Modulo 2 - Appendice 2
Breve rassegna di comandi (esterni) in ambiente Linux

Laboratorio di Sistemi Operativi I
Anno Accademico 2008-2009

Comandi: `cat` - `tac` - `rev`

- **Comando `cat` (concatenate):**
  - Stampa i file su stdout. Utilizzato con redirection, serve a concatenare file:

```
cat file.1 file.2 file.3 > file.123
```

- **Comando `tac` (inverso di cat):**
  - Stampa le linee dei file in ordine inverso, partendo dall'ultima linea
- **Comando `rev` (reverse):**
  - Stampa (invertite) le linee dei file, partendo dalla prima linea

Comando: `find` (I)

- **Comando: `find` *`pathlist expression`***
  - L'utility **`find`** analizza ricorsivamente i path contenuti in ***`pathlist`*** e applica ***`expression`*** ad ogni file
- **Sintassi di expression:**
  - ***`-name pattern`***

    True se il nome del file fa matching con pattern, che può includere i metacaratteri di shell: * [ ] ?
  - ***`-perm permission`***

    True se permission corrisponde ai permessi del file
  - ***`-print`***

    stampa il pathname del file e ritorna true

Comando: `find` (II)

- **Sintassi di expression:**
  - **`-ls`**

    Stampa gli attributi del file e ritorna true
  - **`-user` *`username`*`, -uid` *`userId`***

    True se il possessore del file è ***`username`*** / ***`userId`***
  - **`-group` *`groupname`*`, -gid` *`groupId`***

    True se il gruppo del file è ***`groupname`*** / ***`groupId`***
  - **`-atime | -mtime | -ctime -`*`count`***

    True se il file è stato “acceduto | modificato | modificato oppure cambiati gli attributi” negli ultimi ***`count`*** giorni

Comando: `find` (III)

- **Sintassi di expression:**
  - **`-type b | c | d | p | f | l | s`**

    True se il file è di tipo a blocchi | a caratteri | una directory | un named pipe | un file regolare | un link | un socket
  - **`-exec` *`command`***

    True se l'exit status di ***`command`*** è 0.
    `command` deve essere terminato da un "escaped ;" (\;).
    Se si specifica il simbolo {} come argomento di ***`command`*** , esso viene sostituito con il pathname del file corrente
  - **`-not, !, -a, -and, -o, -or`**

    Operatori logici (not, and, or)

Comando `find`: esempi

- `find . -name "*.c" -print`
  - Stampa il nome dei file sorgente C nella directory corrente e in tutte le sottodirectory
- `find / -mtime -14 -ls`
  - Elenca tutti i file che sono stati modificati negli ultimi 14 giorni
- `find . -name "*.bak" -ls -exec rm {} \;`
  - Cancella tutti i file che hanno estensione .bak

Comando: `xargs`

- **Comando: `xargs` *`command`***
  - **`xargs`** legge una lista di argomenti dallo standard input, delimitati da blank oppure da newline
  - esegue ***`command`*** passandogli la suddetta lista di argomenti
- **Esempio: concatena tutti i file `*.c`**

```
% find . -name "*.c" -print
./a.c
./b.c
% find . -name "*.c" -print | xargs cat > prova
```

Esempi: `find` e `xargs`

- `vi $(find . -name "*.java" | xargs grep -l "Alfa")`
  - Utilizza vi per visualizzare tutti i file nella directory corrente e nelle sottodirectory che hanno estensione java e contengono la parola "Alfa".
- `find ~ \( -name "*~" -o "#*#" \) -print | \`
  `xargs --no-run-if-empty rm -vf`
  - Rimuove tutti i file di backup o temporanei di emacs dalla home directory (ricorsivamente)
- `find . -type d -not -perm ug=w | xargs chmod ug+w`
  - Aggiunge il diritto di scrittura a tutti le directory nella directory corrente e nel suo sottoalbero

Esercizi

- E' possibile eseguire la funzione del primo esempio del lucido precedente senza usare il comando **`xargs`**?
- E' possibile eseguire la funzione del secondo esempio del lucido precedente senza usare il comando **`find`**?
- E' possibile eseguire la funzione del terzo script del lucido precedente senza usare il comando **`find`?**

Spazi e caratteri speciali

- `find ~ \( -name "*~" -o "#*#" \) -print0 | \`
  `xargs --no-run-if-empty --null rm -vf`
- **Attenzione agli spazi e ai caratteri speciali:**
  - **`xargs`** separa i nomi dei file tramite spazi o new line
  - il file **`relazione 1.txt`** viene trattato come due file
  - l'opzione **`-print0`** di **`find`** stampa stringhe null-terminated
  - l'opzione **`--null`** di **`xargs`** prende stringhe null-terminated
  - questa versione gestisce correttamente qualunque tipo di carattere speciale (come ")

Esempio

```
#!/bin/bash

# Move (verbose) all files in current directory

# to directory specified on command line.

if [ -z "$1" ];  then

   echo "Usage: `basename $0` directory-to-copy-to"

   exit 65

fi

ls . | xargs -i -t mv {} $1

 # This is the exact equivalent of mv * $1

 # unless any of the filenames has "whitespace"
   characters.

exit 0
```

Comandi per la gestione del testo

- **Comando:** `head [-n] file`
  - Lista le prime *n* linee di un file (10 default)
- **Comando:** `tail [-n] file`
  - Lista le ultime *n* linee di un file (10 default)
- **Comando:** `cut`
  - Un tool per estrarre campi dai file. Nonostante esistano altri tool (più sofisticati), **`cut`** è utile per la sua semplicità.
  - Due opzioni particolarmente importanti:
    - **`-d delimiter`**: specifica il carattere di delimitazione (tab default)
    - **`-f fields`**: specifica quali campi stampare

Comandi per la gestione del testo

- **Comandi: `expand, unexpand`**
  - L'utility **`expand`** converte i tab in spazi. L'utility **`unexpand`** converte gli spazi in *tab.*
- **Comando: `uniq`**
  - Questa utility rimuove linee duplicate (consecutive) dallo standard input. Viene usato spesso nei pipe con sort.
- **Comando: `sort`**
  - Ordina lo standard input linea per linea. E' in grado di eseguire ordinamenti lessicografici sulle linee, o di gestire l'ordinamento dei vari campi.
  - Ad esempio: l'opzione **`-g`** ordina in modo numerico secondo il primo campo dell'input

Esempio

```
% du -s /home/*
10000 /home/penguin
500 /home/black
2345 /home/white
26758 /home/gray
% du -s /home/* | sort -gr
26758 /home/gray
10000 /home/penguin
2345 /home/white
500 /home/black
```

Esempio (cont.)

```
% du -s /home/* | sort -gr | head -2

26758 /home/white

10000 /home/penguin

% du -s /home/* | sort -gr | head -2 | cut -f2

/home/white

/home/penguin

% for homedir in $(du -s /home/* | sort -gr | \
  head -2 | cut -f2); do echo $(basename $homedir) ; \
  done

white

penguin
```

Comandi per la gestione del testo

- **Comando: `wc` (word count)**
  - Conta linee, parole, caratteri
  - **`-l | -w | -c`** conta solo le linee | le parole | i caratteri
  - Certi comandi includono le funzionalità di **`wc`** come opzioni:

    `... | grep foo | wc -l` è equivalente a

    `... | grep --count foo`

- **Comando: `tr`**
  - Utility per la conversione di caratteri, seguendo un insieme di regole definite dall'utente:
  - Esempio: **`tr A-Z a-z < filename`**
    - Stampa filename trasformando tutti i caratteri in minuscoli
  - Esempio: **`tr -d 0-9 < filename`**
    - Stampa filename eliminando tutte i caratteri numerici

Esempio: filenames-to-lowercase

```
#! /bin/bash
# Changes every filename in working directory
# to all lowercase.
for filename in * ; do
  fname=`basename $filename`
  n=`echo $fname | tr A-Z a-z`
    # Change name to lowercase.
  if [ "$fname" != "$n" ] ; then
    # Rename only files not lowercase.
    mv "$fname" "$n"
fi
done
exit 0
```

Esempio: dos2unix

```
!/bin/bash

# dos2unix.sh: DOS to UNIX text file converter.

E_WRONGARGS=65

if [ -z "$1" -a -z "$2"]; then

  echo "Usage: `basename $0` file-source file-dest"

  exit $E_WRONGARGS

fi

CR='\015'  # Lines in DOS text files end in a CR-
  LF.

tr -d $CR < $1 > $2 # Delete CR and write to $2

exit 0
```

Comandi per il confronto di file

- **Comando:** `cmp file1 file2`
  - Ritorna true (exit status 0) se due file sono uguali, ritorna false (exit status != 0) altrimenti
  - Stampa la prima linea con differenze
  - Con l'opzione `-s` non stampa nulla (utile per script)
- **Comando:** `diff file1 file2`
  - Ritorna true (exit status 0) se due file sono uguali, ritorna false (exit status != 0) altrimenti
  - Stampa un elenco di differenze tra i due file (linee aggiunte, linee modificate, linee cancellate)
- **Comando:** `diff dir1 dir2`
  - Confronta due directory e mostra le differenze (file presenti in una sola delle due)

Comandi per la gestione di file

- **Comando: `locate, slocate, updatedb`**
  - I comandi **`locate`** e **`slocate`** (secure version di **`locate`**) cercano file utilizzando un database apposito. Il database riflette il contenuto del file system, ma va aggiornato con **`updatedb`** (solo root)
- **Comando: `file` *`file`***
  - Identifica il tipo di file a partire dal suo *magic number* (quando disponibile). L'elenco dei magic number si trova in **`/usr/share/magic`** (o altre posizioni, consultate info file)
  - Esempio:

```
% file prova.sh

prova.sh: Bourne-Again shell script text executable
```

Comandi per la gestione dei file

- **Comando: `basename` *`file`***
  - rimuove l'informazione del path da un pathname
- **Comando: `dirname` *`file`***
  - rimuove l'informazione del nome di file da un pathname
- **Nota: sono funzioni di stringa, non agiscono su un file effettivo**
- **Esempi:**

```
% basename /home/penguin/index.html
index.html
% dirname /home/penguin/index.html
/home/penguin
echo "Usage: `basename $0` arg1 arg2 ... argn"
```

Archiviazione e compressione

- **Compressione:**
  - Comandi: **`compress`**, **`uncompress`**
  - Comandi: **`gzip`**, **`gunzip`**
  - Comandi: **`bzip2`**, **`bunzip2`**
  - Comprimono e decomprimono file. **`compress`** è ormai in disuso (originario dei primi sistemi Unix); **`bzip2`** è meno diffuso (almeno per ora), ma è in alcuni casi più efficiente di **`gzip`**
- **Archivazione: tar**
  - Archivazione: **`tar zcvf`** ***`archive-name {file}*`***
    - **`z`**=comprimi, **`c`**=crea, **`v`**=verbose, **`f`**=su file
  - Estrazione: **`tar zxvf`** ***`archive-name`***
    - **`z`**=espandi, **`x`**=estrai, **`v`**=verbose, **`f`**=da file

Comandi per la gestione del tempo

- **Comando: `touch` *`file`***
  - utility per modificare il tempo di accesso/modifica di un file, portandolo ad un tempo specificato (**`touch -d`**)
  - può essere utilizzata anche per creare un nuovo file
  - creare file vuoti può essere utile per memorizzare solo la data
- **Comando: `date`**
  - Stampa informazioni sulla data corrente, in vari formati
- **Comando: `time` *`command`***
  - esegue il comando *command* e stampa una serie di statistiche sul tempo impiegato
  - Esempio: **`time find / -name "*.bak" -print`**