)
- Immediatamente centinaia di persone scaricarono ed iniziarono a fornire contributi per la crescita di Linux
- Linux fu rilasciato con licenza GNU in modo che chiunque potesse copiare, studiare e cambiare il codice



Chi usa Linux oggi?

- Linux oggi è un sistema operativo multiplatforma
 - ▣ PC, PDA, Supercomputer,...
- L'utilizzo di una GUI ne ha favorito ulteriormente la diffusione



Alle gen 12 2009 11:02:54 GMT, ci

sono

129418

utenti registrati

145498

macchine registrate

Le distribuzioni

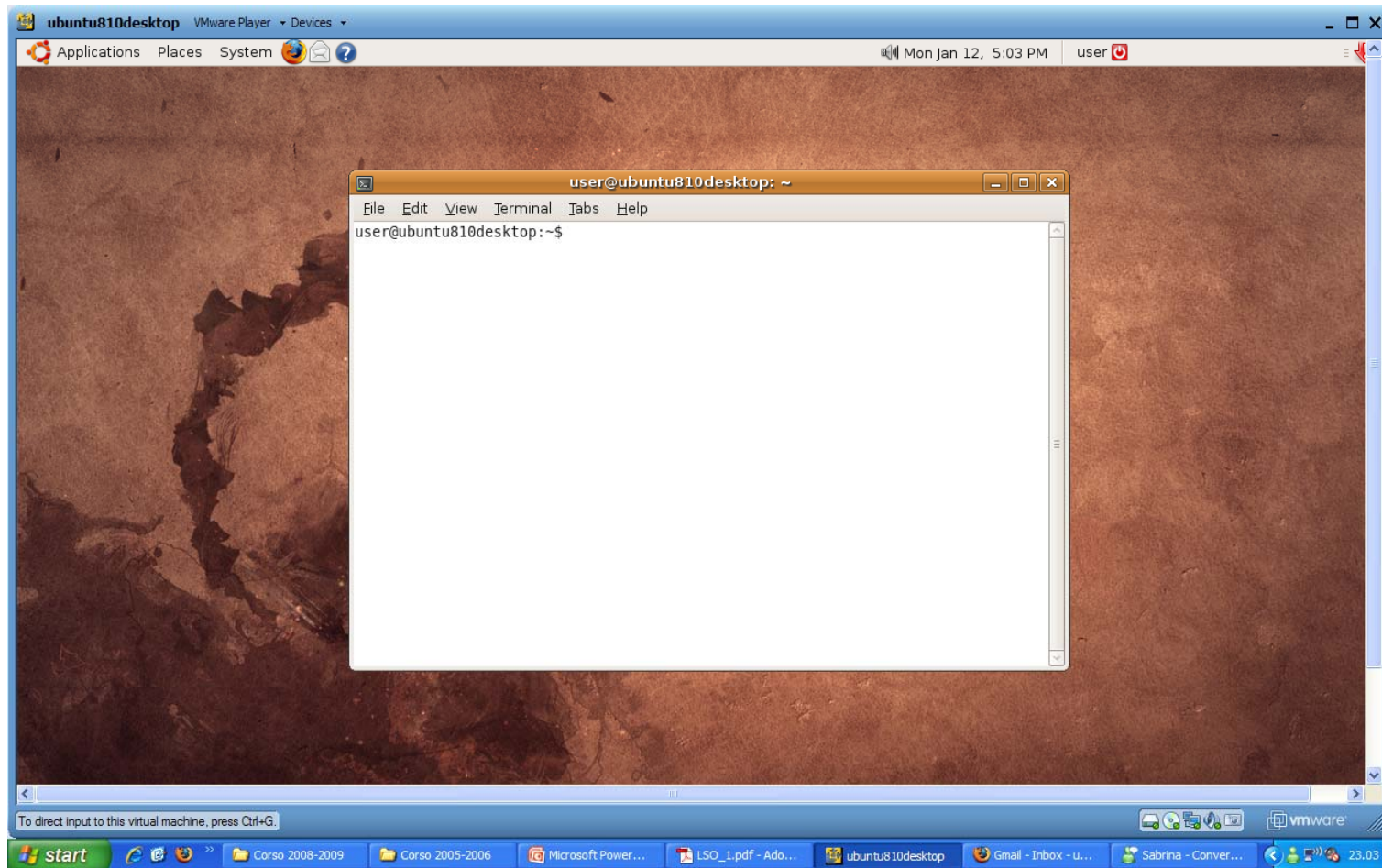
- **Red Hat Linux**
 - ▣ Una prime distribuzioni di Linux
 - ▣ La versione commerciale è la Red Hat Enterprise Linux
 - ▣ La versione free è la Fedora Project
- **Debian GNU/Linux**
 - ▣ Distribuzione free. Utilizzata sui server. Non è una distribuzione per principianti
- **SuSE Linux**
 - ▣ Distribuzione commerciale in quanto contiene diversi software a pagamento. Esiste anche una versione completamente free
- **Ubuntu**
 - ▣ Distribuzione orientata agli utenti. Completamente free. Semplice da installare.
- **Gentoo Linux**
 - ▣ Distribuzione per programmatori

La Shell



- In un ambiente testuale la shell rappresenta l'interfaccia tra l'utente ed il sistema operativo
- La shell interpreta i comandi dell'utente e lancia il programma (o i programmi) corrispondente
- Alcune shell disponibili per Linux
 - ▣ `bash` : Bourne Again Shell
 - ▣ `tcsh`: TC Shell

La Shell di Ubuntu



Home directory

- In fase di accesso al sistema ogni utente è loggato nella propria home directory
- In ogni istante dalla shell possiamo verificare in qualche directory ci troviamo mediante il comando

`pwd`

La shell Bash



- La shell bash, oltre a generare processi che eseguono programmi, è l'interprete di un linguaggio di programmazione e può eseguire:
 - ▣ Linee di comandi (modalità interattiva)
 - ▣ File di comandi, detti shell script (modalità non interattiva)
 - Nasce una nuova shell non interattiva che esegue lo script

Console e console virtuali



- Il termine console è nato in ambiente UNIX per indicare tastiera e monitor di un sistema che a cui gli utenti accedevano da remoto
- Oggi indica anche tastiera e monitor di un sistema che può essere acceduto solo localmente
- Le console virtuali permettono a un utente di collegarsi più volte dalla console, con differenti richieste di login testuali

Accesso alle console virtuali

- Le console virtuali utilizzabili su un sistema Linux sono associate ad alcuni tasti funzione (F1, F2, .., F12), in genere da F1 a F4
- Per visualizzarle e spostarsi da una all'altra si usa il tasto Alt in combinazione col tasto funzione relativo
- La console virtuale di default è quella che presenta la richiesta di login quando si fa partire la macchina. È accessibile con ALT + F1

Accesso locale grafico

- Se il sistema è predisposto per partire in modalità grafica, la richiesta di login comparirà in una finestra
- L'ambiente grafico può essere lanciato anche dopo un login testuale, chiamando il server X, con il comando

startx

Shell di login e shell normale

- In ambiente linux distinguiamo due tipi di shell
 - ▣ **Shell di login**
 - Shell generate dal processo di login
 - ▣ **Shell normale**
 - Shell generate da istanze di xterm
- Istanze di shell di natura diversa hanno file di configurazione diversi

File per la Shell di login

- Quando bash parte, tenta di eseguire nell'ordine i seguenti script di inizializzazione dell'ambiente
 - ▣ **/etc/profile**
 - File in comune per tutti gli utenti
 - Modificabile solo da root
 - ▣ **~/.bash_profile o ~/.bashrc o ~/.profile**
 - File specifici per ogni utente (~ indica /home/larry)
 - Modificabili dall'utente per personalizzare l'ambiente
- Quando bash termina tenta di eseguire lo script **~/.bash_logout**

File per la Shell Normale

- Quando bash parte, tenta di eseguire nell'ordine i seguenti script di inizializzazione dell'ambiente
 - **~/.bashrc**
 - File specifico per ogni utente
 - Modificabile dall'utente per personalizzare l'ambiente
 - Per prima cosa esegue lo script **/etc/bashrc**
 - **/etc/bashrc**
 - File in comune per tutti gli utenti
 - Modificabile solo da root
 - Eseguito in ogni caso se **~/.bashrc non esiste**

Connessione tramite terminale

- La connessione tramite terminale permette di utilizzare direttamente una shell

```
$ssh bravo
```

```
bravo login: pippo
```

```
Password:*****
```

```
Last login: Tue Mar 1 19:50:38 from kudos
```

```
[max@bravo max]$ ...
```

Manuale di Linux

- Per ogni comando esiste normalmente un manuale consultabile attraverso il comando

`man <comando>`

- Il manuale è suddiviso in 10 sezioni:
 1. User Commands
 2. System Calls
 3. Subroutines
 4. Devices
 5. File Formats
 6. Games
 7. Miscellaneous
 8. System Administration
 9. Local
 10. New

Manuale di Linux

- In alcuni casi ci può essere più di un manuale con lo stesso nome ma riferiti a comandi diversi. Ad esempio

`man write`

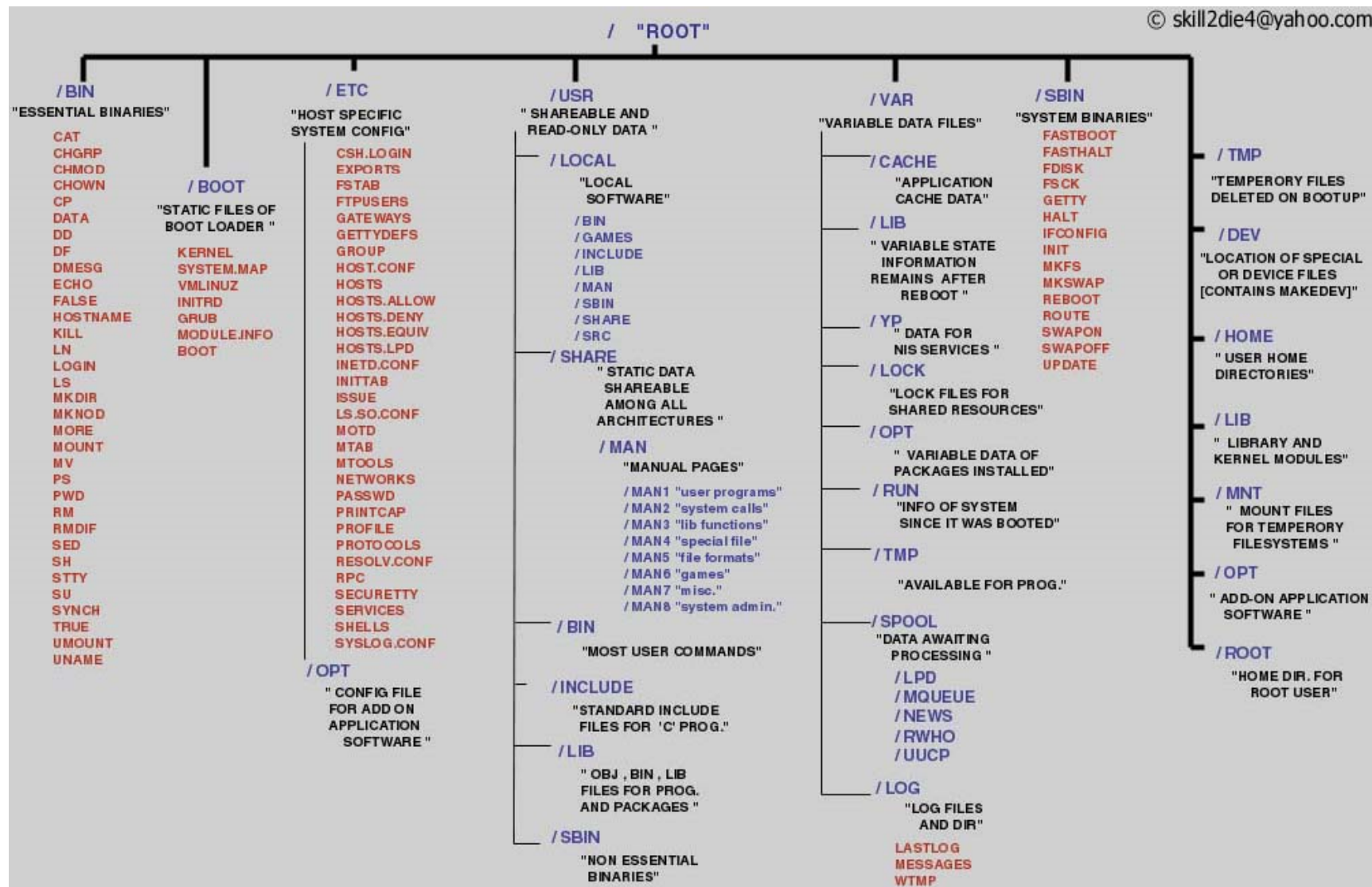
`man 2 write`

Trovare informazioni

- Ogni manuale ha una descrizione del comando a cui si riferisce
- E' possibile cercare una stringa all'interno dei manuali utilizzando il comando

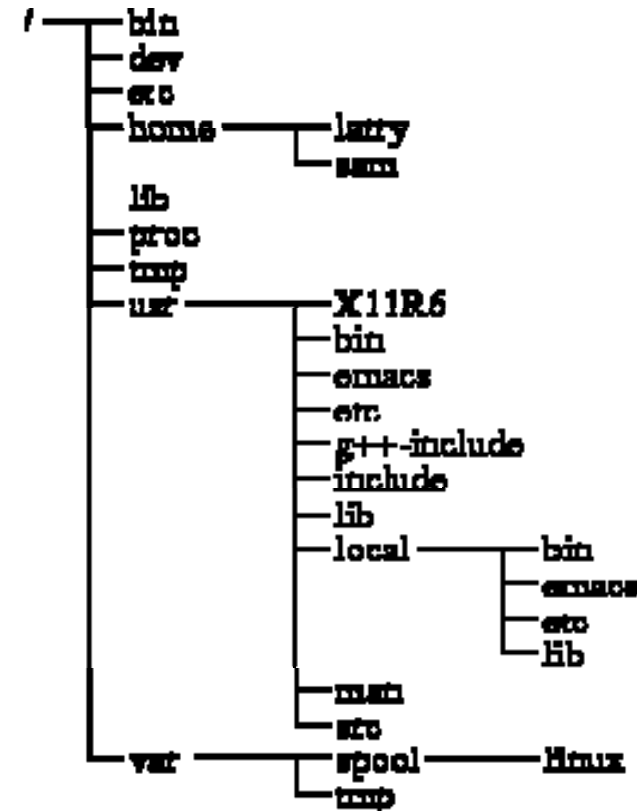
apropos <keyword>

Directory Tree - 1



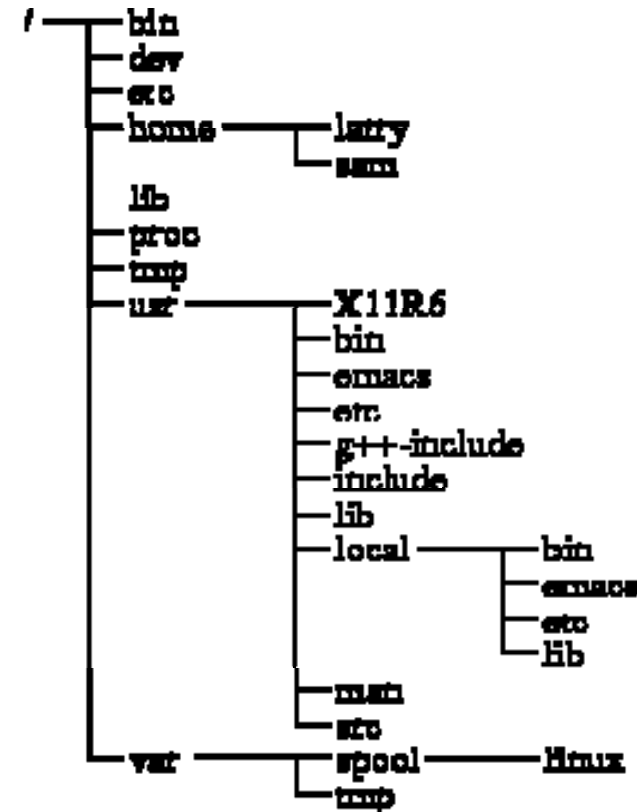
Directory Tree - 2

- **/bin** : Comandi di sistema per tutti gli utenti
- **/boot** : File necessari al boot
- **/dev** : Driver di sistema
- **/etc** : File di configurazione per il sistema operativo
- **/home** : Directory principale degli utenti. Ogni utente ne possiede una. Ad esempio l'utente larry possiede la directory /home/larry
- **/lib** : Librerie di sistema



Directory Tree - 3

- **/mnt** : Punto di accesso per i dispositivi di massa come CD-ROM, pen drive, etc...Ad esempio il lettore CD-ROM è montato in /mnt/cdrom
- **/root** : Directory dell'utente root ovvero dell'amministratore
- **/sbin** : Comandi essenziali per l'amministrazione del sistema
- **/tmp** : File temporanei. Molte distribuzioni cancellano periodicamente il contenuto di questa directory
- **/usr** : Programmi e dati da condividere e che non hanno bisogno di essere modificati
- **/var** : Dati da cambiare frequentemente. Ad esempio log di sistema, dati da mandare in stampa



L'editor di testo `vi`

- Per creare file di testo useremo l'editor di testo
`vi`
- Per aprire un file con `vi`, bisogna digitare
`vi nome_file`
- in un `xterm` o in una console virtuale
- Il file `nome_file` viene creato se non esiste,
invece è aperto se esiste

L'editor di testo `vi`

- Per poter modificare un file aperto con `vi` (nuovo o esistente)
 - ▣ Premere il tasto `i` (insert) oppure il tasto `a` (append)
 - ▣ Comparirà la parola `insert` nella parte basse della finestra
 - ▣ Alla fine delle modifiche:
 - Premere il tasto `esc`
 - Per salvare il file e chiudere `vi`, digitare `:wq`
 - Per uscire senza salvare, digitare `:q!`
- Per salvare e continuare a lavorare, digitare `:w`

Alcuni comandi di `vi` - 1

- Per spostarsi all'interno del testo
 - Premere il tasto `esc`
 - Per spostarsi alla fine del testo, digitare `G`
 - Per spostarsi alla linea n-esima, digitare `nG`
 - Per spostarsi n linee più giù, digitare `nj`
 - Per spostarsi n linee più su, digitare `nk`
 - Per spostarsi n caratteri a destra, digitare `n1`
 - Per spostarsi n caratteri a sinistra, digitare `nh`

Alcuni comandi di vi - 2

- Per spostarsi all'interno del testo
 - ▣ Premere il tasto `esc`
 - Per spostarsi di una pagina in giù, digitare `ctrl+f`
 - Per spostarsi di una pagina in su, digitare `ctrl+b`
 - Per spostarsi all'inizio dello schermo, digitare `H`
 - Per spostarsi ad n linee dall'inizio dello schermo, digitare `nH`
 - Per spostarsi alla fine dello schermo, digitare `L`
 - Per spostarsi ad n linee dalla fine dello schermo, digitare `nL`

Alcuni comandi di `vi` - 3

- Per spostarsi all'interno del testo
 - Premere il tasto `esc`
 - Per spostarsi alla metà dello schermo, digitare `M`
 - Per spostarsi all'inizio di una linea, digitare `|`
 - Per spostarsi alla fine di una linea, digitare `$`
 - Per spostarsi all'inizio della prossima parola, digitare `w`
 - Per spostarsi all'inizio della parola precedente, digitare `b`
 - Per spostarsi alla fine di una parola, digitare `e`

Alcuni comandi di `vi` - 4

- Per cercare una stringa nel testo
 - ▣ Premere il tasto `esc`
 - Per cercare dal cursore in giù, digitare `/stringa_da_cercare`
 - Per cercare dal cursore in su, digitare `?stringa_da_cercare`
 - Per cercare la prossima occorrenza, digitare `n`
 - Per cercare la prossima occorrenza invertendo la direzione di ricerca, digitare `N`

Alcuni comandi di `vi` - 5

- Per cancellare del testo
 - ▣ Premere il tasto `esc`
 - Per cancellare `n` caratteri, digitare `nx`
 - Per cancellare la parola successiva, digitare `dw`
 - Per cancellare la parola precedente, digitare `db`
 - Per cancellare `n` linee, digitare `ndd`
 - Per cancellare il resto di una linea, digitare `D`
 - Per cancellare il resto del file, digitare `dG`

Alcuni comandi di `vi` - 6

- Per fare copia, taglia, incolla e undo
 - ▣ Premere il tasto `esc`
 - Per copiare n linee, digitare `nY`
 - Per tagliare linee, caratteri e parole, utilizzare i comandi per cancellare
 - Per incollare testo tagliato o copiato, digitare `p` (più `p` incollano di nuovo)
 - Per annullare l'ultima modifica data in modalità comandi (dopo aver premuto `esc`), digitare `u`

Esecuzione di comandi

- Per mandare in esecuzione un programma basta digitare il nome dell'eseguibile in una shell
- La shell cerca il programma in directory predefinite, in genere in
 - `/bin, /usr/local/bin, /usr/bin`assegnati alla variabile d'ambiente PATH
- Se il programma è in qualsiasi path non predefinito occorre digitare il path assoluto
- Se il programma è nella directory corrente (.) e questa non è inserita nei path di default, occorre digitare
 - `./nome_eseguibile`

Manipolare file

- Copiare
 - ▣ `cp fileorigine filedestinazione`
 - ▣ `cp ciao.c ciaoCopia.c`
- Rinominare
 - ▣ `mv fileorigine filedestinazione`
 - ▣ `mv ciaoCopia.c copia.c`
- Spostare
 - ▣ `mv fileorigine pathdestinazione/`
 - ▣ `mv a.out eseguibili/`
- Cancellare
 - ▣ `rm file`
 - ▣ `rm ciao`
- Visualizzare a video (in alcuni casi)
 - ▣ `more file more /etc/shells`
 - ▣ `cat file cat ~/.bashrc copia.c`
 - ▣ `less file less /etc/profile`

Visualizzare file nascosti

- La home directory contiene file e directory nascosti, i cui nomi iniziano con un punto, visualizzabili con **ls -a**
- In genere sono i file di configurazione della shell e di alcune applicazioni installate
- Nota “.” e “..” non sono file contenuti nella home. Compaiono in tutte le directory e indicano, rispettivamente, la directory corrente e la sua directory padre

Manipolare le directory

- Copia
 - ▣ `cp -R dirorigine dirdestinazione`
 - ▣ `cp -R eseguibili/ dircopia`
- Rinominare
 - ▣ `mv dirorigine dirdestinazione`
 - ▣ `mv dircopia esecopia`
- Spostare
 - ▣ `mv dirorigine pathdestinazione/`
 - ▣ `mv esecopia/ ..`
- Cancellare
 - ▣ `rm -R dir rm -R ../esecopia/`
 - ▣ `rmdir dir (per directory vuote)`
- Visualizzare da ls come un file
 - ▣ `ls -d dir`
 - ▣ `ls -d eseguibili/`

Comandi base su file e directory

- Cambiare directory

cd <nuova directory>

- Mostrare il contenuto di una directory

ls

- Mostrare il contenuto di un file

cat <nomefile>

- Cancellare un file

rm <nomefile>

rm -i <nomefile>

- Mostrare un file in maniera interattiva

more <nomefile>

Gestire i files

- Copiare un file

```
cp source-file destination-file
```

```
cp -i source-file destination-file
```

- Spostare o rinominare un file

```
mv existing-filename new-filename
```

- Mostrare l'inizio di un file

```
head <filename>
```

- Mostrare la fine di un file

```
tail <filename>
```

Cercare programmi

- Linux dispone di diversi strumenti per la ricerca di file. Se cerchiamo un programma possiamo usare

`which -a nome_programma`

- Questo restituisce il path assoluto di tutti gli eseguibili e/o gli shell script, che si chiamano `nome_programma`, presenti nei path predefiniti
- I path predefiniti sono memorizzati nella variabile d'ambiente `PATH`, che è impostata in uno dei file di configurazione della shell

Alias

- Un alias è un nome assegnato ad un comando
- Può essere comodo per comandi usati spesso
Per esempio

```
alias myDir='cd /home/roberto/esequibili/'  
alias ll='ls -l'
```

- Gli alias definiti all'interno di una shell esistono finché esiste la shell
- Per renderli permanenti bisogna metterli in un file di configurazione della shell
- Il comando **alias** visualizza tutti gli alias definiti

Comando source

- ❑ I comandi **source** e **.** (da non confondere con il riferimento alla directory corrente) sono equivalenti tra loro ed interni alla shell.
- ❑ La loro sintassi è *source nomefile [arg...]* o *. nomefile [arg...]* e il loro effetto è di mandare in esecuzione
- ❑ all'interno della shell corrente i comandi contenuti in nomefile.
- ❑ Il file nomefile può essere un file di configurazione o uno shell script, nel qual caso potrebbe volere degli argomenti.

Cercare un file

- Il comando `locate` permette di cercare un file qualsiasi all'interno del file system. La sua sintassi è

`locate <pattern>`

- Per aggiornare il database utilizzare il comando (da root)

`updatedb`

Cercare stringhe nei files

- Per trovare una stringa in uno o più file possiamo usare il comando grep la cui sintassi è

```
grep stringa nomefile [nomefile ...]
```

User ID e Group ID

- Ad ogni utente del sistema sono associati due numeri non negativi chiamati :
 - ▣ **UID: User ID** (numero di utente)
 - ▣ **GID: Group ID** (numero di gruppo)
- Questi due numeri sono stabiliti una volta per tutte dall'amministratore del sistema e sono univoci
 - ▣ Si possono leggere anche nel file di sistema **`/etc/passwd`**
- Lo User ID è univocamente associato allo **username**
- Il Group ID al **groupname**

Il comando id

- Il comando **id** consente di conoscere i valori numerici e simbolici (username e groupname) dello User ID e del Group ID

- Ad esempio:

```
> id
```

```
uid=332(rossi) gid=15(medicina)
```

Cambiare password

- Il comando **passwd** permette di cambiare la propria password
- È necessario inserire la vecchia password e immettere due volte quella nuova

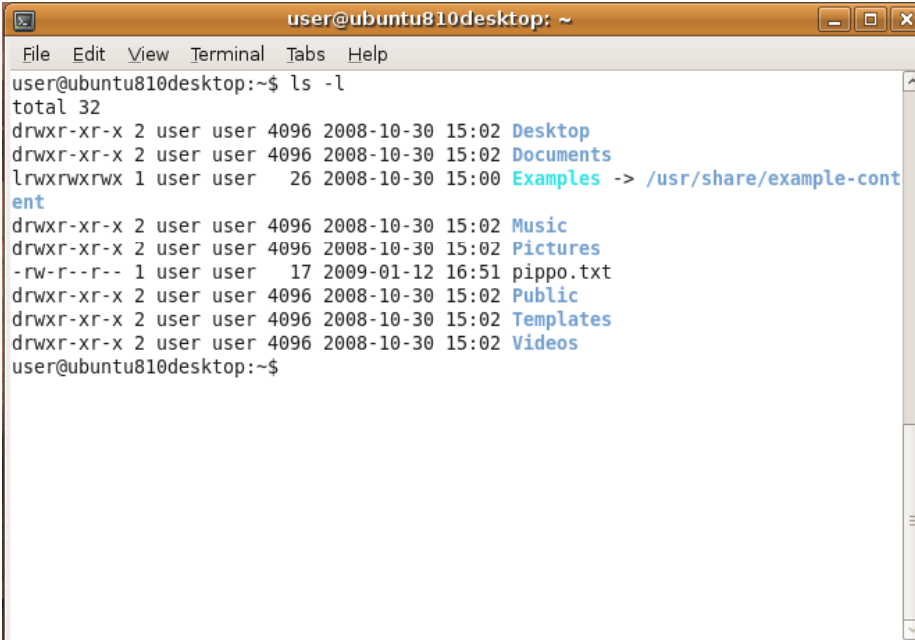
Proprietà di File e Directory

- Per ogni file o directory nel filesystem, Linux conserva una serie di proprietà, tra cui
 - ▣ Tipo di file
 - ▣ Proprietario
 - ▣ Gruppo di appartenenza del proprietario
 - ▣ Data dell'ultima modifica
 - ▣ Dimensione in byte
 - ▣ Permessi di accesso
 - ▣ Numero di link fisici

Visualizzazione delle proprietà

- Per visualizzare le principali proprietà dei file nella directory corrente, possiamo usare

ls -l



```
user@ubuntu810desktop: ~  
File Edit View Terminal Tabs Help  
user@ubuntu810desktop:~$ ls -l  
total 32  
drwxr-xr-x 2 user user 4096 2008-10-30 15:02 Desktop  
drwxr-xr-x 2 user user 4096 2008-10-30 15:02 Documents  
lrwxrwxrwx 1 user user 26 2008-10-30 15:00 Examples -> /usr/share/example-cont  
ent  
drwxr-xr-x 2 user user 4096 2008-10-30 15:02 Music  
drwxr-xr-x 2 user user 4096 2008-10-30 15:02 Pictures  
-rw-r--r-- 1 user user 17 2009-01-12 16:51 pippo.txt  
drwxr-xr-x 2 user user 4096 2008-10-30 15:02 Public  
drwxr-xr-x 2 user user 4096 2008-10-30 15:02 Templates  
drwxr-xr-x 2 user user 4096 2008-10-30 15:02 Videos  
user@ubuntu810desktop:~$
```

Tipi di file supportati da Linux

- Il tipo di un file è indicato dal primo carattere sulla linea che lo descrive nell'output di `ls -l`
 - `-` file normale (eseguibile, immagine, testo, ...)
 - `d` directory
 - `c` file speciale che rappresenta una device a carattere
 - `b` file speciale che rappresenta una device a blocchi
 - `l` link simbolico
 - `s` file speciale che rappresenta un socket
 - `p` file speciale che rappresenta una pipe con nome

Link fisici

- Un link fisico è un riferimento al file fisico e ogni modifica a un link fisico è una modifica sul file stesso
- Ogni nuovo file ha un solo link fisico
- Ogni nuova directory ha 2 link fisici
- I link fisici possono essere creati solo per i file normali, usando il comando

`ln nomefile nomelink`

Link simbolici

- Un link simbolico è un file che contiene il nome di un altro file
- Per creare un link simbolico si usa

`ln -s nomefile nomelink`

Classi di permessi

- A ciascun file e directory vengono assegnati dei permessi che riguardano:
 - L'*utente* che ne è il proprietario (solitamente il suo creatore)
 - Il *gruppo di appartenenza* (solitamente il gruppo principale del suo creatore)
 - Un utente può far parte di più gruppi
 - Tutti gli *altri* che corrispondono agli utenti che non ricadono nei due casi sopra elencati

Permessi di base

- **Letture (r)**
 - ▣ Applicato ai file permette di leggerne il contenuto
 - ▣ Applicato alle directory consente di elencare i nomi dei file e delle sottodirectory che contengono
- **Scrittura (w)**
 - ▣ Applicato ai file permette di modificarne il contenuto
 - ▣ Applicato alle directory permette di aggiungere o rimuovere in esse dei file e altre sottodirectory
- **Esecuzione (x)**
 - ▣ Applicato ai file permette di eseguirli
 - ▣ Applicato alle directory permette di attraversarle per accedere ai file ed alle sottodirectory in esse contenute
 - Ma non di elencarne il contenuto, per il quale serve anche il permesso di lettura

Altri permessi

- Set user ID (detto anche setuid o suid)
 - ▣ Indica che il programma va eseguito con i privilegi dell'utente proprietario del file
 - ▣ Viene comunemente usato per consentire ad utenti ordinari di eseguire programmi che richiedono particolari privilegi di sistema di cui normalmente dispone solo l'amministratore
- Set group ID (detto anche setgid)
 - ▣ indica che il programma va eseguito con i permessi del gruppo assegnato al file anziché quelli del gruppo principale dell'utente che lo avvia.
 - ▣ Applicato alle directory indica che i nuovi file e sottodirectory creati al loro interno avranno come gruppo assegnato quello della directory che li contiene anziché quello principale dell'utente che crea il file o la directory;
- Sticky
 - ▣ Indica invece che i file di una directory possono essere cancellati e spostati solamente dagli utenti che ne sono proprietari, o dall'utente proprietario della directory che li contiene

Permessi

- I permessi di accesso a un file sono indicati dai 9 nove caratteri che seguono il tipo sulla relativa linea nell'output di `ls -l`
- I permessi riguardano:
 - ▣ Il proprietario del file (3 caratteri a sinistra)
 - ▣ Gli utenti nel gruppo del proprietario del file (3 caratteri centrali)
 - ▣ Tutti gli altri utenti (3 caratteri a destra)
- In ogni tripla il carattere
 - ▣ a sinistra indica il permesso di lettura (r per sì, - per no)
 - ▣ Centrale indica il permesso di scrittura (w per sì, - per no)
 - ▣ a destra indica il permesso di esecuzione (x per sì, - per no)

Esempi

- `-rw-----` indica un file leggibile e scrivibile solo dal proprietario
- `drwx-----` indica una directory leggibile, scrivibile e attraversabile dal proprietario, ma inaccessibile per tutti gli altri
- `drwxr-xr-x` indica una directory leggibile, scrivibile e attraversabile dal proprietario, leggibile e attraversabile per il gruppo e per gli altri
- `-rw-r--r--` indica un file leggibile da tutti, ma scrivibile solo dal proprietario
- `-----r--` indica un file leggibile da tutti *eccetto* il proprietario ed dal gruppo di utenti assegnato al file
- `-r-sr-xr-x` indica un file eseguibile leggibile ed eseguibile da tutti con anche il permesso speciale *set user ID*
- `-r-Sr--r--` indica un file leggibile da tutti con anche il permesso speciale *set user ID* ma senza il permesso di esecuzione
- `drwxrws---` indica una directory leggibile, scrivibile e attraversabile dal proprietario e dal gruppo di utenti assegnato alla directory, con anche il permesso speciale *set group ID*
- `drwxrwxrwt` indica una directory leggibile, scrivibile e attraversabile da tutti e con il permesso *sticky*

Modificare i Permessi di Accesso

- Per modificare i permessi di accesso a un file bisogna essere il proprietario o l'utente root
- Il comando da usare è

chmod mode nomefile

dove mode è una rappresentazione dei nuovi permessi

- ▣ Può essere in formato ottale
- Ad ogni tripla si associa una cifra ottale in cui un 1 abilita il permesso corrispondente e uno 0 lo disabilita, ad esempio

chmod 600 ciao.c

- In formato simbolico per ogni tipo di utente (uoga) un + abilita e un - disabilita un permesso (rwx), ad esempio

chmod a+rwx ciao.c

Identificatori del controllo di accesso



- Tutte le operazioni del sistema vengono compiute dai processi
- Per poter implementare un controllo sulle operazioni occorre poter identificare chi è che ha lanciato un certo programma
- Pertanto anche a ciascun processo dovrà essere associato ad un utente e ad un gruppo

Identificatori del controllo di accesso

| Suffisso | Gruppo | Denominazione | Significato |
|--------------|------------|------------------------|--|
| uid | real | user-ID reale | indica l'utente che ha lanciato il programma |
| gid | ” | group-ID reale | indica il gruppo principale dell'utente che ha lanciato il programma |
| euid | effective | user-ID effettivo | indica l'utente usato nel controllo di accesso |
| egid | ” | group-ID effettivo | indica il gruppo usato nel controllo di accesso |
| – | – | group-ID supplementari | indicano gli ulteriori gruppi cui l'utente appartiene |
| – | saved | user-ID salvato | è una copia dell'euid iniziale |
| – | ” | group-ID salvato | è una copia dell'egid iniziale |
| fsuid | filesystem | user-ID di filesystem | indica l'utente effettivo per l'accesso al filesystem |
| fsgid | ” | group-ID di filesystem | indica il gruppo effettivo per l'accesso al filesystem |

Process Real User/Group ID

- Per ogni processo che viene creato (per esempio al login ne viene creato uno) sono definiti i due numeri:
 - ▣ **Process Real User ID (uid)**
 - ▣ **Process Real Group ID (gid)**
- Vengono ereditati dallo User ID e Group ID dell'utente che ha creato il processo
- Normalmente "Process Real User ID" e "Process Real Group ID" rimangono inalterati per tutta la vita del processo

Process Effective User/Group ID

- Il processo è però caratterizzato anche da
 - ▣ **Process Effective User ID (euid)**
 - ▣ **Process Effective Group ID (egid)**
- Questi identificatori normalmente sono identici ai corrispondenti del gruppo real ma...
- Nel caso in cui il programma che si è posto in esecuzione abbia i bit suid o sgid impostati essi saranno impostati all'utente e al gruppo proprietari del file

Assegnare euid/guid ad un processo

- Per ogni programma possiamo indicare una proprietà che decide cosa assegnare a euid e/o guid in fase di esecuzione
- Se al posto della x nel permesso di esecuzione del proprietario e/o del gruppo a cui il proprietario appartiene abbiamo una s allora: i campi euid e/o guid del processo saranno uguali a uid e/o gid del proprietario del file

Un esempio

- I programmi come “ls”, “cat”, “cp” vengono eseguiti dai normali utenti attraverso il proprio UID
- Un programma come `passwd` necessita di essere eseguito con privilegi di sistema in quanto deve modificare il file `/etc/passwd`
- Quando l'utente “pippo” esegue il comando “passwd” i privilegi del processo sono:

```
Real-UID = ananta
Effective-UID = ananta
Saved-UID = root
```

Il programma chiama una system call `seteuid(0)` modificandolo in:

```
Real-UID = ananta
Effective-UID = root
Saved-UID = root
```

- In questo modo il processo `passwd` può modificare il file `/etc/passwd` e cambiare la password per l'utente pippo
- L'utente continua a non poter modificare il file `/etc/passwd`

Il comando ps - 1

- Per visualizzare i processi in esecuzione si usa il comando

ps

- Da solo mostra i processi in esecuzione nella shell da cui è chiamato
- Alcune opzioni:
 - ▣ **-u** dà alcune informazioni aggiuntive sui processi, tipo percentuali di CPU e memoria usate, utente proprietario, stato, ...
 - ▣ **-j** dà informazioni aggiuntive diverse, tipo pid del padre, gruppo e sessione di appartenenza, stato, ...
 - ▣ **-x** mostra tutti i processi dell'utente che lo chiama
 - ▣ **-ux** dà le informazioni aggiuntive di -u, con -jx quelle di j

Il comando ps - 2

- Altre opzioni:
 - **-u user_id** mostra tutti i processi dell'utente user_id,
 - **j -u user_id** dà le informazioni aggiuntive di j
 - **-e** mostra tutti i processi in esecuzione nel sistema, di qualsiasi utente siano, ma con poche informazioni
 - **j -e** mostra tutti i processi in esecuzione nel sistema, con le informazioni aggiuntive date dall'opzione j
 - **-aux** si ottengono quelle dovute all'opzione -u

Archiviazione dei file

- Il comando **tar** permette di creare archivi e estrarre file da archivi, ma non effettua compressione, ad esempio
- **tar cvf archivio.tar file-da-includere**
 - ▣ Crea l'archivio archivio.tar e ci mette dentro i file-da-includere
 - ▣ File-da-includere può essere una directory
- **tar uvf archivio.tar file-da-aggiungere**
 - ▣ Aggiunge file-da-aggiungere all'archivio archivio.tar
- **tar xvf archivio.tar**
 - ▣ Estrae i file contenuti nell'archivio archivio.tar nella directory corrente
- **tar tvf archivio.tar**
 - ▣ Lista i file contenuti nell'archivio archivio.tar senza estrarli

Compressione di file

- I comandi **gzip** e **gunzip** permettono rispettivamente di comprimere e decomprimere file, ad esempio
- **gzip -9 ese.tar**
 - ▣ Comprime ese.tar (comprime anche file non .tar) in ese.tar.gz
 - ▣ -9 indica massima compressione
- **gunzip ese.tar.gz**
 - ▣ Decomprime ese.tar.gz nella directory corrente
- **tar cfvz ese.tar.gz /mydir**
 - ▣ z permette a tar di comprimere /mydir a volo usando gzip

Il Prompt della Shell

- Il prompt è una stringa di caratteri che la shell mostra quando è in attesa di un comando
- La stringa per il prompt è definita nella variabile di ambiente **PS1**
- Possiamo modificarla con **PS1="quello che vi pare"**
- Esistono dei simboli speciali per assegnare alcuni valori particolari alla variabile PS1

Alcuni simboli speciali per PS1

- `\t` orario
- `\d` data
- `\s` bash
- `\w` directory corrente
- `\u` username dell utente
- `\h` nome host
- `\` `\` `\`
- `\$` # per root, \$ per gli utenti

Esempi

- Possiamo comporre i simboli speciali con altri caratteri

```
> PS1=' \u@\h: '
```

```
> PS1='Hello, world'
```

```
> PS1='#@#'
```

Carattere speciale &

- Durante l'esecuzione di un programma non si può interagire con la shell che lo esegue perché il programma è in foreground
- Se la linea di comando che lancia il programma viene terminata con il carattere speciale &, l'esecuzione va in background ed è possibile continuare a lavorare con la shell
- Nel primo caso la shell chiama la system call wait, nel secondo no

Processi in background

- Un programma in background si blocca per ricevere input, ma il suo output interferisce con la shell.
- Questo accade perchè i due processi, la shell e il programma, condividono lo standard output
- In genere si mettono in background i programmi che aprono finestre grafiche e durano a lungo, es. **firefox**, **gedit**, **startx**, .

Sospensione dei programmi

- L'esecuzione di un programma in foreground può essere sospesa usando **ctrl+z**
- Il programma sospeso compare in stato di stopped nella tabella della shell
- La sua esecuzione può essere ripresa in background con il comando **bg** e in foreground con il comando **fg**
- Se più programmi vengono sospesi, la loro riattivazione avviene in ordine LIFO

Il carattere speciale |

- Una pipe è un canale di comunicazione tra due processi, in cui un processo scrive qualcosa che l'altro leggerà
- La shell riconosce il carattere speciale | per creare una pipe tra due processi
- `proc_a | proc_b` significa che l'output di `proc_a` sarà l'input di `proc_b`
 - ▣ es. `ls /dev | more` manda l'output di `ls /dev` in input a `more`

Il carattere speciale |: esempio

- Proviamo ad usare | per sapere se la variabile che contiene il valore del prompt è una variabile di ambiente
- In pratica vogliamo sapere se l'output del comando env contiene la stringa PS1
- Possiamo mandare l'output di env in input a grep con una pipe:

```
env | grep PS1
```

La Ridirezione dell'input con <

- Una pipe è una forma di ridirezione di I/O
- Un altro modo consiste nel ridirigere l'I/O da/su file
- La shell riconosce il carattere speciale < per la ridirezione dello standard input
- `proc < input_file` significa che proc leggerà l'input da `input_file`, ad esempio

`sort < elenco`

esegue il comando `sort` con lo standard input ridiretto sul file `elenco`

La Ridirezione dell'input con >

- La shell riconosce il carattere speciale > per la ridirezione dello standard output
- `proc > output_file` significa che proc scriverà l'output su output_file, ad esempio

`ls > dir_cont`

esegue il comando ls con lo standard output ridiretto sul file dir_cont

- Il file su cui è ridiretto lo standard output di un processo è creato se non esiste, sovrascritto se esiste

Ancora sulla ridirezione

- La coppia di caratteri speciali >> produce l'effetto di accodamento sul file su cui è ridiretto l'output, ad esempio

```
ls >> dir_cont
```
- accoda l'output di ls al file dir_cont
 - ▣ Il file viene comunque creato se non esisteva