



Sistemi Operativi¹

Mattia Monga

Dip. di Informatica e Comunicazione
Università degli Studi di Milano, Italia
mattia.monga@unimi.it

a.a. 2008/09

¹ © 2009 M. Monga. Creative Commons Attribuzione-Condividi allo stesso modo 2.5 Italia License.
<http://creativecommons.org/licenses/by-sa/2.5/it/>. Immagini tratte da [?] e da Wikipedia.



Lezione XIV: Unix power tools e primi esperimenti col kernel



find

Per selezionare file con determinate caratteristiche si usa `find`
`find percorso predicato`
Seleziona, nel sottoalbero definito dal percorso, tutti i file per cui il **predicato** è vero (ce ne sono moltissimi di base che possono essere combinati con i connettivi logici: vedi manuale)
`find /usr/src -type f -name "*.c"`



xargs

`find` è spesso usato insieme a `xargs`
`find percorso predicato | xargs comando`
funzionalmente equivalente a
comando `$(find percorso predicato)`
ma evita i problemi di lunghezza della riga di comando perché `xargs` si preoccupa di “spezzarla” opportunamente.

Un utile tecnica



Sistemi Operativi

Bruschi Martignoni Monga

Unix power tools
find
Archivi
Segnali

Primi esperimenti col kernel

I nomi dei file possono contenere “spazi”. Ciò rende scorretti comandi del tipo

```
for i in $(find percorso predicato); do ... done
```

Inoltre la “riga di comando” è limitata in lunghezza, quindi quando i risultati sono troppi non funziona

Soluzione

```
1 find / -type f | while read x; echo $x; done
```

271

Esercizi



Sistemi Operativi

Bruschi Martignoni Monga

Unix power tools
find
Archivi
Segnali

Primi esperimenti col kernel

- 1 Trovare il file più “grosso” in un certo ramo
- 2 Copiare alcuni file (ad es. il cui nome segue un certo pattern) di un ramo in un altro mantenendo la gerarchia delle directory
- 3 Calcolare lo spazio occupato dai file di proprietà di un certo utente
- 4 Scrivere un comando che conta quanti file ci sono in un determinato ramo del filesystem
- 5 Determinare l'occupazione totale dei file di proprietà dell'utente `ast`

272

Archivi



Sistemi Operativi

Bruschi Martignoni Monga

Unix power tools
find
Archivi
Segnali

Primi esperimenti col kernel

Un archivio *archive* è un file di file, cioè un file che contiene i byte di diversi altri file e i relativi *metadati*. (Cfr. con una *directory*, che è un file speciale, che sostanzialmente contiene solo l'elenco dei file)

- **ar** L'archivatore classico, generalmente utilizzato per le librerie (provare `ar t /usr/lib/i86/libc.a`)
- **tar** Tape archive, standard POSIX
`tar cvf archivio.tar lista_files`

Gli archivi possono essere compressi con `compress` o, più comunemente ma assenti nel setup di MINIX di base, con `gzip` o `bzip2`

I file `.zip` sono archivi compressi.

273

Montare un file system



Sistemi Operativi

Bruschi Martignoni Monga

Unix power tools
find
Archivi
Segnali

Primi esperimenti col kernel

Il file system di UNIX è un albero (in realtà un DAG, perché come si vedrà ci possono essere dei link “trasversali”) con una sola radice.

L'introduzione di un nuovo pezzo di file system (per esempio contenuto in un dischetto, una chiavetta USB, un disco di rete) si dice *montare* il file system. Occorre scegliere un punto di montaggio: il file system diventerà un sottoalbero la cui radice è tale punto (eventuali sottoalberi già esistenti non saranno più accessibili)

```
mount /dev/fd0 /mnt
```

274



Altre utility “standard” di cui è bene conoscere almeno l'esistenza

Prog. (sez. man)	Descrizione
od (1)	dump files in octal and other formats
uniq (1)	report or omit repeated lines
cut (1)	remove sections from each line of files
tr (1)	translate or delete characters
dd (1)	convert and copy a file
stat (1)	display file or file system status
test (1)	check file types and compare values
tee (1)	read from standard input and write to standard output ...
basename (1)	strip directory and suffix from filenames
dirname (1)	strip non-directory suffix from file name
sed (1)	stream editor for filtering and transforming text

Inoltre è molto utile conoscere le espressioni regolari (man 7 re_format), usate da grep, sed, ecc.



- 1 Creare un archivio tar.gz contenente tutti i file la cui dimensione è minore di 50KB
- 2 Rinominare un certo numero di file: per esempio tutti i file .png in .jpg
- 3 Creare un file da 10MB costituito da caratteri casuali (usando /dev/random) e verificare se contiene la parola MINIX
- 4 Trovare l'utente che ha il maggior numero di file nel sistema
- 5 Trovare i 3 utenti che, sommando la dimensione dei loro file, occupano piu' spazio nel sistema.



```

1 /* Regular signals. */
2 #define SIGHUP 1 /* hangup */
3 #define SIGINT 2 /* interrupt (DEL) */
4 #define SIGQUIT 3 /* quit (ASCII FS) */
5 #define SIGILL 4 /* illegal instruction */
6 #define SIGTRAP 5 /* trace trap (not reset when caught) */
7 #define SIGABRT 6 /* IOT instruction */
8 #define SIGBUS 7 /* bus error */
9 #define SIGFPE 8 /* floating point exception */
10 #define SIGKILL 9 /* kill (cannot be caught or ignored) */
11 #define SIGUSR1 10 /* user defined signal # 1 */
12 #define SIGSEGV 11 /* segmentation violation */
13 #define SIGUSR2 12 /* user defined signal # 2 */
14 #define SIGPIPE 13 /* write on a pipe with no one to read it */
15 #define SIGALRM 14 /* alarm clock */
16 #define SIGTERM 15 /* software termination signal from kill */
17 #define SIGEMT 16 /* EMT instruction */
18 #define SIGCHLD 17 /* child process terminated or stopped */
19 #define SIGWINCH 21 /* window size has changed */
    
```



```

1 /* POSIX requires the following signals to be defined, even if they are
2 * not supported. Here are the definitions, but they are not supported.
3 */
4 #define SIGCONT 18 /* continue if stopped */
5 #define SIGSTOP 19 /* stop signal */
6 #define SIGTSTP 20 /* interactive stop signal */
7 #define SIGTTIN 22 /* background process wants to read */
8 #define SIGTTOU 23 /* background process wants to write */
    
```

Mandare segnali



DICo

Sistemi Operativi

Bruschi Martignoni Monga

Unix power tools
find
Archivi Segnali

Primi esperimenti col kernel

- I segnali possono essere emessi col programma `kill`
- È possibile “intrappolare” i segnali ricevuti, ossia eseguire una routine di risposta

```
1 #!/bin/sh
2
3 trap "echo Ho ricevuto il segnale USR1" 10
4 trap "echo Ho ricevuto il segnale USR2" 12
5 trap "echo Ho ricevuto il segnale INT" 2
6
7 while true ; do
8     sleep 1
9     echo "ciao: sono $(id)"
10 done
```

- SIGKILL non può essere catturata.

279

Link



DICo

Sistemi Operativi

Bruschi Martignoni Monga

Unix power tools
find
Archivi Segnali

Primi esperimenti col kernel

- <http://www.gnu.org/software/fileutils/fileutils.html>

280

Iniziamo a dare un'occhiata al codice



DICo

Sistemi Operativi

Bruschi Martignoni Monga

Unix power tools
find
Archivi Segnali

Primi esperimenti col kernel

Per ricompilare il sistema

```
1 cd /usr/src/tools
2 make clean
3 make image
4 make hdboot
```

Ci mette 3-5 minuti. Esercizio: Cambiare Il nome del sistema che appare nella prima riga col copyright

281

Debug dumps



DICo

Sistemi Operativi

Bruschi Martignoni Monga

Unix power tools
find
Archivi Segnali

Primi esperimenti col kernel

Premendo i tasti F1-F7, F10-F12, Shift+F1-F4 vengono stampate informazioni sulle strutture dati del kernel

- Identificare i valori del parametro di boot `memory`

282



- Accedere al boot monitor
- Provare il comando `help`
- Cosa succede cambiando il parametro di boot `memory=800...` in `memory=900...`?
- Cambiare qualche campo di `kinfo` (p.es. `release`) modificandone l'inizializzazione in `cstart`