LAB COMPLETO

Managing user password aging

- `usermod -L <user>`: blocca l'account specificato
- `su - <user>`: switcha all'utente specificato
- `su -`: torna su root
- `usermod -U <user>`: sblocca l'account precedentemente bloccato
- `chage -M [days] <user>`: imposta il periodo massimo (in giorni) per cui una password può essere utilizzata prima che l'utente sia obbligato a cambiarla
- `chage -l <user>`: mostra le informazioni relative alla scadenza della password e dell'account per un determinato utente
- `chage -d 0 <user>`: forza l'utente a cambiare la password al prossimo login (in questo caso: modifica la data dell'ultimo cambio password a oggi, equivalente a `0` giorni)
- `date -d "+[date] days"`: calcola e visualizza la data di tot giorni specificati nel futuro rispetto alla data attuale
- `chage -E [year-month-day] <user>`: imposta la data di scadenza dell'account utente. dopo questa data, l'utente non potrà più accedere
- `chage -E -1 romeo`: per rimuovere la scadenza dell'account

Managing Groups using command-line tools

- `groupadd -g GID group name`: crea un nuovo gruppo con un identificativo specifico
- `groupadd <group name>`: crea un nuovo gruppo con un GID assegnato automaticamente dal sistema
- `usermod -G <group name> <user name>`: aggiunge un utente (già esistente) a un gruppo secondario, sostituendo i gruppi secondari a cui apparteneva in precedenza.
- `usermod -aG <group name> <user name>`: aggiunge l'utente a un nuovo gruppo senza rimuoverlo dagli altri.
- `id <user name>`: mostra informazioni sull'utente specificato: UID, GID, gruppi secondari
- `useradd <user name>`: crea un nuovo account utente

Creating users using command-line tools

- `su -`: diventa root
- `useradd <user name>`: aggiungere un utente
- `passwd <user name>`: cambiare la password dell'utente
- `grep <user name> /etc/passwd` & `grep <user name> /etc/shadow`: per assicurarsi che l'utente sia stato aggiunto correttamente

Managing file security from the command line

1. Gestione Gruppi

- `groupadd` : crea un nuovo gruppo
  `groupadd [opzioni] nome_gruppo`
  opzioni comuni:
  `-g GID` specifica l'id del gruppo
- `grep` : cerca pattern in file o output
  `grep [opzioni] pattern file`

2. Gestione Utenti

- `useradd` : crea un nuovo utente
  `useradd [opzioni] nome_utente`
  opzioni comuni:
  - `-m` crea la home directory
  - `-g` gruppo primario
  - `-G` gruppi supplementari
- `passwd` : cambia la password di un utente
  `passwd [nome_utente]`
  senza argomenti cambia la password dell'utente corrente
- `usermod` : modifica un account utente esistente
  `usermod [opzioni] nome_utente`
  opzioni chiave:
  - `-aG` aggiunge a gruppi supplementari
  - `-l` cambia il nome login
- `id` : mostra UID, GID e gruppi di un utente
  `id [nome_utente]`

3. Gestione Permessi

- `chown` : cambia proprietario e gruppo di file/directory
  `chown [opzioni] utente:gruppo file`
- `chmod` : modifica i permessi di accesso
  `chmod u+rwx, g+rx, o-r file`
  permessi base:
  - `r` - lettura
  - `w` - scrittura
  - `x` - esecuzione
  - `o` - si riferisce alla categoria `others` (altri utenti, non proprietari nè membri del gruppo associato al file)
  - `-` - indica la rimozione di un permesso
- `newgrp` : cambia temporaneamente il gruppo primario
  `newgrp nome_gruppo`
  utile per ereditari i permessi di un gruppo specifico

4. Gestione File System

- `mkdir` - crea directory
  `mkdir [opzioni] nome_directory`
  opzione `-p` crea path completo

- `touch` - crea file vuoti o aggiorna timestamp
  `touch nome_file`
- `ls` - lista contenuto directory
  opzioni utili:
  - `-l` formato lungo con permessi
  - `-a` mostra file nascosti
  - `-d` info sulla directory stessa

5. Controllo accessi

- `su` - cambia utente
  `su - nome_utente` (il `-` carica l'ambienta completo)
- `sudo` - esegue comandi come superuser
  `sudo comando`
  richiede privilegi sudoers

6. Strumenti diagnostici

- `find` - cerca file nel sistema
  `find [path] [espressione]`
  es. `sudo find / -name minniefile`
- `cat` -mostra contenuto file
  `cat file`
  usato anche per concatenare file: `cat file1 file2 > tot`

Background and Foreground processes

- `while true; do echo -n "rock " >> ~/outfile; sleep 1; done)`
  - `echo -n "[word]"` : scrive la stringa contenuta tra " " nel file `~/outfile` senza andare a capo (`-n` evita il newline)
  - `>> ~/outfile` : aggiunge (append) il testo alla fine del file outfile nella home directory `~`
  - `sleep 1` : attende 1 secondo prima di ripetere il ciclo
  - `done` : indica la fine del blocco while
    il comando continua a scrivere "rock " nel file ogni secondo, finché non viene interrotto manualmente (`ctrl+c`)
- `while true; do echo -n "rock " >> ~/outfile; sleep 1; done) &`
  - uguale a prima, ma con l'aggiunta di `&` , che lo esegue in background, in modo tale che il terminale rimanga libero per eseguire altri comandi
- `tail -f outfile`
  - `tail` : mostra le ultime righe di un file
  - `-f` (follow): mantiene aperto il file e aggiorna in tempo reale il contenuto mostrato, utile per monitorare file di log o file in scrittura continua
    mentre outfile viene aggiornato dal loop in background, questo comando mostrerà ogni nuova scrittura in tempo reale
- `jobs` : mostra i processi in background attualmente attivi nella sessione del terminale. L'output tipico è:

```
[1]+  Running                 while true; do echo -n "rock " >> ~/outfile; sleep
1; done &
```

dove:

- `[1]` : ID del job;
- `+` : indica l'ultimo job avviato;
- `Running / Stopped` : stato del job
  Il risultato è la lista dei job attivi in background

- `fg` : porta in foreground l'ultimo job sospeso o in background. Se hai un solo job in background, eseguire `fg` lo riporta in primo piano
  Risultato: il processo torna attivo in foreground nel terminale
  foreground := si riferisce a un processo attivo che sta interagendo direttamente con il terminale, quando un processo è in foreground:
  - occupa il terminale e riceve direttamente l'input dell'utente
  - può stampare output direttamente sullo schermo
  - deve terminare o essere sospeso prima che tu possa eseguire altri comandi nello stesso terminale
- `fg %[ID job]` : riporta in foreground un job specifico, indicato con il suo `[ID job]` dall'output di `jobs`
- `ps j` : mostra informazioni sui processi in formato job control. L'output tipi include:
  - PGID (Process Group ID): gruppo del processo
  - SID (Session ID): sessione a cui appartiene il processo
  - TTY (Terminal Type): terminale da cui è stato avviato
  - STAT: Stato del processo (`R` =Running; `S` =Sleeping; `T` =Stopped, ecc.)
  - COMMAND: comando eseguito
  `ctrl+c` ⇒ interrompe
  `ctrl+z` ⇒ sospende

Killing processes

- `kill -SIGSTOP %number` : invia il segnale `SIGSTOP` a un job in background gestito dalla shell
  - `%number` : si riferisce a un job ID
  - `SIGSTOP` : sospende immediatamente l'esecuzione del processo, non può essere ignorato.
- `kill -SIGTERM %number` : invia il segnale `SIGTERM` a un job in background
  - `SIGTERM` : richiede una terminazione "gentile" del processo. Il processo può ignorarlo o eseguire operazioni di cleanup prima di chiudersi, se il processo non risponde a `SIGTERM` , si può usare `SIGKILL` che ne forza la terminazione
- `kill -SIGCONT %number` : invia il segnale `SIGCONT` a un job sospeso
  - `SIGCONT` : riprende l'esecuzione di un processo precedentemente sospeso con `SIGSTOP` o `SIGSTP`
  - utile per riattivare processi fermati (es. dopo un `ctrl+z` nella shell)
- `pkill -SIGTERM tail` : invio del segnale `SIGTERM` a tutti i processi il cui nome contiene "tail"

Using grep with Regular Expressions

- comandi utili `vim`

    - `i` → inserimento (scrivere testo)
    - `Esc` → torna in modalità comando
    - `:` → modalità comando
    - `v` → modalità visuale (per selezionare il testo)
    - `V` → modalità visuale a riga
    - `ctrl + v` → modalità visuale a blocchi
    - `R` → sostituzione (scrive sovrascrivendo)
    - **gestione file**:
        - `:w` → salva il file
        - `:q` → esci
        - `:wq` o `zz` salva ed esci
        - `:q!` → esci senza salvare
        - `:x` → salva ed esci (solo se ci sono modifiche)
        - `:e filename` → apri un file
        - `:r filename` → inserisci il contenuto di un file corrente
    - **ricerca e sostituzione**
        - `/parola` → cerca parola
        - `n` → va alla prossima occorrenza
        - `N` → vai all'occorrenza precedente
- `grep 'dog' animals.txt` → cerca la stringa esatta `dog` nel file `animals.txt`
    - case-sensitive (distingue maiuscole/minuscole)
- `grep -i 'cat' animals.txt` → cerca la stringa `cat` ignorando maiuscole/minuscole
    - `-i` → modalità case-insensitive
- `grep '^C' animals.txt` → cerca le righe che iniziano con la lettera C
- `grep 't$' animals.txt` → cerca le righe che finiscono con la lettera t
- `grep '^dog$' animals.txt` → cerca le righe che contengono solo la parola `dog`
- `grep '^...$' animals.txt` → cerca le righe che hanno una parola di tre lettere
- `grep '^[bcd]' animals.txt` → cerca le righe che iniziano con b, c o d
- `grep '^[^a]' animals.txt` → cerca le righe che non iniziano con la lettera a
- `grep -e 'dog' -e 'cat' animals.txt` → cerca le righe che contengono o la parola `dog` o la parola `cat`
- `grep 'o.*g' animals.txt` → cerca le righe che contengono la lettera `o` seguita da un qualunque numero di lettere, che termina con `g`
- `grep 'e\+' animals.txt` → cerca le righe che hanno una parola che contiene almeno una `e`
- `grep 'e\{2\}' animals.txt` → cerca le righe che hanno una parola con esattamente 2 `e`
- `grep '\bant\b' animals.txt` → cerca la parola intera `ant`
    - `\b` → word boundary (confine di parola)
- `grep -v 'dog' animals.txt` → mostra le righe che non contengono `dog`
    - `-v` inverte il match
- `grep -r 'cat' .` → cerca `cat` in tutti i file nella directory corrente e sottodirectory
    - `-r` → ricorsivo

Configuring a web server

- `vim /etc/httpd/conf/httpd.conf` per cambiare la riga che inizia con `ServerAdmin` in: `ServerAdmin webmaster@example.com`
- creare la default page:
  creare `index.html` nella directory `/var/www/html/` e aggiungere il contenuto in index.html con `vim`
- startare e abilitare httpd service
  `systemctl start httpd.service`
  `systemctl enable httpd.service`
- aprire tutte le porte rilevanti per http sul firewall del server
  `systemctl stop firewalld`
  `systemctl disable firewalld`

Configuring a virtual host

- Crea la directory per il sito:
  `mkdir -p /srv/{default,www.class-example.com}/www
- crea il file index.html:
  `vim /srv/default/www/index.html
  inserire il testo desiderato e salvare
- per il sito:
  `vim /srv/www.class-example.com/www/index.html
  inserire il testo e salvare
- disabilitare SELinux (temporaneamente e permanentemente)
  - temporaneamente: `setenforce 0
  - permanentemente: `vim /etc/selinux/config` e cambiare la linea "`SELINUX=enforcing`" in "`SELINUX=disabled`"
- configurare il virtual host di default:
  creare il file di configurazione:
  `vim /etc/httpd/conf.d/00-default-vhost.conf
  e inserire:

```
        <VirtualHost _default_:80>
            DocumentRoot /srv/default/www
            CustomLog "logs/default-vhost.log" combined
        </VirtualHost>
        <Directory /srv/default/www>
            Require all granted
        </Directory>
```

- configurare il virtual host per www.class-example.com:
  creare il file di configurazione:
  `vim /etc/httpd/conf.d/01-www.class-example.com-vhost.conf
  e inserire:

```
        <VirtualHost *:80>
            ServerName www.class-example.com
            ServerAlias www
            DocumentRoot /srv/www.class-example.com/www
```

```
        CustomLog "logs/www.class-example.com.log" combined
    </VirtualHost>
    <Directory /srv/www.class-example.com/www>
        Require all granted
    </Directory>
```

- Avvia e abilita il servizio Apache:
  avvia il servizio Apache:
  `systemctl start httpd abilita all'invio:` sudo systemctl enable httpd
- configura il firewall:
  `firewall-cmd --permanent --add-service=httpd` firewall-cmd –reload
  in caso di messaggio d'errore tipo: " `FirewallD is not running` ", attivare il firewall con:
  `sudo systemctl start firewalld` sudo systemctl enable firewalld
  e per verificarne lo stato:
  `sudo systemctl status firewalld → (Active: active(running)) in caso di messaggio d'errore tipo: "` Error: INVALID_SERVICE: Zone 'FedoraServer': 'httpd' not among existing services`" riprovare a configurare il firewall con "http" invece che "httpd"
- Modificare il file /etc/hosts su Fedora(Guest):
  trova l'IP con:
  `ip a s modificare /etc/hosts:` vim /etc/hosts
  e inserire
  `(HOSTIP trovato) www www.class-example.com default.class-example.com default
- testa la configurazione
  1. dal terminale:
     `curl http://127.0.0.1` curl http://default.class-example.com
     `curl http://www.class-example.com
  2. da browser:
     http://127.0.0.1
     http://default.class-example.com
     http://www.class-example.com

PROBLEMA:

avevo già un file index.html e quando provo a testare la configurazione esce il testo precedente e non "coming soon!"
controllando il file index.html con:
`cat /srv/default/www/index.html il comando mostra il contenuto attuale del file. Nel mio caso il contenuto è corretto ma facendo "` curl http://default.class-example.com`" esce ancora il contenuto precedente.

La causa potrebbe essere:

- cache del server
- configurazione errata del virtual host
- errore nella risoluzione del DNS

passaggi per risolvere il problema:

1. controllare la configurazione del virtual host:
   `vim /etc/httpd/conf.d/00-default-vhost.conf assicurarsi che la` DocumentRoot `punti correttamente a` /srv/default/www`:

```
        <VirtualHost _default_:80>
            DocumentRoot /srv/default/www
            CustomLog "logs/default-vhost.log" combined
        </VirtualHost>
            <Directory /srv/default/www>
            Require all granted
        </Directory>
```

2. cancellare la cache di Apache:
   `sudo systemctl restart httpd
3. verificare i permessi delle directory, dobbiamo essere sicuri che la directory e il file index.html abbiano i permessi corretti:
   `sudo chown -R apache:apache /srv/default/www` sudo chmod -R 755 /srv/default/www
4. verificare che il sistema stia risolvendo correttamente il nome `default.class-example.com`:
   `vim /etc/hosts e assicurarsi che la linea con il nostro HOSTIP sia:` HOSTIP default.class-example.com default

   possiamo inoltre testare la risoluzione del DNS con:
   `ping default.class-example.com
   che dovrebbe restituire l'IP corretto del server.

Riprovare a testare la configurazione.

Combining partial debug and verbose modes

```bash
#!/bin/bash

# Directory containing files
DIR="/home/student/"

# Start verbose mode
set -v

# Loop through each file in the directory
for FILE in "$DIR"/*; do
    # Start debug mode for this block
    set -x

    # Check if the file exists
    if [ -e "$FILE" ]; then
        # Get the file size
        SIZE=$(stat --printf='%s' "$FILE")

    # Output the file information
    echo "File: $FILE has size: $SIZE bytes."
    else
    echo "File: $FILE does not exist."
```

```
    fi

    # End debug mode
    set +x
done

# End verbose mode
set +v
```

Questo script Bash (`file_info.sh`) ha lo scopo di esplorare i file in una directory specificata (`/home/student/`), mostrare le loro dimensioni e utilizzare modalità di debug e verbose per una migliore comprensione dell'esecuzione dello script.

Riga per riga

- `#!/bin/bash` → specifica che lo script deve essere eseguito usando Bash
- `DIR="/home/student/"` → definisce una variabile `DIR` che contiene il percorso della directory da analizzare
- `set -v` → attiva la modalità "verbose", che stampa ogni comando eseguito prima della sua esecuzione
- `for FILE in "$DIR"/*; do` → scorre tutti i file presenti nella directory `/home/student/`
- `set -x` → attiva la modalità di debug, che mostra dettagliatamente ogni comando eseguito e i suoi argomenti
- `if [ -e "$FILE" ]; then` → controlla se il file esiste (`-e "$FILE"` verifica l'esistenza del file)
- `SIZE=$(stat --printf="%s" "$FILE")` → ottiene la dimensione del file in byte usando il comando `stat`
- `echo "File: $FILE has size: $SIZE bytes."` → stampa a schermo il nome del file e la sua dimensione in byte
- `else echo "File: $FILE does not exist."` → se il file non esiste, stampa un messaggio di errore.
- `fi` → fine dell'istruzione `if`
- `set +x` → disattiva la modalità di debug
- `done` → fine del ciclo `for` : ripete il processo per tutti i file nella directory
- `set +v` disattiva la modalità verbose
  per eseguire il programma:
- `chmod +x file_info.sh` → per rendere lo script eseguibile
  - `+x` concede i permessi di esecuzione al file
- `./file_info.sh` → per eseguire lo script

Writing bash scripts

Consegna:
Your company provides hosting service to customers, and you have been tasked with writing a Bash shell script called /usr/local/sbin/mkaccounts to automate the process of creating accounts for new customers. At the end of each day, a colon-separated data file called /tmp/support/newusers is created and contains information on new customers that have signed up. The script will read through this data file and create a user account for each new customer. The sales department has requested that the script also generate a report breaking down the new customers by support tier, so sales staff can stay

informed regarding trends of support tier purchases. For each support tier, they would like the report to detail the total number of new customers and the percentage of the day's new customers that chose the tier type.

- creare user file
  - `mkdir /tmp/support/`
  - `cat > /tmp/support/newusers << HERE`
- creare lo script:
  - `vim /usr/local/sbin/mkaccounts`

```
#!/bin/bash

#Variables
NEWUSERSFILE=/tmp/support/newusers

#Loop
for ENTRY in $(cat $NEWUSERSFILE); do
        #Extract first, last, and tier fields
        FIRSTNAME=$(echo $ENTRY | cut -d: -f1)
        LASTNAME=$(echo $ENTRY | cut -d: -f2)
        TIER=$(echo $ENTRY | cut -d: -f4)

        #Make account name
        FIRSTINITIAL=$(echo $FIRSTNAME | cut -c 1 | tr 'A-Z' 'a-z')
        LOWERLASTNAME=$(echo $LASTNAME | tr 'A-Z' 'a-z')
        ACCTNAME=$FIRSTINITIAL$LOWERLASTNAME

        #Create account
        useradd $ACCTNAME -c "$FIRSTNAME $LASTNAME"
done

TOTAL=$(wc -l < $NEWUSERSFILE)
TIER1COUNT=$(grep -c :1$ $NEWUSERSFILE)
TIER2COUNT=$(grep -c :2$ $NEWUSERSFILE)
TIER3COUNT=$(grep -c :3$ $NEWUSERSFILE)
TIER1PCT=$(( $TIER1COUNT * 100 / $TOTAL ))
TIER2PCT=$(( $TIER2COUNT * 100 / $TOTAL ))
TIER3PCT=$(( $TIER3COUNT * 100 / $TOTAL ))

#Print report
echo "\"Tier1\",\"$TIER1COUNT\",\"$TIER1PCT%\""
echo "\"Tier 2\",\"$TIER2COUNT\",\"$TIER2PCT%\""
echo "\"Tier 3\",\"$TIER3COUNT\",\"$TIER3PCT%\""
```

- appunti script:
  - si assume che il percorso del file contenente l'elenco dei nuovi utenti da creare (con cui viene inizializzata la variabile `$NEWUSERSFILE`) abbia le righe formattate come: `Nome:Cognome:Altro:Tier`
  - il comando `$(cat $NEWUSERSFILE)` legge il file riga per riga e assegna ogni riga a `ENTRY`
  - `FIRSTNAME=$(echo $ENTRY | cut -d: -f1)` utilizza il comando `cut` per estrarre parti della riga, separando i campi con `:`:

        - `-d:` → usa `:` come delimitatore
        - `-f1` → primo campo
    - generazione del nome utente:
        - `FIRSTINITIAL`: prende la prima lettera del `FIRSTNAME` con `cut -c 1` e la converte in minuscolo con `tr 'A-Z' 'a-z'`.
        - `LOWERLASTNAME`: converte l'intero `LASTNAME` in minuscolo con `tr 'A-Z' 'a-z'`
        - `ACCTNAME`: concatena la prima iniziale del nome `FIRSTINITIAL` con il cognome in minuscolo `LOWERLASTNAME`
    - creazione dell'account:
        - crea un nuovo utente con `useradd` usando `$ACCTNAME` come nome utente e aggiunge il nome completo come commento con `-c $FIRSTNAME $LASTNAME`
    - conteggio totale degli utenti:
        - `wc -l` restituisce il numero di righe (una per utente) nel file $NEWUSERSFILE
    - conteggio utenti per tier:
        - `grep -c :numero$ $VARFILENAME` conta le righe che terminano con `:numero` in `$VARFILENAME`
    - stampa del report:
        - **virgolette precedute da** `\` (`\"`): il backslash serve per "escapare" le virgolette doppie all'interno della stringa, cioè dire a bash di trattarle come parte della stringa e non come delimitatore. Ad esempio, `\"Tier1\"` fa sì che l'output generi `"Tier1"` (con le virgolette).
- `chmod u+x /usr/local/sbin/mkaccounts` → `u+x` rende il file eseguibile per l'utente proprietario del file:
    - `u` → si riferisce all'utente proprietario del file (user)
    - `+x` → aggiunge il permesso di esecuzione
- `/usr/local/sbin/mkaccounts` → per eseguire lo script
- `tail /etc/passwd` → per vedere se gli utenti sono stati creati correttamente
creare un altro script per eliminare gli account appena creati:
- `vim /usr/local/sbin/rmaccounts`
```bash
#!/bin/bash

#Percorso file con gli utenti creati
NEWUSERSFILE=/tmp/support/newusers

for ENTRY in $(cat $NEWUSERSFILE); do
    FIRSTNAME=$(echo $ENTRY | cut -d: -f1)
    LASTNAME=$(echo $ENTRY | cut -d: -f2)

    FIRSTINITIAL=$(echo "$FIRSTNAME" | cut -c 1 | tr 'A-Z' 'a-z')
    LOWERLASTNAME=$(echo "$LASTNAME" | tr 'A-Z' 'a-z')
    ACCTNAME=$FIRSTINITIAL$LOWERLASTNAME

    if id "$ACCTNAME" &>/dev/null; then
            userdel -r "$ACCTNAME"
            echo "Utente $ACCTNAME eliminato."
    else
```

```
                echo "Utente $ACCTNAME non trovato, skip"
        fi
```

done < "$NEWUSERSFILE"

echo "eliminazione completata"
`` $>/dev/null `-> viene usato in bash per reindirizzare sia l'output standard (stdout) che l'output di errore (stderr) a` /dev/null `, un "buco nero" virtuale: tutto ciò che vi viene inviato viene scartato. -` > `-> reindirizza solo l'output standard -` 2> `-> reindirizza solo gli errori -` &> `-> reindirizza sia output standard che errori -` /dev/null` → dispositivo speciale che scarta tutto l'output

v. 2 - Bash conditionals and control structures

Consegna: Your company provides hosting service to customers, and currently uses a Bash shell script called /usr/local/sbin/mkaccounts to automate the daily task of creating accounts for new customers by processing a colon-separated data file located at /tmp/support/newusers. You have been tasked with extending the script to add the following new features: • The script should accept a command-line argument so that it can either be run in verbose mode or generate a usage message. • When in verbose mode, the script should generate a message to indicate the creation of each account. • Prior to the creation of an account, the script should check existing accounts and report if a conflict or duplication occurs.

```
#!/bin/bash

#Variables
NEWUSERSFILE=/tmp/support/newusers
OPTION=$1

case $OPTION in
        '')
                ;;
        -v)
                VERBOSE=y
                ;;
        -h)
                echo "Usage: $0 [-h|-v]"
                echo
                exit
                ;;
        *)
                echo "Usage: $0 [-h|-v]"
                echo
                exit -1
                ;;
esac

#Loop
for ENTRY in $(cat $NEWUSERSFILE); do

        #Extract first, last, and tier fields
        FIRSTNAME=$(echo $ENTRY | cut -d: -f1)
```

```
        LASTNAME=$(echo $ENTRY | cut -d: -f2)
        TIER=$(echo $ENTRY | cut -d: -f4)

        #Make account name
        FIRSTINITIAL=$(echo $FIRSTNAME | cut -c 1 | tr 'A-Z' 'a-z')
        LOWERLASTNAME=$(echo $LASTNAME | tr 'A-Z' 'a-z')
        ACCTNAME=$FIRSTINITIAL$LOWERLASTNAME

        #controlla se l'user esiste già

        #test per i duplicati e i conflitti
        ACCTEXIST=''
        ACCTEXISTNAME=''

        id $ACCTNAME &> /dev/null

        if [ $? -eq 0 ]; then
                ACCTEXIST=y
                ACCTEXISTNAME="$(grep ^$ACCTNAME: /etc/passwd | cut -f5 -d:)"
        fi

        #test per duplicati e conflitti
        if [ "$ACCTEXIST" = 'y' ] && [ "$ACCTEXISTNAME" = "$FIRSTNAME $LASTNAME" ];
then
                echo "Skippin $ACCTNAME. Duplicate found."
        elif [ "$ACCTEXIST" = 'y' ]; then
                echo "Skipping $ACCTNAME. Conflict found."
        else
                useradd $ACCTNAME -c "$FIRSTNAME $LASTNAME"
                if [ "$VERBOSE" = 'y' ]; then
                        echo "Added $ACCTNAME."
                fi
        fi
done
TOTAL=$(wc -l < $NEWUSERSFILE)
TIER1COUNT=$(grep -c :1$ $NEWUSERSFILE)
TIER2COUNT=$(grep -c :2$ $NEWUSERSFILE)
TIER3COUNT=$(grep -c :3$ $NEWUSERSFILE)

TIER1PCT=$(( $TIER1COUNT * 100 / $TOTAL ))
TIER2PCT=$(( $TIER2COUNT * 100 / $TOTAL ))
TIER3PCT=$(( $TIER3COUNT * 100 / $TOTAL ))

#Print report
echo "\"Tier1\",\"$TIER1COUNT\",\"$TIER1PCT%\""
echo "\"Tier 2\",\"$TIER2COUNT\",\"$TIER2PCT%\""
echo "\"Tier 3\",\"$TIER3COUNT\",\"$TIER3PCT%\""
```

Docker

LAB 1

- `docker ps` → mostra i container attualmente in esecuzione, restituisce un elenco di tutti i container che sono in stato "running", con dettagli come l'ID del container
- `docker ps -a` → mostra tutti i container, sia in esecuzione che non in esecuzione (inclusi quelli fermati).
- `docker images` → elenca tutte le immagini Docker salvate localmente sulla tua macchina. Vengono mostrati dettagli come il repository, il tag, l'ID dell'immagine e la data di creazione.
- `docker search hello-world` → cerca nell'indice pubblico di Docker Hub le immagini che corrispondono al termine di ricerca
- `docker rm CONTAINER ID` → rimuove un container specificato dall'ID (o dal nome) passato al comando. Funziona solo su container che non sono in esecuzione (se il container è in esecuzione, bisogna fermarlo prima con `docker stop`)
- `docker rmi hello-world` → rimuove l'immagine Docker chiamata "hello-world". Se l'immagine è utilizzata da un container (anche se fermo), il comando restituirà un errore a meno che non si rimuovano prima i container associati

LAB 2

- `docker run - -detach - -name web_test nginx:latest` → avvia un nuovo container basato sull'immagine `nginx:latest` in modalità staccata (`- - detach`), cioè il container verrà avviato in background. Il container sarà chiamato `web_test` grazie all'opzione `- -name web_test`
- `docker stop NAME` → ferma un container che sta attualmente girando
- `docker run - -detach -p 80:80 - -name web nginx:latest` → avvia un nuovo container basato sull'immagine `nginx:latest` in modalità staccata e mappa la porta 80 del container alla porta 80 della macchina host (grazie a `-p 80:80`), ciò significa che il container sarà accessibile tramite la porta 80 della tua macchina host.

LAB 3

- `docker rename OldName NewName` → per rinominare un container

LAB 5

- `docker pull` scarica l'immagine.
- `docker save` salva l'immagine in un file.
- `ls -la` controlla il file salvato.
- `docker load` ricarica l'immagine dal file.

LAB 6

- `docker run busybox` → avvia un container senza un comando specifico, se il container non trova un processo da eseguire in foreground, si chiude immediatamente
- `docker run busybox echo "hello from busybox"` → avvia un container, esegue il comando `echo` e dopo l'esecuzione si arresta automaticamente
- `docker run -it busybox sh` → avvia un container in modalità interattiva (`-it`), esegue una shell `sh` all'interno del container, permettendo di interagire con esso. Uscendo dalla shell (`exit`), il container si chiude.

- `docker ps -a -q -f status=exited` → mostra tutti gli ID (`-q` , quiet mode) dei container che si sono fermati (`-f status=exited`), utile per trovare i container terminati che non servono più
- `docker rm $(docker ps -a -q -f status=exited)` → rimuove tutti i container terminati `exited` . Il comando `$(docker ps -a -q -f status=exited)` genera una lista di container terminati, che viene passata a `docker rm` per eliminarli.

Concetti docker

1. **Images**: le immagini docker sono come die blueprint(modelli) per creare i container. Un'immagine contiene tutto ciò che serve per eseguire un'applicazione, inclusi il sistema operativo di base, le dipendenze e il codice dell'applicazione
2. **Container**: sono istanze in esecuzione delle immagini docker. Sono isolati dal sistema operativo host e contengono tutto ciò che serve per eseguire l'applicazione
3. **Docker Daemon**: è il servizio in background che gestisce tutto il funzionamento di docker. Si occupa della creazione, gestione ed esecuzione dei container. È il componente principale di docker, e il client docker comunica con esso.
4. **Docker Client**: è lo strumento a riga di comando (`docker CLI`) che permette agli utenti di interagire con Docker. Quando si utilizzano comandi come `docker run` o `docker pull` , il client invia queste richieste al Docker Daemon per essere eseguite.
5. **Docker Hub**: È il **registro ufficiale delle immagini Docker**, una specie di "negozio online" dove puoi trovare e scaricare immagini già pronte per essere utilizzate.

LAB 7

- `docker run --name static-site-2 -e AUTHOR="andre" -d -p 80:80 dockersamples/static-site`
- `docker run --name static-site -e AUTHOR="andre" -d -P dockersamples/static-site`
  - `-d` → avvia il container in modalità distaccata, cioè in background, senza bloccare il terminale
  - `-P` → mappa tutte le porte esposte dal container su porte casuali del sistema host
  - `-p` → permette di specificare le porte da esporre invece di usare porte casuali
  - `-e` → permette di passare variabili d'ambiente al container
  - `- - name` → permette di assegnare un nome personalizzato al container (invece di uno generato casualmente)
- `docker port NAME` → mostra le porte utilizzate dal container specificato

LAB 8

- `docker run -it --name myfedora -v /localvolume:/usr/localvolume:ro fedora bash` → crea un container, volume montato: `/localvolume` dell'host e `/usr/localvolume` nel container; in modalità di sola lettura `ro` : il container può leggere, ma non scrivere in `/usr/localvolume`
- `docker volume create shared_data` → crea un volume chiamato `shared_data` che Docker gestirà per la persistenza dei dati
- `docker run -dit --name myfedora2 -v shared_data:/data fedora bash`
- `docker run -dit --name myfedora3 -v shared_data:/data fedora bash`
  avvio di due container con un volume permettendo ai due di condividere i file
- `docker exec -it myfedora2 bash`

- `docker exec -it myfedora3 bash`
  accesso ai container
- `docker volume ls` → elenco dei volumi
- `docker volume prune` → eliminazione di tutti i volumi inutilizzati
- `docker volume rm NAME` → eliminazione di un volume specifico

LAB 9

Docker attach

1. **collegarsi a un container in esecuzione con `docker attach`**
   `docker attach` permette di collegare il terminale locale a un container Docker in esecuzione. In pratica, ti permette di vedere l'output del container e interagire con esso come se stessi eseguendo il processo direttamente nel tuo terminale.
   Cose importanti:
   - mostra l'output del processo principale del container (quello definito in ENTRYPOINT o CMD nel Dockerfile)
   - non genera un nuovo processo, ma si collega a quello esistente
   - se il processo non genera output il container è inattivo
   - puoi attaccarti più volte allo stesso container da terminali diversi
   - per uscire dal container:
     - `ctrl p+q`
2. **usare reti bridge definite dall'utente in Docker**
   docker permette di creare reti personalizzate per far comunicare tra di loro i container senza usare la rete predefinita.
   Esempio:
   1. creiamo una rete personalizzata:
   `docker network create - -driver bridge alpine-net`
   `- -driver bridge` specifica che stiamo creando una rete di tipo bridge
   2. avviamo due container Alpine collegati alla rete `alpine-net`:
   `docker run -dit --name alpine1 --network alpine-net alpine`
   `docker run -dit --name alpine2 --network alpine-net alpine`
   `alpine1` e `alpine2` possono comunicare tra di loro, ma non con i container sulla rete predefinita.
   3. creiamo un terzo container senza collegarlo alla rete `alpine-net`:
   `docker run -dit - -name alpine3 alpine`
   `alpine3` è collegato alla rete predefinita di Docker, quindi non può comunicare con `alpine1` e `alpine2`
   4. creiamo un quarto container collegato a entrambe le reti:
   `docker run -dit - -name alpine4 - -network alpine-net alpine`
   `docker network connect bridge alpine4`
   `alpine4` può comunicare sia con `alpine1` e `alpine2` (su `alpine-net`) sia con `alpine3` (sulla rete predefinita)
   Riepilogo:
3. creo una rete personalizzata `alpine-net`
4. avvio quattro container Alpine, alcuni sulla rete `alpine-net`, uno sulla rete predefinita `bridge`
5. connetto `alpine4` sia a `alpine-net` che a `bridge`
6. testo la connettività tra container:

1. mi attacco ad `alpine1` per verificare la comunicazione con `alpine2`: **`docker container attach alpine1`**
2. dentro il container eseguo il ping: **`ping -c 2 alpine2`**
3. esco dal container.
   e con internet:
4. mi attacco ad `alpine1`: **`docker container attach alpine1
5. eseguo il ping verso google: **`ping-c 2 google.com`**