Cap 4: Usando ARCHIVOS

Descriptor de archivo estándar

Hay tres archivos que se abren para cada proceso cuando se crea un proceso, y siempre se les asigna el siguiente número de descriptor de archivo.

0	**entrada estándar**
1	**salida estándar**
2	**error estándar**

Estos archivos suelen estar asociados con la terminal del usuario. Por lo tanto, la entrada estándar se puede utilizar para leer lo que escribe el usuario y la salida estándar se puede utilizar para enviar la salida al usuario. El error estándar se usa generalmente para mensajes de error para que puedan mantenerse separados de la salida normal.

Sintaxis de redirección

El **shell** proporciona las siguientes sintaxis para redirigir la entrada y salida de los comandos. Cada una de estas sintaxis se discutirá encada de las secciones que siguen.

>archivo escribe la salida estándar al **archivo**

>>archivo agrega la salida estándar al **archivo**

>&m escribe la salida estándar en el descriptor de archivo **m**

>&- cerrar salida estándar

<archivo lee la entrada estándar del **archivo**

<&m leer la entrada estándar del descriptor de archivo **m**

<&- cerrar entrada estándar

<<palabra lectura de la entrada estándar de un "documento aquí"

Cualquiera de los símbolos de redirección de archivos puede tener un prefijo con un descriptor de archivo para especificar un archivo que se utilizará en lugar del archivo predeterminado para esa sintaxis.

2>archivo

También se puede preceder cualquiera de los símbolos de las sintaxis de redirección de archivos con el descriptor de archivo para el archivo predeterminado para esa sintaxis.

>archivo ⇔ **1>archivo**

>>archivo ⇔ **1>>archivo**

>&n ⇔ **1>&n**

>&- ⇔ **1>&-**

<archivo ⇔ **0<archivo**

<&n ⇔ **0<&n**

<&- ⇔ **0<&n**

<<palabra ⇔ **0<<palabra**

Un descriptor de archivo debe ser un número en el rango de 0 a 9. Los descriptores de archivo 0,1 y 2 son los descriptores de archivo estándar descritos en la sección anterior: "Abrir archivos" más adelante en este capítulo, se describe cómo abrir nuevos archivos y asignarles los descriptores de archivo.

La sintaxis de redirección de archivos se puede colocar en cualquier lugar de la línea de comando, pero en la práctica, casi siempre se coloca al final de la línea de comando.

comando parámetro1 . . . parámetron >/dev/null

La sintaxis de redirección de archivos se evalúa de izquierda a derecha; por lo tanto, si una notación depende de otra, debe asegurarse de que se especifiquen en el orden correcto. Por ejemplo, si desea descartar tanto la salida estándar como el error estándar, puede escribir un comando como este:

comando >/dev/null 2>&1

Este comando significa que se redirige la salida estándar a **/dev /null**, y luego se redirige el error estándar al mismo archivo de la salida estándar.

Si invirtiera inadvertidamente estas dos especificaciones, como en el siguiente ejemplo, no obtendría lo que esperaba. La siguiente declaración dice redirigir el error estándar al mismo archivo que la salida estándar, probablemente la terminal, y luego redirigir la salida estándar a **/dev/null**. Es decir, el error estándar se redirige a donde estaba la salida estándar antes de que se redirigiera la salida estándar.

comando 2>&1>/dev/null #Probablemente equivocada

El **shell** realiza la subtitulación de variables antes de que se apliquen estas sintaxis; por lo tanto, los nombres de los archivos y los descriptores de archivo se pueden especificar en variables.

STDOUT=/dev/null

comando >$STDOUT

Sintaxis de redirección de la salida

La salida estándar de un comando normalmente está conectada a la terminal, pero se puede redirigir a un archivo como se muestra a continuación.

comando >archivo

Si el archivo ya existe, se sobrescribirá, de lo contrario, se creará el archivo. Cuando el archivo es sobrescrito, su longitud se truncará a cero y el archivo se reescribirá; Dado que el archivo no se vuelve a crear, los atributos del archivo, como los permisos de acceso y el propietario, no se modifican.

La salida estándar de un comando se puede agregar a un archivo usando la siguiente sintaxis.

comando >>archivo

De manera similar, el error estándar se puede redirigir como se muestra en el siguiente ejemplo.

comando 2>archivo

comando 2>>archivo

La siguiente sintaxis se puede utilizar para asociar un archivo con otro, de modo que siempre que escriba en el descriptor de archivo **n**, la información se escriba realmente en el descriptor de archivo **m**. Si se omite **n**, la salida estándar se escribe en el descriptor de archivo **m**.

comando n>&m

Esto se usa comúnmente para escribir la salida en un error estándar y viceversa. Por ejemplo, la cámara de **echo** y escribe su salida en la salida estándar. Si desea escribir un mensaje en el error estándar, puede redirigir la salida estándar del comando **echo** al error estándar de la siguiente manera.

echo “Error message. “ 1>&2

Observe que el **1** en la sintaxis de redirección es el descriptor de archivo para la salida estándar. Dado que la salida estándar es la predeterminada para esta sintaxis de redirección, especificar el **1** es opcional.

Sintaxis de redirección de la entrada

La entrada estándar de un comando se puede redirigir para que el comando lea información de un archivo que no sea el terminal. Por ejemplo:

comando < archivo

También puede redirigir la entrada estándar de un comando para leer desde un archivo en línea usando la siguiente sintaxis. Este tipo de redirección se denomina lectura de un documento aquí y se trata con más detalle en la sección "Documentos aquí", más adelante en este capítulo.

comando << palabra

. . .

palabra

La siguiente sintaxis se puede utilizar para asociar un archivo con otro, de modo que siempre que lea desde el descriptor de archivo **n**, la información realmente se lea desde el descriptor de archivo **m**. Si se omite **n**, la entrada estándar se lee del descriptor de archivo **m**.

comando n<&m

Redirigiendo sentencias de Shell

Las sintaxis de redirección de archivos se utilizan para redirigir la entrada estándar, la salida y el error estándar de comandos simples. También puede utilizar estas sintaxis para redirigir los archivos estándar de las declaraciones de **shell**. Sin embargo, en la mayoría de los **shells**, el uso de la redirección de archivos con una sentencia de **shell** hará que esa sentencia se ejecute en un **subshell**. Esto, a su vez, tiene efectos que pueden resultar indeseables. La sección "Lectura de archivos" analiza estos efectos secundarios y algunas alternativas con más detalle.

Los siguientes ejemplos muestran cómo pueden utilizar los operadores de redirección para redirigir la entrada estándar, la salida y el error estándar de las sentencias de **shell.**

```
{
  comando
 . . .
} <stdin >stdout  2>stderr
```

```
if  lista_de comandos
then
  comando
 . . .
fi <stdin >stdout  2>stderr

for variable  [ en   lista_de palabras ]
do
  comando
 . . .
done <stdin >stdout  2>stderr

while  lista_comandos
do
  comando
 . . .
done <stdin >stdout  2>stderr
```

```
case cadena in
  pattern )   lista_comandos;;
 . . .
esac <stdin >stdout  2>stderr
```

También puede conectar sentencias de **shell** en una tubería para que su entrada o salida estándar se lea o escriba en una tubería. El siguiente ejemplo muestra un bucle **while** en el que tanto su entrada como su salida estándar están conectadas a una tubería.

```
comando  |
while lista_comando
do
    comando
   . . .
done  | comando
```

Abriendo los archivos

El comando **exec** se usa normalmente para ejecutar un comando en el **shell** actual sin crear un nuevo proceso. Sin embargo, si la línea de comando es pasada al comando **exec** y esta no contiene un comando, pero sí contiene la sintaxis de redirección de archivos, el comando **exec** aplicará la redirección de archivos al proceso de **shell** actual. Esto le permite redirigir la entrada estándar, la salida y el error estándar del script de **shell** actual independientemente de dónde estaban conectados cuando se invocó el shell. Por ejemplo:

```
exec <archivo          # Leer stdin del archivo
exec 1>&2              # Escribir stdout en stderr
exec >/dev/null 2>&1   # descartar stdout y stderr
exec >>file            # Agregar stdout al archivo
```

Si bien cada uno de los archivos estándar se puede redirigir mediante el comando **exec**, el comando **exec** se usa con mayor frecuencia para redirigir la entrada estándar de un script en el **shel**l. Esto se discutirá con más detalle en **"Lectura de archivos"** más adelante en este capítulo.

El comando **exec** también se puede utilizar para abrir archivos arbitrarios. Para abrir un **archivo** y escribirle, puede utilizar una de las siguientes sentencias. Como era de esperar, la primera instrucción abrirá un **archivo** para que la información se agregue al final del **archivo**. En ambos casos, el **archivo** abierto se le asigna el descriptor de archivo **n**.

exec n>archivo

exec n>> archivo

El siguiente ejemplo abre un **archivo** para lectura y asigna al **archivo** al descriptor de **archivo n.**

exec n<archivo

Del mismo modo, puede abrir un documento aquí para leerlo como se muestra en el siguiente ejemplo. De nuevo, el **archivo** abierto se asigna al descriptor de **archivo n.**

exec n<< palabra

. . .

Palabra

Escribir en archivos

Una vez que se ha abierto un archivo para escritura, se puede redirigir la salida de cualquier comando para escribir en ese archivo. El comando **echo** se usa con frecuencia para escribir datos desde un script de **shell** en un archivo.

Dado que la salida del comando **echo** normalmente se escribe en esta salida estándar, se puede redirigir su salida a un archivo arbitrario como se muestra a continuación.

echo “Mensaje” 1>&n

Este ejemplo asume que el descriptor **n** se abrió para escritura, como se muestra en la sección anterior.

Cuando se escribe en un archivo, normalmente no es necesario abrir el archivo y luego escribir en el archivo a través de un descriptor de archivo, como se describe arriba. En cambio, es más común simplemente redirigir la salida de uno o más comandos para escribir directamente en el archivo como se muestra en el siguiente ejemplo. El primer comando sobrescribe el archivo y el comando posterior agrega información al archivo.

comando > archivo

comando >> archivo

comando >> archivo

. . .

Leyendo Archivos

En el siguiente ejemplo, la entrada estándar se fusiona temporalmente con el descriptor de archivo **3**. Una vez que se completa el ciclo, el descriptor de archivo **3** se mueve de nuevo a la entrada estándar y el descriptor de archivo **3** se cierra.

```
exec  3<&0  < file

while read LINE

do

. . .

done

exec 0<&3  3<&-
```

A menos que esté utilizando una versión muy antigua del **shell,** también puede redirigir la entrada estándar del comando de lectura para leer directamente desde un archivo abierto. En el siguiente ejemplo, el comando **exec** se usa para abrir un archivo para leerlo y asignarlo al descriptor de archivo **3.** Luego, la entrada estándar del comando de lectura se redirige para leer desde el descriptor de archivo **3.** Por lo tanto, la entrada estándar del script de shell ni la entrada estándar del bucle **while** necesita ser redirigida.

```
exec  3<&0 < archivo
while read LINE
do
. . .
done
exec 0<&3   3<&-
```

Cerrar los archivos

Las siguientes sintaxis se pueden utilizar para cerrar la salida y la entrada estándar respectivamente.

exec >&-

exec <&-

Las siguientes dos sintaxis son equivalentes y se pueden usar para cerrar cualquier descriptor de archivo; simplemente reemplace **n** con el descriptor de archivo que desea cerrar.

exec n>&-

exec n<&-

Tenga cuidado al cerrar los archivos estándar de comandos que está a punto de ejecutar. Cerrar un archivo no es lo mismo que redirigirlo a /dev /null. Por ejemplo, si un comando escribe en un descriptor de archivo cerrado, esto es un error. Si ese comando no está diseñado para manejar el error, puede abortar. Si el comando escribe en /dev /null, la salida simplemente se descarta. Si no desea la salida de un comando, debe redirigir su salida estándar a /dev /null.

comando >/dev/null

Truncar un archivo

Se pueden usar los siguientes dos comandos para restablecer un archivo de modo que su longitud sea cero, o si el archivo no sale, para crear el archivo. Si el archivo ya existe, los atributos del archivo, como sus permisos de acceso, no se cambiarán.

>archivo

: >archivo

La notación **<archivo** está permitida sin un comando para este propósito. Algunas personas prefieren usar esta notación junto con el comando nulo **(:)**. Son equivalentes y ambos se utilizan comúnmente.

Estos comandos equivalen a ejecutar el siguiente comando:

cat /dev/null >file

Aquí Documentos

//Estos comandos equivalen a ejecutar el siguiente comando:

El shell proporciona un mecanismo que le permite almacenar la entrada estándar para un comando en el script de shell junto con el comando, lo que permite que el script de shell sea autónomo. Este mecanismo se llama documento aquí, presumiblemente porque los datos están "aquí" en la secuencia de comandos, y los datos generalmente se usan para información que podría denominarse documento. Cuando se ejecuta el comando, los datos del documento aquí se redireccionan a la entrada estándar del comando.

La sintaxis para redirigir la entrada estándar desde un documento aquí es:

<< [-] palabra

El símbolo << se utiliza para redirigir la entrada estándar del documento aquí. Este símbolo va seguido de una palabra que, cuando se vuelve a ver en una sola línea, marca el final del documento aquí. La palabra puede ser cualquier secuencia de caracteres que no estén en blanco. Por ejemplo:

cat << END

Este es un documento aquí.

END

Esto hace que las líneas entre <<END y la cadena delimitadora, END, se escriban en la entrada estándar del comando cat.

La sustitución de variables y comandos se realiza en este documento antes de que se escriba en la entrada estándar del comando. Se pueden usar comillas en el documento aquí para evitar esta sustitución, y las comillas que son parte del documento deben citarse ellas mismas o serán eliminadas. Puede evitar la sustitución de variables y comandos en todo el documento aquí citando la cadena delimitadora. Esto evita la necesidad de citar signos de dólar individuales y las comillas que no desea que se evalúen en el documento aquí. Por ejemplo:

```
comando <<\END          comando <<'END'
. . .                   . . .
END                     END
```

Si el símbolo << va seguido de un guión, <<-, los tabuladores iniciales, *pero no los espacios*, se eliminarán de las líneas del documento aquí y de la cadena delimitadora. Sin usar el guión, cada línea del documento aquí y la cadena delimitadora deberían comenzar en el margen izquierdo. En el siguiente ejemplo, se permite que el documento aquí se ajuste a la identificación del código circundante.

```
if …
then
    comando <<-END
    Este es un documento aquí.
    END
fi
```

El siguiente ejemplo muestra cómo se puede almacenar un mensaje grande en un documento aquí e imprimir el error usando el comando cat. Este ejemplo utiliza una función de shell denominada **Uso** para imprimir el mensaje.

```
Uso() {
        cat 1>&2  <<-EOF
        Uso: $0 [ - opciones ] [ etc. ]
         . . .
EOF  }
```

Para usar un documento aquí al mismo tiempo que otra redirección de archivos, todos los operadores de redirección pertenecen a la línea de comando, no siguiendo el documento aquí. Por ejemplo:

comando <<EOF >*stdout* 2>*stderr*

…

EOF

o:

comando1 <<EOF | comando2

…

EOF

Un documento he aquí también se puede usar para pasar información a un comando para que se ejecute sin leer desde la terminal. El siguiente ejemplo muestra un conjunto de comandos de auto comando simple que hará que el editor **ed** invierta las líneas en un archivo y luego guarde el archivo.

ed – archivo <<-!

g/^/m0

w

q

!

Tenga en cuenta que la palabra utilizada para marcar el final del documento aquí en este ejemplo es el carácter de exclamación. Recuerde que la palabra utilizada para marcar el final del documento aquí puede ser cualquier secuencia de caracteres que no estén en blanco.

EJEMPLOS DE REDIRECCIÓN DE ARCHIVOS

Cada comando en una secuencia de comandos debe especificar su propia redirección de archivos. En el siguiente ejemplo, la redirección del error estándar de un comando no afecta el error estándar del otro comando.

```
comando 2>archivoError1 |comando 2>archivoError2
```

No utilice el mismo archivo para la entrada y la salida estándar. Recuerde, el operador de redirección de salida (>) hace que el archivo se sobrescriba. Como se muestra en el siguiente ejemplo, el archivo se destruye antes de que el comando sed tenga la oportunidad de leer su contenido.

```
$ echo “foo” > archivo
$ sed ‘s/foo/bar/’ < archivo > archivo
$ cat archivo
$
```

El siguiente ejemplo es un extracto de un script de shell utilizado para imprimir archivos en una de varias impresoras en una red. Dependiendo de qué impresora esté disponible, una de las tres primeras líneas no estará comentada, lo que a su vez hará que se seleccione uno de los casos. El comando **pr** escribe los archivos enumerados en la línea de comando ($*) en su entrada estándar de la declaración del **case** y también es la entrada estándar de todos los comandos ejecutados dentro de la declaración del **case**. Cuando se ejecuta uno de los casos, el comando **rsh** copiará su entrada estándar al host remoto donde se convierte en la entrada estándar del comando **lp**. Finalmente, el comando **lp** imprime su entrada estándar en el dispositivo apropiado.

```
DEVICE=lj1
# DEVICE=lj2
# DEVICE=lj3
Pr $* |
      case  $DEVICE in
            lj1 ) rsh host1 lp -dlj1 ;;
            lj2 ) rsh host2 lp -dlj2 ;;
            lj3 ) rsh host3 lp -dlj3 ;;
      esac
```

El siguiente ejemplo muestra cómo escribir información constante en la entrada estándar de un comando y luego reanudar la lectura desde la entrada estándar original. Las llaves se utilizan para recopilar la salida estándar de varios comandos y canalizarla a la entrada estándar de otro comando. El comando **echo** se usa para escribir las cadenas constantes y el comando **cat** se usa para copiar la entrada estándar a la salida estándar después de las cadenas constantes.

```
{
  echo “line 1”
  echo “line 2”
  echo “line 3”
  cat  -
} | comando
```

El siguiente ejemplo muestra bucles **while** anidados. cada uno con su propia redirección de entrada estándar. El ciclo externo lee una lista de nombres de archivos, mientras que el ciclo interno lee cada archivo una línea a la vez.

```
while   read   FILENAME
do
    while read LINE
    do
       …
   done < $FILENAME
done < archivo
```

Para leer desde la terminal mientras la entrada estándar se redirige a otro archivo, puede leer el archivo especial /dev/tty.

```
while   read   FILENAME
 do
    echo " Ud. quiere purgar $FILENAME?"
    read ANSWER < /dev/tty
   …
done < archivo
```

Bibliografía:

Bruce Blinn

Portable Shell programming

An extensive Collection of Bourne Shell examples

Prentice Hall PTR, Upper Saddle River, New Jersey 07458 1996.