Laboratorio práctico: Bash Scripting Advanced

Tiempo estimado necesario: **30** minutos

Objetivos

Después de completar esta práctica de laboratorio, podrá utilizar las siguientes funciones del shell bash

- Metacaracteres
- Cotización
- Variables
- Sustitución de mando
- Redirección de E / S
- Tubos y filtros
- Argumentos de la línea de comandos

Acerca de Skills Network Cloud IDE

Skills Network Cloud IDE (basado en Theia y Docker) proporciona un entorno para laboratorios prácticos para laboratorios relacionados con cursos y proyectos. Theia es un IDE (entorno de desarrollo integrado) de código abierto, que se puede ejecutar en el escritorio o en la nube. Para completar este laboratorio, usaremos el IDE de la nube basado en Theia ejecutándose en un contenedor Docker.

Aviso importante sobre este entorno de laboratorio

Tenga en cuenta que las sesiones para este entorno de laboratorio no se conservan. Cada vez que se conecta a este laboratorio, se crea un nuevo entorno para usted. Todos los datos que haya guardado en la sesión anterior se perderán. Planifique completar estas prácticas de laboratorio en una sola sesión para evitar perder sus datos.

Ejercicio 1 - Metacaracteres

Ejecute los comandos que se indican a continuación en una terminal recién abierta.

▶ Haga clic aquí para obtener una pista

Hay varios caracteres que tienen significados especiales para el caparazón.

A continuación se muestran algunos de los caracteres especiales y su uso.

1.1 '#' - Para agregar comentarios

Las líneas que comienzan con # (¡con la excepción de #!) Son comentarios y no se ejecutarán.

```
# This is a comment line
```

1,2 ';' - Separador de comandos

Se pueden separar varios comandos entre sí mediante un; cuando se usa en una sola línea de comando.

```
pwd;date
```

1.3 * - comodín utilizado en la expansión del nombre de archivo

El carácter '*' coincide con cualquier número de cualquier carácter en los patrones de nombre de archivo. Por sí mismo, coincide con todos los nombres de archivo de un directorio determinado.

El siguiente ejemplo enumera todos los archivos cuyo nombre termina con un '.conf' en el directorio / etc.

```
ls /etc/*.conf
```

1.4 '?' - comodín utilizado en la expansión del nombre de archivo

El '?' carácter representa un solo carácter en un patrón de nombre de archivo.

El siguiente comando enumera todos los archivos cuyo nombre comienza con cualquier carácter seguido de 'grep'.

```
ls /bin/?grep
```

Ejercicio 2 - Cotización

Si algún carácter especial debe tratarse sin su significado especial, debemos citarlo.

Los siguientes ejemplos muestran cómo se realizan las cotizaciones en shell.

2.1 Citando usando barra invertida (\)

La barra invertida elimina el significado del carácter especial que le sigue.

```
echo The symbol for multiplicaton is \*
```

2.2 Cotización utilizando comillas simples (')

Un par de comillas simples elimina los significados especiales de todos los caracteres especiales dentro de ellas (excepto otra comilla simple).

```
echo 'Following are some special characters in shell - < > ; " ( ) \ [ ]  '
```

2.3 Citando usando comillas dobles (")

Un par de comillas dobles elimina los significados especiales de todos los caracteres especiales dentro de ellos, *excepto otra comilla doble, sustitución de variable y sustitución de comando. .*

Pruebe los ejemplos a continuación con comillas dobles y comillas simples para ver la diferencia entre su uso.

```
echo "Current user name: $USERNAME"
```

```
echo 'Current user name: $USERNAME'
```

Ejercicio 3: trabajar con variables

Acerca de las variables

Las variables ayudan a almacenar datos para el script. Los datos pueden tener la forma de un número o una cadena de caracteres.

Puede crear, eliminar o mostrar las variables.

Veamos ahora cómo se utilizan en el shell.

3.1 Enumere las variables ya definidas en el shell.

```
set
```

Debería ver muchas variables en la salida.

3.2 Crear nuevas variables

Utilice la sintaxis **nombre_variable = valor** .

Cree una nueva variable llamada "saldo" con un valor de 10000. Enumere todas las variables nuevamente.

```
balance=10000
```

Ejecute el comando set para comprobar si `balance`se ha creado la variable .

```
set
```

3.3 Crear una variable de entorno

Las variables de entorno son como cualquier otra variable. Se diferencian en el hecho de que se copian en cualquier proceso hijo creado desde el shell.

El comando de **exportación** se puede utilizar para convertir una variable regular en una variable de entorno.

Haga de la variable 'balance' una variable de entorno.

```
export balance
```

3.4 Lista de variables de entorno

Utilice el siguiente comando para enumerar todas las variables de entorno.

```
env
```

Debería ver muchas variables en la salida.

3.5 Mostrar el valor de una variable

Para mostrar o interpolar el valor de una variable en un comando, usamos la función de shell llamada **sustitución de variables** .

Se realiza anteponiendo el nombre de la variable con un símbolo de **$** (dólar).

El siguiente comando imprime el valor de la variable $ balance.

```
echo "Current account balance is $balance"
```

3.6 Eliminar una variable

Para eliminar variables, use el comando **unset** .

Eliminar la variable "saldo".

```
unset balance
```

Ejecute el comando set para verificar si la variable `balance`ha sido eliminada.

```
set
```

Ejercicio 4 - Sustitución de comandos

La sustitución de comandos es una característica del shell, que ayuda a guardar la salida generada por un comando en una variable.

También se puede usar para anidar varios comandos, de modo que los comandos externos puedan usar la salida del comando más interno.
El comando interno está encerrado en **$ ()** y se ejecutará primero.

Probemos con los siguientes ejemplos.

** 4.1 Almacene la salida del comando `hostname -i`en una variable llamada $ myip

```
myip=$(hostname -i)
echo $myip
```

4.2 Imprima el siguiente mensaje en pantalla:

"Ejecutando en el host: *host_name* ",

Donde 'host_name' debería ser su nombre de host actual.

```
echo "Running on host: $(hostname)"
```

La sustitución de comandos también se puede realizar utilizando la sintaxis de comillas inversas.

```
ls -l `which cat`
```

La salida del comando `which cat`es la ruta al comando `cat`. Esta ruta se envía a ls -l como entrada. Debería ver los permisos para el archivo `cat`en la salida.

Ejercicio 5: redireccionamiento de E / S

Linux envía la salida de un comando a **la salida estándar (pantalla)** y cualquier error generado se envía a **error estándar (pantalla)** .

De manera similar, la entrada requerida por un comando se recibe desde **la entrada estándar (teclado)** .

Si es necesario cambiar estos valores predeterminados, cáscara proporciona una característica llamada **de E / S de redirección** .

Esto se logra utilizando los siguientes caracteres especiales.

Símbolo	Significado
<	Redirección de entrada
>	Redirección de salida
>>	Agregar salida
2>	Redirección de errores

Probemos algunos ejemplos.

5.1 Guarde los detalles de la configuración de la red en un archivo llamado `output.txt`

En este ejemplo, enviaremos la salida del `ifconfig`comando al archivo en lugar de la salida estándar (pantalla).

```
ifconfig > output.txt
```

Consulte el contenido de output.txt

```
cat output.txt
```

5.2 Guarde la salida del `date`comando en el archivo 'output.txt' .

```
date > output.txt
```

Consulte el contenido de output.txt

```
cat output.txt
```

¿Notó que se sobrescribieron los contenidos anteriores de output.txt?

Cuando redirige utilizando >el contenido del archivo de destino se sobrescribe.

5.3 Agregar salida a un archivo

Ahora, probaremos la siguiente secuencia, donde usamos '>>' en lugar de '>'.

Ejecute los siguientes comandos.

```
uname -a >> newoutput.txt
date >> newoutput.txt
```

Consulte el contenido de newoutput.txt

```
cat newoutput.txt
```

Debería ver la salida de `uname`y los `date`comandos adjuntos al archivo newoutput.txt

5.4 Reproduzca el contenido del archivo 'newoutput.txt' en mayúsculas.

Puede utilizar el comando **tr** para esta traducción.

El comando **tr** no acepta nombres de archivo como argumentos. Pero acepta entrada estándar.

Entonces, redirigiremos el contenido del archivo 'newoutput.txt' a la entrada del comando tr.

```
tr "[a-z]" "[A-Z]" < newoutput.txt
```

Debería ver todas las letras mayúsculas en la salida.

Ejercicio 6 - Tuberías y filtros

La canalización de comandos es una característica del shell que nos ayuda a combinar diferentes comandos no relacionados de tal manera que la salida de un comando se envía directamente como entrada al siguiente comando. De esta manera, lo que no es posible con un solo comando puede ser posible conectando múltiples comandos.

Solo los comandos de filtrado se pueden utilizar de esta manera .

Un **comando de filtro** es un comando que puede aceptar una entrada de una entrada estándar y enviar una salida a una salida estándar.

Veamos algunos ejemplos usando algunos comandos de filtro que ya hemos discutido.

6.1 Cuente el número total de archivos en su directorio actual.

Dado que el comando **ls** no proporciona una opción para obtener un recuento, obtengamos ayuda del comando **wc** .

Al combinarlos usando la sintaxis de la canalización de comandos, obtenemos el siguiente comando.

```
ls | wc -l
```

6.2 Encuentre el uso total de espacio en disco.

El comando **df -h** proporciona el uso del disco para todos los sistemas de archivos individuales, incluido el uso total en el servidor debajo del encabezado `overlay`.

Puede obtener el uso general del disco si desea `grep`superponer la salida de`df -h`

```
df -h|grep overlay
```

6.3 Enumere los cinco archivos más grandes.

La opción -S del comando **ls** ordena los archivos de mayor a menor.

Enviaremos esta lista ordenada a través de una tubería al comando **head** .

```
ls -lS /bin | head -6
```

Debería ver la lista de los cinco archivos más grandes del directorio / bin.

Ejercicio 7 - Argumentos de la línea de comandos

Los argumentos de la línea de comandos son una forma muy conveniente de pasar entradas a un script.

Se puede acceder a los argumentos de la línea de comandos dentro del script como $ 1, $ 2, etc. $ 1 es el primer argumento, $ 2 es el segundo argumento.

7.1 Cree un script bash simple que maneje dos argumentos.

Guarde el siguiente código como wish.sh

```
#! /bin/bash

echo "Hi $1 $2"

#$1 is the first argument passed to the script

echo "$1 is your firstname"

#$2 is the second argument passed to the script
echo "$2 is your lastname"
```

Haga que el script sea ejecutable para todos.

```
chmod +x wish.sh
```

Ejecute el script con los dos argumentos como se muestra a continuación.

```
./wish.sh Ramesh Sannareddy
```

Debería ver la siguiente salida.

Hola Ramesh Sannareddy

Ramesh es tu primer nombre

Sannareddy es tu apellido

7.2 Encuentre el uso total de espacio en disco.

Creemos un script bash llamado `dirinfo.sh`que toma el nombre del directorio como argumento e imprime el número total de directorios y el número de archivos que contiene.

Usaremos el comando find con la opción -type, que listará solo archivos o directorios dependiendo del uso de `d`switch o `f`switch respectivamente.

El comando `wc -l`contará las líneas.

Guarde el siguiente código como dirinfo.sh

```
#! /bin/bash

dircount=$(find $1 -type d|wc -l)

filecount=$(find $1 -type f|wc -l)

echo "There are $dircount directories in the directory $1"

echo "There are $filecount files in the directory $1"
```

Haga que el script sea ejecutable para todos.

```
chmod +x dirinfo.sh
```

Ejecute el script con un argumento como se muestra a continuación.

```
./dirinfo.sh /tmp
```

En la salida, debería ver la cantidad de archivos y directorios en el directorio / tmp.

Ejercicios de práctica

1. Problema.

   Cree una variable llamada 'color' y almacene la cadena 'verde claro' en ella.

▶ Haga clic aquí para obtener una pista

▶ Haga clic aqui para SOLUCION

2. Problema.

   Muestra la lista de todos los archivos cuyo nombre comienza con 'b' y termina con '.log' en el directorio / var / log.

▶ Haga clic aquí para obtener una pista

▶ Haga clic aqui para SOLUCION

3. Problema.

   Muestra el recuento de todos los archivos cuyo nombre comienza con 'c' en el directorio / bin.

▶ Haga clic aquí para obtener una pista

▶ Haga clic aqui para SOLUCION

4. Problema.

   Muestra el valor de la variable 'color'.

▶ Haga clic aquí para obtener una pista

▶ Haga clic aqui para SOLUCION

5. Problema.

Almacene el valor de la variable 'color' en un archivo 'color.txt'

▶ Haga clic aquí para obtener una pista

▶ Haga clic aqui para SOLUCION

6. Problema.

Escriba un script de shell llamado `latest_warnings.sh`que imprima las últimas 5 advertencias del archivo /var/log/bootstrap.log.

▶ Haga clic aquí para obtener una pista

▶ Haga clic aqui para SOLUCION

Autores

Ramesh Sannareddy

Otros colaboradores

Rav Ahuja

Cambio de registro

Fecha (AAAA-MM-DD)	Versión	Cambiado por	Cambiar Descripción
2021-05-30	0,1	Ramesh Sannareddy	Versión inicial creada del laboratorio

Anterior Continuar