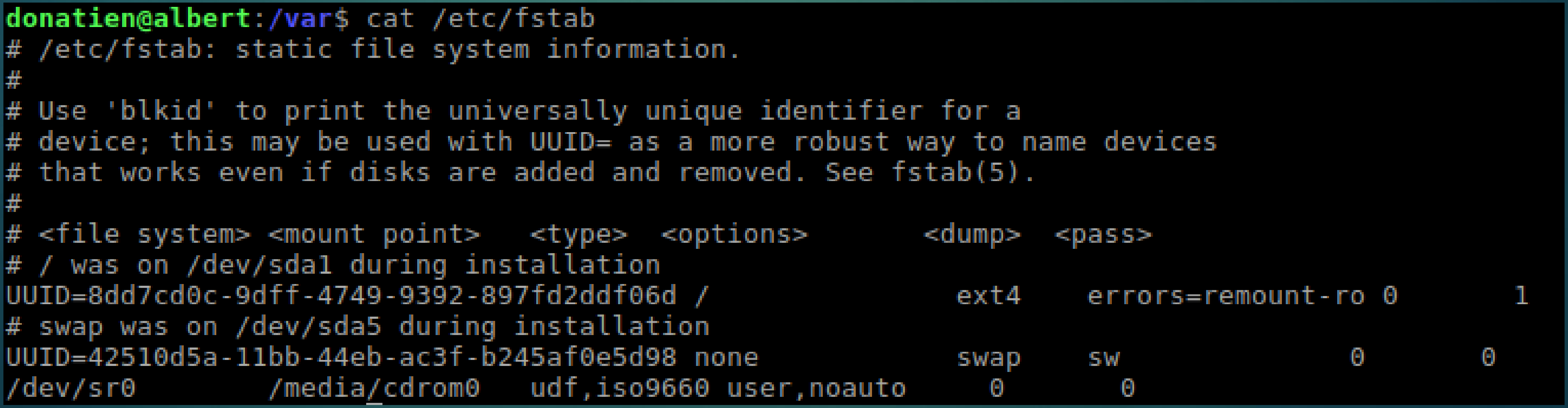
I106B
5. nano

HTTPS://DOC.UBUNTU-FR.ORG/NANO

Editeur de texte en mode terminal

- Beaucoup d'aspects de Linux sont gérés par des fichiers textes (cf. la philosophie Linux).

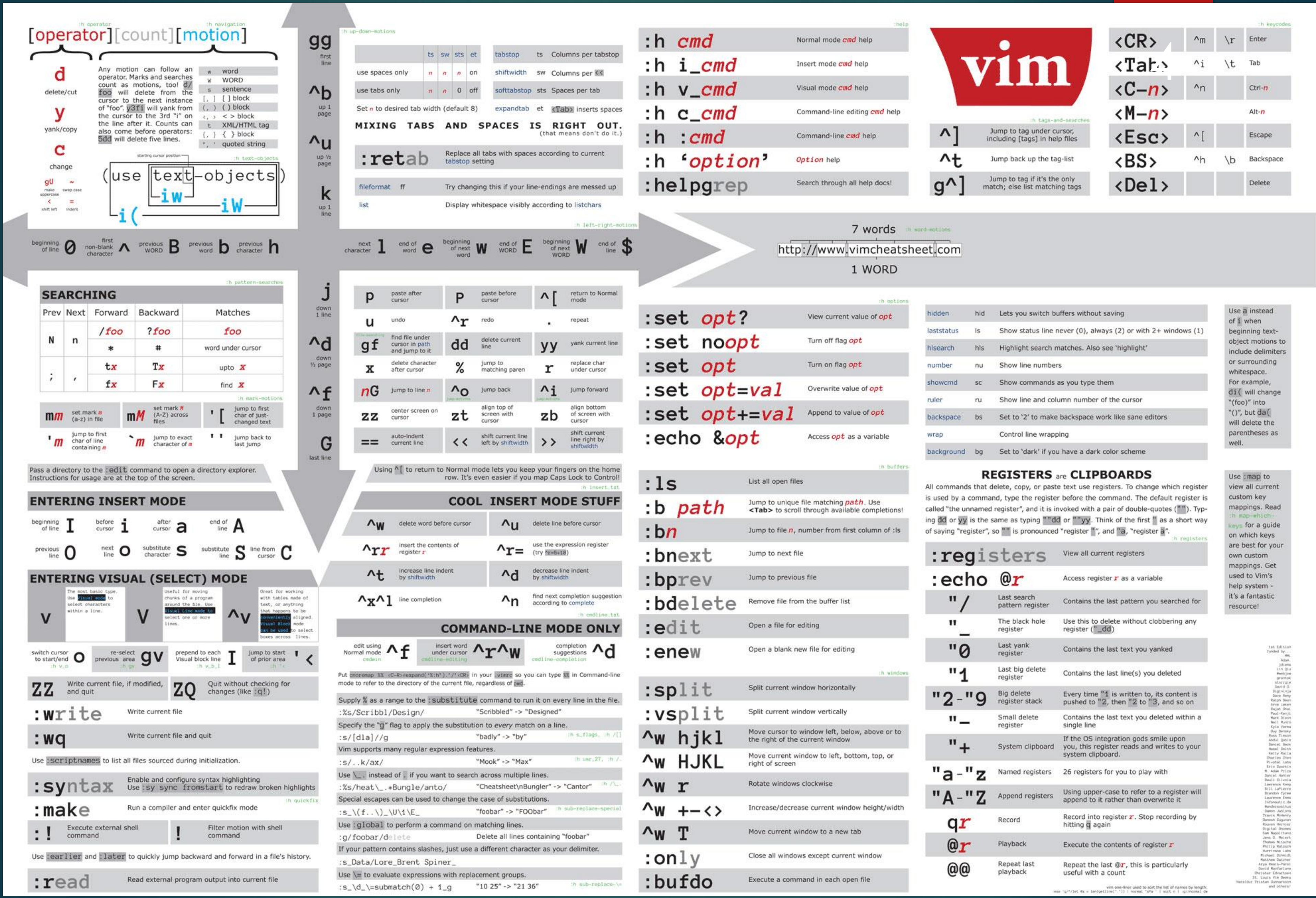
  ex : /etc/fstab gère le montage du système de fichiers



```
donatien@albert:/var$ cat /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sda1 during installation
UUID=8dd7cd0c-9dff-4749-9392-897fd2ddf06d /               ext4    errors=remount-ro 0       1
# swap was on /dev/sda5 during installation
UUID=42510d5a-11bb-44eb-ac3f-b245af0e5d98 none            swap    sw              0       0
/dev/sr0        /media/cdrom0   udf,iso9660 user,noauto     0       0
```

Editeur de texte

- Il faut un éditeur de texte pour modifier ces configurations.
- Un éditeur qu'on trouve sur toutes les distributions Linux
- Historiquement : vi
- Version moderne : vim



:h operator :h navigation

[operator][count][motion]

d	delete/cut
y	yank/copy
c	change
gU	make uppercase
~	swap case
<	shift left
=	indent

Any motion can follow an operator. Marks and searches count as motions, too! d/foo will delete from the cursor to the next instance of "foo". y3fi will yank from the cursor to the 3rd "i" on the line after it. Counts can also come before operators: 5dd will delete five lines.

w	word
W	WORD
s	sentence
[,]	[ ] block
(,)	( ) block
<, >	< > block
t	XML/HTML tag
{, }	{ } block
", '	quoted string

starting cursor position :h text-objects

```
(use text-objects)
```
iw iW i(

:h up-down-motions

gg	first line
^b	up 1 page
^u	up ½ page
k	up 1 line
j	down 1 line
^d	down ½ page
^f	down 1 page
G	last line

	ts	sw	sts	et
use spaces only	n	n	n	on
use tabs only	n	n	0	off

Set n to desired tab width (default 8)

tabstop	ts	Columns per tabstop
shiftwidth	sw	Columns per <<
softtabstop	sts	Spaces per tab
expandtab	et	<Tab> inserts spaces

MIXING TABS AND SPACES IS RIGHT OUT.
(that means don't do it.)

:retab	Replace all tabs with spaces according to current tabstop setting
fileformat ff	Try changing this if your line-endings are messed up
list	Display whitespace visibly according to listchars

:h left-right-motions

0	beginning of line
^	first non-blank character
B	previous WORD
b	previous word
h	previous character
l	next character
e	end of word
w	beginning of next word
E	end of WORD
W	beginning of next WORD
$	end of line

:help

:h cmd	Normal mode cmd help
:h i_cmd	Insert mode cmd help
:h v_cmd	Visual mode cmd help
:h c_cmd	Command-line editing cmd help
:h :cmd	Command-line cmd help
:h 'option'	Option help
:helpgrep	Search through all help docs!

vim

:h tags-and-searches

^]	Jump to tag under cursor, including [tags] in help files
^t	Jump back up the tag-list
g^]	Jump to tag if it's the only match; else list matching tags

:h keycodes

<CR>	^m	\r	Enter
<Tab>	^i	\t	Tab
<C-n>	^n		Ctrl-n
<M-n>			Alt-n
<Esc>	^[		Escape
<BS>	^h	\b	Backspace
<Del>			Delete

7 words :h word-motions

```
http://www.vimcheatsheet.com
```

1 WORD

:h pattern-searches

SEARCHING

Prev	Next	Forward	Backward	Matches
N	n	/foo	?foo	foo
		*	#	word under cursor
;	,	tx	Tx	upto x
		fx	Fx	find x

:h mark-motions

mm	set mark m (a-z) in file
mM	set mark M (A-Z) across files
'[	jump to first char of just-changed text
'm	jump to first char of line containing m
`m	jump to exact character of m
''	jump back to last jump

Pass a directory to the :edit command to open a directory explorer. Instructions for usage are at the top of the screen.

p	paste after cursor
P	paste before cursor
^[	return to Normal mode
u	undo
^r	redo
.	repeat
gf	find file under cursor in path and jump to it
dd	delete current line
yy	yank current line
x	delete character after cursor
%	jump to matching paren
r	replace char under cursor
nG	jump to line n
^o	jump back
^i	jump forward
zz	center screen on cursor
zt	align top of screen with cursor
zb	align bottom of screen with cursor
==	auto-indent current line
<<	shift current line left by shiftwidth
>>	shift current line right by shiftwidth

Using ^[to return to Normal mode lets you keep your fingers on the home row. It's even easier if you map Caps Lock to Control!

:h options

:set opt?	View current value of opt
:set noopt	Turn off flag opt
:set opt	Turn on flag opt
:set opt=val	Overwrite value of opt
:set opt+=val	Append to value of opt
:echo &opt	Access opt as a variable

hidden	hid	Lets you switch buffers without saving
laststatus	ls	Show status line never (0), always (2) or with 2+ windows (1)
hlsearch	hls	Highlight search matches. Also see 'highlight'
number	nu	Show line numbers
showcmd	sc	Show commands as you type them
ruler	ru	Show line and column number of the cursor
backspace	bs	Set to '2' to make backspace work like sane editors
wrap		Control line wrapping
background	bg	Set to 'dark' if you have a dark color scheme

Use a instead of i when beginning text-object motions to include delimiters or surrounding whitespace. For example, di(will change "(foo)" into "()", but da(will delete the parentheses as well.

ENTERING INSERT MODE

I	beginning of line
i	before cursor
a	after cursor
A	end of line
O	previous line
o	next line
s	substitute character
S	substitute line
C	line from cursor

ENTERING VISUAL (SELECT) MODE

v	The most basic type. Use Visual mode to select characters within a line.
V	Useful for moving chunks of a program around the file. Use Visual Line mode to select one or more lines.
^v	Great for working with tables made of text, or anything that happens to be conveniently aligned. Visual Block mode can be used to select boxes across lines.
o	switch cursor to start/end :h v_o
gv	re-select previous area :h gv
I	prepend to each Visual block line :h v_b_I
'<	jump to start of prior area :h '<

ZZ	Write current file, if modified, and quit
ZQ	Quit without checking for changes (like :q!)
:write	Write current file
:wq	Write current file and quit

Use :scriptnames to list all files sourced during initialization.

:syntax	Enable and configure syntax highlighting Use :sy sync fromstart to redraw broken highlights
:make	Run a compiler and enter quickfix mode
:!	Execute external shell command
!	Filter motion with shell command

:h quickfix

Use :earlier and :later to quickly jump backward and forward in a file's history.

:read	Read external program output into current file

:h insert.txt

COOL INSERT MODE STUFF

^w	delete word before cursor
^u	delete line before cursor
^rr	insert the contents of register r
^r=	use the expression register (try ^r=5+10)
^t	increase line indent by shiftwidth
^d	decrease line indent by shiftwidth
^x^l	line completion
^n	find next completion suggestion according to complete

:h cmdline.txt

COMMAND-LINE MODE ONLY

^f	edit using Normal mode (cmdwin)
^r^w	insert word under cursor (cmdline-editing)
^d	completion suggestions (cmdline-completion)

Put cnoremap %% <C-R>=expand('%:h').'/'<CR> in your .vimrc so you can type %% in Command-line mode to refer to the directory of the current file, regardless of pwd.

Supply % as a range to the :substitute command to run it on every line in the file.

```
:%s/Scribbl/Design/                 "Scribbled" -> "Designed"
```

Specify the "g" flag to apply the substitution to *every* match on a line.

```
:s/[dla]//g                         "badly" -> "by"              :h s_flags, :h /[]
```

Vim supports many regular expression features.

```
:s/..k/ax/                          "Mook" -> "Max"              :h usr_27, :h /.
```

Use _. instead of . if you want to search across multiple lines.

```
:%s/heat\_.*Bungle/anto/            "Cheatsheet\nBungler" -> "Cantor"   :h /\_.
```

Special escapes can be used to change the case of substitutions.

```
:s_\(f..\)_\U\1\E_                  "foobar" -> "FOObar"         :h sub-replace-special
```

Use :global to perform a command on matching lines.

```
:g/foobar/delete                    Delete all lines containing "foobar"
```

If your pattern contains slashes, just use a different character as your delimiter.

```
:s_Data/Lore_Brent Spiner_
```

Use \= to evaluate expressions with replacement groups.

```
:s_\d_\=submatch(0) + 1_g           "10 25" -> "21 36"           :h sub-replace-\=
```

:h buffers

:ls	List all open files
:b path	Jump to unique file matching path. Use <Tab> to scroll through available completions!
:bn	Jump to file n, number from first column of :ls
:bnext	Jump to next file
:bprev	Jump to previous file
:bdelete	Remove file from the buffer list
:edit	Open a file for editing
:enew	Open a blank new file for editing

:h windows

:split	Split current window horizontally
:vsplit	Split current window vertically
^w hjkl	Move cursor to window left, below, above or to the right of the current window
^w HJKL	Move current window to left, bottom, top, or right of screen
^w r	Rotate windows clockwise
^w +-<>	Increase/decrease current window height/width
^w T	Move current window to a new tab
:only	Close all windows except current window
:bufdo	Execute a command in each open file

REGISTERS are CLIPBOARDS

All commands that delete, copy, or paste text use registers. To change which register is used by a command, type the register before the command. The default register is called "the unnamed register", and it is invoked with a pair of double-quotes (""). Typing dd or yy is the same as typing ""dd or ""yy. Think of the first " as a short way of saying "register", so "" is pronounced "register "", and "a, "register a".

:h registers

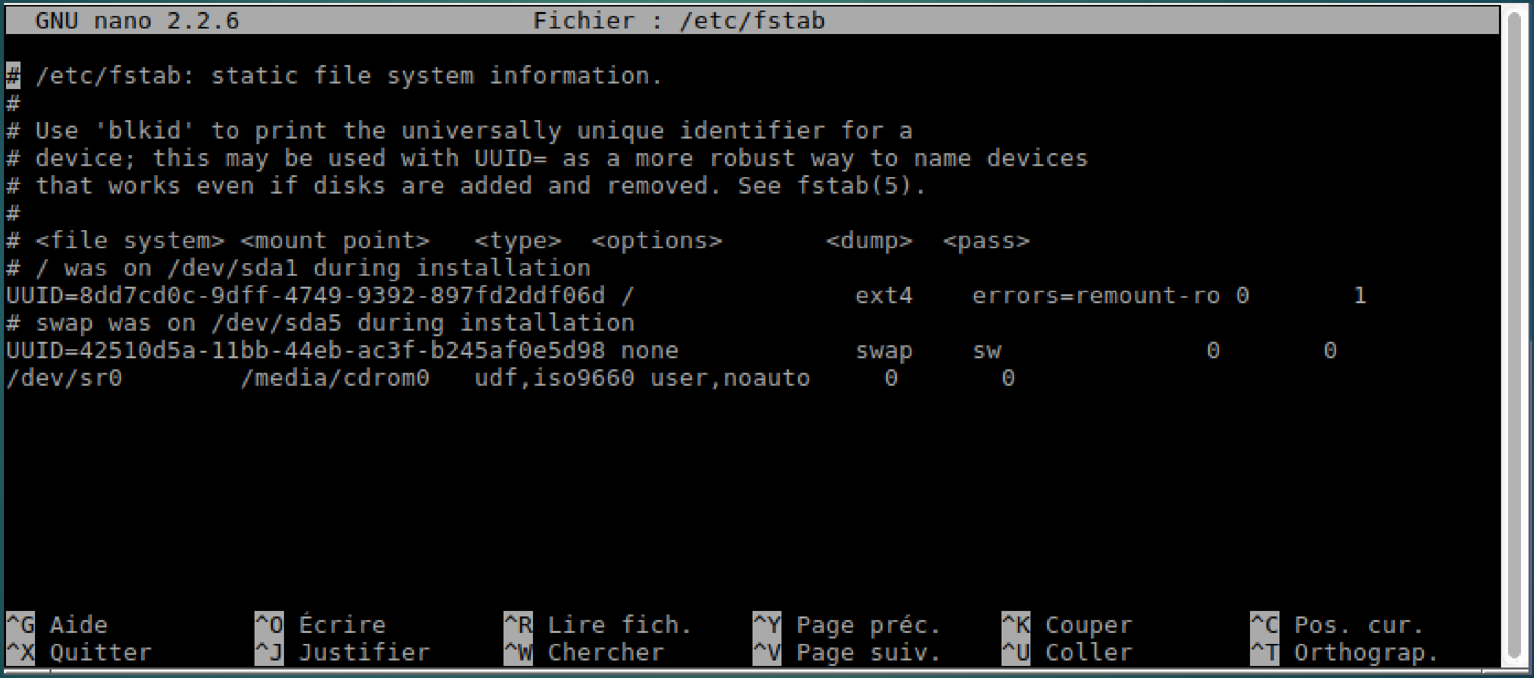
:registers		View all current registers
:echo @r		Access register r as a variable
"/	Last search pattern register	Contains the last pattern you searched for
"_	The black hole register	Use this to delete without clobbering any register ("_dd)
"0	Last yank register	Contains the last text you yanked
"1	Last big delete register	Contains the last line(s) you deleted
"2-"9	Big delete register stack	Every time "1 is written to, its content is pushed to "2, then "2 to "3, and so on
"-	Small delete register	Contains the last text you deleted within a single line
"+	System clipboard	If the OS integration gods smile upon you, this register reads and writes to your system clipboard.
"a-"z	Named registers	26 registers for you to play with
"A-"Z	Append registers	Using upper-case to refer to a register will append to it rather than overwrite it
qr	Record	Record into register r. Stop recording by hitting q again
@r	Playback	Execute the contents of register r
@@	Repeat last playback	Repeat the last @r, this is particularly useful with a count

Use :map to view all current custom key mappings. Read :h map-which-keys for a guide on which keys are best for your own custom mappings. Get used to Vim's help system - it's a fantastic resource!

1st Edition funded by... MML Adam. jdiems Lin Qiu #webjoe grantok storrgie David O. Digininja Dave Remy Ralph Bean Arve Leken Rajat Dhai Paul-Kenji Mark Dixon Neil Munro Kyle Verma Guy Bensky Ross Timson Abdul Qabiz Daniel Beck Hamel Smith Kelly Raila Charles Chen Pivotal Labs Eric Sporkin M. Adam Price Daniel Hahler Rauli Silvola Lawrence Kemp Bill LaPierre Brandon Tyree Laurence Emms Infonautic.de #andersonsthus Damon Jablons Travis McHenry Danesh Suguran Rouven Hernier Digital Gnomes Sam Napolitano Jens O. Meiert Thomas Nitsche Philip Ratzsch Hurricane Labs Michael Schmidt Matthew Datcher Arya Beals-Parsi David Macfarlane Christer Edvartsen St. Louis Vim Geeks Haraldur Tristan Gunnarsson and others!

vim one-liner used to sort the list of names by length:
[illegible]

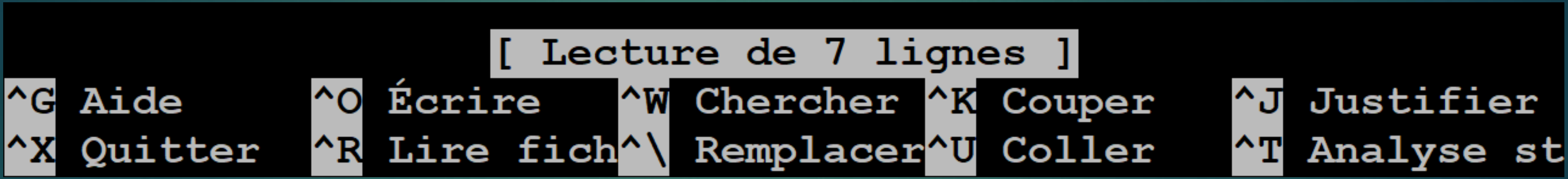
Editeur de texte compréhensible par un humain normalement constitué : **nano**



```
  GNU nano 2.2.6                    Fichier : /etc/fstab

# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sda1 during installation
UUID=8dd7cd0c-9dff-4749-9392-897fd2ddf06d /               ext4    errors=remount-ro 0       1
# swap was on /dev/sda5 during installation
UUID=42510d5a-11bb-44eb-ac3f-b245af0e5d98 none            swap    sw              0       0
/dev/sr0        /media/cdrom0   udf,iso9660 user,noauto     0       0









^G Aide          ^O Écrire        ^R Lire fich.    ^Y Page préc.    ^K Couper        ^C Pos. cur.
^X Quitter       ^J Justifier     ^W Chercher      ^V Page suiv.    ^U Coller        ^T Orthograp.
```

Raccourcis

Les principaux raccourcis de nano sont affichés au bas de l’écran :

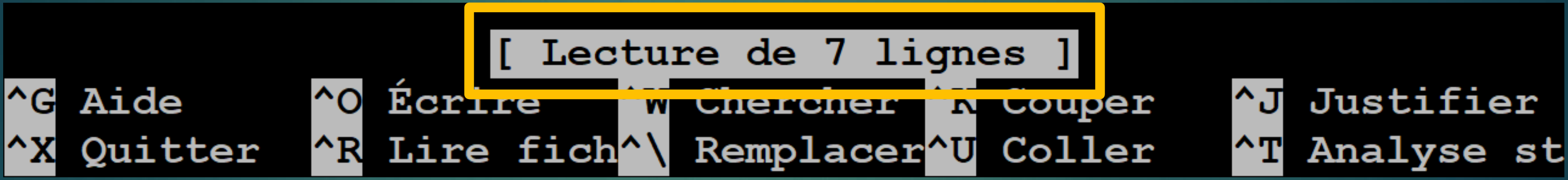


```
                         [ Lecture de 7 lignes ]
^G Aide          ^O Écrire     ^W Chercher  ^K Couper      ^J Justifier
^X Quitter       ^R Lire fich^\ Remplacer^U Coller      ^T Analyse st
```

- **Ctrl-K** → coupe toute la ligne courante
- Pour couper une sélection quelconque de texte, placer une marque sur le curseur avec **Alt-A**, puis déplacer le curseur pour sélectionner du texte. Ctrl-K permet alors de couper cette sélection
- **Ctrl-U** → colle une sélection

Numéro de ligne

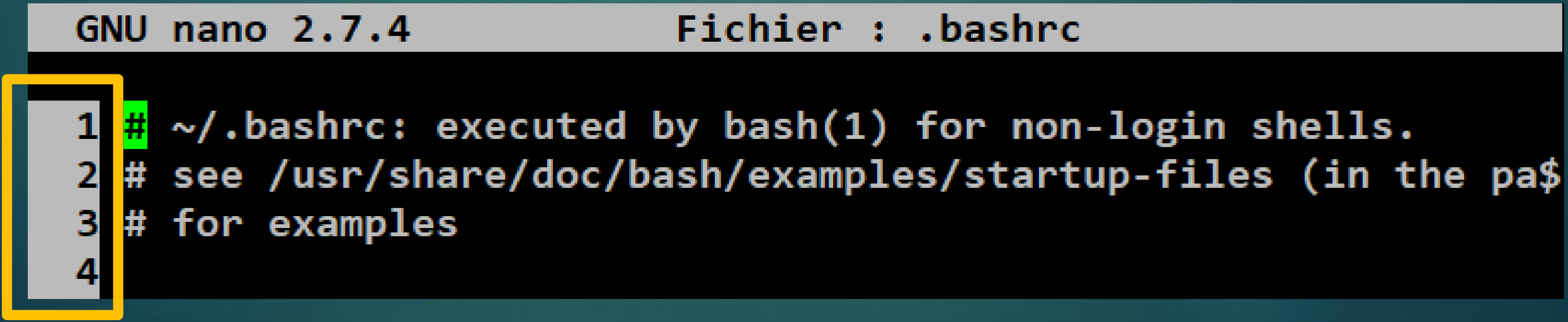
- Par défaut, nano affiche le nombre de lignes du fichier



```
                         [ Lecture de 7 lignes ]
^G Aide        ^O Écrire    ^W Chercher  ^K Couper     ^J Justifier
^X Quitter     ^R Lire fich^\ Remplacer^U Coller       ^T Analyse st
```

- Pour afficher le numéro des lignes, il faut ajouter l'option **-l** (voir fichier *I106B_Linux_@_home*) :

```
nano -l .bashrc
```



```
  GNU nano 2.7.4              Fichier : .bashrc

 1 # ~/.bashrc: executed by bash(1) for non-login shells.
 2 # see /usr/share/doc/bash/examples/startup-files (in the pa$
 3 # for examples
 4
```

Autres commandes utiles

- **cat *fichier*** : affiche le contenu de *fichier*
- **more *fichier*** : affiche le contenu de *fichier*, écran par écran
- **less *fichier*** : identique à *more*, mais peut revenir en arrière et effectuer des recherches (comme dans le manuel *man*)