

# CORSO BASE UNIX

**Bernardo Damele**

`<bernardo.damele@gmail.com>`

# 1 – Concetti generali

- Le componenti di un computer si possono dividere in due categorie
  - *Hardware* – L'insieme dei componenti fisici del computer, cioè tutto quello che si vede e si tocca
  - *Software* – L'insieme dei programmi necessari per sfruttare al meglio le potenzialità dell'hardware
    - **Software di base** – Sistema operativo, un insieme di programmi che consente di gestire tutte le risorse indispensabili del computer rendendone possibile l'utilizzo
    - **Software applicativo** – Applicazioni sviluppate per assolvere ad esigenze specifiche dell'utente (scrittura, programmazione, sicurezza dati, multimedia...)

## 2 – Sistemi operativi

- Alcuni dei principali sistemi operativi sono
  - Unix
  - GNU/Linux
  - Mac OS
  - MS/DOS
  - MS Windows

# 3 – Unix

- **Unix** è un sistema operativo multiuser e multitasking
  - *Multiuser* – si possono avere più utenti collegati al sistema contemporaneamente
  - *Multitasking* – permette ad ogni utente di lavorare in contemporanea su più programmi e la CPU li gestisce
  
- E' stato appositamente progettato per essere indipendente dall'hardware; è quindi facilmente portabile essendo scritto nel linguaggio ad alto livello C
  
- Nato con un'interfaccia a comandi (*comand-line*) oggi dispone di molteplici interfacce grafiche (KDE, Gnome, FluxBox, WindowMaker, Afterstep...)

# 4 – Versioni di Unix

- Alcuni tra i sistemi operativi Unix derivati sono
  - *SCO UNIX*
    - Implementazione di System V.3.2.5
    - Funziona sui PC
  - *Sun OS*
    - Basato sui sistemi operativi BSD
    - Innovativo grazie alla prima implementazione di *NFS*
  - *AIX*
    - Il primo sistema operativo basato su Unix della IBM
  - *Linux*
    - Basato su Unix è portabile, funziona su ogni piattaforma
    - Linus Torvalds iniziò a scrivere il kernel Linux da solo, ora la comunità di sviluppatori conta oltre 1000 programmatori in giro per il mondo

# 5 – Storia di Unix

- 1965 – AT&T, M.I.T. e General Electrics
  - Questi tre enti si uniscono nel progetto di un sistema operativo che sia multiuser, multitasking e multiprocessore, *Multics*
  
- 1969 – Il progetto Multics “fallisce”
  - Presso i Bell Laboratories della AT&T nasce UNIX
  - Sviluppato da Ken Thompson e Dennis Ritchie per un PDP-7
  - Incorpora alcune caratteristiche e supera i limiti di sistemi operativi preesistenti supportando, in modo efficiente, multiutenza e multitasking
  - Il nome UNIX (inizialmente Unics) è scelto da Kernighan in opposizione a MULTICS

# 5 – Storia di Unix

- 1973 – Unix e il linguaggio ad alto livello C
  - La terza versione (V3) di Unix viene scritta nel linguaggio di programmazione ad alto livello C, linguaggio sviluppato ai Bell Labs da Dennis Ritchie proprio per supportare Unix
  
- Metà anni '70 – AT&T rende Unix largamente disponibile, offrendolo alle Università a basso costo e distribuendo i sorgenti di varie versioni
  
- Fine anni '70 – Alcuni professori in Australia insegnano UNIX partendo dal codice sorgente sviluppato presso gli AT&T Lab
  
- 1979 – L'ultimo vero Unix (V7)

# 5 – Storia di Unix

- 1984 – Berkeley Software Distribution
  - UNIX è ampiamente utilizzato nel settore della ricerca
  - Presso l'Università di Berkeley (California) nasce la *Berkeley Software Distribution* (BSD)
    - TCP/IP e i socket per la programmazione a livello rete
    - Versione 4.3BSD, sviluppata originariamente per il VAX, è una delle versioni più influenti. Verrà “portata” su molte altre piattaforme
  - I sorgenti di BSD sono disponibili a tutti
  
- 1984 – System V
  - Molte caratteristiche della BSD vengono incorporate nella nuova versione di AT&T, la *System V*

# 5 – Storia di Unix

- 1983 – Richard Stallman (M.I.T.)
  - Stallman ricerca un modo per preservare la libertà
    - Portabilità
    - *GPL* (GNU General Public Licence) – Licenza che impone che i codici sorgenti siano sempre proprietà della comunità di sviluppatori che con gli anni si è creata attorno al software libero
  - Inizia quindi una reimplementazione “free” di Unix, detta GNU (acronimo ricorsivo di “GNU is Not Unix”)
  - Il progetto GNU darà vita a tool quali `emacs`, `gcc`, `gdb` e molti altri

# 5 – Storia di Unix

- 1991 – L'Università di Berkeley rilascia una versione “free” di Unix, rimuovendo il codice proprietario rimanente della AT&T
  - Progetti volontari (FreeBSD, NetBSD e OpenBSD) continueranno poi il suo sviluppo
  
- 1991 – Linus Torvalds, uno studente ad Helsinki
  - Minix: un sistema operativo
  - Mancanze in Minix
  - Torvalds inizia a scrivere il **kernel Linux** in C con alcuni amici
  - Lo pubblica sotto licenza GPL
  - Arricchito con i tools del progetto GNU esistenti ed altri sviluppati da volontari, diviene un sostituto completo di Unix

# 6 – GNU/Linux

- *GNU* – Insieme di programmi del progetto iniziato nel 1983 da R. Stallman
- *Linux* – Kernel (cuore del sistema operativo) sviluppato da L. Torvalds ed altri
- Col tempo dall'unione del *progetto GNU* e del *kernel Linux* sono nate varie distribuzioni **GNU/Linux** ognuna delle quali ha i propri pro e contro, ha un proprio sistema di gestione dei programmi e una serie di utilities personalizzabili
  - Debian GNU/Linux <<http://www.debian.org>>
  - Slackware Linux <<http://www.slackware.com>>
  - Gentoo Linux <<http://www.gentoo.org>>
  - Mandrake Linux <<http://www.mandrakesoft.com>>
  - SuSE Linux <<http://www.suse.com>>
  - Red Hat Linux <<http://www.redhat.com>>

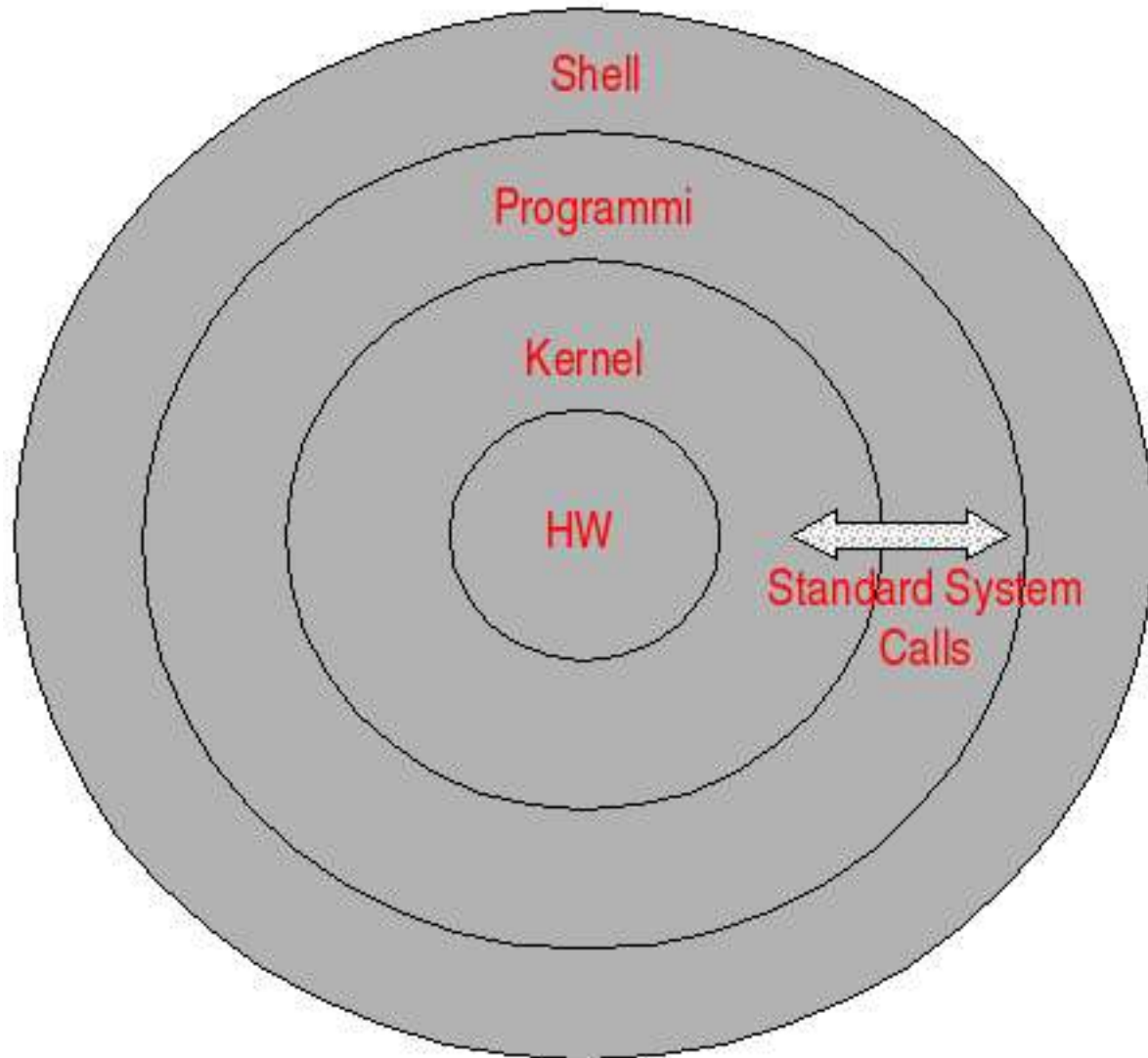
# 7 – Caratteristiche principali di GNU/Linux

- Di seguito elencate alcune delle caratteristiche principali di GNU/Linux
  - E' un sistema operativo multitasking time-sharing con una struttura a livelli
  - Supporta processi multipli (*scheduling dei processi con priorità*)
  - E' un sistema operativo multiuser
  - Ha un file system con directory strutturate ad albero
  - Ha un interfaccia testuale (*command-line*), detta anche **shell** semplice, intuibile ed estremamente potente
  - Supporto per automatizzare attività di routine
  - La licenza del software è libera. Nella maggior parte dei casi è la GPL

## 7.1 – Struttura a livelli

- ❑ *Kernel* – Nucleo centrale, cuore del sistema operativo. Fornisce le funzionalità di base per gestire le risorse del sistema: scheduling dei processi, file system, gestione della memoria...
- ❑ *System call* – Facilitano la programmazione permettendo di accedere a funzionalità del kernel (manipolazione di file, gestione dei processi concorrenti e comunicazione tra processi...)
- ❑ *Shell* – Interprete dei comandi, è l'interfaccia tra l'utente e il kernel
- ❑ *Utilities* – Programmi di utilità (editor, operazioni sui files, stampa, supporto alla programmazione...)

# 7.1 – Struttura a livelli



## 7.2 – Multitasking time-sharing

- GNU/Linux, Unix in generale, è un sistema operativo **multitasking time-sharing**
  - I processi sono basati sul modello *fork / exec*
  - Ogni processo ha un'*identità* e un *possessore*, informazioni utilizzate per protezione e scheduling
  - La CPU è gestita in *time-sharing*, suddividendo il tempo di CPU in quanti di tempo (*slice*) assegnati ai processi in base a priorità e comportamento passato
  - La gestione della memoria è basata sulla *paginazione*
    - Modello con varianti quali *buddy system*

## 7.2.1 – Processi

- Ad ogni processo è associato un *PID* (Process ID) che lo identifica univocamente dagli altri
  - Un processo è un programma in esecuzione nel suo spazio virtuale
  - Ogni processo è generato da un altro processo detto *processo padre*
  - Ci sono tre tipi di processi: *interattivi*, *batch* e *demoni*
  - Alcuni degli attributi di un processo sono
    - *PID*: Process ID
    - *PPID*: Parent process ID
    - *TTY*: La shell sulla quale è in esecuzione il processo
    - *STAT*: Lo stato in cui si trova il processo
    - *USER*: L'utente che ha mandato in esecuzione il processo
    - *START*: La data di inizio dell'esecuzione del processo
  - Per maggiori dettagli vedere le pagine di manuale del comando `ps`.

## 7.2.1 – Processi

- Per terminare forzatamente un processo si utilizza il comando `kill`. La sintassi è `kill [-SEGNALE] <pid>` e manda un segnale al processo identificato dal `pid`. L'uso più comune è quello di mandare il segnale `TERM` per terminare il processo o `KILL` per “uccidere” il processo e i suoi eventuali figli
  - Per la lista dei possibili segnali si veda con il comando `kill -l`
- Controllo dei processi
  - `bg` – Riprende in background l'esecuzione di un processo sospeso
  - `fg` – Riprende in foreground l'esecuzione di un processo sospeso o attivo in background
  - `jobs` – Elenca i processi attivi in background e quelli sospesi

## 7.3 – Multiuser

- GNU/Linux, Unix in generale, è un sistema operativo **multiuser**
  - Ogni utente ha la sua utenza nel file `/etc/passwd` e la sua password, in formato criptato, nel file `/etc/shadow`
  - Gli attributi di ogni utente sono il *login name*, la *password*, il *group name*, l'*user ID* (UID), il *group ID* (GID), la *home directory* e la *shell*
    - Esempio

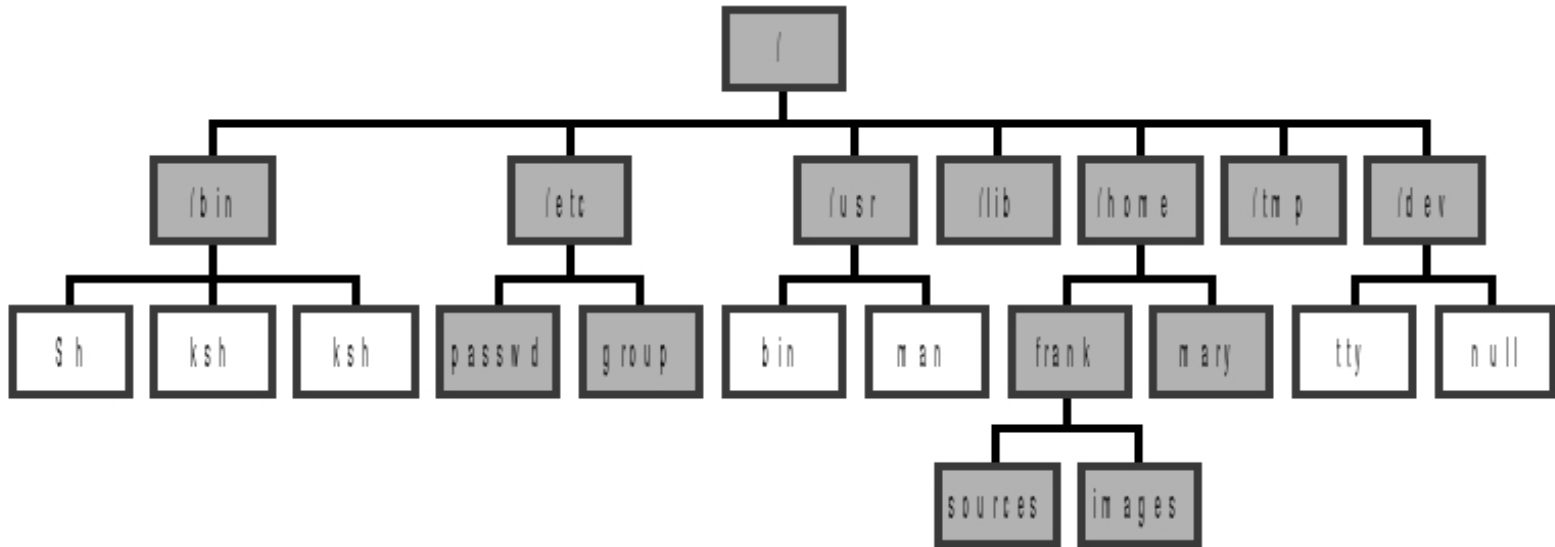
```
$ head -1 /etc/passwd  
root:x:0:0:root:/root:/bin/bash
```

## 7.4 – File system

- ❑ Il file system fornisce una *visione logica della memorizzazione e dell'organizzazione dei file sul disco*
- ❑ Il file system è estremamente importante perché Unix è un *sistema orientato ai file* nel senso che molte operazioni vengono effettuate tramite essi (accesso a periferiche e comunicazione tra processi)
- ❑ Un nome (*file*) nel file system può riferirsi a
  - File dati (che può anche essere una directory)
  - Dispositivi di input/output (disco, nastro, stampante, floppy, mouse...)
  - Memoria interna
  - Informazioni sul sistema hardware e software (contenute in `/proc` e `/sys`)

# 7.4 – File system

- Un file può essere visto come una sequenza di byte, indipendentemente dal tipo di file
- Una directory può essere vista come un contenitore di file o di directory, si ha quindi una struttura gerarchica ad albero (lasciando da parte i link file)



## 7.4.1 – Organizzazione del file system

- Per tradizione esistono alcune directory che nel file system assumono un ruolo specifico
  - / – Radice, root directory
  - /bin – Programmi eseguibili da tutti gli utenti. Alcuni sono links a files che sono sotto nella directory /usr/bin
  - /boot – Contiene solitamente l'immagine eseguibile del kernel
  - /dev – Device directory
  - /etc – File di configurazione, tra cui i boot scripts
  - /home – Home directory degli utenti
  - /mnt – Directory convenzionale dove montare i devices esterni (cd, dvd...)
  - /proc – Il sistema qua salva su file di testo molte informazioni del sistema
  - /sbin – Programmi eseguibili di sistema
  - /usr – File di sistema, librerie...

## 7.4.2 – Navigare nel file system

- Ci si può spostare all'interno del file system e quindi modificare la directory di lavoro corrente (*working directory*) con il comando `cd`

- `cd [directory]`

L'argomento `directory` è opzionale. Se non viene specificato, il comando porta nella propria home directory (`/home/username`)

- In ogni nuova directory creata vengono automaticamente creati due file
  - `.` – Si riferisce alla directory stessa
  - `..` – Si riferisce alla directory padre
- I file di configurazione personali iniziano per punto (`.`) e sono dei file “nascosti”

## 7.4.3 – Pathname

- Il **pathname** specifica la locazione di un file all'interno di un file system
  - Il *filename* è una sequenza di caratteri tranne lo slash (/)
  - Il *pathname* è una sequenza di filename separati ognuno da uno slash (/)
    - *Assoluto* – Dice come raggiungere un file partendo dalla radice del file system, inizia sempre con slash (/)
    - *Relativo* – Dice come raggiungere un file a partire dalla directory di lavoro corrente o dalla home directory di un utente
  - Ogni processo ha una directory di lavoro corrente rispetto a cui vengono risolti i pathname relativi

## 7.4.4 – Files

- ❑ Sotto Unix ogni cosa è identificata da un file. I device sono files, i processi sono files, le directory sono file
  
- ❑ Ogni file è identificato da quattro informazioni principali
  - *Time stamp* – Ad ogni file sono associate tre date
    - Cambiamento attributi
    - Data di ultima modifica
    - Data di ultimo accesso
  
  - *User (proprietario)* – Ogni file sotto Unix è di proprietà di un utente del sistema
  
  - *Group (gruppo)* – Ogni file è associato ad un gruppo di utenti. Quello più comune per i file utente è chiamato `users`, ed è generalmente condiviso da tutti gli utenti del sistema
  
  - *Permessi* – Ad ogni file sono associati dei permessi di accesso (chiamati anche *privilegi*) che specificano chi (**u**ser, **g**roup, **o**thers) e in che modo (**r**ead, **w**rite, **e**xecute) può accedere al file

## 7.4.4 – Files

- ❑ Le informazioni sul file possono essere visualizzate tramite il comando `ls -l`
  - Esempio

```
$ ls -l /bin/bash
-rwxr-xr-x 1 root root 739800 Jan 23 00:43 /bin/bash
```
- ❑ Ogni linea, tra le altre informazioni, include
  - Il tipo del file (primo carattere)
  - I permessi di accesso al file (caratteri 2-10)
    - 2-4: **user**
    - 5-7: **group**
    - 8-10: **others**
  - Il proprietario (user) del file
  - Il gruppo (group) del file

## 7.4.4 – Files

- I permessi di accesso ad un file sono sostanzialmente quattro
  - **r** = permesso di lettura (*read*)
  - **w** = permesso di scrittura (*write*)
  - **x** = permesso di esecuzione (*execute*)
  - **–** = mancanza di permesso
  
- I permessi di accesso ad una directory sono quattro
  - **r** = listare il contenuto della dir (*read*)
  - **w** = creare e cancellare file nella dir (*write*)
  - **x** = spostarsi all'interno della dir (*execute*)
  - **–** = mancanza di permesso

## 7.4.4 – Files

- Per cambiare i permessi di accesso di un file o di una directory si usa il comando `chmod`

- Sintassi

```
$ chmod [chi]<operazione><permessi> nomefile
```

<b>Tabella</b>		
chi	operazione	permessi
u = proprietario	+ aggiunge un permesso	r = lettura
g = gruppo	- elimina un permesso	w = scrittura
o = altri	= imposta un permesso	x = esecuzione
a = tutti		

- Esempio

```
$ chmod o-rx /bin/bash
```

```
$ ls -l /bin/bash
```

```
-rwxr-x--- 1 root root 739800 Jan 23 00:43 /bin/bash
```

## 7.4.4 – Files

- I permessi di accesso `rxw` possono essere rappresentati come un numero ottale
  - Il permesso `r` è rappresentato dal numero 4, `w` da 2 e `x` da 1
  - Per ciascuna categoria di utenti (`user`, `group`, `others`) questi valori possono essere sommati
  - Schematizzando

0 = ---

1 = --x

2 = -w-

3 = -wx (2+1)

4 = r--

5 = r-x (4+1)

6 = rw- (4+2)

7 = rwx (4+2+1)

- I permessi di accesso possono allora essere espressi con tre cifre ottali
  - Esempi

		user	group	others
<code>chmod 640</code>	<code>nomefile</code>	<code>rw-</code>	<code>r--</code>	<code>---</code>
<code>chmod 754</code>	<code>nomefile</code>	<code>rwx</code>	<code>r-x</code>	<code>r--</code>
<code>chmod 664</code>	<code>nomefile</code>	<code>rw-</code>	<code>rw-</code>	<code>r--</code>

## 7.4.4 – Files

- Per cambiare il proprietario (*user*) di un file si usa il comando `chown`

- Esempio

```
$ chown pippo /bin/bash
```

```
$ ls -l /bin/bash
```

```
-rwxr-x--- 1 pippo root 739800 Jan 23 00:43 /bin/bash
```

Nella maggior parte dei sistemi Unix per modificare il proprietario di un file bisogna essere un super-user (es. l'amministratore del sistema, root)

- Il proprietario di un file può cambiare il gruppo (*group*) di un file con un qualsiasi altro gruppo a cui l'utente stesso appartenga, con il comando `chgrp`

- Esempio

```
$ chgrp users /bin/bash
```

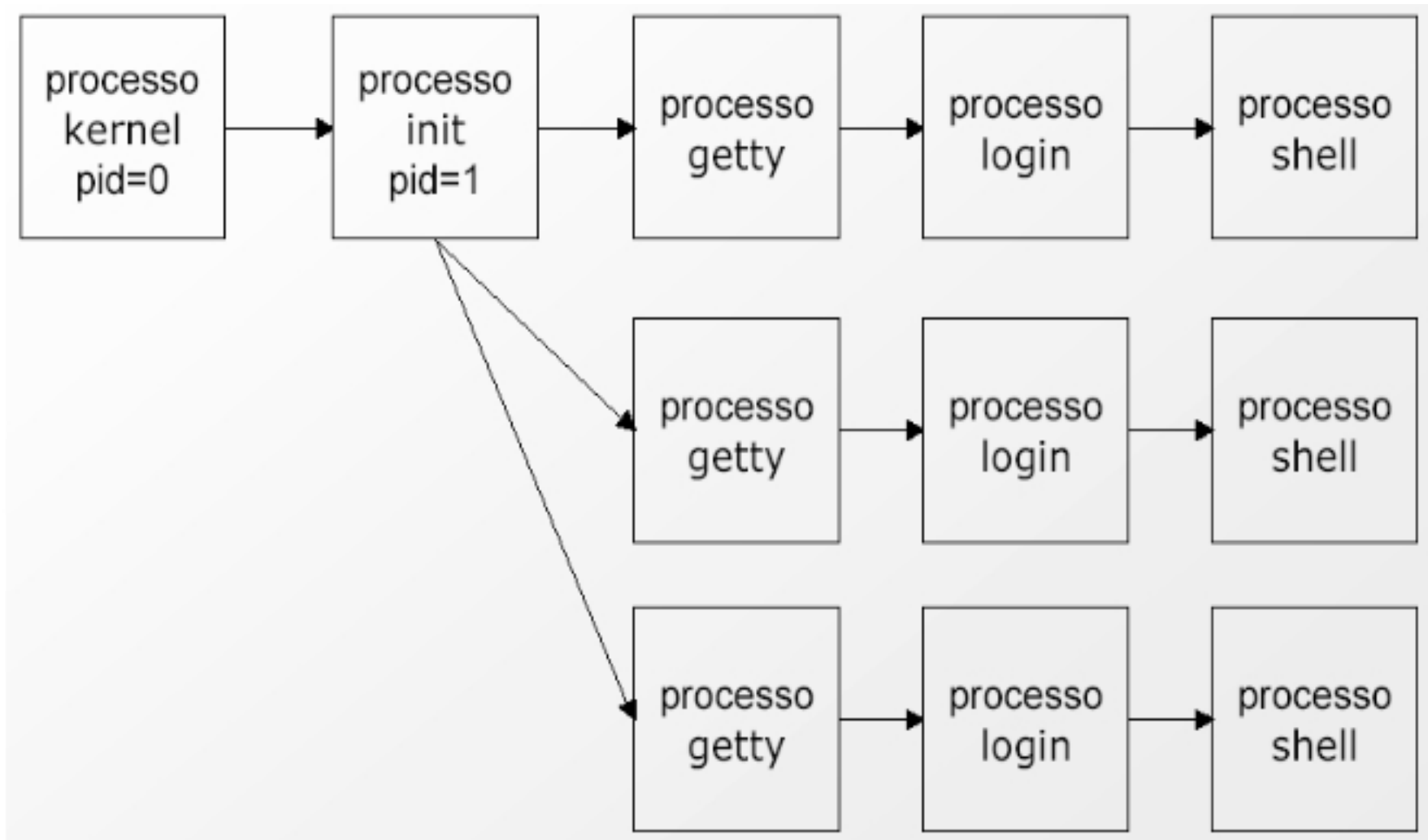
```
$ ls -l /bin/bash
```

```
-rwxr-x--- 1 pippo users 739800 Jan 23 00:43 /bin/bash
```

# 8 – Avvio di Unix

- Le fasi che un sistema Unix esegue all'avvio del computer sono
  - Caricamento e inizializzazione del kernel
  - Localizzazione e configurazione dei devices
  - Creazione dei processi spontanei di sistema – *init*
  - Esecuzione degli script di sistema – *boot scripts*
  - Operazioni in modalità multiuser

# 8 – Avvio di Unix



# 9 – Documentazione

- Ci sono due tipi di documentazione
  - Pagine di manuale (man)
    - Comandi
    - Controlli del flusso
  - Altra documentazione
    - Documentazione cartacea (libri, riviste...)
    - Documentazione online <[www.tldp.org](http://www.tldp.org), [pluto.linux.it](http://pluto.linux.it), [www.gutenberg.org](http://www.gutenberg.org), [www.google.com](http://www.google.com)...>
    - Supporti ottici (CDROM e DVD) e documentazione sul disco (/usr/doc)
    - RFC (Request For Comments) per i protocolli <[www.rfc-editor.org](http://www.rfc-editor.org)>

# 9 – Documentazione

- ❑ Le pagine di manuale
  - Si trovano sotto la directory `/usr/man/man# 0 /usr/share/man/man#`
  - Sono scritte in formato *troff*, *SGML*...
  - ...e compresse nel formato *gunzip* (*gzip*)
  
- ❑ Per leggere le pagine di manuale si usa il comando `man`
  - Esempio  
\$ `man ls`
  
- ❑ Per uscire da una pagina di manuale si usa la lettera `q`
  
- ❑ Il carattere `#` è da sostituirsi ad un numero come spiegato di seguito

# 9 – Documentazione

- Le pagine di manuale sono divise secondo il seguente schema

Solaris/HP-UX	Linux	Contenuto
1	1	User-level commands and applications
2	2	System calls and kernel error codes
3	3	Library calls
4	5	Standard file formats
5	7	Miscellaneous files and documents
6	6	Games and demonstrations
7	4	Device drivers and network protocols
1m	8	System administration commands
9	9	Obscure kernel specs and interfaces