INTRODUZIONE A UNIX E AL SOFTWARE "LIBERO" DI INTERESSE PER CHIMICI

(questi schemi delle lezioni sono su `http://www.chim.unifi.it/~signo/did/unix`)

20 maggio 2002

1 Introduzione

- organizzazione:

	LU	MA	ME	GI	VE
8:30-10:30				**A**	
10:30-12:30			**lezione**		
14:00-16:00		**D**		**B**	
16:00-18:00		**E**		**C**	

lezione	**Esercitazione: turno A**	**turni B,C**	**turni D,E**
12/12/2001	13/12	14/12	18/12
19/12	20/12	21/12	08/01/2002
09/01	10/01	11/01	15/01
16/01	17/01	18/01	22/01

- scopo del corso
- struttura del corso

1. Software libero; Uso essenziale di Unix
2. Unix: approfondimenti
3. Uso di programmi "liberi" di utilità per chimici (calcolo di proprietà molecolari, ...)
4. Ricerca in WWW di informazione e software "libero"

- bibliografia

2 Unix e il software libero

2.1 Generalità

Secondo alcuni, potere disporre di strumenti informatici "liberi" (nel senso che chiariremo subito)

- dà più flessibilità (portabilità, adattabilità a esigenze nuove)
- favorisce lo sviluppo di nuove applicazioni e la collaborazione

Questo vale soprattutto per l'ambiente di lavoro: il sistema operativo e le utilità di base (UNIX si può considerare, almeno in parte, uno strumento libero, e forse in questo sta la chiave del suo crescente successo); ma anche per i programmi applicativi.

Altri sostengono che senza la possibilità di sfruttare economicamente la "proprietà intellettuale" non c'è stimolo alle invenzioni. Ma cosa sarebbe successo

- se Gutenberg avesse brevettato la stampa a caratteri mobili?
- se tutti i costruttori di macchine avessero dovuto pagare i diritti a chi ha inventato la chiave inglese?

2.1.1 software libero
(Basato su [AIL] e http://www.gnu.org/philosophy/free-sw.html)

È "libero" il s/w che permette a chiunque di:

1. usarlo;
2. studiare come funziona (accesso al codice);
3. ridistribuirlo;
4. modificarlo e distribuirlo modificato

s/w di dominio pubblico = senza copyright né licenza (qualcuno può "appropriarsene")

copyleft = licenza che garantisce diritti di s/w libero, ma impone di mantenerlo libero (controesempio: X11, è distribuito con licenza libera, ma non impedisce restrizioni- alcune versioni sono a pagamento e sono le uniche a funzionare su un certo h/w)

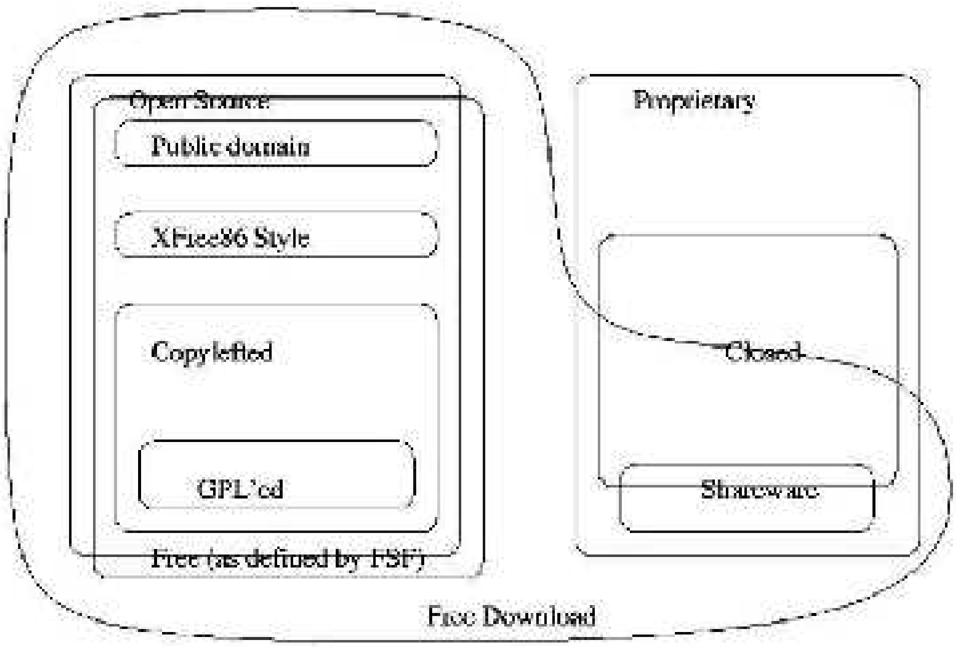
GNU Public License (GPL) = particolare licenza con *copyleft* che permette anche l'uso commerciale del s/w

s/w semi-libero = s/w libero, ma senza permesso di trarne profitto

freeware = s/w gratuito, spesso senza sorgente

shareware = s/w gratuito per tempo limitato

Figura 1: vari tipi di software libero



2.1.2 sistemi operativi

Cos'è un SO:

- permette a utente gestione di tutte le risorse del calcolatore (CPU, dispositivi di ingresso/uscita, dischi, periferiche, ...) e di allocarle agli utenti.
- Fa girare programmi, mette a disposizione un gestore di file.
- Solleva l'utente da interagire direttamente con la CPU

Differenza tra SO, hardware e programmi applicativi

SO == "kernel"

+~ comandi, shell, ambiente grafico a finestre

+~ editor, comunicazioni

> [Free OnLine Dic Of Computing]:
> operating system:
> The low-level software which handles the interface to peripheral hardware, schedules tasks, allocates storage, and presents a default interface to the user when no application program is running.
> The OS may be split into a kernel which is always present and various system programs which use facilities provided by the kernel

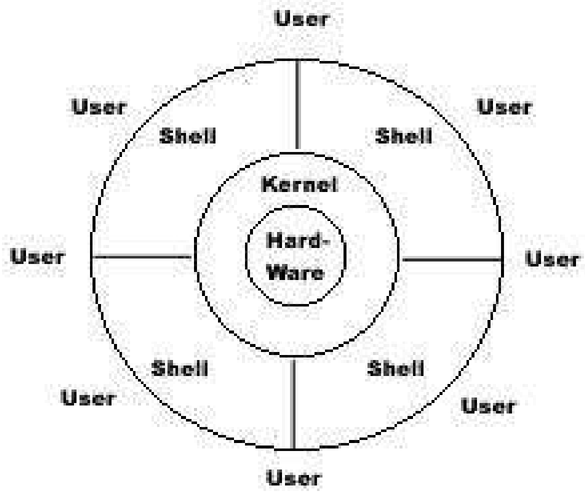


Figura 2: vari strati di *software* tra utenti e *hardware*

to perform higher-level house-keeping tasks, often acting as servers in a client-server relationship.

Some would include a graphical user interface and window system as part of the OS, others would not. The operating system loader, BIOS, or other firmware required at boot time or when installing the operating system would generally not be considered part of the operating system, though this distinction is unclear in the case of a rommable operating system such as RISC OS.

The facilities an operating system provides and its general design philosophy exert an extremely strong influence on programming style and on the technical cultures that grow up around the machines on which it runs.

Example operating systems include 386BSD, AIX, AOS, Amoeba, Angel, Artemis microkernel, BeOS, Brazil, COS, CP/M, CTSS, Chorus, DACNOS, DOSEXEC 2, GCOS, GEORGE 3, GEOS, ITS, KAOS, Linux, LynxOS, MPV, MS-DOS, MVS, Mach, Macintosh operating system, MINIX, Multics, Multipop-68, Novell NetWare, OS-9, OS/2, Pick, Plan 9, QNX, RISC OS, STING, System V, System/360, TOPS-10, TOPS-20, TRUSIX, TWENEX, TYMCOM-X, Thoth, Unix, VM/CMS, VMS, VRTX, VSTa, VxWorks, WAITS, Windows 3.1, Windows 95, Windows 98, Windows NT.

2.1.3 UN*X

Caratteristiche di UNIX:

- multi-utente
- multi-processo
- "filosofia" = attrezzi (efficienti), combinabili

- connettività in rete (TCP/IP)
- ambiente grafico (X-window system)

Punti di forza:

- adatto a fare da "server"
- gira su piattaforme piccole ed economiche

Breve storia di UNIX:
1969 Thompson (Bell Laboratories, NJ, USA)
Principali famiglie di discendenti (anni '80-'90):

SystemV	(AT&T)	commerciale, solida
BSD	(Università Berkeley)	non-commerciale, dinamica

BSD deriva da versioni date alle Università per scopi di studio, compreso il codice sorgente. Dal 1993 c'è versione (4.4BSD Lite) ripulita dal codice proprietario AT&T.

> [AIL]
> I primi utenti di UNIX sono state le università, a cui in particolare questo sistema operativo veniva fornito a costo contenuto, ma senza alcun tipo di supporto tecnico, né alcuna garanzia. Proprio questa assenza di sostegno da parte della casa che lo aveva prodotto, stimolava la cooperazione tra gli utenti competenti, in pratica tra le università.
>
> Il maggior fermento intorno a UNIX si concentrò presso l'università della California a Berkeley, dove a partire dal 1978 si cominciò a distribuire una variante di questo sistema operativo: BSD (Berkeley Software Distribution).
>
> Per difendere il software prodotto in questo modo, nacque una licenza d'uso che rimane il progenitore della filosofia del software libero: la licenza BSD.
>
> Per molto tempo, la variante BSD di UNIX rimase relegata all'ambito universitario o a quello di aziende che avevano acquistato i diritti per utilizzare il codice sorgente dello UNIX originale. Ciò fino a quando si decise di ripulire lo Unix BSD dal codice proprietario.
>
> Il risultato iniziale fu 386BSD, che venne rilasciato nel 1992 con la versione 0.1. [...] si svilupparono altri progetti indipendenti per ottenere, finalmente, un sistema BSD libero. Il primo di questi fu nominato NetBSD, al quale si aggiunse subito dopo FreeBSD; più tardi, apparve anche OpenBSD.
>
> [...]
>
> Allo stato attuale, le tre varianti *BSD sono tutte riconducibili a BSD 4.4-Lite, dove le differenze più importanti riguardano le piattaforme hardware in cui possono essere installate e l'origine della distribuzione. Infatti, il punto di forza della variante OpenBSD, sta nel fatto di essere realizzata in Canada, da dove possono essere distribuiti anche componenti per la comunicazione crittografica.

Negli anni '90 dal punto di vista del linguaggio le due famiglie (BSD e SysV) si sono avvicinate; comunque sopravvivono due gruppi:

- UNIX commerciali (generalmente contengono almeno una parte dell'originale sorgente AT&T, e la relativa licenza)
- UNIX (più o meno) liberi: FreeBSD, NetBSD, ...: GPL: Linux, Hurd

Punti di riferimento:

- UNIX è un marchio registrato

  "UNIX® is a registered trademark of The Open Group""

deriva da quello di Bell Lab -> AT&T -> ...

però di fatto si è trasformato in uno standard; è proprietà di un'organizzazione più interessata agli standard industriali che alla proprietà intellettuale. Ci sono test passati i quali un S.O. può dirsi legalmente UNIX.

- POSIX: sono standard puri; praticamente, un sottoinsieme di UNIX®

> POSIX (Portable Operating System Interface):
>
> A set of IEEE standards designed to provide application portability between Unix variants. IEEE 1003.1 defines a Unix-like operating system interface, IEEE 1003.2 defines the shell and utilities and IEEE 1003.4 defines real-time extensions.

2.1.4 Linux

- libero (GPL)
- POSIX

Linux kernel + comandi, shell + TCP/IP + X11 + Desktop Manager + ... = "Linux distribution"

- Debian
- Caldera
- RedHat
- Slackware
- SuSE
- ...

Noi = RedHat 7.2

2.1.5 GNU

è un insieme di strumenti da applicare a un kernel UNIX: GNU/Hurd, GNU/Linux
Free Software Foundation (FSF)

> [AIL]
> Nel 1985, Richard Stallman ha fondato la FSF, Free Software Foundation, con lo scopo preciso di creare e diffondere la filosofia del «software libero». Libertà intesa come la possibilità data agli utenti di distribuire e modificare il software a seconda delle proprie esigenze e di poter distribuire anche le modifiche fatte.
>
> Queste idee filosofiche si tradussero in pratica nella redazione di un contratto di licenza d'uso, la General Public License, studiato appositamente per proteggere il software libero in modo che non potesse essere accaparrato da chi poi avrebbe potuto impedirne la diffusione libera. Per questo motivo, oggi, il copyright di software protetto in questo modo, viene definito copyleft.
>
> Il software libero richiede delle basi, prima di tutto il sistema operativo. In questo senso, l'obiettivo pratico che si prefiggeva Richard Stallman era quello di realizzare, con l'aiuto di volontari, un sistema operativo completo.
>
> Nacque così il progetto GNU (Gnu's Not Unix), con il quale, dopo la realizzazione di un compilatore C, si volevano costruire una serie di programmi di servizio necessari nel momento in cui il cuore del sistema fosse stato completo.
>
> Il progetto GNU diede vita così a una grande quantità di software utilizzabile sulla maggior parte delle piattaforme Unix, indirizzando implicitamente il software libero nella direzione dei sistemi di questo tipo.
>
> Nel 1990 inizia lo sviluppo del kernel Hurd e intorno al 2000 inizia la distribuzione del sistema GNU/Hurd (sistema GNU basato su kernel Hurd).

2.2 Uso essenziale di UNIX

2.2.1 login/logout

- maiuscole e minuscole sono diverse!
- la shell: invito ai comandi (*prompt*)
- un comando importante: `passwd` [`yppasswd`]
- `logout` e `exit`

2.2.2 manuali

`man` *comando*	guida esauriente su comando (qualche pagina)
`whatis` *comando*	definizione (una riga)
`apropos` *xxx*	lista di comandi che hanno a che vedere con l'argomento xxx

Es:

```
man passwd; whatis ls; apropos calendar
```

2.2.3 utenti e gruppi

- si accede sempre come un utente
- ogni utente fa parte di un gruppo
- utenti e gruppo hanno accesso differenziato a risorse
- l'utente root ha tutti gli accessi: serve per gestione sistema (trascuriamo)
- un utente ha:
  - nome
  - password
  - (nome reale)
  - directory di partenza (home)
  - shell di partenza

Es:

```
who, finger
```

2.2.4 file, directory, filesystem

- sistema a directory gerarchico
- filesystem: punto di innesto (*mount point*)
- le proprieta' dei file: `rwx...`
- il carattere iniziale ".": directory e file "nascosti"
- tutto è file: anche i dispositivi (*device*)
- link: soft e hard

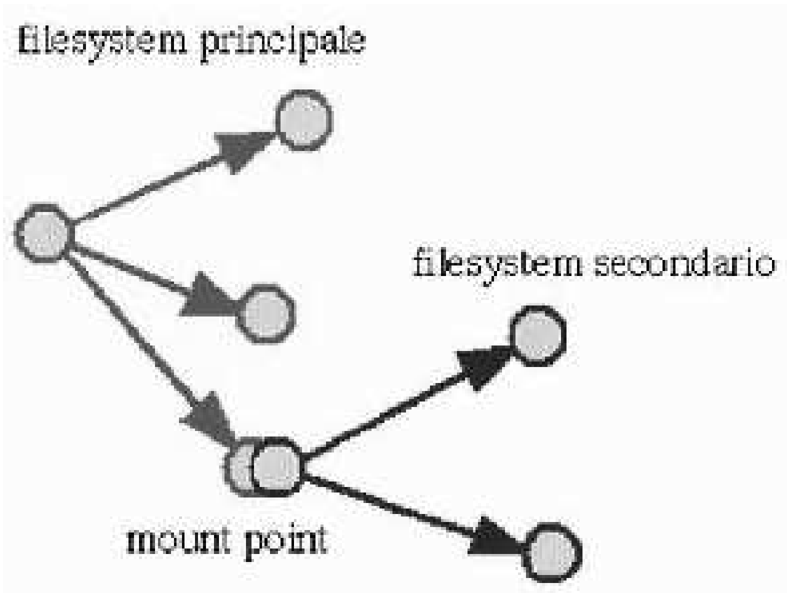


Figura 3: punto di innesto di un filesystem su un altro

2.2.5 percorso assoluto e relativo; directory corrente; le directory . .. ~

```
/--.
   |
   +-home/--+-tizio/-----+-Desktop
   |        |            |
   +-dev/   +-caio/      +-ese/     pippo
   |        |            |          ...
   +-usr/   +            ...
   |        |
   +-tmp/   +
   |        |
  ...       +-infochim/--+ ...
            |            |
           ...           +-wrk/---+ ...
                         |        |
                        ...       +-e3/  gigi
                                  |      ...
                                 ...
```

Il percorso assoluto per i due file indicati è, rispettivamente:

/home/tizio/ese/pippo
/home/infochim/wrk/e3/gigi

Se si vuole copiare un file nell'altro si può sempre dare il comando:

```
cp /home/tizio/ese/pippo /home/infochim/wrk/e3/gigi
```

La denominazione di un file può essere data anche come percorso relativo rispetto alla directory corrente. Se portiamo la directory corrente a /home/tizio/ese, possiamo riferirci al primo file con il suo percorso relativo:

```
cd /home/tizio/ese
cp pippo /home/infochim/wrk/e3/gigi
```

Ovvero:

```
cd /home/infochim/wrk/e3
cp /home/tizio/ese/pippo gigi
```

Per conoscere quale è la directory corrente, usare il comando `pwd`

Ci sono dei nomi speciali di directory:

`.`	la directory corrente
`..`	la directory "genitrice" (=sopra la corrente)
`~`	la mia home directory
`~tizio`	la home directory dell'utente `tizio`

Notare che

- `.` `..` `~` sono directory diverse a seconda di dove ci si trova e di che utente siamo.
- riferirsi a un file come *./pippo* o come *pippo* è identicamente la stessa cosa.

Esempio dell'uso di `~`: se ho fatto login come utente `tizio`, il comando che copia un file nell'altro specificando il percorso assoluto di entrambi può essere:

```
cp ~/ese/pippo ~infochim/wrk/e3/gigi
```

2.2.6 proprietà delle shell

queste sono proprietà per lo più comuni a tutte le shell

analisi e interpretazione di una riga di comando (parsing)

- una riga di comando è ciò che si digita al prompt, fino al tasto <invio>
- una riga di comando si scompone in:
  - comando (la prima parola)
  - opzioni (switch) (tutto ciò che cominicia con il carattere -)
  - argomenti

Es:

```
ls -l *.rtf *.txt; tar --help
```

- percorso di ricerca dei comandi (path):

esecuzione in serie e in parallelo; job

`a ; b`	esegui `a`, poi `b`
`a && b`	esegui `a`; se ha esito positivo, esegui `b`
`a || b`	esegui `a`; se non ha esito positivo, esegui `b`
`a & b`	esegui `a` e contemporaneamente `b`
`a &`	esegui `a` e ritorna al *prompt* (esegui `a` in *background*)

reindirizzamento, pipeline

- ci sono tre "file" speciali:

standard input	(tastiera)
standard output	(video)
(*standard error*)	(video)

comando > *file*	dirigi output di *comando* su *file*
comando < *file*	prendi input di *comando* da *file*
comando1 | *comando2*	l'output di *comando1* diventa input di *comando2*
comando >> *file*	aggiungi output di *comando* in coda a *file*

sostituzioni:

`alias`	sinonimo per un comando (e argomenti)
`*?[]`	sostituisci nomi di file della directory ("globbing") differenza da DOS!
`$`	variabili (di ambiente e locali)
`~`	directory di partenza (home directory) degli utenti
`!`	cronologia
`%`	job
‘	output di comandi

proteggere da sostituzione:

\	protegge carattere successivo
’’	protegge tutto il testo contenuto tra ’ e ’
""	protegge una frase, tranne `$` ‘ e \ (che vengono interpretati)

altri caratteri speciali:

`^C` Arresta definitivamente il processo in corso

`^Z` Interrompi temporaneamente il processo in corso (si ripristina con `bg` o `fg`)

`^D` Fine input, file, sessione

2.2.7 lista di comandi utili

guide

```
man     - format and display the on-line manual pages
apropos - search the whatis database for strings
whatis  - search the whatis database for complete words
```

file, directory, dischi

```
ls     - list directory contents
rm     - remove files or directories
mv     - move (rename) files
cp     - copy files and directories
cd     - Change working directory
pwd    - print name of current/working directory
mkdir  - make directories
rmdir  - remove empty directories
chmod  - change file access permissions
chown  - change file owner and group
find   - search for files in a directory hierarchy
df     - report filesystem disk space usage
du     - estimate file space usage
```

manipolazione di file

```
cat     - concatenate files and print on the standard output
sort    - sort lines of text files
head    - output the first part of files
tail    - output the last part of files
grep    - print lines matching a pattern
wc      - print the number of bytes, words, and lines in files
diff    - find differences between two files
more    - file perusal filter for crt viewing
less    - opposite of more
```

manipolazione avanzata di file

```
sed           - a Stream EDitor
gawk [awk]    - pattern scanning and processing language
```

stampa e simile

```
lpr     - off line print
lpq     - spool queue examination program
echo    - display a line of text
```

archiviazione e compressione

```
tar     - The GNU version of the tar archiving utility
compress [uncompress] - compress and expand data
gzip    - compress or expand files
```

processi, utenti

```
ps      - report process status
kill    - terminate a process
w       - Show who is logged on and what they are doing
who     - show who is logged on
finger  - user information lookup program
passwd  - update a user's authentication tokens(s)
```

2.2.8 vi
(ottimo manuale: http://www.geek-girl.com/Unixhelp/vi/index.html [UHU])

concetti base

- modalità di operazione
  - modalità comando
  - modalità inserimento
- il registro (buffer)

avviamento e apertura di file

- un file esistente
- un file nuovo
- senza argomenti

salvare e uscire

`:w [nomefile]`	salvare
`:q`	uscire
`:wq`	salvare e uscire
`:q!`	uscire senza salvare

inserimento di testo

`a`	inserire testo dopo il cursore
`i`	inserire testo prima del cursore
`o`	aggiungere una riga sotto
`O`	aggiungere una riga sopra

muoversi

`l`	cursore a destra
`h`	cursore a sinistra
`k`	cursore su
`j`	cursore giù

cancellare

`x`	carattere
`dh`	carattere precedente
`dd`	intera riga
`d$`	fino a fine riga
`p`	annulla ultima cancellazione

cercare e sostituire

`/`	cercare una serie di caratteri
`?`	cercare una serie di caratteri, all'indietro
`n`	cercare di nuovo in avanti
`N`	cercare di nuovo indietro
`:s/abc/xyz`	sostituire `abc` con `xyz` su una riga
`:g/abc/s//xyz/gc`	sostituire `abc` con `xyz` in tutto il documento, in modalità interattiva

3 UNIX: Approfondimenti

3.1 Regular Expression (espressioni regolari o formali)

- Sono espressioni che descrivono sequenze di caratteri. Vengono usate in funzioni e comandi (come `grep` o `awk`) che effettuano una ricerca di sequenze di caratteri all'interno di righe o stringhe.
- Si dice che c'è corrispondenza (*match*) tra una RE e una sequenza di caratteri contenuta in una stringa (e v.v.)
- La forma più semplice di una RE è una pura sequenza di caratteri, che corrisponde soltanto alla sequenza stessa:

Es: La RE `/ranc/` corrisponde alla sequenza "`ranc`" contenuta sia nella stringa "`Francesco`" che nella stringa "`melarancia`"

- Con caratteri speciali si possono costruire RE che corrispondono a più di una sequenza di caratteri. Ad es., il punto (`.`) rappresenta un qualsiasi carattere.

Es: `/or.o/` corrisponde sia a "`orto`" che a "`orco`", che a ...
Una RE è composta da:

elemento:	corrisponde a:
`c`	il carattere (non-speciale) `c`
`\c`	il carattere speciale `c`, preso letteralmente. Caratteri speciali: `.*\[]^$-"'`
`.`	qualsiasi carattere
`[abc...]`	uno qualsiasi dei caratteri `abc`...
`[v-z]`	= `[vwxyz]`
`[^abc...]`	qualsiasi carattere eccetto `abc`...
`[^v-z]`	= `[^vwxyz]`
`^`	inizio stringa
`$`	fine stringa

elemento:	*corrisponde a:*
`c`	il carattere (non-speciale) `c`
`\c`	il carattere speciale `c`, preso letteralmente. Caratteri speciali: `.*\[]^$-¨'`
`.`	qualsiasi carattere
`[abc...]`	uno qualsiasi dei caratteri `abc`...
`[v-z]`	= `[vwxyz]`
`[^abc...]`	qualsiasi carattere eccetto `abc`...
`[^v-z]`	= `[^vwxyz]`
`^`	inizio stringa
`$`	fine stringa

che si possono combinare nel modo seguente:

r1r2	l'espressione *r1* seguita dall'espressione *r2*
*r**	l'espressione *r* ripetuta zero o più volte
r+	l'espressione *r* ripetuta una o più volte

Esempi:

RE	*è contenuta in (in neretto sottolineato la corrispondenza)*	*non è contenuta in*
`/ranc/`	`"Francesco", "melarancia"`	
`/or.o/`	`"orco", "orto"`	`"oro"`
`/si do/`	`"famosi dottori"`	`"famosi  dottori"`
`/[Ff]ranc/`	`"Francesco", "francesi", "Francia"`	`"melarancia"`
`/Franc.*a/`	`"Francesca", "Francia"`	`"Francesco", "francesca"`
`/[Ff]rancesc[oa]/`	`"Francesco", "Francesca", "francesco", "francesca", "Francescato", "Pierfrancesco"`	`"Francia"`
`/tel\.com/`	`"www.alcatel.com"`	`"telecom"`
`/^car.*o/`	`"carota", "carico", "cartone"`	`"cara", "Icaro"`
`/car.*o$/`	`"Icaro"`	`"carota"`
`/bana[^n]e/`	`"banale"`	`"banane"`
`/ta.*a/`	`"tanto va la gatta al lardo"`	

In generale, una RE corrisponde alla stringa più lunga possibile.
Es:
nella stringa `"tanto va la gatta al lardo"`, la RE `/ta.*a/` corrisponde alla sottostringa `"tanto va la gatta al la"`.

3.2 awk

3.2.1 caratteristiche generali

- È un comando che serve per elaborare file (di testo o numerici).

Es:

```
awk '{print $1}' /etc/hosts
```

- Di fatto, è un linguaggio di programmazione *interpretato*, con sua sintassi e comandi. Si possono scrivere programmi in *awk*.

- C'è uno standard POSIX di linguaggio *awk*.
- Su Linux RedHat trovate la versione GNU di *awk* che si chiama anche *gawk*

3.2.2 manuali

- *http://www.gnu.org/manual/gawk/index.html*
- `man awk`

3.2.3 sintassi del comando awk

Da `man awk`:

```
SINTASSI
      awk [ opzioni ] testo-del-programma file ...
      awk [ opzioni ] -f file-di-programma file ...
```

Cioè:

- si possono dare istruzioni
  - O sulla riga di comando (poche istruzioni, cfr. esempio precedente)
  - O in un programma già scritto, con l'opzione -f
- awk è sempre dipendente da un file. (C'è comunque un "trucco" per eseguire un programma senza file di input)

3.2.4 istruzioni di awk

- processa una riga (record) dell'input alla volta
- su ciascuna riga esegue tutte le istruzioni, nell'ordine

Le istruzioni sono sempre del tipo

```
pattern {action}
```

che significa: se la riga contiene una stringa corrispondente a `pattern`, viene eseguita l'istruzione `action`.

Esempio:

```
awk '/polosci/ {print $1,$2}' /etc/hosts
```

Se `pattern` è assente, l'azione `action` viene sempre eseguita; se l'`action` è assente, si stampa l'intera riga

3.2.5 Campi

Ogni volta che awk legge un record lo spezza in campi, usando il valore della variabile FS (espressione regolare) come separatore di campo (default=sequenza di spazi o un tabulatore o un accapo). Se FS è un singolo carattere, i campi sono separati da quel carattere.

Es: il file */etc/passwd* ha il formato `user:password:userid:groupid:name:home:shell`

```
xavier:x:1224:1000:Francesco:/home/xavier:/bin/bash
```

Ogni campo nel record in ingresso può essere individuato dalla sua posizione: `$1`, `$2`, e così via. `$0` è l'intero record.

3.2.6 Variabili

Si possono definire ed utilizzare delle variabili:

```
r=25.
b="ciao"
crf=r*2*3.14
```

il riferimento alle variabili è fatto in modo diretto, senza il carattere `"$"` (differenza da shell!)

I loro valori sono numeri in virgola mobile, o stringhe, o entrambe le cose, a seconda di come sono usati.

AWK dispone di vettori monodimensionali (i vettori multidimensionali possono essere simulati):

```
x[i]=15
```

Al lancio del programma sono impostate parecchie variabili predefinite (es: FS)

3.2.7 Sequenze (*pattern*)

- espressioni regolari riportate tra / e /
- oppure: espressioni relazionali
  cioè quelle che usano (tra gli altri) gli operatori
  `==` `!=` `<` `>` `<=` `>=`
- combinazioni di pattern, ad esempio con operatori logici && ||
- BEGIN e END

3.2.8 azioni

Le istruzioni possono essere assegnamenti o istruzioni condizionali o iterative o di I/O come si trovano nella maggior parte dei linguaggi.

Gli operatori, le strutture di controllo e le istruzioni di input/output ricalcano le corrispondenti del linguaggio C

Operatori `+ - * / ^` etc.

Istruzioni di controllo

```
if (condizione) istruzione [ else istruzione ]
exit
```

Istruzioni di I/O

```
print [expr-list]
next
```

Funzioni predefinite

- numeriche: `cos()`, `exp()`, ...
- di stringa; es `length()`

3.2.9 un semplice esempio

Nel file *~infochim/wrk/esercitaz* sono riportati i punteggi ottenuti nelle prime due esercitazioni dai vari studenti del corso. Il file contiene nell'ordine: nome account, turno, punteggio eserc.1, punteggio eserc. 2:

```
acciai      D 2 5
allegra     A 2 3
allegri     A 0 0
arturoni    A 2 5
...
```

La media dei punteggi della prima esercitazione si ottiene con il seguente programma awk:

```
                {s=s+$3;n=n+1}
END             {print "media= ",s/n}
```

Possiamo scrivere questo programma nel file *media.awk* e eseguire il comando

```
awk -f media.awk esercitaz
```

Sull'output viene riportata la risposta:

```
media= 1.54082
```

Se ora vogliamo il punteggio medio di chi appartiene al turno A, possiamo farlo introducendo un pattern nella prima istruzione:

```
/A/             {s=s+$3;n=n+1}
```

Questo sistema funziona solo se siamo sicuri che l'account è sempre in minuscolo, e che quindi la stringa `"A"` può essere presente solo nel campo del turno.

Un sistema più corretto e sicuro è richiedere che la `A` si trovi nel secondo campo. In questo caso il pattern deve essere non un'espressione regolare, ma un'espressione relazionale:

```
$2=="A"         {s=s+$3;n=n+1}
```

Se invece vogliamo stampare in fondo a ogni riga la somma dei due punteggi:

```
{print $0, $3+$4}
```

Questa istruzione invece stampa solo le righe relative a studenti che hanno preso almeno 4 nella seconda esercitazione:

```
$4 > 3
```

3.3 APPLICAZIONI

3.3.1 gnuplot

- È un programma che serve per diagrammare
  - file di dati
  - oppure funzioni
- Si lancia con '`gnuplot`'
- Il comando principale è '`plot`'

per diagrammare un file di dati esistente:

`plot 'a.0'`	usa la prima colonna come x, la seconda come y
`plot 'a.0' using 1:3`	" " " " , la terza come y
`plot [1:10] 'a.0'`	nell'intervallo [x=1, x=10]
`plot [1:10][-100:100] 'a.0'`	nell'intervallo [x=1, x=10] e [y=-100, y=100]
`plot 'a.0' with lines`	formato del diagramma

per diagrammare una funzione:

`plot sin(x)`	100 punti nell'intervallo [x=-10,x=10]

- si possono definire costanti e funzioni
- Per uscire: `quit`
- Per guida: `help`

Esempio: il pH di una soluzione per aggiunta di base forte. Per diagrammare i punti sperimentali contenuti nel file *~infochim/wrk/titol* e la curva teorica $pH(x) = 14 + \log_{10}(x)$ tra $x = 0.01$ e $x = 1$:

```
cd ~infochim/wrk
gnuplot
gnuplot> plot [0.01:1] 'titol', 14.+log(x)/log(10.)
```

3.3.2 grafica molecolare: jmol (openscience)

3.4 esercizi

calcolo proprietà di una molecola da geometria:

- dipolo
- centro di massa, momento di inerzia

4 Risorse in rete

4.1 database fattuali e bibliografici

4.2 ricerca informazione in rete

5 Note e bibliografia

5.1 nota: la tastiera

UNIX fa uso di alcuni caratteri che non sono presenti sulle tastiere italiane. Sui calcolatori dell'aula didattica (aula 40) questi caratteri si ottengono con le seguenti combinazioni:

Per scrivere:	***digitare:***
~	`AltGr í`
{	`AltGr 7`
}	`AltGr =`
‘	`AltGr ú`

Riferimenti bibliografici

[UNIX] B.W. Kernighan, R. Pike, UNIX - Zanichelli, 1989 - ISBN 88-08-04376-2

[LinuxRef] R. Petersen, Linux: the Complete Reference, Sec - Osborne McGraw-Hill, 1998 - ISBN 0-07-882461-3

[LinuxFlash] M. A. Ricar, Linux flash - Apogeo, 1999 -ISBN 88-7303-506-X

[AIL] Appunti di Informatica Libera:.http://www.pluto.linux.it/ildp/AppuntiLinux/a2.html

[BIU] A Brief Introduction to Unix: http://staff.washington.edu/~corey/unix-intro+man.html

[UHU] Unix Help for Users: http://www.geek-girl.com/Unixhelp/

[4LW] Unix is a Four-Letter Word:http://www.caspur.it/guidautenti/unix/4ltrwrd/