

Sistemi Operativi (Laboratorio)

Lorenzo Martignoni

Dipartimento di Informatica e Comunicazione
Università degli Studi di Milano, Italia
lorenzo@security.dico.unimi.it

a.a. 2008/09

Lezione III: Shell 2

Settimana prossima le lezioni saranno invertite:

- ▶ Lunedì 23, 18:30–21:30, Laboratorio
- ▶ Venerdì 27, 18:30–21:30, Teoria

Un vero linguaggio di programmazione

La shell è un vero (Turing-completo) linguaggio di programmazione (interpretato)

- ▶ Variabili (create al primo assegnamento, uso con \$, **export** in un'altra shell). `x="ciao"; y=2 ; /bin/echo "$x $y $x"`
- ▶ Istruzioni condizionali (valore di ritorno 0 \rightsquigarrow true)
`if /bin/ls piripacchio; then /bin/echo ciao; else /bin/echo buonasera; fi`
- ▶ Iterazioni su insiemi `for i in a b c d e; do /bin/echo $i; done`
- ▶ Cicli

```
/usr/bin/touch piripacchio
while /bin/ls piripacchio; do
  /usr/bin/sleep 2
  /bin/echo ciao
done & ( /usr/bin/sleep 10 ; /bin/rm piripacchio )
```

Un vero linguaggio di programmazione

La shell è un vero (Turing-completo) linguaggio di programmazione (interpretato)

▶ Variabili (create al primo assegnamento, uso con \$, **export** in un'altra shell). `x="ciao"; y=2; /bin/echo "$x $y $x"`

▶ Istruzioni condizionali (valore di ritorno 0 \rightsquigarrow true)

`if /bin/ls piripacchio; then /bin/echo buonasera; fi`

▶ Iterazioni su insiemi `man sh` `echo $i; done`

▶ Cicli

`/usr/bin/touch piripacchio`

`while /bin/ls piripacchio; do`

`/usr/bin/sleep 2`

`/bin/echo ciao`

`done & (/usr/bin/sleep 10 ; /bin/rm piripacchio)`

Il primo programma

```
#!/bin/sh
```

```
for i in /etc/*  
do  
    echo $i  
done
```

Il primo programma

```
#!/bin/sh
```

```
for i in /etc/*  
do  
    echo $i  
done
```

```
sh prog.sh
```

Il primo programma

```
#!/bin/sh
```

```
for i in /etc/*  
do  
    echo $i  
done
```

```
sh prog.sh
```

```
chmod +x prog.sh  
./prog.sh
```


Bill Joy (co-fondatore della SUN), 1976, per BSD UNIX

- ▶ Modal editor
 - ▶ modo input
 - ▶ modo comandi
- ▶ I comandi di movimento e modifica sono sostanzialmente ortogonali
- ▶ small and fast
- ▶ fa parte dello standard POSIX

vi in una slide

- ▶ Modalità comandi: (ESC)
 - ▶ :w [filename] save file :wq save & exit
 - ▶ :e filename open file
 - ▶ :q exit
- ▶ Modifica:
 - ▶ i, a insert before/after
 - ▶ o, O add a line
 - ▶ d, c, r delete, change, replace
 - ▶ y, p “to yank” and paste
 - ▶ u undo . redo
 - ▶ s/reg/rep/[g] search and replace
- ▶ Movimento:
 - ▶ h, j, k, l (o frecce)
 - ▶ 0, beginning of line, \$, end of line
 - ▶ w, beginning of word, e, end of word
 - ▶ (num)G, goto line num, /, search
 - ▶ (,), sentence

vi in una slide

▶ Modalità comandi: (ESC)

```
#!/bin/sh
cmdname=`basename "$0"`
MOZ_DIST_BIN=`dirname "$0"`
MOZ_DEFAULT_NAME="./${cmdname}-bin"
MOZ_APPRUNNER_NAME="./mozilla-bin"
MOZ_VIEWER_NAME="./viewer"
MOZ_PROGRAM=""

exitcode=0
#####
##
moz_usage()
{
echo "Usage:  ${cmdname} [options] [program]"
echo ""
echo "  options:"
echo ""
echo "    -g                Run in debugger."
echo "    --debug"
echo ""
:w /tmp/script.sh
```

- ▶ (num)G, goto line num, /, search
- ▶ (,), sentence

Input e Output

In generale il paradigma UNIX permette alle applicazioni di fare I/O tramite:

Input

- ▶ Parametri riga di comando
- ▶ Variabili *d'ambiente*
- ▶ File (tutto ciò che può essere gestito con le syscall `open`, `read`, `write`, `close`)
 - ▶ Terminale (interfaccia testuale)
 - ▶ Device (per es. il mouse potrebbe essere `/dev/mouse`)
 - ▶ Rete (socket)

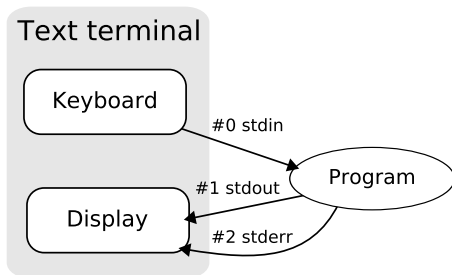
Output

- ▶ Valore di ritorno
- ▶ Variabili *d'ambiente*
- ▶ File (tutto ciò che può essere gestito con le syscall `open`, `read`, `write`, `close`)
 - ▶ Terminale (interfaccia testuale)
 - ▶ Device (per es. lo schermo in modalità grafica potrebbe essere `/dev/fb`)
 - ▶ Rete (socket)

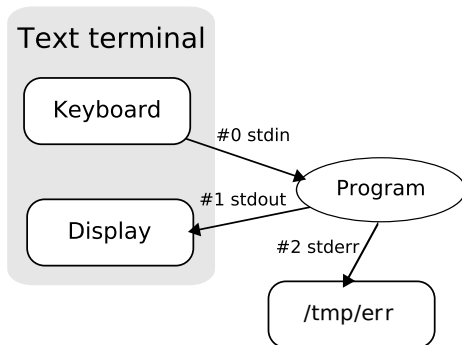
Standard input, standard output e standard error

Ad ogni processo sono sempre associati tre file (già aperti)

- ▶ **Standard input** (Terminale, tastiera)
- ▶ **Standard output** (Terminale, video)
- ▶ **Standard error** (Terminale, video, usato per le segnalazione d'errore)



Standard input, output e error possono essere **rediretti**:

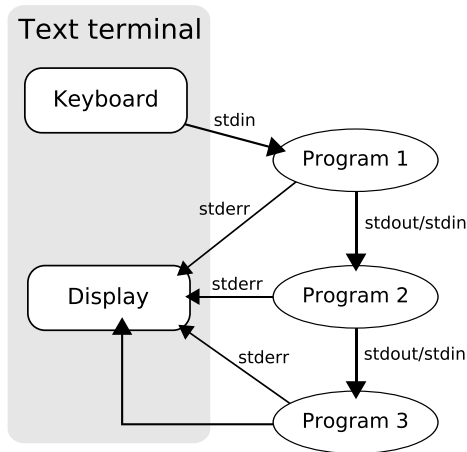


Redirezione (alcuni esempi)

- ▶ `/usr/bin/sort < lista` (lo `stdin` è il file `lista`)
- ▶ `/bin/ls > lista` (lo `stdout` è il file `lista`)
- ▶ `/bin/ls >> lista` (lo `stdout` è il file `lista`, ma non viene sovrascritto)
- ▶ `/bin/ls piripacchio 2> lista` (lo `stderr` è il file `lista`)
- ▶ `(echo ciao & date ; ls piripacchio) 2> errori 1>output`

Pipe

La **pipe** è un canale, analogo ad un file, bufferizzato in cui un processo scrive e un altro legge. Con la shell è possibile collegare due processi tramite una pipe anonima.



Pipe (alcuni esempi)

Lo stdout del primo diventa lo stdin del secondo:

- ▶ `/bin/lis | sort`

- ▶ `ls -lR / | sort | more`

funzionalmente equivalente a

`ls -lR >tmp1; sort <tmp1 >tmp2; more<tmp2; rm tmp*`

Molti programmi copiano lo stdin su stdout dopo averlo elaborato:
sono detti **filtri**.

Command substitution

Con una pipe è possibile “collegare” lo stdout di un programma con lo stdin di un altro.

Per usare l'output di un programma sulla riga di comando di un altro programma, occorre usare la **command substitution**:

- ▶ `/bin/ls -l $(/usr/bin/which sort)`
- ▶ `/bin/ls -l '/usr/bin/which sort'`
- ▶

```
for f in `find /etc -type f`  
do  
    echo $f  
    stat $f  
done
```

Settimana prossima le lezioni saranno invertite:

- ▶ Lunedì 23, 18:30–21:30, Laboratorio
- ▶ Venerdì 27, 18:30–21:30, Teoria

1. Verificare qual è il valore di ritorno di una pipe, anche in caso che qualcuno dei “filtri” fallisca.
2. Scrivere una *pipeline* di comandi che identifichi il le informazioni sul processo `/bin/floppy` (`ps`, `grep`)
3. Scrivere una *pipeline* di comandi che identifichi il solo processo che occupa più spazio in memoria (`ps`, `sort`, `tail`)
4. Ottenere il numero totale dei file contenuti nelle directory `/usr/bin` e `/var` (`ls`, `wc`, `bc`)
5. Si immagini di avere un file contenente il sorgente di un programma scritto in un linguaggio di programmazione in cui i commenti occupino intere righe che iniziano con il carattere `#`. Scrivere una serie di comandi per ottenere il programma senza commenti. (`grep`)
6. Ottenere la somma delle occupazioni dei file delle directory `/usr/bin` e `/var`

- ▶ “A Brief Introduction to Unix (With Emphasis on the Unix Philosophy)”, Corey Satten
<http://staff.washington.edu/corey/unix-intro.pdf>
- ▶ http://en.wikipedia.org/wiki/Unix_philosophy
- ▶ “The UNIX Time-Sharing System”, Ritchie; Thompson
<http://www.cs.berkeley.edu/~brewer/cs262/unix.pdf>

Ad ogni file vengono associati dei **permessi**, che definiscono le azioni permesse sui dati del file

- ▶ **Read:** leggere il contenuto del file o directory
- ▶ **Write:** scrivere (cambiare) il file o directory
- ▶ **eXecute** eseguire le istruzioni contenute nel file o accedere alla directory

R	W	X	
1	1	0	6
1	0	1	5
1	0	0	4
1	1	1	7

I permessi possono essere diversi per 3 categorie di utenti del sistema:

- ▶ **User:** il “proprietario” del file
- ▶ **Group:** gli appartenenti al gruppo proprietario
- ▶ **All:** tutti gli altri

Agire sui permessi

- ▶ Cambiare il proprietario
 - ▶ `chown utente[:gruppo] file`
- ▶ Cambiare il gruppo
 - ▶ `chgrp gruppo file`
- ▶ Cambiare i permessi
 - ▶ `chmod 755 file`
 - ▶ `chmod +x file`
 - ▶ `chmod a=rw file`
 - ▶ `chmod g-x file`
- ▶ (per creare un utente: `adduser`)

Il proprietario di un processo in esecuzione è normalmente **diverso** dal proprietario del file contenente un programma (e diverso ad ogni esecuzione)

- ▶ effective UID bit: il processo assume come proprietario il proprietario del file del programma
- ▶ SUID root
- ▶ `chmod 4555 file`
- ▶ `chmod u+s file`



DIJKSTRA, E. W.

My recollections of operating systems design.

[http:](http://www.cs.utexas.edu/users/EWD/ewd13xx/EWD1303.PDF)

[//www.cs.utexas.edu/users/EWD/ewd13xx/EWD1303.PDF](http://www.cs.utexas.edu/users/EWD/ewd13xx/EWD1303.PDF),
2001.

EWD-1303.



TANENBAUM, A., AND WOODHULL, A.

Operating Systems Design and Implementation, III ed.

Prentice Hall, 2006.

© 2009 Mattia Monga & Lorenzo Martignoni

Creative Commons Attribuzione-Condividi allo stesso modo 2.5
Italia License.

<http://creativecommons.org/licenses/by-sa/2.5/it/>.

Immagini tratte da [2] e da Wikipedia.