Universidad Nacional de La Matanza
***Ingeniería en Informática*-Taller de GNU/Linux**

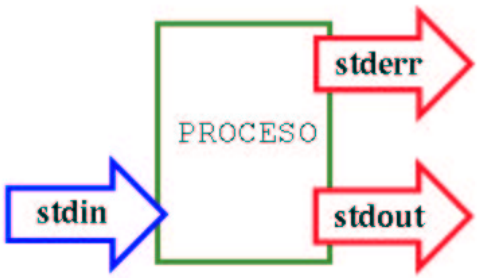
TP N° 4
Redirección de entrada y salida estándar

Objetivos:

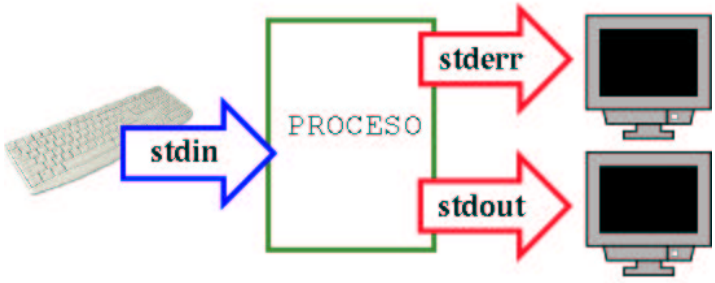
−Conceptos de entrada y salida estándar.
−Utilizar los operadores ">",">>","<".
−Programas filtros.

Entrada y salida estándar

Muchos comandos UNIX toman su entrada de algo conocido como entrada estándar y envían su salida a la salida estándar (a menudo abreviado como "stdin" y "stdout"). Además existe una salida especial para los mensajes de error de cada programa (stderr).



El intérprete de comandos configura el sistema de forma que la entrada estándar es el teclado y la salida la pantalla.

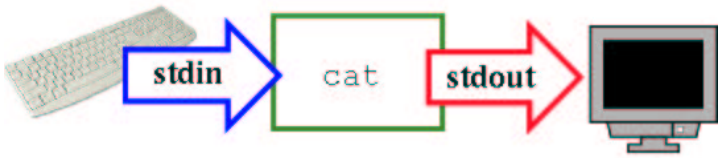


Veamos un ejemplo con el comando **`cat`**. Normalmente **`cat`** lee datos de los archivos cuyos nombres se pasan como argumentos en la línea de comandos y envía estos datos directamente a la salida estándar. Luego, usando el comando

```
/home/larry/papers# cat history-final masters-thesis
```

mostrará por pantalla el contenido del archivo history−final seguido por masters−thesis.
Si no se le pasan nombres de archivos a **`cat`** como parámetros, leerá datos de **stdin** y los enviara a **stdout**. Veamos un ejemplo.

```
/home/larry/papers# cat
Hello there.
Hello there.
Bye.
Bye.
   [ctrl-D]
/home/larry/papers#
```



Como se puede ver, cada línea que el usuario teclea (impresa en itálica) es inmediatamente reenviada al monitor por cat. Cuando se esta leyendo de la entrada estándar, los comandos reconocen el fin de la entrada de datos cuando reciben el carácter EOT (end−of−text, fin de texto). Normalmente es generado con la combinación ***[ctrl−D]***.

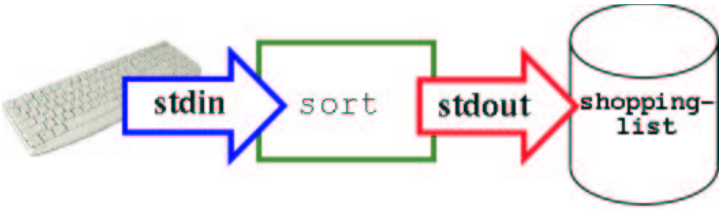
Veamos otro ejemplo. El comando **sort** toma como entrada líneas de texto (de nuevo leerá desde **stdin** si no se le proporcionan nombres de archivos en la línea de comandos), y devuelve la salida ordenada a **stdout**. Pruebe lo siguiente:

```
/home/larry/papers# sort
peras
manzanas
bananas
    [ctrl-D]
bananas
manzanas
peras
/home/larry/papers#
```

Redireccionando la salida

Ahora, supongamos que queremos que la salida de sort vaya a un archivo para poder salvar la lista ordenada de salida. El intérprete de comandos nos permite redireccionar la salida estándar a un archivo usando el símbolo ">". Veamos como funciona.

```
/home/larry/papers# sort > shopping-list
peras
manzanas
bananas
    [ctrl-D]
/home/larry/papers#
```

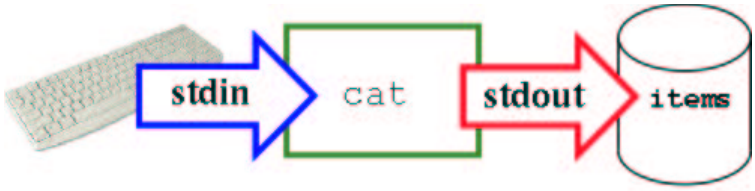


Como puede ver, el resultado de sort no se muestra por pantalla, en su lugar es salvado en el archivo **shopping−list**. Echemos un vistazo al archivo.

```
/home/larry/papers# cat shopping-list
bananas
manzanas
peras
/home/larry/papers#
```

Ya podemos ordenar la lista de la compra y además guardarla.
Creemos ahora otro listado desordenado de nuestras futuras compras en el shopping:

```
/home/larry/papers# cat > items
corbata
anteojos
bufanda
    [ctrl-D]
/home/larry/papers#
```



Al no especificar al **cat** un nombre de archivo, tomará la entrada de la entrada estándar (teclado) y la salida se redirecciona a un archivo llamado **items**.

Redireccionando la entrada

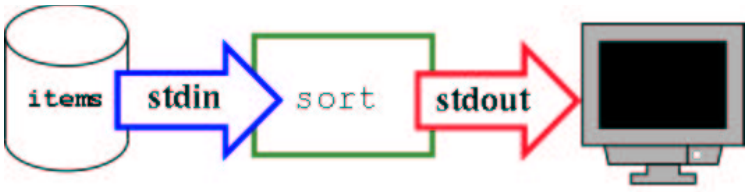
Ya tenemos guardada nuestra lista de compra desordenada original en el archivo **items**. Una forma de ordenar la información y salvarla en un archivo podría ser darle a **sort** el nombre del archivo a leer en lugar de la entrada estándar y redireccionar la salida estándar como hicimos arriba.

```
/home/larry/papers# sort items > shopping-list
/home/larry/papers# cat shopping-list
anteojos
bufanda
corbata
/home/larry/papers#
```

(archivo argumento; stdout=archivo)

Hay otra forma de hacer esto. No solo puede ser redireccionada la salida estándar, también puede ser redireccionada la entrada estándar usando el símbolo "<".

```
/home/larry/papers# sort < items
anteojos
bufanda
corbata
/home/larry/papers#
```



Técnicamente, **`sort < items`** es equivalente a **`sort items`**, pero nos permite demostrar que **`sort < items`** se comporta como si los datos del archivo fueran tecleados por la entrada estándar. El intérprete de comandos es quien maneja las redirecciones. **sort** no recibe el nombre del fichero (items) a leer, desde el punto de vista de sort, esta leyendo datos de la entrada estándar como si fueran tecleados desde el teclado.

Esto introduce el concepto de filtro:

"Un filtro es un programa que lee datos de la entrada estándar, los procesa de alguna forma, y devuelve los datos procesados por la salida estándar. Usando la redirección la entrada estándar y/o salida estándar pueden ser archivos."

sort es un filtro simple: ordena los datos de entrada y envía el resultado a la salida estándar.

cat es incluso mas simple, no hace nada con los datos de entrada, simplemente envía a la salida cualquier cosa que le llega.

Redirección no destructiva

El uso de ">" para redireccionar la salida a un archivo es destructivo: en otras palabras, el comando

```
/home/larry/papers# ls > file-list
```

sobrescribe el contenido del fichero **file−list**.

Si en su lugar, usamos el símbolo ">>", la salida será añadida al final del archivo nombrado, en lugar de ser sobrescrito.

```
/home/larry/papers# ls >> file-list
```

Añadirá la salida de **ls** al final de **file−list**.
Es conveniente tener en cuenta que las redirecciones son características proporcionadas por el intérprete de comandos. Este, proporciona estos servicios mediante el uso de la sintaxis ">", ">>" y "<".

Algunos filtros

Listamos a continuación algunos programas que funcionan como filtros y que utilizaremos más adelante en los ejercicios:

```
cat             - concatenate files and print on the standard output
sort            - sort lines of text files
head            - output the first part of files
tail            - output the last part of files
wc              - print the number of bytes, words, and lines in files
more            - file perusal filter for crt viewing
strings         - print the strings of printable characters in files.
sed             - a Stream EDitors
grep            - print lines matching a pattern
```

El texto es una adaptaciones de:

http://lucas.hispalinux.es/Manuales-LuCAS/LIPP/lipp-1.1-html-2/lipp.htm

Ejercicios:

1.1) Crear un archivo llamado "**listado_bin**" que contenga el listado del directorio **/bin**.
Uso obligatorio de: ls; ">"

```
$
```

1.2)Crear un archivo llamado "**listado_sbin**" que contenga el listado del directorio **/sbin**.
Uso obligatorio de: ls; ">"

```
$
```

1.3)Crear un archivo llamado "**binarios**" que contenga ambos listados.
Uso obligatorio de: cat; ">"

```
$
```

1.4)Ordenar alfabéticamente el listado "**binarios**" y guardar el resultado en un archivo "**binarios2**".
Uso obligatorio de: sort; "<"; ">"

```
$
```

1.5)Verificar que los datos en "**binarios2**" sean correctos.

2.1)Crear un archivo llamado "**datosv**" con los siguientes datos personales dentro: Nombre, apellido y DNI.
Uso obligatorio de: cat; ">"

```
$
```

2.2)Agregar a "**datosv**" una línea que indique el directorio actual.
Uso obligatorio de: pwd; ">>"

```
$
```

2.3)Agregar a "**datosv**" un listado en formato largo del directorio **/etc**.
Uso obligatorio de: ls; ">>"

```
$
```

2.4)Observar (por pantalla) el archivo "**datosv**" resultante a través del filtro **more** y verificar que los datos estén correctos.
Uso obligatorio de: more; "<"

```
$
```