Settembre 1, 1994
Alessandra Richetti

UNIX
Un mondo da scoprire

Questa introduzione è nata dalla necessità di consentire l'acquisizione di una conoscenza generale e di alcune nozioni di base del sistema operativo UNIX. È quindi orientata ai cosiddetti nuovi utenti o a chi, pur non prevedendo di operare in questo ambiente, desiderasse confrontarsi con un sistema operativo che è stato ed è, un grosso protagonista nel mondo della cultura informantica.

Informazioni revisione e aggiornamento: Prima Edizione

CAPITOLO 1

INTRODUZIONE AL SISTEMA OPERATIVO UNIX

Unix è stato ideato presso i "BELL LABORATORIES" nel 1969 e si è sviluppato in ambienti Universitari e di Ricerca. Progettato da D.M. RITCHIE e K. THOMSON in codice Assembler per macchine PDP-7 E PDP-9 venne poi riscritto per essere supportato anche dai PDP-11 (1971 data in cui risale la prima edizione supportata dal manuale). Uno dei primi obiettivi fu di renderlo *indipendente dalla macchina*. Per far ciò nel 1973 venne utilizzato il linguaggio **C** progettato da RITCHIE : il sistema è praticamente scritto in C.

Risale al 1979 la prima versione standar a cui fanno riferimento quelle attuali. In quegli anni, i Bell Laboratories decisero di fornire, a prezzo simbolico, il codice sorgente agli ambienti di ricerca e universitari: da qui una delle caratteristiche di UNIX : *l'ampio spettro dei servizi che il sistema offre*. Questa fu anche la causa dello svilupparsi rapido dei cosiddetti *'dialetti unix'*.

Nel 1980 l'Università di Berkeley presentò un rilevante progetto di modifica dell'originale, conosciuto come **UNIX BSD** (Berkeley software distribution): le modifiche interessarono soprattutto i servizi di rete, l'editor di schermo, il file system così strutturato ed altri servizi orientati soprattutto all'utente.

L'At&T impose, poi, licenze notevolmente restrittive e così determinò, per contrapposizione la nascita dell'Open Software Fundation (OSF) i cui membri sono rappresentati da HP, Apollo e Digital. Il primo prodotto dell'OSF di grosso interesse commerciale fu il MOTIF un'applicazione tipo X-Window .

L'insediamento di UNIX fu esplosivo, proprio per la sua principale caratteristica. Unix può esser caricato su un ampio spettro di macchine : stazioni di lavoro, mainframe, minicomputer, personal ...
Naturalmente questa sua condizione è stata la causa dello svilupparsi di servizi che hanno appesantito sempre di più l'originale snellezza, e quindi velocità, del sistema stesso. L'iniziale disponibilità ha così determinato una serie di "separatismi", con dettagli e caratteristiche sempre più nascoste e incontrollabili, con conseguente incompatibilità fra i vari "dialetti". Per limitare queste evoluzioni vari organismi a livello Europeo e non, hanno definito sei gruppi principali (standard): *SunOS; SCO Xenix; Hp-UX; Ultrix; At&T System V; IBM AIX.*

CAPITOLO 2

CONCETTI DI UNIX ...

1. sistema operativo multiutente e multiprocesso;
2. file system ovvero l'idea di file;
3. kernel;
4. interprete dei comandi:la shell;
5. comandi sintetici - sintassi standard;
6. pipe;
7. background & foreground;
8. ideale ambiente di programmazione.

2.1 Sistema operativo multiutente e multiprocesso

Le risorse, quali la memoria (main memory), il processore (CPU) e le periferiche (dischi...) vengono condivise fra un certo numero di processi o task in competizione fra loro. La principale conseguenza per un sistema multi utente è la necessità di protezione dei file e dei processi stessi. Ogni utente è provvisto di un nome (username) con cui gli viene riconosciuta la possibilità di accedere alle risorse del sistema stesso.

2.2 Il filesystem

Produrre un cospicuo numero di files è relativamente facile. È importante evitare che tale operazione conduca ad un'inevitabile confusione. Nasce l'esigenza di organizzare i dati. Unix presenta un'organizzazione di direttori strutturata ad albero conosciuta con il nome di **filesystem**. La "radice" di questa struttura prende il nome di **root** e viene rappresentata come **/**. Un nome di file completo, o **pathname** viene espresso come una sequenza composta da parole separate da /. Se questa inizia con / si indica il pathname come *"absolute pathname"* si valuta così la collocazione del file stesso nella struttura ad albero partendo dalla "radice". Altrimenti il pathname è detto *"relative pathname"* e il punto di riferimento è il direttorio *corrente*.
Il realtà il concetto *file system* viene utilizzato per riferirsi sia, al meccanismo software che è responsabile della gestione di tutti i file (organizzazione gerarchica, protezione ...) e sia, alla parte di memoria di massa divisa in un certo definito numero di blocchi disco. I file systems possono essere locali (il disco fisico è collegato direttamente alla macchina su cui si lavora) o NFS (*Network File System* il dispositivo fisico è collegato ad una macchina remota)

```
ale2ts.ts.infn.it> df
Filesystem         Total    kbytes   kbytes   %
node               kbytes   used     free     used  Mounted on
/dev/rz2a            18767   11468    5423    68%   /
/dev/rz1g           840718  331079  425568    44%   /usr
/dev/rz2g           111935   91374    9368    91%   /var
/dev/rz3c           191958   70277  102486    41%   /utenti
/dev/rz2d            14087    3324    9355    26%   /utenti1
/dev/rz4c           186957  181436       0   108%   /usr/lib/dxbook
alexts:/usr/utenti  227079  178815   25557    87%   /alexts
ale2ts.ts.infn.it>
```

2.2.1 File e directory

I *files* hanno un importanza primaria in Unix - questo non è vero per altri sistemi operativi. I comandi, per esempio, sono, *files eseguibili*; le comunicazioni tra processi, tra device, le operazioni di I/O ... qualsiasi entità è vista come file ed è indipendente dai dispositivi fisici. I files sono di tre tipi :

- files ordinari (esecutivi, dati, procedure ....)
- directory (file con le informazioni su altri file o directory)
- files speciali

Immediatamente dopo il collegamento la directory corrente è la *home directory* dell'utente.

```
Login: alex
Password:
Last login: Tue May 18 10:46:34 from VSTST0
ULTRIX V4.3 (Rev. 44) System #2: Tue Nov 19 10:32:14 GMT-1:00 1991
UWS V4.3 (Rev. 10)

               Digital Equipment Corporation
               Nashua, New Hampshire

Tue May 18 11:52:20 GMT-1:00 1993
csh> pwd
/utenti1/alex
csh>
```

L'accesso ai vari files viene regolamentato dalle protezioni e dal concetto di proprietà. Si può garantire o negare la scrittura, la lettura e l'esecuzione di un file.

```
csh> ls -la
total 149
drwxrwxrwx  2 alex          512 Oct 25 08:23 .
drwxr-xr-x 16 root          512 Sep 24 16:13 ..
-rw-r--r--  1 alex          190 Jun 10 09:52 .Xdefaults
-rw-r--r--  1 alex          189 Jun 10 09:52 .Xdefaults.old
-rwxr-x--x  1 alex          284 Sep  8 19:01 .cshrc
-rw-r--r--  1 alex            0 Sep  3 19:08 .gopherrc
-rwxr-x--x  1 alex          284 Oct  1 11:31 .login
-rw-r--r--  1 alex          300 Oct 25 09:48 .plan
-rwxr-x--x  1 alex          182 Jun 10 09:52 .profile
-rw-r--r--  1 alex          311 Sep  8 19:46 .uam_log
-rw-r--r--  1 alex            0 Aug 26 11:26 .ue_profile
-rw-r--r--  1 alex          438 Aug 18 16:10 default.DECterm
-rw-r--r--  1 alex         1343 Aug 17 09:46 dum
-rw-r--r--  1 alex         1136 Aug 19 10:54 dum1
-rw-r--r--  1 alex       117009 Aug 17 11:47 dxlist
-rwxr--r--  1 alex           55 Jun 10 09:52 dxv0
-rwxrwxrwx  1 alex           90 Jun 10 09:52 my
-rwxr-----  1 alex           79 Aug 19 09:43 data
drwx------  2 alex          512 Sep  2 11:45 test
```

Nell'esempio si richiede attraverso il comando **ls** la lista di tutti i files "conosciuti" dal direttorio corrente e di alcune informazioni inerenti ad essi. Su ogni linea compare la chiave di accesso al file (-rwxrwxrwx) e chi ne è il proprietario (alex). Analizziamo i dati che si riferiscono al file di nome *my* i **r**ead, **w**rite, ed **e**xecute vengono ripetuti per tre volte. La prima tripletta (rwx) indica il tipo di operazioni che possono essere richieste dall'**o**wner stesso; la seconda i permessi dati ai membri del gruppo (**g**roup) a cui l'owner appartiene e la terza quelli dati a tutti gli utenti (**u**sers) del sistema stesso.

Attenzione!! :

Unix non conserva copie multiple di un file.

2.2.2 File speciali

Vengono indicati con tale nome tutti i dispostivi fisici e logici di I/O : unità disco, memoria centrale, unità cassette, terminali e stampanti. Tutte le operazioni sulle periferiche vengono effettuate come se si operasse su file. I file speciali risiedono nella directory /dev .

2.2.3 Links

Unix supporta due tipi di link : un **hard link** e un **symbolic link**. Quando il primo è creato, il file ottenuto diventa indistinguibile dal file originale : se io modifico una copia anche l'altra subisce la modifica, ecc... Il symbolic link, invece, è una referenza ad un file.

2.3 Kernel

La **Kernel** è il programma di controllo, costantemente residente in memoria, che garantisce le attività di base del sistema; crea e gestisce i processi, fornisce gli strumenti per accedere ai filesystem e per le comunicazioni. È l'unica parte di UNIX che un utente non può sostituire.

2.4 Shell

È lo strumento che UNIX offre per la comunicazione fra utenti e il sistema La shell è un programma che interpreta ed espande i comandi, ridirireziona l'input e l'output dei comandi, personalizza l'ambiente di lavoro, carica in memoria i programmi richiesti, e offre un potente strumento per la programmazione: include avanzate tecniche (*if/then/else, case, while, for*) per il controllo della sequenza dell'esecuzione (*control flow*). Quando un utente si collega in un sistema UNIX, una shell (un processo) viene creata insieme ai *descrittori* di standard input (**0**), output (**1**) ed error (**2**) legati al terminale stesso. I descrittori sono numeri interi univoci assegnati dal sistema ad un elemento di un processo comunicativo.
Si possono creare file (detti procedure) che contengono comandi per adattare l'ambiente di lavoro alle proprie esigenze.

- sh (Bourne shell - Bell Lab.);
- csh (C shell - Berkeley);
- tcsh (versione della C shell - HP);
- ksh (Korn shell - AT&T).

2.5 i comandi

La "linea comando" viene scritta come sequenza di parole separate da spazi. La sintassi segue degli standard :

- "comando" ["opzione"] ["oggetto"]

La prima parola è una istruzione (per es. richiamare un compilatore, editare un file, visualizzare ...) da eseguire. Può essere personalizzata grazie l'uso di argomenti (opzioni) per operare su un oggetto (eventualmente dichiarato) il "file".

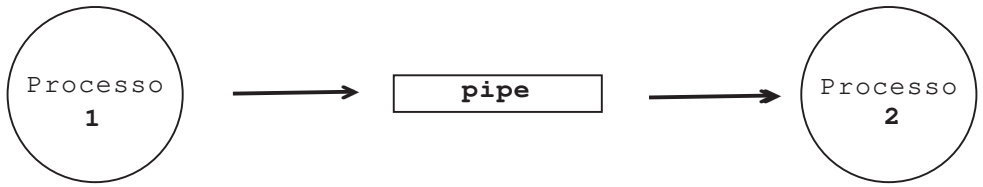
Attenzione!! :

- *La maggior parte dei comandi UNIX è costituita da parole scritte in minuscolo.*
- *I comandi non si possono abbreviare.*

2.6 Pipe

È una delle caratteristiche più interessanti e utili per gli utenti. Viene rappresentata con il simbolo | e permette l'output di un programma diventi l'input per un altro programma.

Figura 1: Pipe.




```
$> who
euro     ttyp0   May 14 11:42   (:0.0)
euro     ttyp1   May 14 11:52   (:0.0)
alex     ttyp2   May 17 09:03   (vstst0::0.0)
euro     console May 14 11:42
who => Chi è collegato ?
$> who |awk '{print $1}'
euro
euro
alex
euro
awk => Edita il flusso di input stampa il primo record
$> who | awk '{print $1}' | sort
alex
euro
euro
euro
sort => Ordina il flusso di input in modo alfabetico
$ > who | awk '{print $1}' | sort |uniq
alex
euro
uniq => Elimina le linee ripetute
```

La serie di processi dell'esempio sopra citato, in cui lo standard output di un processo è connesso allo standard input del successivo attraverso una pipe prende il nome di **pipeline**

2.7 Background & foreground

Per eseguire un comando, la shell crea (*fork( )*) normalmente un nuovo processo (*child*) **foreground** e attende che questo sia terminato. Il processo **background** è quello per il quale l'interprete dei comandi non attende il completamento dell'esecuzione e che viene "lanciato" con il metacarattere **&**.
Ogni processo che lavora interattivamente ed occupa il terminale può essere sospeso dall'utente grazie al comando *ctrl-z*.

Chi fra gli utenti VMS, "scontrandosi" con l'ostico editor **vi**, non ha sperimentato, suo malgrado, l'effetto del *ctrl-z*:

```
Cinquecento cavalieri,
con la spada sguainata
con la testa insanguinata,
indovina che cos'è
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
Stopped
csh>jobs
[1]  + Stopped              vi ritornello
```

Quando cerco di concludere la sessione, se ci sono processi stopped ottengo il messaggio "There are suspended jobs." Con il comando **fg** posso richiamarli per completarne l'esecuzione o per interromperli (*ctrl-c*). Oppure concludere direttamente l'operazione con **kill**.

2.8 Programmazione

Unix facilita la programmazione perchè prevede una lunga serie di strumenti per tale lavoro: compilatori, strumenti di formattazione, interpreti di comandi, strumenti per la costruzione di compilatori e applicazioni varie...

CAPITOLO 3

UNIX E VMS A CONFRONTO

- *interfaccia utente:*

  UNIX
  interpreti di comandi (shell) estremamente potenti e ricchi di utilities già implementati:

  — ricche strutture di controllo

  — command substitution

  — pipe

  — job control

  — ridirezione dell'I/O

  — variabili di shell La shell diventa il filtro per la comunicazione diretta tra utente e kernel non vi sono limitazioni nell'esecuzione delle istruzioni se non quelle create dalla protezione dei file stessi.

  VMS
  molte di queste caratteristiche sono assenti o più pesanti da usare

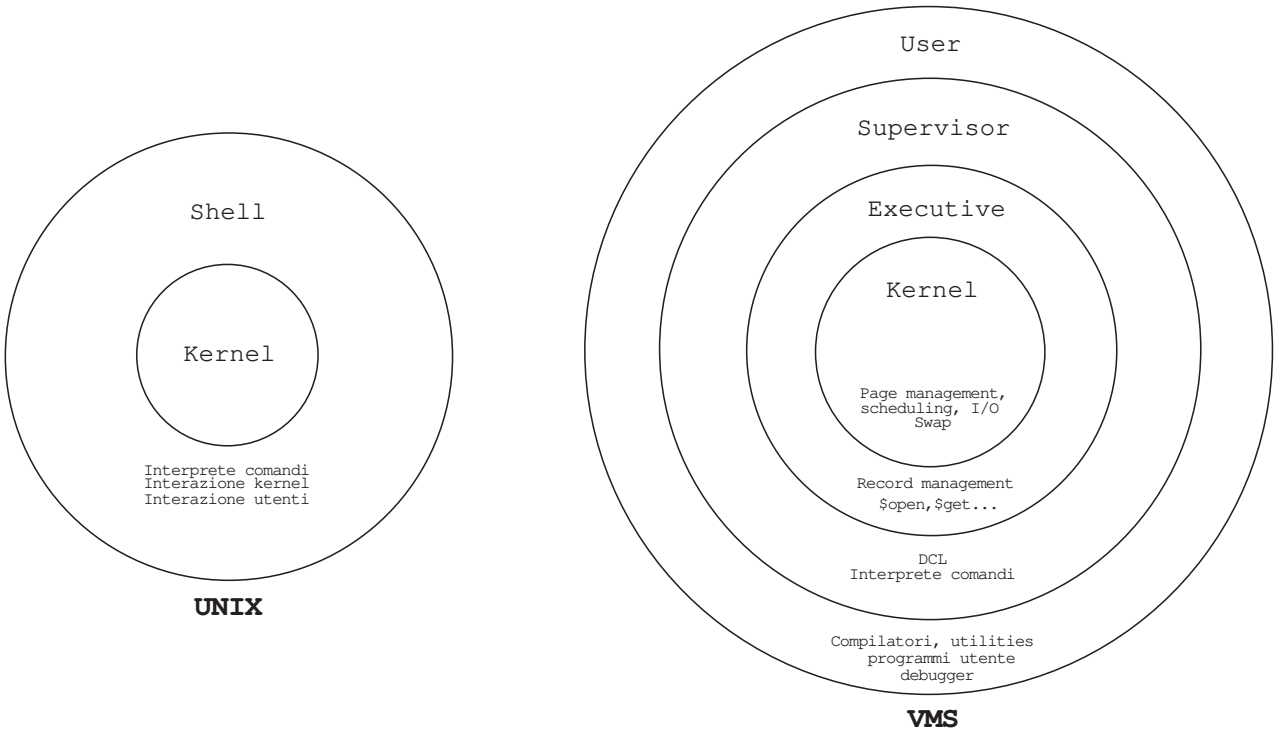
- *portabilità del Sistema Operativo*:
  UNIX dà accesso ad un maggior numero di piattaforme hardware.

- *portabilità delle applicazioni*:
  è, ovviamente, influenzata dalla portabilità del S.O. Tuttavia qui il discorso và separato tra portabilità a livello eseguibile (garantita sulle architetture VAX e ALPHA ma assolutamente assente sui sistemi UNIX che sono sviluppati su piattaforme hardware completamente eterogenee) e a livello sorgente. Qui il confronto è largamente positivo per UNIX ed in particolare per applicazioni "POSIX compiants". Anche se POSIX implementato anche su OpenVMS, è comunque nato in ambiente UNIX risentendone abbastanza a livello di sintassi e "filosofia".

- *modularità*:
  Anche qui vantaggio di UNIX. Per l'utente ciò concerne soprattutto i comandi che possono essere facilmente combinati per costruire applicazioni di notevole complessità con poco sforzo.

- *astrazione*:
  UNIX fornisce un ambiente più omogeneo del VMS dal punto di vista della "macchina virtuale" vista dall'utente attraverso il doppio filtro delle primitive di sistema e dell'interprete di comandi. Molte delle dipendenze da particolarità di hardware diversi e delle restrizioni dovute alle particolarità fisiche dei diversi devices sono completamente nascoste e trasparenti all'utente.

Figura 2: Gerarchia di accesso. UNIX-VMS



- *multiutenza* : per l'utente UNIX e VMS abbastanza simili.
- *multiprocesso* :

  UNIX
  l'esistenza del meccanismo del *back/fore*ground costituisce uno strumento molto potente per gestire in modo semplice e controllabile molti processi interattivi dello stesso user. Và però aggiunto che, tuttora, il supporto fornito da UNIX a dei veri job batch è limitato e spesso insufficiente.
  VMS
  i meccanismi dello SPAWN VMS o il RUN/DETACHED sono molto più limitati e meno semplici da usare.

- *filesystem* :

  UNIX
  gerarchico con: files non strutturati, devices visti come files, trasparentemente dalle routines di I/O, ed espandibili senza problemi (MVS IBM obbliga a l'utente a dichiarare l'occupazione massima di memoria fisica relativa ad un file all'atto della sua creazione e ad allocare esplicitamente memoria aggiuntiva in caso di necessità!!)
  VMS
  gerarchico. Sono legati al numero dei dischi ciò rende meno trasparente il supporto fisico.

- *maturità*: da questo punto di vista sia UNIX sia VMS sono in fase matura garantendo l'utente dai problemi delle versioni 1.00 (OpenVMS AXP escluso, naturalmente...)

- *ambiente di programmazione* : A livello di compilatori, text editors, linkers, debuggers c'è una sostanziale equivalenza (però con una interfaccia utente decisamente migliore della media dei debuggers UNIX per il DEBUG VMS). UNIX fornisce per default anche una serie di utilities per text processing, costruzione di compilatori, comunicazione tra processi (pipe) . Inoltre molto software pubblico per migliorare l'efficienza delle varie fasi del ciclo di vita del software sono disponibili per UNIX.

Caratteristiche negative in UNIX:

- Interfaccia a volte troppo potente per lo user inesperto
- Documentazione in_line (apropos è OK ma talvolta un help vms_like sarebbe preferibile.
- Dialetti diversi + shell diverse = babele
- Supporto debole per il batch
- La sintassi dei comandi non completamente standardizzata, le opzioni possono essere o no precedute dal carattere (-), vedi per esempio **ls**(1) e **tar**(1).

CAPITOLO 4

COME FACCIO ... IN UNIX

UNIX è un sistema operativo immenso. La seguente lista è appena la definizione dello stretto necessario per le operazioni più comuni.

1. Richiedere aiuto e/o informazioni;
   - Corso on-line di ULTRIX: **learn**
   - Informazioni sui comandi: **man, apropos, whereis**
   - Ora e data: **date**
   - Chi è connesso: **who**
2. Operare sui file;
   - Ottenere informazioni sui...: **ls**
   - Creare: **vi, cat, mkdir**
   - Visualizzare: **cat, more, pr**
   - Stampare: **lpr**
   - Copiare: **cp**
   - Modificare il nome: **mv**
   - Proteggere: **chmod, chown**
   - Ordinare: **sort**
   - Cancellare: **rm, rmdir**
3. Operare sui processi;
   - Ottenere informazione sui...: **ps**
   - Sospendere l'esecuzione **CTRL-Z**
   - Fermare l''esecuzione **kill**
4. Comunicare con gli utenti;
   - **mail**
   - **write**
5. Effettuare operazioni via rete;
   - Collegamenti remoti: **telnet, dlogin**
   - Trasferimento File: **rcp, ftp**
   - Eseguire un comando in remoto: **rsh**

6. Non tutto, ma di tutto ...
   - Modifica un file in base a comandi: **sed**
   - Filtro ed elaboratore di testi: **awk**
   - Cambiare password: **passwd**
7. E per finire ...
   - **logout**
   - **exit**
   - *ctrl-d*

APPENDICE A

BIBLIOGRAFIA

Bourne, S.R. 1985. *Il sistema Unix*. Addison Wesley P.C., Masson Italia
Deitel, H.M. 1984. *An Introduction to OPERATING SYSTEMS*. Addison-wesley Publishing company.

Sechrest, J., Krueger , J. 1991 *An Introduction to UNIX Computer Systems*. Computer Science Support Staff. September 1991. Oregon

Foxley, E. 1985. *UNIX for Super-Users*. Addison-Wesley

Leffler, S.J.,McKusick, M.K.,Karels, M.J.,Quarterman, J.S. 1989 *The Design and Implementation of the 4.3BSD Unix Operating System*. Addison-Wesley P.C.

Quartiroli,I., Fusaro, M., Smareglia, S. 1983. *UNIX Introduzione al Sistema Operativo* Clup Milano

Silvester, P.P. 1988. *The Unix System Guidebook: Second Edition*. Springer-Verlag: New York.

Yager, T., Smith, T. *Is Unix Dead ?. Byte,* vol. 17, no. 9, pp. 134-146, September 1992

APPENDICE B

UNA SESSIONE.

```
ULTRIX V4.3 (Rev. 44) (ale2ts.ts.infn.it)

login: alex
Password:
Last login: Mon Nov  8 11:11:02 from vstst0.ts.infn.it
ULTRIX V4.3 (Rev. 44) System #2: Wed Sep 29 17:38:07 WET 1993
UWS V4.3 (Rev. 10)

Wed Nov 10 10:12:42 MET 1993
ale2ts> finger alex
Login name: alex                        In real life: Alessandra Richetti
Directory: /utenti/alex                 Shell: /usr/local/tcsh
On since Nov 10 10:12:41 on ttyp3 from vstst0.ts.infn.i
Plan:
        System Manager
                                        Dip. Fis. Teorica di Trieste / INFN

ale2ts> passwd
Changing password for alex
Old password:
Enter new password:
Verify:
ale2ts> df
Filesystem         Total    kbytes   kbytes   %
node               kbytes   used     free     used  Mounted on
/dev/rz2a            18767   11468     5423    68%   /
/dev/rz1g           840718  331079   425568    44%   /usr
/dev/rz2g           111935   91374     9368    91%   /var
/dev/rz3c           191958   70277   102486    41%   /utenti
/dev/rz2d            14087    3324     9355    26%   /utenti1
/dev/rz4c           186957  181436        0   108%   /usr/lib/dxbook
ale2ts:/usr/utenti  227079  178815    25557    87%   /alexts
ale2ts> who
vstst5   tty00   Nov 10 09:32
senatore ttyp0   Nov  8 19:34   (axpts0::0.0)
pastore  ttyp1   Nov 10 08:58   (ale1ts::0.0)
euro     ttyp2   Nov 10 09:01   (:0.0)
alex     ttyp3   Nov 10 10:12   (vstst0.ts.infn.i)
euro     console Nov 10 09:00
ale2ts> finger @uts340.univ.trieste.it
[uts340.univ.trieste.it]
Login       Name              TTY Idle    When            Office
soldano  Soldano Diego         03    3 Wed 09:48
agati    Agati Paolo           p0      Wed 10:54              +39-481-32568
biblio   Dipendente Bibliotec  p1   47 Wed 09:29
ale2ts> history
1 10:40   finger alex
2 10:47   passwd
3 10:49   df
4 10:49   who
5 10:50   finger @uts340.univ.ts.it
6 10:51   history
```

```
ale2ts> !4
who
euro     ttyp2   Nov 10 09:01   (:0.0)
alex     ttyp3   Nov 10 10:12   (vstst0.ts.infn.i)
euro     console Nov 10 09:00
ale2ts> setenv
HOME=/utenti/alex
SHELL=/usr/local/tcsh
TERM=vt300
USER=alex
PATH=/usr/ucb:/bin:/usr/bin:/etc:/etc/sec:/usr/etc:/usr/etc/sec:/usr/local:/usr.
:.
SHLVL=1
PWD=/utenti/alex
LOGNAME=alex
HOST=ale2ts.ts.infn.it
HOSTTYPE=decstation
MAIL=/usr/spool/mail/alex
DECW_BOOK=/usr/lib/dxbook /usr/lib/local
EXINIT=set redraw wm=8
ale2ts> setenv DISPLAY vstst0.ts.infn.it:0.0
ale2ts> setenv DISPLAY
ale2ts> echo $DISPLAY
vstst0.ts.infn.it:0.0
ale2ts> /usr/bin/X11/xman
^Z
Suspended
ale2ts> bg
[1]    /usr/bin/X11/xman &
ale2ts> gopher
^Z
Suspended
ale2ts> jobs
[1]    Running               /usr/bin/X11/xman
[2]  + Suspended             gopher
ale2ts> apropos 'line printer'
lg02of (8)              - LG02 Matrix Line Printer
lp (1)                  - send requests to an LP line printer
lpc (8)                 - line printer control program
lpd (8)                 - line printer daemon
lpf (8)                 - general purpose line printer filter
lprm (1)                - remove jobs from line printer queue
lpv (4)                 - LPV11 parallel line printer
lpx (8)                 - line printer exerciser
print (1)               - pr to the line printer
ale2ts> man lp

                                                                        lp(1)

   Name
     lp - send requests to an LP line printer

   Syntax
     lp [-c] [-d dest] [-n number] [-] [files]

   Description
     The lp command arranges for the named files and associated information
     (collectively called a request) to be printed by a line printer.  If no
     ...
```

```
ale2ts> more /etc/printcap
# @(#)printcap  3.1 (ULTRIX) 4/20/90
lp0|lp|0|local line printer:\
        :lp=/dev/lp:\
        :of=/usr/lib/lpdfilters/lpf:\
        :sd=/usr/spool/lpd:\
        :lf=/usr/adm/lpd-errs:
lps | an LPS20 print server with a4 paper:\
     ...
ale2ts> grep print /etc/printcap
# @(#)printcap  3.1 (ULTRIX) 4/20/90
lp0|lp|0|local line printer:\
lps| an LPS20 print server postscript with a4 paper:\
lpt|LPS20 print server txt:\
#lj250| a local LJ250 DEColorwriter Companion Color printer:\
#lp0|lp|0|local line printer:\
ale2ts> man tee | lpr -Plpt
ale2ts> cat > unaltra.a
lista
^D
ale2ts> ls *.a
aa.a            prova.a         unaltra.a
ale2ts> mv *.a *.b
usage: mv [-if] f1 f2 or mv [-if] f1 ... fn d1 (‘fn’ is a file or directory)
ale2ts> foreach f (*.a)
foreach? mv $f $f:r.b
foreach? end
ale2ts> ls *.b
aa.b            prova.b         unaltra.b
ale2ts>  /usr/local/bin/archie -r ’steph.*.gif’ |tee stephgif.addr

Host gatekeeper.dec.com

    Location: /.8/graphics/data/soc.motss
           FILE -r--r--r--      94230  Mar 25 1991  stephan_hamann.gif
           FILE -r--r--r--     227795  Jul 19 1991  stephen_chappell.gif

Host ursa-major.spdcc.com
 .....
ale2ts> head stephgif.addr

Host gatekeeper.dec.com

    Location: /.8/graphics/data/soc.motss
           FILE -r--r--r--      94230  Mar 25 1991  stephan_hamann.gif
           FILE -r--r--r--     227795  Jul 19 1991  stephen_chappell.gif

Host ursa-major.spdcc.com

    Location: /pub/motss.pix/s
  ....
ale2ts> logout
```

APPENDICE C

PROGRAMMI UNIX E VMS

Questa procedura DCL ha il solo scopo di illustrare alcune delle possibilità offerte da DCL nella creazione di files di comandi. Lo scopo della procedura è di controllare se l'utente il cui nome è passato come parametro (es: @WATCH PIPPO) e correntemente collegato sul sistema. In caso contrario la procedura continua periodicamente a ripetere il controllo finchè l' utente non si colleghi o la procedura venga interrotta.

```
$! Procedura DCL
$ bell[0,8] = %x07                              ! ASCII code for bell
$ begin_loop:
$    show user /output=sys$scratch:usr.tmp      ! scrive su file l' out di sh us
$    open/read userf sys$scratch:usr.tmp        ! apre in lettura il file
$ 10:   read userf line /end_of_file=19         ! legge linea per linea
$       i = f$locate(P1,line)                   ! cerca l' argomento del comando
$                                               ! (P1) nella linea corrente
$       if i .eq. f$length(line) then goto 10   ! se non ha trovato, prossima
$                                               ! linea del file
$       write sys$output  bell                  ! EUREKA !!!!!!!!!
$       write sys$output " user ",P1," is logged in"
$       exit
$ 19:
$    close userf                                ! il file va chiuso...
$    delete/noconf sys$scratch:usr.tmp;         ! ... e cancellato
$    write sys$output " wait... "               ! per far vedere qualcosa...
$    wait 00:00:30                              ! attende 30 sec
$  goto begin_loop
```

Procedura di comandi csh (script) per svolgere lo stesso compito della precedente procedura DCL

```
# Procedura csh
begin_loop:
set watched_user = (`who | grep $1 `)  # nella variabile (in realta'
                                       # e' una wordlist) il risultato
                                       # del comando
                                       # $1 e' l' argomento del comando
while ( $#watched_user == 0 )          # se 0 elementi (user assente)...
  echo " wait... "
  sleep 30                             # ... aspettando Godot ...
  goto begin_loop
end
echo "^G"                              # FINALMENTE (nota: la stringa tra
                                       # doppi apici e' ctrl-G = ascii per bell)
echo " user " $1 " is logged in "
```

APPENDICE D

E IL CONFRONTO CONTINUA ...

Tabella 1: Comandi : VMS - UNIX

VMS	UNIX
CREATE file	cat > file
CREATE/DIRECTORY [. dirname]	mkdir dirname
CTRL/Y or CTRL/C	CTRL/C
CTRL/Z	CTRL/D
DEFINE name meaning	alias name meaning
DELETE file	rm file
DELETE/CONFIRM file	rm -i file
DIRECTORY	ls
DIRECTORY/FULL	ls -l
HELP topic	apropos topic
HELP command	man command
LOGOUT	logout
MAIL	mail to read messages, or to send mail
PRINT/QUEUE=queue file	lpr -P printer file
RENAME file1 file	mv file1 file2
RUN [dirname]program	path/program
SEARCH file string	grep expression file
SET DEFAULT dirname	cd dirname"
SHOW DEFAULT	pwd
SHOW TIME	date
SHOW USERS	who
TYPE file	cat file
TYPE/PAGE file	more file
* (wildcard)	*
% (wildcard)	?
-	\
LOGIN.COM (file)	.login

GLOSSARIO

& (ampersand): Metacarattere utilizzato per iniziare un processo in background

*** (asterisco)**: Metacarattere utilizzato per rappresentare ogni stringa di zero o più caratteri

\ **(backslash)**: Metacarattere utilizzato per impedire alla shell l'interpretazione di tutti i metacaratteri.

? (Punto esclamativo): Metacarattere utilizzato per rappresentare ogni singolo carattere.

/ **(slash)**: Il simbolo che rappresenta la **root** directory. Viene utilizzato anche come separatore delle parti che compongono il pathname.

~ (tilde): Metacarattere utilizzato per rappresentare la **home** directory

| **(pipe)**: Metacarattere utilizzato per indicare alla shell di utilizzare l'output di del comando1 come input per un comando2.

comando1 | comando2

<: Metacarattere utilizzato per indicare alla shell di utilizzare un file come input per un comando.

comando < file

>: Metacarattere utilizzato per indicare alla shell di ridirigere l'output di comando su un file.

comando > file

absolute pathname: Direttory espressa in relazione alla radice della struttura gerarchica.

alias: Il nome alternativo per un comando o per una linea comando.

alias vt100 "set term = vt100"

Argomento: La maggior parte dei comandi Unix ricevono una lista di "oggetti" su cui operare:

ls file1 file2

il comando consiste nell'istruzione (ls) e di due argomenti: file1, file2.

Assembler: Linguaggio di programmazione con istruzioni a basso livello immediatamente comprensibili alla macchina. Ad ogni istruzione macchina corrisponde una e una sola istruzione assebler.

Background: È un comando per cui l'interprete dei processi (la shell) non aspetta l' ultimazione.

Bit: La più piccola unità di misura dell'informazione. Può assumere il valore di 1 o 0.

Blocco: Nel file system, è l'unità di lettura o scrittura di una memoria di massa. (512 bytes in Unix)

Bourne shell: Uno degli interpreti di comandi disponibile in UNIX. Sviluppato da Bell Lab.

Byte: Unità di misura elementare applicabile alla memoria, corrisponde per loppiù a 8 bit. Un carattere è rapresentabile con un byte.

C shell: Un interprete di comandi, con particolare strutture di controllo, disponibile in UNIX. Sviluppato dall'università di Berkeley.

Comando: Un programma del sistema operativo.

Compilatore: Strumento per la traduzione

CPU: (Central processing unit) È l'unità primaria di elaborazione. È in grado di eseguire operazioni elementari.

.cshrc: File di comandi per modificare le caratteristiche di lavoro della shell.

CTRL-C: Abortisce il processo corrente.

CTRL-D: Segnala la conclusione dei dati.

CTRL-Z: Ferma temporaneamente il processo corrente.

Current directory: È la directory dalla quale i *relative pathnames* sono interpretati dal processo

descrittore: Sono numeri interi univoci asseganti dal sistema ad un elemento di un processo comunicativo(files, pipes e sockets).

directory: File speciale che contiene i riferimenti di altri files. Per convezione contiene le informazioni di almeno due file riferiti come (.) e (..) : il direttorio stesso e il padre.

editor: Programma che permette di creare e di modificare i file di testo.

Escape: Viene utilizzato dall'editor *vi* come segnalatore di fine inserzione. Si può utilizzare come Escape il CTRL/[.

file speciali: In UNIX i devices vengono visti e trattati come files.

file system: Struttura gerarchica composta da file e directory

filtro: Programma che riformatta e rimuove informazioni non desiderate dall'input e scrive sullo standard output. Per es. *grep, sort, awk, sed*.

foreground: Processo il cui input ed output sono legati al terminale.

gruppo: Unix considera gli utenti suddivisi in gruppi che svolgano attività affini. Viene utilizzato per regolamentare l'accesso ai files.

hardware: Indica la parte *fisica* del calcolatore.

help: Per ottenere aiuto dal sistema in UNIX si possono utilizzare comandi come *apropos, man, whereis*

home directory: È la current directory al momento dell'accesso al sistema.

Kernel: Il nucleo del sistema operativo che risiede sempre in memoria.

Korn shell: Uno degli interpreti di comandi di UNIX. È una versione estesa della Bourne shell che include molte caratteristiche della C shell.

.login: File di comandi per modificare le caratteristiche di lavoro della shell

logging in: Procedura di identificazione per iniziare una sessione di lavoro su un sistema operativo.

logging out: Il concludere una sessione di lavoro.

memoria di massa: Memoria su cui risiedono permanentemente i file per es. dischi.

pipe: Meccanismo con cui UNIX permette il flusso di dati tra processi asincroni.

processo: Unità di esecuzione identificata da un numero (*ID*).

programma: File che contiene una serie di comandi per il Sistema Operativo.

prompt: Simbolo o messaggio che cui il sistema segnala la disponibilità a ricevere il prossimo comando.

ridirezione: Unix offre un meccanismo di ridirezione dell'input e dell'output da e su altri files al posto del terminale

root: Il vertice della struttura gerarchica del file system.

shell: Il programma adibito al controllo delle interazioni dell'utente con il sistema operativo. Interprete dei comandi.

socket: Una delle estremità del processo comunicativo.

software: Programmi in genere, dal sistema operativo ai programmi utente.

Standard error: Canale di comunicazione utilizzato per trasmettere eventuali messaggi di errore di un processo. Solitamente è il terminale associato al processo.

Standard input: Canale di comunicazione utilizzato per ricevere eventuali dati dal terminale associato ad esso.

Standard output: Canale di comunicazione utilizzato per trasmettere eventuali dati di un processo. Solitamente è il terminale associato al processo.

Super user: È l'utente *privilegiato* addetto alla manutenzione e alla gestione del sistema.

ULTRIX: Digital Equipment Corporation's version of the UNIX operating system.

INDICE ANALITICO

&
    background, 7
/
    root, 3

A

astrazione, 9

B

background, 7, 10
batch, 10

C

child, 7
comandi, 4, 6, 11
csh, 17
ctrl-z, 7

D

descrittori, 5
directory, 4
direttorio corrente, 3, 4

E

error
  standard, 5

F

file, 4, 10
    ordinari, 4
    proprietà, 4
    protezione, 3, 4
    speciali, 4, 5
filesystem, 3, 10
    locale, 3
    NFS, 3
finger, 17
foreground, 7, 10
fork( ), 7

G

group, 5

H

home directory, 4

I

input, 7
    standard, 5
interfaccia utente, 9

J

job, 10
jobs, 7

K

kernel, 5

L

link, 5

M

modularità, 9
multiprocesso, 10
    Sistema operativo, 3
multiutente
    Sistema operativo, 3
multiutenza, 10

O

output, 7
    standard, 5
owner, 5

P

password
    passwd, 17
pathname, 3
    absolute, 3
    relative, 3

pipe, 11
    |, 6
portabilità, 9
processo
    protezione, 3
programmazione, 7, 11

R

root, 3

S

shell, 5, 9

U

user, 5
    login, 17
username, 3