Linux Classroom Series – 04/Sept/2020

Reading input from files

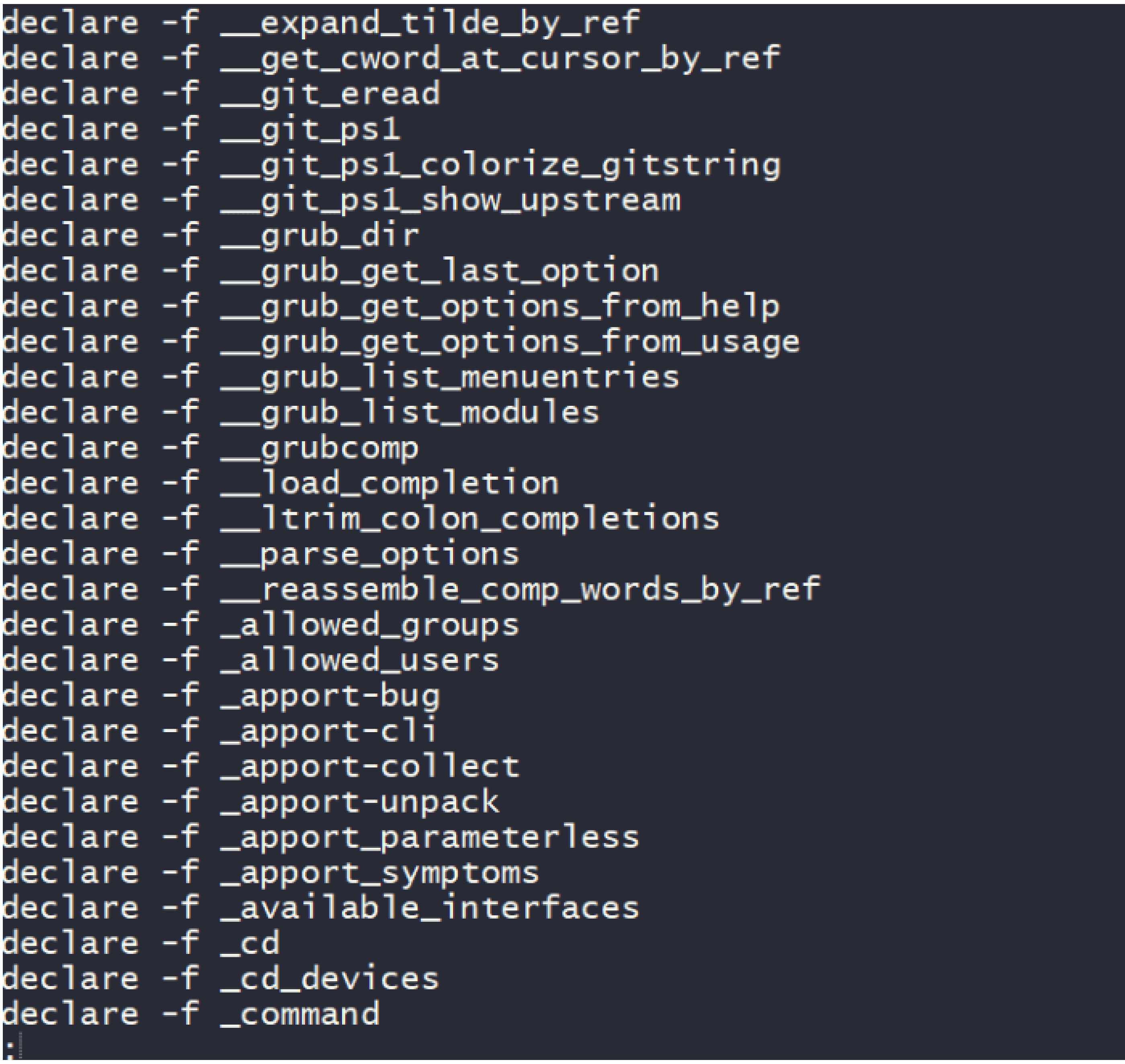
- Lets assume we have list of server ip addresses or hosts in the file called as servers.txt.
- We are asked to find which servers are up
- This script is working but it is unable to redirect the output to a text

```
#!/bin/bash

# Usage: ./checkservers <servers-filepath>
# servers-filepath is a text file with each server
in new line

if [ ! -f "$1" ] ;
then
    echo "The input to $0 should be a file"
fi
```

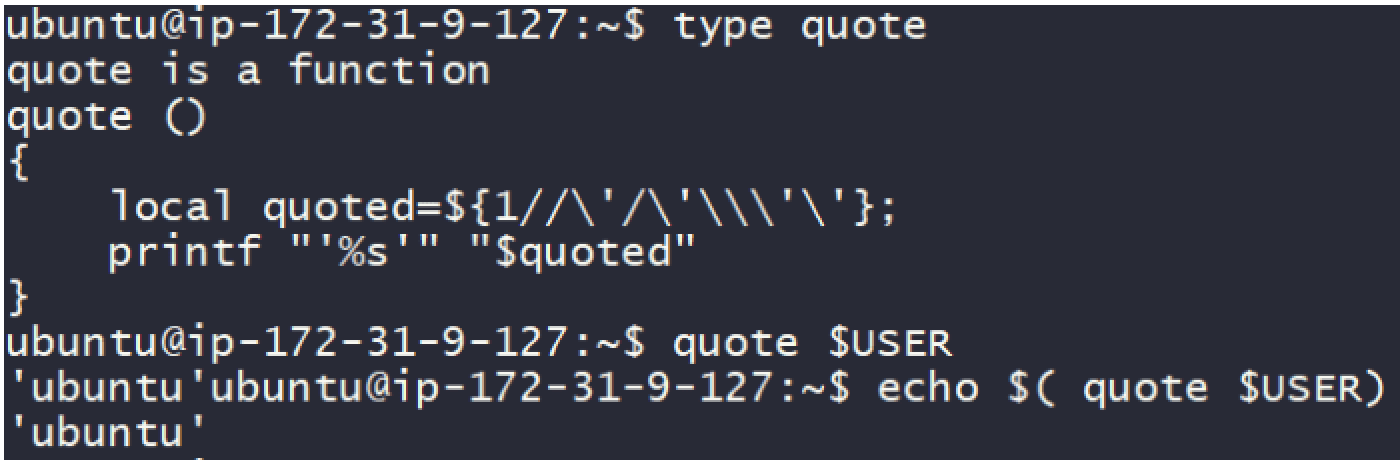
```
echo "The following servers are up on $(date +%x)"
> checkservers.out
while read server;
do
    ping -c1 "$server"&& echo "Serverup $server"
>> checkservers.out
done < $1

cat checkservers.out
```

Building Blocks for reusability : Functions

- DRY principle(Don’t Repeat Your Self)
- We will cover the following aspects
  - Introduction
  - Passing Parameters to the functions
  - Variable scope
  - Returning values from functions
  - Recursive functions
- Functions are internally represented as blocks of code in memory as *named elements*. These elements can be created within shell environment, as well as within the script execution.
- Execute `declare -F` in the bash. The output of this command might vary with
  distribution



```
declare -f __expand_tilde_by_ref
declare -f __get_cword_at_cursor_by_ref
declare -f __git_eread
declare -f __git_ps1
declare -f __git_ps1_colorize_gitstring
declare -f __git_ps1_show_upstream
declare -f __grub_dir
declare -f __grub_get_last_option
declare -f __grub_get_options_from_help
declare -f __grub_get_options_from_usage
declare -f __grub_list_menuentries
declare -f __grub_list_modules
declare -f __grubcomp
declare -f __load_completion
declare -f __ltrim_colon_completions
declare -f __parse_options
declare -f __reassemble_comp_words_by_ref
declare -f _allowed_groups
declare -f _allowed_users
declare -f _apport-bug
declare -f _apport-cli
declare -f _apport-collect
declare -f _apport-unpack
declare -f _apport_parameterless
declare -f _apport_symptoms
declare -f _available_interfaces
declare -f _cd
declare -f _cd_devices
declare -f _command
:
```



```
ubuntu@ip-172-31-9-127:~$ type quote
quote is a function
quote ()
{
    local quoted=${1//\'/\'\\\'\'};
    printf "'%s'" "$quoted"
}
ubuntu@ip-172-31-9-127:~$ quote $USER
'ubuntu'ubuntu@ip-172-31-9-127:~$ echo $( quote $USER)
'ubuntu'
```

- Functions can be created using the following two syntaxes
- Syntax 1:
  ```
  function-name() {
      <code to be executed>
  }
  ```
- Syntax 2:
  ```
  function <function-name> {
      <code to be executed>
  }
  ```
- Lets start with a simple function
  ```
  show_system_details() {
      echo "Uptime is"
      uptime
      echo "Cpu details"
      lscpu
      echo "User list"
      who
  }

  is_file() {
      if [ ! -f "$1" ]; then
          echo "$1 is not a file"
          exit 2
      fi
  }

  backup_file() {
      is_file "$1"
      new_file_loc="${1}.bak"
      cp $1 $new_file_loc
      echo "file is copied to $new_file_loc"
  }

  backup_file "/home/ubuntu/1.txt"
  ```