



IES Rodrigo Caro

Dpto de Informática



Implantación de Sistemas Operativos de Manuel Fco Domínguez Tienda tiene licencia Creative Commons Reconocimiento y compartir bajo la misma licencia 3.0 España.

Las imágenes proceden de Internet y pueden tener copyright.

6.- Edición y visualización de ficheros de textos.

Índice:

- 1.- Introducción.
- 2.- El editor de textos nano.
- 3.- Introducción al editor de textos vi/vim
- 4.- El editor de textos gedit.
- 5.- Visualizar el contenido de ficheros.
- 6.- Comparar contenido de ficheros.

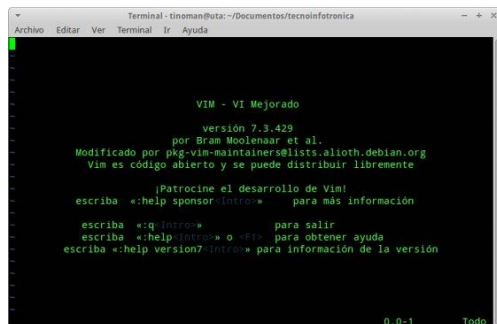


1.- Introducción.

El mundo Linux está repleto de editores de texto:

Vi, vim, nano, pico, emacs, joe, gedit, etc

Nosotros nos vamos a centrar el editor de textos nano, aunque haremos una pasada por el editor vim (vi modificado) y el editor gráfico gedit.



```
Terminal - tinoman@utac:~/Documentos/tecnoinformatica
Archivo  Editar  Ver  Terminal  Ir  Ayuda

      VIM - VI Mejorado
      versión 7.3.429
      por Bram Moolenaar et al.
      Modificado por pkg-vim-maintainers@lists.allyth.debian.org
      Vim es código abierto y se puede distribuir libremente

      ¡Patrocine el desarrollo de Vim!
      escriba «:help sponsorintro» para más información

      escriba «:q !intro» para salir
      escriba «:help intro» o «ft» para obtener ayuda
      escriba «:help version7 intro» para información de la versión

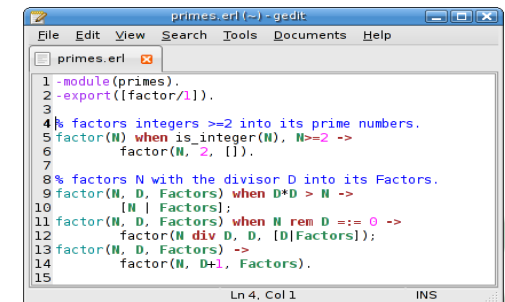
      0.0-1      Todo
```



```
GNU nano 1.3.12      File: test.htm

[New File]

Get Help  WriteOut  Read File  Prev Page  Cut Text  Cur Pos
Exit      Justify   Where Is  Next Page  UnCut Text  To Spell
```



```
primes.erl - gedit
File Edit View Search Tools Documents Help

primes.erl
1 -module(primes).
2 -export([factor/1]).
3
4 % factors integers >=2 into its prime numbers.
5 factor(N) when is_integer(N), N>=2 ->
6     factor(N, 2, []).
7
8 % factors N with the divisor D into its Factors.
9 factor(N, D, Factors) when D*D > N ->
10     [N | Factors];
11 factor(N, D, Factors) when N rem D == 0 ->
12     factor(N div D, D, [D|Factors]);
13 factor(N, D, Factors) ->
14     factor(N, D+1, Factors).
15

Ln 4, Col 1      INS
```

2.- El editor nano

Ventaja: Es muy sencillo de manejar.

Inconveniente: No es un editor de textos que esté en todas las distribuciones.

2.1.- Operaciones habituales:

Crear un archivo nuevo: \$nano prueba

Ayuda: Ctrl+G

Guardar: Ctrl+O

Salir: Ctrl+X

Abrir: \$nano prueba

Desplazarse a una línea: Alt +g

Mover una línea: Ctrl+k (cortamos la línea) ctrl+U (pegamos)

2.- El editor nano

2.1.- Operaciones habituales:

Mover varias líneas: ctrl+k, ctrl+k, ctrl+k, Ctrl+u (pegamos las líneas)

Mover un texto: ctrl + 6 (Activamos el marcado)
 Seleccionamos
 ctrl +k (cortamos)
 Nos colocamos en el destino
 ctrl+u (pegamos)

Copiar línea: ALT+6 (copiamos) ctrl + u (pegamos)

Copiar un texto: ctrl + 6 (Activamos el marcado)
 Seleccionamos
 ALT +6 (copiamos)
 Nos colocamos en el destino
 ctrl+u (pegamos)

2.- El editor nano

2.2.- Otras operaciones:

Buscar un texto:

`ctrl+w`

`alt+w` → Busca la siguiente cadena.

Reemplazar un texto:

`alt+r`

Insertar otro fichero en el actual:

`ctrl+r`

Escribimos el nombre o bien `^T`

Ejercicio: Ejecuta los siguientes comandos y observa la diferencias:

`$nano prueba`

`$nano -c prueba` → Aparecerá nº de línea, nº de columna, nº de carácter.

2.- *El editor nano*

2.2.- Otras operaciones:

Dejar de forma permanente la numeración de líneas:

1.- # nano /etc/nanorc

2.- Buscamos la línea que pone: set const

3.- Y la descomentamos

3.- El editor vi/vim

- ❖ Vi es un editor de texto usado para editar ficheros de texto, archivos de configuración del sistema o programas escritos en C.
- ❖ Aunque hay muchos editores de texto en GNU/LINUX (seguramente más fáciles de usar y mas auto explicativos), el editor vi es el único editor que está disponibles en todas las distribuciones de UNIX.
- ❖ Uno de los puntos fuertes del editor vi es la posibilidad de manipular el editor sin mover las manos del teclado.

Vim (del inglés *Vi IMproved*) es una versión mejorada del editor de texto vi.

Vim No viene por defecto → `#aptitude install vim`

3.- El editor vi/vim

Existen tres modos de operación:

- **Modo comando:** En este modo, vi espera una acción a ser ejecutada como copiar, pegar o mover el cursor. Este es su comportamiento por defecto, así que cuando abres un archivo y empiezas a escribir no verás lo que estás escribiendo porque estarás ejecutando comandos. **ESC**
- **Modo Insertar:** En este modo de hecho puedes escribir en el archivo, pero antes debes usar uno de los comandos de insertar como **i**
- **Modo comando extendido:** Este es un modo especial donde puedes pasarle argumentos a los comandos. Para entrar en este modo solo escribe **dos puntos** y el comando. Por ejemplo:

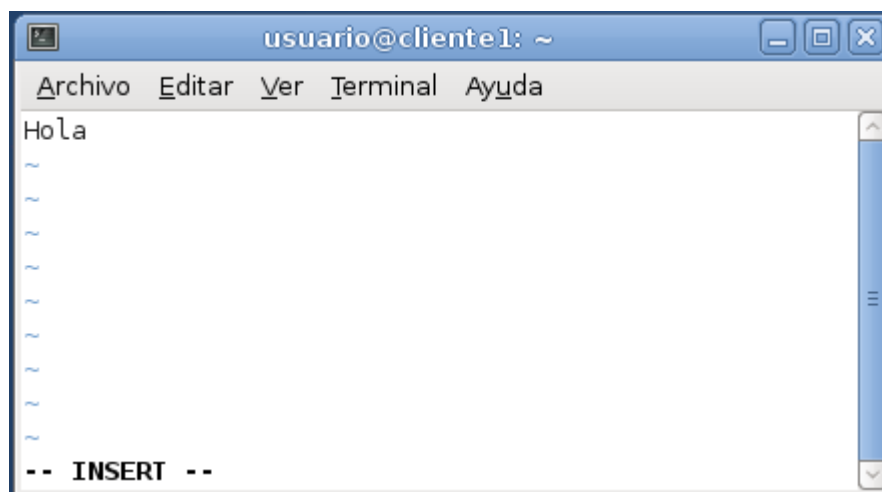
```
:1,4 y
```

Esto indica que se debe copiar de la línea 1 a la 4. Ahora vamos a ver algunos comandos agrupados según su funcionalidad:

3.- El editor vi/vim

MODO COMANDO EXTENDIDO

:set showmode → Nos muestra cuando estamos en modo inserción con la palabra INSERT.



:set number	Activar numeración de línea
:syntax on	Reconocimiento de sintaxis de lenguaje.
:set ai	Indentación automática
>>	Agregar un nivel de indentación
<<	Quitar un nivel de indentación

Saliendo

:w	Guardar sin salir
:wq	Guardar y salir
:q	salir (no sale si han habido cambios)
:q!	Salir sin guardar

Desplazarse a una línea

: num_linea

3.- El editor vi/vim

MODO COMANDO EXTENDIDO

EJECUCIÓN DE ÓRDENES DEL SHELL

El editor vi permite la ejecución del intérprete de comandos desde la línea de órdenes. El carácter "!" indica al editor que la siguiente instrucción es una orden del shell. La orden ":r !orden" permite insertar en la posición actual el resultado de la orden del shell.

Por ejemplo, el comando del editor vi

```
:r !ls -la /root
```

inserta en la posición del cursor el listado de archivos del directorio /root

3.- El editor vi/vim

MODO COMANDO

Cortar y pegar

yy	Copiar línea
p	Pegar después del cursor
P	Pegar antes del cursor
dd	Borrar línea
x	Borra el carácter actual

Modo Visual (marcando texto)

v	Modo visual por carácter
V	Modo visual por línea
Ctrl+v	Modo visual por bloque

Comandos del modo visual

aw	marcar palabra
ab	Marcar lo que este en <u>paréntesis</u>
aB	marcar lo que está en corchetes
>	Desplazar a la derecha
<	Desplazar a la izquierda
y	Copiar el texto marcado
d	Borrar el texto macado
~	intercambio de mayúsculas y minúsculas

BUSCAR TEXTO

comando	descripción
/texto	Busca hacia delante "texto".
?texto	Busca hacia tras "texto".
n	Busca la próxima coincidencia.
N	Busca la anterior coincidencia.

REEMPLAZAR TEXTO

comando	descripción
:g/texto1/s//texto2/g	Reemplaza texto1 por texto2 en todo el fichero.
:g/texto1/s//texto2/gc	Reemplaza texto1 por texto2 en todo el fichero. Pide confirmación en cada sustitución.

DESHACER UN CAMBIO

comando	descripción
u	Deshace el último cambio.
nu	Deshace los n últimos cambios.
U	Deshace todos los cambios de la línea actual.

3.- El editor vi/vim

Abrir varios ficheros a la vez con vim

tatai

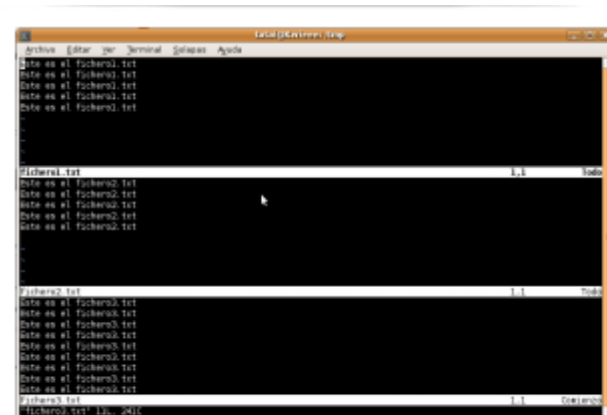
Cuando queremos abrir varios ficheros a la vez desde la consola, la forma más habitual es la de escribir el comando vim y después añadir cada uno de los ficheros separados por al menos un espacio. De esta forma, mediante los comandos :next y :previous (recordar pulsar <Esc> antes de introducir comandos) podemos pasar de un fichero a otro.

Esto está bien cuando no te importa tenerlas en ventanas distintas, ¿pero y si queremos tenerlos todos a la vista a la vez? Bien, una de las formas es decirle a vim que no abra cada una en una ventana distinta, sino mediante *horizontal splits*, es decir, que si tenemos dos ficheros abiertos, cada uno ocupa la mitad de la pantalla (en horizontal).

Para conseguir esto, lo único que tenemos que hacer es escribir en la consola vim -o y luego todos y cada uno de los ficheros separados al menos por un espacio. Es decir, lo haremos con la opción -o (letra o minúscula). Tal que así:

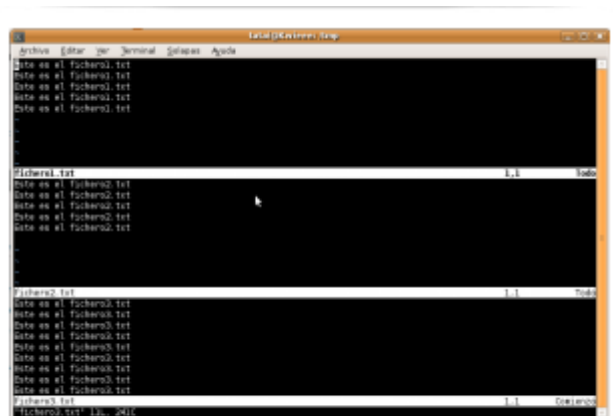
```
vim -o fichero1.txt fichero2.txt fichero3.txt
```

Bien, todo tiene un límite. Vim abrirá tantos ficheros como le indiqueis... pero es evidente que si abris muchos a la vez, no tendremos espacio para verlos. Ideal para usar con algunos scripts 😊



vim con la opción -o

3.- El editor vi/vim



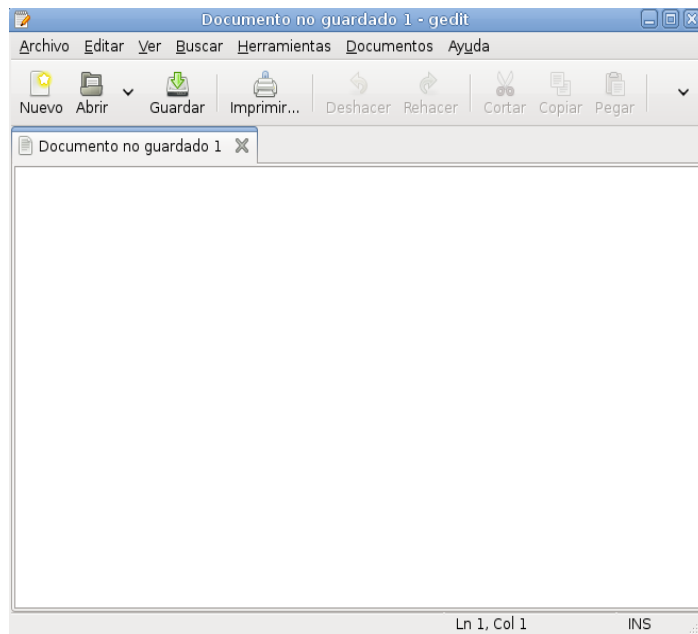
vim con la opción -o

Una vez que has creado varias ventanas, puedes moverte entre ellas y manipularlas con las siguientes órdenes, todas empiezan con *Control+w*:

- *Control+w Flecha Arriba*: Mueve el cursor activo hacia la ventana situada por arriba de la actual. Si en vez de Flecha Arriba pulsas Flecha Abajo, Flecha Derecha o Flecha Izquierda, el cursor cambiará de ventana en consecuencia. Evidentemente, si estás ya en la ventana de más arriba, no hará nada.
- *Control+w Control+w*: Mueve el cursor hacia la siguiente ventana. Lo hace de forma cíclica, repitiendo la secuencia el cursor irá moviéndose por todas las ventanas.
- *Control+w _*: Maximizar la ventana actual.
- *Control+w =*: Igualar el tamaño de todas las ventanas.
- *Control+w +*: Aumenta en una línea el tamaño de la ventana activa.

4.- El editor gedit

\$gedit prueba



Características:

Además de las funcionalidades básicas que son habituales en un editor de texto, como copiar, cortar y pegar texto, imprimir, etc. **gedit** incorpora, entre otras, las siguientes funcionalidades:

- .- Soporte de textos internacionalizados, usando la codificación UTF-8.

- .- Coloreado del texto según la sintaxis de varios lenguajes de programación: C++ ,Java, **Script Linux**, etc

- .- Corrector ortográfico multi-idioma.

- .- Incorporación de plugins para ampliar las funcionalidades básicas del programa.

- .- Etc

5.- Visualizar ficheros de texto.

5.1.- El comando cat

Nos permite visualizar el contenido de un fichero de textos.

Opción:

-n → Enumera las líneas de salida

Ejemplos:

```
$ cat -n prueba
```

```
$ cat -n prueba |less → Nos muestra la información paginada.
```

5.2.- El comando head

Nos permite visualizar las 10 primeras líneas de un fichero.

Opción:

-n n^o_líneas → Muestra las primeras n^o_líneas

Ejemplos:

```
$head -n 5 prueba
```

```
$head -n 5 prueba |cat -n
```


5.- Visualizar ficheros de texto.

5.3.- El comando `tail`

Nos permite visualizar las 10 últimas líneas de un fichero.

Opción:

`-n nº_líneas` → Muestra las primeras nº_líneas

Ejemplos:

`$tail -n 5 prueba`

`$tail -n 5 prueba |cat -n`

Ejercicio:

1.- ¿Sabes qué es un fichero log?

2.- Visualiza el contenido del fichero `syslog`.
`/var/log/syslog`

3.- Visualiza las últimas 5 líneas.

6.- Comparar el contenido de ficheros.

El shell de linux: Comando diff

El comando diff nos permite comparar dos ficheros línea a línea y nos informa de las diferencias entre ambos ficheros. Diff tiene muchas opciones. Las que más uso son -w, -q, -y.

La sintaxis del comando es la siguiente:

diff [opciones] [fichero1] [fichero2]

Si queremos comparar dos ficheros, ignorando los espacios en blanco, utilizaremos el parámetro -w:

diff -w fichero1 fichero2

Si lo que queremos es que no nos muestre las diferencias, sino que tan sólo nos informe de si son diferentes o no:

diff -q fichero1 fichero2

Si queremos que nos muestre la salida con las diferencias marcadas a dos columnas:

diff -y fichero1 fichero2

Como en muchos otros comandos, también podemos utilizar la opción -i, que ignora la diferencia entre mayúsculas y minúsculas.

```
usuario@cliente1: ~  
Archivo Editar Ver Terminal Ayuda  
GNU nano 2.2.4 Fichero: prueba1  
Hola  
adios  
Hasta mañana
```

```
usuario@cliente1: ~  
Archivo Editar Ver Terminal Ayuda  
GNU nano 2.2.4 Fichero: prueba2  
Hola  
Adios  
Hasta maana
```

