APPELLI

Appello_1 12 gennaio 2024

Exercise 1: Command line and shell environment

- Comeutente exercise1
- Creare uno script bash che stampi in output “Exam1” chiamato exam. Il file dovrà appartenere all’utente exercise1
- Posizionare il file bash “exam”, in una directory all’interno della home directory dell’utente exercise1 (dopo averlo creato), con i giusti permessi di accesso.
- Assicuratevi possa essere lanciato da riga di comando senza specificare il path assoluto, configurando la variabile PATH del solo utente exercise1 in modo appropriato.
- `useradd exercise1`
- `su - exercise1`
- `mkdir mydir`
- `cd mydir`
- `vim exam`
- `#!/bin/bash`
  `echo "exam1"`
- `cd` (per tornare alla home)
- `vim .bashrc`
- scorro e aggiungo `export PATH=$HOME/mydir:$PATH`
- `source ~/.bashrc`

Exercise 2: User and Group

- Create il gruppo students con GID 3500
- Creare l’utente appartenente al gruppo students: student
  - L’utente student avrà le seguenti caratteristiche:
  - UID3010
  - l’account scadrà dopo un anno dalla sua creazione
  - dovrà avere come gruppo secondario: exam
  - Modificare una volta creato l’utente student umask (in modo permanente) così che tutti i file e directory creati dall’utente non abbiano nessun permesso OTHER settato. OWNER e GROUP a piacere
- `groupadd -g 3500 students`
- `sudo usermod -aG students student`
- `sudo usermod -u 3010 student`
- `chage -E 2026-02-05 student`
- `groupadd exam`
- `usermod -G exam student`
- `id student`
- `su - student`

- `vim .bashrc`
- aggiungere in fondo al file `umask 007`
- per applicare la modifica `source .bashrc`

Exercise 3: file permission

- Creare una directory sotto /exam/exercise4 dove gli utenti che possono accedere al gruppo exam potranno condividere files
- Tutti i file creati sotto la directory /exam/exercise4 dovranno essere assegnati automaticamente al gruppo exam
- `sudo mkdir -p /exam/exercise4` → crea la directory, `-p` assicura che le directory intermedie vengano create se non esistono
- `sudo chown :exam /exam/exercise4` → ora la directory appartiene al gruppo `exam`
- `ls -ld /exam/exercise4` → per verificare l'appartenenza della directory
- `sudo chmod 2775 /exam/exercise4` → assicura che tutti i file e directory creati dentro `/exam/exercise4` abbiano automaticamente il gruppo `exam`. Spiegazione `2775`
  - `2` → imposta il bit SGID, che forza l'eredità del gruppo
  - `775` → permette lettura, scrittura ed esecuzione a OWNER e GROUP, ma non a OTHER

Exercise 5: Bash script

- Creare uno script bash sotto /exam/exercise5 chiamato exam_wcounter.sh che accetti in ingresso un numero arbitrario di parametri (al minimo 1) e che elabori i parametri passati uno ad uno per riportare il numero di caratteri per ogni parametro passato e numero di parole (command line wc man wc per le command info)
- Si prevedano i seguenti controlli
  - Se non viene passato almeno un parametro lo script deve restituire un errore e uscire con exit code 10
  - Se vengono passati più di 20 caratteri per entry lo script esca con il testo: “WARNING: Too many characters”
  - Prevedete una funzione chiamata usage da richiamare ogni volta che i controlli falliscono che riporti il seguente testo: USAGE: "At least one word/sentence, no more than 20 characters maximum per word/sentence."

```
#!/bin/bash

# Funzione per mostrare il messaggio di utilizzo
usage() {
    echo "USAGE: At least one word/sentence, no more than 20 characters maximum per
word/sentence."
}

# Controllo se è stato passato almeno un parametro
if [ $# -lt 1 ]; then
    usage
    exit 10
fi

# Elaborazione dei parametri
for param in "$@"; do
```

```
    length=${#param}  # Conta il numero di caratteri

    # Se il parametro ha più di 20 caratteri, mostra un warning e termina
    if [ $length -gt 20 ]; then
        echo "WARNING: Too many characters"
        exit 1
    fi

    word_count=$(echo "$param" | wc -w)  # Conta il numero di parole
    echo "'$param' -> Characters: $length, Words: $word_count"

done
```

- `if [ $# -lt 1 ]; then`
  - `#` indica il numero di parametri passati allo script
  - se il numero di parametri `$#` è inferiore `-lt` a `1`, chiama la funzione `usage`, stampa il messaggio e termina con il codice di errore 10
- `for param in "$@"; do`
  - `$@` rappresenta tutti i parametri passati allo script
  - il `for` scorre ogni parametro (`param`) uno alla volta
- `length=${#param}`
  - `${#param}` → restituisce la lunghezza del parametro (numero di caratteri)
- `if [ $length -gt 20 ]; then`
  se la lunghezza del parametro (`$length`) è maggiore di `20`, stampa un avviso e termina lo script con codice di uscita `1`
- `word_count=$(echo "$param" | wc -w)`
  - usa `echo "$param" | wc -w` per contare il numero di parole all'interno del parametro
    - `wc -l` → conta il numero di righe
    - `wc -w` → conta il numero di parole
    - `wc -c` → conta il numero di byte
    - `wc -m` → conta il numero di caratteri
- `echo "'$param' -> Characters: $length, Words: $word_count"` → stampa il parametro, il numero di caratteri e il numero di parole

Exercise 6: docker-compose

- La directory /exam/exercise6 dovrà contenere i seguenti files e directory:
  - Dockerfile
  - entrypoint.sh
  - docker-compose.yml
  - la directory content
- Il servizio tramite docker compose dovrà gestire l'applicazione hello-exam.
- Dockerfile conterrà le istruzioni per la gestione della vostra applicazione in container il cui servizio dovrà essere avviato tramite script di ENTRYPOINT. Nessun vincolo su immagine di base ecc.…
- entrypoint.sh avrà il compito di scrivere sul file exam.txt il valore della variabile di ambiente EXAM e poi uscire
- docker-compose.yml verrà utilizzato per:
  - gestire start/build della immagine

    - inizializzare la variabile di ambiente EXAM con valore a piacere
    - gestire il bind locale con la directory /exam/exercise6/content che conterrà il file exam.txt popolato dallo script di ENTRYPOINT della applicazione in container
- `mkdir -p /exam/exercise6/content` → creare la directory
- `cd /exam/exercise6` → accesso alla directory
- `vim Dockerfile` → creare il Dockerfile

```
FROM alpine:latest
WORKDIR /app
COPY entrypoint.sh /entrypoint.sh
  RUN chmod +x /entrypoint.sh
  ENTRYPOINT ["/entrypoint.sh"]
```

`from alpine:latest` : specifica l'immagine base per il container Docker, questa immagine è usata come base per costruire il nostro container.
`WORKDIR /app` → imposta `/app` come directory di lavoro all'interno del container, tutti i comandi successivi eseguiti nel Dockerfile verranno eseguiti dentro /app
`COPY entrypoint.sh /entrypoint.sh` → copia il file `entrypoint.sh` (che deve trovarsi nella stessa directory del Dockerfile) all'interno del container, posizionandolo nella root `/` e rinominandolo `entrypoint.sh` . In pratica, mette lo script nel filesystem del container.
`RUN chmod +x /entrypoint.sh` → `RUN` esegue comandi durante la creazione dell'immagine
`ENTRYPOINT ["/entrypoint.sh"]` → imposta il comando predefinito che verrà eseguito quando il container parte.

- crea lo script

```
root@andre:/exam/exercise6# vim entrypoint.sh
#!/bin/bash
echo "\$EXAM" > /content/exam.txt
```

- `chmod +x entrypoint.sh` → esegue script
- creare il file `docker-compose.yml`

```
root@andre:/exam/exercise6# vim docker-compose.yml
version: '3.8'
services:
 hello-exam:
  build: .
  environment:
   - EXAM=HelloExam
  volumes:
   - ./content:/content
```

- `docker compose up - -build -d` → avviare il container con Docker Compose
    - il flag `- -build` forza la ricostruzione dell'immagine
    - il flag `-d` avvia il container in background

Appello_2 16 febbraio 2024

Exercise 1:

- Crea un nuovo utente exercise1
- Imposta una password per l'utente.
- Fai in modo che la password dell'utente scada dopo 6 mesi e che venga cambiata al primo login
- Crea una directory all'interno della home directory del nuovo utente con nome exercise1_dir.
- Imposta i permessi sulla directory exercise1_dir in modo che l'utente exercise1 possa solo leggere e eseguire, mentre il gruppo developers possa anche scrivere.
- Modifica i permessi di default per i nuovi file creati dall'utente exercise1 in modo che siano 640.
- Crea uno script bash che automatizzi tutti i passaggi precedenti chiamato exercise1.sh sotto /exam/exercise1 ad eccezione della creazione password
- Fai in modo che lo script possa essere chiamato tramite Alias command chiamato exercise1_user disponibile a tutti gli utenti del sistema

Svolgimento:

- `useradd exercise1`
- `passwd exercise1` per settare la password
- `chage -M 180 -d 0 exercise1`
  → `-M 180` la password deve essere cambiata ogni sei mesi
  → `-d 0` la password deve essere cambiata al primo login
- `chage -l exercise1` per assicurarsi che le operazioni siano state effettuate con successo
- `mkdir /home/exercise1/exercise1_dir` per creare la directory
- `chmod 750 /home/exercise1/exercise1_dir` consente all'utente di leggere ed eseguire
- `chown exercise1:developers /home/exercise1/exercise1_dir` assegna la proprietà della directory all'utente e al gruppo
- `chmod 640 /home/exercise1`
  `setfacl -d -m u:exercise1:rw /home/exercise1`
  imposta il permesso di lettura e scrittura per i file nuovi creati dall'utente (ad esclusione della directory) a 640
  → `setfacl` comando per gestire le ACL(access control list), si applicano ai nuovi file o directory che verranno creati all'interno della directory.
  → `-m` sta per "modifica" e viene usato per modificare l'ACL esistente
  → `u:exercise1:rw` specifica i permessi da applicare:
  `u:exercise1` indica che stiamo configurando i permessi per l'utente exercise1
  `rw` l'utente avrà permessi in lettura e scrittura sui nuovi file
- `touch /home/exercise1/exercise1.sh`
- `vim /home/exercise1/exercise1.sh`

```
#!/bin/bash

#creo utente
useradd exercise

#creo la directory
mkdir /home/exercise/exercise_dir

#imposto i permessi
```

```
chmod 750 /home/exercise/exercise_dir
chown exercise:developers /home/exercise/exercise_dir

#modifica dei permessi di default
chmod 640 /home/exercise
setfacl -d -m u:exercise1:rw /home/exercise

#scadenza password
chage -M 180 -d 0 exercise

echo "L'utente exercise è stato creato e configurato con successo"
```

- `chmod +x /home/exercise1/exercise1.sh`
  Per settare l'alias:
- `vim .bashrc`
- aggiungere: `alias exercise1_user="bash /home/exercise1/exercise1.sh"`
- `source .bashrc`
- `exercise1_user` per testare

Exercise 2:

- Creare i seguenti file e directory /exam/exercise2/exercise2_dirX/fileY con X compreso tra 1 e 100 e Y compreso tra 10 e 20
- Utilizzate il comando “find /exam/exercise2” tramite redirezione dello standard output creare il file /exam/exercise2/regex/find.txt
- Riportare le regular expression in /exam/exercise2/regex/regex.txt da utilizzare all’interno del file /exam/exercise2/regex/find.txt in modo che riportino:
  - le entry che contengono il pattern 12 (uno seguito da due)
  - le entry che finiscano con il numero 9

Svolgimento:

- `mkdir exam`
- `mkdir exam/exercise2`
- `mkdir exam/exercise2/exercise2_dir{1..100}`
- `mkdir exam/exercise2/exercise2_dir{1..100}/file{10..20}`
- `find exam/exercise2 > exam/exercise2/regex/find.txt` il comando find trova tutte le directory e i file sotto il path indicato, l'output viene poi rediretto nel file find.txt
- `echo ".*12.*" > exam/exercise2/regex/regex.txt`
- `echo ".*9$" >> exam/exercise2/regex/regex.txt`
- `grep -f exam/exercise2/regex/regex.txt exam/exercise2/regex/find.txt`

Exercise 3:

- Creare un nuovo comando sotto /exam/exercise3 chiamato gstat.
- gstat dovrà eseguire il seguente comando: ps-eo pid,tid,class,rtprio,ni,pri,psr,pcpu,stat,wchan:14,comm.
- Il proprietario di gstat sarà l’utente root
- Il file dovrà avere i permessi di lettura scrittura ed esecuzione così impostati: rwxr-xr-x

- Fare in modo che chiunque lanci il comando gstat, lo faccia con le grant dell'utente root e non con quelle dell'utente che lancia il comando.

Svolgimento:

- `mkdir -p /exam/exercise3`
- `touch /exam/exercise3/gstat`
- `chmod 755 /exam/exercise3/gstat`
  il proprietario (root) ha permessi di lettura, scrittura ed esecuzione
  il gruppo e gli altri utenti hanno permessi di lettura ed esecuzione
- `chown root:root /exam/exercise3/gstat`
- `chmod u+s /exam/exercise3/gstat` il bit `setuid` permette di eseguire il file con i permessi dell'utente proprietario

Exercise 4:

- Creare uno script bash sotto /exam/exercise4 chiamato sysinfo.sh ● Il file /proc/cpuinfo in Linux contiene informazioni dettagliate su tutti i core CPU del sistema. Ecco alcuni dei dati che puoi trovare:
  - processor: Numero identificativo del processore.
  - vendor_id: Identificatore del fornitore del processore.
  - cpufamily: Famiglia del processore.
  - model:Modelo del processore.
  - etc…
- Il file /proc/meminfo in Linux fornisce informazioni dettagliate sull'utilizzo della memoria del sistema. Ecco alcuni dei dati che puoi trovare:
  - MemTotal: Quantità totale di memoria RAM installata nel sistema.
  - MemFree:Quantità di memoria RAM libera disponibile.
  - MemAvailable: Quantità di memoria RAM disponibile per l'allocazione.
  - etc…
- lo script sysinfo.sh accetterà in ingresso il tipo di statistica da riportare come primo parametro
  - cpu(per /proc/cpuinfo)
  - mem(per/proc/meminfo)
- il secondo argomento è opzionale, se non viene definito verranno visualizzare tutte le informazioni contenute sotto /proc/cpuinfo o /proc/meminfo a seconda che il primo parametro sia cpu oppure mem Se viene fornito sarà una lista di informazioni contenuta in una unica stringa e come separatore potete usare quello che volete. esempio: bash sysinfo.sh cpu "param1 param2 param3" bash sysinfo.sh cpu "param1|param2|param3" bash sysinfo.sh cpu "param1,param2,param3" bash sysinfo.sh cpu "param1:param2:param3"
- prevedete dei controlli che verifichino il primo parametro sia presente e uguale a cpu oppure mem, altrimenti lo script deve uscire con una stringa di istruzioni su parametri e formati accettati per la lista di parametri
  Svolgimento:
- `mkdir /exam/exercise4`
- `touch /exam/exercise4/sysinfo.sh`
- `vim /exam/exercise4/sysinfo.sh`

```
#!/bin/bash
usage() {
echo "Uso: $0 <cpu|mem> [parametri]"                                                    echo "
<cpu> : per visualizzare le informazioni sulla CPU (da /proc/cpuinfo)"
        echo " <mem> : per visualizzare le informazioni sulla memoria (da
/proc/meminfo)"
        exit 1
}

#verifico che il primo parametro sia stato passato
if [ -z "$1" ]; then
        echo "errore: fornire il tipo di statistica"
        usage
fi

#determino se il primo parametro è cpu o mem
stat_type=$1
shift #rimuove il primo parametro

#controllo la validità del primo parametro
if [[ "$stat_type" != "cpu" && "$stat_type" != "mem" ]]; then
        echo "errore: parametro invalido"
        usage
fi

#funzione per estrarre le info della cpu
get_cpu_info() {
        if [ -z "$1" ]; then
                cat /proc/cpuinfo
        else
                for param in $1; do
                        grep -i "^$param" /proc/cpuinfo #per ogni parametro, grep
cerca (insensitive-case) le righe che iniziano con $param
                done
        fi
}

#funzione per estrarre le info della mem
get_mem_info(){
        if [ -z "$1" ]; then
                cat /proc/meminfo
        else
                for param in $1; do
                        grep -i "^$param" /proc/meminfo
                done
        fi
}

#eseguo il comando appropriato in base al tipo di statistica
if [ "$stat_type" == "cpu" ]; then
        if [ -z "$1" ]; then
                get_cpu_info
        else
                param_list=$(echo "$1" | tr ',:|' ' ')
```

```
                get_cpu_info "$param_list"
        fi
elif [ "$stat_type" == "mem" ]; then
        if [ -z "$1" ]; then
                get_mem_info
        else
                param_list=$(echo "$1" | tr ':,|' ' ')
                get_mem_info "$param_list"
        fi
fi
```

- `chmod +x /exam/exercise4/sysinfo.sh`
  per assicurarci che funzioni:
- `/exam/exercise4/sysinfo.sh` → stamperà il messaggio di errore e le istruzioni sull'utilizzo
- `/exam/exercise4/sysinfo.sh cpu`
- `/exam/exercise4/sysinfo.sh cpu "processor|vendor_id"`

Exercise 5:

- La directory da utilizzare è /exam/exercise5
- Il servizio tramite docker-compose dovrà gestire l'applicazione hello-exam-httpd.
- Obbligatorio gestire al minimo:
  - Dockerfile dovrà utilizzare ENTRYPOINT per lo start del servizio. Nessun vincolo sulla immagine di base.
  - entrypoint.sh avrà il compito di lanciare il servizio HTTPD
  - docker-compose.yml verrà utilizzato per:
    - gestire start/build della immagine
    - inizializzare la variabile di ambiente HELLO con valore a piacere
    - gestire eventuali bind locali qualora fosse necessario
- NB: La gestione del contenuto da visualizare tramite index.html da posizionare nella document root del servizio è a piacere.
  - creato direttamente da entrypoint.sh
  - bind locale con un file template
  - copiato in fase di build della immagine
  - posizionato direttamente nel Dockerfile
  - L'importante è che se ne possa modificare il contenuto utilizzando la variabile di ambiente $HELLO
- codice HTML da utilizzare:

```
<!DOCTYPE html>
<html>
<head>
<title>Exam 2</title>
</head>
<body>
<h1><contenuto della variabile HELLO></h1>
</body>
</html>
```

Svolgimento:

- `mkdir /exam/exercise5`
- `vim /exam/exercise5/Dockerfile`

```
FROM alpine:latest
COPY entrypoint.sh /entrypoint.sh
RUN chmod +x /entrypoint.sh
ENTRYPOINT ["/entrypoint.sh"]
```

- `vim /exam/exercise5/entrypoint.sh`

```
#!/bin/sh

# Sostituisce il contenuto di index.html con la variabile HELLO
echo "<!DOCTYPE html>
<html>
<head>
<title>Exam 2</title>
</head>
<body>
<h1>${HELLO}</h1>
</body>
</html>" > /usr/local/apache2/htdocs/index.html

# Avvia il server HTTPD
httpd-foreground
```

Exercise 2:

- dato il file test.txt che troverete sotto /exam/exercise2/
- Riportare sotto /exam/exercise2/answer.txt le regular expression che riportino:
  - le entry che contengono il pattern ao
  - le entry che finiscano con un numero
  - le entry che iniziano e finiscono con una lettera maiuscola

```
grep -E "ao" /exam/exercise2/test.txt >> /exam/exercise2/answer.txt
grep -E ".*[0-9]$" /exam/exercise2/test.txt >> /exam/exercise2/answer.txt
grep -E "^[A-Z].*[A-Z]$" /exam/exercise2/test.txt >> /exam/exercise2/answer.txt
```

Exercise 3:

- Creare un nuovo alias command chiamato exercise3.
- exercise3 dovrà riportare un output contenente il numero di utenti connessi al sistema e la data corrente
- i singoli comandi che potete utilizzare possono essere (ma potete usare quello che volete)
  - who|wc-l
  - date

  Svolgimento:
- `vim .bashrc`

- `alias exercise3="(who | wc -l) && date"`
- `source .bashrc`

Exercise 4:

- Preparate uno script Bash sotto /exam/exercise4 chiamato configure_virtualhost.sh che configuri un nuovo virtual host su un server HTTP/Apache. L'utente fornirà il nome del virtual host e il contenuto della pagina index.html. L'esercizio comprende l'installazione del servizio HTTP/Apache se non è già presente.
- Lo script deve prendere in input il nome del virtual host e il contenuto della pagina index.html.
- Se Apache non è già installato, lo script deve installare il servizio.
- Configura il virtual host per il nuovo nome di dominio.
- Assicurati che l'indirizzo del nuovo nome di dominio risolva sull'IP privato del sistema modificando il file /etc/hosts NB. il file /etc/hosts non va sovrascritto per cui assicuratevi di non usare redirezione tramite il singolo '>' ma utilizzare la modalità append '>>'. Per sicurezza create tramite script una copia di backup del file /etc/hosts in /etc/BK-hosts
- Crea la pagina index.html nel Document Root del nuovo virtual host con il contenuto fornito.
  - configurate la DocumentRoot a piacere ma a partire dal path /exam/exercise4/vhosts_document_root/
- Riavvia il servizio Apache per applicare le modifiche.
- Prevedete i controlli sul numero di parametri inseriti e un output che spieghi cosa lo script si aspetta nel caso non vengano forniti o forniti in modo errato.
  Svolgimento:
- `mkdir /exam/exercise4`
- `touch /exam/exercise4/configure_virtualhost.sh`
- `vim /exam/exercise4/configure_virtualhost.sh`

```
#!/bin/bash

usage() {
    echo "Inserire come primo parametro il nome del virtual host"
    echo "Inserire come secondo parametro il contenuto dell'index.html"
    exit 1
}

# Controllo dei parametri
if [ -z "$1" ] || [ -z "$2" ]; then
    echo "Uno o più parametri sono vuoti"
    usage
fi

NOME=$1
CONTENUTO=$2

# Verifica se Apache è installato, altrimenti lo installa
if ! dnf list installed | grep -q httpd; then
    echo "Apache non è installato. Procedo con l'installazione..."
    sudo dnf install -y httpd
fi
```

```
# Creazione della directory per il virtual host
DIRECTORY="/srv/www.$NOME.com"
mkdir -p "$DIRECTORY"

# Creazione del file index.html con il contenuto fornito
touch "$DIRECTORY"/index.html
echo "$CONTENUTO" >> "$DIRECTORY/index.html"

# Backup del file /etc/hosts
cp /etc/hosts /etc/hosts.backup

# Aggiunta dell'host al file /etc/hosts
echo "127.0.0.1 www.$NOME.com" >> /etc/hosts

# Creazione della configurazione per il virtual host
VHOST_CONF="/etc/httpd/conf.d/$NOME-vhost.conf"
cat > "$VHOST_CONF" <<EOF
<VirtualHost *:80>
    ServerName www.$NOME.com
    DocumentRoot $DIRECTORY
    CustomLog /var/log/httpd/www.$NOME.com.log combined
</VirtualHost>

<Directory $DIRECTORY>
    Require all granted
</Directory>
EOF

# Riavvio di Apache per applicare la configurazione
sudo systemctl restart httpd

# Disabilita SELinux temporaneamente (per testing, sarebbe meglio configurarlo
correttamente)
sudo setenforce 0

echo "Virtual host www.$NOME.com configurato con successo!"
```

Exercise 5:

- La directory /exam/exercise5 dovrà contenere i seguenti files e directory:
  - Dockerfile
  - entrypoint.sh
  - docker-compose.yml
  - la directory copy
- Il servizio tramite docker-compose dovrà gestire la vostra applicazione.
- Dockerfile conterrà le istruzioni per la creazione della vostra immagine il cui servizio applicativo dovrà essere avviato tramite script di ENTRYPOINT.
- entrypoint.sh avrà il compito di copiare il contenuto della directory passata come variabile di ambiente (chiamatela DIRECTORY) sotto la directory copy locale al sistema linux
- docker-compose.yml verrà utilizzato per:
  - gestire start/build della immagine
  - inizializzare la variabile di ambiente DIRECTORY con valore a piacere

- gestire il bind locale con la directory /exam/exercise6/copy e il vostro container
Svolgimento:
Dockerfile

```
FROM alpine:latest
WORKDIR /app
COPY entrypoint.sh /app/entrypoint.sh
RUN chmod +x /app/entrypoint.sh
ENTRYPOINT ["/app/entrypoint.sh"]
```

entrypoint.sh

```
#!/bin/bash

DIRECTORY=$1

cp -r "$DIRECTORY" /app/copy/

echo "contenuto della directory copy"
ls -l /app/copy/
```

docker-compose.yml

```
version: '3.8'
services:
  exercise5:
    build: .
    environment:
      - DIRECTORY=/exercise5
    volumes:
      - ./copy:/app/copy
```

Appello_4

Exercise 1: Command line and shell environment

- Come utente exercise1
- Creare uno script bash che si chiamerà exam.
- Posizionare il file bash “exam”, in una directory all’interno della home directory dell’utente exercise1 (dopo averlo creato), con i giusti permessi di accesso.
- Assicuratevi possa essere lanciato da riga di comando senza specificare il path assoluto, configurando la variabile PATH del solo utente exercise1 in modo appropriato.
- Lo script andrà a creare un file nel path /home/exercise1 chiamato date.txt contenente la data di lancio dello script exam. Ad ogni esecuzione verrà sovrascritto il contenuto precedente in date.txt
Svolgimento:
- `mkdir mydir`
- `touch mydir/exam.sh`
- `vim mydir/exam.sh`

```
#!/bin/bash

touch /home/exercise1/date.txt
echo $(date) > /home/exercise1/date.txt
cat /home/exercise1/date.txt
```

in root:

- `chmod +x /home/exercise1/mydir/exam.sh`
- `vim .bashrc`
- inserire: `export PATH="$PATH:/home/exercise1/mydir"`
- `source .bashrc`

Exercise 2: User and Group

- Create il gruppo students con GID 3500
- Creare l'utente appartenente al gruppo students: student
  - L'utente student avrà le seguenti caratteristiche:
  - UID 3010
  - l'account scadrà a sei mesi dalla sua creazione
  - dovrà avere come gruppo secondario: exam4
  - Modificare una volta creato l'utente student umask (in modo permanente) così che tutti i file e directory creati dall'utente non abbiano nessun permesso OTHER settato.
  Svolgimento:
- `groupadd -g 3500 students`
- `useradd -u 3010 student`
- `usermod -aG students student`
- `chage -E 2025-09-07 student`
- `usermod -G exam4 student`
  per modificare l'umask:
- `su - student`
- `vim .bashrc`
- inserire: `umask 0777`
- `source .bashrc`

Exercise 3: file permission

- Creare una directory sotto /exam/exercise4 dove gli utenti che possono accedere al gruppo exam potranno condividere files
- Tutti i file creati sotto la directory /exam/exercise4 dovranno essere assegnati automaticamente al gruppo exam
  Svolgimento:
- `mkdir /exam/exercise4`
- `chown :exam dir4`
- `chmod 770 dir4`
- `chmod g+s dir4`

Exercise 5: Bash script

- Creare uno script bash sotto /exam/exercise5 chiamato filesystem_stat.sh che accetti in ingresso due parametri
  - il primo conterrà il path assoluto della directory che vogliamo analizzare
  - il secondo l'informazione che vogliamo ottenere
    - list_all: in output verrà visualizzata la lista dei file e directory presenti nel path passato (comando da utilizzare ls con le dovute opzioni)
    - list_directory: la lista delle sole directory presenti nel path passato (comando da utilizzare ls con le dovute opzioni)
    - tree: la rappresentazione ad albero della struttura delle directory (comando da utilizzare tree)
  - Si prevedano i seguenti controlli
    - Se non vengono passati i parametri attesi lo script deve restituire un errore e uscire con exit code 10
    - Se il parametro 1 il path assoluto non è una directory esca con output una stringa di errore a piacere.
    - Se il parametro 2 non è una delle opzioni attese un generico USAGE per lo script

```
#!/bin/bash

usage(){
        echo "Errore! I comandi disponibili sono: "
        echo "list_all"
        echo "list_directory"
        echo "tree"
        exit 10
}

if [ "$#" -ne 2 ]; then
        echo "Errore! Numero di parametri errato"
        usage
fi

DIRECTORY=$1
FUN=$2

if [ ! -d "$DIRECTORY" ]; then
        echo "Errore! '$DIRECTORY' non è valida"
        exit 1
fi

list_all() {
        ls -l "$DIRECTORY"
}

list_directory() {
        ls -R "$DIRECTORY"
}

tree_func() {
        tree "$DIRECTORY"
}
```

```
if [ "$FUN" == "list_all" ]; then
        list_all
elif [ "$FUN" == "list_directory" ]; then
        list_directory
elif [ "$FUN" == "tree" ]; then
        tree_func
else
        usage
fi
```

Exercise 6: docker-compose
Attenzione: da command line utilizzare docker-compose al posto di "docker compose"
La directory /exam/exercise6/advanced logging dovrà contenere i seguenti file e directory:

- Dockerfile
- entrypoint.sh
- docker-compose.yml
- la directory logs
  Il servizio tramite Docker Compose dovrà gestire l'applicazione log-manager.
- Dockerfile: conterrà le istruzioni per la gestione della vostra applicazione in container, il cui servizio dovrà essere avviato tramite uno script di ENTRYPOINT. L'immagine di base può essere scelta liberamente.
- entrypoint.sh: avrà il compito di creare un file log.txt nella directory /logs all'interno del container e scrivere al suo interno un messaggio di log contenente la data e l'ora attuali, e il valore della variabile di ambiente LOG_MESSAGE. Una volta scritto il file, lo script continuerà a scrivere un messaggio di log ogni 10 secondi fino a quando il container non verrà fermato.
- docker-compose.yml: verrà utilizzato per:
  - gestire la costruzione e l'avvio dell'immagine del container
  - inizializzare la variabile di ambiente LOG_MESSAGE con un valore a vostra scelta
  - gestire il bind locale con la directory /exam/exercise6/advanced_logging/logs che conterrà il file log.txt popolato dallo script di ENTRYPOINT dell'applicazione in container
  Dockerfile

```
FROM alpine:latest
COPY entrypoint.sh /entrypoint.sh
RUN chmod +x /entrypoint.sh
ENTRYPOINT ["/entrypoint.sh"]
```

entrypoint.sh

```
#!/bin/sh

LOG_FILE="/logs/log.txt"
touch "$LOG_FILE"

while true; do
```

```
        echo "$(date) - $LOG_MESSAGE" >> $LOG_FILE
        sleep 10
done
```

docke-compose.yml

```
version: '3.8'
services:
  log-manager:
    build: .
    container_name: log_manager
    environment:
      - LOG_MESSAGE="Messaggio di log predefinito"
    volumes:
      - /exam/exercise6_2/logs:/logs:rw
```

Appello_5

Exercise 1: Command line and shell environment

- Creare uno script bash che si chiamerà ex1. Posizionare il file bash ex1 in una directory chiamata scripts all'interno della home directory dell'utente exercise1.
- Impostare i permessi del file ex1 in modo che solo l'utente exercise1 possa leggere, scrivere ed eseguire il file.
- Configurare la variabile PATH dell'utente exercise1 in modo che lo script possa essere lanciato da riga di comando senza specificare il path assoluto.
- Lo script dovrà:
  - All'interno della directory home di exercise1, creare un file chiamato ex1_file contenente la data di lancio dello script ex1 (si utilizzi il comando date senza nessuna formattazione).
  Svolgimento:
- `mkdir /home/exercise1/scripts`
- `cd /home/exercise1/scripts`
- `touch ex1.sh`
- `vim ex1.sh`

```
#!/bin/bash

DIRECTORY="/home/exercise1/scripts"
touch "$DIRECTORY"/ex1_file

date > "$DIRECTORY"/ex1_file
```

- `chmod 700 /home/exercise1/scripts/ex1.sh`
  come utente `exercise1` :
- `vim /home/exercise1/.bashrc`
- aggiungere alla fine: `export PATH=$PATH:/home/exercise1/scripts`
- `source /home/exercise1/.bashrc`

Exercise 2: User and Group

- Create il gruppo students con GID 3500
- Creare l'utente appartenente al gruppo students: student
  - L'utente student avrà le seguenti caratteristiche:
  - UID3010
  - dovrà avere come gruppo secondario: exam5
  - Modificare una volta creato l'utente student umask (in modo permanente) così che tutti i file e directory creati dall'utente abbiano massimi permessi per tutti.
  Svolgimento:
- `groupadd -g 3500 students`
- `useradd -u 3010 -g students -G exam5 student`
  per modificare l'umask:
- `vim /home/student/.bashrc`
- aggiungere alla fine: `umask 0000`
- `source /home/student/.bashrc`

Exercise 3: HTTPD

- Installare sul sistema il servizio HTTP/Apache (yum install-y httpd)
- Questo server web dovrà visualizzare il contenuto "Hello exercise3" quando viene richiesto l'URL http://www.exercise3.myexam.com
- NB:Siutilizzi un file di configurazione dedicato chiamato /etc/httpd/conf.d/exercise3.conf
- Potete verificare che il tutto funzioni tramite comando curl
  Svolgimento:
- `yum install -y httpd`
  Configurare Apache per visualizzare "Hello exercise3"
- `vim /etc/httpd/conf.d/exercise3.conf`

```
<VirtualHost *:80>
    ServerName www.exercise3.myexam.com
    DocumentRoot /var/www/html/exercise3

    <Directory /var/www/html/exercise3>
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog /var/log/httpd/exercise3_error.log
    CustomLog /var/log/httpd/exercise3_access.log combined
</VirtualHost>
```

- **server name**: impostato su `www.exercise3.myexam.com`
- **document root**: la directory in cui verrà messo il file html che mostrerà "Hello exercise3"
- **ErrorLog e CustomLog**: percorsi per i log di errore e accesso a specifici per questo sito
  Creare la pagina HTML
- `mkdir -p /var/www/html/exercise3`
- `vim /var/www/html/exercise3/index.html`

```
<!DOCTYPE html>
<html>
<head>
    <title>Exercise 3</title>
</head>
<body>
    <h1>Hello exercise3</h1>
</body>
</html>
```

Abilitare e avviare il servizio Apache

- `systemctl start httpd`
- `systemctl enable httpd`
  Configurare il DNS
- `vim /etc/hosts`
- aggiungere una riga per l'host `127.0.0.1 www.exercise3.myexam.com`
  Verifica
- `curl http://www.exercise3.myexam.com`

Exercise 4: Bash script

- Creare uno script bash sotto /exam/exercise4 chiamato dir_analyze.sh che accetti in ingresso due parametri:
  - Il primo parametro conterrà il path assoluto della directory che vogliamo analizzare.
  - Il secondo parametro indicherà l'azione da eseguire, scegliendo tra le seguenti opzioni:
    - count_files: visualizza il numero totale di file presenti nella directory
    - count_dirs: visualizza il numero totale di directory presenti nella directory
    Si prevedano i seguenti controlli:
- Se non vengono passati i parametri attesi, lo script deve restituire un errore e uscire con exit code 10.
- Se il primo parametro (il path assoluto) non è una directory, lo script deve visualizzare un messaggio di errore e uscire con exit code 20.
- Se il secondo parametro non è una delle opzioni attese (count_files o count_dirs), lo script deve visualizzare un messaggio di utilizzo generico e uscire con exit code 30.
  Svolgimento:

```
#!/bin/bash

usage(){
        echo "Scegliere tra le opzioni: "
        echo " - count_files"
        echo " - count_dirs"
        exit 30
}

if [ $# -ne 2 ]; then
        echo "errore"
        exit 10
fi
```

```
DIRECTORY=$1
FUN=$2

count_files() {
        num_files=$(find "$DIRECTORY" -maxdepth 1 -type f | wc -l)
}

count_dirs(){
        num_dirs=$(find "$DIRECTORY" -maxdepth 1 -type d | wc -l)
        num_dirs=$((num_dirs - 1))
}

if [ ! -d "$DIRECTORY" ]; then
        echo "Errore"
        exit 20
elif [ "$FUN" == "count_files" ]; then
        count_files
        echo "$num_files"
elif [ "$FUN" == "count_dirs" ]; then
        count_dirs
        echo "$num_dirs"
else
        usage
fi
```

Exercise 5: Docker compose

- Creare la propria immagine a partire da un'immagine di base a piacere
- La directory /exam/exercise5 dovrà contenere:
  - Dockerfile
  - logger.sh
  - docker-compose.yml
  - loggerdir (questa è una directory)
- logger.sh sarà lo script di entrypoint il cui compito sarà quello di stampare all'interno di un file chiamato logger.log la stringa "hello docker" NUMBER volte e poi uscire
- NUMBER sarà una variabile di ambiente contenente un numero di default a piacere.
- docker-compose.yml lo utilizzerete per:
  - gestire un bind locale con la directory loggerdir e il container, che conterrà il file logger.log
  - la build della immagine

  Svolgimento:

  Dockerfile

```
FROM alpine:latest
COPY logger.sh /logger.sh
RUN chmod +x /logger.sh
ENTRYPOINT ["/logger.sh"]
```

logger.sh

```sh
#!/bin/sh

LOG_FILE="/loggerdir/logger.log"
: ${NUMBER:=10}  # Valore di default se non specificato

mkdir -p /loggerdir

for i in $(seq 1 $NUMBER); do
    echo "hello docker" >> "$LOG_FILE"
done
```

docker-compose.yml

```yaml
version: '3.8'

services:
  logger:
    build: .
    container_name: logger_container
    environment:
      - NUMBER=5  # Numero di volte che verrà stampato "hello docker"
    volumes:
      - ./loggerdir:/loggerdir
```

- `docker compose up --build`
- `cat loggerdir/logger.log`
- `docker compose down`