Entornos de desarrollo universitarios : Emacs

Eduardo Cermeño GTQ
Asociación para el Fomento del Software Libre
UAM

September 8, 2003

Abstract

Esta ponencia trata sobre GNU Emacs (Emacs en adelante), la implementación de Emacs realizada por GNU. La licencia del programa es GPL lo que nos permite la libre distribución, copia y modificación del mismo. Se puede adquirir gratuitamente en la web www.gnu.org tanto para plataformas Unix (Linux) como para Windows NT, Windows 95, y MS-DOS. Estas características nos interesan mucho porque permiten que sea una herramienta muy interesante para el desarrollo en las Universidades. No es el objetivo de esta ponencia hablar de software libre en general pero como Emacs lo es y ello aporta beneficios vamos a comentarlos.

Al ser gratuito el coste de adquisición es nulo. Como podemos copiarlo y distribuirlo libremente se puede generalizar su uso sin ningún coste económico. La posibilidad de modificarlo puede ser útil en ciertos ambientes de investigación. Finalmente su alta disponibilidad facilita el intercambio de documentos entre distintas plataformas y la adaptación de los usuarios a un nuevo entorno. Es genial poder usar Emacs allá donde vayas Windows o Linux.

La definición más común de Emacs es : 'advanced, self-documenting, customizable extensible real-time display editor' La parte de real-time display no es muy interesante puesto que se refiere a que todo lo que se inserte aparece en pantalla de forma casi immediata. El resto de características de la definición nos van a resultar muy útiles para mostrar las aplicaciones de Emacs.

1 Emacs: Editor configurable (customizable)

Emacs ofrece la posibilidad de modificar sus keymaps. El keymap o mapa de teclas es lo que caracteriza a un editor. Emacs por ejemplo utiliza las siguientes combinaciones de teclas para cortar y pegar ctrl-space ctrl-w (para cortar) y ctrl-y (para pegar).

Emacs posee distintos tipos de keymaps. El principal se denomina global y afecta a todos los modos de edición. Para cada modo denominado 'major' existe otro keymap que define las características propias del modo y finalmente para los modos 'minor' tenemos más keymaps que definen las teclas especiales de esos modos. Solo necesitamos tener una idea aproximada de lo que son los modos en Emacs, y espero que según avancemos esta idea se concrete poco a

poco. Las definiciones global se pueden ver como la base de la pirámide sobre la que se basa todo el resto y sin la cual no existiria nada. Un poco como el sistema operativo para un ordenador. Los major modes se sobreponen al global de forma que aportan novedades, es decir nuevas utilidades y en caso de conflicto tienen mayor prioridad porque consideramos que estamos adaptando nuestro editor a nuestras necesidades concretas que ya no son tan generales. Finalmente los minor modes son una capa superior que con la misma lógica afinan más el editor.

Podemos modificar cualquiera de estos keymaps de forma sencilla mediante los comandos : M-x global-set-key o M-x local-set-key

Estas modificaciones incluyen tanto la redefinición de teclas como la definición de nuevas secuencias de teclas que ejecuten nuevas acciones.

Otra forma interesante de configurar Emacs es redefiniendo sus 'variables'. Estas variables determinan de muchas formas el comportamiento del editor puesto que definen valores clave para Emacs. Por ejemplo cuando se realiza el wrapping en Emacs?. Pues el valor que determina esto se encuentra en la variable 'fill-column' que podemos cambiar a nuestro gusto con el comando M-x set-variable. (Para más detalles consultar el manual on-line sobre la variable que se quiera modificar tecleando C-h v).

Estos son los conceptos básicos sobre la configuración del editor. Debería empezar a quedar claro que en Emacs podemos acceder a casi todo. Esto se debe a que la mayor parte del código está escrito en LISP y nos permite crear nuevos comandos y modificar los ya existentes en una misma sesión, de forma interactiva.

Una pequea nota sobre la asignación de teclas. Conviene utilizar las combinaciones de tipo ctrl-c seguido por una letra porque este rango esta reservado a este fin. De otra forma podrían surgir conflictos.

2 Emacs: Auto-ayuda (self-documenting)

Hasta ahora hemos mostrado una diminuta parte de las posibilidades de Emacs, sin embargo puede parecer que el número de comandos que entrarán en juego va a ser muy grande. La verdad es que conocer diez o veinte comandos o secuencias de teclas suele bastar para la mayor parte de la actividades habituales. Pero para poder aprovechar de forma real y efectiva la potencia de Emacs tiene que existir un mecanismo de ayuda que nos permita aprender nuevas funciones a medida que las vayamos necesitando, o que simplemente nos recuerde como se utilizaba tal o tal función.

Si no recordamos la secuencia de teclas que realiza una función siempre podemos probar a teclear M-x y una parte del nombre que creemos pueda haber sido elegido para nombrar la función deseada. Por ejemplo si queremos hacer algo tan sencillo como borrar un carácter y no recordamos que hay que pulsar ctl-d, podemos teclear M-x delete y despúes una o dos veces el tabulador. Se abrirá una ventana nueva (entiendase como ventana una 'window' de Emacs es decir una subdivisión del 'frame' que es el objeto que se abre al iniciar Emacs) con posibles soluciones. Para nuestro ejemplo habremos acertado puesto que no

hay muchas y entre ellas se encuentra delete-char. Si solo hubiera una solución el hecho de pulsar el tabulador hubiera completado automáticamente el nombre de la función. Además del enorme juego que proporciona el tabulador Emacs posee una importante ayuda. Vamos a describir de forma muy breve una serie de comandos de ayuda de forma que nos hagamos una idea de como funcionan pero sin entrar en detalle.

C-h a topic RET Busca los comandos que tengan coincidencias con el topic C-h i d m emacs RET i topic RET Busca el topic en los indices del manual on-line de Emacs. Mostrando el primero por pantalla. C-h F Muestra la sección FAQ de Emacs donde se pueden realizar búsquedas como si se tratase de un documento normal de Emacs. C-h c seguido por una serie de teclas nos indica a que función están asociadas las teclas.

Siempre es interesante teclear C-h ? para ver el resto de ópciones que ofrece la ayuda de Emacs.

3 Emacs: Editor avanzado

Hemos presenatdo un editor altamente configurable y con una potente ayuda, ahora vamos a mostrar que detrás de la sobria pantalla de Emacs se esconde un editor muy potento lleno de funcionalidades. No hemos podido evitar hablar de las 'windows' de Emacs. De forma breve vamos a describir una sesión típica de Emacs. Abrir el editor con la forma deseada (emacs -nw si no se desea activar el entorno gráfico y queremos trabajar desde consola), una vez abierto abrir tantos archivos como queramos y al terminar cerraremos Emacs. Como se indica anteriormente el objeto que abre Emacs al arrancar se denomina 'frame'. Esto es independiente de si estamos en un entorno gráfico o no puesto que si lo estamos se abrirá la clásica ventanita de programa y sino el programa se ejecutará en consola. Esa es la única diferencía que vamos a notar al editar, a excepción de las funciones que aporta el ratón en entornos gráficos.

Una vez tengamos nuestro 'frame' podemos abrir tantos buffers como queramos. En general un buffer corresponde a un archivo pero también se crean buffers para mostrar todo tipo de ayuda, subprocesos...En cualquier momento podemos cambiar de buffer tecleando ctrl-x b o accediendo al menú Buffers (con la tecla F-10 o con el ratón). Al terminar con un buffer no tenemos más que 'matarlo' tecleando ctrl-x k. De esta forma en una única sesión podemos tener multitud de archivos y abiertos. Además podemos tener múltiples 'windows' de Emacs. Esto son divisiones del 'frame' que nos permite visualizar de forma simutánea varios buffers. Nótese que esto se puede realizar en consola sin ningún problema.

Las tareas básicas de un editor son insertar y borrar. Un editor avanzado debe incluir más funciones. Emacs ofrece multitud de ellas, desde las más comunes como copiar, pegar, cortar, buscar, reeemplazar, macros, mover bloques, a funciones más complejas y específicas como son los modos para lenguajes de programación.

3.1 Programación en C

Al cargar un archivo con extensión .c se carga automáticamente el modo C. Este modo conoce la estructura de los programas C y asigna un color diferente para diferenciar cometarios, instrucciones de precompilador, cadenas de caracteres, palabras reservadas .... El tabulador sangra el texto de forma que alinea los bloques condicionales y bucles de repetición. Si el sangrado es incorrecto muy probablemente podemos deducir que las llaves o paréntesis no están balanceadas de forma correcta. Al cerrar una llave el cursor nos indica la apertura, y si ésta no se encuentra en la pantalla, el minibuffer nos indica el contenido de la línea donde se abrió la llave.

El modo C también provee comandos para comentar regiones enteras de forma automática y saltar las condiciones de precompilador entre otras. Para compilar podemos utilzar el comando M-x compile que por defecto ejecuta make -k. Este comando esta determinado por la variable compile-command que podemos modificar como hemos sealado en la parte de configuración. Al compilar se abre un buffer *compilation* donde se nos indican los posibles errores que hayan surgido. Si pulsamos ctrl-c ctrl-c encima de una de las lineas de error el cursor se posicionará en la posición donde se detectó el error.

El proceso de corrección de fallos o bugs se lleva a cabo con el comando M-x gdb que ejecuta dicho debugger. Se abre un buffer *gud-nombredelejecutable* donde podemos ejecutar los distintos comandos de debugger como haríamos normalemente. Pero además en el buffer donde editamos el programa C aparece un indicator que seala el punto de ejecución donde nos encontramos. De hecho se pueden ejecutar comandos en este buffer como por ejemplo ctr-x SPC que inserta un breakpoint. La interacción entre los dos buffers facilita de forma importante la tarea de debuggeo.

3.2 Programación en Java

La edición de programas en Java es bastante parecida a la de C, sin embargo existe un proyecto llamado JDEE que proporciona herramientas muy poderosas para el desarrollo de programas Java en Emacs. Por ejemplo tenemos a nuestra disposición un comando que genera documentación lista para JavaDoc de forma automática. Al detectar una palabra reservada de Java intenta completarla y en algunos casos como por ejemplo if, for... nos pone los paréntesis, las llaves y un comentario al final. Como éstas existen diversas ayudas para la edición que naturalmente pordemos desactivar y modificar a nuestro gusto.

Un punto que merece la pena destacar es el menú que muestra los métodos de un objeto. Si tecleamos ctrl-c ctrl-v ctrl-. después de teclear un punto, aparecerá un menú con todos los métodos del objeto. Si además del punto hemos escrito el inicio del método nos restringirá el tamao del menú y en el mejor de los casos nos completará con el método deseado, que indica entre paréntesis los tipos de los argumentos que requiere.

La generación de código se ve mejorada por los wizards y templates. Los wizards son:

Import Wizard

Method Override Wizard
Interface Wizard
Delegate Wizard
Abstract Class Wizard
Get/Set Wizard

Y los templates generan ;
Get/Set Pair, Liesteners, Println u otros definidos por el propio usuario.

La compilación de los .java se lleva a cabo con el comando activado por las teclas ctrl-c ctrl-v ctrl-c. Al igual que en la compilación de C surge un nuevo buffer que nos informa del resultado de la compilación. Si esta ha sido o no correcta y donde se han producido los errores. Pulsando ctrl-c ctrl-c pasaremos al buffer de edición al punto exacto donde se produjo el error.

Para ejecutar una aplicación Java podemos teclear ctrl-c ctrl-v ctrl-r desde el archivo que contenga la clase main. Todas estas acciones están gobernadas por variables que podemos definir de forma que los path, y comandos de ejecuciónsean exactamente los que queremos. La modificación de estas variables de realiza como indicamos en la parte de configuración y el nombre de las mismas se indica en la documentación de JDEE que se puede consultar en su web: http://jdee.sunsite.dk.

3.3 Programación en COMMON-LISP

Hemos dicho antes que Emacs está escrito en LISP, concretamente en una versión llamada emacs-lisp. Si lo que queremos es editar common-lisp Emacs nos ofrece las comodidades habituales, indentación, colores etc... pero además podemos utilizar emacs junto con una aplicación llamada CLISP que es libre.

Basta con aadir unas líneas a nuestro archivo .emacs (del qeu hablaremos más tarde) para tener todas las funcionalidades de CLISP en Emacs. Abrimos dos buffers en uno editaremos nuestro código LISP y en el otro tecleamos M-x run-lisp. Esto cargará el proceso con CLISP y podremos utilizarlo directamente para evaluar expresiones que insertemos o bien, para evaluar expresiones del buffer de edición con código LISP.

Los siguientes comandos son los más utilizados para la evaluzación : ctrl-c ctrl-e evalua el código situado bajo el cursor
ctrl-c ctrl-r evalua una región ctrl-c ctrl-l evalua todo un archivo. Para más información sobre el uso de CLISP consulte el manual de la web: http://clisp.cons.org/.

3.4 Edición en VHDL

Este lenguaje se utiliza para la creación de hardware. Al abrir un archivo VHDL Emacs carga el modo VHDL que nos permite editar de forma muy cómoda por su sangrado, colores, y generación automática de código, entre otras utilidades. El gran pero que he encontrado ha sido la dificultad de encontrar un simulador de VHDL en Linux, pese a que hay varios proyectos abiertos a día de hoy no he probado ninguno de forma satisfactoría de forma que espero poder completar esta sección en un futuro.

3.5 Mucho más...

Estos son solo algunos ejemplos con los que tengo más experiencia. Existen muchos más modos de edición de Emacs que sin duda facilitan la tarea de programadores y usuarios. Este documento ha sido escrito en Emacs, que entre otras cosas facilita la creación de documentos Latex.

Para terminar con las posibilidades que ofrece Emacs como entorno de desarrollo vamos a sealar la speedbar y los ctags. La speedbar (o barra rápida) se ejecuta tecleando M-x speedbar (entre otras formas). Se crea un nuevo 'frame' en el que se incluye toda la información del directorio. No vamos a describir en detalle el speedbar porque depende del modo en el que se encuentre Emacs. Por ejemplo en Java resulta especialmente útil porque para cada subdirectorio (package) del proyecto incluye un menú que podemos desplegar y acceder a sus clases. Además de las clases se incluyen sus métodos de forma que podmeos saltar de un archivo a otro de forma rápida y sencilla.

La limitación del speedbar es seguir una función. Es decir si conocemos la función podemos acceder rapidamente a su código haciendo click en el speedbar pero si no sabemos donde se haya la función tendremos que buscarla entre todos los menús y esto puede ser largo. Para ello existen las Ctags (o también las etags que son propias de Emacs) que nos permiten un rápido acceso a toda la información que necesitemos. No vamos a profundizar mucho en su uso pues puede resultar complejo. A nivel básico basta con generar un archivo TAG ejecutando ctags en un directorio. Cuando tengamos el fichero desde Emacs tenemos a nuestra disposición los siguientes comandos :

`M-x visit-tags-table <RET> FILE <RET>`
Selecciona el fichero TAG
`M-. [TAG] <RET>`
Busca la primera definición de TAG
`M-*`
Regresa a la invocación previa de "M-.".
`C-u M-.`
Busca una nueva definición del último tag.

Ctags está licenciado bajo la GPL y se puede adquirir gratuitamente en la web: http://ctags.sourceforge.net/ donde además se puede consultar más información.

Otra utilidad interesante de Emacs para programadores es su integración para trabajar con CVS. En el menú Tools, bajo el epígrafe PCL-CVS encontramos 'Check-out module' que nos pediá el nombre del proyecto y la carpeta donde deseemos que se copien de forma temporal los archivos. Para actualizar la versión del servidor es suficiente con acceder a través del menú a 'check in/out' o pulsar ctrl-x v v, nos pedirá si queremos realizar algún comentario y para terminar pulsaremos ctrl-c ctrl-c. Conviene navegar un poco por los menús para comprobar las utilidades de CVS que nos ofrece Emacs.

4 Emacs: Editor extensible

En alguna ocasión hemos mencionado el fichero .emacs propio de cada usuario. Este fichero se utiliza para poder personalizar Emacs de forma que se adapte a las necesidades del usuario. En él podemos incluir toda la configuración que deseemos puesto que al cargar, Emacs comprueba si existe un .emacs y carga su contenido. Sirva como ejemplo las configiuraciones que indicamos en el primer apartado. Si quisieramos que estuvieran presentes cada vez que iniciamos Emacs, tendriamos que aadir líneas como estas a nuestro fichero .emacs:

```
(global-set-key [right] 'forward-char)
(global-set-key "\C-z" 'shell)
```

La información de este archivo esta escrita en LISP, sin embargo en muchas ocasiones no es necesario conocer este lenguaje puesto que basta con copiar y pegar ciertas líneas para obtener los resultados deseados. Por ejemplo para que funcione JDEE tenemos que indicar (require 'jde'). Con esto es suficiente para que cargue la información de JDEE. Para que funcione CLISP al teclear M-x run-lisp hay que incluir una línea de la forma: (setq inferior-lisp-program "/usr/bin/clisp").

Esta última línea asigna a la variable inferior-lisp-program el valor del path donde se encuentra CLISP. Si aprendemos un poco de LISP rápidamente podremos modificar de forma increíable Emacs. Incluso podremos crear archivos con programas y utilidades nuevas y aadirlas a Emacs simplemente aadiendo a nuestro .emacs algo del tipo (load-library "pathdelarchivoconcodgolisp.el"). De esta forma podemos ampliar las capacidades de Emacs para que realice todo lo que se nos ocurra. Sirvan de ejemplo las siguientes utilidades de Emacs:
ejecutar una shell
jugar al tetris : M-x tetris
navegar por internet http://www.cs.indiana.edu/elisp/w3/docs.html
leer el correo con Rmail
reproducir mp3
emulaciones ,de Vi por ejemplo ;-)
mensajería instantánea (AIM, MSN)

y todo lo que se nos ocurra y se pueda programar.

5 Conclusión

Emacs fue creado en laboratórios de Inteligencia Artificial, y por ello posee unas propiedades únicas. Puede no resultar muy intuitivo en un principio pero las ventajas que ofrece merecen en muchos casos la pena. En el entorno Universitario Emacs puede ser adaptado fácilmente a las tareas desarrolladas en la docencia, utilizando siempre software libre y de una calidad que dificilmente podemos encontrar en herramientas propietarias. Por ejemplo el caso del CLISP es mucho más potente que el programa alternativo que se ofrece en mi escuela: Allegro.

Con un poco de esfuerzo inicial para empezar a dominar este editor podemos beneficiarnos de no tener que aprender a usar una aplicación distinta para cada tarea que intentemos desarrollar, tener la seguridad que vamos a tener un entorno de desarrollo conocido puesto que Emacs existe en la mayor parte de las platarformas y unas posibilidades de personalización dificilmente igualables.

6 Enlaces: Emacs en la red

* Gnu Emacs Manual:
- http://www.gnu.org/manual/emacs/index.html
Gnu Emacs Manual Top:
- http://astreo.ii.uam.es/%7Ephaya/emacs/Top.html
- http://math-www.uio.no/doc/gnu/emacs/top.html
Tutoriales y otros manuale:
- http://jeremy.zawodny.com/emacs/emacs.html
- http://seamons.com/emacs/
Emacs Lisp:
- manual: http://www.gnu.org/manual/emacs-lisp-intro/emacs-lisp-intro.html
- decenas de programas elisp : http://www.anc.ed.ac.uk/ stephen/emacs/ell.html
Personalización de Emacs:
- varios: http://www.emacswiki.org/cgi-bin/wiki
- navegador: http://www.cs.indiana.edu/elisp/w3/docs.html
- JDE: http://jdee.sunsite.dk/
Xemacs:
- http://www.xemacs.org/

7 Licencia