

BASH Programming

30 Bash Script Examples

6 years ago

by Fahmida Yesmin

Bash scripts can be used for various purposes, such as executing a shell command, running multiple commands together, customizing administrative tasks, performing task automation etc.

So knowledge of bash programming basics is important for every Linux user.

This article will help you to get the basic idea on bash programming.

Most of the common operations of bash scripting are explained with very simple examples here.

The following topics of bash programming are covered in this article.

“Gli script Bash possono essere utilizzati per vari scopi, come eseguire un comando shell, eseguire più comandi insieme, personalizzare attività amministrative, eseguire l'automazione delle attività, ecc.

Quindi la conoscenza delle basi della programmazione bash è importante per ogni utente Linux.

Questo articolo Vi aiuterà a farVi un'idea di base sulla programmazione bash.

La maggior parte delle operazioni più comuni dello scripting bash sono spiegate qui con esempi molto semplici.

I seguenti argomenti di programmazione bash sono trattati in questo articolo.”

Sommario

Create and Execute First BASH Program

- *Hello World*
- *Echo Command*
- *Use of comment*
- *Use of Multi-line comment*
- *Using While Loop*
- *Using For Loop Procedure*
- *Get User Input*

- *Using if statement*
- *Using if statement with AND logic*
- *Using if statement with OR logic*
- *Using else if (elif) statement*
- *Using Case Statement*
- *Get Arguments from Command Line*
- *Get arguments from command line with names*
- *Combine String variables*
- *Get substring of String*
- *Add Two Numbers*
- *Create Function*
- *Create function with Parameters*
- *Pass Return Value from Function*
- *Make Directory*
- *Make directory by checking existence*
- *Read a File*
- *Delete a File*
- *Append to File*
- *Test if File Exist*
- *Send Email*
- *Get Parse Current Date*
- *Wait Command*
- *Sleep Command*

Hello World

You can run **bash script** from the terminal or by executing any bash file. Run the following command from the terminal to execute a very simple bash statement.

The output of the command will be ‘Hello World’.

*“Puoi eseguire lo **script bash** dal terminale o eseguendo qualsiasi file bash.*

Esegui il seguente comando dal terminale per eseguire un'istruzione bash molto semplice: l'output del comando sarà "Hello World".

```
$ echo "Hello World"
```

Open any editor to create a bash file.

Here, nano editor is used to create the file and filename is set as 'First.sh'

“Apri qualsiasi editor per creare un file bash.

Qui, l'editor nano viene utilizzato per creare il file e il nome del file è impostato come First.sh”

```
$ nano First.sh
```

Add the following bash script to the file and save the file.

“Aggiungi il seguente script bash al file e salva il file”

```
#!/bin/bash
```

You can run bash file by two ways. One way is by using bash command and another is by setting execute permission to bash file and run the file. Both ways are shown here.

“Puoi eseguire il file bash in due modi.

Un modo è utilizzare il comando bash e un altro è impostare l'autorizzazione di esecuzione sul file bash ed eseguire il file. Entrambi i modi sono mostrati qui.”



```
$ bash First.sh
```



```
$ chmod a+x First.sh
```

```
$ ./First.sh
```

Ritorna a  **Sommario**

Echo Command

You can use **echo command** with various options. Some useful options are mentioned in the following example. When you use 'echo' command without any option then a newline is added by default. '-n' option is used to print any text without new line and '-e' option is used to remove backslash characters from the output. Create a new bash file with a name, 'echo_example.sh' and add the following script.

*“Puoi usare il **comando echo** con varie opzioni. Alcune opzioni utili sono menzionate nell'esempio seguente. Quando usi il comando "echo" senza alcuna opzione, per impostazione predefinita viene aggiunta una nuova riga. L'opzione '-n' viene utilizzata per stampare qualsiasi testo senza una nuova riga e l'opzione '-e' viene utilizzata per rimuovere i caratteri barra rovesciata dall'output. Crea un nuovo file bash con un nome "echo_example.sh" e aggiungi il seguente script.”*

```
#!/bin/bash
echo "Printing text with newline"
echo -n "Printing text without newline"
echo -e "\nRemoving \t backslash \t characters\n"
```

Run the file with bash command:

Esegui il file con il comando bash:

```
$ bash echo_example.sh
```

Ritorna a  **Sommario**

Use of comment

'#' symbol is used to add single line comment in bash script.

Create a new file named 'comment_example.sh' and add the following script with single line comment.

'#'Il simbolo viene utilizzato per aggiungere un commento a riga singola nello script bash.

Crea un nuovo file denominato "comment_example.sh" ed aggiungi il seguente script con un commento a riga singola.

```
#!/bin/bash  
  
# Add two numeric value  
  
((sum=25+35))  
  
#Print the result  
  
echo $sum
```

Run the file with bash command.

Esegui il file con il comando bash

```
$ bash comment_example.sh
```

Go to top

Vai all' inizio

Ritorna a  *Sommario*

Use of Multi-line comment

You can use **multi line comment** in bash in various ways.

A simple way is shown in the following example.

Create a new bash named 'multiline-comment.sh' and add the following script.

Here, ':' and " ' " symbols are used to add multiline comment in bash script.

This following script will calculate the square of 5.

*Puoi utilizzare il **commento su più righe** in bash in vari modi.*

Un modo semplice è mostrato nell'esempio seguente.

Crea una nuova bash denominata "multiline-comment.sh" e aggiungi lo script come a seguire.

Ricorda che i simboli " : " e " ' " vengono utilizzati per aggiungere commenti su più righe nello script bash.

Il seguente script calcolerà il quadrato di 5.

```
#!/bin/bash  
:  
'
```

The following script calculates the square value of the number, 5.

Lo script seguente calcola il valore quadrato del numero, 5.

```
((area=5*5))  
echo $area
```

Run the file with bash command.

Esegui il file con il comando bash.

```
$ bash multiline-comment.sh
```

You can check the following **link** to know more about the use of bash comment.

*Puoi controllare il seguente **link** per saperne di più sull'uso del commento bash.*

Go to top

Vai all'inizio

Ritorna a  **Sommario**

Using While Loop Procedure

Create a bash file with the name, 'while_example.sh', to know the use of while loop.

In the example, **while** loop will iterate for 5 times.

The value of count variable will increment by 1 in each step.

When the value of count variable will 5 then the while loop will terminate.

Crea un file bash con il nome " while_example.sh ", per conoscere l'uso del ciclo while.

*Nell'esempio, il ciclo **while** verrà ripetuto per 5 volte.*

Il valore della variabile count aumenterà di 1 in ogni passaggio.

Quando il valore della variabile count sarà 5, il ciclo while terminerà.

```
#!/bin/bash
valid=true
count=1
while [ $valid ]
do
echo $count
if [ $count -eq 5 ];
then
break
fi
((count++))
done
```

Run the file with bash command.

Esegui il file con il comando bash

```
$ bash while_example.sh
```

You can check the following link to know more about the use of bash while loop.

*Puoi controllare il seguente **link** per saperne di più sull'uso di bash while loop*

Go to top

Vai all'inizio

Ritorna a  **Sommario**

Using For Loop Procedure

The basic **for** loop declaration is shown in the following example. Create a file named 'for_example.sh' and add the following script using for loop.

Here, for loop will iterate for 10 times and print all values of the variable, counter in single line.

*La dichiarazione di base del ciclo **for** è mostrata nell'esempio seguente.*

Crea un file denominato "for_example.sh" e aggiungi il seguente script utilizzando il ciclo for. Qui, il ciclo for ripeterà per 10 volte e stamperà tutti i valori della variabile, contatore in un'unica riga.

```
#!/bin/bash
for (( counter=10; counter>0; counter-- ))
do
echo -n "$counter "
done
printf "\n"
```

Run the file with bash command.

Esegui il file con il comando bash

```
$ bash for_example.sh
```

You can use **for** loop” for different purposes and ways in your bash script. You can check the following link to know more about the use of bash for loop.

Puoi utilizzare il ciclo for per scopi e modi diversi nel tuo script bash.

*Puoi controllare il seguente **link** per saperne di più sull'uso di bash for loop.*

Go to top

Vai all'inizio

Ritorna a  **Sommario**

Get User Input

The **'read'** command is used to take input from user in bash.

Create a file named **'user_input.sh'** and add the following script for taking input from the user.

Here, one string value will be taken from the user and display the value by combining other string value.

Il comando 'read' viene utilizzato per ricevere input dall'utente in bash.

Crea un file denominato "user_input.sh" e aggiungi il seguente script per ricevere input dall'utente.

Qui, un valore di stringa verrà preso dall'utente e visualizzerà il valore combinando un altro valore di stringa.

```
#!/bin/bash
echo "Enter Your Name"
read name
echo "Welcome $name to LinuxHint"
```

Run the file with bash command.

```
$ bash user_input.sh
```

You can check the following link to know more about the use of bash user input.

Puoi controllare il seguente collegamento per saperne di più sull'uso dell'input dell'utente bash

Go to top

Vai all'inizio

Ritorna a  **Sommario**

Using if statement

You can use “if” condition with single or multiple conditions. Starting and ending block of this statement is defined by ‘if’ and ‘fi’. Create a file named ‘simple_if.sh’ with the following script to know the use of “if” statement in bash. Here, 10 is assigned to the variable “n”. If the value of “\$n” is less than 10, then the output will be “**It is a one digit number**”, otherwise the output will be “**It is a two digit number**”. For comparison, ‘-lt’ is used here. For comparison, you can also use ‘-eq’ for equality, ‘-ne’ for not equality and ‘-gt’ for greater than in bash script.

Puoi utilizzare la condizione “if” con condizioni singole o multiple. Il blocco iniziale e finale di questa affermazione è definito da “if” e “fi”.

Crea un file denominato “simple_if.sh” con il seguente script per conoscere l’uso dell’istruzione if in bash.

Qui, 10 è assegnato alla variabile “n.”

Se il valore di “\$n” è inferiore a 10, l’output sarà:

“È un numero a una cifra”, altrimenti l’output sarà:

“È un numero a due cifre”.

Per confronto, qui viene utilizzato “-lt”.

Per fare un confronto, puoi anche usare “-eq” per uguaglianza, “-ne” per non uguaglianza e “-gt” per maggiore rispetto allo script bash.

```
#!/bin/bash
n=10
if [ $n -lt 10 ];
then
echo "It is a one digit number"
else
echo "It is a two digit number"
fi
```

Run the file with bash command.

Esegui il file con il comando bash.

```
$ bash simple_if.sh
```

Go to top

Vai all’inizio

Ritorna a  **Sommario**

Using *if* statement with AND logic

Different types of logical conditions can be used in *if* statement with two or more conditions.

In the following example., how you can define multiple conditions in *if* statement using **AND** logic

'&&' is used to apply **AND** logic of *if* statement.

Create a file named 'if_with_AND.sh' to check the following code.

Here, the value of **username** and **password** variables will be taken from the user and compared with '**admin**' and '**secret**'.

If both values match then the output will be "**valid user**", otherwise the output will be "**invalid user**".

È possibile utilizzare diversi tipi di condizioni logiche nell'istruzione if con due o più condizioni.

Nell'esempio seguente viene illustrato come definire più condizioni nell'istruzione if utilizzando la logica AND.

'&&' viene utilizzato per applicare la logica AND dell'istruzione if. Crea un file denominato "if_with_AND.sh" per verificare il seguente codice.

Qui, il valore delle variabili "nome utente" e "password" verrà preso dall'utente e confrontato con "admin" e "segreto".

Se entrambi i valori corrispondono, l'output sarà "utente valido", altrimenti l'output sarà "utente non valido".

```
#!/bin/bash
echo "Enter username"
read username
echo "Enter password"
read password
if [[ ( $username == "admin" && $password ==
"secret" ) ]]; then
echo "valid user"
else
echo "invalid user"
fi
```

Run the file with bash command.

Esegui il file con il comando bash.

```
$ bash if_with_AND.sh
```

Go to top

Vai all'inizio

Ritorna a  **Sommario**

Using *if* statement with OR logic

'||' is used to define **OR** logic in **if** condition.

Create a file named 'if_with_OR.sh' with the following code to check the use of OR logic of if statement.

Here, the value of **n** will be taken from the user. If the value is equal to 15 or 45 then the output will be “**You won the game**”, otherwise the output will be “**You lost the game**”.

*"||" viene utilizzato per definire la logica **OR** nella condizione **if**.*

Crea un file denominato "if_with_OR.sh" con il seguente codice per verificare l'uso della logica OR dell'istruzione if.

*Qui, il valore di **n** verrà preso dall'utente. Se il valore è uguale a 15 o 45 allora l'output sarà “**Hai vinto la partita**”, altrimenti l'output sarà “**Hai perso la partita**”.*

```
#!/bin/bash
echo "Enter any number"
read n
if [[ ( $n -eq 15 || $n -eq 45 ) ]]
then
echo "You won the game"
else
echo "You lost the game"
fi
```

Run the file with bash command.

Esegui il file con il comando bash.

```
$ bash if_with_OR.sh
```

Go to top

Vai all'inizio

Ritorna a  **Sommario**

Using else if (elif) statement

The use of **else if** condition is little different in bash than other programming language.

'**elif**' is used to define **else if** condition in bash.

Create a file named, 'elseif_example.sh' and add the following script to check how else if is defined in bash script.

*L'uso della condizione **else if** è leggermente diverso in bash rispetto ad altri linguaggi di programmazione.*

*'**elif**' è usato per definire la condizione **else if** in bash.*

Crea un file denominato "elseif_example.sh" ed aggiungi il seguente script per verificare in quale altro modo è definito nello script bash.

```
#!/bin/bash
echo "Enter your lucky number"
read n
if [ $n -eq 101 ];
then
echo "You got 1st prize"
elif [ $n -eq 510 ];
then
echo "You got 2nd prize"
elif [ $n -eq 999 ];
then
echo "You got 3rd prize"
else
echo "Sorry, try for the next time"
fi
```

Run the file with bash command.

Esegui il file con il comando bash.

```
$ bash elseif_example.sh
```

Go to top

Vai all'inizio

Ritorna a  [Sommar](#)

Using Case Statement

Case statement is used as the alternative of “if-elseif-else” statement.

The starting and ending block of this statement is defined by ‘**case**’ and ‘**esac**’.

Create a new file named, ‘case_example.sh’ and add the following script. The output of the following script will be same to the previous else if example.

*L'istruzione **Case** viene utilizzata come alternativa all'istruzione “if-elseif-else”.*

*Il blocco iniziale e finale di questa istruzione è definito da "**case**" e "**esac**".*

Crea un nuovo file denominato "case_example.sh" e aggiungi il seguente script.

L'output del seguente script sarà il medesimo del prec. “else if”.

```
#!/bin/bash
echo "Enter your lucky number"
read n
case $n in
101)
echo echo "You got 1st prize" ;;
510)
echo "You got 2nd prize" ;;
999)
echo "You got 3rd prize" ;;
*)
echo "Sorry, try for the next time" ;;
esac
```

Run the file with bash command.

Esegui il file con il comando bash.

```
$ bash case_example.sh
```

Go to top

Vai all'inizio

Ritorna a  *Sommario*

Get Arguments from Command Line

Bash script can read **input** from command line argument like other programming language.

For example: \$1 and \$2 variable are used to read first and second command line arguments.

Create a file named “command_line.sh” and add the following script.

Two argument values read by the following script and prints the total number of arguments and the argument values as output.

*Lo script Bash può leggere **l'input** dall'argomento della riga di comando come piace in altri linguaggi di programmazione.*

Ad esempio: le variabili \$1 e \$2 vengono utilizzate per leggere il primo e il secondo argomento della riga di comando.

Crea un file denominato "command_line.sh" e aggiungi il seguente script.

Ottieni due valori di argomento che verranno interpretati dallo script seguente, stampando il numero totale di argomenti e i singoli valori di argomenti.

```
#!/bin/bash
echo "Total arguments : $#"
```

```
echo "1st Argument = $1"
```

```
echo "2nd argument = $2"
```

Run the file with bash command.

Esegui il file con il comando bash.

```
$ bash command_line.sh Linux Hint
```

You can check the following link to know more about the use of bash command line argument.

Puoi controllare il seguente collegamento per saperne di più sull'uso dell'argomento della riga di comando bash

Go to top

Vai all'inizio

Ritorna a  **Sommario**

Get arguments from command line with names

How you can read command line arguments with names is shown in the following script.

Create a file named, 'command_line_names.sh' and add the following code.

Here, two arguments, X and Y are read by this script and print the sum of X and Y.

Come leggere gli argomenti della riga di comando con i nomi è mostrato nello script seguente.

Crea un file denominato "command_line_names.sh" e aggiungi il seguente codice.

Qui, due argomenti, X e Y, vengono letti da questo script e stampano la somma di X e Y.

```
#!/bin/bash
for arg in "$@"
do
index=$(echo $arg | cut -f1 -d=)
val=$(echo $arg | cut -f2 -d=)
case $index in
X) x=$val;;

Y) y=$val;;

*)
esac
done
((result=x+y))
echo "X+Y=$result"
```

Run the file with bash command and with two command line arguments.

Esegui il file con il comando bash e con due argomenti della riga di comando.

```
$ bash command_line_names X=45 Y=30
```

Go to top

Vai all'inizio

Ritorna a  **Sommario**

Combine String variables

You can easily combine string variables in bash.

Create a file named “string_combine.sh” and add the following script to check how you can combine string variables in bash by placing variables together or using ‘+’ operator.

Puoi facilmente combinare variabili stringa in bash.

Crea un file denominato "string_combine.sh" ed aggiungi il seguente script per verificare come combinare le variabili stringa in bash posizionando insieme le variabili o utilizzando l'operatore "+".

```
#!/bin/bash
string1="Linux"
string2="Hint"
echo "$string1$string2"
string3=$string1$string2
string3+=" is a good tutorial blog site"
echo $string3
```

Run the file with bash command.

Esegui il file con il comando bash.

```
$ bash string_combine.sh
```

Go to top

Vai all'inizio

Ritorna a  **Sommario**

Get substring of String

Like other programming language, bash has no built-in function to cut value from any string data.

But you can do the task of substring in another way in bash that is shown in the following script.

To test the script, create a file named 'substring_example.sh' with the following code.

Here, the value, 6 indicates the starting point from where the substring will start and 5 indicates the length of the substring.

Come altri linguaggi di programmazione, bash non ha una funzione incorporata per tagliare il valore da qualsiasi dato di stringa.

Ma puoi eseguire il compito di sottostringa in un altro modo in bash, mostrato nel seguente script.

Per testare lo script, crea un file denominato "substring_example.sh" con il seguente codice.

Qui, il valore 6 indica il punto iniziale da cui inizierà la sottostringa e 5 indica la lunghezza della sottostringa.

```
#!/bin/bash
Str="Learn Linux from LinuxHint"
subStr=${Str:6:5}
echo $subStr
```

Run the file with bash command.

Esegui il file con il comando bash.

```
$ bash substring_example.sh
```

Go to top

Vai all'inizio

Ritorna a  **Sommario**

Add Two Numbers

You can do the arithmetical operations in bash in different ways.
How you can add two integer numbers in bash using double brackets is shown in the following script.
Create a file named 'add_numbers.sh' with the following code.
Two integer values will be taken from the user and printed the result of addition.

*Puoi eseguire le operazioni aritmetiche in bash in diversi modi.
Come aggiungere due numeri interi in bash utilizzando le doppie parentesi è mostrato nello script seguente.
Crea un file denominato "add_numbers.sh" con il seguente codice.
Due valori interi verranno presi dall'utente e verrà stampato il risultato della loro addizione.*

```
#!/bin/bash
echo "Enter first number"
read x
echo "Enter second number"
read y
(( sum=x+y ))
echo "The result of addition=$sum"
```

Run the file with bash command.

Esegui il file con il comando bash.

```
$ bash add_numbers.sh
```

You can check the following **link** to know more about bash arithmetic.

*Puoi controllare il seguente **link** per saperne di più sull'aritmetica di bash.*

Go to top

Vai all'inizio

Ritorna a  *Sommario*

Create Function

How you can create a simple function and call the function is shown in the following script.

Create a file named 'function_example.sh' and add the following code. You can call any function by name only without using any bracket in bash script.

Come puoi creare una funzione semplice e chiamare la funzione è mostrato nello script seguente.

Crea un file denominato "function_example.sh" e aggiungi il seguente codice

Puoi chiamare qualsiasi funzione solo per nome senza utilizzare alcuna parentesi nello script bash.

```
#!/bin/bash
function F1()
{
echo 'I like bash programming'
}

F1
```

Run the file with bash command.

Esegui il file con il comando bash.

```
$ bash function_example.sh
```

Go to top

Vai all'inizio

Ritorna a  **Sommario**

Create function with Parameters

Bash can't declare function parameter or arguments at the time of function declaration.

But you can use parameters in function by using other variable.

If two values are passed at the time of function calling then \$1 and \$2 variable are used for reading the values.

Create a file named 'function_parameter.sh' and add the following code. Here, the function, 'Rectangle_Area' will calculate the area of a rectangle based on the parameter values.

Bash non può dichiarare parametri o argomenti di funzione al momento della dichiarazione della funzione.

Ma puoi usare i parametri in funzione usando altre variabili.

Se vengono passati due valori al momento della chiamata della funzione, allora le variabili \$1 e \$2 verranno utilizzate per leggere questi valori.

Crea un file denominato "function_parameter.sh" e aggiungi il seguente codice.

Qui, la funzione "Rectangle_Area" calcolerà l'area di un rettangolo in base ai valori dei parametri \$1 e \$2.

```
#!/bin/bash
Rectangle_Area() {
    area=$(( $1 * $2 ))
    echo "Area is : $area"
}
Rectangle_Area 10 20
```

Run the file with bash command.

Esegui il file con il comando bash.

```
$ bash function_parameter.sh
```

Go to top

Vai all'inizio

Ritorna a  *Sommario*

Pass Return Value from Function

Bash function can pass both numeric and string values.

How you can pass a string value from the function is shown in the following example.

Create a file named, 'function_return.sh' and add the following code.

The function greeting() returns a string value into the variable val which prints later by combining with other string.

La funzione Bash può passare sia valori numerici che stringhe. Nell'esempio seguente viene illustrato come passare un valore stringa dalla funzione.

Crea un file denominato "function_return.sh" e aggiungi il seguente codice.

La funzione greet() restituisce un valore di stringa nella variabile val che viene stampato successivamente combinandolo con un'altra stringa.

```
#!/bin/bash
function greeting() {
  str="Hello, $name"
  echo $str
}
echo "Enter your name"
read name
val=$(greeting)
echo "Return value of the function is $val"
```

Run the file with bash command.

Esegui il file con il comando bash.

```
$ bash function_return.sh
```

You can check the following **link** to know more about the use of bash functions.

*Puoi controllare il seguente **link** per saperne di più sull'uso delle funzioni bash.*

Go to top

Vai all'inizio

Ritorna a  *Sommario*

Make Directory

Bash uses '**mkdir**' command to create a new directory.
Create a file named 'make_directory.sh' and add the following code to take a new directory name from the user.
If the directory name is not exist in the current location then it will create the directory, otherwise the program will display error.

*Bash utilizza il comando "**mkdir**" per creare una nuova directory.
Crea un file denominato "make_directory.sh" e aggiungi il seguente codice per prendere un nuovo nome di directory dall'utente.
Se il nome della directory non esiste nella posizione corrente, creerà la directory, altrimenti il programma visualizzerà un errore.*

```
#!/bin/bash  
echo "Enter directory name"  
read newdir  
'mkdir $newdir'
```

Run the file with bash command.

Esegui il file con il comando bash.

```
$ bash make_directory.sh
```

Go to top

Vai all'inizio

Ritorna a  *Sommario*

Make directory by checking existence

If you want to check the existence of directory in the current location before executing the '**mkdir**' command then you can use the following code.

Option '**-d**' is used to test a particular directory is exist or not.

Create a file named, 'directory_exist.sh' and add the following code to create a directory by checking existence.

*Se vuoi verificare l'esistenza della directory nella posizione corrente prima di eseguire il comando '**mkdir**', puoi utilizzare il seguente codice.*

*L'opzione '**-d**' viene utilizzata per verificare che una particolare directory esista o meno.*

Crea un file denominato "directory_exist.sh" e aggiungi il seguente codice per creare una directory verificandone l'esistenza.

```
#!/bin/bash
echo "Enter directory name"
read ndir
if [ -d "$ndir" ]
then
echo "Directory exist"
else
`mkdir $ndir`
echo "Directory created"
fi
```

Run the file with bash command.

Esegui il file con il comando bash.


```
$ bash directory_exist.sh
```

You can check the following link to know more about bash directory creation.

Puoi controllare il seguente collegamento per saperne di più sulla creazione della directory bash.

Go to top

Vai all'inizio

Ritorna a  *Sommario*

Read a File

You can read any file line by line in bash by using loop.
Create a file named, 'read_file.sh' and add the following code to read an existing file named, 'book.txt'.

*Puoi leggere qualsiasi file riga per riga in bash usando loop.
Crea un file denominato "read_file.sh" ed aggiungi il seguente codice per leggere un file esistente denominato "book.txt"*

```
#!/bin/bash
file='book.txt'
while read line; do
echo $line
done < $file
```

Run the file with bash command.

Esegui il file con il comando bash.

```
$ bash read_file.sh
```

Run the following command to check the original content of 'book.txt' file.

Esegui il comando seguente per verificare il contenuto originale del file "book.txt".

```
$ cat book.txt
```

You can check the following **link** to know the different ways to read file in bash.

*Puoi controllare il seguente **link** per conoscere i diversi modi di leggere il file in bash.*

Go to top

Vai all'inizio

Ritorna a  **Sommario**

Delete a File

The **'rm'** command is used in bash to remove any file.
Create a file named 'delete_file.sh' with the following code to take the filename from the user and remove.
Here, '-i' option is used to get permission from the user before removing the file.

*Il comando "**rm**" viene utilizzato in bash per rimuovere qualsiasi file. Crea un file denominato "delete_file.sh" con il seguente codice per prendere il nome del file dall'utente e rimuoverlo.
Qui, l'opzione '-i' viene utilizzata per ottenere l'autorizzazione dall'utente prima di rimuovere il file.*

```
#!/bin/bash
echo "Enter filename to remove"
read fn
rm -i $fn
```

Run the file with bash command.

Esegui il file con il comando bash.

```
$ ls
$ bash delete_file.sh
$ ls
```

Go to top

Vai all'inizio

Ritorna a  **Sommario**

Append to File

New data can be added into any existing file by using ‘>>’ operator in bash.

Create a file named ‘append_file.sh’ and add the following code to add new content at the end of the file.

Here, ‘Learning Laravel 5’ will be added at the of ‘book.txt’ file after executing the script.

Nuovi dati possono essere aggiunti a qualsiasi file esistente utilizzando l'operatore ">>" in bash.

Crea un file denominato "append_file.sh" e aggiungi il seguente codice per aggiungere nuovo contenuto alla fine del file.

Qui, "Learning Laravel 5" verrà aggiunto al file "book.txt" dopo aver eseguito lo script.

```
#!/bin/bash
echo "Before appending the file"
cat book.txt
echo "Learning Laravel 5">> book.txt
echo "After appending the file"
cat book.txt
```

Run the file with bash command.

Esegui il file con il comando bash.

```
$ bash append_file.sh
```

Go to top

Vai all'inizio

Ritorna a  *Sommario*

Test if File Exist

You can check the existence of file in bash by using ‘-e’ or ‘-f’ option.

The option ‘-f’ is used in the following script to test the file existence. Create a file named, ‘file_exist.sh’ and add the following code.

Here, the filename will pass from the command line.

Puoi verificare l'esistenza del file in bash utilizzando l'opzione "-e" o "-f".

L'opzione '-f' viene utilizzata nel seguente script per testare l'esistenza del file.

Crea un file denominato "file_exist.sh" e aggiungi il seguente codice. Qui, il nome del file passerà dalla riga di comando.

```
#!/bin/bash
filename=$1
if [ -f "$filename" ]; then
echo "File exists"
else
echo "File does not exist"
fi
```

Run the following commands to check the existence of the file.

Here, book.txt file exists and book2.txt is not exist in the current location.

Esegui i seguenti comandi per verificare l'esistenza del file.

Qui, il file book.txt esiste e book2.txt non esiste nella posizione corrente

```
$ ls
$ bash file_exist.sh book.txt
$ bash file_exist.sh book2.txt
```

Go to top

Vai all'inizio

Ritorna a  **Sommario**

Send Email

You can send email by using ‘mail’ or ‘sendmail’ command.
Before using these commands, you have to install all necessary packages.
Create a file named, ‘mail_example.sh’ and add the following code to send the email.

*Puoi inviare e-mail utilizzando il comando "mail" o "sendmail".
Prima di utilizzare questi comandi, è necessario installare tutti i pacchetti necessari.*

Crea un file denominato "mail_example.sh" e aggiungi il seguente codice per inviare l'e-mail.

```
#!/bin/bash
Recipient="admin@example.com"
Subject="Greeting"
Message="Welcome to our site"
`mail -s $Subject $Recipient <<< $Message`
```

Run the file with bash command.

Esegui il file con il comando bash.

```
$ bash mail_example.sh
```

Go to top

Vai all'inizio

Ritorna a  *Sommario*

Get Parse Current Date

You can get the current system date and time value using ``date`` command. Every part of date and time value can be parsed using `'Y'`, `'m'`, `'d'`, `'H'`, `'M'` and `'S'`.

Create a new file named `'date_parse.sh'` and add the following code to separate day, month, year, hour, minute and second values.

*È possibile ottenere la data e l'ora del sistema corrente utilizzando il comando **"date"**.*

*Ogni parte del valore di data e ora può essere analizzata utilizzando **"Y"**, **"m"**, **"d"**, **"H"**, **"M"** e **"S"**.*

*Crea un nuovo file denominato **"date_parse.sh"** e aggiungi il seguente codice per separare i valori di giorno, mese, anno, ora, minuto e secondo.*

```
#!/bin/bash
Year=`date +%Y`
Month=`date +%m`
Day=`date +%d`
Hour=`date +%H`
Minute=`date +%M`
Second=`date +%S`
echo `date`
echo "Current Date is: $Day-$Month-$Year"
echo "Current Time is: $Hour:$Minute:$Second"
```

Run the file with bash command.

Esegui il file con il comando bash.

```
$ bash date_parse.sh
```

Go to top

Vai all'inizio

Ritorna a  *Sommario*

Wait Command

“wait” is a built-in command of Linux that waits for completing any running process.

“wait” command is used with a particular “process id” or “job id.”

If no process id or job id is given with wait command then it will wait for all current child processes to complete and returns exit status.

Create a file named ‘wait_example.sh’ and add the following script.

“wait” è un comando integrato di Linux che attende il completamento di qualsiasi processo in esecuzione.

Il comando “wait” viene utilizzato con un particolare ID processo o ID lavoro.

Se non viene fornito alcun ID processo o ID lavoro con il comando wait, attenderà il completamento di tutti i processi figlio correnti e restituirà lo stato di uscita.

Crea un file denominato "wait_example.sh" e aggiungi il seguente script.

```
#!/bin/bash
echo "Wait command" &
process_id=$!
wait $process_id
echo "Exited with status $?"
```

Run the file with bash command.

Esegui il file con il comando bash.

```
$ bash wait_example.sh
```

You can check the following **link** to know more about bash linux wait command.

*Puoi controllare il seguente **link** per saperne di più sul comando wait di bash linux.*

Go to top

Vai all'inizio

Ritorna a  **Sommario**

Sleep Command

When you want to pause the execution of any command for specific period of time then you can use sleep command.

You can set the delay amount by seconds (s), minutes (m), hours (h) and days (d).

Create a file named 'sleep_example.sh' and add the following script.

This script will wait for 5 seconds after running.

Quando desideri mettere in pausa l'esecuzione di qualsiasi comando per un periodo di tempo specifico, puoi utilizzare il comando sleep.

È possibile impostare l'importo del ritardo in secondi (s), minuti (m), ore (h) e giorni (d).

Crea un file denominato "sleep_example.sh" e aggiungi il seguente script.

Questo script attenderà 5 secondi dopo l'esecuzione.

```
#!/bin/bash
echo "Wait for 5 seconds"
sleep 5
echo "Completed"
```

Run the file with bash command.

Esegui il file con il comando bash.

```
$ bash sleep_example.sh
```

You can check the following **link** to know more about bash linux sleep command.

*Puoi controllare il seguente **link** per saperne di più sul comando wait di bash linux.*

Go to top

Vai all'inizio

Ritorna a  **Sommario**

Hope, after reading this article you have got a basic concept on bash scripting language and you will be able to apply them based on your requirements.

About the author

Fahmida Yesmin

I am a trainer of web programming courses.

I like to write article or tutorial on various IT topics.

I have a YouTube channel where many types of tutorials based on Ubuntu, Windows, Word, Excel, WordPress, Magento, Laravel etc. are published:

[Tutorials4u Help](#).

[View all posts](#)

RELATED LINUX HINT POSTS

[Bash Cut Examples](#)

[How to Copy a Directory to Another Directory in Bash on Linux](#)

[How to Prompt Bash for User Input](#)

[Advanced Bash File Operations](#)

[How to Understand and Implement Bash Regular Expressions](#)

[How to Use Variables Effectively in Bash](#)

[How to Find Yesterday Date in Bash](#)

[Linux Hint LLC, editor@linuxhint.com](#)

[1309 S Mary Ave Suite 210, Sunnyvale, CA 94087](#)

[Privacy Policy and Terms of Use](#)

[Update Privacy Preferences](#)

[A Raptive Partner Site](#)
