105.1 Scripts

Un fichero, que nos permite a nosotros como usuarios ejecutar múltiples tareas o acciones por medio de una acción. Ejemplos de lenguajes de programación:

BASH / SH / DASH / C++

.EXE

Primer ejemplo de script:

cat Script_Variable.sh

#!/bin/bash

echo "El script que imprime el valor de la variable PRUEBA"

echo " "

echo "El valor que muestra la variable Prueba es:" $PRUEBA

Ahora vamos a ver una variable definida localmente:

```
El valor que muestra la variable Prueba es:
root@cesarm:/home/cesarm/Ejercicios# PRUEBA=Valor1
root@cesarm:/home/cesarm/Ejercicios# echo $Prueba
Valor1
```

Con el primer comando ENV no se nos muestra información porque la variable no a sido exportada aún, más el segundo SET nos muestra tanto variables locales de ambientes como las exportadas de usuario.

```
root@cesarm:/home/cesarm/Ejercicios# env|grep PRUEBA
root@cesarm:/home/cesarm/Ejercicios# set |grep PRUEBA
PRUEBA=Valor1
```

El valor no es encontrado porque no encontré la variable prueba:

```
root@cesarm:/home/cesarm/Ejercicios# ./Script_Variable.sh
El script que imprime el valor de la variable PRUEBA

El valor que muestra la variable Prueba es:
```

Hay una forma de ejecutar el script en la sección local y es por eso por lo que existe el comando **source o punto espacio**, ejemplo:

```
root@cesarm:/home/cesarm/Ejercicios# source Script_Variable.sh
El script que imprime el valor de la variable PRUEBA

El valor que muestra la variable Prueba es: Valor1
root@cesarm:/home/cesarm/Ejercicios# . Script_Variable.sh
El script que imprime el valor de la variable PRUEBA

El valor que muestra la variable Prueba es: Valor1
```

Recordando el comando alias nos permite crear un atajo como el siguiente ejemplo:

```
root@cesarm:/home/cesarm/Ejercicios# alias dt="date +%H:%M"
root@cesarm:/home/cesarm/Ejercicios# dt
09:14
root@cesarm:/home/cesarm/Ejercicios# alias
alias dt='date +%H:%M'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
```

Así como Alias crea rutinas en nuestro ambiente vamos a ver algunas otras utilidades. El comando function normalmente es utilizado para crear una rutina de comandos y podríamos utilizar if, loop, while que estudiaremos más adelante.

function funcion1 {

> date;

> uptime;

> uname -a;

> echo "Fin de la funcion";

> }

```
root@cesarm:/home/cesarm/Ejercicios# function funcion1 {
> date;
> uptime;
> uname -a;
> echo "Fin de la funcion";
> }
root@cesarm:/home/cesarm/Ejercicios# funcion1
vie  1 feb 09:22:48 CST 2019
 09:22:48 up 52 min,  2 users,  load average: 0,00, 0,00, 0,00
Linux cesarm 4.15.0-43-generic #46-Ubuntu SMP Thu Dec 6 14:45:28 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux
Fin de la funcion
```

¿Como veo la definición de esta función? Digitamos el comando set y al final no debería aparecer en algún momento la funcion que definimos:

```
funcion1 ()
{
    date;
    uptime;
    uname -a;
    echo "Fin de la funcion"
}
quote ()
{
    local quoted=${1//\'/\'\\\'\'};
    printf "'%s'" "$quoted"
}
quote_readline ()
{
    local quoted;
    _quote_readline_by_ref "$1" ret;
    printf %s "$ret"
}
```

Ahora a hacer otros ejemplos de una función como se muestra a continuación:

```
root@cesarm:/home/cesarm/Ejercicios# function funcion2 { date; uptime; }
root@cesarm:/home/cesarm/Ejercicios# funcion2
vie  1 feb 09:34:42 CST 2019
 09:34:42 up  1:04,  2 users,  load average: 0,00, 0,00, 0,00
root@cesarm:/home/cesarm/Ejercicios#
```

```
root@cesarm:/home/cesarm/Ejercicios# funcion3 () { uptime ; uname -v ; hostname; }
root@cesarm:/home/cesarm/Ejercicios# funcion3
 09:38:11 up  1:07,  2 users,  load average: 0,00, 0,00, 0,00
#46-Ubuntu SMP Thu Dec 6 14:45:28 UTC 2018
cesarm
root@cesarm:/home/cesarm/Ejercicios#
```

```
root@cesarm:/home/cesarm/Ejercicios# funcion4 () {
> uptime;
> echo prueba;
> }
root@cesarm:/home/cesarm/Ejercicios# funcion4
 09:40:11 up  1:09,  2 users,  load average: 0,00, 0,00, 0,00
prueba
```

También podemos crear una lista de valores o array como en se nos muestra a continuación:

```
root@cesarm:/home/cesarm/Ejercicios# lista=(valor1 valor2 valor3 valor4)
root@cesarm:/home/cesarm/Ejercicios# set |grep lista
lista=([0]="valor1" [1]="valor2" [2]="valor3" [3]="valor4")
root@cesarm:/home/cesarm/Ejercicios# echo ${lista[2]}
valor3
root@cesarm:/home/cesarm/Ejercicios# echo ${lista[0]}
valor1
```

```
root@cesarm:/home/cesarm/Ejercicios# array=(A B C D)
root@cesarm:/home/cesarm/Ejercicios# echo ${array[3]}
D
root@cesarm:/home/cesarm/Ejercicios# echo ${array[0]}
A
```

Ahora si queremos removerlo utilizamos el siguiente comando:

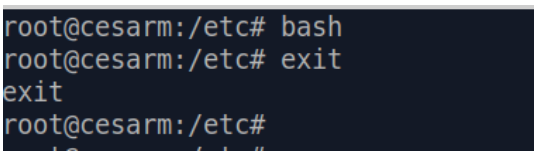
```
root@cesarm:/home/cesarm/Ejercicios# env |grep array
root@cesarm:/home/cesarm/Ejercicios# unset array
root@cesarm:/home/cesarm/Ejercicios#
```

113. 105.1 Personalización y Uso del Ambiente Shell - Archivos de configuración

Vamos a ver los archivos que definen las configuraciones de todos los usuarios:

```
root@cesarm:/etc# ls /etc/profile
/etc/profile
root@cesarm:/etc# ls /etc/bash.bashrc
/etc/bash.bashrc
root@cesarm:/etc#
```

Si digitamos bash abrimos una nueva sesión de bash y luego con el comando exit salimos de esta sesión, esto no incluyo un procedimiento de logearse en el Shell.



Lo mismo sucede de igual forma si abrimos cualquier nueva sesión de terminal:

Siempre que comenzamos un nuevo login estamos abriendo un nuevo Shell o bash; mas no siempre que estoy abriendo un nuevo bash estoy logeandome.

etc/profile: Es usado cuando cualquier usuario del sistema hace un procedimiento de logearse; tanto para interfaz gráfica como interfaz de solo texto.

etc/bash.bashrc: Es aplicado cuando se abre una nueva terminal, una nueva sesión de bash o Shell.

vi /etc/profile: Vamos a tener aquí adentro varias rutinas que ya fueron implementadas en el sistema. La primera no dice que si encuentra el fichero bashrc que ejecute el mismo bashrc:

```
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).

if [ "${PS1-}" ]; then
  if [ "${BASH-}" ] && [ "$BASH" != "/bin/sh" ]; then
    # The file bash.bashrc already sets the default PS1.
    # PS1='\h:\w\$ '
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
```

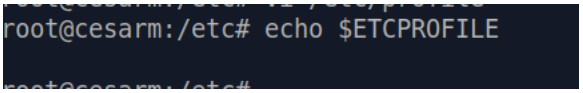
vi /etc/profile: Otra rutina que tenemos es que si yo tengo el /etc/profile.d ejecute todo lo que este dentro de este perfil:

```
20 if [ -d /etc/profile.d ]; then
21   for i in /etc/profile.d/*.sh; do
22     if [ -r $i ]; then
23       . $i
24     fi
25   done
26   unset i
27 fi
:set number
```

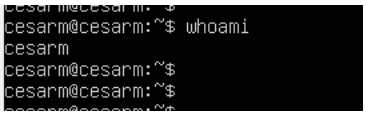
Estos son algunos ejemplos de scripts que están corriendo el sistema. Vamos a hacer un ejemplo con una variable; nos logeamos como root (raíz).

```
if [ -d /etc/profile.d ]; then
  for i in /etc/profile.d/*.sh; do
    if [ -r $i ]; then
      . $i
    fi
  done
  unset i
fi
ETCPROFILE=Valor1
:wq
```

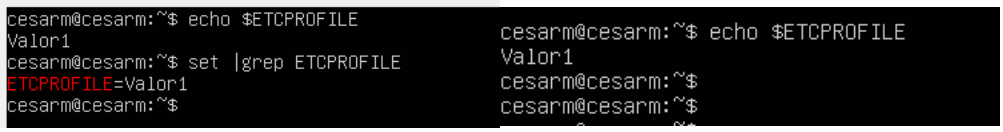
Si abrimos una nueva terminal y ejecutamos echo $ETCPROFILE, no va a encontrar nada porque no hicimos un procedimiento de login.



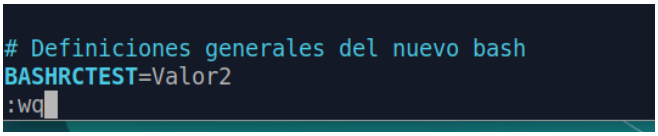
Ahora nos logeamos en la interfaz de texto como usuario local:



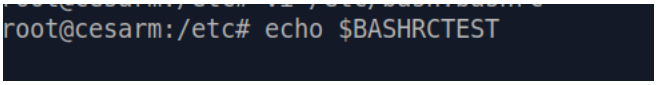
Ahora si se nos a reflejar el valor1 como se muestra a continuación:



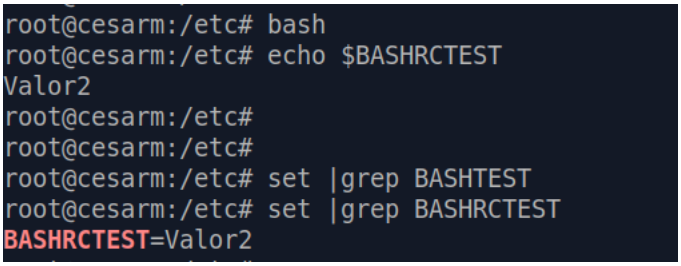
Vamos a editar el directorio /etc/bash.bashrc al final del documento como se muestra a continuación:



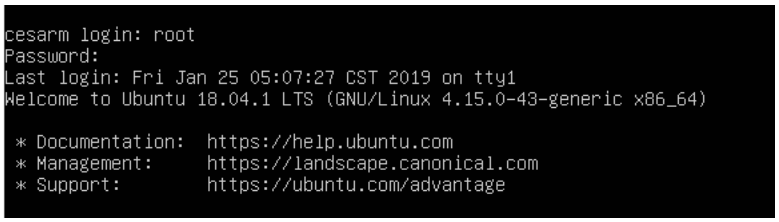
De momento no vamos a encontrar nada porque no hemos iniciado una nueva sesión bash o del Shell:

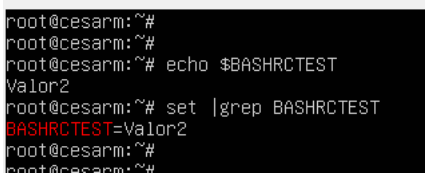


Vamos a realizar la prueba luego de abrir una nueva sesión de Shell:

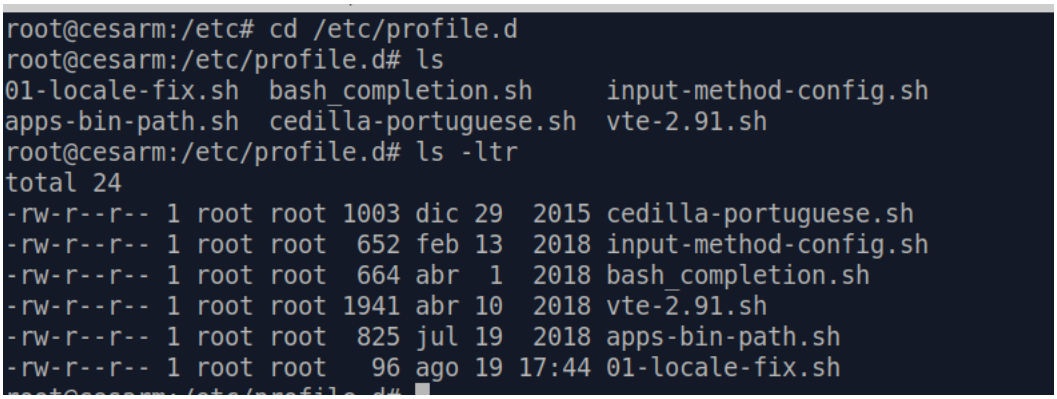


Accedamos el terminal numero 2 (tty2) como usuario raíz



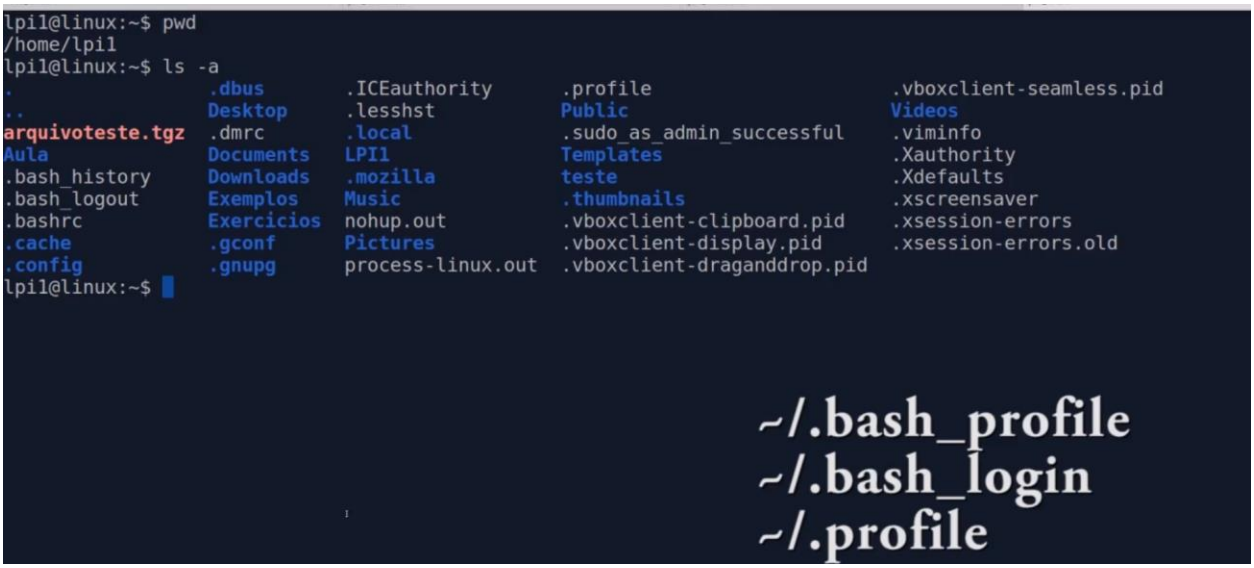


Entonces estas son las diferencias entre el etc/bash.bashrc y el etc/profile



Cada vez que nos logueamos; estos scripts se van a cargar. Entonces son estos 2 archivos que hacen las definiciones generales que son aplicadas a todos los usuarios.

Dependiendo de la distribución de Linux podríamos tener algunos de los archivos que se muestran en la imagen a continuación:



.profile: Es donde voy a colocar mis definiciones personales que van a ser colocadas en todo login nuevo.

vi .profile: vamos a realizar los siguiente cambios al final del documento, para que cada vez que nos logueamos cargue estas variables.

```
# set Path so it includes user's private bin directories
PATH="$HOME/bin:$HOME/.local/bin:$PATH"
LOCALPROFILE="TEST"
alias tt="date;uptime"
```

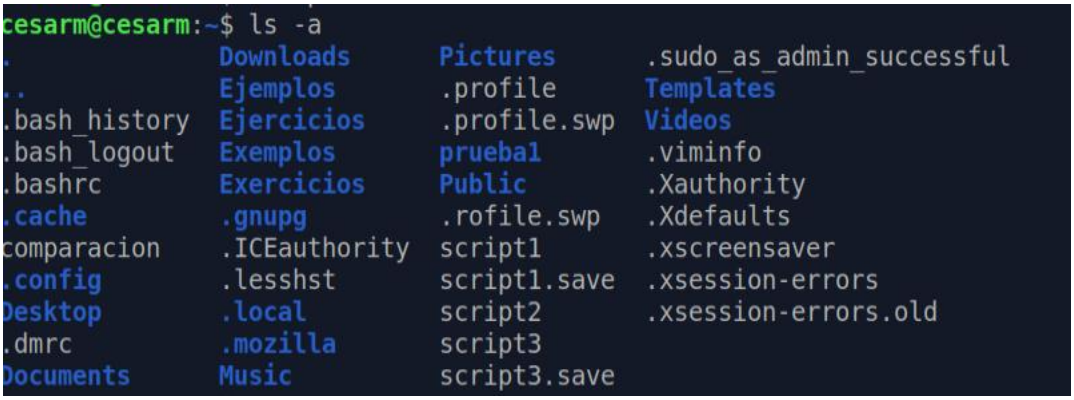
.bashrc: Este si es correspondiente a toda vez que realizamos una nueva sesión. Aquí tenemos por ejemplo varios aliases que están definidos aquí:

```
# enable color support of ls and also add handy aliases
if [ -x /usr/bin/dircolors ]; then
    test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)" || eval "$(dirc
lors -b)"
    alias ls='ls --color=auto'
    #alias dir='dir --color=auto'
    #alias vdir='vdir --color=auto'

    alias grep='grep --color=auto'
    alias fgrep='fgrep --color=auto'
    alias egrep='egrep --color=auto'
fi

# colored GCC warnings and errors
#export GCC_COLORS='error=01;31:warning=01;35:note=01;36:caret=01;32:locus=01:c
ote=01'

# some more ls aliases
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'
```

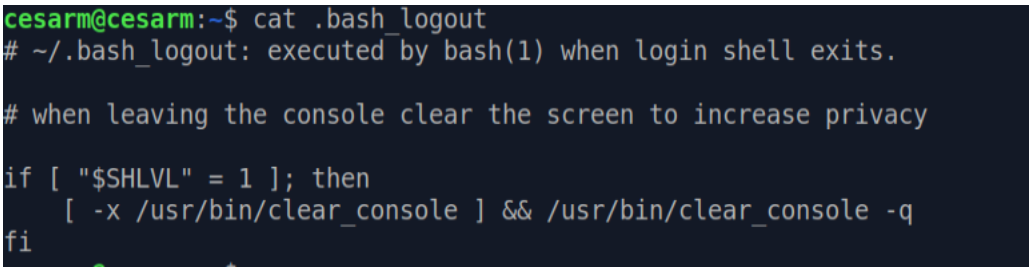
Entonces estos son los archivos utilizados para hacer definiciones globales de todos los usuarios y de cada usuario específicamente.

```
cesarm@cesarm:~$ vi .bashrc
cesarm@cesarm:~$ vi .profile
cesarm@cesarm:~$
```

El usuario tiene otro archivo importante que es el .bash_logout



Esto quiere decir lo que tenemos dentro de este archivo va a ser ejecutado toda vez que el usuario va a ser un logout:



/etc/inputrc: Es una forma de definir comandos en nuestro terminal, entonces si tenemos alguna pregunta de este tema se refiere a definiciones sobre el terminal.

```
# "\e[6~": end-of-history

# alternate mappings for "page up" and "page down" to search the history
# "\e[5~": history-search-backward
# "\e[6~": history-search-forward

# mappings for Ctrl-left-arrow and Ctrl-right-arrow for word moving
"\e[1;5C": forward-word
"\e[1;5D": backward-word
"\e[5C": forward-word
"\e[5D": backward-word
"\e\e[C": forward-word
"\e\e[D": backward-word

$if term=rxvt
"\e[7~": beginning-of-line
"\e[8~": end-of-line
"\eOc": forward-word
"\eOd": backward-word
$endif
```

vi /etc/skel: otra definición de ambiente importante es el directorio esqueleto los archivos que estén dentro van a ser creados cada vez que hagamos un nuevo usuario en el home del usuario.

```
root@cesarm:~# touch test-skel /etc/skel
root@cesarm:~# ls -la /etc/skel
total 40
drwxr-xr-x   3 root root  4096 feb  1 18:32 .
drwxr-xr-x 129 root root 12288 feb  1 14:02 ..
-rw-r--r--   1 root root   220 abr  4  2018 .bash_logout
-rw-r--r--   1 root root  3771 abr  4  2018 .bashrc
drwxr-xr-x   3 root root  4096 abr 26  2018 .config
-rw-r--r--   1 root root   807 abr  4  2018 .profile
-rw-r--r--   1 root root  1600 feb 28  2018 .Xdefaults
-rw-r--r--   1 root root    14 feb 28  2018 .xscreensaver
root@cesarm:~# ls -a
.   .bash_history  .cache   .dbus   .local    test-skel
..  .bashrc        .config  .gnupg  .profile  .viminfo
root@cesarm:~#
```