Solución 4.4: Configuración avanzada de shell

Descripción de la tarea

En este laboratorio, los estudiantes practicarán la personalización de la shell y la programación básica de shell realizando las siguientes tareas:

- Ver y modificar variables locales y de entorno para personalizar la shell.
- Crear, usar y evitar alias para ejecutar comandos dentro de la shell.
- Definir funciones que ejecuten múltiples comandos.
- Usar listas de comandos para ejecutarlos de forma lógica.
- Personalizar los archivos de inicialización de bash.
- Crear varios scripts bash sencillos.

Pasos de la tarea

1. Recuerda que las variables de la shell contienen información útil para la propia shell y otros programas ejecutables. Muestra todas las variables (tanto locales como de entorno) ejecutando el comando *set*.

```
root@debian:~# set
BASH=/bin/bash
BASHOPTS=checkwinsize:cmdhist:complete_fullquote:expand_aliases:extglob:extquote:f
orce_fignore:globasciiranges:globskipdots:interactive_comments:login_shell:patsub_
replacement:progcomp:promptvars:sourcepath
BASH_ALIASES=()
BASH_ARGC=([0]="0")
BASH_ARGV=()
BASH_CMDS=()
...
```

2. En algunos casos, querrás distinguir entre variables de entorno y variables locales. Para mostrar solo las variables de entorno, ejecuta el comando *env*:

```
root@debian:~# env | head
SHELL=/bin/bash
PWD=/root
LOGNAME=root
HOME=/root
LANG=es_ES.UTF-8
TERM=xterm-256color
```

Nota: Existen variables de entorno y locales.

- **Variable de entorno:** Es accesible en la shell actual y también en cualquier proceso o shell hijo que se inicie desde ella, ya que se hereda automáticamente. Estas variables tienen que ser exportadas. Para verlas usamos el comando *env*.

- **Variable local:** Solo existe y es accesible en la shell o sesión donde se creó; no se hereda a otros procesos o shells hijos. Para verlas usamos el comando *set*.

3. Para establecer la variable local PS1, ejecuta el siguiente comando `PS1='\# \$ '`. Esta variable local cambiará el prompt para mostrar el número de historial del comando actual y mostrará el carácter del prompt como *$*. El número que se mostrará refleja un conteo continuo de los comandos en el historial. Este número aumentará con cada comando que ejecutes. El número que aparezca en tu propia ventana puede variar, dependiendo de cuántos comandos hayas ejecutado desde que iniciaste la shell.

```
root@debian:~# PS1='\# \$ '
2 #
```

4. Antes de cambiar la variable de entorno COLUMNS, observa cómo esta variable afecta el número de columnas que muestran comandos como *ls /bin*.

```
2 # ls /bin
'['                             gtbl                            rmdir
 aa-enabled                     gunzip                          rnano
 aa-exec                        gzexe
rotatelogs
 aa-features-abi                gzip                            routel
 ab                             h2ph                            rpcgen
 acpi_listen                    h2xs                            rpcinfo
 addpart                        hardlink                        rrsync
```

Nota: Se ha omitido parte de la salida en la terminal anterior para mayor brevedad. La salida parcial mostrada casi llena el ancho de la terminal porque el valor predeterminado de la variable COLUMNS es 125.

```
3 # echo $COLUMNS
125
```

5. Cambia la variable COLUMNS a un valor de 60 y usa el comando export para convertirla en una variable de entorno:

```
4 # export COLUMNS=60
5 # env | grep COLUMNS
COLUMNS=60
26 # ls /bin
...
```

6. Muestra la variable de entorno PATH, que determina en qué directorios se buscarán los comandos ejecutados desde la línea de comandos.

```
6 # echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

7. Para probar la variable PATH, primero crea un nuevo directorio en tu directorio personal llamado scripts ejecutando el siguiente comando.

```
root@debian:~# mkdir scripts && ls
scripts
```

8. Crea un script que muestre la fecha actual.

```
root@debian:~# echo '#!/bin/bash' > scripts/today
root@debian:~# echo 'echo Today is $(date +%D)' >> scripts/today
root@debian:~# cat scripts/today
#!/bin/bash
echo Today is $(date +%D)
```

Nota: Cada comando echo toma el texto entre comillas simples que le sigue inmediatamente y redirige esa salida al archivo scripts/today. Cuando un solo símbolo > (mayor que) sigue al texto, ese texto sobrescribe cualquier contenido existente en el archivo. Cuando dos símbolos >> (mayor que) siguen al texto, ese texto se añade al final del contenido existente en el archivo.

9. Ejecuta el script creado.

```
root@debian:~# bash scripts/today
Today is 05/16/25
```

10. Si quieres poder ejecutar el script directamente (sin usar el comando bash), los permisos del usuario deben incluir permisos de lectura y ejecución. De lo contrario, se mostrará un mensaje indicando que el permiso ha sido denegado. Por defecto, los archivos que se crean nunca tienen permiso de ejecución automáticamente. Intenta ejecutar el script, sabiendo que no tiene permiso de ejecución.

```
root@debian:~# scripts/today
-bash: scripts/today: Permiso denegado
```

11. Ver el listado detallado del archivo con el comando ls te permitirá ver los permisos del archivo. Usa *ls -l* para ver los detalles del archivo de script. Ejecuta el script en el contorno actual de tu shell (ten en cuenta que esto si fuera el caso podría afectar al entorno actual). A continuación, dale permisos de ejecución con el comando *chmod* y ejecuta el script.

```
root@debian:~# ls -l scripts/
total 4
-rw-r--r-- 1 root root 38 may 16 12:31 today

root@debian:~# . scripts/today
Today is 05/16/25

root@debian:~# source scripts/today
Today is 05/16/25

root@debian:~# chmod u+x scripts/today
root@debian:~# ls -l scripts/today

root@debian:~# scripts/today
Today is 05/16/25

root@debian:~# /root/scripts/today
Today is 05/16/25
```

12. Aunque las rutas relativas y absolutas funcionan para ejecutar el script, usar solo el nombre del script fallará porque el directorio scripts no está incluido como valor en la variable PATH. Intenta ejecutar el script usando solo su nombre; fallará porque el directorio actual no se busca para comandos.

```
root@debian:~# today
-bash: today: orden no encontrada

root@debian:~/scripts# today
-bash: today: orden no encontrada
```

13. Muestra nuevamente la variable PATH para confirmar que el directorio scripts no está incluido en ella. Modifica la variable PATH para que contenga el directorio scripts y vuelve a mostrar la variable PATH para verificar el cambio.

```
root@debian:~# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
root@debian:~# PATH=$PATH:$HOME/scripts
root@debian:~# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/root/scripts
root@debian:~# today
Today is 05/16/25
```

14. Los cambios realizados en las variables dentro de la shell no persisten después de que el usuario cierra sesión en el sistema. Escribe *exit* para cerrar la sesión, vuelve a conectarte, muestra la variable PATH y trata de ejecutar el sctipt.

```
root@debian:~# exit
cerrar sesión

vagrant@debian:~$ exit
cerrar sesión
Connection to 192.168.33.11 closed.

PS C:\Users> ssh vagrant@192.168.33.11

root@debian:~# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

root@debian:~# today
-bash: today: orden no encontrada
```

15. Modifica el archivo `.profile` puede usarse para personalizar variables como PATH. La variable PATH se establecerá con el valor de la variable PATH existente y el directorio $HOME/bin. Observa el carácter dos puntos (😃 que separa las rutas.

```
root@debian:~# tail -n3 .profile
### Modificación del PATH
PATH=$PATH:$HOME/scripts
export PATH

root@debian:~# . .profile

root@debian:~# today
Today is 05/16/25
```

Nota: Ejecutamos . *.profile* para cargar el archivo *.profile*, ahora bien, si volvieramos a acceder como root automáticamente se cargaría la variable PATH modificada. Es mucho más seguro probar los cambios en los archivos de inicialización de Bash usando el comando source en vez de cerrar sesión y volver a iniciarla. Si cometes un error al modificar uno de estos archivos y luego cierras sesión, podrías impedir que el usuario pueda volver a iniciar sesión correctamente. En cambio, si hay un error al "sourcear" el archivo mientras ya estás conectado, el problema se puede corregir fácilmente, ya que el entorno de la shell sigue activo y puedes editar el archivo para solucionar el error.

16. Muestra la variable PATH para verificar que ahora incluye el directorio scripts después de haber ejecutado el comando source sobre el archivo `~/.profile`.

```
root@debian:~# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/root/scripts
```

17. Los archivos que están en el directorio /etc/skel se copian automáticamente al directorio personal de un nuevo usuario cuando se crea su cuenta. Ejecuta el siguiente comando para ver estos archivos *ls -a /etc/skel*:

```
root@debian:~# ls -a /etc/skel
.  ..  .bash_logout  .bashrc  .profile
```

18. Para añadir cuentas de usuario o copiar archivos en el directorio `/etc/skel` se requieren privilegios de superusuario. Copia el archivo `.profile` personalizado desde el directorio personal del usuario superadministrador. Al colocar este nuevo .bash_profile en el directorio /etc/skel, cualquier nueva cuenta BASH tendrá automáticamente el contenido de este archivo en el directorio personal del nuevo usuario. Esto permite al administrador crear personalizaciones que afectarán a todas las nuevas cuentas de usuario.

```
root@debian:~# cp .profile /etc/skel/.profile
```

19. Crea el usuario *prueba* y comprueba su PATH. En este punto ya tenemos añadido al PATH del usuario la ruta donde deberían figurar los scipts del nuevo usuario.

```
root@debian:~# useradd -m -s /bin/bash -p $(mkpasswd 'abc123.') prueba
root@debian:~# tail -n1 /etc/passwd
prueba:x:1003:1003::/home/prueba:/bin/bash

root@debian:~# su - prueba
prueba@debian:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/home/prueba/scripts
```

Nota: Como el archivo `~/.profile` se procesa solo una vez, normalmente cuando el usuario inicia sesión, es un buen lugar para poner asignaciones personalizadas de variables de entorno, ya que estas variables *solo necesitan crearse una vez por cuenta*. El archivo `~/.bashrc` se procesa cada vez que un usuario inicia una nueva shell bash, como al abrir una nueva ventana de terminal. El archivo `~/.bashrc` es un buen lugar para poner alias y funciones que deben ejecutarse para definirse cada vez que el usuario inicia una nueva shell.

20. Antes de añadir alias y funciones al archivo /etc/skel/.bashrc, se recomienda que los pruebes en una shell interactiva para asegurarte de que funcionan como esperas. Primero, visualiza los alias que están definidos actualmente en la shell ejecutando el comando alias.

```
root@debian:~$ alias
alias ls='ls --color=auto'
```

21. Para añadir un alias a tu shell, utiliza el comando alias. Ejecuta los siguientes comandos para definir un nuevo alias llamado lt, que mostrará los archivos en orden inverso de tiempo, y luego muestra los alias actuales de la shell.

```
prueba@debian:~$ alias lt='ls -lrt'
```

```
prueba@debian:~$ lt
total 0

prueba@debian:~$ lt /var
total 40
drwxrwsr-x  2 root staff 4096 mar 29  2024 local
drwxr-xr-x  2 root root  4096 jul 23  2024 opt
lrwxrwxrwx  1 root root     4 jul 23  2024 run -> /run
lrwxrwxrwx  1 root root     9 jul 23  2024 lock -> /run/lock
drwxr-xr-x  3 root root  4096 abr  5 10:25 www
drwxr-xr-x 11 root root  4096 abr  5 10:25 cache
drwxr-xr-x  2 root root  4096 may  9 09:39 backups
drwxr-xr-x 27 root root  4096 may  9 20:09 lib
drwxr-xr-x  5 root root  4096 may  9 20:09 spool
drwxr-xr-x  9 root root  4096 may 12 09:01 log
drwxrwxrwt  5 root root  4096 may 16 15:08 tmp
drwxrwsr-x  2 root mail  4096 may 16 21:46 mail
```

22. Para eliminar la definición del alias de tu sesión actual de shell, utiliza el comando unalias junto con el nombre del alias. Ejecuta los siguientes comandos para eliminar el alias lt y verifica que ha sido eliminado.

```
prueba@debian:~$ unalias lt
prueba@debian:~$ lt /var
-bash: lt: orden no encontrada
```

23. Aunque los alias suelen ser nombres más cortos para comandos largos, las funciones te permiten ejecutar múltiples comandos y trabajar con argumentos que se pasan a estos comandos. Para ver las funciones que están definidas en tu shell Bash actual, ejecuta el comando *declare -f*.

```
prueba@debian:~$ declare -f
__expand_tilde_by_ref ()
{
    if [[ ${!1-} == \~* ]]; then
        eval $1="$(printf ~%q "${!1#\~}")";
    fi
}
...
```

24. Una función puede definirse escribiendo primero el nombre que se le va a asignar, seguido de un par de paréntesis, y luego los comandos y opciones de la función entre un par de llaves. Por ejemplo, la siguiente función listará los archivos en orden inverso por tamaño:

```
lS() { ls -lrS $@; }
```

La función utiliza el comando *ls* con las opciones *-lrS* para listar los archivos en orden inverso por tamaño y la variable *$@* para referirse a cualquier argumento que se le haya pasado a la función. Ejecuta los siguientes comandos para crear la función y verificar que existe (nota: el nombre de la función es lS, una "l" minúscula y una "S" mayúscula).

```
prueba@debian:~$ lS() { ls -lrS $@; }
prueba@debian:~$ declare -f
...
```

Nota: Solo hay funciones locales a la sesión, no de entorno.

```
prueba@debian:~$ set | grep lS
lS ()
prueba@debian:~$ env | grep lS
```

25. El comando *unset* se puede usar para eliminar la definición de una función dentro de una shell. Ejecuta los siguientes comandos para eliminar la función y verificar que ya no existe.

```
prueba@debian:~$ unset lS
prueba@debian:~$ declare -f
...
```

26. Después de haber probado el alias *lt* y la función *lS*, ya estás listo para ponerlos a disposición de los usuarios añadiéndolos al archivo .bashrc en el directorio `/etc/skel`.

```
root@debian:~# echo "alias lt='ls -lrt'" >> /etc/skel/.bashrc
root@debian:~# echo 'lS() { ls -lrS $@; }' >> /etc/skel/.bashrc
root@debian:~# tail -n2 /etc/skel/.bashrc
alias lt=ls -lrt
lS() { ls -lrS $@; }
```