Manuale Operativo Linux: Riga di Comando e Filtri di Testo

Appunti di Laboratorio - Teoria, Logica ed Esercizi Pratici

INDICE

1. La Filosofia della Command Line

La riga di comando di Linux (spesso la shell *Bash*) si basa su una filosofia modulare: fornire piccoli programmi che fanno **una sola cosa, ma la fanno benissimo**.

1.1 Sintassi dei Comandi

La struttura universale di un comando è:

comando [opzioni] [argomenti]

- **comando**: L'azione da eseguire (es. ls, head).
- **opzioni**: Modificano il comportamento del comando. Iniziano quasi sempre con un trattino - (es. -l, -n).
- **argomenti**: L'oggetto su cui il comando deve agire, tipicamente un file o una directory (es. /usr/share/dict/words).

1.2 Standard Streams e Pipeline (|)

I comandi Linux comunicano attraverso flussi di dati (Streams):

- **Standard Input (stdin)**: I dati in ingresso.
- **Standard Output (stdout)**: I risultati stampati a schermo.

La **Pipeline (il simbolo |)** è il concetto più potente della shell. Serve a collegare lo stdout del primo comando allo stdin del comando successivo.

Invece di salvare file temporanei, si crea una **catena di montaggio**: l'output grezzo del primo comando viene "incanalato" nel secondo, che lo rifinisce, e così via.

1.3 Variabili d'Ambiente

Le variabili d'ambiente sono "contenitori" di memoria che il sistema usa per configurare la

sessione dell'utente.

- Si visualizzano tutte con il comando env.
- Per richiamare il valore di una specifica variabile, si antepone il simbolo **$**.
    - *Esempio:* echo $PATH. La variabile PATH contiene l'elenco delle cartelle in cui il sistema va a cercare i programmi quando digiti un comando. Senza il $, stamperesti solo la parola "PATH".

2. Gli Strumenti: I Filtri di Testo

I filtri sono programmi progettati per leggere testo, manipolarlo secondo regole precise, e sputarlo fuori modificato.

2.1 Tagli Verticali: head e tail

Servono ad estrarre blocchi di righe dall'alto (head) o dal basso (tail). L'opzione fondamentale è -n (numero di righe), ma il suo comportamento cambia con i segni matematici:

- **head -n 20**: Prende le *prime 20* righe.
- **head -n -20**: Prende *tutto tranne* le ultime 20 righe.
- **tail -n 20**: Prende le *ultime 20* righe.
- **tail -n +20**: Parte dalla riga 20 e prende *tutto fino alla fine* del file.

> **Regola di Efficienza:** I comandi che partono dall'inizio (head -n 20, tail -n +20) sono velocissimi. Quelli che contano dal fondo (tail -n 20, head -n -20) sono inefficienti per file enormi, perché costringono il sistema a leggere prima l'intero file per trovare dove finisce.

2.2 Tagli Orizzontali (Colonne): cut vs awk

Servono ad estrarre specifiche colonne da dati tabellari.

- **cut**: Usa il parametro -d (delimiter) e -f (field).
    - *Problema:* cut è rigido. Se usi lo spazio come delimitatore (-d' '), andrà in crisi se ci sono più spazi consecutivi (come in ls -l), trattandoli come campi vuoti.
- **awk**: Molto più di un filtro, è un linguaggio di programmazione.
    - *Vantaggio:* Di default, awk considera **qualsiasi sequenza di spazi vuoti** come un unico grande delimitatore. È perfetto per output "sporchi" come quello di ls -l.
    - *Sintassi base:* awk '{print $5, $9}' (stampa la colonna 5 e la colonna 9).

2.3 Ordinamento: sort

Per default, sort ordina in modo **lessicale** (come un dizionario: 10 viene prima di 2, perché 1 viene prima di 2).

- **-n**: Forza l'ordinamento **Numerico** (2 viene prima di 10).
- **-h**: Forza l'ordinamento numerico comprendendo i suffissi umani (K, M, G).
- **-k [num]**: Specifica quale colonna (Key) usare per l'ordinamento (es. -k5 per la quinta

colonna).
- **-r**: Reverse, inverte l'ordine.

2.4 Generazione Casuale: $RANDOM vs shuf

- **$RANDOM**: È una variabile interna della shell (non un comando) che genera un numero intero casuale tra **0 e 32.767**. Non è adatta per processare file più lunghi di 32mila righe.
- **shuf**: È il filtro perfetto per la casualità. Prende un flusso di testo e ne scombussola completamente l'ordine (crea permutazioni casuali), indipendentemente da quante milioni di righe ci siano.

3. Espressioni Regolari (Regex)

Le Regex (da usare con grep -E o egrep) sono pattern per trovare sequenze di testo complesse.

L'opzione fondamentale da ricordare per gli script è **grep -o**: invece di stampare tutta la riga dove ha trovato la corrispondenza, stampa *esclusivamente* la stringa matchata.

Anatomia base di una Regex:

- **^**: Ancora di Inizio riga.
- **$**: Ancora di Fine riga.
- **[...]**: Charset (Insieme di caratteri). Es. [a-z] (qualsiasi lettera minuscola), [0-9] (qualsiasi numero).
- **.**: Metacarattere che indica "un carattere qualsiasi". Per cercare un punto vero e proprio, va "escapato" con il backslash: **\.**

Moltiplicatori (Quantificatori):

- **+**: L'elemento precedente deve apparire *una o più volte*.
- **{n}**: L'elemento precedente deve apparire *esattamente n volte*.
- **{n,m}**: L'elemento precedente deve apparire *da n a m volte*.

4. Laboratorio: Esercizi Svolti e Commentati

Di seguito la risoluzione degli esercizi proposti sulle slide, con la spiegazione logica.

Esercizi 2 e 3: Estrazione con head e tail

Obiettivo: Estrarre dal file /usr/share/dict/words le righe dalla 30.000 alla 30.020.

Approccio 1: Rispetto all'inizio del file (Metodo Efficiente)

Vogliamo il blocco fisico in alto nel file (zona delle parole con la "B").

tail -n +30000 /usr/share/dict/words | head -n 21

Spiegazione: tail salta le prime 29.999 righe e stampa il resto. La pipe lo passa ad head, che prende solo le prime 21 righe (dalla 30k alla 30.020) e chiude l'operazione senza leggere il resto del dizionario.

Approccio 2: Rispetto alla fine del file (Inefficiente e posizione diversa)

Se usiamo i riferimenti dal basso, pescheremo parole completamente diverse (zona "P").

```
tail -n 30020 /usr/share/dict/words | head -n 20
```

Spiegazione: Prende le ultime 30.020 righe del file e ne tiene solo le prime 20 in alto.

L'Ottimizzazione del Sistemista (L'Hint wc -l)

Per estrarre un blocco dal fondo in modo efficiente, si contano prima le righe totali.

1. wc -l /usr/share/dict/words -> Risultato: 104.334
2. Calcolo: 104.334 - 30.020 = 74.314
3. Esecuzione efficiente dall'alto:

```
head -n 74334 /usr/share/dict/words | tail -n 20
```

Esercizio 4: Estrazione di Colonne (awk)

Obiettivo: Estrarre dimensioni e nomi dei file da ls -l /usr/bin.

```
ls -l /usr/bin | awk '{print $5, $9}'
```

Spiegazione: cut -d' ' fallisce miseramente a causa degli spazi multipli usati per impaginare l'output di ls -l. awk risolve il problema alla radice considerando colonne fisse separate da "spazio vuoto".

Esercizio 5: Ordinamento (sort vs ls)

Obiettivo: Ordinare i file di /usr/bin per dimensione in modo corretto.

Metodo 1 (Pipeline completa):

```
ls -l /usr/bin | sort -n -k5
```

Spiegazione: sort viene forzato a leggere in modo Numerico (-n) basandosi unicamente sulla

quinta colonna (-k5), trascinandosi dietro però l'intera riga.

Metodo 2 (Ottimizzazione nativa):

ls -lS /usr/bin

Spiegazione: Invece di usare le pipe, deleghiamo il lavoro al programma nativo. ls conosce già la dimensione dei file senza dover "leggere" testo a schermo. L'opzione -S maiuscola lo obbliga a stampare i file dal più grande al più piccolo. Aggiungendo -r (-lSr) si inverte l'ordine.

Esercizio 6: Regex e Shuf

Task A: Estrarre indirizzi IP della macchina.

ip a | grep -E -o '([0-9]{1,3}\.){3}[0-9]{1,3}'

Spiegazione: * [0-9]{1,3}\. : Cerca da 1 a 3 numeri seguiti da un punto (es. 192.).

- (...) : Raggruppa l'espressione precedente.
- {3} : Ripete il gruppo 3 volte (es. 192.168.1.).
- [0-9]{1,3} : Chiude cercando gli ultimi 1-3 numeri senza punto finale.
- L'opzione -o si assicura di scartare tutto il testo inutile attorno all'IP.

Task B: Estrarre una parola casuale formata solo da minuscole, lunga da 4 a 7 caratteri.

egrep '^[a-z]{4,7}$' /usr/share/dict/words | shuf | head -n 1

Spiegazione: 1. egrep (o grep -E): Usa l'ancora iniziale ^ e finale $, assieme al charset [a-z] e al moltiplicatore {4,7}, per filtrare *solo* le parole perfette.

2. La variabile $RANDOM viene scartata perché non superando quota 32.767 non potrebbe mai pescare le parole nella seconda metà del dizionario.

3. Si usa shuf che prende il mazzo di migliaia di parole filtrate e le "mescola" casualmente.

4. head -n 1 pesca la carta in cima al mazzo.

5. Appendice: Gestione Dischi e Partizioni

Note riassuntive sulle procedure di System Administration hardware.

Per ridimensionare una partizione (es. per farle spazio e crearne una nuova) senza perdere dati, l'ordine logico delle operazioni è **Software -> Hardware**:

1. **Smontaggio:** sudo umount /dev/sdb1
2. **Controllo Integrità:** sudo fsck -f /dev/sdb1 (Obbligatorio prima del resize).
3. **Resize Filesystem:** sudo resize2fs /dev/sdb1 500M (Il software ora pensa che il disco sia di 500M).
4. **Resize Hardware (fdisk):** * Si entra in fdisk /dev/sdb. Si cancella la partizione 1 e la si ricrea della nuova dimensione (es. +500M).
    - *Attenzione cruciale:* Deve iniziare dallo **stesso identico settore iniziale** (es. 2048) di prima.
    - Alla domanda sulla rimozione della "signature ext4", rispondere rigorosamente **NO** per conservare i dati e il filesystem appena ristretto.
5. **Montaggio Permanente:** Si modifica il file /etc/fstab aggiungendo ad esempio: /dev/sdb2 /cartella_mount ext4 defaults 0 2