ORVIETOLINUX APS
PRESENTA:

SERATA TEMATICA PER I SOCI!

Introduzione al

BASH

THE BOURNE-AGAIN SHELL

Scripting

INDICE DEGLI ARGOMENTI

- DEFINIZIONI
- STRUTTURA DI UNO SCRIPT IN BASH
- SHEBANG
- COMMENTI E VARIABILI
- FUNZIONI, CORPO E CHIUSURA DEGLI SCRIPT
- EDITOR NATIVI E ONLINE
- ESERCIZI PRATICI

PARTIAMO DALLE DEFINIZIONI

B.A.SH. è un acronimo che
significa **B**ourne **A**gain **SH**ell

Si tratta della **Shell** predefinita di GNU Linux.

Ma di cosa parliamo?

Una Shell è uno dei numerosi programmi che fanno parte del sistema GNU ed ha la caratteristica di poter **ricevere comandi testuali** dall'utente tramite un interfaccia, di **interpretarli** e di **farli eseguire** al sistema operativo

PARTIAMO DALLE DEFINIZIONI

Perchè il nome “Shell”?

Shell in inglese indica una “conchiglia” e di fatto si tratta di una interfaccia esterna (proprio come un guscio); è stata introdotta perchè un utente umano non può interagire direttamente con il kernel Linux e “parlare” il linguaggio macchina, pertanto la shell svolge il ruolo di interfaccia tra un utente e il kernel.

In pratica, io utente, digito un comando testuale, la shell interpreta il comando e lo fornisce al kernel che lo elabora a livello di sistema.

PARTIAMO DALLE DEFINIZIONI

Perchè Bourne “Again” Shell?

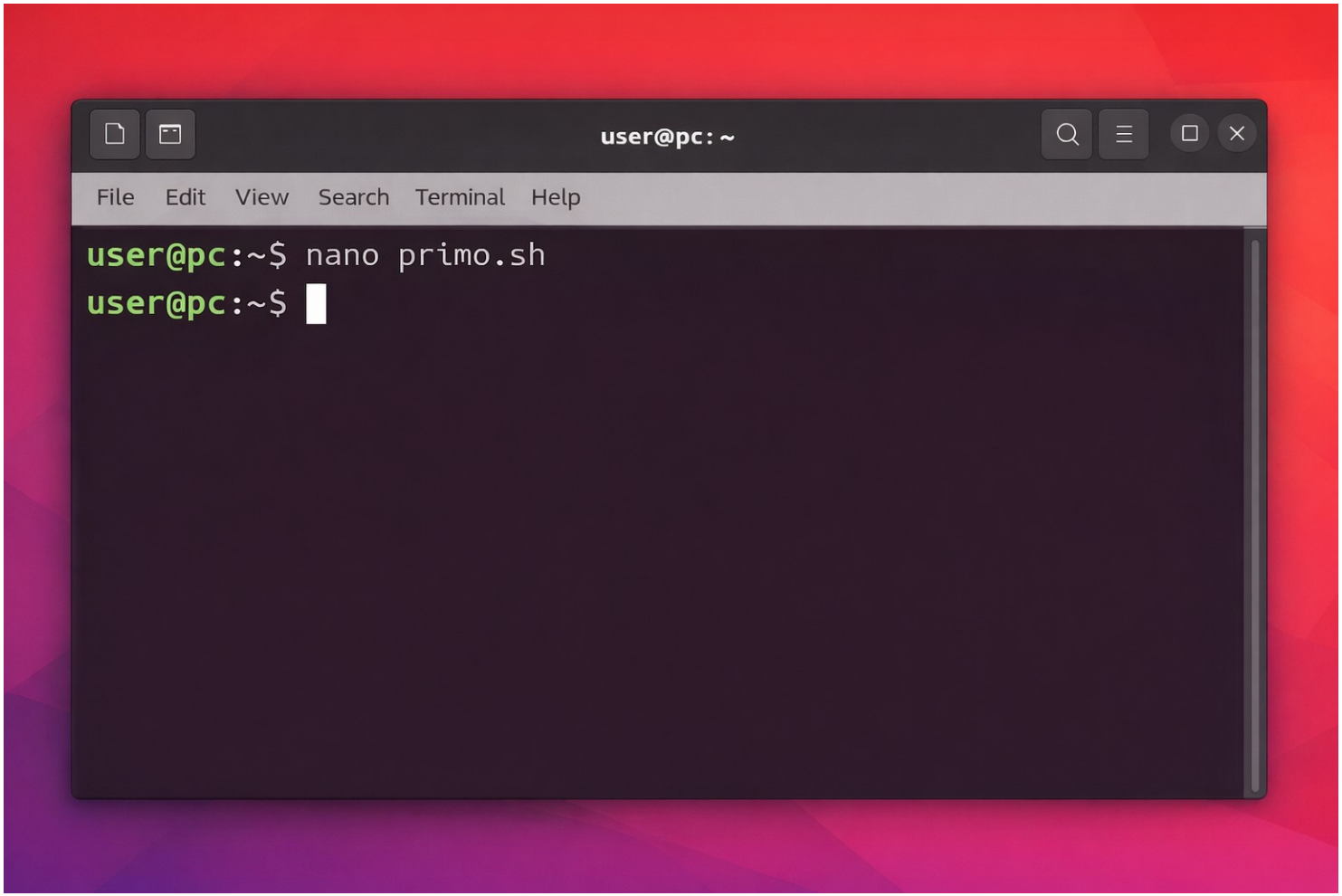
Perchè si tratta di una “rinascita”, anche filosofica!
Nei sistemi UNIX anni ‘70 la shell proprietaria di sistema si chiamava Bourne Shell (nota anche come sh); questo invece è un free software incluso da Stallman nei sistemi GNU!

Lavoriamo direttamente nella Shell quindi? “NI”

Quando apriamo quello che chiamiamo il “terminale” o la “riga di comando” in realtà apriamo un **emulatore** nell’**ambiente grafico** della SHELL; questo - all’interno di una finestra - ci permette di immettere comandi sotto forma di testo.

UN ESEMPIO DI TERMINALE EMULATO

BASH
THE BOURNE-AGAIN SHELL



DEFINIZIONI

Cosa si intende quindi con Bash Scripting?

L'argomento principale di questo incontro è un linguaggio di programmazione interpretato da BASH, tramite cui si realizzano script (file di testo che contengono delle istruzioni e procedure, resi poi eseguibili).

Creeremo quindi piccoli programmi basati sui comandi normalmente lanciati manualmente da SHELL.

Ad esempio, posso creare un programma per **stampare a video un saluto**, dopo che l'utente avrà fornito il proprio nome: "Ciao Marco!" , uno script per **scaricare un archivio zip** da un indirizzo sul web e lo decomprima nella mia cartella oppure uno che faccia **pulizia del sistema**.

ESEMPIO DI SCRIPT IN BASH

BASH
THE BOURNE-AGAIN SHELL

```bash
#!/usr/bin/env bash

# Legge da tastiera il nome da usare nel saluto.
# read: built-in della shell che acquisisce input standard e lo salva in una variabile.
# Opzioni usate:
# -r   non interpreta il backslash (\) come carattere di escape.
# -p   mostra il prompt prima di leggere l'input.
# (none) senza opzioni, read leggerebbe comunque l'input ma senza prompt automatico
#        e con interpretazione dei backslash.
read -r -p "Inserisci il nome della persona: " nome

# Verifica che l'utente abbia inserito almeno un valore.
# [[ -z ... ]]: condizione vera se la stringa è vuota (zero caratteri).
if [[ -z "${nome}" ]]; then
  # Stampa un messaggio di errore e termina lo script con codice 1.
  echo "Errore: non hai inserito alcun nome."
  exit 1
fi

# Stampa il saluto finale.
# echo: built-in che scrive testo su output standard.
# Opzioni usate:
# (none) qui non servono opzioni; stampiamo semplicemente la stringa.
echo "Ciao, ${nome}!"
```

STRUTTURA DI UNO SCRIPT CON ESEMPIO PRATICO

BASH
THE BOURNE-AGAIN SHELL

```
#!/usr/bin/env bash

# 1. Variabili globali
NOME="Marco"

# 2. Funzioni
saluta() {
    echo "Ciao $1"
}

# 3. Corpo principale dello script
echo "Avvio script..."
saluta "$NOME"

# 4. Uscita
exit 0
```

ANALIZZIAMO LA STRUTTURA

Uno script Bash è generalmente composto da:

Shebang

Variabili

Funzioni

Corpo principale

Exit code

Nota bene: non è obbligatorio avere tutto questo!

SHEBANG

```
#!/usr/bin/env bash
```

Uno shebang / hashbang è la sequenza **#!**

 # → sharp o hash

! → bang

sh + bang = shebang

Indica al sistema operativo quale interprete usare

Qui, dice al sistema: “Esegui questo file con Bash”

VARIABILI E COMMENTI

```
# 1. Variabili globali
```

Questa parte col # anteposto definisce i commenti, posso ad esempio inserire delle spiegazioni sul codice (i programmatori cani non lo fanno)

```
NOME="Marco"
```

Questa parte definisce invece le variabili a livello globale nello script, significa che in questo caso specifico richiamando la variabile NOME recupererò quel valore (Marco)

VARIABILI E COMMENTI

Se impostassi lo script con queste due righe in sequenza:

```
NOME="Marco"
NOME="Leonardo"
```

e successivamente richiamassi la variabile globale NOME avrei come risultato il secondo (Leonardo) perchè ha sovrascritto il primo valore (Marco)

FUNZIONI

```
saluta() {
echo "Ciao $1"
}
```

Le funzioni sono blocchi di codice che servono a ottimizzare le istruzioni ed a non ripetere ogni volta le istruzioni per esteso nel file degli script!

```
saluta () #nome della funzione
{ } #contengono le operazioni svolte
$1 #è il primo parametro inviato alla funzione
echo #stampa a video un testo
```

RISULTATO A VIDEO: Ciao Marco

CORPO DELLO SCRIPT

```
echo "Avvio script…"
saluta "$NOME"
```

Nel corpo c’è lo svolgimento del nostro script
In questo caso viene prima mostrato a video il messaggio:
”Avvio script…”
e successivamente viene richiamata la funzione che permette di salutare il nome impostato nello script

RISULTATO A VIDEO:

```
Avvio script…
Ciao Marco
```

CHIUSURA DELLO SCRIPT

```
# 4. Uscita

exit 0
```

Se tutto va a buon fine lo script si chiude senza errori

PERMESSI ED ESECUZIONE

```
#nomenclatura, salviamo
#lo script con estensione .sh
nome_script.sh

#rendere eseguibile uno script
chmod +x nome_script.sh

#eseguire uno script BASH
./nome_script.sh
```

EDITOR NATIVI SU LINUX

nano

abhishek@LHB: ~

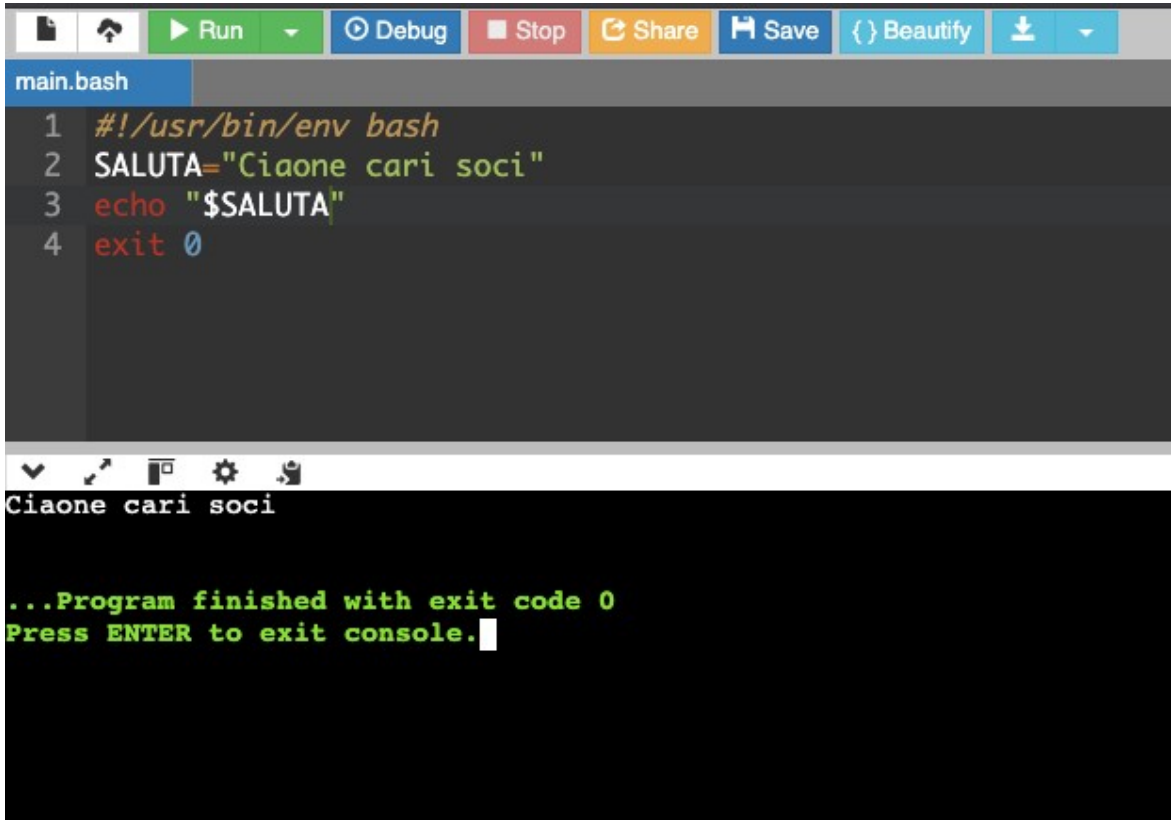
```
  GNU nano 7.2                    test.sh
# Define variables
name="John"
age="thirty"

# Display a greeting
echo "Hello, my name is $name"

# Perform conditional checks
if [ $age -le 18 ]; then
    echo "I'm not an adult yet"
else
    echo "I'm an adult"
fi




                [ Read 13 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut       ^T Execute
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify
```

EDITOR NATIVI SU LINUX

vim

EDITOR ONLINE

https://www.onlinegdb.com/online_bash_shell



ESERCIZIO #1

Creare uno script BASH che, una volta avviato, mostri a video un messaggio in cui descrivi come si sta svolgendo la serata finora, con un testo di massimo 3-4 righe

ESERCIZIO #1 – SOLUZIONE A


```
#!/usr/bin/env bash
#soluzione banale

echo "Questa sera abbiamo partecipato all'assemblea dei soci di Orvieto Linux e dopo abbiamo iniziato la prima serata tematica del 2026 sul BASH scripting"

exit 0
```

ESERCIZIO #1 – SOLUZIONE B


```
#!/usr/bin/env bash
#soluzione intermedia

DESCRIZIONE="Questa sera abbiamo partecipato all'assemblea dei soci di Orvieto Linux e dopo abbiamo iniziato la prima serata tematica del 2026 sul BASH scripting"

echo "$DESCRIZIONE"

exit 0
```

ESERCIZIO #1 – SOLUZIONE C


```
#!/usr/bin/env bash
#soluzione con complessità superiore

DESCRIZIONE="Questa sera abbiamo
partecipato all'assemblea dei soci di
Orvieto Linux e dopo abbiamo iniziato la
prima serata tematica del 2026 sul BASH
scripting"

mostra () { echo "$1" }
mostra $DESCRIZIONE

exit 0
```

SCRIPT CHE RICEVONO INFORMAZIONI DALL'UTENTE

Finora abbiamo visto degli script che consentono di mostrare a video alcuni valori già preimpostati all'interno del codice tramite variabili, come il nome del primo esempio o la descrizione dell'esercizio.

Con gli script BASH è possibile anche creare programmi che ricevano in ingresso (INPUT) una informazione da parte dell'utente e che eseguano operazioni con il dato che forniamo.

Ad esempio, creo un programma che mi permetta di presentarmi con il mio nome e che, una volta ricevuta questa informazione mi saluti nominandomi.

IL COMANDO “READ”

Per questo scopo, cioè fornire dati allo script BASH, ci viene in aiuto un comando chiamato read, la cui sintassi è la seguente:

```
read nomevariabile
```

Che nei casi precedenti poteva seguire una sintassi come le seguenti:

```
read NOME
read DESCRIZIONE
```

STRUTTURA DELLO SCRIPT

```
#!/usr/bin/env bash

echo "Come ti chiami?"
read nome

echo "Ciao $nome"
```

```
#!/usr/bin/env bash

read -p "Come ti chiami? " nome

echo "Ciao $nome"
```

STRUTTURA DELLO SCRIPT (2)

```
#!/usr/bin/env bash

echo "Quale è il tuo nome?"
read nome

echo "Quale è il tuo cognome?"
read cognome

echo "Ciao $nome $cognome"
```

STRUTTURA DELLO SCRIPT (2)

```
#!/usr/bin/env bash

read -p "Quale è il tuo nome? " nome

read -p "Quale è il tuo cognome? " cognome

echo "Ciao $nome $cognome"
```

ESERCIZIO #2

Creare uno script BASH che chieda all'utente i seguenti dati:

- **Nome**
- **Cognome**
- **Codice Fiscale**
- **Città di Residenza**
- **Telefono**

E li stampi a video come riepilogo, con una frase decisa da te.

ESERCIZIO #2 – SOLUZIONE A


```
#!/usr/bin/env bash
#soluzione banale

echo "Quale è il tuo nome?"
read nome
echo "Quale è il tuo cognome?"
read cognome
echo "Quale è il tuo codice fiscale?"
read cf
echo "In quale città vivi?"
read residenza
echo "Quale è il tuo numero di telefono?"
read telefono

echo "Ciao $nome $cognome so che abiti a
$residenza e ti posso chiamare al $telefono"
```

ESERCIZIO #2 – SOLUZIONE B


```
#!/usr/bin/env bash
#soluzione compatta con meno righe

read -p "Quale è il tuo nome?" nome
read -p "Quale è il tuo cognome?" cognome
read -p "Quale è il tuo codice fiscale?" cf
read -p "In quale città vivi?" residenza
read -p "Quale è il tuo numero di telefono?" telefono

echo "Ciao $nome $cognome so che abiti a $residenza e ti
posso chiamare al $telefono"
```

ESERCIZIO #2 – SOLUZIONE C


```
#!/usr/bin/env bash
#soluzione elegante

read -p $'Quali sono il tuo nome, cognome, codice fiscale, \ncittà di residenza e telefono? \nSepara con uno spazio I dati, senza virgole! \n' \
  nome cognome  cf residenza telefono

echo "Ciao $nome $cognome so che abiti a $residenza e ti posso chiamare al $telefono"
```

L'IDEA DI FONDO

Vogliamo ora raggiungere questo obiettivo:

```
./script.sh informazione
```

Cioè fornire allo script, da riga di comando, il dato da gestire, in modo che lo possa elaborare. Ad esempio:

```
./saluta.sh Marco
```

potrebbe stampare a video un saluto a Marco, tipo "Ciao Marco! Benvenuto alla serata tematica."

STRUTTURA DELLO SCRIPT

```
#!/usr/bin/env bash

echo "Ciao $1! Benvenuto alla
serata tematica."
exit 0
```

Da terminale, lancerò:

```
./benvenuto.sh Marco
```

con output:

```
Ciao Marco! Benvenuto alla
serata tematica.
```

FORNIRE VALORI DA RIGA DI COMANDO

Bash permette di inviare dei valori ad uno script, direttamente dal comando di avvio lanciato dalla CLI.

Ad esempio posso inviare 5 valori (o quanti voglio) allo script, al momento della richiesta di esecuzione:

```
./script.sh $1 $2 $3 $4 $5
```

Lo script Bash acquisirà quindi un array di valori in input.

ESERCIZIO #3

Creare uno script BASH che chieda all'utente i seguenti dati, stavolta da riga di comando:

- **Nome**
- **Cognome**
- **Email**

E li stampi a video come riepilogo, con una frase decisa da te.

ESERCIZIO #3 - SOLUZIONE


```
#!/usr/bin/env bash

echo "Nome: $1"

echo "Cognome: $2"

Echo "Email: $3"
```

```
./script.sh nome cognome email
```

RIPASSO DEI COMANDI POSIX

Per realizzare script in BASH dobbiamo anche ricordarci alcuni comandi Linux!

unzip – estrae un archivio compresso in Zip

basename – a fronte di un percorso assoluto di un file mostra solo il nome del file

wget – scarica una risorsa web, una volta fornito un URL

man nomecomando – se non ci ricordiamo la sintassi

ESERCIZIO #4

Creare uno script BASH che chieda all'utente un indirizzo web di un file .zip e lo estragga nella directory corrente, cioè quella in cui è stato salvato il file dello script.sh

L'URL è il seguente:

https://github.com/orvieto-linux/bash-scripting/archive/refs/heads/main.zip

ESERCIZIO #4 - SOLUZIONE


```
#!/usr/bin/env bash

URL="$1"
#basename ottiene il nome del file
FILE_NAME=$(basename "$URL")
wget "$URL"
unzip "$FILE_NAME"
echo "Scaricamento e decompressione
effettuati"
```

```
./script.sh
https://github.com/orvieto-linux/bash-
scripting/archive/refs/heads/main.zip
```

NON FUNZIONA LO SCRIPT? ATTENZIONE AGLI ERRORI COMUNI!

- Verificare sempre gli **spazi** attorno a = e la **sintassi** in generale
- Controllare eventuali **virgolette mancanti**
- Spesso ci si dimentica di **rendere eseguibile** con **chmod** il file dello script, questo è necessario per farlo funzionare!

Interessati a un corso sul BASH scripting?

CONTATTI

info@orvietolinux.it

www.orvietolinux.it

@orvietolinux