



VARIABLE SCOPE OF FUNCTION

>> how we can work with the variables & functions.

```
add(){  
  local num1=5 { local variables }  
  local num2=3  
  echo $((num1+num2))  
}  
  
div() X
```



```
num1=5 { global variables }  
num2=3  
multi(){  
  echo $((num1*num2))  
}  
  
div() ✓
```



>> Variables are divided into 2 types according to how they are declared.



>> Variables declared directly inside the function are called as 'Local Variables'.

>> The particular function can only have the access of the local variables, and it cannot be accessed outside the function.

>> Variables declared in a script by default(not inside any function) are called 'Global Variables'.

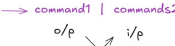
>> We can access these global variables throughout the script.

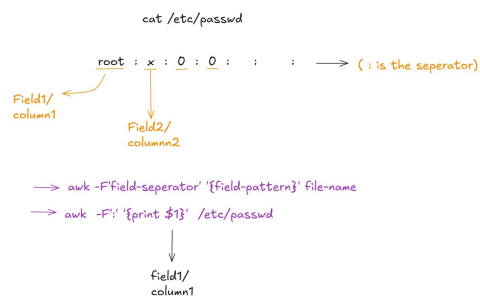
>> If you want to make a variable local, you have to explicitly mention 'local' in front of the variable. If not mentioned, that variable will be considered as a 'global' variable only, no matter how & where the variable is declared.

SCRIPT:

1. DECLARE A GLOBAL VAR, greetings = "Good Morning"
2. CREATE A FUNCTION show_greet() & ACCESS THE GLOBAL VARIABLE & PRINT IT.
3. CREATE ANOTHER FUNCTION, modify_greet() and try to edit value of greetings variable.
4. INSIDE modify_greet(), PRINT greeting VARIABLE.

ADDITIONAL COMMANDS

1. **grep**: Finding matching patterns in a file.
→ `grep "pattern" file-name`
2. **head**: used to view the first few line of a file.
→ `head file-name` (by default it shows the starting 10 lines of a file)
→ `head -5 file-name` (top/starting 5 lines)
3. **wc -l**: counts the no. of lines in a file.
→ `wc -l file-name`
4. **nl**: numbers of line.
→ `nl file-name`
5. **tail**: used to view the last few line of a file.
→ `tail file-name` (by default it shows the bottom 10 lines of a file)
→ `tail -5 file-name` (last 5 lines)
6. **pipeline (|)**: connects the o/p of the 1st commands as an input to the 2nd command.

7. **wget**: Downloading files from Internet (Tooplate)
→ `wget URL`
→ `wget -O new-name URL` (renaming the downloaded file)
8. **sort**: sorts the file in ascending order.
→ `sort file-name`
→ `sort -r file-name`
9. **curl**: Download the data from internet (transfers data using URL)
→ `curl URL`
10. **pgrep**: fetches the Process ID of a particular process.
→ `pgrep process-name`
11. **ps aux**: shows all the processes running.
12. **awk**: splits each line into field/columns based on a separator.



SCRIPT: CREATE A DIR AND CREATE 5 FILES IN IT. THEN, WRITE A SCRIPT TO COUNT THE FILES IN THAT DIRECTORY.

SCRIPT: WRITE A SCRIPT TO COPY ALL *.SH FILES TO A PATRICULAR DIRECTORY

```
cp *.sh dir_name
```

SCRIPT: WRITE A SCRIPT TO FIRST LIST THE ITEMS IN HOME DIRECTORY & THEN IDENTIFY WHETHER IT IS A FILE OR A DIRECTORY

```
ls
for i in *
do
    if [[ -d "$i" ]];then          -d -> whether dir exists or not
    echo "$i is a Directory"      -f -> whether a regular file
    elif [[ -f "$i" ]];then
    echo "$i is a File"
    fi
done
```

```
file1 10kb
file2 20kb
file3 17kb
dir1 40kb
dir2 50kb
file4 0kb
file5 ls -l35kb
```

SCRIPT: DISPLAY TOP 5 LARGEST ENTITY BY SIZE IN YOUR DIRECTORY

```
ls -ls | grep "=" |head -5
```

SCRIPT: INSTALL THE LIST OF TOOLS IN THE SYSTEM USING FOR LOOP

```
git, plocate, zip, unzip
```