Session 3: Vim

P. S. Langeslag

1 November 2018

`sed` Replacement

Operation	Effect
`sed s/He/She/ file`	Replace all instances of `He` in `file` with `She` (`-s` for "substitute"); print result to `stdout`, leaving `file` untouched
`sed -n s/He/She/p file`	Ditto, but print affected lines only (`-n` for "no output"; `p` for "print")
`sed -n s/He/She/pI file`	Ditto, but case insensitive (`I` for "insensitive"); note unwanted effects
`sed -n s/He/She/gpI file`	Ditto, replace beyond the first match in the string (sentence) (`g` for "global")
`sed -i.bak s/He/She/ file`	Ditto, but save results to original file and copy original file to backup file `file.bak` (`-i` for "insert"); no output
`sed -i s/He/She/ file`	Ditto, but without backup file! Risky!
`sed -i s/He/She/ *`	Ditto, but for every file in working directory. Highly risky!

Teleprinter (Teletype/TTY, *Fernschreiber*)

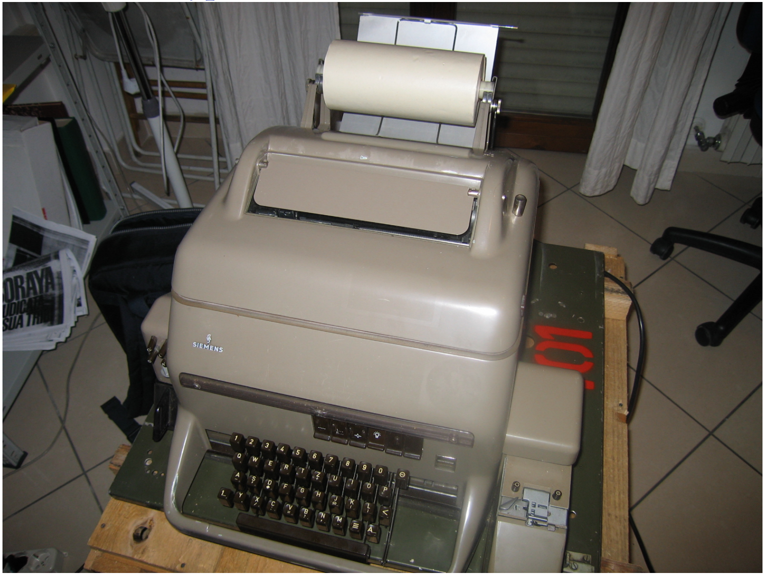

Figure: Siemens Fernschreiber 100 (copyleft WMC user)

A Selective History of Modern Editors

1965–66	QED	Line editor, exclusively designed for **teleprinters**. UNIX developer Ken Thompson modified it for the CTSS timesharing operating system and added support for regular expressions.
1971	`ed`	**Modal** UNIX editor inspired on QED, supporting regular expressions and input from `stdin`
1973	`sed`	UNIX line-oriented **stream editor** based on `ed`, with regular expression capability. NB in-place editing was only added in the GNU implementation.
1976	`ex`	BSD editor modifying `ed` for use on **video terminals**
1978	`vi`	A version of `ex` that **defaults to visual mode**
1991	Vim	Highly **customizable** fork of `vi`

Vim Modes

1. Normal mode (command mode)
2. Insert mode (accessed through insert commands `i`, `a`, `A`, `c`, `o`, `O`)
3. Visual mode (`v`)

- `<ESC>` returns you to normal mode.
- In normal mode, `<ESC>` cancels any unfinished commands

Interlinear Navigation

`j`	One line down
`k`	One line up
`5j`	Five lines down, etc.
`gj`	One screen line down (when wrapped)
`gk`	One screen line up (when wrapped)
`5gj`	Five screen lines down (when wrapped), etc.
`H`	To top of current screen
`M`	To vertical centre of current screen
`L`	To bottom of current screen
`Ctrl+F`	One screen page down (or use `<PgDown>`)
`Ctrl+U`	One screen page up (or use `<PgUp>`)
`gg`	To top of file
`G`	To bottom of file
`:n`	To line n

- NB I have mapped the cursor keys to `gj`, `gk`!
- See `:help up-down-motions`

Intralinear Navigation

`h, l`	Single-step cursor navigation
`5h`	Five characters to the left, etc.
`w`	To start of next word
`5w`	Five words to the right, etc.
`e`	To end of current/next word
`b`	To beginning of current/last word
`0`	To start of line
`^`	To first nonblank character in line
`$`	To end of line
`gm`	To middle of screen line
`5|`	To column 5 (i.e. character position 5)

- See `:help word-motions`

Text Object Navigation

`(, )`	One sentence back or forward
`5(, 5)`	Five sentences back or forward, etc.
`{, }`	One paragraph back or forward
`[[, ]]`	One section back or forward
`%`	Go to matching parenthesis, bracket, or curly brace (of the next such opening item in the current line); in e.g. LaTeXit matches opening/closing tags.

- See
  - `:help object-motions`
  - `:help various-motions`

Modification Commands

`i, a, A`	Insert/append text at cursor position/to end of line
`o, O`	Insert text on a new line after or before current
`x`	Delete current character
`5x`	Delete five characters starting at current
`dw`	Delete remainder of current word
`5dw`	Delete five words starting at current character
`db`	Delete back to beginning of current word
`D`	Delete rest of line
`dd`	Delete current line
`5dd`	Delete five lines starting at current
`r[char]`	Replace a single character
`5r[char]`	Replace five characters with five times `[char]`
`cw`	Delete remainder of word and drop into insert mode
`cb`	Delete beginning of word and drop into insert mode
`c5w`	Delete five words and drop into insert mode
`c$`	Delete remainder of line and drop into insert mode

▶ See `:help change.txt`

Undo and Redo

`u`	Undo last operation
`CTRL+R`	Redo last operation
`U`	Undo all the last operations on last modified line
`:undol[ist]`	List branching nodes
`:u[ndo] n`	Go to text state following change number `n`
`g-`	Go to earlier text state
`5g-`	Go five text states back up the undo tree
`g+`	Go to newer text state
`:earlier 1h`	Go to text state of one hour ago
`:later 30s`	Go to text state of 30 seconds after current undo node

- `u` and `CTRL+R` treat undo history as a single branch
- `g-`, `g+`, `:earlier`, and `:later` move through all changes
- Everything you do between entering and leaving insert mode counts as one change; train yourself to leave it regularly!
- See `:help undo.txt`

Yank and Put

`yw`	Copy remainder of word to default register
`5yw`	Copy five words to register, starting at current character
`y$`	Copy remainder of line to register
`yy`	Copy current line to register
`5yy`	Copy five lines to register, starting at current
`p`	Paste from default register
`:reg`	Display registers
`"5p`	Paste fifth-last stored string
`"ayy`	Copy full line to named register `a`
`"bdw`	Delete word and copy to named register `b`
`"ap`	Paste from named register `a`
`5p`	Paste from default register five times

- Anything you cut using `x` or `d` also goes into the buffer
- See `:help copy-move`

Visual Mode

`v` to enter visual mode; then use word motions to make a visual selection, `y` to copy to buffer, or `x` to cut.

Search and Repeat

`/`	Input search query (accepts regular expressions)
`n`	Repeat search (navigate to next hit)
`N`	Repeat search (navigate to previous hit)
`.`	Apply last edit to the current position

- Vim's regex implementation differs from PCRE!
- See `:help pattern.txt`; http://vimregex.com

File Operations

`:w`	Save file
`:w filename`	Save file to `filename`
`:q`	Exit without saving
`:q!`	Exit without saving; disregard warnings
`:wq` / `ZZ`	Save file and exit

- When `file` is not successfully closed, the swap file `.file.swp` remains and you receive a warning prompt when next opening it. Check that the swap file is identical to the file itself, then run

  ```
  rm .file.swp
  ```

Settings

`:set ic`	Ignore case (I have set this as default)
`:set noic`	Heed case
`:set wrap`	Use line wrapping (default)
`:set nowrap`	Disable line wrapping
`:syntax on`	Enable syntax highlighting (default)
`:syntax off`	Disable syntax highlighting

- See `:help options`
- For help on individual options, use single quotes: `:help 'syntax'`
- Set persisent options in `~/.vimrc`

Interacting With Files and Programs

`:r[ead] file`	Read contents of `file` into current file
`:so[urce] file`	Interpret `file` as a sequence of Vim commands
`:shell`	Open a shell; return with `exit`
`:!cmd`	Run `cmd`; e.g. `:!ls`

- See
  - `:help :r`
  - `:help :so`
  - `:help :shell`
  - `:help :!cmd`

Windows

`:new` / `CTRL+W n`	Start a new file in a new window (horizontal split)
`:vne`	Start a new file in a new window (vertical split)
`CTRL+W j` (or `<DOWN>`)	Make the lower window active
`CTRL+W k` (or `<UP>`)	Make the upper window active
`:q`	Close the active window
`:on[ly]`	Close all windows except the active one

Getting Help

`:help`	Main help file
`:help quickref`	Terse list of motions and commands
`:help usr_toc.txt`	Table of contents of user manual

- Use the names of commands, options, and tags to get help, e.g.
  - `:help c` for help on the `c` command
  - `:help up-down-motions` for interlinear navigation commands
  - `:help 'ic'` for an explanation of the `ignorecase` option
- In the help files, tags serve as hyperlinks:
  - `CTRL+]` to visit
  - `CTRL+O` to return
- Close a help pane as you would any read-only Vim file, with `:q`

vi Mode on the Command Line

- Add the following line to ~/.bashrc:

```
set -o vi
```

Vim on Your Own System

OS X
Included; open a terminal and enter `vim`

Linux
Included or in package repositories; install the `gvim` package for full clipboard functionality

Windows
Download from http://www.vim.org

Terminal or Graphical?
Whatever works for you

References and Recommended Reading

Moolenaar, Bram. “Vim Online,” n.d. http://www.vim.org.

Raisky, Oleg. “Vim Regular Expressions 101,” n.d. http://vimregex.com.

Robbins, Arnold. *vi and Vim Editors: Pocket Reference*. 2nd ed. Sebastopol, CA: O’Reilly, 2011.

Robbins, Arnold, Elbert Hannah, and Linda Lamb. *Learning the vi and Vim Editors: Pocket Reference*. 7th ed. Sebastopol, CA: O’Reilly, 2008.