L'interprete i comandi non e` difficile come si immagina

Alessandro Rubini

<rubini@linux.it> <rubini@gnudd.com> <rubini@unipv.it>

libero professionista

circuiti stampati e affini

device driver e firmware per microcontrollori

professore a contratto in hardware and software codesign per l'anno accademico 2023-2024

Mantova, 13 Aprile 2024

La shell non puzza come l'ascell

Alessandro Rubini

<rubini@linux.it> <rubini@gnudd.com> <rubini@unipv.it>

libero professionista

circuiti stampati e affini

device driver e firmware per microcontrollori

professore a contratto in hardware and software codesign per l'anno accademico 2023-2024

Mantova, 13 Aprile 2024

Partiamo dalle origini

Ogni sistema informatico necessita di un'interfaccia di controllo

La cosa piu` facile per gli umani e` comunicare in forma testuale

La cosa piu` facile per un programmatore e` dare nomi alle azioni

Alcune azioni possono aver bisogno di uno o piu` parametri

Ed ecco quindi che un pezzo alla volta nasce l'interprete di comandi

```
ls -l /home/rubini
cp tesi.odt backup.odt
sleep 2
echo ok
```

Il mondo dei microcontrollori (e boot loader)

I sistemi informatici piu` piccoli sono ancora rimasti a quello:
alcuni comandi con argomenti,
un 'ambiente' dove salvare parametri globali e stato.

Comandi:

```
bootm zImage

echo    'Loading Linux'
linux   /boot/vmlinuz-5.10.0 ro  quiet
```

Stato:

```
setenv console=ttyAMA0
setenv bootargs console=${console} root=/dev/sda3
```

Il mondo dei "sistemi operativi"

La piu` grande differenza e` il concetto di processo

- I processi (task) esistono anche in forme piu` semplici
- _questo_ processo e` qualcosa di molto piu` grande

Ma cos'e` un processo?

Il processo e` una piccola macchina virtuale

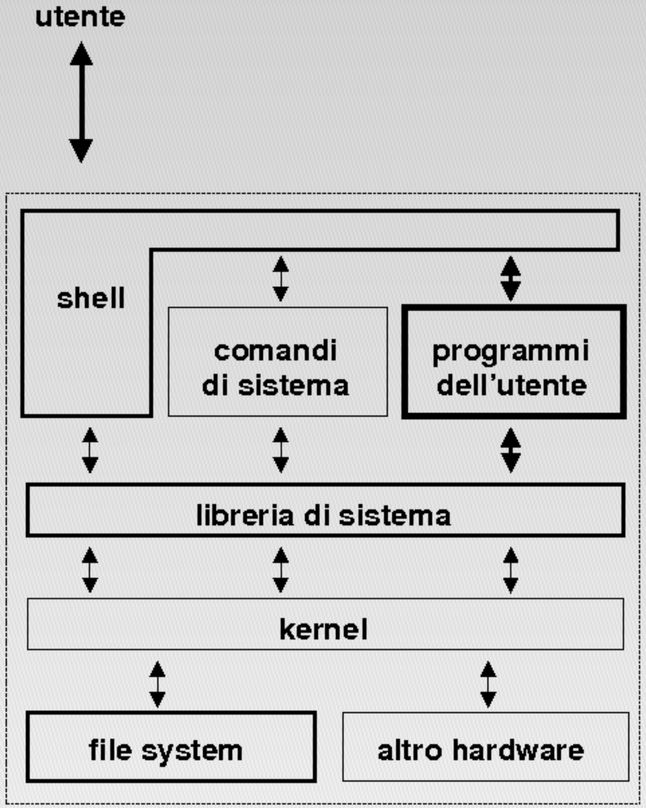
- E` un flusso di calcolo che gira autonomamente
- E` collegato con i suoi canali di input/output
- Ha il suo accesso personale allo spazio dati (filesystem)
- Ha il suo stato (environment) e i suoi attributi

Questo ovviamente apre un sacco di possibilita`

- La potenza del sistema e` decuplicata
- E molto potere richiede molta responsabilita`

L'interprete di comandi diventa quindi un gestore di processi

Da dove vengono "shell" (conchiglia) e "bash"



Una volta il sistema si disegnava cosi`

Oppure arrotolato con la shell tutto attorno
E il kernel (nucleo) nel mezzo

Quindi shell voleva dire "interfaccia utente"

Oggi invece sta per "interprete di comandi"

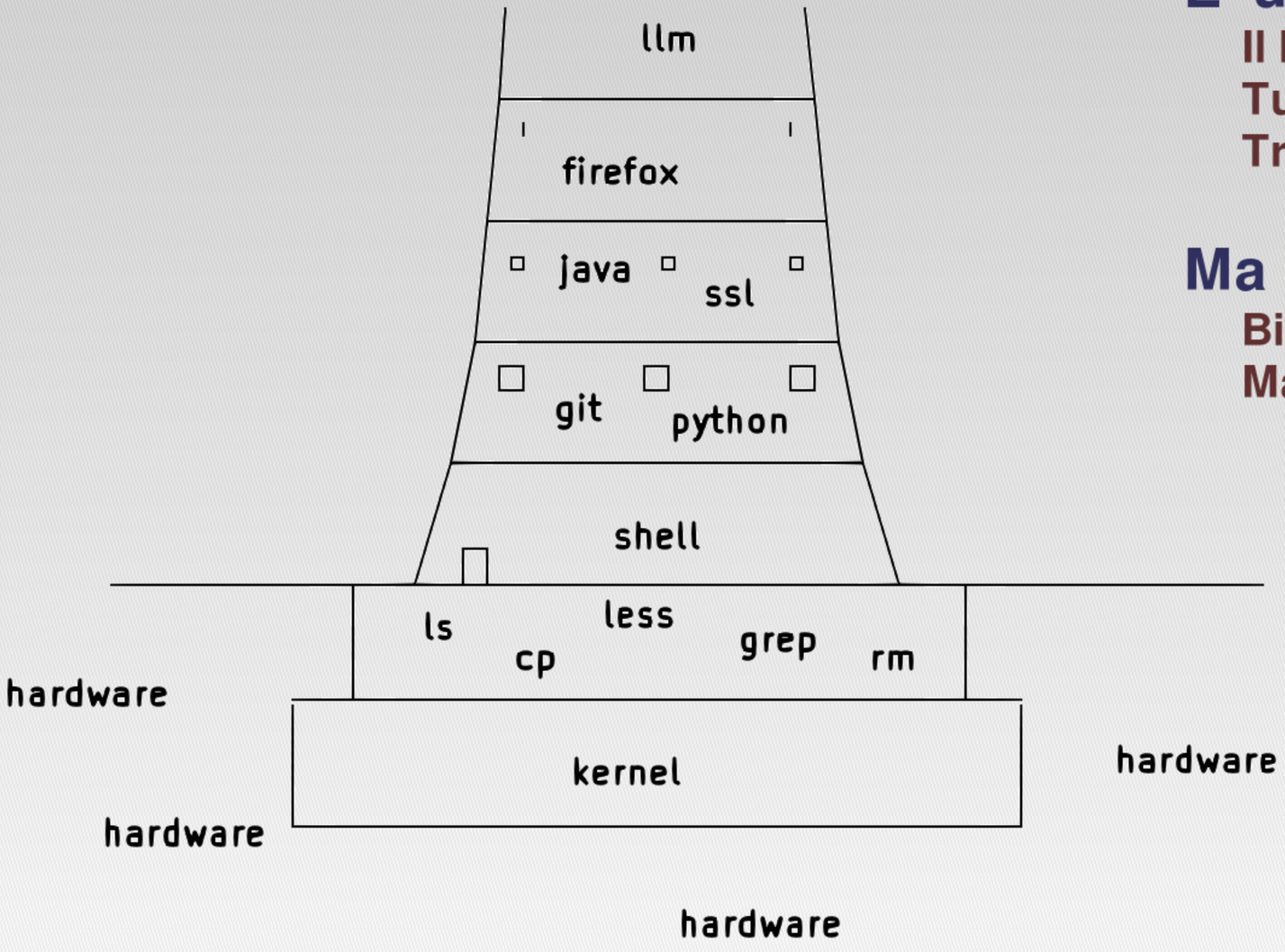
La shell di Unix e` stata scritta da Steve Bourne

Si chiamava "bourne shell" per distinguerla dalle altre

La shell di GNU e` compatibile, ma riscritta

Non poteva che chiamarsi "bourne again shell"

Una diversa rappresentazione del sistema



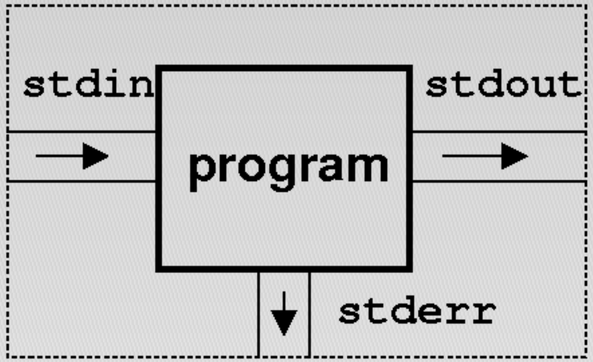
E` una torre di babele

- Il livello shell e` il piano terra
- Tutto ci si appoggia sopra
- Tranne quello che sta sotto

Ma il piano terra ha la porta

- Bisogna entrare da lì, per forza
- Ma si puo` anche uscire, per fortuna

Il processo e stdio



Il processo, o programma, ha tre canali aperti

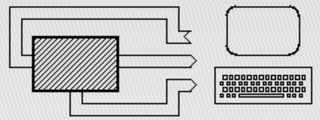
E` la shell, normalmente, che li predispone

E` possibile quindi redirigere input, output, error

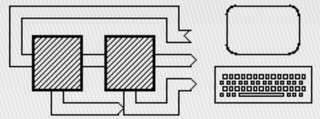
```
ls -l /tmp > tmplist
grep meteo < tmplist
find /etc -name ssh 2> /dev/null
```

Normalmente stdin/out/err sono collegati al terminale

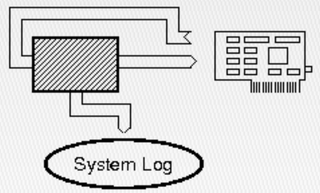
Ma si possono pensare cose molto piu` interessanti



Normal operation



Pipeline



Network Connection through sockets

La pipe (il tubo)

Collegando output e input di piu processi otteniamo grandi risultati

- Ogni programma fa una cosa, una sola, e la fa bene
- L'utente puo` collegare tanti mattoncini per ottenere lo scopo desiderato

Non lamentiamoci se in Linux ci sono troppi comandi

- Sono solo mattoncini di colore diverso, non serve usarli tutti
- La cosa che e` utile sapere usare, invece, sono malta e cazzuola

La forma piu` facile di comunicazione tra processi:

- La riga di comando per configurare il funzionamento
- Il valore di ritorno per comunicare successo o errore
- stdin e stdout per il passaggio dei dati

```
du | sort -n | tail
ls -l | sort -n -k 5
locate div64.c | xargs md5sum | sort
```

Comandi piu` importanti

cat

conCATena vari file (elemento neutro dell'operatore pipe)

grep

Global Regular Expression Printer, estrae righe di testo

ls, cp, mv, rm, chmod, chown, ln, mknod, mkfifo df, du, wc

gestione dei file, loro metadati, dimensione

echo, pwd, touch, yes, true, false, tee

vari programmi banali ma utili

file

mostra il tipo di un file (il nome e` ininfluente)

less

paginatore, per gestire un file lungo a schermo (less is better than more)

dd

data dump (?), per trasferire blocchi di dati

test, [

valutazione di condizioni

man

il manuale, per poter dimenticare quello che non serve

Dispositivi piu` importanti

/dev/null

Il buco nero (mangia tutto, vuoto in lettura)

/dev/zero

Sorgente di zeri (mangia tutto anche lui)

/dev/random

Sorgente di numeri casuali

/dev/urandom

Sorgente piu` veloce ma meno buona

/dev/tty

Il terminale di controllo del processo corrente

/dev/mem

La memoria fisica (accessibile solo all'amministratore)

/dev/stdin

/dev/stdout

/dev/stderr

Condizioni

L'oggetto di interesse della shell e` il processo

Percio`, ovviamente, la condizione e` il successo o meno di un comando

```
if grep -q $USER /etc/passwd; then echo esisto; fi
```

Ecco quindi perche` [ e` sinonimo di test (che valuta condizioni)

```
if [ -d /home/$USER ]; then echo "ho una casa"; fi
```

Ed ecco perche` devono essereci spazi attorno alle quadre

La parentesi quadra non e` sintassi di shell, e` un comando

Cicli

L'altro oggetto di interesse e` il nome di file

Percio`, ovviamente, la "for" cicla su un insieme di stringhe

```
for n in *.c; do a2ps -4 $n > /tmp/$n.ps; done
```

Il ciclo while, invece, ri-valuta nel tempo un comando

```
while read a; do echo $(date +%H:%M:%S) $a; done
```

Per completezza, c'e` la "switch" che si chiama case (case/esac)

Ma e` cosi` brutta che non la faccio vedere.

Variabili e variabili di ambiente

La configurazione e lo stato stanno in variabili "di shell"

- PS1/PS2: "prompt string" 1 e 2
- ... e mille altre se usate bash (che vi mazzuola)

Le variabili si possono usare anche nei programmi e cicli (vedi sopra)

- Si tratta comunque di valori interni all'interprete di comandi

Le variabili di ambiente (environment) sono gestite dal sistema

- Anche queste sono stringhe con un nome alfanumerico
- Pero` in questo caso si tratta di attributi del processo
- Ogni processo, alla sua creazione, eredita tutti gli attributi del processo madre
- Ma le variabili di shell sono interne, il sistema operativo non le vede

Percio` "export" dice alla shell di esportare una variabile al sistema

- Si tratta di una sorta di promozione, necessaria per passare la stringa ai processi figlio
- Prima si assegna la variabile, al solito modo, e poi la si promuove
- Facile e semplice, quando si conosce la ragione

Gli script

Normalmente con l'interprete di comandi si risolvono problemi

- E posso aspettarmi che il problema si ripeta in futuro
- Quindi e` normale copiare su un file di testo quanto realizzato

Lo script e` nato come "trascrizione" di una serie di comandi

- Se poi attivo il permesso di esecuzione, ecco un nuovo comando utile

Il sistema riconosce un programma eseguibile da un testo

- L'eseguibile viene caricato in memoria, il testo viene passato alla shell

Ma col tempo sono nati altri linguaggi che interpretano del testo

- Sarebbe stato bello rendere eseguibile questi testi

La convenzione "#!" (shebang) permette di dichiarare un interprete

- Ogni testo eseguibile dichiara chi deve interpretarlo
- Il sistema chiama l'interprete, passandogli lo script come argomento
- Dopo di che "#" e` carattere di commento in ogni linguaggio, quindi #! viene ignorato

Worse is better (Richard P. Gabriel, 1989)

Simplicity

The design must be simple, in implementation and if possible in interface.
Simplicity is the most important consideration in a design.

Correctness

The design must be correct in all observable aspects.
"Simpler" must be preferred to "correct".

Consistency

The design must not be overly inconsistent.
Consistency can be sacrificed for simplicity in some cases.

Completeness

The design must cover as many important situations as is practical.
Completeness can be sacrificed in favor of any other quality.

Other useful rules

Follow the KISS rule (Keep it simple, stupid)
First make it work, then make it work well (if time allows)

Informazione e conoscenza

Alessandro Rubini
programmi e soluzioni GNU
Corso Carlo Alberto 38, Pavia

+39 0382-529.554
+39 349-26.89.041

rubini@gnudd.com
rubini@linux.it
rubini@unipv.it

PCB and hw/sw design
embedded systems
device drivers
courses

PowerPC
ARM
AVR
Cortex
LM32
ColdFire
Cris
M68k
SPARC64
Alpha
SPARC
RiscV
MIPS
x86-64
i386

AD0A BCC7 31BE AE01 9FD9 38FF E857 F9F7 210D 0374

```
.data
        2 etti di farina bianca
        2 etti di farina gialla
        2 etti di zucchero
        2 etti di burro
        2 etti di mandorle tritate
        2 tuorli

.text
        amalgamare il tutto
        sbriciolare nella teglia
        cuocere 1 ora a 180 gradi
        spruzzare di lassativo

.bss
        terrina
        tortiera
        forno
```

```
echo mem | sudo tee -a /sys/power/state
```