Práctica 4: comandos para manipulación de archivos

Dr. Hermes Francisco Montes Casiano

1. Introducción

El shell de GNU/Linux es una utilidad interactiva especial. Proporciona una forma para que los usuarios inicien programas, administren archivos en el sistema de archivos y administren procesos que se ejecutan en el sistema Linux. El núcleo del shell es el símbolo del sistema. El símbolo del sistema es la parte interactiva del shell. Le permite ingresar comandos de texto, interpretar los comandos y luego ejecutar los comandos en el núcleo.

El shell contiene un conjunto de comandos internos que utiliza para controlar cosas como copiar archivos, mover archivos, renombrar archivos, mostrar los programas que se ejecutan actualmente en el sistema y detener los programas que se ejecutan en el sistema. Además de los comandos internos, el shell también le permite ingresar el nombre de un programa en el símbolo del sistema. El shell pasa el nombre del programa al núcleo para iniciarlo.

2. Comandos para manipulación de archivos

Desde la terminal también es posible manipular archivos mediante el uso de comandos para: procesamiento de datos, manipulación de contenido, utilidades para la administración y configuración del equipo de trabajo, etc; en la tabla 1 puede consultar algunos:

Tabla 1: Comandos para manipulación de archivos

Comando	**Descripción**	**Uso**
stat	Proporciona un resumen completo del estado de un archivo en el sistema de archivos	$ stat archivo
file	Tiene la capacidad de mirar dentro de un archivo y determinar qué tipo de archivo es	$ file archivo
cp	Copia archivos y carpetas de un origen a un destino	$ cp origen destino
mv	Se puede utilizar para cambiar el nombre de un archivo o directorio, pero también para moverlos de una ubicación a otra	$ mv origen destino
rm	Elimina el o los archivos o directorios indicados	$ rm archivoA
mkdir	Crea un nuevo directorio la ruta indicada	$ mkdir directorio

Continua en la siguiente página

Tabla 1 – continuación de la página anterior

Comando	Descripción	Uso
rmdir	Elimina el directorio indicado siempre y cuando esté vacío	$ rmdir directorio

Actividad 1: Ejecución de comandos

1. Abra una terminal
2. Ejecute cada uno de los comandos que se listan en la tabla 1.
3. Para obtener mayor detalle de la ejecución, conocer cuáles son las opciones y parámetros de cada comando, utilice el comando *man*.

```
1 $ man stat
2 ...
3 stat [-FLnq] [-f format | -l | -r | -s | -x] [-t
    timefmt] [file ...]
4 readlink [-n] [file ...]
5 ...
```

Ejemplo 1: Consultar el manual

4. Pruebe cada comando con al menos tres opciones y describa el resultado.

3. Redirección de flujos

En muchas ocasiones se requiere la funcionalidad proporcionada por un comando, con el objetivo de explotar la información que provee y no para simplemente imprimirlo en la salida estándar.

Si se utiliza el carácter > después de cualquier comando que escribe el resultado de su ejecución en la salida estándar, la salida de ese comando se escribirá en un archivo (ver **??**), en lugar de su terminal. Si se utiliza el carácter >>, la salida del comando se concatenará con el contenido existente en el archivo.

```
1 $ echo hola mundo del shell scripting > salida.txt
2 $ cat salida.txt
3 hola mundo del shell scripting
4 $
```

Ejemplo 2: Redirección de flujo, creando el archivo o eliminando el contenido existente

```
$ echo siguiente linea >> users
$ cat users
hola mundo del shell scripting
siguiente linea
$
```

Ejemplo 3: Redirección de flujo, creando el archivo o concatenando la salida al contenido existente

Así como la salida de un comando se puede redirigir a un archivo, la entrada de un comando se puede redirigir desde un archivo; para redirigir el flujo de entrada se usa el carácter menor que <. Los comandos que normalmente toman su entrada de la entrada estandar pueden redirigir su entrada desde un archivo.

```
$ wc -l salida.txt
2 salida.txt
$
```

Ejemplo 4: Contar el número de líneas que tiene un archivo

```
$ wc -l < salida.txt
2
$
```

Ejemplo 5: Contar el número de líneas que tiene un archivo mediante la redirección del flujo de entrada

Pregunta 1

¿Cuál es la diferencia en la salida de la ejecución de los comandos de los ejemplos 4 y 7 y por qué?

Se puede ejecutar un programa interactivo dentro de un script de shell sin la acción del usuario al proporcionar la entrada requerida para el programa interactivo o el script de shell interactivo (*document here*).

```
$ comando << delimitador
documento
delimitador
$
```

Ejemplo 6: Sintaxis del uso de *document here*

Aquí, el shell interpreta el operador « como una instrucción para leer la entrada hasta que encuentra una línea que contiene el delimitador especificado. Todas las líneas de entrada hasta la línea que contiene el delimitador se introducen en la entrada estándar del comando.

El delimitador le dice al shell que el documento aquí se ha completado. Sin él, el shell continúa leyendo la entrada para siempre. El delimitador debe ser una sola palabra que no contenga espacios ni tabuladores.

```
$wc -l << EOF
Voz de la guitarra mía al despertar la mañana
Quiere cantar su alegría a mi tierra mexicana

Yo le canto a tus volcanes, a tus praderas y flores
Que son como talismanes del amor de mis amores
EOF
5
$
```

Ejemplo 7: Entrada del comando wc -l para contar el número total de líneas

Actividad 2: Uso de *document here* en un script

1. Abra el editor de texto *Visual Studio Code*.
2. Con base en el ejemplo 7 escriba un script que haga uso de la redirección del flujo de entrada mediante *here document*.
3. Abra una terminal y cambie el directorio de trabajo a la ubicación del script.
4. Ejecute el script que acaba de crear. Recuerde que si un ejecutable no se encuentra en el *PATH*, para ejecutarlo debe colocar el prefijo *./* al nombre del binario ejecutable.
5. En caso de que el script no se pueda ejecutar, realice las acciones necesarias para poder ejecutarlo. Ver el comando *chmod*.
6. Modifique el script del punto 2 para que el script cree un archivo con el nombre *archivo_prueba.txt* y le guarde el contenido *Contenido agregado sin la intervención de una persona* y lo guarde en el mismo directorio en el que se encuentra el script.

4. Tuberías

Hay muchas ocasiones en las que la ejecución de un comando de forma aislada no resuelve una problemática, y lo que se necesita es conectar la salida de un comando con la entrada de otro, a este mecanismo se le conoce como tubería. El símbolo de una tubería es es el operador de barra (|): comando1 | comando2.

Una tubería proporciona una forma de vincular comandos para proporcionar resultados

mas detallados. Sin embargo, no piense en la tubería como un mecanismo en el que ejecuta dos comando, uno tras otro. *Linux* ejecuta los dos comandos al mismo tiempo conectándolos internamente en el sistema.

```
1 $ apt-cache searh postgres | sort | more
2 akonadi-backend-postgresql - PostgreSQL storage ...
3 algol68g - Implementation of Algol 68 as ...
4 aolserver4-nspostgres - AOLserver 4 ...
5 apgdiff - Another PostgreSQL Diff Tool
6 autopostgresqlbackup - Automated tool to ..
7 backupninja - lightweight, extensible ..
8 bacula-common-pgsql - network backup ..
9 bacula-director-pgsql - network backup service ...
10 bandwidthd-pgsql - Tracks usage of TCP/IP ...
11 bareos-database-postgresql - Backup Archiving ...
12 --More--
13 $
```

Ejemplo 8: Interconectando comandos usando tuberias

En el ejemplo 8 se muestra el comanto *apt-cache*, que en las distribuciones basadas en debian permite hacer una búsqueda de los paquetes con base en un nobre o descripción. Por su parte, el comando *sort* ordena alfabéticamente la salida del comando *apt-cache*. El comando *more* pagina el resultado ya ordenado del comando *sort*.

Actividad 3: Uso de *document here* en un script

1. Abra una terminal y ejecute el comando del ejemplo 8.

   En caso de que tenga instalada una distribución de linux que no sea base debian cambie el comando apt-cache por el equivalente en su distribución, por ejemplo: *$ yum search postgres*, en una distribución base *redhat* o *fedora*.

2. Cambie las tuberías por una redirección de flujo y ejecute el comando.

3. ¿El resultado obtenido es el mismo? justifique su respuesta