POLITECNICO DI MILANO





Introduzione a GNU/Linux

Laboratorio Software 2008-2009

C. Brandolese M. Grotto

Sommario

Un po' di storia

Cos'è GNU/Linux

- GNU/Linux è un sistema operativo libero di tipo Unix costituito dall'integrazione del **kernel Linux** con elementi del sistema **GNU** e di altro software sviluppato e distribuito con licenza GNU GPL o con altre licenze libere.
- Linux è il nome del kernel sviluppato da Linus Torvalds a partire dal 1991 che, integrato con i componenti già realizzati dal progetto GNU (compilatore gcc, libreria Glibc e altre utility) e da software di altri progetti, è stato utilizzato come base per la realizzazione dei sistemi operativi e delle distribuzioni che vengono normalmente identificate con lo stesso nome.
- Il nome Linux a dispetto dell'assonanza tra il nome dell'ideatore e quello del sistema (LINus UniX) è da attribuire a Ari Lemke, l'amministratore che rese per primo disponibile Linux su Int a FTP.

Un po' di storia

La nascita di UNIX

- Realizzato negli anni '60, Unix e il software relativo furono creati da Dennis Ritchie, Ken Thompson e da altri programmatori esperti presso i Bell Labs.
- All'inizio chiunque fosse interessato poteva richiedere il software e i relativi manuali al solo costo della spedizione.
- I singoli centri di ricerca potevano modificare il codice sorgente ampliando le funzionalità.
- La popolarità di UNIX aumentò nel corso degli anni. Dopo la cessione ad AT&T del codice sorgente il sistema operativo diventò un prodotto commerciale proprietario: AT&T UNIX.
- Il costo era elevato ed il codice sorgente non più incluso.

Un po' di storia

Il sistema GNU

- Agli inizi degli anni '80 lo sviluppo di software proprietario da parte di società commerciali divenne la norma.
- Nel 1983 Richard Stallman lascia il suo lavoro al MIT e comincia a sviluppare un sistema operativo che permettesse a chiunque di vedere il codice, di modificarlo, di eseguirlo e di condividerlo con gli altri liberamente.
- L'annuncio originale (27 settembre) è seguito dal rilascio della prima versione del Manifesto GNU (GNU's Not Unix). Lo sviluppo del sistema inizia nel 1984.
- Nel 1990 il Sistema GNU aveva al suo interno un editor di testi (Emacs), il compilatore GCC (GNU C Compiler) e la maggior parte delle librerie e delle utility di un sistema Unix standard.
- Mancava però il componente centrale, il kernel.

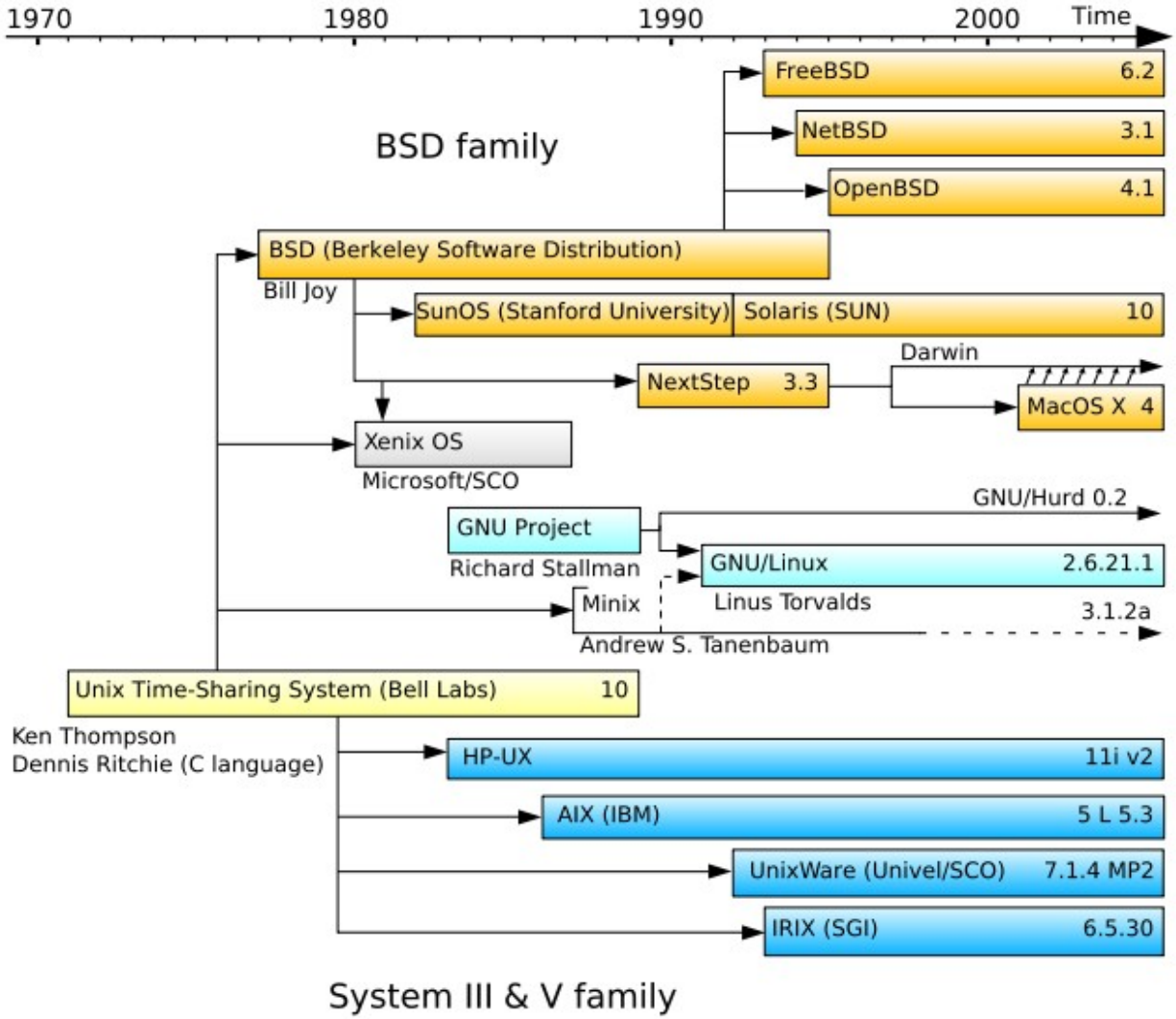
Un po' di storia

La nascita di GNU/Linux

- Agli inizi degli anni 90 Linus Torvalds, uno studente finlandese in scienze dell'informazione all'età di 21 anni, iniziò ad apportare variazioni a Minix, un sistema operativo di tipo UNIX per personal computer utilizzato a scopi didattici.
- Alla fine del 1991 pubblicò la prima versione del kernel su Internet e la battezzò Linux.
- Nel pubblicare Linux, Torvalds utilizzò la licenza GNU.
- In questo modo Torvalds invitava altri programmatori a fornire supporto.
- Da qui la nascita del sistema operativo GNU/Linux, un sistema operativo completo, moderno che può essere utilizzato sia dai programmatori che dagli utenti comuni.

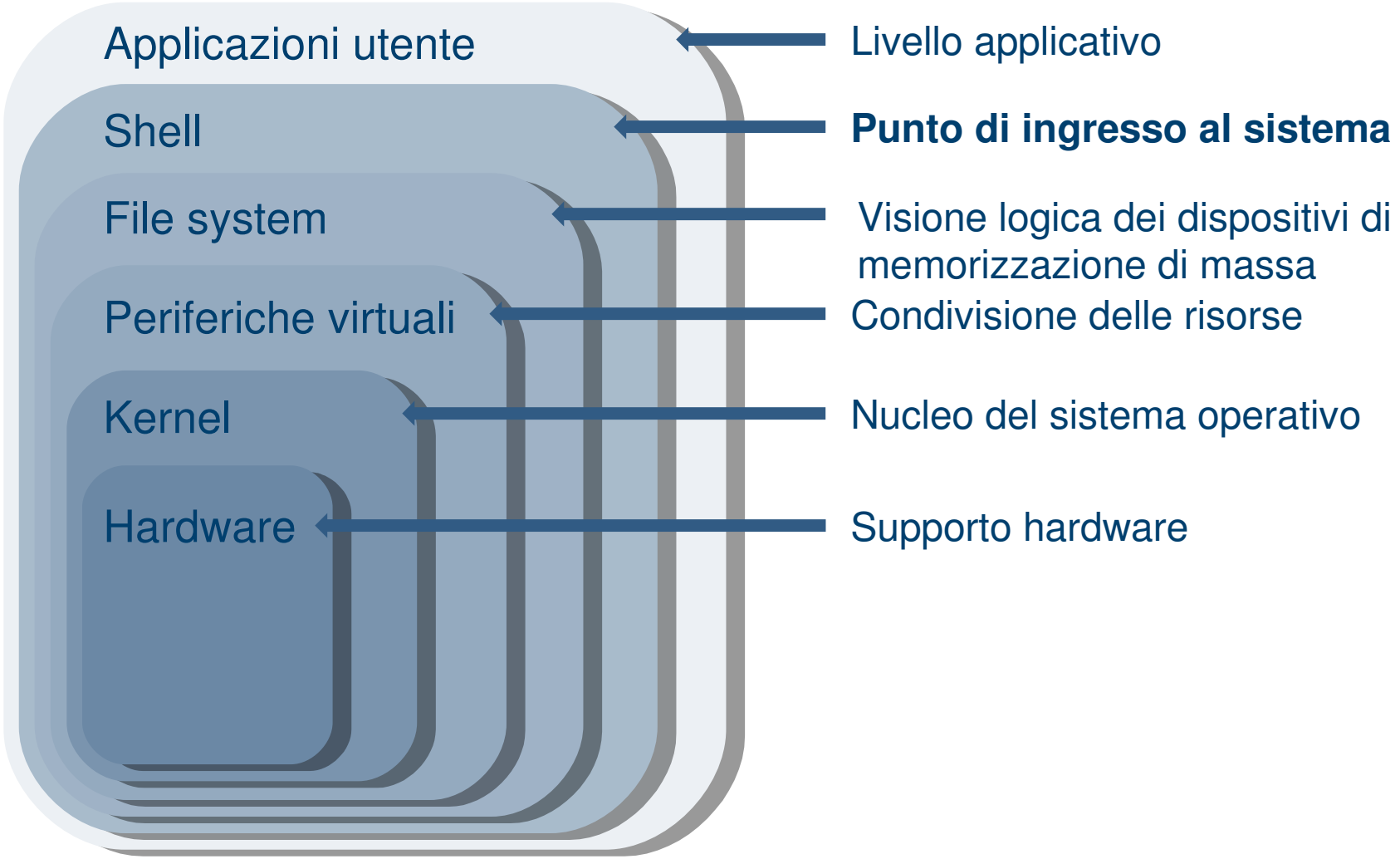
Un po' di storia

Evoluzione



Introduzione

La struttura a strati



Introduzione

Accesso al sistema

- Per accedere ad un sistema è necessario un *account*
- Un utente è identificato tramite:
  - username
  - group
  - password
- Le informazioni relative agli utenti sono raccolte nel file `/etc/passwd`

```
gdm:x:108:118:Gnome Display Manager:/var/lib/gdm:/bin/false
grotto:x:1000:1000:Matteo Grotto,,,:/home/grotto:/bin/bash
statd:x:109:65534::/var/lib/nfs:/bin/false
libuuid:x:110:120::/var/lib/libuuid:/bin/sh
pulse:x:111:121:PulseAudio daemon,,,:/var/run/pulse:/bin/false
polkituser:x:112:125:PolicyKit,,,:/var/run/PolicyKit:/bin/false
mysql:x:113:119:MySQL Server,,,:/var/lib/mysql:/bin/false
sshd:x:114:65534::/var/run/sshd:/usr/sbin/nologin
...
```

Introduzione

Accesso al sistema

- L'ingresso ad un sistema viene detto *login*
- Una tipica sessione di login:

```
Gr8-laptop login: grotto
password: *******
Last login: Fri Apr 18 12:10:29 CEST 2008 on tty1
Linux Gr8-laptop 2.6.24-21-generic #SMP Mon Aug 25 17:32:09 UTC 2008
i686
grotto@Gr8-laptop:~$
```

- La *password non viene visualizzata per motivi di sicurezza*
- La stringa "**grotto@Gr8-laptop:~$**"
  - Viene detta *shell prompt*
  - E' definibile dall'utente
- I comandi digitati al prompt sono interpretati dalla *shell*

Introduzione

La shell

- Rappresenta l'interfaccia verso il sistema operativo
- Definisce tre file standard per gestire l'I/O
  - `stdin:` standard input default: la tastiera
  - `stdout:` standard output default: il terminale corrente
  - `stderr:` standard error default: il terminale corrente
- E' l'interprete dei comandi GNU/Linux
- La shell esegue i comandi:
  - In modo interattivo: esegue i comandi immessi dall'utente al prompt
  - In modo batch: esegue comandi memorizzati in uno *script file*
- Esistono diverse shell:
  - `sh, bash, csh, tcsh`
  - Tutte le shell forniscono circa le stesse funzionalità
  - Differiscono principalmente per quanto riguarda la sintassi
  - Nel seguito faremo riferimento alla shell sh (`/bin/sh`)

Introduzione

Formato comandi

- Un comando ha la seguente sintassi generale:
  - `cmd [-opt] [file[...]] [--special] [;]`
- I vari campi indicano:
  - `cmd` Nome del comando
  - `-opt` Opzioni standard. Sono costituite dal segno - (dash) seguito da un singolo carattere, eventualmente uno o più spazi ed un'argomento.
  - `file[...]` Lista di nomi di file separata da spazi
  - `--special` Opzioni speciali (long options). Iniziano con -- e sono seguite da una stringa alfanumerica
  - `;` Termina un comando e separa i comandi di una *lista*
- Una lista di comandi ha la forma:
  - `cmd1; cmd2; ...`
- I comandi di una lista vengono eseguiti in sequenza

Introduzione

Nomi di file

- Indicati *esplicitamente* oppure mediante *globbing*
- Il *globbing* si basa su alcuni caratteri speciali combinati a formare un *pattern*
  - `~/` La home directory dell'utente corrente
  - `~user` La home directory dell'utente ***user***
  - `*` Una sequenza di zero o più caratteri qualsiasi
  - `?` Un carattere qualsiasi
  - `[s-e]` Uno dei caratteri compresi tra ***s*** e ***e***
- Il pattern viene espanso nei nomi di file che sono ad esso sovrapponibili
- Esempi:
  - `*.c` File con estensione `.c`
  - `foo[0-9]` File con nome `foo` seguito da una cifra

Comandi Fondamentali

Gestione utenti

- `passwd [user]`
  - Consente all'utente corrente di modificare la propria password
- `su user`
  - Simula la login di un utente differente rispetto a quello corrente
- `exit`
  - Abbandona la shell corrente
  - L'utente rimane autenticato sul sistema
- `who`
  - Riporta la lista degli utenti connessi alla macchina

```
root     tty2         2008-10-10 11:00
grotto   tty7         2008-10-10 10:38 (:0)
grotto   pts/0        2008-10-10 11:15 (dell-esd.cefriel.it)
```

Comandi Fondamentali

Gestione utenti

- `useradd [options]` ***`login`***
  - Crea un nuovo utente in base alle `options`
  - Il comando aggiunge una linea al file `/etc/passwd`
- `userdel [-r]` ***`login`***
  - Elimina un account dal sistema
- `groupadd [-g gid]` ***`group`***
  - Crea un nuovo gruppo di utenti
  - Il comando aggiunge una linea al file `/etc/group`
- `groupdel` ***`group`***
  - Elimina un gruppo dal sistema

Comandi Fondamentali

Variabili di ambiente

- Le varibili di ambiente
  - Contengono informazioni relative alla sessione corrente
  - Determinano alcuni aspetti della configurazione dell'ambiente
- Varibili di uso comune:
  - `HOME:` Path della home directory dell'utente
  - `USER:` Username dell'utente
  - `GROUP:` Gruppo di appartenenza dell'utente
  - `DISPLAY:` Nome del display in uso
  - `SHELL, SHLVL:` Tipo e livello di shell
  - `PATH:` Lista dei percorsi di ricerca degli eseguibili
  - `MANPATH:` Lista di ricerca delle pagine di help
  - `LD_LIBRARY_PATH:` Lista di ricerca delle librerie dinamiche
  - `HOST:` Nome della macchina corrente

Comandi Fondamentali

Variabili di ambiente

- `export var=[value]`
  - Aggiunge la variabile di ambiente *var* con il valore *value*
  - Con il solo argomento *var*=, assegna alla variabile una stringa nulla
- `env`
  - mostra le variabili ed i rispettivi valori correnti

```
…
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
PWD=/home/grotto
LANG=en_US.UTF-8
SHLVL=1
HOME=/home/grotto
LOGNAME=grotto
SSH_CONNECTION=192.168.55.30 1297 192.168.55.33 22
…
```

- `unset var`
  - Elimina la variabile *var* dall'ambiente

Comandi Fondamentali

Documentazione

- `man [-s n] name`
  - fornisce una manualistica sui comandi
    - `-s n` Sezione dei manuali cui fare riferimento
    - *name* Nome del comando di cui mostrare il manuale
  - Differenti sezioni a seconda dell'argomento
    - 1: comandi utente
    - 2: chiamate di sistema
    - 3: funzioni di libreria standard
    - 8: comandi di sistema e di amministrazione
  - Per le funzioni la man page riporta header e librerie che forniscono dichiarazione ed implementazione
- `apropos key`
  - Mostra un elenco delle pagine che contengono la keyword *key*
- `whatis name`
  - Mostra un sommario del manuale del comando *name*

Comandi Fondamentali

Storico comandi

- `history`
  - Mostra la storia dei comandi eseguiti dal prompt della shell

```
…
513  export VAR=value
514  env
515  unset VAR
516  set -x
517  ls -la
518  alias
519  set +x
520  history
```

- `!{`***`string`***`|`***`num`***`}`
  - Ripete un comando precedentemente eseguito
    - ***`string`*** Esegue l'ultimo comando che inizia con ***`string`***
    - ***`num`*** Esegue il comando numero ***`num`*** nella history
  - `!!`
    - Ripete l'ultimo comando eseguito

Comandi Fondamentali

Alias

- `alias [name string]`
  - Definisce una nome alternativo per un comando
  - Senza argomenti mostra gli alias correntemente definiti
    - *name* Nome alternativo. Se il comando *name* esiste già, diviene inaccesibile fino a quando il comando alternetico è definito
    - *string* Definisce il comando da assegnare al nome alternativo

```
grotto@Gr8-laptop:~$ alias ll='ls -l'
grotto@Gr8-laptop:~$ ll
drwxr-xr-x 2 grotto grotto     4096 2008-10-06 10:49 Desktop
drwxr-xr-x 3 grotto grotto     4096 2008-10-01 15:54 Documents
grotto@Gr8-laptop:~$ alias hello='echo Hello $USER'
grotto@Gr8-laptop:~$ hello
Hello grotto
grotto@Gr8-laptop:~$ alias
alias hello='echo Hello $USER'
alias ll='ls -l'
grotto@Gr8-laptop:~$
```

Comandi Fondamentali

Alias

- `unalias` *`name`* `... | -a`
  - Rimuove le definizioni di nomi alternativi
    - `-a` Rimuove tutti i nomi alternativi correntemente definiti
    - *`name`…* Lista dei nomialternativi da eliminare

```
grotto@Gr8-laptop:~$ alias ls= ‘echo file not found'
grotto@Gr8-laptop:~$ ls
file not found
grotto@Gr8-laptop:~$ unalias ls
grotto@Gr8-laptop:~$ ls
Desktop         Documents        Music            Public
```

- `\`*`name`*
  - Rimuove temporaneamente la definizione del nome alternativo *`name`*

```
grotto@Gr8-laptop:~$ ls
file not found
grotto@Gr8-laptop:~$ \ls
Desktop         Documents        Music            Public
```

Comandi Fondamentali

Gestione del File System

- Il File System
  - Il *File System* fornisce all'utente una *visione logica* dei dispositivi di memorizzazione di massa, quali
    - Hard Disk,
    - CD ROM,
    - Floppy
    - DAT
  - Sistemi operativi diversi utilizzano diverse implementazioni di file system
  - La struttura di un file system si basa su due concetti fondamentali:
    - File
    - Directory

Comandi Fondamentali

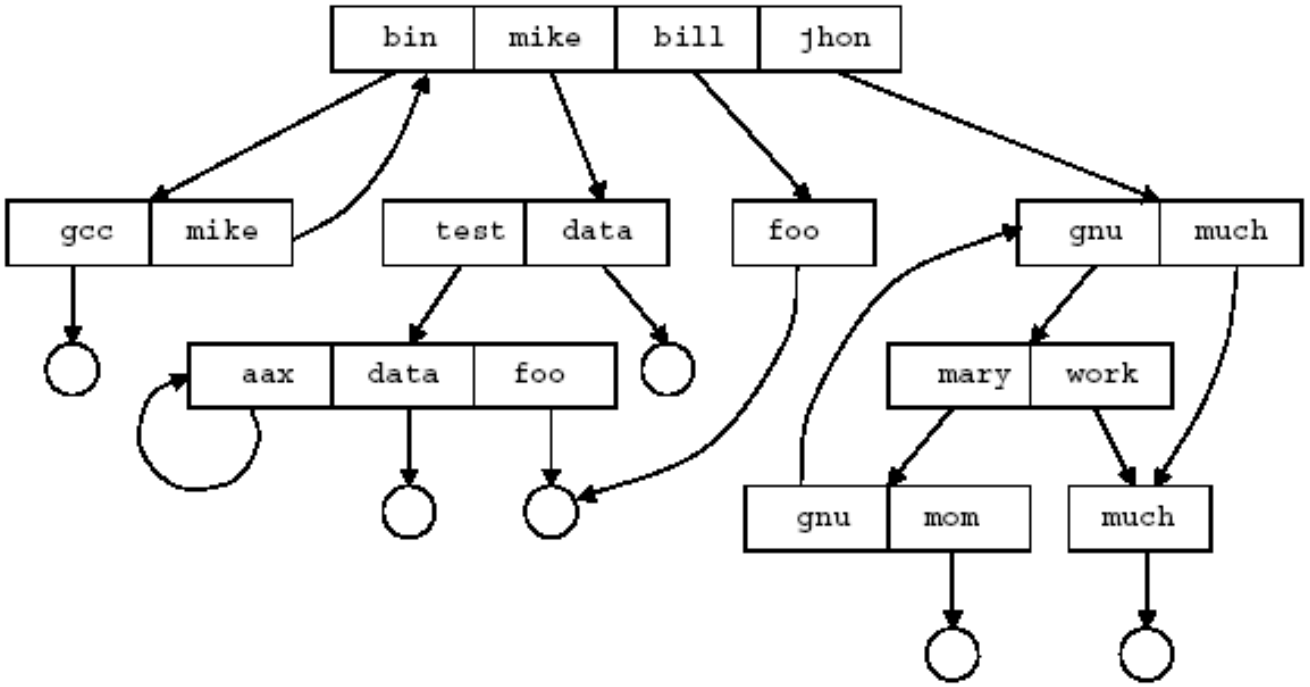
Gestione del File System

- File
  - Il concetto di *file* offre una visone omogenea delle informazioni memorizzate
  - La visone non dipende dal tipo di dispositivo fisico su cui le informazioni vengono memorizzate
- Directory
  - Esistono diverse strutture di directory:
    - Directory a singolo livello
    - Directory a due livelli
    - Directory ad albero
    - Directory a grafo aciclico
    - Directory a grafo generale
  - GNU/Linux adotta una struttura di directory a grafo generale:
    - Sono consentiti riferimenti circolari
    - Il sistema operativo è più complesso in quanto la ricerca o la creazione di un elenco dei file potrebbe generare loop infiniti
    - Una semplice soluzione consiste nel limitare la profondità o la lunghezza (numero di caratteri) del path name

Comandi Fondamentali

Gestione del File System

- Un esempio di directory a grafo generale è il seguente:



Comandi Fondamentali

Gestione del File System

- Protezione
  - Il file system GNU/Linux gestisce gli accessi ai file secondo lo schema detto *Simplified Access Control List*
  - Gli *utenti* sono identificati in base a
    - *Username:* Identificativo dell'utente
    - *Group:* Identificativo di gruppo,condiviso da più utenti
  - Gli utenti sono raggruppati in tre *classi*
    - *Owner:* Solo il proprietario del file
    - *Group:* Solo i membri del gruppo del proprietario del file
    - *Other:* Tutti gli utenti

Comandi Fondamentali

Gestione del File System

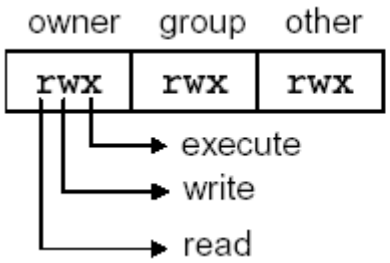
- Protezione
  - Le operazioni che un utente può effettuare su un file sono raggruppate in tre classi
    - Read: Lettura, copia
    - Write: Scrittura, modifica, eliminazione
    - Execute: Esecuzione
  - Le operazioni che un utente può effettuare su una directory sono raggruppate in tre classi
    - Read: Elenco dei file contenuti
    - Write: Creazione/rimozione dei file contenuti
    - Execute: Entrata nella directory, creazione di file nella directory, elenco dei file contenuti

Comandi Fondamentali

Gestione del File System

- `ls [ -alR ] [`*`path`*`]`
  - Mostra l'elenco dei file nella directory *`path`* o in quella corrente
    - `-a` Mostra anche i file invisibili, il cui nome inizia con un . (punto).
    - `-l` Mostra informazioni aggiuntive nella lista
    - `-R` Mostra il contenuto della directory path (o quella corrente) e tutte le sottodirectory.

```
grotto@Gr8-laptop:~$ ls -l
drwxr-xr-x 2 grotto grotto     4096 2008-10-06 10:49 Desktop
drwxr-xr-x 3 grotto grotto     4096 2008-10-01 15:54 Documents
grotto@Gr8-laptop:~$
```



owner	group	other
rwx	rwx	rwx
111	101	101
7	5	5

Comandi Fondamentali

Gestione del File System

- `pwd`
  - Mostra il path assoluto della directory corrente
- `cd` *`[path]`*
  - Cambia la directory corrente
  - Con un argomento, rende *`path`* la directory corrente
  - Senza argomenti, rende corrente la home directory dell'utente
- Esistono alcuni nomi speciali di directory:
  - `.` Directory corrente
  - `..` Directory padre della directory corrente
  - `/` Root directory
  - `~/` Home directory dell'utente
  - `~user` Home directory dell'utente `user`.
- Diversi livelli di directory sono separati dal carattere / (slash)
- I pathname che iniziano con / sono detti *assoluti* metre in tutti gli altri casi si parla di pathname *relativi*

Comandi Fondamentali

Gestione del File System

- **`cp [-r][-p][-i][-f] src dest`**
  - Copia uno o più file
    - **`-r`** Ricorsiva. Copia una directory e tutto il suo contenuto
    - **`-p`** Conserva gli attributi del file originale (proprietario, gruppo, diritti di accesso e data di creazione e accesso)
    - **`-i`** Interattivo. Chiede conferma prima di sovrascrivere un file
    - **`-f`** Forza la copia di uno o più file anche se la destinazione esiste
  - Sorgente e destinazione possono essere:
    - ***`file1 file2`*** Copia ***`file1`*** in ***`file2`***
    - ***`file... dest`*** Copia tutti i file nella directory ***`dest`***
    - ***`dir... dest`*** Copia le directoy nella directory ***`dest`*** (solo con l'opzione ***`-r`***)
- **`mkdir [-p] dir`**
  - Crea una nuova directory. La directory padre di ***`dir`*** deve esistere
  - **`-p`** Crea la nuova directory ***`dir`*** e tutte le directory intermedie che sono necessarie

Comandi Fondamentali

Gestione del File System

- `touch [-a|-m] [-r ref] [file...]`
  - Modifica le date di accesso o di modifica di uno o più file
  - Se un file non esiste viene creato
    - `-a` Cambia la data di accesso usando la data corrente
    - `-m` Cambia la data di amodifica usando la data corrente
  - `-r ref` Usa la data del file `ref` invece della data corrente
- `mv [-i|-f] src dest`
  - Sposta uno o più file. L'utente deve avere i diritti di scrittura sui file sorgenti, in quanto devono essere cancellati
    - `-i` Interattivo. Chiede conferma prima di sovrascrivere un file
    - `-f` Forza lo spostamento
  - Sorgente e destinazione possono essere:
    - `file1 file2` Sposta il `file1` in `file2` (anche directory)
    - `file... dest` Sposta i file nella directory `dest`
    - `dir... dest` Sposta le directory nella directory `dest`

Comandi Fondamentali

Gestione del File System

- `rm [-i|-f] [-r dir...] [file...]`
  - Rimuove uno o più file
    - `-i` Interattivo. Chiede conferma prima di eliminare un file
    - `-f` Forza la rimozione
    - `-r dir...` Elimina le directory ed il loro contenuto
- `rmdir [-p] dir...`
  - Rimuove una o più directory. Le directory devono essere vuote o, al più, contenere solo altre directory
    - `-p` Rimuove la directory `dir` e la sua directory padre

Comandi Fondamentali

Gestione del File System

- `du [-k][-s][-a] dir...`
  - Mostra lo stato di utilizzo dei dischi
    - `-k` Mostra le informazioni in kilobytes
    - `-s` Mostra solo il totale
    - `-a` Mostra le informazioni per tutti i file
- `df [-b] [-l] [dir|device]`
  - Mostra lo spazio disponibile sui vari file system
    - `-b` Lo spazio è espresso in kilobytes
    - `-l` Mostra i dati relativi ai soli file system locali

Comandi Fondamentali

Gestione dei Processi

- Programma
  - Operazioni che devono essere svolte da un calcolatore
- Processo
  - È una istanza di un programma in esecuzione
- Nel sistema operativo GNU/Linux
  - Tutti i processi discendono dal processo `init`
- I processi possono essere eseguiti in due modalità
  - *Foreground* Il processo figlio eredita e ritiene tutte le risorse del padre. Il padre è sospeso in attesa della terminazione del figlio
  - *Backgroud* Il processo figlio eredita tutte le risorse del padre. Il processo padre rimane in esecuzione. È lo scheduler ad assegnare le risorse ai due processi
  - L'utente può specificare la modalità di esecuzione di un processo
  - La modalità di esecuzione può cambiare durante la vita di un processo

Comandi Fondamentali

Gestione dei Processi

- Su un processo si possono eseguire alcune operazioni
  - Creazione (create)
    - Un processo viene creato in foreground fornendo ad una shell il nome del programma corrispondente
    - Per creare un processo in background si usa il carattere **&** alla fine della linea di comando della shell
  - Terminazione (release)
    - Un processo in foreground viene terminato premendo opportune combinazioni di tasti, tipicamente `CTRL-C`
    - Un processo in background viene terminato inviandogli un opportuno segnale (`SIGKILL`, `SIGABRT`) tramite il comando `kill`
  - Sospensione (suspend)
    - Un processo in foreground viene sospeso premendo opportune combinazioni di tasti, tipicamente `CTRL-Z`
    - Un processo in background viene sospeso inviandogli un opportuno segnale (`SIGSTOP`) tramite i comandi `kill` e `stop`

Comandi Fondamentali

Gestione dei Processi

- Continuazione (resume)
  - Si può forzare la continuazione di un processo sospeso utilizzando i comandi `fg` e `bg`
  - È anche possibile inviare al processo sospeso un opportuno segnale (`SIGCONT`) tramite il comando `kill`
- Duplicazione
  - Un processo può essere duplicato solo attraverso chiamate a funzioni di sistema operativo
- Analisi
  - È possibile visualizzare alcune informazioni relative ai process i utilizzando i comandi `ps` e `jobs`

Comandi Fondamentali

Gestione dei Processi

- `program [&]`
  - Crea un nuovo processo eseguendo una immagine di **`program`**
  - Normalmente il processo viene eseguito in foreground
    - `&` Crea il nuovo processo e lo esegue in background
- `ps [-a][-u][-x]`
  - Mostra le informazioni più utili riguardanti i processi in esecuzione
    - `-a` Mostra tutte le informazioni
    - `-u` Mostra il nome del proprietario dei processi
    - `-x` Mostra i processi di tutti gli utenti, non solo i propri

Comandi Fondamentali

Gestione dei Processi

- `jobs [-l]`
  - Mostra la lista dei processi sospesi, figli della shell corrente
  - Ogni processo è individuato da un identificatore detto ***`jobid`***
    - `-l` Include nella lista anche il process id
- `fg [%`***`jobid`***`]`
  - Porta un processo in foreground
    - Senza argomenti porta in foreground il processo indicato con il simbolo `'+'` nella lista prodotta dal comando `jobs`
    - `%`***`jobid`*** Porta in foreground il processo identificato dall'identificatore ***`jobid`***, secondo quanto mostrato dal comando `jobs`
- `bg [%`***`jobid`***`]`
  - Porta un processo in background
    - Senza argomenti porta in background il processo indicato con il simbolo `'+'` nella lista prodotta dal comando `jobs`
    - `%`***`jobid`*** Porta in background il processo identificato dall'identificatore ***`jobid`***, secondo quanto mostrato dal comando `jobs`

Comandi Fondamentali

Gestione dei Processi

- `kill {-l|[-signal] {pid|%jobid}}`
  - Invia un segnale ad un processo, per default il segnale `SIGKILL`
    - `-l` Mostra la lista dei nomi dei segnali disponibili
    - `-signal` Invia il seganle *`signal`* al processo

- `time command [args]`
  - Esegue il comando command con eventuali argomenti e mostra i tempi consumati: user time, system time e real time
  - Alcune implementazioni di questo comando mostrano anche la percentuale di utilizzo medio della CPU
  - Il formato di uscita dipende dalla implementazione specifica e dalla shell in uso

Ambiente di sviluppo

Il compilatore Gcc

- Esistono molti compilatori
  - Commerciali
  - Freeware o GPL
- Gli strumenti di compilazione hanno raggiunto in più di 30 anni un ottimo livello di maturità
- L'ambiente GNU gcc
  - È gratuito e open-source sotto GPL
  - Supporta alcune decine di processori
- Lavora in ambienti:
  - UNIX: Solaris, HPUX, AIX, Linux, FreeBSD, SystemV
  - Windows: NT, 2000, XP
  - Mac: OS-X

Ambiente di sviluppo

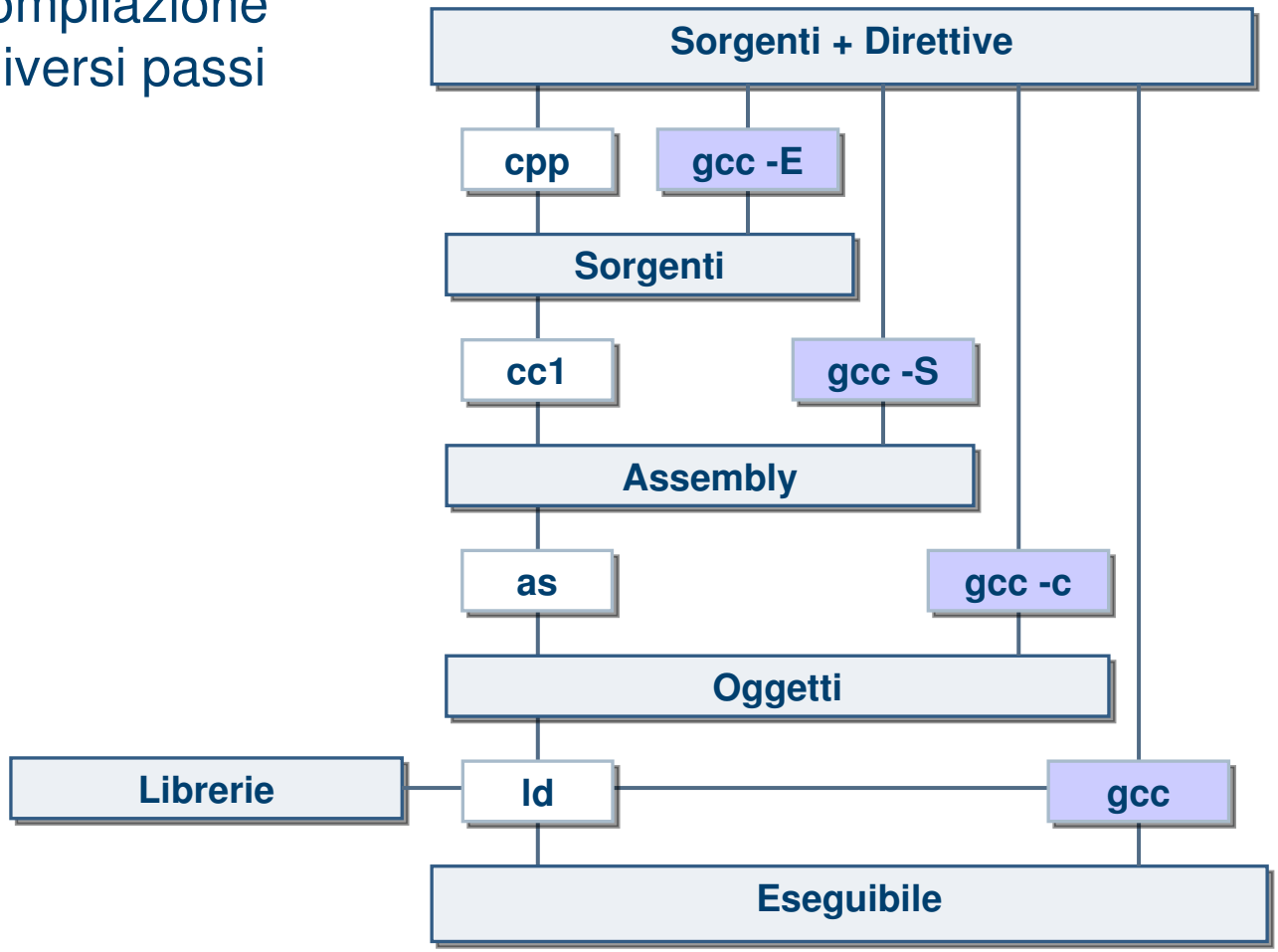
Il compilatore Gcc

- È un ambiente a linea di comando
- Comprende:
  - Preprocessore
  - Compilatore
  - Linker
  - Assembler
  - Librerie standard
- Supporta:
  - C, C++, ObjectiveC
  - Fortran
  - Pascal
  - Java
  - ...

Ambiente di sviluppo

Il compilatore gcc

- Il processo di compilazione si compone di diversi passi
- Preprocessore
  - **gcc -E**
  - **cpp**
- Compilazione
  - **gcc –S**
  - **cc1**
- Assemblaggio
  - **gcc –c**
  - **as**
- Linking
  - **gcc**
  - **ld**



Ambiente di sviluppo

Il compilatore gcc

- `gcc [-c|-S|-E] [-g] [-On] [-Idir] [-Ldir] src [-o exec]`
- Opzioni
  - `-c` Compilazione, assemblaggio senza linking, produce file oggetto con estensione `.o`
  - `-S` Solo compilazione, produce file assembly con estensione `.s`
  - `-E` Solo preprocessing
  - `-g` Genera informazioni per il debugging
  - `-On` Livello di ottimizzazione, da `-O0` (bassa) a `-O3` (alta)
  - `-I` Indica le directory dove cercare i file header
  - `-L` Indica le directory dove cercare le librerie
  - `-l` Indica le librerie da considerare per il linking
  - `src` Elenco di file sorgenti
  - `-o exec` Specifica il nome del file eseguibile prodotto. Se omesso, il file eseguibile è `a.out`

Ambiente di sviluppo

Il compilatore gcc

- `gcc -E main.c > main_all.c`
  - Viene rediretto in `main_all.c` il codice sorgente dopo il preprocessing
- `gcc -S main.c`
  - Il risultato è `main.s` (codice assembly)
- `gcc -c main.c`
  - Il risultato è `main.o` (codice oggetto)
- `gcc -c reciprocal.cpp`
  - Il risultato è `reciprocal.o` (codice oggetto)
- `gcc -o reciprocal main.o reciprocal.o`
  - Viene effettuato il linking

Esempio

Ambiente di sviluppo

Automatizzazione: make

- Idea di base molto semplice:
  - Voglio indicare cosa voglio costruire e come
  - Voglio inoltre indicare delle dipendenze
- Soluzione: Makefile
  - Vengono definiti dei target che identificano cosa voglio costruire
  - Ad ogni target è associata l'azione specifica
- Tre target nel nostro caso:
  - `main.o, reciprocal.o, reciprocal`
- Target speciale: `clean`
  - Rimuove tutti i file generati

Esempio

Ambiente di sviluppo

Debug

- È necessario ricompilare aggiungendo le informazioni per il debug
- `make CFLAGS=-g`
- `gcc` include informazioni aggiuntive nei file oggetto e nell'eseguibile
- `gdb` usa queste informazioni per ricordurre le operazioni specifiche alla linea di codice corrispondente
- `gdb reciprocal`
- Azioni principali: `run, where, up, break, print`

Esempio