# **Basic commands Overview :**

<https://www.javatpoint.com/linux-commands>

**General Purpose Utilities**

* **cal**: Displays a calendar.
* **date**: Shows the current system date and time.
* **cat**: Concatenates and displays file content.
* **ls**: Lists directory contents.

**Text Processing**

* **echo/printf**: Displays messages.
* **wc**: Counts lines, words, and characters in a file.
* **man**: Accesses manual pages for commands.

**System Information**

* **who**: Shows who is logged in.
* **pwd**: Displays the current directory.
* **tty**: Prints the terminal name.

**File Operations**

* **cp**: Copies files or directories.
* **rm**: Removes files or directories.
* **mv**: Moves or renames files or directories.
* **chmod**: Changes file permissions.
* **chown**: Changes file ownership.

**Directory Management**

* **cd**: Changes the current directory.
* **mkdir**: Creates a new directory.
* **rmdir**: Removes an empty directory.

**Viewing and Editing Files**

* **cat**: Displays file content.
* **more/less**: Paginates file content.
* **head**: Displays the first few lines of a file.
* **tail**: Displays the last few lines of a file.
* **grep**: Searches for patterns in files.

**System Status**

* **hostname**: Shows or sets the system’s hostname.
* **uptime**: Displays how long the system has been running.
* **uname**: Prints system information.

**Process Management**

* **ps**: Lists running processes.
* **top**: Displays system tasks and status.
* **kill**: Terminates a process.

**Networking**

* **ifconfig/ip**: Configures network interfaces.
* **ping**: Checks connectivity to a host.
* **netstat**: Displays network statistics.

# **Directory Commands:**

1. Pwd:
2. mkdir: makes directory
3. rmdir: deletes directory
4. ls: lists all directories
5. cd: change directory
6. touch: to create file
7. cat: create file/display content
8. rm: delete file
9. mv: move file to the specified location
10. cp: copy the content of 1 file to another
11. rename: rename file : rename ‘s/\.txt$/\.pdf/’.txt

# **File Contents Commands:**

head display content of file

tail: display the last 10 lines of the file content

tac: displays the file content in reverse order (from the last line).

More: more command displays screenful output at a time

Less :

# **User Commands:**

su: allows access of the Linux shell to another user. su<user name>

id: used to display the user ID (UID) and group ID (GID).

Useradd:  used to add or remove a user on a Linux server.

Passwd: used to create and change the password for a user.

groupadd: used to create a user group.

cut: used to select a specific column of a file.

cut -d(delimiter) -f(columnNumber) **<fileName>**

grep: used to search specified word from file

Linux Commands with Examples

Comm: to compare two files or streams. By default, it displays three columns, first displays non-matching items of the first file, second indicates the non-matching item of the second file, and the third column displays the matching items of both files.

wc: count the lines, words, and characters in a file.

od: used to display content of file in different format

1. od -b **<fileName>**      // Octal format
2. od -t x1 **<fileName>**   // Hexa decimal format
3. od -c **<fileName>**     // ASCII character format

gzip: truncate the file size. It is a compressing tool. It replaces the original file by the compressed file having '.gz' extension.

gunzip: to decompress file

# **Utility commands**

Find: [find](https://www.javatpoint.com/linux-find) command is used to find a particular file within a directory. It also supports various options to find a file such as byname, by type, by date, and more.

The following symbols are used after the find command:

(.) : For current directory name

(/) : For root

Locate

Date

Sleep

Time

zcat: display the compressed files.

df:  [df](https://www.javatpoint.com/linux-df) command is used to display the disk space

mount:

exit:

clear

# **Shell Variables**

* Shell variables are character strings in a shell that store some value, which could be an integer, filename, string, or a shell command itself.123
* There are rules that must be followed when writing variables in a script, such as variable names must start with an alphabet or underscore and cannot start with a number.456
* Variables can be defined using the syntax **variable\_name=<variable data>.** These are called scalar variables as they can hold one value at a time.7
* Variables can be accessed by appending the variable name with $, such as **$VAR\_1$VAR\_2.8**
* The **unset command can be used to delete a variable and its stored data from the list of variables.**8
* Read-only variables are variables whose values cannot be modified later in the script.91011

## **Types of Shell Variables**

* Local Variables: Variables that are specific to the current instance of the shell and not available to other programs or shells.12131415
* Environment Variables: Variables commonly used to configure the behavior of scripts and programs run by the shell. They are created once and can be used by any user.161718
* Shell Variables: Variables set by the shell itself to help the shell work with functions correctly. They can be both environment and local variables.192021

## **Shell Scripting**

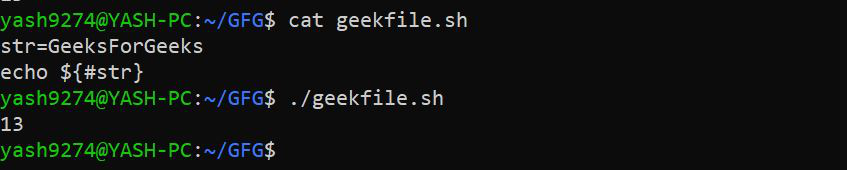
* Shell scripting is a way of writing scripts or programs that are executed in a terminal or shell.2223
* Shell variables are essential for shell scripting, allowing you to store data, take input from users, print values, and store temporary data or command outputs.2425
* Shell scripting is powerful because it can automate tasks and utilize programming constructs like loops, conditionals, and functions.26

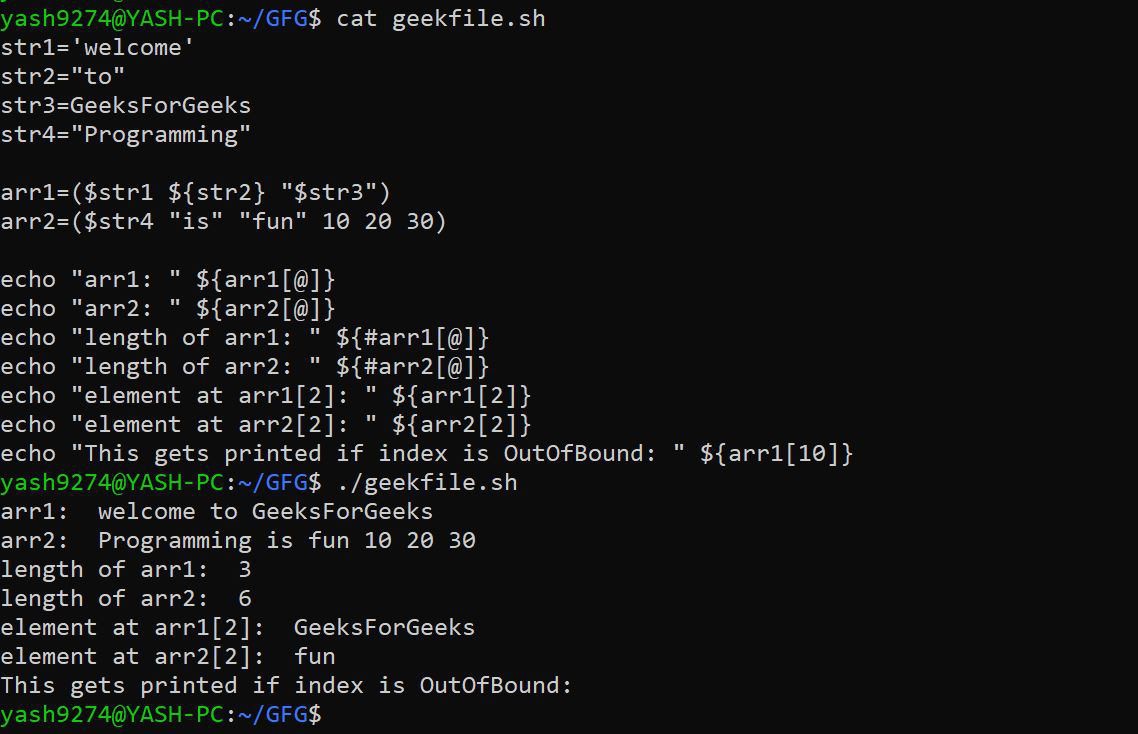
**Examples**

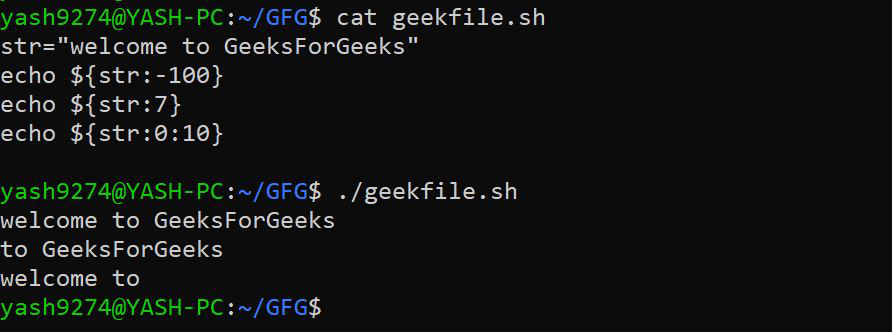
* Example of storing user data in variables and calculating the area of a rectangle.2728
* Example of storing and displaying a message based on the time of day using shell variables

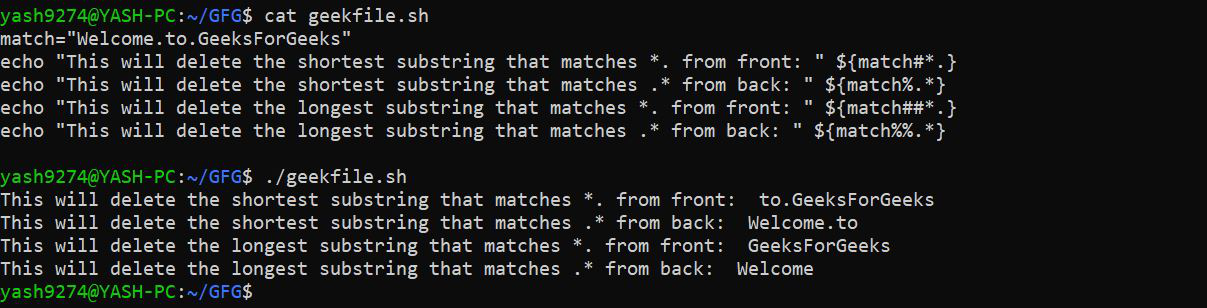
# **String manipulation:**









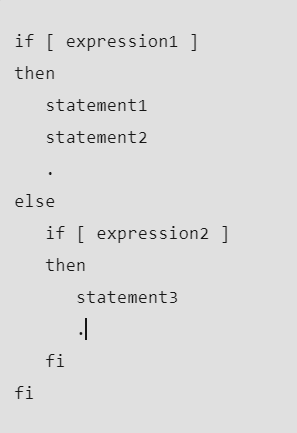


# **Arrays:**

|  |
| --- |
| *# To declare a static Array arr=(“Jayesh” “Shivang” “1” “Vipul” “Nishant” “2”)*  *# To print all elements of the array echo “All elements of the array:” echo “${arr[@]}” echo “${arr[\*]}”*  *# To print the first element echo “The first element:” echo “${arr[0]}”*  *# To print a selected index element selected\_index=3 echo “Selected index element at index $selected\_index:” echo “${arr[$selected\_index]}”*  *# To print elements from a particular index echo “Elements from a particular index:” echo “${arr[@]:2}” # Prints elements starting from index 2 echo “${arr[\*]:2}” # Prints elements starting from index 2*  *# To print elements in a range echo “Elements in a range:” echo “${arr[@]:1:3}” # Prints elements from index 1 to 3 echo “${arr[\*]:1:3}” # Prints elements from index 1 to 3* |

|  |
| --- |
| *# Declare a static Array arr=(“Jayesh” “Shivang” “1” “rishabh” “Vipul” “Nishtan”)*  *# Count the length of a particular element in the array element\_length=${#arr[2]} echo “Length of element at index 2: $element\_length”*  *# Count the length of the entire array array\_length=${#arr[@]} echo “Length of the array: $array\_length”*  *# Search in the array search\_result=$(echo “${arr[@]}” | grep -c “Jayesh”) echo “Search result for ‘Jayesh’: $search\_result”*  *# Search and replace in the array replaced\_element=$(echo “${arr[@]/Shivang/SHIVANG}”) echo “Array after search & replace: ${replaced\_element[\*]}”*  *# Delete an element in the array (index 3) unset arr[3]*  *echo “Array after deletion: ${arr[\*]}”* |

# Control structures:



|  |
| --- |
| CARS="bmw"  #Pass the variable in string  case "$CARS" in  #case 1  "mercedes") echo "Headquarters - Affalterbach, Germany" ;;    #case 2  "audi") echo "Headquarters - Ingolstadt, Germany" ;;    #case 3  "bmw") echo "Headquarters - Chennai, Tamil Nadu, India" ;;  esac |

# Loop:

|  |
| --- |
| for i in 1 2 3 4 5; do  echo "Number: $i"  done |

|  |
| --- |
| count=1  while [ $count -le 5 ]; do  echo "Count: $count"  count=$((count + 1))  done |

Functions:

**What are Functions in Shell Scripting?**

* A function is a collection of statements that execute a specified task.12
* Functions help break down complicated procedures into simpler subroutines that can be reused throughout the script.2

**Benefits of Using Functions**

* Code Reusability: Functions allow you to write code once and use it multiple times.3
* Enhanced Readability: Functions help organize code into logical blocks, making scripts easier to read and understand.4
* Modularity: Functions enable modular programming, where different tasks are handled by separate, self-contained functions.5
* Ease of Maintenance: Functions simplify debugging and updating code since changes can be made in one place rather than throughout the script.6

**Basic Structure of a Function**

* The basic syntax for defining a function in shell scripting is:

 The body of the function can include any sequence of valid shell commands or statements.8

function\_name() {

// body of the function

}

**Examples of Functions**

* The example demonstrates a function 'is\_prime()' that checks if a given number is prime and prints all the prime numbers in a given range.9101112

**Types of Functions**

1. Functions that return a value to the caller using the 'return' keyword. [[13]] [[14]] [[15]] [[16]] [[17]]
2. Functions that terminate the shell using the 'exit' keyword. [[18]] [[19]] [[20]] [[21]]
3. Functions that alter the value of a variable or variables. [[22]] [[23]]
4. Functions that echo output to the standard output. [[24]] [[25]]