

10 - Editar text: vim

Curs 2020 - 2021

ASIX M01-ISO UF1-A01-02 ordres de fitxers

Editar text: vim

Descripció

En el món GNU/Linux és molt normal utilitzar editors de text en mode text (valgui la redundància), sense mode gràfic. N'hi ha molts i cada professional utilitza el que li agrada més (vim, emacs, nano, pico, etc). Sovint cal editar el contingut de fitxers de text de configuració des d'una consola i un editor omnipresent és el **vim**.

En el vim destaquem que hi ha varis **modes**:

- ☐ mode normal
- ☐ mode inserció
- ☐ mode comanda
- ☐ mode visual

Edició bàsica

Editar un text simple

El propòsit bàsic d'utilitzar el vim és editar un text simple per exemple amb l'ordre: vim document.txt. Un cop s'obre la pantalla del vim s'està en el mode normal (no s'està inserint). Es pot observar això mirant la a baix de tot a l'esquerra.

Per entrar en mode **inserir** cal prémer el **caràcter i**. Observar que ara a la part inferior esquerra indica que s'està en mode inserció. En el mode inserció escriure el text desitjat.

Per finalitzar l'edició, desar el contingut i sortir cal fer una seqüència de passes:

- primerament prémer el **caràcter esc (escape)** per sortir del mode inserció (podem observar a la part inferior esquerra que ja no ho diu).

- Seguidament prémer el **caràcter dos punts** : per entrar en mode comanda. Observem que a la part inferior esquerra apareixen els dos punts.
- Indicar la comanda o accions a realitzar. Per desar el document, fer un **write**, cal prémer el **caràcter w**.
- Per sortir del programa vim prémer l'opció de **quit** o sortir, amb el **caràcter q** (el típic caràcter de GNU/Linux que indica sortir).

Exemple en el moment d'obrir el document nou:

```

root@a36:~
File Edit View Search Terminal Help
"document.txt" [New File] 0,0-1 All

```

Exemple de mode inserció:

```

En aquest moment
el document està en mode
inserció
-- INSERT -- 3,10-9 All

```

Exemple en mode comanda :wq just abans de prémer enter:

```

En aquest moment
el document està en mode
inserció
Observar que ara està en mode comanda
per desar i sortir
:wq

```

Editar i desar

El més sensat en escriure i editar un document és anar-lo desant a mida que e streballa, mai esperem a tenir-lo enllestit per desar. cal anar desant periòdicament, per exemple a cada paràgraf.

El procediment consisteix en anar alternant el mode edició amb el mode comanda write:

- deixar d'editar el document prement el **caràcter *esc***. Observem que a la part inferior ja no indica que s'està inserint text, s'ha passat a mode normal.
- prémer el **caràcter *:*** per entrar en **mode comanda** i escriure la comanda **write** prement la **tecla *w*** i enter. Això desa el document.
- Ara cal tornar al mode **inserció** per continuar treballant, per fer-ho cal prémer el **caràcter *i***.

Descartar els canvis

A vegades es fan canvis a un document i no es volen desar, es volen descartar. Per exemple s'ha editat un document i s'ha acabat espatllant-lo i en lloc de desar volem abortar, cancel·lar l'edició i sortir. Per fer-ho cal indicar que es vol sortir del vim sense desar.

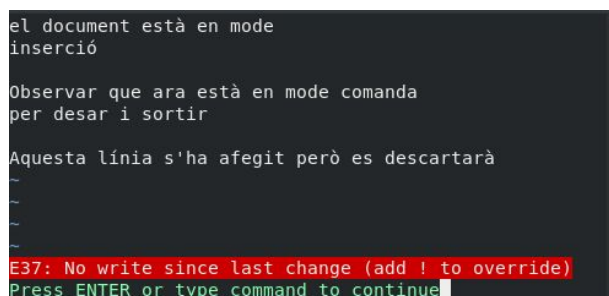
Si en un editor de text com per exemple el LibreOffice s'edita i es modifica un document i s'intenta tancar el programa no deixa, primer adverteix que no s'han desat les modificacions i demana confirmació de si cal descartar o no les modificacions (sovint és que ens havíem oblidat de desar!).

El vim té un comportament similar, si s'han fet canvis en el document i s'intenta sortir sense desar-los, prement ***esc : q*** (escape seguit de dos punts i la comanda quit) el programa mostra un missatge d'error informant que no es pot sortir perquè no s'han desat els canvis.

Per forçar a abandonar l'edició d'un document cal:

- Anar a mode comanda prement la tecla ***escape*** i els ***dos punts : i*** fer la comanda ***q!***. Observar que hi ha el símbol d'exclamació al final, que és una manera de manar, forçar que es vol plegar tant si com no.

Exemple on podem veure el missatge d'error informant que es vol abandonar el programa sense haver desat els canvis:



```
el document està en mode
inserció

Observar que ara està en mode comanda
per desar i sortir

Aquesta línia s'ha afegit però es descartarà
~
~
~
E37: No write since last change (add ! to override)
Press ENTER or type command to continue
```

Exemple en el moment en que es fa la comanda d'abandonar l'edició:

```
En aquest moment  
el document està en mode  
inserció
```

Observar que ara està en mode comanda
per desar i sortir

Aquesta línia s'ha afegit però es descartarà

```
~  
~  
~  
:  
:q!
```

Comandes d'edició

Sovint els principiants en el vim utilitzen les tecles del cursos (les fletxes) per anar amunt, avall a l'esquerra i a la dreta per tot el document i esborren el text caràcter a caràcter. Això a part de ser poc eficient posa molt nerviós al professor...

Quan s'està en el mode edició podem desplaçar-nos amb:

- *inici* La tecla inici porta al porta al inici de línia
- *fi* La tecla fi porta al final de línia
- *pgup* desplaça una pàgina cap a munt la visualització.
- *pgdown* desplaça una pàgina cap a avall la visualització.
- *g* La tecla g porta al principi de tot del document (o control + inici).
- *G* La tecla G porta al final de tot del document (o control + fi).
- *del* La tecla sup esborra el caràcter en la posició del cursor.
- *back* La tecla <- esborra el caràcter anterior de la posició del cursor.

En mode normal hi ha moltíssimes combinacions de tecles amb significats especials, algunes d'elles són les següents:

- **u** undo elimina la última acció feta
- **dd** esborra una línia, la línia actual
- **ndd** escriure un número n i prémer dd esborra les n línies següents. per exemple prémer 2dd esborra dues línies.
- **.** el punt repeteix la última comanda. Es pot anar prement el caràcter punt i cada cop repeteix la mateixa acció. Si per exemple s'ha fet 3dd i després es prem el punt, s'esborren tres línies més, si es torna a prémer el punt doncs tres línies més...
- **n<acció>** en general prémer un número abans d'una acció fa que l'acció es repeteixi n vegades.
- **nG** prémer un número n i la lletra G posa el cursor a la línia n. És a dir, anar de dret a la línia.
- ***** Trobar la següent paraula en el text igual que el word que hi ha actualment en el cursor. Molt útil per seguir una variable dins el codi d'un programa.
- **#** trobar la precedent paraula en el text igual a la que hi ha sota el cursor.
- **%** amb el caràcter % es salta als parèntesis que van aparellats a la posició actual del cursor. Molt útil per seguir expressions de parèntesi aniuades.
- **o O** les lletres o tant minúscula com majuscule generen una nova línia en blanc sota la línia actual i es passa al mode inserció a l'inici d'aquesta línia.

Desar una còpia

En tot moment es pot desar una còpia del document amb un altre nom, similar a l'opció 'save as'. Atenció: es desa el document amb un altre nom però es continua editant el propi document. Així si per exemple es fa "vim carta.txt" i durant l'edició es vol desar una còpia es pot fer:

- w nom-a-desar

És a dir, escape, dos punts i la tecla w i el nom amb el que cal desar la còpia. És important entendre que el document que es continua editant és l'original (carta.txt en l'exemple) i no la còpia desada.

Buscar contingut

L'editor vim té el mateix funcionament que l'ordre less per fer recerca de contingut. Podem buscar un subtext en el document fent:

- **esc /text-a-buscar** amb aquesta combinació buscara el text indicat
- **n** de next per trobar la següent ocurrència del subtext.
- **N** per trobar la anterior.

Identació i moure blocs

Quan s'escriu codi de programació és molt usual indentar el codi per fer-lo més comprensible. Existeixen tot de combinacions de tecles per facilitar el posicionament del codi més a la dreta o més a l'esquerra. Algunes d'aquestes opcions són:

- **^t** desplaça el text de la línia a la següent tabulació (a la dreta).
- **^d** desplaça el text de la línia a la tabulació anterior (a l'esquerra)
- **>> <<** Igual que ^t i ^d desplacen el text a la següent o anterior tabulació.
- **n>> n<<** Desplaça n línies a la següent o anterior tabulació.

Definició d'opcions

L'editor vim té predefinides innumbrables opcions de funcionament i visualització que podem modificar. Des del mode comande es poden escriure comandes de personalització, per exemple:

```
set tabstop=4
set shiftwidth=4
set cursorline
set number
set paste
```

Definició d'opcions per defecte

Per establir opcions per defecte al vim de manera que cada usuari es pugui personalitzar el funcionament al seu gust, vim llegeix les opcions de configuració d'un fitxer anomenat:

- **.vimrc**

Aquest fitxer es troba en el home de l'usuari. És un fitxer ocult que té el típic nom de fitxer de control o configuració (<nom>rc). Cada usuari es pot personalitzar el seu fitxer.

Exemple de fitxer:

```
$ cat ~/.vimrc
set tabstop=4
set shiftwidth=4
set number
set cursorline
```

Edició de codi i identació

Una qüestió recurrent en els programadors de codi és com gestionar l'identat del codi. Generalment les línies de codi que han d'anar identades es prem la tecla <tab> que salta a la següent tabulació, típicament 8 espais. Segons el codi o el llenguatge un salt de 8 espais és molt i desplaça el codi massa a la dreta per fer-lo entenedor.

Amb l'opció:

- **set tabstop 4** se li indica que cada cop que es prem <tab> ha de saltar 4 espais. Que les tabulacions van de 4 en quatre. Es pot establir el valor que es volgui.

Ara bé, quan desplacem el codi amb ^t ^d o amb >> i <<, el vim utilitza el valor definit en la directiva:

- **set shiftwidth 4** per exemple indica que cada desplaçament (shift) ha de ser de 4 espais.

És recomanable establir el mateix valor a les dues directives (tabstop i shiftwidth) d'aquesta manera saltin a les mateixes posicions.

Finalment hi ha un altre aspecte que moltes vegades torna boigs als programadors, si la identació s'ha fet amb espais o amb tabuladors. A vegades el programador prem <tab>, d'altres escriu com un ruquet els espais equivalents... Llavors el codi no és uniforme, hi ha identacions tabulades i d'altres espaiades. Hi ha llenguatges com el Python que això no ho toleren.

Un mecanisme per evitar tenir identacions barrejades d'espais i tabulacions és tabular sempre el codi només amb espais, i fer que el propi vim quan premem tabulador hi posi en lloc d'una tabulació un conjunt d'espais. Cal usar la directiva:

- **set expandtabs**

Pàgines de documentació:

<https://www.vim.org/docs.php>

<https://www.cs.oberlin.edu/~kuperman/help/vim/indenting.html>

Exercicis d'exemple:

1. Crear un fitxer anomenat [carta.txt](#) al directori actiu, amb el següent contingut i desa'l.

aquest és un text qualsevol.
podem escriure múltiples línies
com per exemple aquestes.

l·listat de noms:

pau pou
marta pou
anna pou
jordi mas
pere mas
admin sys

2. Edita el fitxer [carta.txt](#) del directori actiu i fes:
 - elimina amb *dd* el nom "pau pou"
 - elimina amb *3dd* les tres línies següents
 - amb *u* anul·la l'última acció.
 - torna a prémer *u* per anul·lar l'esborrat de "pau pou".
3. Desa una còpia anomenada [carta.bk](#) del contingut actual. Des d'una altra sessió fes un cat per mostrar el contingut de carta.bk
4. Edita el fitxer [carta.txt](#) i fes les accions següents:
 - Ves al final del document prement *G*.
 - Ves al principi del document prement *gg*.
 - Ves a la cinquena línia prement *5G*.
 - Insereix una línia nova a sota prement *o* i escriu el contingut: "julia mas".
5. Edita el fitxer carta.txt i situa el cursor a la cinquena línia sobre la paraula "mas", sense estar en mode inserció. Fes les accions:
 - Prem *** i observa que es va desplaçant a cada paraula "mas" que troba en el text.
 - Prem *#* i observa que es va desplaçant a la anterior aparició de la paraula "mas".
 - Situa el cursor a "julia mas" (sobre mas).
 - Prem *dd* per esborrar una línia.
 - Prem el caràcter punt *.* dues vegades per repetir la última acció dos cops.
 - Surt del fitxer sense desar amb *q!*.
6. Edita el text [carta.txt](#) i practica:
 - Situat al principi del document amb *gg*.
 - Busca la paraula pou fent escape /pou
 - Busca la següent aparició de la paraula pou.
 - Busca la anterior aparició de la paraula pou.
7. Escriu el codi següent en un fitxer anomenat [script.sh](#):

```
# @edt ASIX M01-ISO 2020 - 2021
# vim
# exemple de capçalera

if [ $# -ne 1]; then
    echo "Error: número arguments incorrecte"
    echo "usage: $0 num"
    exit 1
fi
doble=$((num*2))
quadrat=$((num*num))
crazy=$((num*(num+10)+(2*num)))
echo "$num $doble $quadrat $crazy"
exit 0
```

8. Edita el fitxer *script.sh* i practica:

- Col·loca el cursor en el primer parèntesi de la línia “doble” i prem el caràcter % un parell de vegades. Observa que es desplaça entre el parèntesi d’obertura i el de tancament.
- Col·loca el cursor al tercer parèntesi de la línia “crazy”. Torna a verificar el funcionament del caràcter de control %.

9. Edita el fitxer *script.sh* i practica:

- En mode inserció situa el cursor a la línia de dins del if i practica ^t i ^d.
- En mode normal (no edició) practica >> i <<.
- Situa el cursor en el if i prem 5>>. Observa que es desplaça tot el bloc.

10. Practica cada un dels casos següents:

1. d<leftArrow> will delete current and left character
2. d\$ will delete from current position to end of line
3. d^ will delete from current backward to first non-white-space character
4. d0 will delete from current backward to beginning of line
5. dw deletes current to end of current word (including trailing space)
6. db deletes current to beginning of current word