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 II: Shell

The UNIX Philosophy

1. Small is beautiful
2. Make each program do one thing well
3. Build a prototype as soon as possible
4. Choose portability over efficiency
5. Store data in flat text files
6. Use software leverage to your advantage
7. Use shell scripts to increase leverage and portability
8. Avoid captive user interfaces
9. Make every program a filter

Mike Gancarz – 1994

Shell

La shell è l'interprete dei comandi che l'utente dà al sistema operativo. Ne esistono grafiche e testuali.

In Minix, il default è una shell testuale `ash`, che fornisce i costrutti base di un linguaggio di programmazione (variabili, strutture di controllo) e primitive per la gestione dei processi e dei file.

Shell (pseudo codice)

```
1  while (1) { /* repeat forever */
2          type_prompt(); /* display prompt on the screen */
3          read_command(command, parameters); /* read input from terminal */
4          if (fork() > 0){ /* fork off child process */
5                  /* Parent code. */
6                  waitpid(1, &status, 0); /* wait for child to exit */
7          } else {
8                  /* Child code. */
9                  execve(command, parameters, 0); /* execute command */
10         }
11 }
```

Lanciare programmi con la shell

- Per iniziare l'esecuzione di un programma basta scrivere il nome del file
  - /bin/ls
- Il programma è trattato come una funzione, che prende dei parametri e ritorna un intero
  (`int main(int argc, char*argv[])`). Convenzione: 0 significa "non ci sono stati errori", > 0 errori (2 errore nei parametri), parametri - $\rightsquigarrow$ opzioni
  - /bin/ls /usr
  - /bin/ls piripacchio
- Si può evitare che il padre aspetti la terminazione del figlio
  - /bin/ls /usr &
- Due programmi in sequenza
  - /bin/ls /usr ; /bin/ls /usr
- Due programmi in parallelo
  - /bin/ls /usr & /bin/ls /usr

Man

La documentazione dei principali comandi del sistema è inclusa in Minix.

Il comando `man` permette di consultare il manuale di un particolare comando:

- `man man`
- `man sh`

Digressione: `vi`

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

Salvare un file e uscire `wq`

- Modalità comandi: ESC
- 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

Esercizi: fork, exec, wait, stdin, stdout, sterr, argv

1. `stdin.c`
2. `stdout.c`
3. `stderr.c`
4. `exit.c`

Esercizi: fork, exec, wait, stdin, stdout, sterr, argv

1. `stdin.c`
2. `stdout.c`
3. `stderr.c`
4. `exit.c`

5. `fork.c`
6. `exec.c`
7. `wait.c`

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 )
```

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 700 prog.sh
$ ./prog.sh
```

Esercizi

1. Per ciascuno dei file `dog`, `cat`, `fish` controllare se esistono nella directory `bin` (hint: usare `/bin/ls` e nel caso scrivere ``` ``Trovato'' ```)
2. Consultare il manuale (programma `/usr/bin/man`) del programma `/bin/test` (man **test**)
3. Riscrivere il primo esercizio facendo uso di `test`

Bibliografia

Dijkstra, E. W.
My recollections of operating systems design.
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.

Copyright

Immagini tratte da [2] e da Wikipedia.