Sistemi Operativi

Laboratorio – linea 2

8

Matteo Re

Dip. di Informatica
Università degli studi di Milano

matteo.re@unimi.it

a.a. 2016/2017 – Sem. II

http://homes.di.unimi.it/re/solabL2.html

Sistemi Operativi

Laboratorio – linea 2

8

Lezione 8:

Unix power tools

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

Spesso usato insieme a 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.

Due espressioni idiomatiche

Spesso si vuole fare un'operazione per ogni file trovato con find. L'espressione piú naturale sarebbe:

```
1 for i in $(find percorso predicato); do
2    comando $i
3 done
```

Questa forma presenta due problemi: può eccedere la misura della linea di comando e non funziona correttamente se i nomi dei file contengono *spazi*

Due espressioni idiomatiche

Un'alternativa è

```
1 find percorso predicato -print0 | xargs -0 -n 1
```

In questo modo (-print0) i file trovati sono separati dal carattere 0 anziché spazi e xargs è capace di adattarsi a questa forma.
Un'alternativa piú generale che mostra la potenza del linguaggio di shell che non distingue fra comandi e costrutti di controllo di flusso (sono tutti "comandi" utilizzabili in una pipeline)

```
1 find percorso predicato | while read x; do
2   comando $x
3 done
```

read x legge una stringa e la assegna alla variabile *x*.

ESERCIZI

1. Trovare il file piú "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

ESERCIZIO 1

E1: Trovare **il file** “più grosso” in un **certo ramo**

Innanzitutto otteniamo una root shell: sudo -s

Toviamo tutti I file in (ad es.) /var	find **/var** -type f **-print0**
Mediante xargs passiamoli come argomenti a du	xargs **-0** du
Ordiniamo la lista ottenuta (ord. crescente)	sort -n
Estraiamo l'ultimo file (Il file più grande)	tail **-1**
In ogni riga 2 campi: size rel.path. . Estrarre il secondo	cut -f2
Usiamo in modo combinato xargs e du per output finale	xargs -I{} du -sh {}

Pipeline soluzione:

```
find /var -type f -print0 | xargs -0 du | sort -n | tail -1 | cut -f2 | xargs -I{} du -sh {}
```

ESERCIZIO 2

E2: Copiare alcuni file (ad es. il cui nome segue un certo **pattern**) di un **ramo** in **un altro** mantenendo la **gerarchia** delle directory

Supponiamo che, nel folder corrente, esista un folder di nome **localinclude**

Toviamo tutti I file s*.h in /usr/include
Mediante xargs passiamoli come argomenti a cp

find /usr/include/ -name '**s*.h**'
xargs -I{} cp **--parents** {} **./localinclude**

per preservare permessi,timestamp ecc.
aggiungere -p (vedere man cp)

NB: preserva struttura directory, provare **s**enza.

Pipeline soluzione:

```
find /usr/include/ -name 's*.h' | xargs -I{} cp --parents {} ./localinclude
```

Osservazioni? Si può migliorare?

ESERCIZIO 3

E3: Calcolare lo spazio occupato dai **file** di proprietà di un certo utente

Innanzitutto otteniamo una root shell: sudo -s

Toviamo tutti I file di un utente in un dato ramo e utilizziamo il parametro exec di find per eseguire du

find /home **-user** user -exec du -k {} \;

Utilizziamo awk per calcolare la somma dello spazio occupato

awk '{ s = s+$1 } **END** {print " Total used: ",s}'

varie opzioni: b,k,m

s: accumulatore
valore prima colonna

Pipeline soluzione:

find /home -user user -type f -exec du -k {} \; | awk '{ s = s+$1 } END {print " Total used: ",s}'

NB: possiamo agire secondo la stessa logica ma selezionando per nome del gruppo:
-group groupname

ESERCIZIO 4

E4: Scrivete un comando che conta quanti file ci sono in un determinato ramo del filesystem

Innanzitutto otteniamo una root shell: sudo -s

Toviamo tutti I file in un dato ramo	find /home -type f
Passiamo tutto a wc con l'opzione conteggio righe	wc -l

Pipeline soluzione:

find /home -type f | wc -l

ARCHIVI

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'archiviatore classico, generalmente utilizzato per le librerie (provare `ar t /usr/lib/i86/libc.a`

NB: provate ad identificare la posizione del file libc.a nel sistema e solo dopo aver determinato la posizione del file utilizzate ar . Per vedere tutti i possibili parametri di ar consultate il manuale (man ar)

ARCHIVI

- `tar` Tape archive, standard POSIX

  tar cvf archivio.tar lista_files

Gli archivi possono essere compressi con `compress` o, piú comunemente, con `gzip` o `bzip2`
I file `.zip` sono archivi compressi.

tar **non comprime i file** … semplicemente li archivia. Poi si appoggia a programmi esterni per comprimere. E' possibile decidere se comprimere e, in questo caso, con quale programma esterno utilizzando in modo appropriato i parametri di tar.

TAR : archiviazione

tar è utilizzato comunemente per archiviare interi rami del filesystem. Seguono alcuni esempi:

Creazione di un archivio (**c**reate) **:**

```
tar -cvf <archive_name>.tar <files>
```

Esaminare il contenuto di un tarball (file *.tar). Stampa lista del contenuto :

```
tar -tvf <archive_name>.tar
```

Estrarre I file dall'archivio (e**x**tract) :

```
tar -xvf <archive_name>.tar
```

NB: Se non è necessario selezionare i file è possibile procedere così:
`tar -cvf myhome.tar /home/user/`

Uso congiunto find e tar (mediante find … -exec …). Il parametro **r** di tar aggiunge (uno alla volta) I file trovati da find all'archivio senza ricrearlo.


```
find /usr/include/ -name 's*.h' -exec tar -rvf file.tar {} \;
```

TAR : archiviazione + compressione

Creazione di un archivio (**c**reate) **e** **compressione** **con** **gzip** **(z) :**

```
tar -cvzf <archive_name>.tar.gz <files>
```

Estrarre I file dall'archivio (e**x**tract) e decompressione gunzip (z) :

```
tar -xvzf <archive_name>.tar.gz
```

(per testare usare t al posto di x)

Se, al posto di gzip/gunzip vogliamo utilizzare come programmi di compressione/decompressione **bzip2 / bunzip2** al posto di **z** dovremo utilizzare **j** . In questo caso l'estensione del file compresso è *.tar.**bz2**

Altre utility

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

Prog. (sez. man)	Descrizione
`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
`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
`seq (1)`	print a sequence of numbers

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

ESERCIZI

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 JOS
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 piú spazio nel sistema.

ESERCIZIO 1

E1: Creare un archivio tar.gz contenente tutti i file la cui dimensione è minore di 50 KB

Otteniamo una root shell : sudo -s

Toviamo tutti I file di dimensione minore di 50 KB	find **/** -size -50k
Passiamo il risultato a tar utilizzando I parametri c (create), z (gzip compression), f (specifica nome file)	tar -cz -f
escludiamo alcune directory	--exclude “pattern”
Usiamo opzione -T per ottenere I nomi dei file da aggiungere all'archivio e come valore usiamo - (SDTIN)	-T -

Pipeline soluzione:

find / -type f **-size -50k** | tar --exclude "/dev/*" --exclude "/sys/*" --exclude "/proc/*" -cz -f test.tgz -T -

tar -ztvf test.tgz (lista il contenuto del file … quanti file ci sono?)

ESERCIZIO 2

E2: Rinominare un certo numero di file: per esempio tutti i file .png in .jpg

Create nella directory corrente alcuni file con estensione png (es. touch test1.png)

1. Toviamo tutti I file il cui nome corrisponde al pattern **.png**	find . -name '*.png'
2. Passiamo l'output di find a stream editor (sed)	sed
3. Chiediamo di eseguire questa sostituzione:	
- Per ogni corrispondenza trovata (intera stringa) scrivi mv pattern pattern	-e 's/**.***/mv & &/'
4. Chiediamo di eseguire questa sostituzione:	
- Cambia ogni occorenza di *.png in *.jpg otteniamo vecchiastringa**png** vecchiastringa**jpg**	-e 's/png**$**/jpg/'
5. Passiamo il risultato a bash (che eseguirà il comando mv)	bash

Pipeline soluzione:

find . -name '*.png' | sed -e 's/**.***/mv & &/' -e 's/png$/jpg/' | bash

nb. qui c'era un errore … ***.png**

ESERCIZIO 3

E3: Creare un file da 10MB costituito da caratteri casuali (usando /dev/random) e verificare se contiene la parola UNIX

Niente pipeline qui …

NOTA: /dev/random è migliore come generatore di caratteri casuali. In particolare assume che la “casualità” dei caratteri prodotti decresca molto velocemente e quindi si appoggia a diverse fonti di entropia quali mouse tastiera ecc. Accade spesso che /dev/random si blocchi in attesa di informazioni provenienti da mouse, tastiera e altro. Per velocizzare utilizziamo /dev/**u**random

dd if=**dev/urandom** of=10mb.file bs=1k count=10240

grep --binary-files=text -o '.\{0,5\}JOS.\{0,5\}' 10mb.file

tratta file binari come se fossero testo

stampa solo match

estensione match : al massimo 5 caratteri a monte e a valle

ESERCIZIO 5 (1)

E5: Trovare i 3 utenti che, sommando la dimensione dei loro file, occupano più spazio nel sistema.

Da risolvere mediante pipeline (tutto su una sola riga)

```
for target in $(awk -F':' '{ print $1}' /etc/passwd); do res=$(find /
-user $target -type f -printf '%p|%s\n' 2>/dev/null | sort -t \| +1 -2 |
awk -F\| '{ s += $2 } END {if(s!=0){print "Sum: ",s}else{print "Sum:
0"}}'); [[ $res != "" ]] && echo "$target $res"; done | sort -k3 -n -r |
head -n 3
```

Step 1: Ottenere una lista degli utenti nel sistema
Step 2: Per ogni utente trovare (find) I suoi file e stampare username e dimensione totale
Step 2: Ordinare in modo decrescente e prelevare le prime tre righe

NB: Step 2 va ripetuto **per ogni utente** nel sistema (for loop)

ESERCIZIO 5 (1)

E5: Trovare i 3 utenti che, sommando la dimensione dei loro file, occupano più spazio nel sistema.

Da risolvere mediante pipeline (tutto su una sola riga)

```
for target in $(awk -F':' '{ print $1}' /etc/passwd); do res=$(find /
-user $target -type f -printf '%p|%s\n' 2>/dev/null | awk -F\| '{ s +=
$2 } END {if(s!=0){print "Sum: ",s}else{print "Sum: 0"}}'); [[ $res != ""
]] && echo "$target $res"; done | sort -k3 -n -r | head -n 3
```

Step 1: Ottenere una lista degli utenti nel sistema

comando: awk **-F':'** '{print $1} /etc/passwd

Il file /etc/passwd contiene tutte le informazioni (e le relative passwork anche se non in forma leggibile) . I campi di questo file sono separati da : . Utilizziamo awk per leggere il file, spezzare ogni riga in corrispondenza del separatore : e stampare il valore presente nel primo campo.

ESERCIZIO 5 (2)

E5: Trovare i 3 utenti che, sommando la dimensione dei loro file, occupano più spazio nel sistema.

Da risolvere mediante pipeline (tutto su una sola riga). Eseguire come root.

```
for target in $(awk -F':' '{ print $1}' /etc/passwd); do res=$(find / -user
$target -type f -printf '%p|%s\n' 2>/dev/null |  awk -F\| '{ s += $2 } END
{if(s!=0){print "Sum: ",s}else{print "Sum: 0"}}'); [[ $res != "" ]] && echo
"$target $res"; done | sort -k3 -n -r | head -n 3
```

Step 2: Ottenere la somma dello spazio occupato dai file dell'utente (1 per volta) e stampare
nomeutente Sum: valore
comando: più di un comando (find + awk) ...

La riga viene salvata in una variabile res utilizzando una subshell che esegue diversi comandi. Il primo è il seguente :

path size

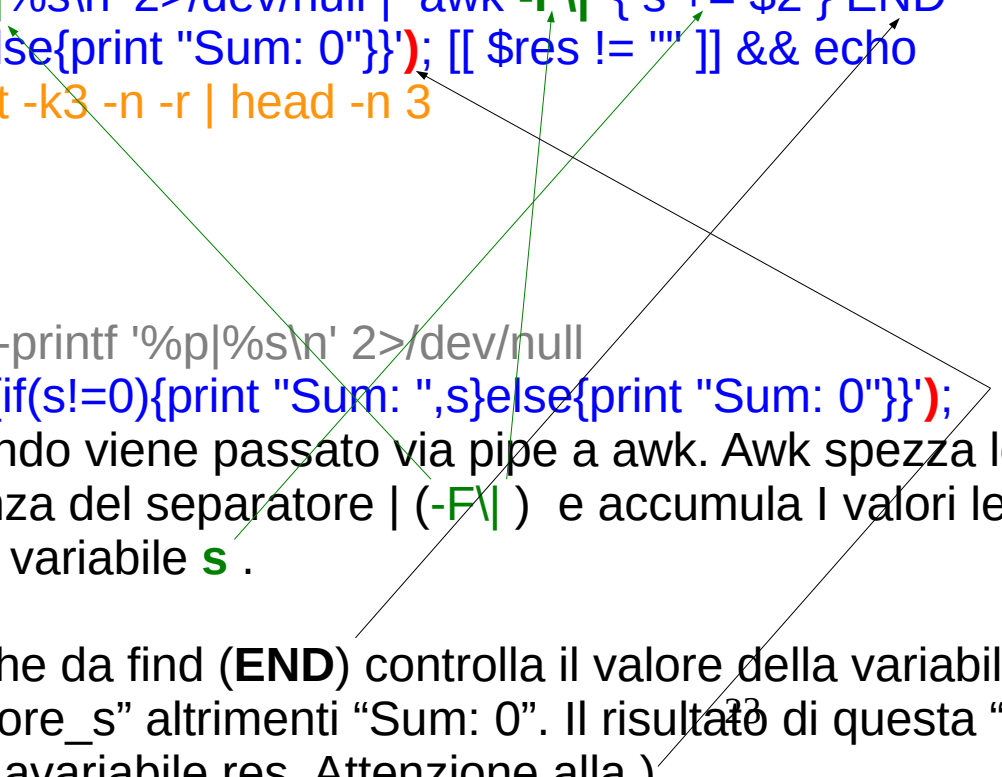
```
find / -user $target -type f -printf '%p|%s\n' 2>/dev/null
```

ESERCIZIO 5 (3)

E5: Trovare i 3 utenti che, sommando la dimensione dei loro file, occupano più spazio nel sistema.

Da risolvere mediante pipeline (tutto su una sola riga)

```
for target in $(awk -F':' '{ print $1}' /etc/passwd); do res=$(find / -user
$target -type f -printf '%p|%s\n' 2>/dev/null |  awk -F\| '{ s += $2 } END
{if(s!=0){print "Sum: ",s}else{print "Sum: 0"}}'); [[ $res != "" ]] && echo
"$target $res"; done | sort -k3 -n -r | head -n 3
```



Step 2: find / -user $target -type f -printf '%p|%s\n' 2>/dev/null
awk **-F\|** '{ s += $2 } END {if(s!=0){print "Sum: ",s}else{print "Sum: 0"}}');
Il risultato del primo comando viene passato via pipe a awk. Awk spezza le righe che riceve in corrispondenza del separatore | (-F\|) e accumula I valori letti nel secondo campo ($2) nella variabile **s** .

Quando non riceve più righe da find (**END**) controlla il valore della variabile s .
Se s!=0 stampa “Sum: valore_s” altrimenti “Sum: 0”. Il risultato di questa “stampa” Viene immagazzinato nell avariabile res. Attenzione alla) .

ESERCIZIO 5 (4)

E5: Trovare i 3 utenti che, sommando la dimensione dei loro file, occupano più spazio nel sistema.

Da risolvere mediante pipeline (tutto su una sola riga)

```
for target in $(awk -F':' '{ print $1}' /etc/passwd); do res=$(find / -user
$target -type f -printf '%p|%s\n' 2>/dev/null |  awk -F\| '{ s += $2 } END
{if(s!=0){print "Sum: ",s}else{print "Sum: 0"}}'); [[ $res != "" ]] && echo
"$target $res"; done | sort -k3 -n -r | head -n 3
```

Step 2:

```
find / -user $target -type f -printf '%p|%s\n' 2>/dev/null
awk -F\| '{ s += $2 } END {if(s!=0){print "Sum: ",s}else{print "Sum: 0"}}');
[[ $res != "" ]] && echo "$target $res";
```

Se tutto è andato a buon fine (se res non è vuota “”) stampa:
nome_utente Sum: valore

Step1 e Step2 vengono eseguiti all'interno di un ciclo for (notate for … do … done)
Step2 viene eseguito tante volte quanti sono I risultati di Step1 (gli utenti nel sistema)

ESERCIZIO 5 (5)

E5: Trovare i 3 utenti che, sommando la dimensione dei loro file, occupano più spazio nel sistema.

Da risolvere mediante pipeline (tutto su una sola riga)

```
for target in $(awk -F':' '{ print $1}' /etc/passwd); do res=$(find / -user
$target -type f -printf '%p|%s\n' 2>/dev/null |  awk -F\| '{ s += $2 } END
{if(s!=0){print "Sum: ",s}else{print "Sum: 0"}}'); [[ $res != "" ]] && echo
"$target $res"; done | sort -k3 -n -r | head -n 3
```

Step 3: Resta da eseguite l'identificazione dei **3 utenti** I cui file occupano più spazio
Passiamo il risultato ottenuto dal ciclo in cui sono coinvolti step1 e step 2 al comando sort e ordiniamo numericamente (-n) sulla base del contenuto della terza colonna (-k3) in modo decrescente (-r).

Infine, sempre via pipe, passiamo tutto a head e preleviamo le prime 3 righe (-n 3)

ESERCIZIO 4 (1)

E4: Trovare l'utente che ha il maggior numero di file nel sistema.

Da risolvere mediante pipeline (tutto su una sola riga). Eseguire come root.

```
for target in $(awk -F':' '{ print $1}' /etc/passwd); do res=$(find / -user
$target -type f -printf '%p|%s\n' 2>/dev/null |  awk -F\| '{ s += $2 } END
{print "N.: ",NR}'); [[ $res != "" ]] && echo "$target $res"; done | sort -k3
-n -r | head -n 1
```

Tutto come in esercizio 5 ma stavolta invece di stampare il valore della somma delle dimensioni dei file (s) stampiamo il numero (NR) di righe passate ad awk, che corrisponde al numero dei file trovati da find per l'utente target.

Inoltre non stampiamo I primi tre valori dopo aver chiamato sort ma il primo.

DISCHI VIRTUALI

Fuori dalla macchina virtuale qemu−img create disco.img 100M
Poi può essere usato aggiungendo −hdb disco.img
In generale (anche al di là delle macchine virtuali) un file può facilmente essere usato come "disco": i *loop device* servono proprio per utilizzare un file (che è uno stream di caratteri) come *device a blocchi*

```
1 dd if=/dev/zero of=prova.img seek=10M bs=1 count=0
2 /sbin/mkfs prova.img
3 sudo mount −o loop prova.img /mnt
4 echo ciao > /mnt/pippo
5 sudo umount prova.img
```

options

-o **loop**