Preparándote para el viaje

Vim es un editor de texto basado en modos: inserción de texto, modo normal (para moverse por el texto y editarlo), modo comando, modos visuales para seleccionar más fácilmente trozos de texto y otros. Cada operación en Vim tiene su correspondiente atajo o combinación de teclas, evitando el uso del ratón por su lentitud. La mayoría de editores, en cambio, sólo disponen de un modo de inserción donde se supone que el usuario sólo introduce texto y luego recurren a menús, botones o combinaciones de teclas para hacer todo lo demás. En Vim sólo se utiliza el teclado

Vim ofrece una experiencia de edición de texto altamente eficiente, veloz y libre de distracciones. Su principal ventaja es la misma que la de la mecanografía: su dominio te permite editar texto sin desviar tu atención hacia otros componentes de la interfaz del programa. A medida que vas memorizando los comandos la edición se convertirá en un acto reflejo y te permitirá concentrarte en el texto, de la misma forma que el escritor que conoce mecanografía no necesita perder tiempo en buscar las teclas en el teclado y se centra en el texto a escribir. Por otro lado, el principal inconveniente de Vim es la pronunciada curva de aprendizaje: si quieres sacar de él su máximo potencial, tendrás que hacer un esfuerzo inicial que luego verás recompensado con creces

Objetivo

Este curso intenta ayudarte a descubrir Vim de una manera lúdica, sabiendo que supone un cierto reto pero evitando caer en el desaliento que puede suponer aprender algo totalmente nuevo (aunque estemos hablando de un editor que tiene 30 años). La recompensa merece la pena: aprenderás a manejar un editor que está disponible en cualquier sistema operativo y que por su tamaño y mínimos requisitos se puede instalar casi en cualquier dispositivo. A la vez que es capaz de tratar con ficheros de mayor tamaño que cualquiera del resto de editores más sofisticados, utilizando menos memoria y a través de cualquier tipo de interfaz por precaria que esta sea. Todo esto lo hace ideal para editar ficheros de configuración de cualquier tipo, programar en todos los lenguajes existentes, modificar textos de manera automatizada, etc.

Instalar Vim

Puedes acceder a Vim directamente desde línea de comandos (viene instalado en casi todos los sistemas Unix), pero si te encuentras más cómodo entre ventanas también puedes instalar una versión gráfica de vim, en cualquier caso el comportamiento es exactamente igual, la versión gráfica trae un menú y algunos botones, pero te recomiendo deshabilitarlos para que tengas una experiencia 100% "in the vim way", al principio te costará, luego te alegrarás.

También existen versiones gráficas de vim para windows, mac o linux, puedes descargarlas desde la página http://www.vim.org/download.html

Ten Vim siempre abierto durante el curso y pon en práctica todo lo que vayas leyendo, experimenta, investiga, intenta incluso ir por delante de lo que te cuento.

Ten también lápiz y papel preparados antes de empezar el curso, estás en territorio desconocido y es bueno ir trazando un mapa. En la próxima etapa de este viaje te doy más detalles, pero de momento apunta todo lo que vayas aprendiendo.

Sobrevivir al primer impacto

Dice Elon Musk (el fundador de Tesla, SpaceX, etc) que él quiere morir en Marte, pero no del primer impacto. Eso quiero conseguir en esta sesión, que sobrevivas al primer impacto con Vim y no salgas corriendo del curso nada más abrir el editor por primera vez.

Así que ponte en situación, ya has salido de la tierra, del mundo del Bloc de Notas, de Word, del Notepad Plus y de cualquier editor del universo que conoces; estás a punto de aterrizar en Vim. Recuerda que esto es otro planeta, aquí no hay barras de menú, ni botones con iconos de colores, ni siquiera deberías usar el ratón, ni los cursores. Todo parece desierto, como en la luna, todo está bajo la superficie, justo bajo las yemas de tus dedos.

Abre el editor, para ello invoca a vim desde la línea de comandos cargando el fichero de ejemplo `vim primeras_ordenes.txt` o abre el fichero desde una versión gráfica de vim.

Si lo has hecho bien aparecerá el contenido del fichero, si algo ha fallado puede que estés viendo algo parecido a esto, no pasa nada, es la pantalla de créditos de vim.

Utiliza las teclas `h, j, k, l` para moverte a izquierda, abajo, arriba y derecha del texto (un truco para no perderte es que la j parece una flecha hacia abajo)

Ve a la última línea del fichero usando la tecla `j`, cuando llegues a la línea en blanco pulsa `i`. Ahora puedes teclear texto como si estuvieras en un editor normal, escribe lo siguiente: "A un panal de rica miel cien mil moscas acudieron..."

Cuando termines pulsa la tecla Escape (`ESC`) y estarás de vuelta al modo normal (normal para Vim, puede que todavía sea un poco raro para ti). Prueba ahora a borrar algunos caracteres usando la tecla `x` observa que puedes borrar pero no puedes escribir. Si quieres volver a escribir sitúate con ayuda de las teclas `h, j, k, l` en donde quieras y entonces pulsa `i` . Observa que tras pulsarla puedes empezar a escribir delante de donde tenías el cursor (la i viene de insertar).

Cuando te canses de practicar pulsa `ESC`, y entonces teclea: `:q`

Este es el comando para salir de Vim sin guardar, si has hecho algún cambio se quejará de que no has guardado todo el trabajo que has hecho, como ahora no queremos guardar escribe entonces: `:q!`

Para recordar el signo de admiración recuerda que puedes sustituirlo mentalmente por cualquier imprecación que necesites soltar. Yo suelo pensar: "¡Salte ya, repámpanos!"

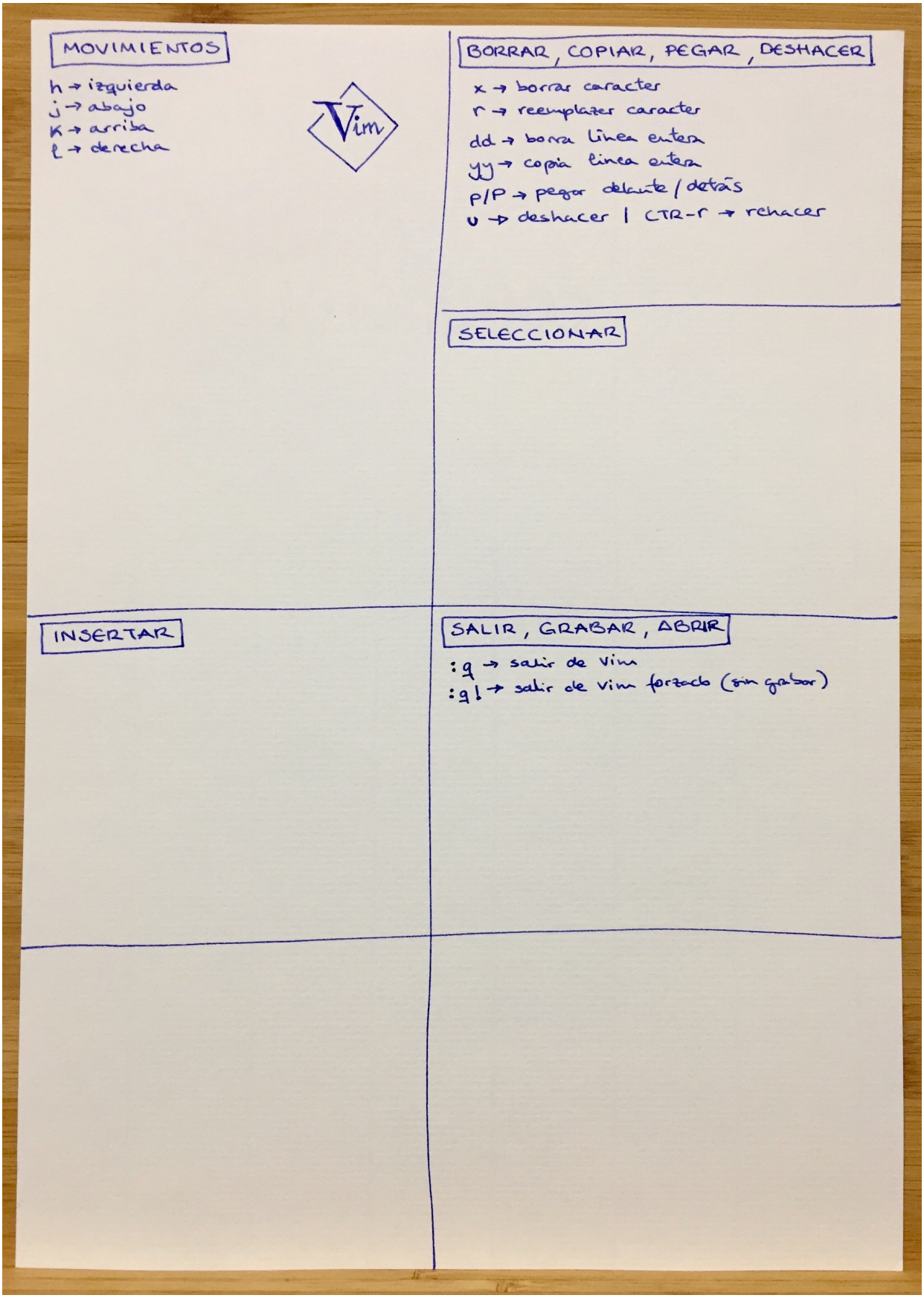
Si quieres guardar el fruto de tu esfuerzo teclea `:wq` para guardar y salir.

Estas tres últimas ordenes que has utilizado son un tipo especial de ordenes a las que llamamos **modo comando**, se distinguen por los dos puntos y porque aparecen siempre en la línea inferior de la ventana.

Tu chuleta de Vim

A lo largo del curso te iré dando algunas chuletas que te podrán ser de utilidad, y que quedarán muy chulas en tu escritorio, pero es muy importante confeccionar una propia.

Coge papel y boli y en un rinconcito apunta estos primeros comandos que has aprendido y para que sirven cada uno de ellos, puedes ver en la ilustración una idea de como podría ser esta chuleta, pero no tiene porqué ser exactamente así. Al final del curso tendrás la hoja entera rellena



Resumen de comandos de esta etapa

Todavía no hemos hablado de los modos, pero los menciono aquí para que te vayas haciendo el oído, hablaremos de ellos en nuestra próxima etapa.

Modo normal

`h` → mover cursor un carácter hacia la izquierda

`j` → mover cursor una línea hacia abajo

`k` → mover cursor una línea hacia arriba

`l` → mover cursor un carácter hacia la derecha

`i` → comenzar a escribir antes del cursor, entra en modo inserción

`x` → borrar el carácter bajo el cursor, no entra en modo inserción

`ESC` → volver a modo normal

Modo comando

`:q` → salir de vim

`:q!` → salir de vim forzado

`:wq` → salir de vim guardando antes

Comenzar a respirar

Si has llegado hasta aquí lo has conseguido, has sobrevivido al primer impacto, pero no te confíes, sigues en un ambiente hostil. Tienes que aprender a respirar de nuevo, a moverte de nuevo, casi que a pensar de nuevo, ya te dije que Vim era diferente.

Los modos de Vim

Una de las características más distintivas de Vim es que es un editor basado en *modos*, dependiendo del modo en que te encuentres Vim se comportará de forma diferente.

El modo normal

Cuando accedes a Vim, entras en Modo Normal, se llama así porque es el modo en que deberíamos esta casi todo el tiempo. En el planeta de donde vienes se llama editores de texto a unos programas que sirven para teclear texto, borrar un poco, volver a teclear, etc. Deberían llamarse "escritores de texto", Vim por el contrario es un auténtico "editor de texto", sabemos que es así porque en modo Normal, sólo se puede editar, para escribir hay que pasar a otro modo que se llama Modo Insertar.

Esta diferencia es la que hace que al principio te explote la cabeza cuando te enfrentas a Vim, todo el mundo te dice que es un editor de texto, tu vienes de un mundo en que un editor de texto es otra cosa, y cuando intentas escribir en Vim no se comporta como esperabas. Pero bueno, has venido aquí por los superpoderes, y los vas a tener, sigamos.

El modo Insertar

En Modo Insertar Vim se parece bastante a lo que tu conoces, tú tecleas y él escribe las letras que has pulsado. Debes acostumbrarte a pasar poco tiempo en este modo. La rutina normal sería:

1. Buscas en que parte del texto quieres trabajar
2. Te mueves hasta esa parte usando los comandos de desplazamiento de Vim
3. Tecleas un poco, pulsas ESC y buscas de nuevo donde tienes que seguir trabajando. O pulsas ESC, te mueves a otro sitio y repites la misma acción que acabas de hacer pulsando sólo una o dos teclas.

El modo Visual

Un tercer modo muy potente es el Modo Visual, en este modo seleccionas un punto de partida y un punto final en el texto para trabajar con él. Puedes usar los comandos de desplazamiento y los objetos de texto para ampliar o acortar tu selección. Una variante del Modo Visual es el Modo Bloque Visual, parecido al anterior pero trabaja con columnas en vez de con líneas, este modo te permitirá hacer maravillas con datos tabulados.

Cambiando de modo

Incluso cuando estés escribiendo de corrida un texto más largo, deberías acostumbrarte a pulsar ESC en las pausas naturales del texto. Esto te permitirá hacer y deshacer más fácil, repetir discrecionalmente las acciones que hayas hecho y te da un pequeño respiro antes de seguir tecleando para que veas como va quedando tu obra maestra.

Al principio quizás te hagas un lío porque no recuerdas en que modo te encuentras cuando estás en Vim. Parece un fastidio eso de tener que teclear "i" o "a" cada vez que se quiere escribir algo. Y si empiezas a escribir estando en modo Normal de pronto empiezas a ejecutar comando que lo lían todo. Por ejemplo abre un documento de texto, uno bien llenito, y escribe, como sin querer: `dG` de pronto has borrado todo el texto.

"¡Aggg! ¿cómo es posible que alguien use este editor?", no pasa nada, pulsas la `u` y todo vuelve a estar igual. Como acabo de contarte Vim está pensado para editar, pero no para editar y punto, para editar a la velocidad del rayo, de manera inteligente, evitando muchísimos errores y ahorrando muchísimas pulsaciones de teclado, o viajes del teclado al ratón y vuelta. En el ejemplo anterior la `d` le dice a Vim que vas a borrar algo y la `G` nos traslada al final del documento, con lo que se borra todo el documento. La `u` deshace la última acción que hayamos realizado, así que quizás sea lo primero que debas aprender: u de "undo", deshacer. Y ya de paso, si te pasas al deshacer pulsa `CTRL-r` para rehacer.

En la etapa anterior aprendiste a usar en modo inserción utilizando la `i` pero hay muchas maneras de entrar al modo inserción, cada una aporta algo distinto, claro:

`a` → Insertar después del cursor

`o` → Insertar una línea bajo la actual

`O` → Insertar una línea sobre la actual

`I` → Insertar al principio de la línea actual

`A` → Insertar al final de la línea actual

Como ves muchos comandos de Vim hacen cosas parecidas cuando están en minúsculas o mayúsculas: `a` / `A` `i` / `I` `o` / `O` etc...

Veamos algo más potente, estando en modo normal, colócate al principio de alguna palabra que te caiga mal y pulsa `cw`. Con esto has borrado una palabra y entras en modo inserción. Esto parece poca cosa pero ya verás las posibilidades que encierra.

Sigamos con algunos movimientos:

Movimientos básicos

`0` → (cero) ve a la primera columna `^` → ve al primer carácter 'visible' de la línea actual `$` → ve al final de la línea `g_` → ve al último carácter visible de la línea (nunca lo he usado) `CTRL-B` → una pantalla hacia arriba `CTRL-F` → una pantalla hacia abajo `/patito` → busca "patito"

Copiar y pegar

`P` → pegar delante, recuerda que `p` es pegar a continuación. `yy` → copia la línea actual, es un atajo de `ddP` (que a su vez es un atajo de `d$P`)

Deshacer y rehacer

`u` → deshacer `<C-r>` → rehacer

Frente a otros editores de texto que deshacen los cambios carácter a carácter la `u` de Vim 'recuerda' las acciones de Vim de manera combinada por lo que el deshacer es mucho más natural y fiable, esto se eleva a la máxima potencia cuando te acostumbres a usar Vim 'a la manera de Vim'

Abrir/Grabar/Salir/Cambiar de fichero (buffer)

`:e <ruta/archivo>` → abre archivo `:w` → graba el archivo actual `:saveas <nueva_ruta/nuevo_archivo>` → graba el archivo actual con otro nombre y/o ruta y ábrelo para editar `:x`, `ZZ` or `:wq` → save and quit (`:x` only save if necessary) `:q!` → quit without saving, also: `:qa!` to quit even if there are modified hidden buffers. `:bn` (resp. `:bp`) → show next (resp. previous) file (buffer)

Tómate un respiro

Comprendo que todo esto que estamos viendo te resulte un poco apabullante. Son demasiados modos, demasiados comandos, verbos, objetos, etc. Es normal, todos los que flipamos con Vim hemos pasado por eso. No te preocupes, a medida que vayas usando Vim un poco más y que vayas pillando su filosofía empezarás a interiorizar todo esto. Y entonces trabajaras de Vim de forma natural, y procesarás archivos a una velocidad endiablada.

Mientras tanto respira, no te agobies, toma tu tiempo y dedica un rato a manejar y mejorar en Vim cada vez que tengas oportunidad. Si tienes que editar algún fichero, piensa, ¿podría hacerlo con Vim?

Vuela como si no te afectara la gravedad

En la anterior etapa has superado el pánico inicial, incluso puede que te sientes cómodo. Es hora de orientarse y tomar el mando de esta misión.

> Si has cerrado y abierto de nuevo Vim puedes saber en que zona del planeta has caído utilizando el comando :pwd (igual que si estuvieras en Linux, pero con los dos puntos delante), con esto sabrás en que directorio del disco te encuentras. Mira ahora la barra inferior, justo encima de la línea de comandos, ahí tienes más información....

En un editor terrestre tienes que ir paso a paso, para moverte, para editar, para todo. Aquí puedes aprovechar la menor gravedad y empezar a moverte dando saltos más grandes. En este sentido Vim te ofrece algo único: un lenguaje de edición. Este lenguaje tiene sus verbos, sus objetos e incluso sus adverbios (modificadores).

Prueba ahora esto: estando en modo normal, colócate al principio de alguna palabra que te caiga mal y pulsa `cw`. Con esto has borrado una palabra y entras en modo inserción. Puedes reemplazar ahora la palabra por otra que te guste más. Pulsa `ESC` para volver a modo normal cuando termines. Esto parece poca cosa pero ya verás las posibilidades que encierra.

Verbos sin objeto

Estos verbos hacen lo que hacen y no se pueden combinar con objetos, se comportan más bien como comandos.

`x` → borra el carácter después del cursor `D` → borra desde el cursor hasta el final de la línea `C` → borra desde el cursor hasta el final de la línea y entra en modo inserción `r` → reemplaza un carácter `J` → une dos líneas

Verbos con objetos

`d` → borra "algo" `c` → borra "algo" y entra en modo inserción (change)

`y` → copia "algo" al portapapeles

Modificadores de objetos

`-a-` → Uno, una → Comprende el objeto completo (incluyendo delimitadores) `-i-` → En (inside, dentro) → Comprende el objeto (sin incluir delimitadores)

Objetos de texto

`w` → palabra (word) Trata las palabras-separadas-por-guiones (o por puntos o por barras) como palabras distintas. `W` → gran palabra (big Word) Trata las palabras-separadas-por-guiones como una sola palabra. `ap` → un párrafo (si, si, la "p" que usamos para "pegar" aquí significa párrafo) `as` → una frase (misma cosa que con la "p") `a"` → texto entre comillas (incluye las comillas) `a)` → texto entre paréntesis (incluye los paréntesis) `ip` → dentro de un párrafo (no incluye el espacio en blanco) `i)` → texto entre paréntesis (sin los paréntesis)

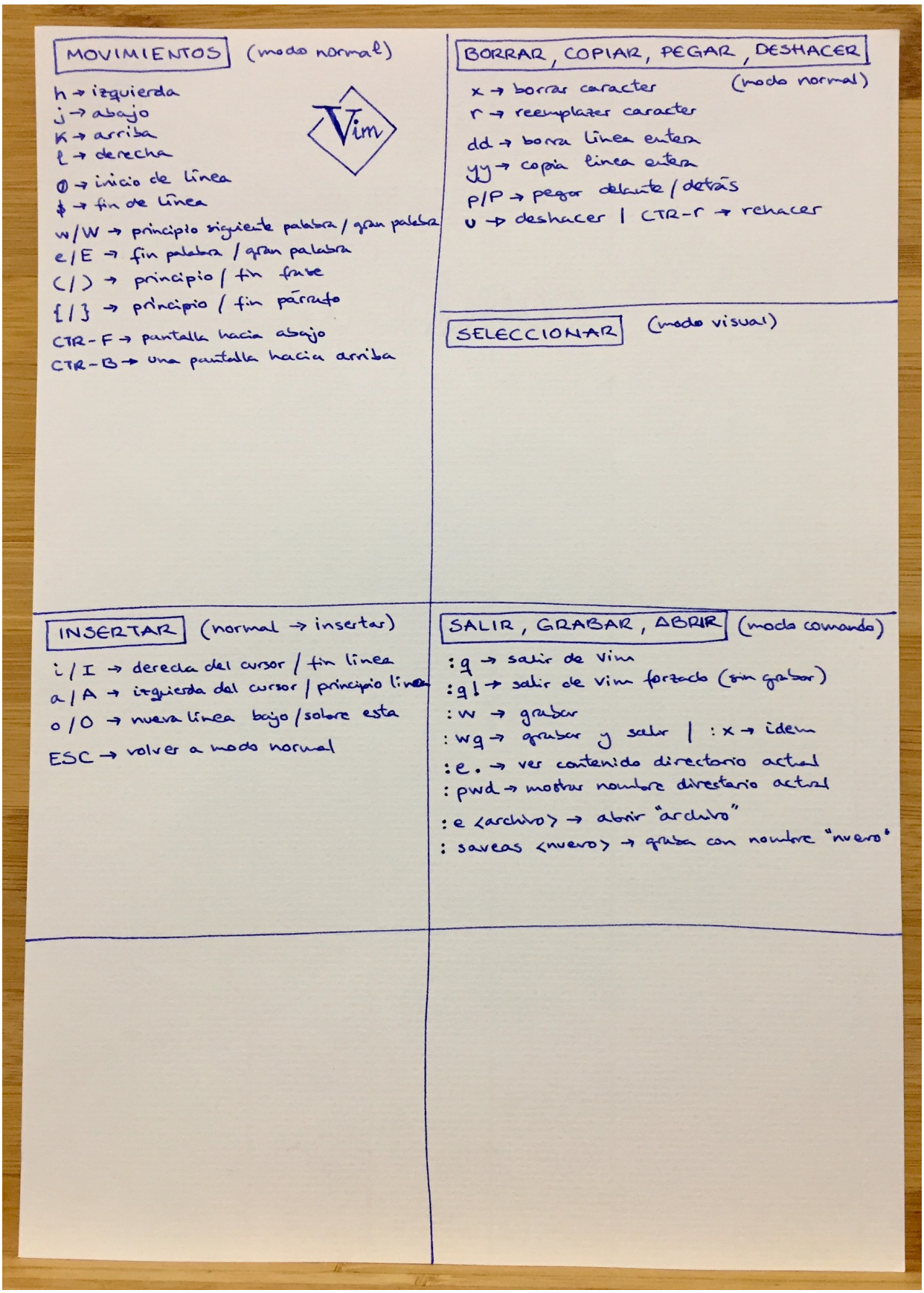
Veamos, y practiquemos, algunos ejemplos

`daw` → borra una palabra `yi}` → copia el texto entre corchetes (muy útil para copiar todo el código de una función) `v/pepe` → selecciona desde la aparición del cursor hasta la aparición de la cadena "pepe".

Ahora te propongo que hagas el ejercicio que aparece en el fichero "copiando_funciones.php". Inténtalo primero por tu cuenta, luego sal sin grabar e inténtalo de nuevo con las instrucciones que aparecen abajo del todo del fichero.

> ### No te confundas
>
> El uso de `t` en algunos comandos parece un modificador de objeto pero no lo es. En realidad es una orden de movimiento que le dice a Vim: "ejecuta el verbo que he puesto antes hasta que encuentres el carácter que voy a poner ahora"
>
> `vtw` → No selecciona una palabra sino que selecciona desde la posición actual del cursor hasta el siguiente carácter "w" en la línea actual

Bien, utiliza estos comandos hasta sentirte cómodo con ellos, una vez que lo hayas hecho te sentirás prácticamente de nuevo como en casa, tomate tu tiempo. Podrás hacer con Vim lo que hasta ahora hacías con un editor normal, recuerda, tienes que adaptarte a tus nuevos superpoderes y el primer paso es poder hacer de nuevo tus acciones cotidianas. Puede que todavía te sientas un poco incomodo, pero sigue conmigo a la siguiente etapa de este alunizaje.



Muévete a la velocidad de la luz

Vim te permite saltar a cualquier línea de un fichero de manera instantanea. Abre el fichero tal y cual. Pulsa 5G para ir a la línea 5, 10G para ir a la línea 10, etcétera. Si te resulta más cómodo también puedes usar 5gg, o 10gg, el resultado es el mismo. Recuerda que con gg a secas vas al principio del fichero, mientras que con G a secas vas al final.

Para ir rápidamente de un sitio a otro de un fichero es muy útil tener activada la numeración de líneas, esto se consigue con `:set number` desde la línea de comandos, pero es más cómodo ponerlo en tu fichero ".vimrc" para no tener que teclearlo cada vez.

En algunos casos puede ser incluso más interesante utilizar la numeración relativa, esto se consigue con `:set relativenumber`

La línea de comandos

Cuando pulsas `:` para ejecutar un comando o `/` para realizar una búsqueda el cursor se va a la línea de comandos, que está en la parte inferior de la pantalla.

El modo comando permite algunos atajos de teclados sencillos para moverse por él, aquí si puedes usar los cursores (si los tienes disponibles, claro). Veamos algunos de estos atajos para ganar rapidez.

Para moverte por caracteres usa los cursores izquierda y derecha, si pulsas las teclas `SHIFT` o `CTRL` junto con estos cursores te moverás por palabras, para ir al principio de la línea pulsa `CTRL+b`, para irte al final pulsa `CTRL+e`.

Los cursores arriba y abajo te permiten recorrer el historial de comandos (o de búsquedas) que has utilizado anteriormente. Si tecleas el comando `:hist` o `:history` podrás ver todo el listado del historial de comandos. Para ver el historial de búsquedas usa `:hist se` o `:history search`

Si no te has cansado de trucos:

- `CTRL+u` borra la línea completa
- `CTRL+w` borra una palabra
- `ESC` se sale del modo comando o búsqueda sin hacer nada

Vista de rayos X para búsquedas y sustituciones

Búsqueda rápida

Una forma rápida de buscar la siguiente aparición de la palabra en la que tienes el cursor es pulsar `*` luego con `n` o pulsando de nuevo `*` puedes buscar la siguiente. Si quieres la aparición anterior pulsa `#`. Estos dos comandos buscan palabras exactas, no fragmentos de cadena.

Búsqueda con / y ?

Ya sabes que para buscar cualquier cadena en un fichero sólo tienes que pulsar `/` en modo normal, después tecleas la cadena que quieres buscar y pulsas `<ENTER>` para que el cursor salte a la siguiente aparición de esa cadena.

Si ahora pulsas la tecla `n` irás a la siguiente aparición de la cadena que buscas, cuando llegues al final del fichero te dará un aviso y volverá a buscar desde el principio. Si quieres ir a la aparición anterior teclea `N`

En realidad el modo búsqueda funciona con expresiones regulares como puede que aún no las conozcas sólo te daré algunos trucos aquí.

- Algunos caracteres no se pueden usar directamente `^`, `.`, `$` o `\` porque tienen un significado especial.
- Si quieres buscar una palabra exacta (p.e "pero") utiliza `/\<pero\>` . Esto encuentra "pero", pero no "ropero" o "perorata". A mi esta construcción siempre se me olvida, así que uso un truco para recordarla, me voy a cualquier palabra del texto en modo normal (¿en que modo estabas si no?) y pulso `*` entonces Vim, en la línea inferior me pinta la búsqueda exacta que acaba de hacer.
- Siguiendo con el ejemplo anterior `/pero\>` te devuelve "pero" y "ropero". Mientras que `/\<pero` te devuelve "pero" y "perorata".

Sustituciones

El modo visual

Algo muy interesante del modo visual es que en el siguen funcionando todos los comandos de movimiento, así que puedes entrar en modo visual y teclear t. para seleccionar hasta el próximo punto. Por supuesto también sirven $, 0, w, }, etc para seleccionar hasta el final de la línea, hasta el principio, una palabra, un párrafo, etc.

Los registros

En Vim el concepto de portapapeles de otras aplicaciones se maneja con algo mucho más versátil: los registros.

Cada vez que copias algo en vim lo puedes hacer en un total de 24 registros distintos, las 24 letras del alfabeto. Luego puedes "pegar" o hacer otras operaciones utilizando cualquiera de estos 24 registros.

Para copiar la línea actual al registro **a** teclea `"ayy` , a continuación puedes pegar con `"ap`

Además de las 24 letras a tu disposición hay una serie de registros por defecto:

- `"*` → portapapeles del sistema
- `""` → registro sin nombre, es al que se copia aquello que borres
- `"0` → registro cero, es al que se copia aquello que copies

Las macros aumentan tus poderes

En la mayoría de los editores de texto es fácil editar y muchísimo más complicado hacer macros en Vim por fin es al revés. Y por qué es esto así pues porque cuenta de que es un lenguaje de visión y si conoces un lenguaje escribir un pequeño cuento es sencillo.

Prácticamente no necesitas aprender nada nuevo para hacer macros en Vim solo tienes que aplicar lo que ya sabes y un par de trucos que te voy a contar.

Las macros en Vim se declaran escribiendo `q` y otra letra cualquiera del alfabeto en minúscula, esto te da 24 macros que podrías mantener en memoria, incluso, si no las sobreescribes, estarán ahí la próxima vez que arranques vim.

Ten en cuenta que las macros se graban en los mismos registros que el "portapapeles" de vim, por lo que si declaras una macro en el registro **a** y luego copias algo en ese registro te cargas la macro. Es cuestión de organizarte, tienes 24 registros.

Un ejemplo de macro

Vas a hacer una macro para subrayar una línea de un título con guiones, pero el numero de guiones tiene que ser igual que el número de caracteres de la línea para que quede con la misma longitud. Con un editor de texto normal tendrías que teclear tantos guiones como caracteres y luego dar un masaje a la yema de tu dedo meñique, con vim lo haces con 6 pulsaciones: `yypVr-`

Como ya te estás vim-acostumbrando me dirás que si tienes que subrayar diez títulos que diez por seis son sesenta y que seguro que vim puede hacerlo mejor ¡faltaría más!

Entonces haz lo siguiente, sitúate sobre la línea que vas a subrayar, da igual la posición.

1. Teclea `qs` para empezar a grabar la macro en el registro **s** (de subrayado)
2. Teclea la macro: `yypVr-`
3. Pulsa `q`
4. Tendrás tu título con un subrayado exacto.
5. Ve al siguiente título que tengas que subrayar y teclea `@s` ¡chan chan!

Editar una macro

Supón que ya tienes una estupenda y laboriosa macro guardada en el registro **a** y necesitas hacer un pequeño, o gran, cambio a la macro para dejarla perfecta. Sigue estos pasos y tendrás un dominio absoluto sobre tus macros:

1. Escribe `:let @a='` (desde el modo comando evidentemente).
2. Pulsar `Ctrl-R Ctrl-R a` para insertar el contenido actual del registro **a**.
3. Edita el texto como necesites.
4. Añade otra comilla simple (`'`) para cerrar y pulsa `ENTER`
5. Escribe `:reg a` para ver el nuevo valor del registro.
6. Teclea `@a` para ejecutar la macro.

Almacenar la macro en tu fichero .vimrc

Las macros se quedan almacenadas en sus registros aunque reinicies vim o el ordenador, pero si estás haciendo un trabajo más largo igual te interesa almacenarla en tu fichero **.vimrc**.

Hacer esto es muy fácil, simplemente sigue los pasos que acabamos de ver en **Editar una macro** pero desde dentro de tu fichero **.vimrc** y sin poner los dos puntos delante de **let**.

Salva tu fichero **.vimrc** y a partir de ahora cada vez que inicies vim esa macro quedará asignada a ese registro.

Buffers, ventanas y pestañas

Cada vez que abras un fichero en vim este se abre en un buffer, una especie de espacio de memoria en el que se carga el contenido del fichero. Los buffers se usan también para mostrar la ayuda (otro fichero al fin y al cabo) o para mostrar la estructura de directorios cuando usas `e.` o `:Ex`. Estos dos últimos casos tienen el atributo de no modificables, aunque puedes moverte por ellos, buscar, seleccionar y copiar texto como en cualquier otro buffer.

Para poder visualizar un buffer necesitas una ventana, puedo tener una sola ventana pero varios buffers abiertos. Puedes reutilizar la misma ventana para ver el contenido de todos los buffers. Si abro una segunda ventana, la pantalla de vim se parte en dos mitades para mostrar cada una de ellas, ya sea mediante una divisoria horizontal `:split` o vertical `:vsplit`. Un mismo buffer puede estar en dos o más ventanas. Puedes seguir partiendo la pantalla en más ventanas hasta que prácticamente sólo te quepa una letra en cada una.

A su vez las ventanas se pueden cargar en distintas pestañas, las pestañas te permiten tener distintas configuraciones de ventanas.

En cualquier caso la lista de buffers abiertos es global y puedes acceder a ellos desde cualquier ventana o pestaña

Comandos para Buffers

`:ls` : lista todos los buffers que tienes abiertos

`:b1` te lleva al buffer marcado como **1** en la lista

`:b filename` : Nos lleva al buffer llamado filename. Este comando permite autocompletar pulsando .

`CTRL-6` : Nos lleva al buffer del que venimos (atajo de :b#).

`:bn` / `:bp` : Va al siguiente / anterior buffer de la lista de buffers.

`:bd` Elimina el buffer de la lista de buffers y cierra la ventana donde estaba si no era la última

`:wall` guarda los cambios en todos los buffers.

`:sba` abre todos los buffers que tengamos en la lista de buffers dividiendo la pantalla para que quepan todos.

`:Ex` abre el explorador de ficheros en la pantalla actual.

`:set hidden` para trabajar cómodamente con múltiples buffers es imprescindible activar esta opción. Si no, no nos permite abandonar un buffer sin guardarlo previamente.

Comandos para Ventanas

`:split` nueva ventana horizontal

`:vsplit` nueva ventana vertical

`:sp fichero` abre el fichero en una nueva ventana horizontal

`CTRL-W CTRL-W`: siguiente ventana

`CTRL-W j, l, k, h` ir a la ventana en la dirección indicada

`CTRL-W =` : pone todas las ventanas al mismo tamaño

`CTRL-W c` : cerrar ventana (nunca cierra la última). Para acordarme pienso: "¡al WC!".

`close` cerrar ventana (nunca cierra la última)

`q` cerrar ventana, si es la última sales de vim

`CTRL-W o`: cerrar todas las ventanas excepto la actual

Comandos para Pestañas o Tabs

`:tabnew` → abrir una nueva pestaña en blanco

`:tabedit <fichero>` → abrir pestaña con un fichero cargado

`:tabnext` o `:tabn` → moverte a la siguiente pestaña

Ejercicios con buffers, ventanas y pestañas

Parte la pantalla de vim en cuatro cuadros en cruz usando los comandos de ventanas

Carga en cada una de las ventanas los ficheros buffer1, buffer2, buffer3 y buffer4

Abre una nueva pestaña y carga los cuatro buffers usando sólo dos comandos

Personalizando Vim: el fichero vimrc

Para configurar Vim, como no podía ser menos, se utiliza un fichero de texto perfectamente legible y editable, el fichero .vimrc.

Este fichero normalmente lo puedes encontrar en:

1. Configuración general de vim para todo el sistema
2. Tu directorio raiz de usuario como fichero oculto: .vimrc
3. Tu directorio raiz, dentro del directorio oculto .vim

Aquí tienes un fichero de configuración bastante completo

```
"
" Alunizando con Vim
" Un buen comienzo para tu fichero .vimrc
"

" Quieres Vim, no vi. Cuando Vim encuentra un fichero .vimrc se pone por defecto en modo 'nocompatible'. En cualquier caso así queda claro cuales son tus intenciones ;-)
set nocompatible

filetype plugin indent on  " Habilita la carga del plugin adecuado para el resaltado de sintáxis de cualquier tipo de fichero y su indentación correcta
syntax on                  " Habilita resaltado y coloreado de sintáxis (en función del tipo de fichero: php, conf, js, etc)

set autoindent             " Indenta la siguiente línea automáticamente en función de la anterior
set expandtab              " Utiliza espacios en lugar de tabuladores
set softtabstop =4         " Cada tabulador equivale a 4 espacios
set shiftwidth  =4         " Los reindentados automáticos serán de 4 espacios
set shiftround             " Los reindentados automáticos serán múltiplos de lo que digas en shiftwidth

set hidden                 " Permite cambiar de buffer sin tener que guardar antes
set laststatus  =2         " Muestra la línea de estado
set showmode               " Muestra el modo actual en la línea de estado
set showcmd                " Muestra las teclas que vamos pulsando en modo comando
set statusline=%<%f%h%m%r%=%b\ 0x%B\ \ %l,%c%V\ %P

set incsearch              " Va buscando a medida que vayamos tecleando en modo / o ?
set hlsearch               " Resalta las búsquedas
set ignorecase             " Insensible a minusculas/mayusculas en las busquedas

set ttyfast                " Acelera el refresco de pantalla
set lazyredraw             " Sólo refresca la pantalla cuando es necesario

set splitbelow             " Abre nuevas ventanas debajo de la actual
set splitright             " Abre nuevas ventanas a la derecha de la actual

set number                  "Numerado de líneas
set relativenumber          "Numeración relativa
```

```
set cursorline              " subraya la línea actual para que sepamos por
donde vamos
colorscheme blue "default blue darkblue slate default delek desert elflord
evening koehler morning murphy pablo peachpuff ron shine slate torte
zellner

"Pasando de backups
set nobackup
set nowb
set noswapfile

" Opciones para un vim más ortodoxo
set guioptions-=m "quita la barra de menu en modo gráfico
set guioptions-=T "quita la barra de herramientas en modo gráfico
nnoremap <up> <nop>
nnoremap <down> <nop>
nnoremap <left> <nop>
nnoremap <right> <nop>
inoremap <up> <nop>
inoremap <down> <nop>
inoremap <left> <nop>
inoremap <right> <nop>

" Truco de regalo: permite salvar ficheros como sudo cuando se te olvide
hacer sudo vim
cmap w!! w !sudo tee > /dev/null %

" Abreviaturas
iabbrev jarr Juan Antonio Ruiz Rivas
iabbrev usev Universidad de Sevilla
```

Hasta el infinito y más allá con los plugins de vim

Pendiente desarrollo. ¡Pronto en sus pantallas!

Temas varios

Mis trucos favoritos

Dedos cruzados

Es habitual que al teclear rápido cambiemos una letra por otra. Por ejemplo: "Widnows". En vim esto se soluciona fácil usando la combinación "xp", ve a la primera letra mal colocada, la "d" y teclea "xp" con esto tendrás "Windows" sin haber salido del modo normal. El comando "x" borra la "d", el cursor avanza hasta la "n", el comando "p" pega de nuevo la "x", pero ahora detrás de la "n" ("P" pegaría delante)

Subrayar un título

Copias la línea de título con: yyp Sustituyes con guiones con: v$r-

Desde el navegador de ficheros

Pulsando “c” - cambia el directorio de trabajo al directorio en que nos encontremos en ese momento. Mira también los comandos que tienes disponibles en pantalla para crear directorios, renombrar ficheros, etc.