1

Storia di Linux

- Fernando José Corbató (MIT, 1965), dirige il progetto per il S.O. **MULTICS**. Grande, complesso e poco efficiente, ma che possiede tutte le caratteristiche dei s.o. moderni
- Kenneth Thompson e Dennis Ritchie (Bell labs, 1971), sviluppano **UNIX** semplificando il progetto MULTICS. Successivamente sviluppano il linguaggio C e riscrivono UNIX nel nuovo linguaggio. Primo S.O. portabile su differenti calcolatori anche di piccole dimensioni
- Richard Stallman (Stanford, 1985), lancia il progetto **GNU** per la distribuzione libera e gratuita del software di Unix. Sviluppa solo alcuni tool (compilatori, debugger, editor)
- Andrew Tanenbaum (Amsterdam, 1986), professore alla Vrije University sviluppa **Minix**, piccolo S.O. Unix per processori Intel (ora sufficientemente potenti) . Sviluppato per usi didattici
- Linus Torvalds (Finlandia, 1992) sviluppa **Linux** a partire da Minix e dai tools realizzati nel progetto GNU. Versione per processori Intel poi adottata da tutte le grandi industrie

2

Distribuzioni Linux

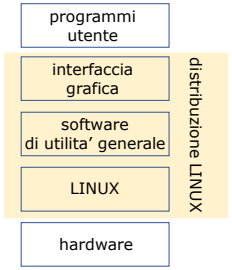
Linux e' il sistema che si occupa solo della gestione della macchina fisica (kernel):

- Gestione dei processi
- Gestione della memoria
- Gestione del file system
- Gestione dell'I/O e rete

Un **distribuzione** e' il sistema Linux corredato da numerosi programmi di utilita' generale

- Compilatori
- Browser, email, multimedia, office automation
- Iterfaccia grafica di tipo Desktop

Principali distribuzioni: Ubuntu, Debian, RedHat, Suse, Fedora, …



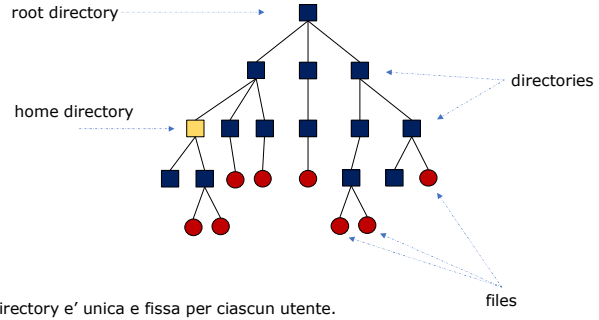
3

alcune denominazioni importanti

- **File**: una raccolta di informazioni definite dall'utente, viste come un'unica entita' accessibile attraverso un nome. Diversi tipi di file: testo, eseguibili, video, ... Sinonimo di Documento
- **Directory**: una raccolta di file (o altre directory) accessibile attarverso un nome. Sinonimo di Cartella oppure Archivio
- **File System**: l'organizzazione logica di file e directory. Linux ha un file system gerarchico strutturato ad albero, in cui gli utenti possono navigare
- **Root Directory**: la directory di piu' alto livello nel file system di Linux. Identificata con **/**
- **Home Directory**: directory di proprieta' dell'utente, in cui puo' leggere e scrivere.
- **Working Directory**: directory in cui si sta lavorando. Concide con la hom directory all'inizio della sessione di lavoro. Poi puo' cambiare.

4

struttura del file system di Linux



la home directory e' unica e fissa per ciascun utente.
Ha il nome convenzionale **~**

la working directory puo' variare nel corso della sessione di lavoro. Ha il nome convenzionale **. (punto)**

La directory di livello superior alla working directory ha il nome convenzionale **.. (doppio punto)**

5

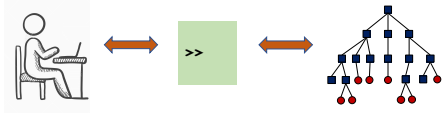
La shell

- Oltre all'interfaccia grafica (desktop) fornita dalla distribuzione, LINUX possiede un'interfaccia "a line di comando" derivante dall'interfaccia di UNIX del 1972
- Tale interfaccia prende il nome di shell (guscio), detto anche terminale.
- La shell avvisa l'utente che e' pronta per accettare un **comando** visualizzando una stringa chiamata **prompt**

esempio:

```
>>  comando
```

- Attraverso la shell l'utente e' in grado di agire sui file e le directory.

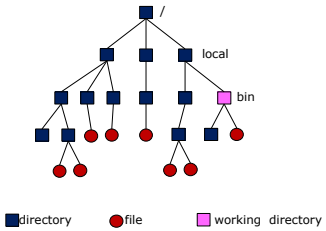


- La shell e' la sola interfaccia utilizzabile quando si opera su **sistemi remoti**

6

Il comando " pwd "

- **Visualizza la directory corrente di lavoro** (pwd = print working directory)
- Sono mostrate i nomi di tutte le directory partendo dalla root directory fino alla working directory, separati da /
- Utile per sapere in quale directory del file system si sta lavorando




```
>> pwd
/local/bin
>>
```

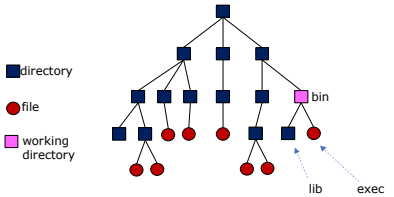
7

Il comando " ls "

- **Visualizza il contenuto di una directory** (ls = LiSt)

Diversi modi di utilizzo

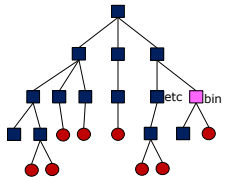
- `ls` visualizza il contenuto della working directory
- `ls directory` visualizza il contenuto della directory specificata
- `ls -l` visualizza ulteriori informazioni (permessi, data e ora creazione,…)




```
>> ls
exec    lib
>>
```

8

Il comando " ls "



directory

file

working directory

```
>> ls ..
bin     etc
>>
>> ls -l ..
dr-xr-xr-x 2 root root 4096 Feb 17 05:12 bin
dr-xr-xr-x 2 root root 4096 Feb 17 05:12 bin
>>
```

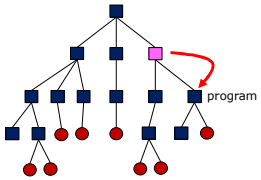
9

Il comando " cd "

- **Cambia la directory corrente di lavoro** (cd = Change Directory)

Diversi modi di utilizzo

- `cd nome` la nuova directory e' quella specificata (errore se non esiste)
- `cd ..` la nuova directory e' quella di livello superiore
- `cd` la nuova directory e' la home directory



Esempio:

```
>> cd program
>>
```

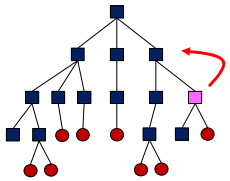
directory file working directory

10

Il comando " cd "

Esempio:

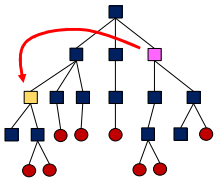
```
>> cd ..
>>
```



directory file working directory

Esempio:

```
>> cd
>>
```

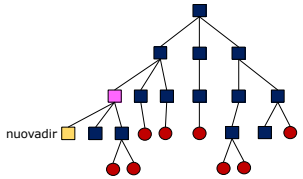


directory file home directory working directory

11

Il comando " mkdir "

- **Crea un nuova directory** (mkdir = MaKe DIRectory)
- Obbligatorio specificare un nome
- Se gia' esiste un file o una directory con il nome specificato, il comando mkdir restituisce un messaggio di errore



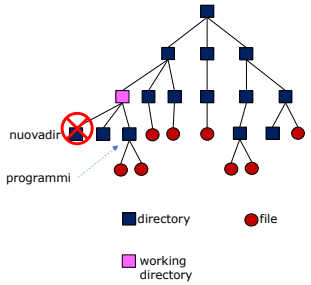

```
>> mkdir nuovadir
>>
```

directory file working directory nuova directory

12

Il comando " rmdir "

- **Elimina una directory esistente** (rmdir = ReMove DIRectory)
- Obbligatorio specificare il nome della directory
- La directory deve essere vuota (altrimenti segnala un errore)

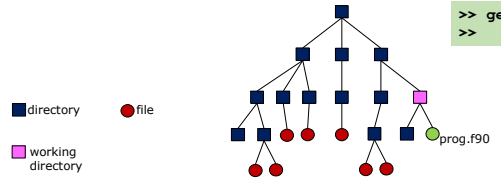



```
>> rmdir nuovadir
>> rmdir nuovadir
Error: No such file or directory
>> rmdir programmi
Error: Directory not empty
>>
```

13

Il comando " gedit "

- **Un semplice editor di testo per scrivere programmi** (o altri file di tipo testo)
- Necessario specificare un nome in fase di creazione
- Se il file gia' esiste, viene riaperto per eventuali modifiche
- N.B. gedit e' l'editor dell'interfaccia grafica e non puo' essere utilizzato su sistemi remoti. Editor della shell sono: vi ed emacs




```
>> gedit prog.f90
>>
```

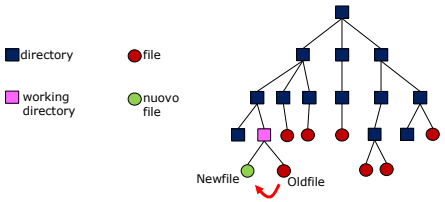
14

Il comando " cp "

Crea una copia di un file esistente (cp = CoPy)

Obbligatori **due argomenti:**

- Primo argomento : il file da copiare
- Secondo argomento puo' essere:
  - Un file: viene creato un nuovo file o viene sovrascritto uno vecchio
  - Una directory esistente: il file viene copiato nella directory specificata, utilizzando lo stesso nome

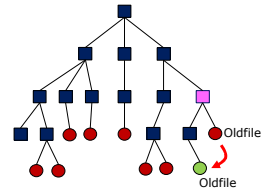



```
>> cp Oldfile Newfile
>>
```

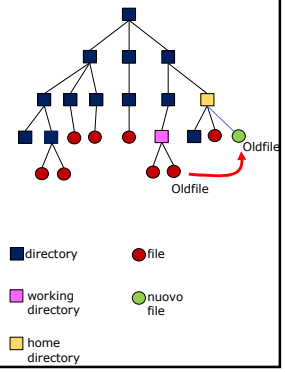
15

Il comando " cp "

```
>> cp Oldfile programmi
>>
```




```
>> cp Oldfile ~
>>
```



16

Il comando " mv "

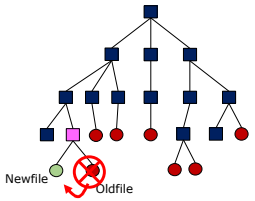
Sposta o rinomina un file esistente (mv = MoVe)

Equivale a una copia e una rimozione (cp seguito da rm)

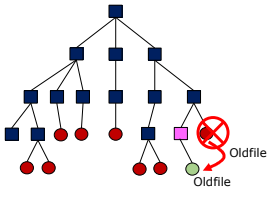
Obbligatori **due argomenti:**

- Primo argomento : il file da spostare o rinominare
- Secondo argomento puo' essere:
  - Un file: viene rinominato il file
  - Una directory: il file viene spostato nella directory specificata

```
>> mv Oldfile Newfile
>>
```



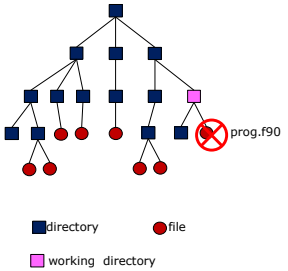

```
>> mv ../Oldfile  .
>>
```



17

Il comando " rm "

- **Elimina un file esistente** (rm = ReMove)
- Obbligatorio specificare il nome del file

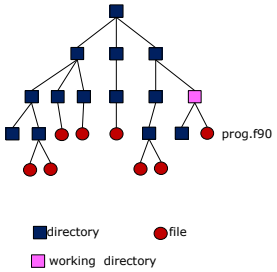



```
>> rm prog.f90
>>
```

18

Il comando " cat "

- **Mostra il contenuto di un file**
- Necessario specificare un nome

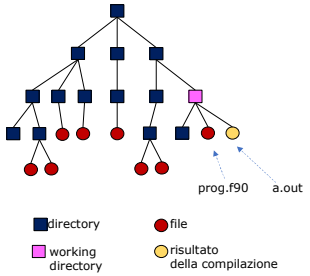



```
>> cat prog.f90
PROGRAM MAIN
INTEGER:: X, Y, Z
READ*, X, Y
Z = X + Y
PRINT*, Z
END
>>
```

19

Il comando " gfortran "

- **Comando per la compilazione di programmi FORTRAN**
- Un file contenente un programma FORTRAN deve avere estensione **.f90** (ad es. prog.f90)
- Segnala eventuali errori di sintassi
- In caso di assenza di errori, produce un file eseguibile di nome **a.out**




```
>> gfortran prog.f90
>> ./a.out
>>
```

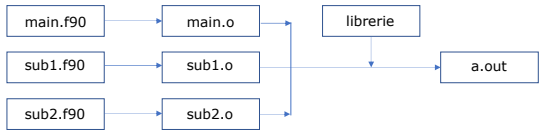
20

Il comando " gfortran "

- Il comando gfortran opera in due fasi
  1. **compilazione separata** dei singoli file che compongono il programma (ad es. programma chiamante e subroutine). Vengono prodotti dei file chiamati file oggetto (hanno estensione .o)
  2. **collegamento (linking) dei file oggetto** con le librerie (ad es. di I/O, di sistema, matematiche)
- In caso di un programma composto da piu' file il commando e'

```
>> gfortran  main.f90  sub1.f90  sub2.f90
>>
```

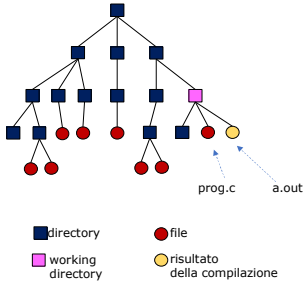
viene prodotto un unico file eseguibile a.out



21

Il comando " cc "

- **Comando per la compilazione di programmi C**
- Un file contenente un programma C deve avere estensione **.c** (ad es. prog.c)
- Segnala eventuali errori di sintassi
- In caso di assenza di errori, produce un file eseguibile di nome **a.out**




```
>> cc prog.c
>> ./a.out
>>
```

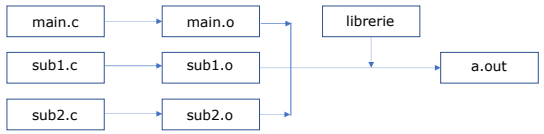
22

Il comando " cc "

- Il comando cc opera in due fasi
  1. **compilazione separata** dei singoli file che compongono il programma (ad es. programma chiamante e subroutine). Vengono prodotti dei file chiamati file oggetto (hanno estensione .o)
  2. **collegamento (linking) dei file oggetto** con le librerie (ad es. di I/O, di sistema, matematiche)
- In caso di un programma composto da piu' file il commando e'

```
>> cc  main.c  sub1.c  sub2.c
>>
```

viene prodotto un unico file eseguibile a.out



23

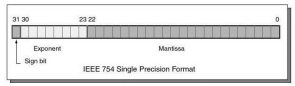
Un po' di storia (14)

William Kahan (1933)

- Matematico e informatico canadese si e' sempre occupato della accuratezza degli algoritmi per il calcolo scientifico, sviluppando tecniche per minimizzare l'errore di round-off.
- Nel 1985, e' stato il principale artefice dello standard IEEE 754, per l'aritmetica floating point utilizzato oggi in tutti i dispositivi di calcolo
- Lo standard definisce il formato della rappresentazione, le regole di arrotondamento, le operazioni e la gestione delle situazioni eccezionali (ad. divisione per 0, forme indeterminate,…)
- Kahan ha avuto una profonda influenza sulle funzionalita' dei compilatori e sull'architettura delle CPU con raccomandazioni finalizzate a migliorare l'accuratezza nei processi di calcolo
- Kahan ha vinto il Turing Award nel 1989

W. Kahan nel 1989

IEEE



24