ESCUELA POLITÉCNICA NACIONAL
ESCUELA DE FORMACIÓN DE TECNÓLOGOS

GUIA DE PRÁCTICAS LABORATORIO

No. Práctica: 4

CARRERA: ASI

ASIGNATURA: Sistemas Operativos 1 **CÓDIGO:** TSI213

FECHA: (26/05/16)

1. PROPÓSITO DE LA PRÁCTICA: (exponga los Resultados de Aprendizaje esperados)
Editar archivos de texto.
Redireccionar de E/S
Manejar comandos utilizando tuberias

2. OBJETIVO GENERAL:

Controlar la entrada y salida de la consola de Linux.

3. OBJETIVOS ESPECÍFICOS:
- Editar archivos de texto con programas de la consola nano y vim.
- Redireccionar la salida de un comando a un archivo de texto.
- Utilizar los datos de salida de un comando como los datos de entrada de otro comando mediante tuberias de la consola de Linux.

4. INSTRUCCIONES:

4.1 Lea y comprenda las instrucciones en el documento adjunto sobre los comando utilizados para editar texto, redireccionar la E/S y las tuberias de la consola de Linux.
4.2 Realice las actividades propuestas en el documento.

ESCUELA POLITÉCNICA NACIONAL
ESCUELA DE FORMACIÓN DE TECNÓLOGOS

GUIA DE PRÁCTICAS DE LABORATORIO

5. DESCRIPCIÓN DE ACTIVIDADES Y PROCEDIMIENTO DE LA PRÁCTICA:

Siga las instrucciones en el documento adjunto.

6. BIBLIOGRAFIA REFERENCIAL:

Erazo, H. (2016). Tutorial del Editor de Texto Nano. Nanotutoriales.com. Retrieved 25 May 2016, from https://www.nanotutoriales.com/tutorial-del-editor-de-texto-nano
Manual para aprender a utilizar VIM | Emezeta. (2008). Emezeta.com. Retrieved 25 May 2016, from http://www.emezeta.com/articulos/manual-para-aprender-a-utilizar-vim

FIRMA DEL DOCENTE:

PRÁCTICA DE LABORATORIO 4

ANEXO

EDITOR DE TEXTO NANO

En este tutorial vamos a conocer el editor de texto en línea de comandos Nano, atajos de teclado o accesos directos. Los editores de texto de la terminal son especialmente útiles cuando estamos conectados de manera remota a una terminal Linux como un servidor.

Ayuda del editor

En una terminal vamos a escribir el siguiente comando:

```
nano -help
```

La terminal nos mostrará la siguiente salida:

Con este comando conocerán todos los parámetros adicionales que podemos utilizar a la hora de trabajar con el editor. Los más destacados son:

Opciones al iniciar el editor

-E ó –tabstospaces

Esto convertirá todas las tabulaciones a espacios.

```
nano -E <archivo>
nano --tabstospaces <archivo>
```

-T ó –tabsize

Esta opción permite definir el número de columnas que conforman una tabulación. Como segundo parámetro recibe en número de columnas.

```
nano -T 2 <archivo>
nano --tabsize 2 <archivo>
```

-S ó –smooth

Con este parámetro se define el modo de desplazamiento vertical en archivos extensos. Por defecto el desplazamiento es de la mitad de la pantalla, con **smooth** se vuelve de línea por línea.

```
nano -S <archivo>
nano --smooth <archivo>
```

Combinemos múltiples opciones

```
nano -ES -T 2 <archivo>
nano --tabstospaces --tabsize 2 --smooth <archivo>
```

Opciones dentro del editor

Dentro del editor tenemos muchas combinaciones de teclado disponibles para sus diferentes funciones.

La tecla **Control**, que vamos a definir con el caracter **“^”** nos será útil para cualquiera de estos atajos de teclado o accesos directos.

En la parte inferior aparece una barra donde se muestran las funciones más utilizadas. Vamos a dar un repaso en algunas de ellas.

Ayuda o ^G

Este comando nos muestra todas las funciones y su acceso directo mediante el teclado. Para salir de esta pantalla ejecutaremos la combinación de teclado **^X**

Combinaciones más utilizadas

^X	Salir del editor
^C	Mostrar la posicion actual (linea/columna)
^O	Guardar
^W	Buscar texto
^\	Buscar y reemplazar
^/	Ir a línea, columna
^K	Cortar la línea actual
^U	Pegar en la línea actual
^ALT+K	Cortar múltiples líneas
^Y	Subir
^V	Bajar

Es importante saber que siempre contamos con la barra inferior. En algunas funciones, puede que se habiliten nuevas opciones en ella.

ESCUELA POLITÉCNICA NACIONAL
ESCUELA DE FORMACIÓN DE TECNÓLOGOS
CAMPUS POLITÉCNICO "ING. JOSÉ RUBÉN ORELLANA"

MANUAL PARA APRENDER A UTILIZAR VIM

Vim es un editor de ficheros de textos muy versátil, que dispone de una gran flexibilidad a la hora de escribir scripts, modificar ficheros de texto, etc. pero sobretodo, a la hora de **programar**.

Las ventajas, son múltiples. VIM ocupa muy poco y existe en prácticamente todos los Linux o Unix disponible. Al ser un programa que se ejecuta en entorno de texto es útil para accesos remotos y edición vía terminal.

En este tutorial se verá el uso básico desde cero y comprobar lo útil que es aprender a usarlo.

VIM: Introducción

No todas las distribuciones de Linux vienen con Vim instalado. Para comprobarlo debemos ejecutar el siguiente comando:

```
vim <archivo>
```

Si no tenemos el programa instalado debemos ejecutar el siguiente comando:

```
sudo apt-get install vim
```

Una vez instalado, arrancar el vim es muy sencillo. Sólo hay que escribir en una terminal **vim**, seguido del nombre del fichero a editar. Nos aparecerá una ventana en negro, donde nos aparecerá el contenido del fichero (*o en negro si está vacío*). En la parte inferior, nos aparecerán los mensajes o comandos que escribamos para manejar el editor, así como la línea en la que estamos, porcentaje del fichero, etc.

Lo primero que hay que aprender de **Vim** (*muy importante*) es que tiene varios modos de uso:

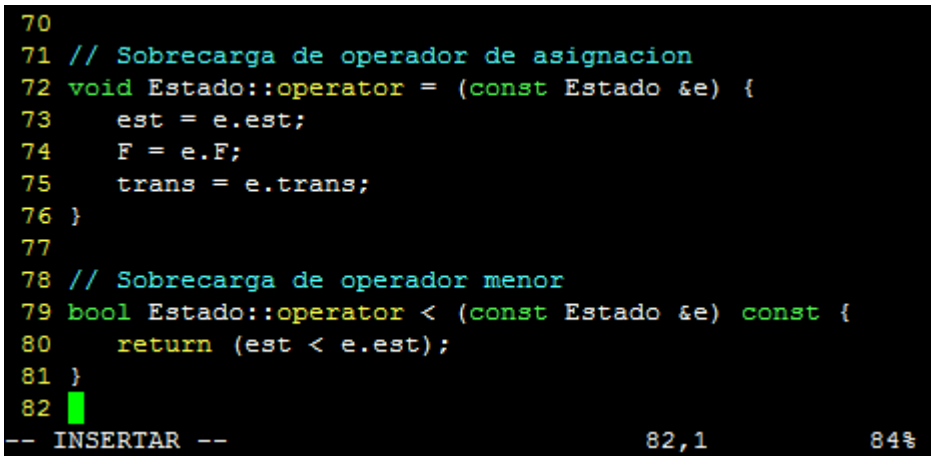
- Nada más entrar en vim a editar un fichero, estamos en el **modo normal**, en el que podremos introducir **atajos** para realizar operaciones (*borrar línea, deshacer, etc.*). **IMPORTANTE**: En este modo no podemos escribir en el fichero. Las teclas que pulsemos probablemente estarán asociadas a una operación determinada. Muchos de estos comandos (*no todos*) comenzarán por **:**.
- Para escribir texto en el fichero tendremos que entrar en el **modo edición**, que es tan fácil como pulsar la tecla **insert** (*o* **i**). Sabrás que has entrado en este modo porque abajo aparecerá el texto **-- INSERTAR --**. Ahora todo lo que tecleemos se estará escribiendo en el fichero de texto. Para volver al **modo normal** sólo hay que pulsar la tecla **ESC** (*Escape*).

Todo esto puede parecer muy lioso al principio, pero conforme comiences a utilizarlo con frecuencia, verás que resulta cómodo y lo haces de forma automática.

Primeras impresiones

Vim reconoce automáticamente por la extensión del fichero, el lenguaje en el que estamos programando (*.C, .sql, .pl, .latex, .php*...), por lo tanto nos hará un **resaltado de sintaxis** con colores, que nos resultará bastante agradable.

Esta opción puede no estar disponible en algunos Linux con versiones minimalistas de **Vim**. Sólo tenemos que instalar la versión completa de vim con **apt-get install vim-common** y escribir (*en el modo normal del Vim*) **:syntax on**.



Sin duda, el resaltado de sintaxis es algo muy valioso para el programador.

Operaciones básicas del editor

Una vez tengamos nuestro texto escrito, necesitaremos saber cómo realizar algunas operaciones como **guardar fichero**, **salir del editor**, etc.

Como hemos dicho antes, para realizar operaciones que no son de escribir en el fichero, necesitamos entrar en el **modo normal** (*pulsando ESC si estamos en el modo edición*) y a continuación los atajos que queramos:

Secuencia	Significado	¡Mnemotécnica!
:q	Salir del editor sin guardar	quit
:q!	Salir del editor sin guardar ni pedir confirmación	quit ya!
:wq!	Salir del editor guardando sin pedir confirmación	write & quit ya!
:w *f2.txt*	Guardar en un fichero llamado **f2.txt** y seguir	write en f2.txt

:e *f1.txt*	Cierra el fichero actual y abre **f1.txt**	edit f1.txt

Operaciones básicas de texto

En **Vim** como en cualquier editor, necesitaremos manipular rápidamente texto, y algo que enseguida se echa en falta en VIM, son las famosas opciones **Cortar, Copiar y Pegar**.

Con los cursores nos desplazamos por el contenido del fichero hasta llegar al inicio de la zona que queremos copiar. Pulsamos **ESC** (*si estamos en el modo edición*) y la tecla **V** para entrar en el **modo visual** y nos desplazamos hacia el final de la zona que queramos copiar. Se verá que se remarca en otro color la zona seleccionada.

Una vez tengamos la zona a copiar seleccionada, sólo tenemos que pulsar C (para cortar) o Y (para copiar). Nos aparecerá abajo un mensaje X lineas copiadas.

Ahora sólo tenemos que desplazarnos a donde queramos pegar ese fragmento y pulsar (*como siempre, en el modo normal, no en el modo edición*) la tecla **P** (*pegar*).

Veamos más operaciones de texto:

Secuencia	Significado	¡Mnemotécnica!
dd	Suprimir línea actual al buffer (*p para pegar*)	delete
u	Deshacer el último cambio en el fichero	undo
CTRL+R	Rehacer el último cambio en el fichero	redo
guu	Convertir a minúsculas la línea actual	lowercase

gUU	Convertir a mayúsculas la línea actual	UPPERCASE
:*num*	Posicionarse en la línea *num* del fichero	
gg	Posicionarse al principio del fichero	
G	Posicionarse al final del fichero	
ga	Muestra código ASCII, hex y octal del caracter actual	

Operaciones de búsqueda y sustitución

Otra función que solemos echar de menos enseguida es la de buscar algún texto, reemplazar, etc. En **vim** no puede faltar esa opción, con sus respectivas mejoras y añadidos:

Para buscar un texto, escribimos (*en modo normal, pulsando antes* **ESC** *si estamos en modo edición*) la secuencia */palabra*. Veremos que se resalta la palabra encontrada (*o nos avisa de que no existe*). Entonces podemos seguir buscando la próxima coincidencia pulsando **n** o buscarla hacia detrás pulsando **N**.

Para sustituir un texto debemos escribir la secuencia **:%s/***texto1***/***texto2***/g**, donde **texto1** es el texto a buscar y **texto2** el texto que será reemplazado. Si incluimos la **g** final (*global*), sustituirá todas las coincidencias que encuentre, sino sólo la primera que encuentre.

REDIRECCIÓN DE E/S

En repetidas ocasiones en la vida de un sistema es mejor tener todo en archivos, ya sea para guardar algún historial o para automatizar ciertas funciones dentro de scripts.

Para almacenar o sacar información de archivos y vincularlas con entradas o salidas estándares se utilizan las Redirecciones.

La redirección se expresa con los símbolos "Mayor" > y "Menor" <. Y se pueden utilizar en forma simple o doble.

Utilizando el comando cat se puede hacer una copia de arch1.txt a arch2.txt utilizando redirección.

```
$ cat arch1.txt > arch2.txt
```

O se puede redireccionar un archivo para visualizarlo con el comando less.

```
$ less < arch1.txt
```

Redirección de escritura

Para escribir un archivo se utiliza >. Hay que tener mucho cuidado de no borrar un archivo sobre-escribiéndolo. Cuando se utilizan redirecciones, debido a su utilidad en los scripts, "no se realizan confirmaciones".

Si el archivo a escribir ya existe desde antes, el redireccionador > lo sobreescribe con flujo de texto nuevo.

En cambio, el operador >> realiza un agregado de texto en el flujo existente.

No hay nada mejor que un ejemplo clarificador:

```
$ escribe-en-salida-estandar > archivo.txt
```

El (falso) comando escribe-en-salida-estándar justamente hace eso, escribe unas cuantas cosas en salida estándar.

Puede ser un comando ls, un comando cal (calendario) o cualquier comando antes visto, así como también una combinación de comandos por tuberías.

En este punto, el contenido de archivo.txt es lo mismo que saldría en pantalla. Si ejecutamos otro comando redireccionado a archivo.txt (nuevamente), éste pierde su contenido y el resultado de la operación pasa a estar en el mismo.

Cuando se necesita tener una lista de acontecimientos, no se quiere que un acontecimiento nuevo borre a todos los anteriores. Para lograr esto agregamos en vez de sobreescribir.

```
$ echo Este es el acontecimiento Nro. 1 > bitacora.log
```

```
$ echo Este es el segundo acontecimiento >> bitacora.log
```

Va a escribir dos líneas en el archivo bitacora.log sin eliminar nada.

Ejemplo: Si queremos combinar el ejemplo de las tuberías con lo aprendido recientemente podríamos escribir:

```
$ cat archivo.txt | sort | sacar-repetidas | diccionario >> glosario.txt
```

Redirección de lectura

Para la lectura es el símbolo menor < y se utiliza de la siguiente manera:

```
$ comando-que-acepta-stdin < archivo-de-entrada.txt
```

Como por ejemplo:

```
$ mail usuario1 usuario2 < correo.txt
```

Dónde correo.txt podría ser un archivo que se genere automáticamente... así como su contenido. Otra facilidad para redireccionar entrada estándar es <<, que después de un comando, permite ingresar, por teclado, un texto que se constituirá en la entrada estándar.

A continuación de << debe ponerse una palabra, que indicará fin de entrada (en nuestro ejemplo, adiós). La entrada estándar constará de las líneas que se digiten a continuación hasta la primera que contenga sólo la palabra que indicaba fin de entrada. Por ejemplo:

```
$ sort <<chau
> Perú
> Argentina
> Brasil
> adiós
Argentina
Brasil
Perú
```

ordenará las palabras dadas (excepto chau que indica el fin de la entrada). Así, << es equivalente a editar un archivo y después redireccionarlo a la entrada estándar de un programa.

TUBERÍAS

Podríamos representar cada programa como una caja negra que tiene una entrada y una salida que se pueden unir entre ellos.

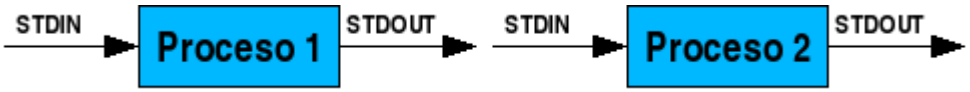
```
Debido a que la entrada por defecto es el teclado y la salida por
defecto es terminal, graficaremos cuando sea necesario ambos.
```

El ejemplo que utilizamos se encuentra esquematizado en la figura

Siendo la entrada estándar el teclado y la salida estándar el terminal o por simplicidad la pantalla.

Vamos a suponer un caso ficticio donde necesitamos todas las definiciones de cada palabra en un texto. Primero las ordenamos alfabéticamente, luego utilizamos un comando ficticio llamado diccionario que toma palabras de la entrada estándar y las reescribe junto a su significado en la salida estándar.

Su esquema se ve en la figura



En este caso nombramos por separado las entradas y salidas estándares de los dos programas, pero la unión entre ambos programas se puede considerar como un sólo tubo.

En ese tubo, el flujo está en un estado intermedio, donde está ordenado, pero no tiene las definiciones de diccionario.

En la línea de comandos esto se escribe de la siguiente manera:

```
$ sort | diccionario
```

Donde el caracter " | " representa la conexión entre la salida estándar de un programa y la entrada estándar de otro.

Con este fuerte y simple concepto se pueden concatenar gran cantidad de programas como si fuera una línea de producción en serie para generar resultados complejos.

Para mejorar nuestro ejemplo sacaremos las palabras repetidas, antes de mostrarlas con definiciones. Suponiendo que exista un programa llamado sacar-repetidas, la línea de comando sería:

```
$ sort | sacar-repetidas | diccionario
```

Simple, utilizando herramientas sencillas logramos algo un poco más complicado. El inconveniente que tenemos en este ejemplo es que hay que escribir aquello a procesar. Normalmente queremos utilizar archivos como entrada de nuestros datos.

Es necesario un comando que envíe a la salida estándar un archivo, así se procesa como la entrada estándar del sort y continúa el proceso normalmente. Este comando es cat. La sintaxis es simple

```
cat nombre-de-archivo
```

Quedando nuestro ejemplo:

```
$ cat archivo.txt | sort | sacar-repetidas | diccionario
```

... esto crea un glosario de las palabras que se encuentren en archivo.txt

La combinación de comandos es incalculable y brinda posibilidades enormes. Veamos algunos ejemplos.

En el caso que se quieran buscar procesos con el string http:

```
$ ps ax | grep http
3343 ?        S      0:00 httpd -DPERLPROXIED -DHAV
3344 ?        S      0:00 httpd -DPERLPROXIED -DHAV
3975 ?        S      0:00 httpd -DPERLPROXIED -DHAV
12342 pts/6    S      0:00 grep http
```

Si queremos eliminar la última línea podemos volver a usar grep con la opción -v

```
$ ps ax | grep http | grep -v grep
3343 ?        S      0:00 httpd -DPERLPROXIED -DHAV
3344 ?        S      0:00 httpd -DPERLPROXIED -DHAV
3975 ?        S      0:00 httpd -DPERLPROXIED -DHAV
```

Se pueden filtrar las líneas que contengan la palabra linux del archivo arch1.txt y luego mostrarlas en un paginador como less.

```
$ cat arch1.txt | grep linux | less
```

Podemos enviar los resultados por correo a un amigo, con un asunto que diga "Tu archivo".

```
$ cat arch1.txt | grep linux | mail -s 'Tu archivo' amigo@email.com
```

EJERCICIOS

1. En su directorio home crear el directorio `lab4`
2. Dentro de `lab4` crear dos directorios llamados: `uno` y `dos` (hacerlo con un solo comando)
3. Escribir UN comando que cree un archivo oculto llamado secreto dentro de lab10 con el contenido "Este es un secreto". Los archivos ocultos en Linux deben empezar con un punto, ejemplo `.oculto`
4. Estando dentro de lab10, escribir un comando que cree un archivo llamado `ficheros_bin_descendente` que contenga el listado de todos los ficheros que se encuentran en el directorio `/bin` ordenados de manera descendente (de z - a).
5. Ingresar al directorio `uno`, Estando en `uno`, escribir un comando que utilice tuberias y filtros para crear, dentro del directorio dos, un archivo llamado `ficheros_bin_ascendente` que contenga el texto del archivo `ficheros_bin_descendente` pero ordenado de manera ascendente.
6. Estando en el directorio uno, escribir un comando que mueva el archivo `ficheros_bin_descendente` al directorio `dos`
7. Estando en el directorio `uno`, escribir un comando que nos permita mover hacia el directorio `dos`
8. Estando en `dos`, escribir un comando que cree un archivo oculto llamado `tmp`
9. Estando en `dos`, escribir un comando que elimine el directorio `uno`
10. Estando en `dos`, escribir un comando que mueva todos los archivos en el directorio actual hacia la carpeta `lab4`
11. Estando en `dos`, eliminar el directorio `dos` (si, el directorio actual)
12. Sin moverse del directorio actual, crear un archivo llamado `tmp`. ¿Qué sucede?
13. Moverse al directorio `lab4`.
14. Escribir un comando que cree un archivo llamado ficheros que contenga el listado de todos los archivos (incluidos los archivos ocultos) que se encuentren en el directorio actual.
15. Escribir un comando que cree un archivo llamado `historial_mk` que contenga el historial de comandos ejecutados que contienen la palabra `mkdir`.

Referencias

Erazo, H. (2016). *Tutorial del Editor de Texto Nano*. *Nanotutoriales.com*. Retrieved 25 May 2016, from https://www.nanotutoriales.com/tutorial-del-editor-de-texto-nano

Manual para aprender a utilizar VIM | Emezeta. (2008). *Emezeta.com*. Retrieved 25 May 2016, from http://www.emezeta.com/articulos/manual-para-aprender-a-utilizar-vim