Sesta Esercitazione

Sommario

- Archiviazione e compressione
  - archiviazione con il comando tar
    - ⇨ operazioni principali (creazione, visulizzazione, estrazione archivi)
    - ⇨ opzioni principali (abilitazione percorsi assoluti, modifica della cartella di destinazione, compressione)
  - compressione file singoli
    - ⇨ comandi gzip, gunzip, bzip2, bunzip2
- Ricerca dei file
  - in percorsi specificati: comando find
    - ⇨ sintassi: percorsi, espressioni (condizioni e operatori)
    - ⇨ esecuzione di un comando sul risultato della ricerca (azione -exec)
  - in un database: comando locate
    - ⇨ invocazione del comando
    - ⇨ creazione del database e opzioni
- Pianificazione dei processi
  - demone cron
    - ⇨ file di configurazione: sintassi
    - ⇨ comando crontab e opzioni

Archiviazione e compressione

Archiviazione con il comando `tar`

- `tar` = **Tape Archive**
- **Un file `tar` è una raccolta di file e/o directory in un unico file.**
- `tar` **genera un archivio non compresso**
- `tar` **viene utilizzato insieme al compressore di default Gnu Zip (`gzip`)**
- **il formato `tar` compresso (file `.tgz` o `.tar.gz`) è diventato ormai lo standard per il passaggio di dati tra sistemi Unix.**

```
tar [azione][switch] [archivio] [file]
```

Opzioni

- azione
  - `c` : crea
  - `x` : estrae
  - `t` : visualizza il contenuto di un archivio
- switch
  - `v` : verbose (per ottenere maggiori informazioni nel corso delle operazioni (debug))
  - `z` : comprime il file `tar` con `gzip`
  - `j` : comprime il file `tar` con `bzip2`
  - `f` : file
    - ⇨ Deve essere messa come ultima opzione
    - ⇨ Usa il path relativo
  - `n` : forza un comportamento non ricorsivo

Creazione

```
tar –cvf archivio.tgz sorgente
tar czvf archivio.tar.gz ~/archivio/*
```

- `archivio.tgz` : **file che state creando**
- `sorgente` : **file/directory che volete inserire nel nuovo file archiviato.**

Visualizzazione

```
tar -tvf filename.tar
```

- **Elenca il contenuto di** `filename.tar`

Estrazione

```
tar –xvf archivio.tgz destinazione
tar xzvf archivio.tar.gz
```

- **Questo comando non rimuove il file `tar`, ma crea copie del suo contenuto**

gzip and gunzip

- Programma di compressione attraverso il quale viene creato un file compresso per ogni file indicato negli argomenti
- Programma indipendente da `tar`
- `gzip`
  - Utility di compressione
- `gunzip`
  - Utility di decomprime

```
gzip archivio.a archizio.gz
```

bzip2 and bunzip2

- **Effettua una compressione maggiore**
- bzip2
  - **Utility di compressione**
- bunzip2
  - **Utility di decompressione**

```
bzip2 archivio.a archizio.bz2
bzip2 file.bz2 file2 file3 /usr/work/school
```

Path

- Nell'archivio i file vengono memorizzati col path
- Non vengono fatti controlli di overwrite
- Se non viene specificata la destinazione si usa la directory corrente ed il path memorizzato nell'archivio
- `-C nome_directory`: estrae i dati nella directory specificata

Ricerca dei file

find

- Esegue una ricerca all'interno di uno o più percorsi per i file che soddisfano delle condizioni determinate
- Le condizioni sono legate all'apparenza esterna e non al contenuto

find

- **Sintassi**

`find [percorso…] [espressione]`

  - `percorso…`
    - ⇨ **insieme di percorsi separati da spazi**
    - ⇨ **Se non specificato è la directory corrente**
  - `espressione`: **insieme di test e azioni separati da operatori**

Espressioni

- `[opzione…] [condizioni]`
- **Opzione**
  - **Modo di configurare il comportamento del programma**
- **Condizioni**
  - **Espressioni che generano un risultato logico**
  - **restituiscono un valore vero/falso**
  - **Per concatenare insieme più condizioni si usano gli operatori booleani**
  - **di default è utilizzato l'AND logico** (`-and`)

Condizioni

- `-name schema`
  - restituisce TRUE se il nome del file corrisponde allo schema specificato
  - si possono utilizzare i caratteri speciali `*` e `?`
- `-size [+|-]n[b|c]`
  - restituisce TRUE se la dimensione del file è uguale (maggiore con + o minore con -) a `n` unità di spazio
  - `b` per unità da 512 bytes (default)
  - `c` per byte
- `-type t`
  - restituisce TRUE se il file è del tipo specificato
  - `d` per cartelle
  - `f` per file normali
  - `l` per link simbolici

Condizioni

- `-user utente`
  - restituisce TRUE se il file/directory appartiene all'utente specificato
- `-group gruppo`
  - restituisce TRUE se il file/directory appartiene al gruppo specificato
- `-perm [-/+]modello`
  - restituisce TRUE se i permessi del file corrispondono esattamente al modello specificato (in forma ottale o simbolica)
  - `-` ignora permessi extra
  - `+` per uno qualsiasi dei modi specificati

Operatori booleani di `find`

- `(espressione)`
  - **Stabiliscono la precedenza nell'esecuzione dei test**
- `!espressione`
  - **nega un'espressione**
- `espressione [-and] espressione`
  - **AND logico tra espressioni (può essere omesso)**
- `espressione -or espressione`
  - **OR logico tra espressioni**

- **Ricordarsi di usare i caratteri di escape per proteggerli dall'espansione della shell**

Azioni di `find`

- **Sono operazioni da compiere per ogni file o directory che si ottiene dalla scansione.**
- `-print`
  - **stampa i nomi dei file trovati separandoli con un terminatore di stringa (carattere `NULL`)**
- `-exec comando [{}] [;|+]`
  - **esegue il comando specificato sui risultati della ricerca**
  - **di default esegue una istanza del `comando` per ogni file trovato (la riga deve terminare con `;`)**
  - **se si usa `{}` i file trovati vengono passati al `comando` come lista (la riga deve terminare con `+`)**

Esempi

```
find . –name prova\* -print
```

- **Cerca tutti i nomi che iniziano con** `prova`

```
find / -name “lib*” -print
```

- **Esegue una ricerca sul file system globale per i file/directory il cui nome inizia per** `lib`
- `\` **protegge i metacaratteri**

Esempi

```
find /home –name “pro*” ! –type d
```

- **Esegue una ricerca a partire da `/home` per i file il cui nome inizia con `pro`**
- **nella ricerca vengono escluse le directory**

- **Le virgolette vengono usate per evitare che la shell trasformi `lib*/pro*` in qualcosa di diverso**

Ricerca nei file

grep

- Cerca in uno o più file le righe contenenti la stringa specificata.

```
grep [opzioni] "stringa" nome_file ...
```

- Esempio:

```
grep “pippo” pluto
```

- Il risultato visualizza le parti del file `pluto` in cui compare la parola `pippo`.
- Se vengono indicati due o più file in cui cercare, nell'output è incluso il nome del file.
- Esempio:

```
grep pippo *
```

- Cerca la parola `pippo` in tutti i file della directory corrente.
- Il comando `grep` e' case sensitive

grep

- È possibile utilizzare espressioni regolari per le ricerche (generalized regular expression printer)
- Esempio

```
grep 'ri.*o' pluto
```

- Cerca tutte le stringhe che iniziano per `ri` e terminano con `o` all'interno del file `pluto`.
- Le virgolette singole impediranno alla shell di trattare l'asterisco come carattere jolly
- `.*` indica 0 o più caratteri qualunque

Opzioni

- `-i` (ignore case)
  - consente di ignorare le distinzioni tra minuscole e maiuscole
- `-v`
  - mostra le linee che NON contengono l'espressione specificata
- `-n`
  - premette il numero di riga davanti ad ogni riga che riporta
- `-c`
  - riporta solo il conteggio delle linee che contengono la sequenza
- `-w`
  - verificare solo parole intere
- `-x`
  - controlla le corrispondenze di linee intere

Metacaratteri (1 di 2)

- Talvolta chiamati anche caratteri jolly o wild-cards
- Hanno un significato particolare
- ^
  - Inizio riga

```
grep '^d' ls.out
```

    - ⇨ Tutte le righe che iniziano per `d`
- $
  - Fine riga

```
grep '\.c$' ls.out
```

    - ⇨ Cerca le righe che finiscono per `.c`
    - ⇨ È stato necessario impiegare anche il metacarattere '\' perché anche il carattere '.' è un metacarattere ma nella ricerca in corso si desiderava inserirlo letteralmente e non nel suo significato di metacarattere.
    - ⇨ '\' neutralizza il significato di metacarattere del carattere che segue
    - ⇨ Per citare letteralmente il carattere '\' è necessario quindi scriverlo due volte: '\\'

Metacaratteri (2 di 2)

- .
  - rappresenta nelle espressioni regolari uno ed un solo carattere qualunque
- *
  - zero o più occorrenze dell'espressione che lo precede

- Le espressioni regolari possono contenere anche più metacaratteri.
- Per ricercare un'ipotetica riga costituita dalla sola stringa "`riga completa`" si usa:

```
'^riga completa$'
```

- Per individuare tutte le righe vuote del file si usa:

```
'^$'
```

Insiemi di caratteri

- `[s]`
  - '[' e ']' sono metacaratteri
  - 's' rappresenta un elenco di caratteri ammessi.
  - L'insieme '[s]' soddisfa UN SOLO qualunque carattere che sia compreso nell'elenco s.
  - Negli insiemi '[s]' si possono specificare intervalli di caratteri usando il carattere '-'

Esempio

```
grep '1[23]:[0-5][0-9]' ls.out
```

- L'espressione regolare specifica che:
  - il primo carattere della stringa cercata deve essere letteralmente un `'1'`
  - il secondo può essere un `'2'` o un `'3'`
  - il terzo deve essere letteralmente un `':'`
  - il quarto può essere `'0'` o `'1'` o `'2'` o `'3'` o `'4'` o `'5'`
  - il quinto deve essere una cifra compresa tra `'0'` e `'9'` (estremi compresi).

Pianificazione dei processi

Pianificazione dei processi

- Esecuzione di processi in date e orari stabiliti
- Il demone `cron` controlla queste esecuzioni
- `crontab`: file contenente le configurazioni
- Solitamente si ha:
  - Un file per ogni utente
  - Uno generale per tutto il sistema

cron

- **Demone funzionante in background**
- **Interpreta i file `crontab` collocati in**
  - `/etc/crontab`
    - ⇨ **per le impostazioni globali**
  - `/var/cron/tabs/`*`nome_utente`*
    - ⇨ **per ogni utente si ha un file con il proprio nome**

crontab

`crontab [opzioni]`

- **permette di creare o modificare il file `crontab` di un utente**
- **Solo root può agire sul file `crontab` di un altro utente**
- **I file `crontab` vengono usati dal demone `cron` che si occupa di eseguire i comandi indicati**

Opzioni

`[-u utente] file`

- **Sostituisce il file `crontab` con il contenuto del file indicato come argomento**

`-l`

- **Visualizza il file `crontab` dell'utente**

`-e`

- **Crea o modifica il file `crontab` dell'utente**

`-r`

- **Cancella il file `crontab` dell'utente**

Variabili di ambiente

- `SHELL`
  - **Stabilisce con quale shell devono essere eseguiti i comandi (`/bin/sh`)**
- `LOGNAME`
  - **Nome dell'utente**
- `HOME`
  - **Directory personale dell'utente**
- `MAILTO`
  - **Destinatario dei messaggi di posta che vengono generati**
  - **"" non viene inviato nessun messaggio**

Formato del file `crontab`

- **campi separati da spaziature**
- **campi relativi all'istante di esecuzione**

`minuto, ora, giorno, mese, giorno della settimana`

  - **0 domenica, 1 lunedì, ..., 7 domenica**

- `utente`
  - **solo per le impostazioni globali (file `/etc/crontab`), per gli altri è implicito (l'utente stesso)**
- `comando`
  - **senza redirezione l'output viene inviato per e-mail all'amministratore**

Caratteri speciali

- *
  - qualsiasi valore
- -
  - per delimitare un insieme di valori compresi tra gli estremi inseriti (es. 1-3)
- ,
  - per separare singoli valori (es. 2,5)
- /
  - per esprimere una granularità (es. /8)

Esempio /var/cron/tabs/*

```
# Utilizza «/bin/sh» per eseguire i comandi, indipendentemente da
# quanto specificato all'interno di «/etc/passwd».
SHELL=/bin/sh
# Invia i messaggi di posta elettronica all'utente «fede»,
# indipendentemente dal proprietario di questo file crontab.
MAILTO=fede
# Esegue 5 minuti dopo la mezzanotte di ogni giorno.
5 0 * * * $HOME/bin/salvataggiodati
# Esegue alle ore 14:15 del primo giorno di ogni mese.
# L'output viene inviato tramite posta elettronica all'utente
«tizio».
15 14 1 * * $HOME/bin/mensile
# Esegue alle 22 di ogni giorno lavorativo (da lunedì al venerdì).
# In particolare viene inviato un messaggio di posta elettronica a
«fede».
0 22 * * 1-5 mail -s "Sono le 22" fede%Fede,%%è ora di smettere!%
# Esegue 23 minuti dopo mezzanotte, dopo le due, dopo le quattro,...,
# ogni giorno.
23 0-23/2 * * * echo "Ciao ciao"
# Esegue alle ore 04:05 di ogni domenica.
5 4 * * 0 echo "Buona domenica"
```

Esempio /etc/crontab

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# Run any at jobs every minute
* * * * * root [ -x /usr/sbin/atrun ] && /usr/sbin/atrun
# run-parts è un programma che avvia tutti gli eseguibili
   contenuti nella directory indicata come argomento
01 * * * * root run-parts /etc/cron.hourly
02 1 * * * root run-parts /etc/cron.daily
02 2 * * 0 root run-parts /etc/cron.weekly
02 3 1 * * root run-parts /etc/cron.monthly
# Remove /tmp, /var/tmp files not accessed in 10 days (240
   hours)
41 02 * * * root /usr/sbin/tmpwatch 240 /tmp /var/tmp
# Remove formatted man pages not accessed in 10 days
39 02 * * * root /usr/sbin/tmpwatch 240 /var/catman/cat?
```

Note

- `run-parts /etc/periodic/hourly`
  - **serve ad avviare tutto quello che c'è nella directory**
- **Per inserire un'elaborazione nei momenti più comuni, basta mettere il programma o lo script relativo nella directory che rappresenta la cadenza desiderata.**

Esercizi

- **Archiviazione e compressione**
  - **creare nella** `home` **un archivio compresso in formato** `gzip` **di nome** `config.tgz` **contenente i file con estensione** `conf` **presenti nella cartella** `/etc`
  - **mostrare i file contenuti nell'archivio**
  - **decomprimere l'archivio con** `gunzip`
  - **estrarre i file contenuti con il comando** `tar`
- **Ricerca dei file**
  - **cercare dentro la cartella** `/etc` **tutti i file il cui nome contiene la stringa** `sys` **e la cui dimensione è superiore a** `10 byte`
  - **cercare nella** `root` **tutti i file che hanno il bit** `SUID` **o** `SGID` **attivo**
  - **concatenare e mostrare a video tutti i file (a partire dalla** `root`**) il cui nome contiene la stringa** `tab`

Esercizi

- **Pianificazione dei processi (utente)**
  - **impostare il proprio file crontab in modo che ogni minuto venga eseguito il comando `date` e il relativo output sia scritto nel file `date1min` nella propria `home`**
  - **impostare il proprio file `crontab` in modo che ogni 2 minuti venga eseguito il comando `date` e il relativo output sia scritto in modalità `append` nel file `date2min` nella propria `home`**
- **Pianificazione dei processi (root)**
  - **impostare il file `crontab globale` in modo che ogni giorno a mezzanotte il contenuto della cartella `/etc/apache2` venga archiviato in `/var/` con nome `web-daily.tar.bz2`**

Soluzione

- **Da utente (a partire dalla propria home)**
  - `tar cvzf config.tgz /etc/*conf`
  - `tar ztvf config.tgz`
  - `gunzip config.tgz`
  - `tar xvf config.tar`
- **Da qualsiasi cartella**
  - `find /etc/ -name \*sys\* -size +10c`
  - `find / -perm -u=s -or -perm -g=s`
  - `find / -name \*tab\* -exec cat {} +`

Soluzione

- **Da utente**
  - `crontab -e`
  - **inserire la riga seguente**

```
*/1 * * * * date > ~/date1min
```

  - `crontab -e`
  - **inserire la riga seguente**

```
*/2 * * * * date >> ~/date2min
```

- **Da root**
  - **aggiungere al file** `/etc/crontab` **la riga seguente**

```
0 0 * * * root tar cjf /var/web-
  daily.tar.bz2 /etc/apache2 >/dev/null
  2>&1
```