Introducción a la programación de los intérpretes de comandos

Ramses Pacheco Ortiz

21 De Febrero Del 2018

1 Introducción

Esta actividad comenzamos a ver un poco mas sobre el manejo de emacs y sus comandos,unas de las ventajas por el que utilizamos estos comandos fue el manejo de los datos, organizacion de archivos y filtrar contenido.

Primero unas de las actividades que realizamos es descargar un archivo que lleva el nombre de script.sh, en ese programa editamos el numero de la estacion escogiendo una de nuestro interes,le cambiamos el derecho a editar con el comando chmod,lo ejecutamos y se nos decargaron 12 archivos, a continuación utilizamos los comandos como less,cat,grep,diff,etc. Mas adelante hablaremos sobre estos comando.

2 Descripción De Comandos

- Cat

  El Comando cat sirve para concatenar varios archivos para posteriormente desplegarlos en pantalla. Admite como argumentos la lista de ficheros que hay que enlazar.El comando cat concatena archivos y/o los muestra como salida. Supongamos que tenemos un archivo llamado Madrigal.txt que contiene una serie de datos o texto,Desde la linea de comandos, sin usar un editor, podemos ver el contenido de este archivo con el comando cat Madrigal.txt

- Chmod

  Hay 3 atributos básicos para archivos simples: lectura, escritura y ejecutar.

  » Permiso de lectura (read) Si tienes permiso de lectura de un archivo, puedes ver su contenido.

  » Permiso de escritura (write) Si tienes permiso de escritura de un archivo, puedes modificar el archivo. Puedes agregar, sobrescribir o borrar su contenido.

» Permiso de ejecución (execute) Si el archivo tiene permiso de ejecución, entonces puedes decirle al sistema operativo que lo ejecute como si fuera un programa.

»Usando chmod para cambiar los permisos chmod (change mode) es el comando utilizado para cambiar permisos, se pueden agregar o remover permisos a uno o mas archivos con + (mas) o – (menos)

» Si quieres prevenirte de modificar un archivo importante, simplemente quita el permiso de escritura en tu "archivo" con el comando chmod

chmod -w tuArchivo

»si quieres hacer un script ejecutable, escribe:

chmod +x tuScript

- Echo

  El comando interno echo es una de las instrucciones más simples de la shell. Se encarga de repetir o desplegar en la salida estándr cualquier argumento que se le indíque(inclusive comodínes), para posteriormente saltar una línea.

  Con la opción -n permite no hacer el intro final, podemos verlo en el siguiente ejemplo: echo ejemplo.....; echo ok ejemplo.....ok

  Añadiendo el -n al primer echo tenemos la siguiente salida:

  echo -n ejemplo.....; echo ok

  ejemplo.....ok

  Por otro lado tenemos el conjunto de opciones -e y -E. Con la -e indicamos que interprete los caracteres escapados con una contrabarra, por el contrario con -E indicamos que no se interpreten (este es el comportamiento por defecto).

- grep

  El comando grep, que significa impresión de expresiones regulares globales, permanece entre los comandos más versátiles en un entorno de terminal Linux. Resulta ser un programa inmensamente poderoso que les da a los usuarios la capacidad de ordenar entradas basadas en reglas complejas, convirtiéndolo así en un enlace bastante popular a través de numerosas cadenas de comando. El comando grep se usa principalmente para buscar texto o buscar en cualquier archivo dado las líneas que contienen una coincidencia con las palabras / cadenas proporcionadas. De forma predeterminada, grep muestra las líneas que coinciden, y se puede utilizar para buscar líneas de texto que coincidan con una / varias expresiones regulares sin complicaciones, y solo muestra las líneas correspondientes,por ejemplo:

  para uscar un archivo con el nombre tom, el comando es el siguiente:

  grep tom /etc/passwd

tambien grep -i hola pp.txt considera idénticas las mayúsculas y minúscula en la cadena de búsqueda hola.

grep -v hola pp.txt lista las líneas que no contengan la cadena hola del archivo pp.txt

grep -n hola pp.txt las líneas concordantes con la cadena hola del archivo pp.txt se mostrarán acompañadas del número de línea.

- less

  Este comando es de mucha utilidad; su función es paginar texto en pantalla. Muchas veces ocurre que cuando se ejecuta algún comando, la salida del mismo aporta demasiada información como para que se pueda leer en la pantalla del monitor. Entonces se puede redireccionar esta salida a less para que permita al usuario leer sin mayores problemas, pudiendo avanzar o retroceder en el texto con las flechas de cursor del teclado. También se utiliza para visualizar archivos de texto almacenados en disco.

  El comando para llevar a cabo esta accion es: less arch1.txt

- ls

  El comando ls es muy útil para ver los archivos y directorios que tenemos dentro del directorio en el que estamos.

  Las opciones disponibles con éste comando son las siguientes:

  - ls -a

    Nos muestra los archivos y directorios dentro del directorio actual, incluyendo los archivos y directorios ocultos.

  - ls -t

    Ordena los archivos por fecha de modificación.

  - ls -X

    Ordena los archivos por extensión.

  - ls -l

    Muestra toda la información: usuario, grupo, permisos, tamaño, fecha y hora de creación.

  - ls -lh

    Muestra la misma información que ls -l pero con las unidades de tamaño en KB, MB, etc.

  - ls -R

    Muestra el contenido de todos los subdirectorios de forma recursiva.

  - ls -S

    Ordena los resultados por tamaño de archivo.

- ws

  wc (word count) es un comando utilizado en el sistema operativo que permite realizar diferentes conteos desde la entrada estándar, ya sea de palabras, caracteres o saltos de líneas.

  El programa lee la entrada estándar o una lista concatenada y genera una o más de las estadísticas siguientes: conteo de líneas, conteo de palabras, y conteo de bytes. Si se le pasa como parámetro una lista de archivos, muestra estadísticas de cada archivo individual y luego las estadísticas generales.

  Para todo archivo procesado por el comando, se despliegan 4 columnas que índica el número de líneas, palabras y caracteres(en ese orden), además del nombre del archivo procesado. Además, si se procesan varios archivos, al final se muestra la cuenta total.

  con el comando wc -c muestra solamente el número de caracteres contenidos en el archivo procesado.

  con el comando wc -w muestra solamente el numero de palabras contenidos en el archivo procesado.

  con el comando wc -l muesra solamente el número de líneas contenidas en el archivo procesado.

- Redirectores

  algunos ejemplos de estos son los siguientes:

  - ls –lR > file
    La salida estándar de la orden ls –lR (listado de archivos y subdirectorios que contiene el directorio actual de forma recursiva) se guarda en el archive file.
  - wc –l < file
    Cuenta el número de líneas del archivo file.
  - ls –l | grep vmlinux
    Redirecciona la salida de la orden ls hacia la orden grep, que mostrará únicamente las líneas que contienen la palabra vmlinux.
  - cp –vf /root /home/alumno » file
    La lista de todos los archivos copiados se escribirá en el archivo file. Si el archivo file ya existe la lista se escribirá al final del archivo, manteniendo la información previa.

3 Sintesis De Shell Script Tutorial

3.1 Introducción

Este tutorial está escrito para ayudar a las personas a comprender algunos de los conceptos básicos de la programación de scripts de shell (también conocido

como shell scripting), y con suerte para presentar algunas de las posibilidades de la programación simple pero potente disponible bajo el shell Bourne.

Un poco de historia sobre Steve Bourne escribió el shell Bourne que apareció en la versión Seventh Edition Bell Labs Research de Unix. Se han escrito muchas otras conchas; este tutorial particular se concentra en los proyectiles Bourne y Bourne Again. Otros shells incluyen Korn Shell (ksh), C Shell (csh) y variaciones como tcsh. Este tutorial no cubre esos shell.

Algunos convenciones tipográficas utilizadas en este tutorial son las palabras importantes se escribirán en cursiva cuando se mencionen por primera vez.

Las entradas de la línea de comando estarán precedidas por el signo de dólar ($). Si su mensaje es diferente, ingrese el comando: PS1 = "$" ; exportar PS1

Tenga en cuenta que para hacer que un archivo sea ejecutable, debe establecer el bit eXecutable, y para un script de shell, también debe establecerse el bit Legible:

$ chmod a + rx my - script . sh $ ./ my - script . sh

3.2 Filosofía

La programación de script de Shell tiene una mala impresión entre algunos administradores de sistemas de Unix. Esto es normalmente debido a una de dos cosas:

La velocidad a la que se ejecutará un programa interpretado en comparación con un programa C, o incluso un programa Perl interpretado.

- La velocidad a la que se ejecutará un programa interpretado en comparación con un programa C, o incluso un programa Perl interpretado.
- Dado que es fácil escribir un script de shell de tipo de trabajo por lotes simple, hay muchos scripts de shell de mala calidad.

  Una debilidad en muchos scripts de shell es líneas tales como:

  cat / tmp / myfile | grep "mystring"

  que se ejecutará mucho más rápido como:

  grep "mystring" / tmp / myfile

3.3 Una Primera Secuencia De Comandos

Primero realizamos un programa que nos imprimiera el hola mundo abriendo un archivo ne emacs desde una terminal usando el comando echo.

Despues realizamos lo mismo pero al hola mundo lo modificamos seperando con diferentes caracteres como por ejemplo: *,tab, cantidad de espacios ,etc.

3.4 Variables-parte 1

En esta seccion utilizamos varibles para introducirles un valor numerico o escrito.Esta seccion consta de 3 scripts elaborados y en cada un de ellos se realizo un cosa que lleva a otra por ejemplo de como introducir un variable para que obtenga un valor o leer un variable, a continuacion se muestra una imagencon con los scripts ejecutados.

3.5 Comodines

Los comodines no son nada nuevo si has usado Unix antes. Sin embargo, no es necesariamente obvio cómo son útiles en los scripts de shell. Esta sección es realmente solo para hacer que las viejas celdas grises piensen cómo se ven las cosas cuando estás en un guión de shell, prediciendo cuál es el efecto de usar diferentes sintaxis.

3.6 Caracteres De Escape

Ciertos personajes son importantes para el caparazón; hemos visto, por ejemplo, que el uso de caracteres de comillas dobles (") afecta la forma en que se tratan los espacios y los caracteres TAB, por ejemplo:

echo Hello World Hello World echo "Hello World" Hello World

El primero y el último "caracteres envuelven todo en un parámetro pasado para echoque el espaciado entre las dos palabras se mantenga como está. Pero el código:

echo "Hello" World "" se interpretaría como tres parámetros: "Hola "Mundo"" Entonces la salida sería Hola mundo

La mayoría de los caracteres (*, ', etc) no se interpretan (es decir, que se toman literalmente) por medio de la colocación entre comillas dobles (""). Se toman como están y se pasan al comando que se llama. Un ejemplo usando el asterisco (*) va:

$echo * caso . shtml escapar . shtml primero . shtml

funciones . shtml consejos . índice shtml . shtml ip - primer . txt raid1 + 0.txt

$echo * txt ip - primer . txt raid1 + 0.txt $ echo "*" * $ echo "* txt" * txt

En el primer ejemplo, * se expande para indicar todos los archivos en el directorio actual.

En el segundo ejemplo, * txt significa todos los archivos que terminan en txt.

En el tercero, ponemos * entre comillas dobles, y se interpreta literalmente.

En el cuarto ejemplo, lo mismo se aplica, pero hemos agregado txta la cadena.

Sin embargo, ", $, ‘, y todavía son interpretados por el shell, incluso cuando están entre comillas dobles. El carácter de barra invertida ($) se utiliza para marcar estos caracteres especiales para que el intérprete no los interprete, sino que los pase al comando que se está ejecutando (por ejemplo, echo).

3.7 Loops

La mayoría de los lenguajes tienen el concepto de bucles: si queremos repetir una tarea veinte veces, no queremos tener que escribir el código veinte veces, con un ligero cambio cada vez. Como resultado, tenemos fory whilebucles en el shell Bourne. Esto es algo menos funciones que otros lenguajes, pero nadie afirmó que la programación del shell tiene el poder de C.

En esta actividad realizamos dos script del area de **for loops** en donde los bucles iteran a través de un conjunto de valores hasta que se agote la lista y otros tres script del area de **while loops** en donde los bucles pueden ser mucho más divertidos (Dependiendo de su idea de diversión y de la frecuencia con la que sale de la casa). por ejemplo: Lo que ocurre aquí es que las instrucciones de eco y lectura se ejecutarán indefinidamente hasta que escriba "bye" cuando se le solicite.

3.8 Test

La prueba es utilizada por prácticamente todos los guiones de shell escritos. Puede que no parezca así, porque a test no se llama directamente. test es más frecuentemente llamado como. Es un enlace simbólico test, solo para hacer que los programas shell sean más legibles. También es normalmente un shell incorporado (lo que significa que el intérprete de comandos interpretará el significado test, incluso si su entorno Unix está configurado de manera diferente).

Este comando basicamente debe de estar en corchete y con ayuda de comilla espacio o asteristicos podremos llevar acabo ciertos test.

3.9 Case

Los case declaración guarda pasar por un conjunto completo de if .. then .. else declaraciones. Su sintaxis es realmente bastante simple,la case línea en sí tiene siempre el mismo formato, y significa que estamos probando el valor de la variable $INPUTSTRING$. en donde puede tomar una varibles que tu le pongas y si no se cumple decir la usuario que es una variable incorrecta que vendria siendo el case default.

3.10 Varibles Parte-2

Ya hay un conjunto de variables establecidas para usted, y la mayoría de ellas no pueden tener valores asignados. Estos pueden contener información útil, que el script puede utilizar para conocer el entorno en el que se está ejecutando. El primer conjunto de variables que veremos son 0 .. 9. La variable 0 es el nombre base del programa como se lo llamó. $1 .. $9 son los primeros 9 parámetros adicionales con los que se invocó el script.

Es importante cuando se trata de IFS en particular (pero de cualquier variable que no esté enteramente bajo su control) darse cuenta de que podría contener espacios, líneas nuevas y otros caracteres "incontrolables". Por lo tanto, es una muy buena idea usar comillas dobles a su alrededor, es decir: en old IFS="IFS"lugar de old IFS=IFS.

3.11 Variables parte -3

Como mencionamos en Variables - Parte I , las llaves alrededor de una variable evitan confusiones.

Sin embargo, eso no es todo, estos brackets de lujo tienen otro uso mucho más poderoso. Podemos tratar con problemas de variables indefinidas o nulas (en el shell, no hay mucha diferencia entre indefinido y nulo).

por ejemplo: Hay otra sintaxis, ": =", que establece la variable por defecto si no está definida:echo "Tu nombre es: $ myname: = John Doe"

Esta técnica significa que cualquier acceso posterior a la $myname variable siempre obtendrá un valor, ya sea ingresado por el usuario, o "John Doe" en caso contrario.

3.12 Programas externos.

Los programas externos a menudo se usan en scripts de shell; hay algunas órdenes internas (echo, which, y test son comúnmente incorporados), pero muchos comandos útiles son en realidad utilidades Unix, tales como tr, grep, expr y cut.

El backtick se usa para indicar que el texto adjunto se debe ejecutar como un comando. Esto es bastante simple de entender,el backtick simplemente captura el resultado estándar de cualquier comando o conjunto de comandos que decidamos ejecutar. También puede mejorar el rendimiento si desea ejecutar un comando lento o un conjunto de comandos y analizar varios bits de su salida.

3.13 Funciones

Una característica que a menudo se pasa por alto de la programación de guiones de shell de Bourne es que puede escribir fácilmente funciones para

usar en su secuencia de comandos. Esto generalmente se hace de una de dos maneras; con un script simple, la función simplemente se declara en el mismo archivo como se llama. Sin embargo, al escribir un conjunto de secuencias de comandos, a menudo es más fácil escribir una "biblioteca" de funciones útiles, y el origen de ese archivo al inicio de los otros scripts que utilizan las funciones.

pero en esta ocasion utilizaremos la primera manera,la definición de una función es tradicionalmente que devuelve un solo valor y no genera nada. Un procedimiento, por otro lado, no devuelve un valor, pero puede producir un resultado. Una función de shell no puede hacer ni una ni la otra ni ambas.

Una función puede devolver un valor en una de cuatro formas diferentes:

- Cambiar el estado de una variable o variables
- Use el exitcomando para finalizar el script de shell
- Utilice el returncomando para finalizar la función y devolver el valor proporcionado a la sección de llamada del script de shell
- echo output to stdout, que será capturado por la persona que llama al igual que c = ‘expr $ a + $ b‘ está atrapado

  que exit detiene el programa y return devuelve el control a la persona que llama. La diferencia es que una función de shell no puede cambiar sus parámetros, aunque puede cambiar los parámetros globales.

  Las funciones se leen, pero básicamente se ignoran hasta que realmente se llaman.

  Los $parámetros se cambian dentro de la función para reflejar cómo se llamó a la función. x sin embargo, la variable es efectivamente una variable global, la myfunc cambió y ese cambio sigue siendo efectivo cuando el control vuelve al guión principal.

3.14 Consejos y Sugerencias

Unix está lleno de utilidades de texto manipulación, algunos de los más poderosos de los cuales ahora vamos a discutir en esta sección de este tutorial. El significado de esto, es que virtualmente todo bajo Unix es texto. Prácticamente cualquier cosa que se pueda imaginar está controlado por un archivo de texto, o mediante una interfaz de línea de comandos (CLI). Lo único que no puede automatizar con un script de shell es una utilidad o función solo GUI. ¡Y en Unix, no hay muchos de ellos!

Los códigos de salida son un número entre 0 y 255, que es devuelto por cualquier comando de Unix cuando devuelve el control a su proceso principal. Se pueden usar otros números, pero estos se tratan con el módulo 256, por lo que exit -10es equivalente a exit 246, y exit 257es equivalente a exit 1. Estos pueden usarse dentro de un script de shell para cambiar el flujo de ejecución dependiendo del éxito o falla de los comandos ejecutados.

3.15 Referencia rápida

Esta es una guía de referencia rápida sobre el significado de algunos de los comandos y códigos menos fáciles de adivinar de los scripts de shell. Por su naturaleza, también son bastante difíciles de encontrar usando los motores de búsqueda. Estos ejemplos incluyen gestión de procesos , argumentos de scripts de shelltest y condiciones de script de shell .

3.16 Shell Interactivo

Aquí hay algunas sugerencias rápidas para usar el shell UNIX o Linux de forma interactiva. Personalmente recomiendo el shell bash para el uso más interactivo; está disponible en casi todos los sabores * nix, y es muy agradable de usar como shell de inicio de sesión. Sin embargo, el shell raíz siempre debe ser / bin / sh, ya sea que apunte a bash o al shell Bourne.

bash tiene algunas herramientas de búsqueda de historia muy prácticas; las teclas de flecha hacia arriba y hacia abajo se desplazarán por el historial de comandos anteriores. Más útilmente, Ctrl + r hará una búsqueda inversa, haciendo coincidir cualquier parte de la línea de comando. Presione ESC y el comando seleccionado se pegará en el shell actual para que pueda editarlo según sea necesario.

4 Bibliografia

- Shellscript.sh. (2018). Shell Scripting Tutorial. [online] Available at: https://www.shellscript.sh/functions.html [recuperado 5 Mar. 2018].
- Atmospheric Soundings. (2018). Weather.uwyo.edu. Retrieved 5 March 2018, from http://weather.uwyo.edu/upperair/sounding.html
- Comando Linux Less |. (2018). Servidores Dedicados Administrados Tel. 93 803 35 69 || M. 636 27 67 86. Retrieved 5 March 2018, from https://www.servidoresadmin.com/less/

5 Apendice

1.-¿Qué fue lo que más te llamó la atención en esta actividad? primero que nada lo larga y fastidiosa que se me hizo esta actividad, pero tambien aprendi mucho a como realizar los scripts

2.-¿Qué consideras que aprendiste? como realizar los scripst y algunos comando para facilitar su realizacion.

3.-¿Cuáles fueron las cosas que más se te dificultaron? Entender le tutorai len general y co el poco tiempo que tenia para terminar la actividad.

4.-¿Cómo se podría mejorar en esta actividad?

Tal vez que nos diera un poco mas de tiempo y un poco mas de explicacion

5.-¿En general, cómo te sentiste al realizar en esta actividad?

Mesinte un poco presionado ya que no organize bien mi tiempo.