

Editores de texto en Linux

Miguel Ortuño
Escuela de Ingeniería de Fuenlabrada
Universidad Rey Juan Carlos

Septiembre de 2023



© 2023 Miguel Angel Ortuño Pérez.
Algunos derechos reservados. Este documento se distribuye bajo la
licencia *Atribución-CompartirIgual 4.0 Internacional* de Creative
Commons, disponible en
<https://creativecommons.org/licenses/by-sa/4.0/deed.es>

Introducción

- Los **editores de texto** crean y modifican ficheros de texto *plano*
Se emplea en programación y en configuración de sistemas
- Los **procesadores de texto** crean y modifican ficheros de texto con formato de fuente (negritas, cursivas, tipos de letra, etc), de página (interlineado, márgenes, etc) e imágenes

En cualquier Linux hay disponibles muchos editores

¿Cuál es mejor?

- Depende en buena parte de gustos personales
- Depende de dónde vayamos a usarlos
- Este es un asunto típico para *guerras de religión*



Tipos de editor de texto

1 Editores en modo gráfico

- Su curva de aprendizaje suele ser más suave
- Adecuados para trabajar como programador en un ordenador *estándar*, local y con gráficos

2 Editores en modo texto (editores de consola)

- Curva de aprendizaje más dura (excepto algunos muy sencillos/simplones)
- Permiten trabajar en remoto con la misma facilidad que en local
 - Podemos administrar sin problemas nuestra máquina Linux p.e. desde un Windows prestado y con mala conexión. O incluso una PDA y un teléfono móvil
- Son los únicos disponibles en sistemas empujados, como routers
- Suelen ser los únicos disponibles en ordenadores a medio instalar, averiados, herramientas de rescate, etc

El editor *estándar* en Unix. Desarrollado por Bill Joy (Co-fundador de Sun Microsystems) en el año 1976.

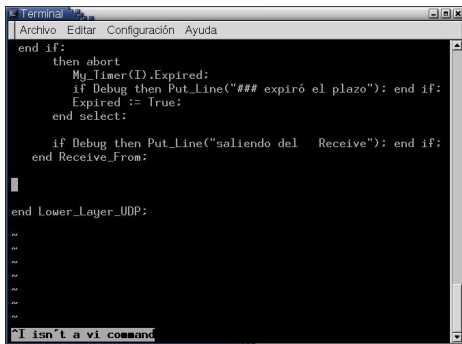
Hoy usamos clones, especialmente vim o más recientemente, neovim

- Si no nos gusta vi/vim/neovim, casi siempre podremos instalar otro
- Pero para poder instalar otro, suele ser imprescindible manejar al menos las órdenes elementales de vi

Ventajas

- Normalmente estará disponible y funcionando en cualquier máquina Unix
- Hay versiones para la mayoría de los SSOO (Windows, macOS...)
- Es muy flexible y potente, conociéndolo bien se puede trabajar a gran velocidad
- Pensado para sesiones remotas con malas conexiones en un *terminal tonto* de los años 70, el ADM-3A
- Si trabajamos en una máquina con gráficos, puede ser conveniente usar un vim en modo gráfico, mejor integrado con el escritorio. Permitirá usar el ratón, funcionará el portapapeles del escritorio y podrá tener menús, de utilidad para ordenes que aún no hemos memorizado
 - En Windows, gvim
 - En Linux, gvim ¹
 - En OS X, MacVim (mvim)

¹el nombre del paquete es vim-gtk

A screenshot of a terminal window titled "Terminal" with a menu bar containing "Archivo", "Editar", "Configuración", and "Ayuda". The terminal displays code in a vi editor. The code includes: "end if:", "then abort", "My_Timer(I).Expired:", "if Debug then Put_Line("### expiró el plazo"): end if:", "Expired := True:", "end select:", "if Debug then Put_Line("saliendo del Receive"): end if:", "end Receive_From:", "end Lower_Layer_UDP:", and several lines of dots representing continuation. At the bottom, the prompt "^I isn't a vi command" is visible.

```
end if:
  then abort
  My_Timer(I).Expired:
  if Debug then Put_Line("### expiró el plazo"): end if:
  Expired := True:
  end select:

  if Debug then Put_Line("saliendo del Receive"): end if:
end Receive_From:

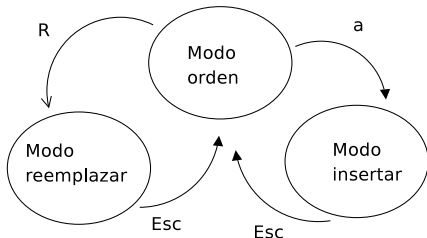
end Lower_Layer_UDP:
~
~
~
~
~
~
~
~
~
~
^I isn't a vi command
```

Inconvenientes

- Interfaz de usuario muy anticuada, el usuario debe memorizar órdenes ¡donde hasta las mayúsculas son significativas!

Modos de vi

- 1 Modo orden (también llamado modo comando, modo normal)
En este modo guardamos el fichero, leemos otro, salimos, copiamos, pegamos, etc
- 2 Modo insertar (también llamado modo texto o modo entrada)
En este modo insertamos texto
- 3 Modo reemplazar (también llamado modo texto o modo entrada, sin distinguirlo del modo insertar)
En este modo reemplazamos texto



Órdenes imprescindibles

Desde la shell

```
koji@mazinger:~$ vi nombre_fichero.txt
```

(Edita el fichero del nombre indicado. Si no existe, lo crea)

Desde vi

a Pasar de modo orden a modo insertar

R Pasar de modo orden a modo reemplazar

Esc Volver a modo orden

x Borrar un carácter

J Unir la línea actual con la línea siguiente

:wq Escribir el fichero y salir

:q! Salir sin guardar el fichero

Este conjunto de órdenes es suficiente para editar cualquier fichero

Órdenes básicas

<code>:r nombre</code>	leer un fichero
<code>:w nombre</code>	escribir fichero
<code>u</code>	Deshacer último cambio
<code>ctrl r</code>	Rehacer lo último deshecho
<code>D</code>	Borrar hasta final de línea
<code>dd</code>	Borrar línea actual
<code>yy</code>	copiar (yanc) línea
<code>p</code>	pegar lo último copiado o borrado
<code>.</code>	Repetir la última orden
<code>/patron</code>	Busca un patrón (hacia adelante)
<code>n</code>	Repetir búsqueda
<code>N</code>	Buscar en dirección inversa a anterior
<code>G</code>	Ir a Final del archivo
<code>5G</code>	Ir a línea 5
<code>%</code>	Salta al paréntesis que se corresponda con el paréntesis actual (o llave, corchete...)

Casi todas las órdenes permiten anteponer un número, que indica cuántas veces se repetirá

<code>dd</code>	Borrar línea actual
<code>10dd</code>	Borrar 10 líneas
<code>u</code>	Deshacer un cambio
<code>3u</code>	Deshacer últimos 3 cambios
<code>cw</code>	Cambiar una palabra
<code>5cw</code>	Cambiar 5 palabras

Otras órdenes

O	ir a principio línea
\$	ir a fin línea
w	ir a siguiente palabra
b	ir a palabra anterior
r	Sustituir 1 carácter
cw	Cambiar palabra (change word)
dw	Borrar hasta fin palabra (delete word)
yw	Copiar palabra
*	Buscar palabra igual a la palabra sobre la que está el cursor
ma	Poner marca de texto a
mb	Poner marca de texto b
'a	ir a marca a
'b	ir a marca b
Ctrl G	Indicar línea actual
~	Pasar de may. a minusc. o al revés

<code>:49,53 w! fichero</code>	Escribir en fichero líneas de 49 a 53
<code>:. ,53 w! fichero</code>	Escribir en fichero desde línea actual hasta línea 53
<code>:1,\$ s/digo/diego/g</code>	Buscar todas las cadenas "digo" desde la línea 1 hasta el final, y reemplazarlas por "diego"
<code>:set nu</code>	Indicar el nº de línea
<code>:set nonu</code>	Desactivar nº de línea
<code>:set ic</code>	Ignore case (Insensible a mayus/min)
<code>:set noic</code>	Desactiva ic

Podemos configurar vim de forma persistente creando un fichero de configuración

- En Unix/Linux
`~/.vimrc`
- En Windows 10 / Windows 11, vim 64 bits
`c:\Archivos de programa\vim_vimrc`
- En Windows 10 / Windows 11, vim 32 bits
`c:\Archivos de programa (x86)\vim_vimrc`

Por ejemplo, el fichero de configuración puede contener:

```
set vb  
set ic  
set tabstop=4  
syntax on
```

Esto activa la *visual bell* (que elimina los molestos pitidos del terminal), ignora mayúsculas/minúsculas, fija el tabulador en 4 espacios y colorea el texto si reconoce la sintaxis

En Windows podemos añadir

```
set enc=utf-8  
set guifont=Consolas:h12:cANSI:qDRAFT
```

De esta forma, empleará por omisión la misma codificación que en Unix/Linux. Y el tipo de letra tendrá un tamaño razonable
Para más información sobre vi, consulta la página web *vi lovers home page*

Editores ligeros

Hemos visto que vi tiene muchas ventajas. Pero si nos *asusta* su interfaz de usuario y necesitamos un editor en modo texto, disponemos de editores ligeros como

- mcedit (editor del mc, midnight commander)
- nano (clon de pico)
- joe

Emacs / XEmacs

Editor clásico en Unix. Uno de los más conocidos, se populariza a mediados de los 80

Emacs trabaja en modo texto, XEmacs en modo gráfico

Ventajas

- Completísimo, es mucho más que un editor. Permite leer correo, news, se integra con gran cantidad de herramientas...
- Módulos para muchos lenguajes de programación
- Da formato y color al fuente, con mucha calidad.
- Completamente personalizable (en lisp)
- Puede emular a vi

Inconvenientes

- Muy grande y pesado, consume muchos recursos.
- Su uso resulta complicado
- Aún para las tareas sencillas, tiene alguna peculiaridad que lo hace poco intuitivo al usuario actual

Usando emacs

```

emacs@papagano.dat.escet.urjc.es
-----
Buffers Files Tools Edit Search Hule TeX Help
\begin{figure}
\centerline{\includegraphics[width=5cm]{figs/xemacs}}
\end{figure}
\end{minipage} \hfill

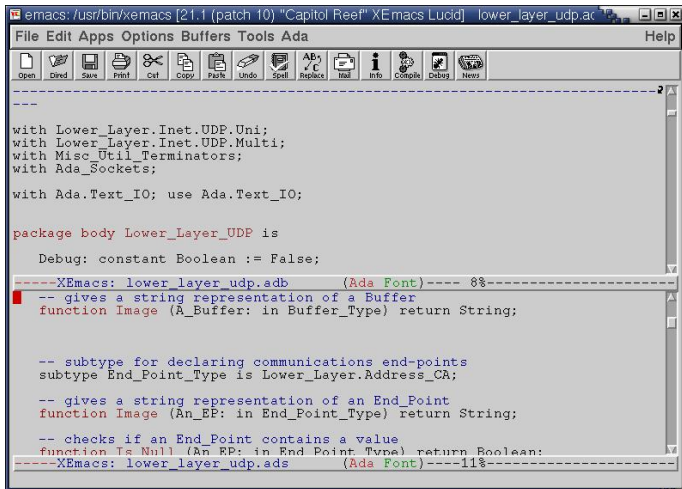
\begin{minipage}{4cm}
\begin{itemize}
\item menu
\item pantalla edición
\item línea de modo
\item línea comandos
\end{itemize}
\end{minipage} \hfill
\end{hslide}

\begin{hslide}
\slsubject{emacs $\neq$ xemacs}
\begin{figure}
\centerline{\includegraphics[width=9cm]{figs/emacs}}
\end{figure}
\end{hslide}

\begin{hslide}
\slsubject{atajos de teclado}
\end{hslide}
!r: editores.tex (E[ font) - 14 / - 49%
wrote /home/jmplaza/docencia/cursos/1maf.2002/editores/editores.tex
  
```

- menu
- pantalla edición
- línea de modo
- línea comandos

emacs ≠ xemacs

The image shows a screenshot of the XEmacs text editor window. The title bar reads "emacs: /usr/bin/xemacs [21.1 (patch 10) 'Capitol Reef' XEmacs Lucid] lower_layer_udp.ac". The menu bar includes "File Edit Apps Options Buffers Tools Ada" and "Help". The toolbar contains icons for Open, Direct, Save, Print, Cut, Copy, Paste, Undo, Spell, Replace, Mail, Info, Compile, Debug, and News. The main text area contains Ada code with syntax highlighting. A red cursor is positioned at the start of a comment line. The code includes package declarations, a package body, and several functions with comments. The code is as follows:

```
-----XEmacs: lower_layer_udp.adb (Ada Font)----- 8%-----
-- gives a string representation of a Buffer
function Image (A_Buffer: in Buffer_Type) return String;

-- subtype for declaring communications end-points
subtype End_Point_Type is Lower_Layer.Address_CA;
-- gives a string representation of an End_Point
function Image (An_EP: in End_Point_Type) return String;

-- checks if an End_Point contains a value
function Is_Null (An_EP: in End_Point_Type) return Boolean;
-----XEmacs: lower_layer_udp.ads (Ada Font)-----11%-----
```

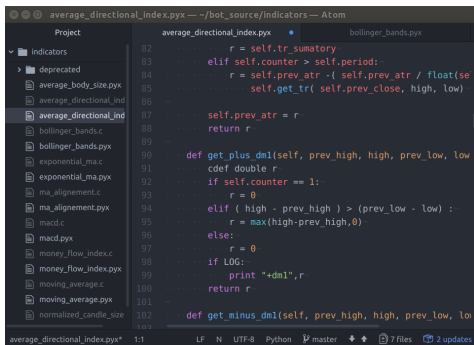
Atajos de teclado

- CTRL-K borrar linea
- ESC-X query-replace, ESC-X replace
- ESC-X goto-line
- CTRL-X-S salvar
- CTRL-X-F encontrar fichero
- CTRL-W=cortar, CTRL-Y=pegar
- CTRL-@=marca

Enlaces sobre Emacs/XEmacs

- Emacs <http://www.gnu.org/software/emacs>
- XEmacs <http://www.xemacs.org>

Atom



```
average_directional_index.pyx
82     r = self.tr_sumatory
83     elif self.counter > self.period:
84         r = self.prev_atr -( self.prev_atr / float(se
85             self.get_tr( self.prev_close, high, low)
86
87     self.prev_atr = r
88     return r
89
90 def get_plus_dm1(self, prev_high, high, prev_low, low
91 cdef double r
92 if self.counter == 1:-
93     r = 0
94 elif ( high - prev_high ) > (prev_low - low) :
95     r = max(high-prev_high,0)
96 else:
97     r = 0
98 if LOG:
99     print "+dm1",r
100 return r
101
102 def get_minus_dm1(self, prev_high, high, prev_low, low
```

- Editor de texto, libre y gratuito, disponible para Windows, Linux y macOS

Ventajas

- Más que un editor, es un IDE (Integrated development environment) con mucha funcionalidad: da formato, color, autocompleta, se integra con el compilador, con git, incluye colaboración en tiempo real (teletype)
- Ampliable mediante paquetes, que se pueden instalar desde el terminal (apm)
- Desarrollado por GitHub
- Moderno: la primera versión es de 2014, se ha vuelto muy popular

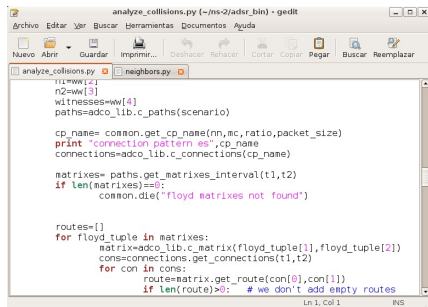
Inconvenientes

- Exige una sesión gráfica

enlaces

- `https://atom.io/`

gedit



```
analyze_collisions.py (-/ms-2/adrs_bin) - gedit
Archivo Editar Ver Buscar Herramientas Documentos Ayuda
Nuevo Abrir Guardar Imprimir... Deshacer Rehacer Cortar Copiar Pegar Buscar Reemplazar
analyze_collisions.py neighbors.py
t1=ww[2]
n2=ww[3]
witnesses=ww[4]
paths=adco_lib.c_paths(scenario)

cp_name= common.get_cp_name(nn,mc,ratio,packet_size)
print "connection pattern es",cp_name
connections=adco_lib.c_connections(cp_name)

matrixes= paths.get_matrixes_interval(t1,t2)
if len(matrixes)==0:
    common.die("floyd matrixes not found")

routes=[]
for floyd_tuple in matrixes:
    matrix=adco_lib.c_matrix(floyd_tuple[1],floyd_tuple[2])
    cons=connections.get_connections(t1,t2)
    for con in cons:
        route=matrix.get_route(con[0],con[1])
        if len(route)>0: # we don't add empty routes
            routes.append(route)
```

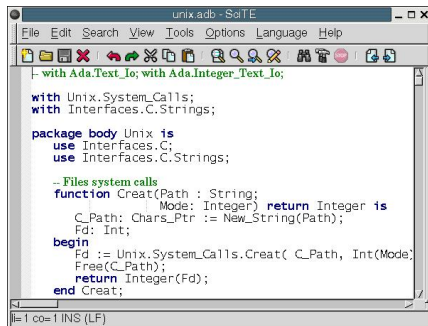
Editor de texto de propósito general, es el *block de notas* de gnome

Ventajas

- Muy sencillo y fácil de manejar

Inconvenientes

- Exige una sesión gráfica
- Ha mejorado mucho, pero sigue teniendo poca funcionalidad
- Tal vez no sea la mejor opción si tenemos disponible editores como atom, scite...

A screenshot of the SciTE text editor window. The title bar reads "unix.adb - SciTE". The menu bar includes "File", "Edit", "Search", "View", "Tools", "Options", "Language", and "Help". The toolbar contains icons for file operations (new, open, save, close, print), editing (undo, redo, cut, copy, paste), and search (find, replace). The main text area contains Ada code with syntax highlighting: comments in green, keywords in blue, and identifiers in black. The code includes package declarations, uses, and a function definition. The status bar at the bottom shows "||=1 co=1 INS (LF)".

```
|- with Ada.Text_IO; with Ada.Integer_Text_IO;  
  
with Unix.System_Calls;  
with Interfaces.C.Strings;  
  
package body Unix is  
  use Interfaces.C;  
  use Interfaces.C.Strings;  
  
  -- Files system calls  
  function Creat(Path : String;  
                Mode: Integer) return Integer is  
    C_Path: Chars_Ptr := New_String(Path);  
    Fd: Int;  
  begin  
    Fd := Unix.System_Calls.Creat( C_Path, Int(Mode);  
    Free(C_Path);  
    return Integer(Fd);  
  end Creat;
```

Editor de texto multiplataforma **Ventajas**

- Muy completo: Da formato, color, se integra con el compilador...
- Versiones para Win32 y X Window
- Muy fácil de manejar
- Es el editor de *anjuta*, el IDE de gnome

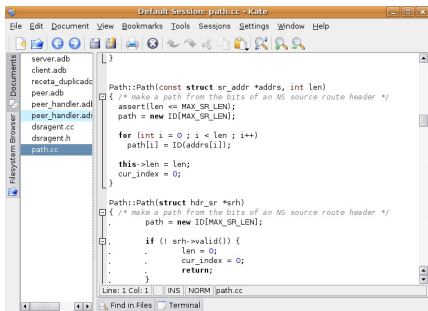
Inconvenientes

- Exige una sesión gráfica
- No muy extendido
- Hay editores más avanzados

enlaces

- <http://www.scintilla.org/SciTE.html>

Kate



```
server.adb
client.adb
receta_duplicad
peer.adb
peer_handler.ad
peer_handler.ad
dsragent.cc
dsragent.h
path.cc

Path::Path(const struct sr_addr *addrs, int len)
{ /* make a path from the bits of an NS source route header */
  assert(len <= MAX_SR_LEN);
  path = new ID[MAX_SR_LEN];
  for (int i = 0; i < len; i++)
    path[i] = ID(addrs[i]);

  this->len = len;
  cur_index = 0;
}

Path::Path(struct hdr_sr *srh)
{ /* make a path from the bits of an NS source route header */
  path = new ID[MAX_SR_LEN];
  if (!srh->valid()) {
    len = 0;
    cur_index = 0;
    return;
  }
}
```

Es el editor del escritorio
KDE

Ventajas

- Muy completo: Da formato, color, se integra con el compilador...
- Muy buen *pretty printing*
- Muy fácil de manejar

Inconvenientes

- Exige una sesión gráfica
- No muy extendido
- Hay cosas editores más avanzados hacen mejor
- Es necesario tener instalado KDE (o al menos buena parte)
- No disponible en otras plataformas

Enlaces

- <http://kate-editor.org>