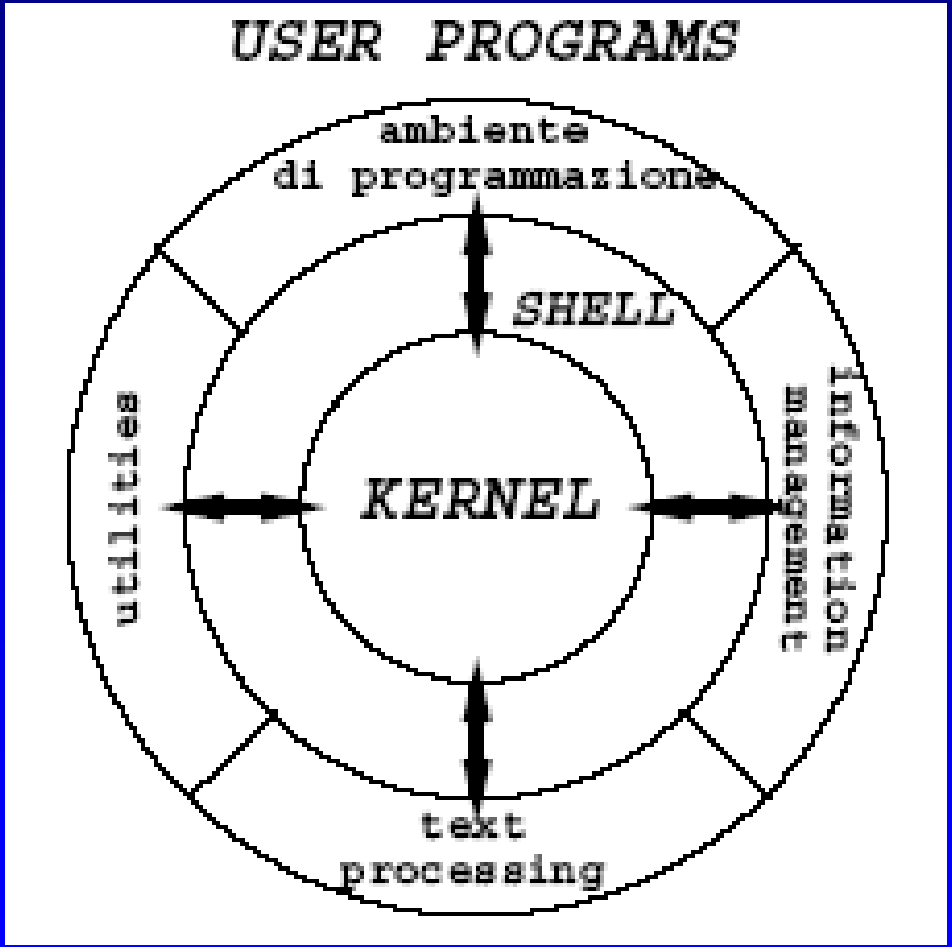
Shell di UNIX

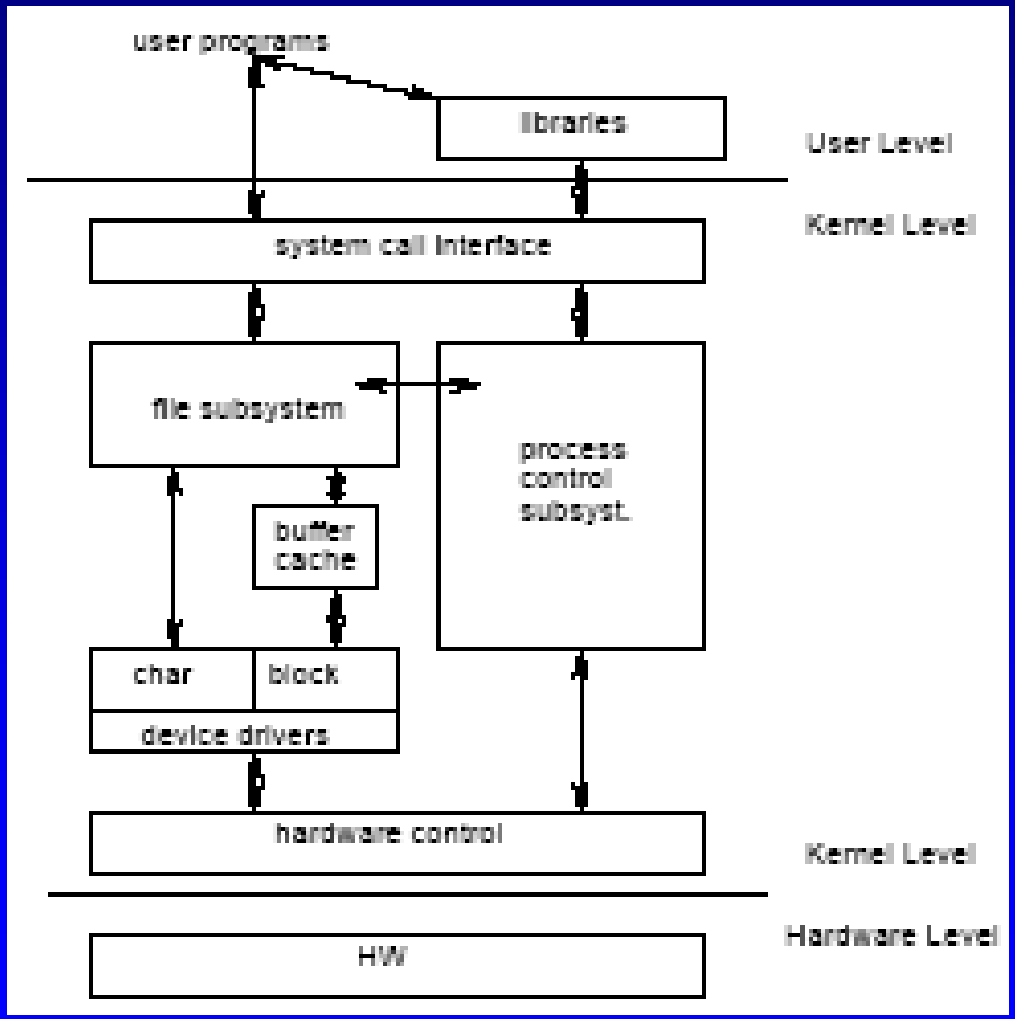
Caratteristiche UNIX

- Caratteristiche principali
  - Multitasking & multiutente
  - Ottima integrazione in rete
  - Interfaccia utente modificabile
  - Modularità
  - File system gerarchico
  - Vari strumenti di ausilio alla programmazione

La struttura



La struttura



Evoluzione

anno	UCB	Bell Labs	AT&T	MS	note
1969		PDP-7			K.Thompson
1971		PDP-11/20			B-language
1973		Ritchie			C-language
1976		V6			licenza
1977			PWB		AT&T internal
1978		V7			portabile
1979		32V			VAX
1979	BSD3				B.Joy
1980	BSD4.1				
1982			System III		in vendita
1983	BSD4.2	V8	System V		Ethernet
1984			SVR2	Xenix	PC
1986	BSD4.3	V9	SVR3		RFS, streams
1988	Tahoe		SVR3.2		80386
1989			SVR4		
1991					USL
1992					UNIVEL

BSD Berkeley Software Distribution

SVR System V Release

USL UNIX System Labs

I dialetti

- **UNIX** è il nome di una famiglia di sistemi operativi, con diverse implementazioni per le varie architetture HW

Nome	*Produttore*
AIX	IBM
A/UX	Apple
BSD	Univ.California Berkeley
HP-UX	Hewlett-Packard
Linux	public-domain
OSF/1	DEC
SCO Unix	Santa Cruz Operation
SunOS	SUN
Solaris	SUN
Ultrix	DEC
System V	vari

Standardizzazione

- Dalla fine degli anni 80 ci sono stati numerosi sforzi per “standardizzare” UNIX
- L’obiettivo è la portabilità delle applicazioni a livello sorgente
  - Programmi C
  - Script di shell
  - Programmi in altri linguaggi
- La competizione dei vari costruttori per il controllo dello Unix “Standard” ha creato una situazione piuttosto complessa

Standardizzazione

- Standard principali
  - POSIX (IEEE dal 1988, poi ISO) “Portable Operating System Interface for Unix”
  - XGP (X/Open, dal 1989) “X/Open Portability Guide”
  - SVID (AT&T, 1989) “System V Interface Definition”
  - OSF (Open Software Foundation)

La filosofia UNIX

- UNIX è più che una famiglia di sistemi operativi
  - Un insieme di programmi
  - Una filosofia basata su di essi
- Scopo di questa parte del corso è fornire una introduzione a questa filosofia
- Per una dettagliata descrizione dei comandi si rimanda ai manuali

I Comandi di base

Una sessione di lavoro

- Inizio di una sessione
  - **`Login:`**
  - **`Passowd:`**
- Fine di una sessione
  - CTRL-d, exit, logout (dipende dall'interprete dei comandi)
- NOTA: i caratteri maiuscoli sono diversi dai minuscoli!

I comandi in UNIX

- Sintassi, in generale, di un comando UNIX

  `Comando [-opzioni] argomenti`

- I comandi troppo lunghi possono essere continuati sulla riga successiva battendo **“\”** come ultimo carattere della riga
- Si possono dare più comandi sulla stessa riga separandoli con **“;”** (saranno eseguiti in sequenza)

  `comando1 ; comando2 ; ...`

Il comando `ls`

- Per visualizzare il contenuto di una directory

  `ls [-opzioni] file ...`

  Opzioni

`-a`	visualizza anche i file che iniziano con il punto
`-l`	output in formato esteso
`-g`	include/sopprime l'indicazione del proprietario
`-r`	ordine inverso (alfabetico o temporale)
`-F`	appende carattere per indicare i file particolari (/ * @)
`-R`	elenca anche i file nelle sottodirectory

Manipolazione di file

`cp [-fir] srci1 src2 ... dest`

copia uno o più file

`rm [-fir] file1 file2 ...`

cancella i file elencati

`mv [-fi] file1 file2 ... dest`

sposta uno o più file/cambia il nome di un file

-f non chiede mai conferma (attenzione!!!)

-i chiede conferma per ciascun file

-r opera ricorsivamente nelle sottodirectory

Manipolazione di directory

`cd` *`directory`*

cambia la directory in quella indicata

`pwd`

mostra path directory corrente

`mkdir` *`directory`*

crea la directory specificata

`rmdir` *`dir1 dir2 …`*

cancella una o più directory (devono essere vuote)

Esempi

- Listing dei files:
  - > ls
  - > ls –l
  - > ls –a
  - > ls –al
  - > ls -l /bin
  - > ...
- Creazione/rimozione di directory:
  - > mkdir d1
  - > rmdir d1
- Copia il file f1 in f2:
  - > cp f1 f2

Esempi

- Sposta/rinomina il file f1 in f2:
  - > mv f1 f2
- cp e mv come primo argomento possono prendere una lista di file; in tal caso il secondo argomento deve essere una directory:
  - > cp f1 f2 f3 d1 (copia f1, f2, f3 nella directory d1)

Visualizzazione di file di testo

`cat file1 file2 ...`

concatena i file sullo std output

`head [-n] file1 file2`

visualizza le prima *n* righe

`tail [-+nrf] file1 file2 ...`

Visualizza le ultime (con + salta le prime) 10 righe

-r visualizza in ordine inverso

-f rilegge continuamente il file

-n visualizza (salta) le ultime (prime) *n* righe

Informazioni sul sistema

- Ogni utente è identificato dal suo login (UID) ed appartiene a uno o più gruppi (GID)
- Per avere informazioni sugli utenti o sul sistema:
  - **`whoami`**
  - **`who am i`**
  - **`who`**
  - **`w`**
  - **`id`**
  - **`groups`**
  - **`finger`**
  - **`uname`**
  - **`passwd`**
  - **`su`**
  - **`date`**

Help in linea

- Tutti i comandi di UNIX sono documentati in linea
  - **man *comando***

  Organizzano in sezioni corrispondenti ad argomenti

1. Commands
2. System Calls
3. Library Functions
4. Administrative Files
5. Miscellaneous Information
6. Games
7. I/O and Special Files
8. Maintenance Commands

Help in linea

- Oltre al man

  `apropos` ***`chiave`***

  elenca le pagine del manuale contenente *`chiave`*

  `whatis` ***`comando`***

  indica le sezioni del manuale in cui si trova *`comando`*

I metacaratteri

I metacaratteri in Unix

- La shell Unix riconosce alcuni caratteri speciali, chiamati metacaratteri, che possono comparire nei comandi.
- Quando l'utente invia un comando, la shell lo scandisce alla ricerca di eventuali metacaratteri, che processa in modo speciale.
- Una volta processati tutti i metacaratteri, viene eseguito il comando.

Esempio

- user> ls *.java
- Albero.java div.java ProvaAlbero.java
- AreaTriangolo.java EasyIn.java ProvaAlbero1.java
- AreaTriangolo1.java IntQueue.java
- Il metacarattere * all'interno di un pathname è un'abbreviazione per un nome di file. Il pathname *.java viene espanso dalla shell con tutti i nomi di file che terminano con l'estensione .java. Il comando ls fornisce quindi la lista di tutti e soli i file con tale estensione.

Abbreviazione del Pahtname

- I seguenti metacaratteri, chiamati *wildcard* sono usati per abbreviare il nome di un file in un pathname:

Metacarattere	Significato
*	stringa di 0 o più caratteri
?	singolo carattere
[ ]	singolo carattere tra quelli elencati
{ }	stringa tra quelle elencate

Esempi

- user> cp /JAVA/Area*.java /JAVA_backup

  (copia tutti i file il cui nome inizia con la stringa *Area* e termina con l'estensione .java nella directory JAVA_backup.)
- user> ls /dev/tty?

  (/dev/ttya /dev/ttyb)

Il “quoting”

- Il meccanismo del quoting è utilizzato per inibire l’effetto dei metacaratteri. I metacaratteri a cui è applicato il quoting perdono il loro significato speciale e la shell li tratta come caratteri ordinari.
- Ci sono tre meccanismi di quoting:
  1. il metacarattere di escape \ inibisce l’effetto speciale del metacarattere che lo segue:
     - user> cp file file\?
     - user> ls file*
       file file?

Il “quoting”

2. tutti i metacaratteri presenti in una stringa racchiusa tra singoli apici perdono l’effetto speciale:
   - user> cat 'file*?‘
   - …
3. i metacaratteri per l’abbreviazione del pathname presenti in una stringa racchiusa tra doppi apici perdono l’effetto speciale (ma non tutti i metacaratteri della shell):
   - user> cat "file*?"
   - ...

I path

- . è la directory corrente
- .. è la directory padre di quella corrente
- I file che iniziano con . sono nascosti
- Path assoluto = /dir1/dir2/…
  - Parte dalla radice del file system
- Path relativo = dir1/dir2/…
  - Parte dalla cartella corrente

I file

- Un solo tipo di file fisico: byte stream
- 4 tipi di file logici
  - Directory
    - Contiene nomi e indirizzi di altri file
  - Special file
    - Entry point per un dispositivo di I/O
  - Link
    - Collegamento ad un altro file
  - File ordinario
    - Tutti gli altri file

Special file

- Ogni device di I/O visto come un file
- I programmi non sanno se operano su file o device di I/O
- Lettura/scrittura su special file causano operazioni di I/O sul relativo device
- Indipendenza dai dispositivi!

Link

- Hard link
  - Un nome (in una directory) che punta a un i-node puntato anche da altri
- Soft link
  - Un file che contiene il nome di un altro file
- Particolarità
  - Non si può fare hard link di directory
  - Non si può fare hard link a file su altri file system
  - Un file viene rimosso quando tutti i suoi hard link sono stati rimossi

Occupazione spazio su disco

- Per controllare l'occupazione dei dischi

  `df [-k –h]`

  Opzioni

  -k mostra l'occupazione in KByte

  -h mostra l'occupazione in formato "umano"

Occupazione spazio su disco

- Per vedere lo spazio (in blocchi) occupato da una directory e tutte le sottodirectory

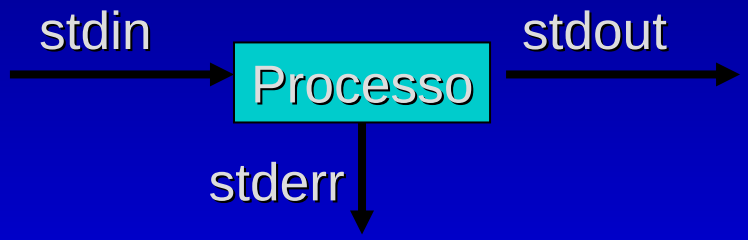
  `du [-opzioni] directory ...`

  Opzioni

  - `-a` mostra l'occupazione di ciascun file
  - `-s` mostra solo il totale complessivo
  - `-k` mostra l'occupazione in KByte

Ri-direzione dell'I/O

- Ogni processo ha tre canali associati



- Ogni canale può essere ri-diretto
  - su file
  - su altro canale
- Il collegamento stdout → stdin si chiama ***pipe*** e crea in memoria un canale di comunicazione diretto tra due processi

Ri-direzione dell'I/O

`comando < file`

stdin da file

`comando > file`

stdout in file (cancellato se esiste)

`comando >> file`

stdout aggiunto in coda al file

`comando >& file`

stderr e stdout in file

`comando 2> file`

stderr in file (cancellato se esiste)

Ri-direzione dell'I/O

`comando > file 2>&1`

`comando &> file`

stdout e stderr sullo stesso file descriptor

`comando1 | comando2`

pipe tra comando1 e comando2 (stdout di comando1 in stdin di comando2)

Esempi

```
user> echo ciao a tutti >file      # ridirezione dell'output
user> more file
ciao a tutti

user> echo ciao a tutti >>file     # ridirezione dell'output (append)
user> more file
ciao a tutti
ciao a tutti
```

- Il comando wc (word counter) fornisce numero di linee, parole, caratteri di un file:
  - user> wc <progetto.txt
    21 42 77
  - user> wc <<delim # here document
    ? queste linee formano il contenuto
    ? del testo
    ? delim
    2 7 44
  - user> man -s2 passwd # ridirezione dei messaggi di errore
    No entry for passwd in section(s) 2 of the manual.
  - user> man -s2 passwd 2>temp

Visualizzazione per pagine

- Esistono tre comandi quasi equivalenti

```
pg   file1 file2 ...
more file1 file2 ...
less file1 file2 ...
```

- Durante la visualizzazione è possibile dare dei comandi interattivi
  - spazio prossima pagina
  - CR prossima riga
  - b pagina precedente
  - */pattern* prossima pagina con *pattern*
  - *?pattern* pagina precedente con *pattern*
  - q termina programma
  - v edita file corrente

Cambio di proprietario

`chgrp [-R] gruppo file`

cambia il gruppo del file

`chown [-R] utente[:gruppo] file`

cambia proprietario [e gruppo] del file

- In entrambi i casi l'opzione –R indica di propagare il comando alle sottodirectory

Cambio protezione

`chmod [-R] protezione file`

Protezioni assolute: un numero di quattro cifre (il primo si può omettere)

	padrone	gruppo	altri
4 2 1	4 2 1	4 2 1	4 2 1
s S t	r w x	r w x	r w x

Protezioni simboliche: una stringa di tre caratteri

`ugoa + - = rwxst`

Cambio protezione

- Esempi
  - **`chmod 640 prova.txt`**
    - Lettura/scrittura per proprietario
    - Lettura per gruppo
    - Nessun permesso per altri
  - **`chmod 755 dir`**
    - Lettura/scrittura/esecuzione per proprietario
    - Lettura/esecuzione per gruppo
    - Lettura/esecuzione per altri

Sticky bit

- Sticky bit (t)
  - Non usato su file
  - Per directory, solo il proprietario del file o root possono cancellare o rinominare i file contenuti
    (es. directory /tmp)

```
$ ls -ld
/tmp drwxrwxrwt 6 root root 1024 Aug 10 01:03 /tmp
```

Setuid e setgid

- Setuid (s)
  - Per diventare temporaneamente il padrone del file
- Setgid (S)
  - Per diventare temporaneamente dello stesso gruppo del padrone del file

```
$ ls -l /usr/bin/passwd
-r-s--x--x 1 root root 17700 Jun 25 2004 /usr/bin/passwd
```

Protezioni standard

`umask` *`maschera`*

Per definire la maschera delle protezioni

- Il comando **`umask`** senza argomento mostra i permessi che sono NEGATI quando si crea un file (la maschera delle protezioni)

- Esempio:

  `umask 027`

  Nega tutti i permessi agli “altri” e i permessi di scrittura al “gruppo”

Ricerca di un file

`find directory espressione`

Visita tutto l'albero sotto la directory specificata e ritorna i file che rendono vera l'espressione

-name pattern (usare gli apici se si usano espressioni regolari)

-type tipo (b c d l f)

-user utente

-group gruppo

-newer file

-atime, mtime, ctime [+/-] giorni

-print

-size [+/-] blocchi

Confronto di file

`diff [-opzioni] file1 file2`

`diff [-opzioni] dir1 dir2`

mostra le righe diverse, indicando quelle da aggiungere (a), cancellare (d) e cambiare (c)

-b ignora gli spazi a fine riga, collassa gli altri

-i ignora la differenza tra maiuscolo e minuscolo

-w ignora completamente la spaziatura

Confronto di file – Esempio

- Prova1
  ciao
  come va?
  bene
  grazie
- Prova 2
  ciao
  come?
  bene
  molto bene
  grazie
- Prova 3
  ciao

```
$ diff Prova1 Prova2
2c2
< come va?
---
> come?
4c4,5
< grazie
---
> molto bene
> grazie
```

```
$ diff Prova1 Prova3
2,4d1
< come va?
< bene
< grazie
```

```
$ diff Prova3 Prova1
1a2,4
> come va?
> bene
> grazie
```

Modifica di attributi di file

`touch [-opzioni] [data] file ...`

aggiorna data e ora dell'ultimo accesso/modifica di un file

- se data non è specificata, usa data e ora corrente
- se il file non esiste lo crea vuoto

-a modifica accesso

-m ultima modifica