ESERCITAZIONE DI GIOVEDÌ 14/03/2024

I vari esercizi sono suddivisi nelle varie tematiche analizzate a lezione!

RIDIREZIONE A VUOTO

1) Usando la sola ridirezione dello standard output, azzerare un file precedentemente creato (nel caso il contenuto sia importante farne prima una copia, casomai preservando i timestamps!)
2) Usando la sola ridirezione dello standard output, creare un file con un nome non esistente nella directory di sistema tmp.

SORT

3) Per questo esercizio usare il file di nome *prova* (quello creato nella precedente esercitazione), ma prima di usarlo controllare che ci siano più parole in ogni linea e, in caso contrario, editare prima il file per fare in modo appunto di avere più parole in ogni linea. Quindi. usando la ridirezione dello standard input e il comando-filtro sort, verificare se il file di nome *prova* è ordinato alfabeticamente.
4) Usando la ridirezione dello standard input e il comando-filtro sort, mostrare il contenuto del file *prova* ordinato secondo il normale ordinamento alfabetico.
5) Usando la ridirezione dello standard input e il comando-filtro sort, mostrare il contenuto del file *prova* ordinato invertendo il normale ordinamento alfabetico.
6) Per questo esercizio usare sempre il file di nome *prova* (come per il punto 3), ma prima di usarlo controllare che ci siano più parole all'inizio di alcune linee che inizino sia con lettera maiuscola che minuscola, in caso contrario, editare prima il file per fare in modo appunto di avere il formato sopra indicato. Quindi, usando la ridirezione dello standard input e il comando-filtro sort, mostrare il contenuto del file *prova* ordinato secondo il normale ordinamento alfabetico, ma ignorando la differenza fra maiuscole e minuscole.
7) Usando la ridirezione dello standard input e il comando-filtro sort, mostrare il contenuto del file *prova* ordinato secondo il normale ordinamento alfabetico (come nel punto 4), ma ridirigendo lo standard output su un file di nome *ordinato*.
8) Usando la ridirezione dello standard input e il comando-filtro sort, verificare se il file di nome *ordinato* (ottenuto con il precedente esercizio) è ordinato alfabeticamente, in modo analogo a quanto fatto nel punto 3) sul file di nome prova.
9) Per questo esercizio usare sempre il file di nome *prova* (come per il punto 3 e 6), ma prima di usarlo controllare che ci siano più righe duplicate o triplicate, in caso contrario, editare prima il file per fare in modo appunto di avere il formato sopra indicato. Quindi, usando la ridirezione dello standard input e il comando-filtro sort, mostrare il contenuto del file *prova* ordinato, ma attuando l'ordinamento alfabetico senza replicazioni e scrivendo lo standard output su un file di nome *ordinato-senza-doppi*.
10) Facendo riferimento ad uno qualunque degli esercizi precedenti sul comando-filtro SORT (anche per tutti), provare anche la versione comando SORT e quindi senza ridirezione dello standard input! Attenzione però che per produrre dei risultati che vanno salvati, come nel punto 9) (ma anche in altri), la ridirezione dello standard output invece va usata.

GREP

11) Usando la ridirezione dello standard input e il comando-filtro grep, cercare nel file *ordinato-senza-doppi* le linee che contengono una certa stringa (o anche un semplice carattere).
12) Usando la ridirezione dello standard input e il comando-filtro grep, cercare nel file *ordinato-senza-doppi* le linee che contengono una certa stringa (o anche un semplice carattere), come fatto precedentemente, mostrando anche i numeri di linea.

13) Assicurandosi di avere nel file *ordinato-senza-doppi* la stessa stringa scritta in maiuscola e in maiuscolo almeno 2-3 volte (altrimenti modificare il file in tal senso), usando la ridirezione dello standard input e il comando-filtro grep, cercare nel file *ordinato-senza-doppi* le linee che contengono una certa stringa (o anche un semplice carattere) ignorando maiuscole/minuscole.
14) Usando la ridirezione dello standard input e il comando-filtro grep, cercare nel file *ordinato-senza-doppi* le linee che NON contengono una certa stringa (o anche un semplice carattere).
15) Usando la ridirezione dello standard input e il comando-filtro grep, cercare nel file *ordinato-senza-doppi* le linee che INIZIANO per una certa stringa (o anche un semplice carattere).
16) Usando la ridirezione dello standard input e il comando-filtro grep, cercare nel file *ordinato-senza-doppi* le linee che TERMINANO per una certa stringa (o anche un semplice carattere).
17) Assicurandosi di avere nel file *ordinato-senza-doppi* almeno una linea che termina per il carattere '.' (PUNTO) (altrimenti modificare il file in tal senso), usando la ridirezione dello standard input e il comando-filtro grep, cercare nel file *ordinato-senza-doppi* le linee che TERMINANO per il carattere '.' (PUNTO).
18) Utilizzando la soluzione di uno degli esercizi precedenti sul GREP, ridirigere lo standard output in un file di nome *prova-grep*.
19) Facendo riferimento ad uno qualunque degli esercizi precedenti sul comando-filtro GREP (anche per tutti), provare anche la versione comando GREP e quindi senza ridirezione dello standard input! ! Attenzione però che per produrre dei risultati che vanno salvati, come nel punto 18), la ridirezione dello standard output invece va usata.

WC

20) Usando la ridirezione dello standard input e il comando-filtro wc, contare le linee del file *p.txt* (quello creato nella precedente esercitazione).
21) Usando la ridirezione dello standard input e il comando-filtro wc, contare i caratteri del file *p.txt*.
22) Usando la ridirezione dello standard input e il comando-filtro wc, contare le parole del file *p.txt*.
23) Verificare che cosa cambia nell'output dei precedenti 3 esercizi se invece che il comando-filtro wc si usa il comando wc!
24) Usando il comando wc su un file di nome *pippo* (che non deve esistere) e ridirigendo lo standard error su /dev/null, verificare il valore di ritorno del comando.
25) Utilizzando il comando ps in piping con il filtro tee temp e in piping con wc –l, verificare: a) il numero visualizzato; b) il contenuto del file di nome temp creato dal filtro tee (supponendo di non avere lanciato alcun comando in background!).

HEAD e TAIL

26) Usando la ridirezione dello standard input e il comando-filtro head, selezionare le prime 10 linee del file *p.txt* (quello creato nella precedente esercitazione).
27) Usando la ridirezione dello standard input e il comando-filtro head, selezionare la prima linea del file *p.txt*.
28) Usando la ridirezione dello standard input e il comando-filtro head, selezionare le prime 3 linee del file *p.txt*.
29) Usando la ridirezione dello standard input e il comando-filtro tail, selezionare le ultime 10 linee del file *p.txt*.
30) Usando la ridirezione dello standard input e il comando-filtro tail, selezionare l'ultima linea del file *p.txt*.
31) Usando la ridirezione dello standard input e il comando-filtro tail, selezionare le ultime 3 linee del file *p.txt*.
32) Utilizzando la soluzione di uno degli esercizi precedenti su HEAD e TAIL, ridirigere lo standard output in un file di nome *prova-head* o *prova-tail* a seconda dei casi.
33) Facendo riferimento ad uno qualunque degli esercizi precedenti sui comandi-filtro HEAD e TAIL (anche per tutti), provare anche le versioni comando HEAD e TAIL e quindi senza ridirezione dello standard input! Attenzione però che per produrre dei risultati che vanno salvati, come nel punto 32), la ridirezione dello standard output invece va usata.
34) Utilizzando il piping dei comandi, isolare in un file di nome *p.txt.terza* la terza linea a partire dall'inizio del file *p.txt*.

35) Utilizzando il piping dei comandi, isolare in un file di nome *p.txt.terzultima* la terza linea a partire dalla fine del file *p.txt*.

REV

36) Usando la ridirezione dello standard input e il comando-filtro rev, verificarne il funzionamento su almeno uno dei file a disposizione.

METACARATTERI [e]

37) Utilizzando esclusivamente la ridirezione a vuoto, creare diversi file con nomi che iniziano e terminano con varie lettere dell'alfabeto maiuscole e minuscole e con numeri; creare anche un paio di directory con l'opportuno comando. Verificare quindi il comportamento dei comandi:
    a) echo [a-z]*
    b) echo [A-Z]*
    c) echo [0-9]*
    d) echo *[a-z]
    e) echo *[A-Z]
    f) echo *[0-9]
38) Rifare i comandi precedenti usando la negazione: ad esempio per a) echo [!a-z]*
39) Utilizzare nuovamente i pattern precedenti utilizzando però il comando ls –l: che cosa cambia?

VARIABILI DI SHELL E DI AMBIENTE

40) Memorizzare in una variabile di shell di nome *a* un valore scelto a proprio piacimento, quindi provare ad eseguire il file comandi `prova.sh` visto a lezione (scaricabile da GITHUB). Quale è il risultato?
41) Rendere la variabile *a* variabile di ambiente e riprovare ad eseguire `prova.sh`. Quale è il risultato?
42) Verificare con il comando env, la presenza della variabile *a* nell'ambiente.
43) Provare ad eseguire il file comandi `prova1.sh` visto a lezione (scaricabile da GITHUB). Quale è il risultato?
44) Verificare il valore della variabile *a* nella shell di partenza.
45) Provare ad eseguire il file comandi `prova1.sh` visto a lezione (scaricabile da GITHUB) usando sh -x. Quale è il risultato?
46) Provare ad eseguire il file comandi `provacomandi.sh` visto a lezione (scaricabile da GITHUB). Quale è il risultato?
47) Verificare il valore della directory corrente e della variabile *PATH* nella shell di partenza.
48) Memorizzare in una variabile di shell di nome *x*, il numero di linee del file *p.txt* e quindi visualizzarne il valore.
49) Memorizzare in una variabile di shell di nome *y*, il numero di linee del file *prova* e quindi visualizzarne il valore.
50) Memorizzare in una variabile di shell di nome *z*, la somma della variabile x e della variabile y e visualizzarne il valore. Provare anche ad effettuare tutte le altre operazioni aritmetiche e quindi sottrazione, moltiplicazione, divisione intera e resto intero della divisione!
51) Con un editor, scrivere un file comandi *prova-variabili.sh* che visualizzi il valore delle variabili di shell *x*, *y* e *z* inserendo anche dei commenti significativi. Rendere eseguibile tale file comandi (verificare che sia eseguibile con ls –l prova-variabili.sh) e mandarlo in esecuzione. Quale è il risultato?
52) Rendere le variabili *x*, *y* e *z* variabili di ambiente e riprovare ad eseguire *prova-variabili.sh*. Quale è il risultato?
53) Verificare con il comando env, la presenza di *x*, *y* e *z* nell'ambiente.
54) Copiare il file comandi *prova-variabili.sh* nel file comandi di nome *prova-variabili-bis.sh*, quindi aggiungere in fondo un comando che modifica il valore della variabile *z* e un comando che visualizza nuovamente il valore, sempre inserendo commenti significativi. Provare ad eseguire *prova-variabili-bis.sh*. Quale è il risultato? Quale valore ha la variabile *z* nella shell interattiva di partenza?