

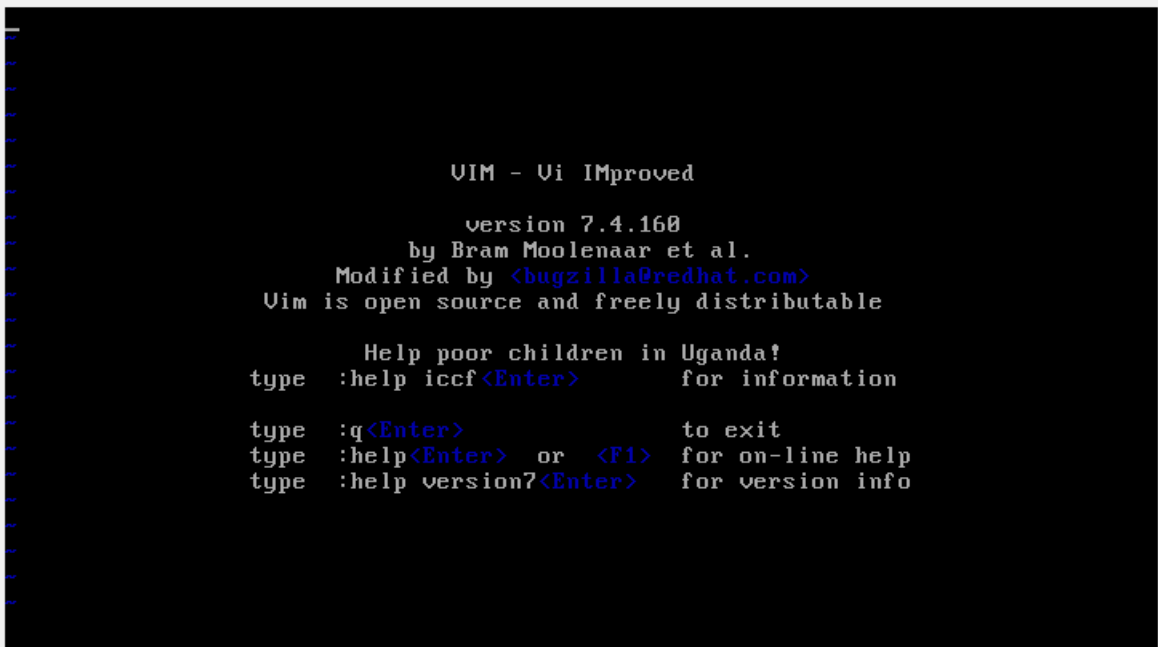


Unidad 3

Editor Vi

Uso básico de vi.

El editor vi es un editor de texto de pantalla completa que maneja en memoria el texto entero de un archivo. Es el editor clásico de UNIX; está en todas las versiones. Puede usarse en cualquier tipo de terminal con un mínimo de teclas; esto lo hace difícil de usar hasta que uno se acostumbra.



```
VIM - Vi IMproved

        version 7.4.160
        by Bram Moolenaar et al.
    Modified by <bugzilla@redhat.com>
Vim is open source and freely distributable


        Help poor children in Uganda!
type  :help iccf<Enter>          for information

type  :q<Enter>                  to exit
type  :help<Enter> or <F1>      for on-line help
type  :help version7<Enter>    for version info
```

Existe un editor vi ampliado llamado Vim que contiene facilidades adicionales, así como diversas versiones del vi original. En todos los casos, el conjunto de comandos básicos es el mismo.



SPONSOR
Vim development

VOTE
for features

the editor [go to HTTPS page](#)

BUY
the Vim book

HELP
Uganda

LEARN
Vim

not logged in ([login](#))

Google Custom Search

[Home](#)
[Advanced search](#)
[About Vim](#)
[Community](#)
[News](#)
[Sponsoring](#)
[Trivia](#)
[Documentation](#)
[Download](#)
[Scripts](#)
[Tips](#)
[My Account](#)
[Site Help](#)

[What is Vim online?](#)
Vim online is a central place for the Vim community to store useful Vim tips and tools. Vim has a scripting language that allows for plugin like extensions to enable IDE behavior, syntax highlighting, colorization as well as other advanced

Vim - the ubiquitous text editor

Vim is a highly configurable text editor built to make creating and changing any kind of text very efficient. It is included as "vi" with most UNIX systems and with Apple OS X.

Vim is rock stable and is continuously being developed to become even better. Among its features are:

- persistent, multi-level undo tree
- extensive plugin system
- support for hundreds of programming languages and file formats
- powerful search and replace
- integrates with many tools

News

Vim 8.0.1173 is the current version

Vimfest 2017 in Berlin

[2017-09-08] Vimfest is happening again! 22 to 24 September in Berlin. Find more information on [the website](#). (Bram Moolenaar)

Vim website now on https

[2017-09-02] Since the Vim website hosts scripts, security is relevant. SourceForge now supports using https, but only on sourceforge.io. That is why you now get redirected from vim.org to vim.sourceforge.io. Let me know if something does not work. (Bram Moolenaar)

Vim voted #1 text editor on diffur

[2017-03-18] It's no surprise to Vim users, but always nice to see how Vim compares to other editors. You can view the list on [diffur.com](#). (Bram Moolenaar)

[more news...](#) [Get a Vim T-shirt from FreeWear](#) [Get a Vim sticker or button](#) [Vim items from Japan](#)

[Buy at Amazon](#)
[Help Uganda](#)

<http://www.vim.org/>

Existen en UNIX otros editores más potentes y versátiles, como Emacs, que provee un ambiente de trabajo completo; también versiones fáciles de manejar como Jove o Pico, o aún mínimas e inmediatas como ae. En ambiente X-Windows hay muchos editores amigables, fáciles de usar y con múltiples capacidades. No obstante, vi está en todos los UNIX, requiere pocos recursos, se usa mucho en administración, para programar y en situaciones de emergencia. En casos de roturas de discos, corrupción de sistemas de archivos, errores en el arranque y otras catástrofes, puede ser el único editor disponible. Como la mayoría de las configuraciones en UNIX se manejan editando archivos, disponer de esta capacidad es esencial en la administración de un sistema.



Emacs 25.3 is out, download it [here!](#)



[About GNU](#) · [Philosophy](#) · [Licenses](#) · [Education](#) · [Software](#) · [Documentation](#) · [Help GNU](#) ·

[JOIN THE FSF](#)

[Home](#) [Features](#) [Releases](#) [Download](#) [Documentation & Support](#) [Tour](#) [Further information](#)



GNU Emacs

An extensible, customizable,
free/libre text editor — and more.

At its core is an interpreter for Emacs Lisp, a dialect of the Lisp programming language with extensions to support text editing.

↓ GNU/Linux

↓ Windows

↓ macOS

<https://www.gnu.org/software/emacs/>



WIKIPEDIA
The Free Encyclopedia

[Main page](#)
[Contents](#)
[Featured content](#)
[Current events](#)
[Random article](#)
[Donate to Wikipedia](#)
[Wikipedia store](#)

[Interaction](#)
[Help](#)
[About Wikipedia](#)
[Community portal](#)
[Recent changes](#)
[Contact page](#)

[Tools](#)
[What links here](#)
[Related changes](#)
[Upload file](#)
[Special pages](#)
[Permanent link](#)
[Page information](#)
[Wikidata item](#)
[Cite this page](#)

Article [Talk](#)

[Read](#) [Edit](#) [View history](#)

[Not logged in](#) · [Talk](#) · [Contributions](#) · [Create account](#) · [Log in](#)

JOVE

From Wikipedia, the free encyclopedia

For other uses, see [Jove \(disambiguation\)](#).



This article **needs additional citations for verification**. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed. (May 2009) [\(Learn how and when to remove this template message\)](#)

JOVE (*Jonathan's Own Version of Emacs*)^[1] is an open-source, Emacs-like text editor, primarily intended for Unix-like operating systems. It also supports MS-DOS and Microsoft Windows. JOVE was inspired by Gosling Emacs but is much smaller and simpler, lacking Mocklisp. It was originally created in 1983 by Jonathan Payne while at Lincoln-Sudbury Regional High School in Massachusetts, United States on a PDP-11 minicomputer.^[2] JOVE was distributed with several releases of BSD Unix, including 2.9BSD, 4.3BSD-Reno and 4.4BSD-Lite2.

As of 2010, the latest development release of JOVE is version 4.16.0.73; the stable version is 4.16. Unlike GNU Emacs, JOVE does not support UTF-8.^[3]

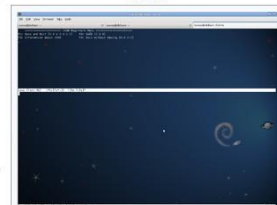
See also [\[edit \]](#)

- List of text editors
- Comparison of text editors

External links [\[edit \]](#)

- JOVE Development FTP site[ⓘ]
- FSF Free Software Directory entry[ⓘ]

JOVE



Jove running in a Debian box

Developer(s)	Jonathan Payne, Hugh Redelmeier
Stable release	4.16 / March 19, 1996; 21 years ago

<https://en.wikipedia.org/wiki/JOVE>



WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction

Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools

What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Article | Talk

Read | Edit | View history

Search Wikipedia

Pico (text editor)

From Wikipedia, the free encyclopedia

Pico (**Pine composer**) is a text editor for Unix and Unix-based computer systems. It is integrated with the Pine e-mail client, which was designed by the Office of Computing and Communications at the University of Washington.

From the Pine FAQ: "Pine's message composition editor is also available as a separate stand-alone program, called PICO. PICO is a very simple and easy-to-use text editor offering paragraph justification, cut/paste, and a spelling checker...".^[1]

Pico does not support working with several files simultaneously and cannot perform a find and replace across multiple files. It also cannot copy text from one file to another (though it is possible to read text into the editor from a file in its working directory). Pico does support search and replace operations.

By comparison, some popular Unix text editors such as vi and Emacs provide a wider range of features than Pico; including regular expression search and replace, and working with multiple files at the same time. By comparison, Pico's simplicity makes it suitable for beginners.^[2]

A clone of Pico called nano, which is part of the GNU Project,^[3] was developed because Pico's earlier license had unclear redistribution terms.^[4] Newer versions of Pico as part of Alpine are released under the Apache License.

Contents [hide]

- Basic commands and navigation
- Command-line options
- See also
- References
- External links

Pico

Developer(s)	University of Washington
Written in	C
Operating system	Unix-like
Available in	English
Type	Text editor
License	Apache License
Website	www.washington.edu/pine/

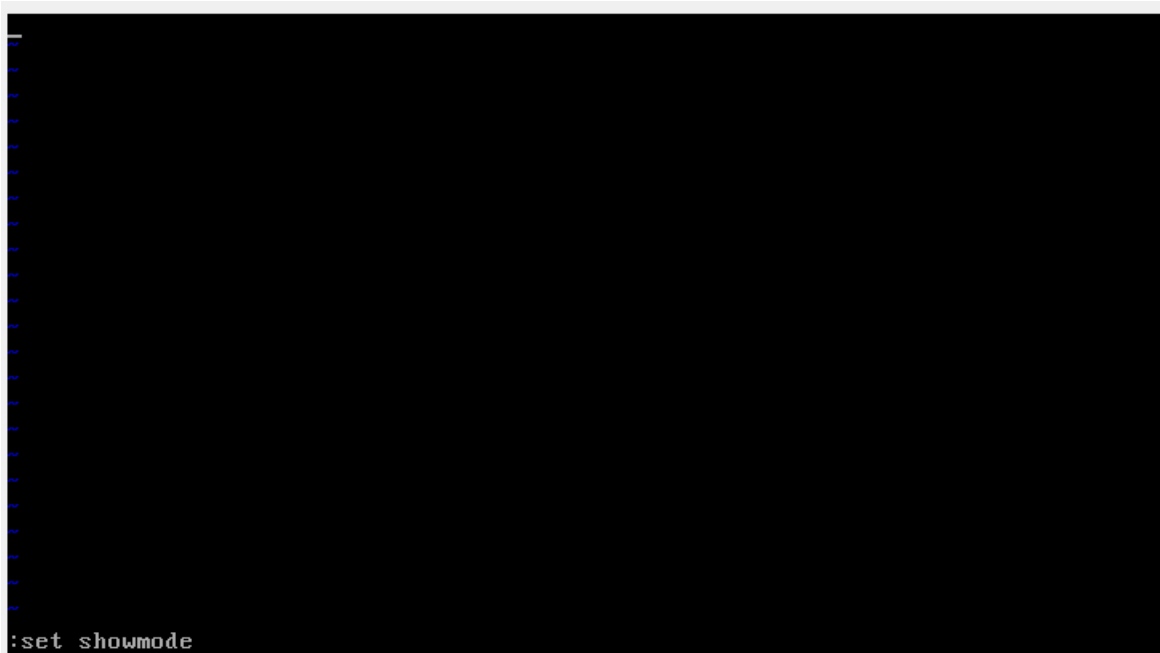
[https://en.wikipedia.org/wiki/Pico_\(text_editor\)](https://en.wikipedia.org/wiki/Pico_(text_editor))

Modos de vi.

Existen tres modos o estados en vi entre los cuales iremos “rotando” dependiendo de que necesitemos hacer:

- **modo comando**: las teclas ejecutan acciones que permiten desplazar el cursor, recorrer el archivo, ejecutar comandos de manejo del texto y salir del editor. Es el modo inicial de vi.
- **modo texto o modo inserción**: las teclas ingresan caracteres en el texto.
- **modo última línea o ex**: las teclas se usan para escribir comandos en la última línea al final de la pantalla.

Confundir un modo con otro es una de las de mayores dificultades para el manejo de vi. Puede activarse un indicador de modo escribiendo “:set showmode”



Esto hace aparecer una leyenda que indica si se está en modo inserción.



Cambios de modo.

Comando a texto

Teclas de inserción: **i**, **l**, **a**, **A**, **o**, **O**, **o**.

Tecla de sobre escritura: **R**.

Texto a comando:

Tecla “ESC”.

Comando a última línea:

Tecla: “/”



Última línea a comando:

Tecla ENTER (al finalizar el comando).

Tecla ESC (interrumpe el comando).

Guía de supervivencia.

Con unos pocos comandos básicos se puede trabajar en vi editando y guardando un texto:

- “**vi prueba.txt**” Vi siempre arranca en modo comando
- “**i**” inserta texto a la izquierda del cursor
- “**a**” agrega texto a la derecha del cursor
- “**ESC**” vuelve a modo comando
- “**x**” borra el caracter bajo el cursor
- “**dd**” borra una línea
- “**h**” o flecha izquierda mueve el cursor un caracter a la izquierda
- “**j**” o flecha abajo mueve el cursor una línea hacia abajo
- “**k**” o flecha arriba mueve el cursor una línea hacia arriba
- “**l**” o flecha derecha mueve el cursor un caracter a la derecha
- “**:w**” salva el archivo (graba en disco)
- “**:q**” sale del editor (debe salvarse primero)





Uso avanzado de vi.

Invocación de vi.

- “vi” abre la ventana de edición sin abrir ningún archivo.
- “vi arch1” edita el archivo arch1 si existe; si no, lo crea.
- “vi arch1 arch2” edita sucesivamente los archivos arch1 y luego arch2.
- “vi +45 arch1” edita el archivo arch1 posicionando el cursor en la línea 45.
- “vi +\$ arch1” edita el archivo arch1 posicionando el cursor al final del archivo.
- “vi +/Habia arch1” edita el archivo arch1 en la primera ocurrencia de la palabra “Habia”.

Modo Comando.

El editor vi, al igual que todo UNIX, diferencia mayúscula y minúscula. *Confundir un comando en minúscula digitando uno en mayúscula suele tener consecuencias catastróficas. Se aconseja evitar sistemáticamente el uso de la traba de mayúsculas; mantener el teclado en minúsculas.*

Números multiplicadores.

Muchos comandos aceptan un número multiplicador antes del comando. La acción es idéntica a invocar el comando tantas veces como indica el multiplicador.

Ejemplos:

- “10j” en modo comando avanza 10 líneas;
- “5Y” copia 5 líneas y las retiene para luego pegar.



Ejemplos de manejo.

Los siguientes ejemplos de manejo asumen que el editor se encuentra en modo comando.

- flechas mueven el cursor (si el terminal lo permite)
- “h, j, k, l” mueven el cursor (igual que las flechas)
- “itexto” + ESC inserta la palabra "texto" y vuelve a comando
- “x” borra el carácter sobre el cursor
- “dw” borra una palabra
- “dd” borra una línea
- “3dd” borra las 3 líneas siguientes
- “u” deshace último cambio
- “ZZ” graba cambios y sale de vi
- “:q!” +ENTER sale de vi sin grabar cambios
- “/expresión” +ENTER busca la expresión indicada
- “3Y” copia 3 líneas para luego pegar
- “:6r arch3” inserta debajo de la línea 6 el archivo arch3

Movimiento del cursor:

- “h o BS” una posición hacia la izquierda
- “l o SP” una posición hacia la derecha
- “k o -” una línea hacia arriba
- “j o +” una línea hacia abajo
- “\$” fin de línea
- “0” principio de línea
- “1G” comienzo del archivo
- “G” fin del archivo
- “18G” línea número 18
- “Ctrl-G” mostrar número de línea actual
- “w” comienzo de la palabra siguiente
- “e” fin de la palabra siguiente
- “E” fin de la palabra siguiente antes de espacio
- “b” principio de la palabra anterior



- “^” primera palabra de la línea
- “%” hasta el paréntesis que aparea
- “H” parte superior de la pantalla
- “L” parte inferior de la pantalla
- “M” al medio de la pantalla
- “23|” cursor a la columna 23

Control de pantalla.

- “Ctrl+f” una pantalla adelante
- “Ctrl+b” una pantalla atrás
- “Ctrl+l” redibujar la pantalla
- “Ctrl+d” media pantalla adelante
- “Ctrl+u” media pantalla atrás

Ingreso en modo texto:

- “i” insertar antes del cursor
- “I” insertar al principio de la línea
- “a” insertar después del cursor
- “A” insertar al final de la línea
- “o” abrir línea debajo de la actual
- “O” abrir línea encima de la actual
- “R” sobrescribir (cambiar) texto

Borrar

- “x” borrar carácter bajo el cursor
- “dd” borrar línea, queda guardada
- “D” borrar desde cursor a fin de línea



- “dw” borrar desde cursor a fin de palabra
- “d\$” borrar desde cursor a fin de línea
- “d0” borrar desde cursor a principio de línea

Copiar y pegar

- “Y o yy” copiar línea
- “P” pegar antes del cursor
- “p” pegar después del cursor
- “yw” copiar palabra
- “y\$” copiar de cursor a fin de línea
- “ap” pegar desde buffer 'a', a la derecha del cursor
- “aP” pegar desde buffer 'a', a la izquierda del cursor
- “bdd” borrar línea y guardar en buffer 'b'
- “bdw” borrar palabra y guardar en buffer 'b'

Búsqueda

- “/str” buscar hacia adelante cadena de caracteres 'str'
- “?str” buscar hacia atrás cadena de caracteres 'str'
- “n” repetir último comando / o ?
- “N” repetir último comando / o ? para el otro lado
- “fc” buscar el siguiente carácter 'c' en la línea
- “Fc” buscar el anterior carácter 'c' en la línea
- “tc” ir al carácter anterior al siguiente 'c'
- “Tc” ir al carácter posterior al precedente 'c'
- “;” repetir el último comando
- “,” último comando en orden inverso

La cadena a buscar en “/” o “?” puede ser una expresión regular.



Reemplazo

Estos comandos admiten multiplicadores: un número delante del comando. Al dar un comando de reemplazo el editor coloca un símbolo \$ en donde termina el pedido de reemplazo. El usuario escribe normalmente, sobre escribiendo, hasta donde necesite, y sale con ESC. Estos comandos admiten multiplicadores: 3cw abre un área de reemplazo para 3 palabras.

- “c” reemplaza caracteres
- “cw” reemplaza palabras
- “C o c\$” reemplaza hasta el fin de línea
- “c0” reemplaza desde el comienzo de línea

Otros

- “J” unir dos líneas en una
- “ZZ” grabar cambios si los hubo y salir
- “u” deshacer última acción
- “U” deshacer todos los cambios en una línea

Modo Texto.

- “BS” borrar carácter hacia la izquierda
- “ESC” pasar a modo comando

Modo ex o última línea.

- “:q” salir si no hubo cambios
- “:q!” salir sin guardar cambios
- “:w” guardar cambios
- “:w arch1” guardar cambios en archivo arch1
- “:wq” guardar cambios y salir
- “:r arch2” insertar un archivo



- “:e arch2” editar un nuevo archivo
- “:e! arch2” idem sin salvar anterior
- “:r! comando” insertar salida de comando
- “:shell” salir al shell (vuelve con exit)

Mover

- “:1” mueve a línea 1
- “:15” mueve a línea 15
- “:\$” mueve a última línea

Opciones

- “:set” cambio de opciones
- “:set nu” mostrar números de línea
- “:set nonu” no mostrar números de línea
- “:set showmode” mostrar modo actual de vi
- “:set noshowmode” no mostrar modo actual de vi

Reemplazo

La sintaxis del comando de búsqueda y reemplazo es la siguiente:

“:<desde>,<hasta>s/<buscar>/<reemplazar>/g”

“<desde>,<hasta>” indican líneas en el archivo;
“<buscar>,<reemplazar>” son cadenas de caracteres o expresiones regulares;
“/” es un separador,
“s” (sustituir) y “g” (global) son letras de comando para el manejo de expresiones regulares.



“:1,\$s/Martes/martes/g”

Cambia “Martes” por “martes” en todo el archivo.

“:.,5s/ayuda/&ndo/g”

Cambia “ayuda” por “ayudando” desde línea actual hasta la 5a. línea.

Tipo de terminal.

Vi es independiente del tipo de terminal, pero la variable de ambiente TERM debe estar fijada correctamente. Si no se conoce o no existe el tipo exacto de terminal, en la mayoría de los terminales remotos el tipo ansi da buenos resultados. Para fijar el terminal en tipo ansi, ingresaremos “TERM=ansi” y luego “export TERM”.

Algunos comandos, especialmente “more” y a veces “vi”, pueden no responder bien en la terminal o el emulador que se está usando. En estos casos, puede usarse “Ctrl+L” para refrescar la pantalla.

```
[root@localhost ~]#  
[root@localhost ~]#  
[root@localhost ~]# echo $TERM  
linux  
[root@localhost ~]# _
```



```
[root@localhost ~]# TERM=ansi
[root@localhost ~]# export TERM
[root@localhost ~]# echo $TERM
ansi
[root@localhost ~]# _
```

En principio la variable TERM estaba configurada como tipo “Linux” lo cual también sirve, fue cambiada solamente a modo de ejemplo.

El comando Diff

El comando “diff” analiza línea por línea y muestra una lista de cambios entre dos archivos. Es un comando muy útil para encontrar modificaciones entre distintas versiones de mismos documentos.

EJEMPLO:

Creemos dos archivos file1.txt y file2.txt e introduzcamos la siguiente información.

Información en file1.txt	Información en file2.txt
HIOX TEST	HIOX TEST
hscripts.com	HSCRIPTS.com
with friends	with friend
Linux rocks	linux Rocks
hiox ARG	



Comparemos ambos archivos ignorando espacios en blanco

```
[root@localhost ~]# diff -w file1.txt file2.txt
4,5c4
< Linux rocks
< hiox ARG
---
> linux Rocks
[root@localhost ~]# _
```

La salida se lee de la siguiente forma:

- < - denota líneas que cambian en file1.txt respecto de file 2
- > - denota líneas que cambian en file2.txt respecto de file 1

La línea que leen al principio “4,5c4” denota las líneas de cada archivo en las que existen las diferencias marcadas debajo:

- La letra “a” significa “adding” o agregado.
- La letra “c” significa cambio.

Si por ejemplo utilizamos el flag “-i” que ignora mayúsculas de minúsculas, lo que obtendremos es lo siguiente:



```
Centos7 [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda

[root@localhost ~]# diff -i file1.txt file2.txt
5d4
< hiox ARG
[root@localhost ~]# _
```

Otra forma de salida que puede ser un poco más amigable es utilizando el flag “-u”

```
[root@localhost ~]# diff -i file1.txt file2.txt
5d4
< hiox ARG
[root@localhost ~]# diff -u file1.txt file2.txt
--- file1.txt      2017-10-03 15:13:45.300129386 -0300
+++ file2.txt      2017-10-03 15:16:13.364682539 -0300
@@ -1,5 +1,4 @@
  HIOX TEST
  hscripts.com
  with friends
-Linux rocks
-hiox ARG
+linux Rocks
[root@localhost ~]# _
```



Ejercicios con Vi

1. Crear un archivo de texto breve. Realizar agregados y correcciones elementales usando los comandos de la Guía de Supervivencia. Grabar y volver a editar ese archivo. Memorizar estos comandos.
2. Crear un archivo de texto breve o usar el anterior (recordar la redirección de archivos). Probar los comandos descritos. Marcar los comandos que se consideren de uso más frecuente dentro de cada categoría de uso.

El editor no interactivo sed

El nombre sed significa *stream editor* (editor de flujo continuo). Su entrada es un flujo de información que se procesa automáticamente según sus instrucciones. Sed es un editor de textos no interactivo que puede realizar todas las funciones básicas de edición.

La sintaxis de sed es sencilla:

“sed 'lista de comandos' archivos”

Las órdenes sed leen una línea a la vez de los archivos de entrada, le aplican los comandos de la lista, por orden, a cada línea y muestran la modificación en la salida estándar.

Por ejemplo, se puede escribir un script sed para cambiar todas las apariciones de la palabra TOTO a Toto con la siguiente orden:

“sed 's/TOTO/toto/' archive”

Luego podrá utilizar este script para editar automáticamente cualquier fichero que desee.



Un archivo con un solo cambio sólo necesita una orden simple. Sin embargo, no es raro crear scripts sed más largos para realizar más de una tarea de edición.

Cuando los comandos sed se almacenan en un archivo para ejecutarlos escribimos:

“sed -f script archivos”

A continuación se resumen los comandos de edición básicos de sed.

En dicha tabla, *addr1*, *addr2* representan el rango de líneas desde *addr1* hasta *addr2* incluidas.

Los comandos que aceptan un rango también pueden trabajar sobre una única línea.

<u>Comando</u>	<u>Función</u>	<u>Uso</u>	<u>Significado</u>
a	Añadir	addr1,addr2 a\text	añade <i>text</i> después de addr2
i	Insertar	addr1,addr2 i\text	inserta <i>text</i> antes de addr1
c	Cambiar	addr1,addr2 c\text	reemplaza las líneas con <i>text</i>
d	Borrar	addr1,addr2 d	Borra las líneas especificadas
s	Sustituir	addr1,addr2	Reemplaza el patrón s/patrón/ <i>text</i>
q	Salir	addr q	Termina en la línea addr
r	Leer	addr r fichero	Lee del archivo antes de la línea \$addr\$
w	Escribir	addr1,addr2 w fichero	Graba las líneas especificadas en fichero



Ejercicios con Sed

Copiar el archivo index.html que se encuentra en /usr/share/doc/HTML/ al home de nuestro usuario para realizar los siguientes ejercicios utilizando sed:

1. Sustituya la cadena de palabras “Welcome to CentOs” por “Bienvenidos a mi Apache!”
2. Sustituya la cadena de palabras “White” por la cadena “Pink”
3. Ingresar a la instancia de apache que hemos configurado anteriormente y documentar los cambios.

GNU AWK

[GNU awk](#) es una herramienta muy útil para modificar archivos, buscar y transformar datos y, en general, realizar cualquier tipo de tratamiento masivo de archivos. Con un programa awk es posible contar el número de líneas de un archivo, seleccionar columnas, aplicar filtros, realizar cruces, borrar el último campo de cada línea, hacer sumas, comprobar duplicados, muestreos, etc.

Awk presenta la siguiente estructura de reglas:

“patrón {acción}”

Donde el patrón es una serie de caracteres o expresión regular y la acción es lo que deseamos que se ejecute sobre ese patrón.

En una regla puede omitirse el patrón o la acción, pero no ambos. Si el patrón se omite, entonces la acción se realiza para todas las líneas. Si se omite la acción, la acción por defecto es imprimir las líneas que cumplan el patrón.



Imaginemos un archivo llamado fechas.txt que contiene el texto siguiente:

Jan 13 25 15 115

Feb 15 32 24 226

Mar 15 24 34 228

Jun 1 26 57 232

Apr 31 52 63 420

May 16 34 29 208

Jan 18 29 25 101

Jun 31 42 75 492

Por ejemplo, para las líneas que contienen “Jan” las imprimimos por pantalla (simulamos comando ‘grep’):

```
“awk '/Jan/ { print $0 }' fechas.txt”
```

“\$0” implica imprimir todas las columnas, en cambio Si quisiéramos únicamente imprimir la segunda columna utilizamos \$2 (el resto las podemos encontrar en \$3, \$4, etc.):

```
“awk '/Jan/ { print $1 }' fechas.txt”
```

También podemos realizar sumas, por ejemplo, sumamos los valores de la quinta columna de las líneas que tengan en la primera columna “Jan”:

```
“awk '$1 == "Jan" { sum += $5 } END { print sum }' fechas.txt”
```

Como se puede observar, se han definido 2 reglas:

- Si la primera columna es “Jan” entonces acumulamos el quinto valor en una variable.
- Si es el final del fichero (END {...}), imprimimos el valor de la variable.



Es muy habitual ver ejemplos de AWK en una única línea, y justamente por ese motivo se ha ganado la fama de complejo. Pero en realidad podemos crear ficheros de texto con todas las instrucciones bien tabuladas y que permiten una mejor comprensión del objetivo del programa. Por ejemplo, podemos crear el archivo 'test.awk':

```
#!/usr/bin/awk -f  
$1 == "Jan" { sum += $5 }  
END { print sum }
```

Los comentarios dentro del programa awk se marcan con "#".

Para ejecutarlo, utilizaremos la siguiente sentencia:

```
awk -f test.awk fechas.txt
```

Expresiones

Operadores aritméticos

Suma: $x+y$

Resta: $x-y$

Negación: $-x$

Multiplicación: $x*y$

División: x/y

Resto: $x\%y$

Exponente: $x**y$



Patrones

En awk se pueden especificar los patrones de varias formas, por ejemplo:

- `$i == ""`, se está comprobando si el campo `$i` está vacío.
- `$i ~ /^$/`, el símbolo `~` se utiliza para indicar la concordancia con una expresión regular.
 - La expresión regular ha de ir encerrada entre las diagonales.
 - En este caso se pregunta por la concordancia con una línea vacía.
- `$i !~ /./`, los símbolos `!~` significan "no corresponde" en contraposición al anterior.
 - Este ejemplo, pregunta si el campo `$i` no concuerda con ningún carácter.
- `length($i) == 0`, `length` es una función predefinida en awk, que contiene la longitud de una cadena de caracteres.

Salida por pantalla

AWK normalmente supone que el espacio en blanco (cualquier número de blancos y tabuladores) es el separador de campos.

Este separador puede cambiarse a cualquier carácter mediante la opción `-F` (mayúscula) de la línea de comandos.

La variable predefinida `NR` es el número del registro o línea actual de la entrada. Por ejemplo, para añadir números de línea a un fichero de entrada su puede usar:

```
awk '{print NR, $0}'
```

En una proposición `print`, los elementos separados por comas se imprimen separados por el separador de campos de salida, que por defecto es un *blanco*.



Si se desea modificar la salida por defecto, se ha de utilizar la proposición `printf` que acepta del mismo conjunto de formato que en el lenguaje C. Por ejemplo, para imprimir los números de línea en un campo de cuatro dígitos, se podría usar la orden:

```
awk '{ printf "%4d %s \n", NR, $0 }'
```

`%4d` indica que se NR se imprima como un número decimal de cuatro dígitos de ancho, `%s` especifica una cadena de caracteres, en este caso, `$0`. Finalmente `\n` introduce un carácter de nueva línea, puesto que `printf` por defecto no lo tiene asociado.

Ejercicios

1. Utilizando el comando “`yum`” listar todos los paquetes instalados en el sistema y guardarlo en un archivo llamado `paquetes.txt`.
2. Utilizar AWK para contar cuantas veces aparece la palabra “`yum`”.

Links Recomendados

- <https://www.gnu.org/software/gawk/manual/gawk.html>
- <https://www.gnu.org/doc/doc.html>
- <http://www.theunixschool.com/p/awk-sed.html>