

Editorul Vim

CDL - Cursul 2

Vlad Dogaru
ddvlad@rosedu.org

ROSEdu

20 martie 2009

Where do we stand?

Până acum, la CDL, am învățat:

Where do we stand?

Până acum, la CDL, am învățat:

- ce este Free/Libre/Open Source Software

Where do we stand?

Până acum, la CDL, am învățat:

- ce este Free/Libre/Open Source Software
- cum să comunicăm în cadrul unui proiect

Where do we stand?

Până acum, la CDL, am învățat:

- ce este Free/Libre/Open Source Software
- cum să comunicăm în cadrul unui proiect
- cum să automatizăm procesul de compilare

Where do we stand?

Până acum, la CDL, am învățat:

- ce este Free/Libre/Open Source Software
- cum să comunicăm în cadrul unui proiect
- cum să automatizăm procesul de compilare
- cum să ținem sursele programului undeva sigur

Where do we stand?

Până acum, la CDL, am învățat:

- ce este Free/Libre/Open Source Software
- cum să comunicăm în cadrul unui proiect
- cum să automatizăm procesul de compilare
- cum să ținem sursele programului undeva sigur

Când învățăm să *scriem* cod?

Acum!

Acum!

De fapt, azi, dar puțin mai târziu.

Acum!

De fapt, azi, dar puțin mai târziu.
Acum învățăm cum să *edităm*, nu cum să scriem cod.

Warning!

Warning!

- Morocňanos

Warning!

- Morocănos
- Prezentare tehnică

Warning!

- Morocănos
- Prezentare tehnică
- Exerciții puține

Warning!

- Morocănos
- Prezentare tehnică
- Exerciții puține
- Puneți mâna singuri

Warning!

- Morocănos
- Prezentare tehnică
- Exerciții puține
- Puneți mâna singuri
 - nu neapărat pe Vim

Warning!

- Morocănos
- Prezentare tehnică
- Exerciții puține
- Puneți mâna singuri
 - nu neapărat pe Vim
 - e important să știm bine *un* editor, nu editorul *meu*

Importanța eficienței

De ce e important să scriem cod eficient:

Importanța eficienței

De ce e important să scriem cod eficient:

- când scriem repede, scriem mai mult

De ce e important să scriem cod eficient:

- când scriem repede, scriem mai mult
 - mai mult nu înseamnă mai bine

De ce e important să scriem cod eficient:

- când scriem repede, scriem mai mult
 - mai mult nu înseamnă mai bine
 - but we're sometimes "in the zone"

De ce e important să scriem cod eficient:

- când scriem repede, scriem mai mult
 - mai mult nu înseamnă mai bine
 - but we're sometimes "in the zone"
- dacă scriem mult și ineficient, o să ajungem să urâm ce facem

De ce e important să scriem cod eficient:

- când scriem repede, scriem mai mult
 - mai mult nu înseamnă mai bine
 - but we're sometimes "in the zone"
- dacă scriem mult și ineficient, o să ajungem să urâm ce facem
 - Da, o să ne doară mâinile.

Importanța eficienței

De ce e important să scriem cod eficient:

- când scriem repede, scriem mai mult
 - mai mult nu înseamnă mai bine
 - but we're sometimes "in the zone"
- dacă scriem mult și ineficient, o să ajungem să urâm ce facem
 - Da, o să ne doară mâinile. **Tare.**

De ce e important să scriem cod eficient:

- când scriem repede, scriem mai mult
 - mai mult nu înseamnă mai bine
 - but we're sometimes "in the zone"
- dacă scriem mult și ineficient, o să ajungem să urâm ce facem
 - Da, o să ne doară mâinile. **Tare.**
- e un exercițiu al minții să edităm eficient

1 Introducere

2 Editare

3 Opțiuni Vim

4 Editarea mai multor fișiere

5 ...and beyond

De ce Vim?

- ~~un editor text-mode este mult mai eficient decât un IDE grafic~~

De ce Vim?

- un editor text-mode este mult mai eficient decât un IDE grafic
- vim face treaba mai bine decât emacs

De ce Vim?

- un editor text-mode este mult mai eficient decât un IDE grafic
- vim face treaba mai bine decât emacs
- contează cu ce ne obișnuim

De ce Vim?

- un editor text-mode este mult mai eficient decât un IDE grafic
- vim face treaba mai bine decât emacs
- contează cu ce ne obișnuim
 - dar ar trebui să ne obișnuim cu ceva flexibil și portabil

vi

- Bill Joy, 1976
- parte a standardului POSIX

vi

- Bill Joy, 1976
- parte a standardului POSIX

Vim

- clonă de vi (Vi iMproved)
- Bram Moolenaar, 1991
- free and open source
- portabil
- interfață grafică
- multe alte facilități

Cei doi termeni (vi și Vim) vor fi folosiți ca sinonime în prezentare, deși nu sunt.

Câteva moduri importante:

- Normal – implicit, așa pornește vi
- deplasare rapidă

Câteva moduri importante:

- Normal – implicit, așa pornește vi
 - deplasare rapidă
- Insert – introducere de text
 - din Normal, apăsăm i
 - ne întoarcem în Normal cu Esc
 - nu rămânem în Insert mai mult decât e necesar

Câteva moduri importante:

- Normal – implicit, așa pornește vi
 - deplasare rapidă
- Insert – introducere de text
 - din Normal, apăsăm i
 - ne întoarcem în Normal cu Esc
 - nu rămânem în Insert mai mult decât e necesar
- Command – introducere de comenzi “avansate”
 - apăsăm : din Normal și ne fuge cursorul pe ultima linie
 - ne întoarcem lansând comanda (Enter) sau cu Esc

Câteva moduri importante:

- Normal – implicit, așa pornește vi
 - deplasare rapidă
- Insert – introducere de text
 - din Normal, apăsăm i
 - ne întoarcem în Normal cu Esc
 - nu rămânem în Insert mai mult decât e necesar
- Command – introducere de comenzi “avansate”
 - apăsăm : din Normal și ne fuge cursorul pe ultima linie
 - ne întoarcem lansând comanda (Enter) sau cu Esc
- Visual – “selectăm” bucăți de text
 - din Normal, apăsând v
 - ne întoarcem cu Esc sau lansând o comandă de editare asupra textului

1 Introducere

2 Editare

3 Opțiuni Vim

4 Editarea mai multor fișiere

5 ...and beyond

Cum edităm un fișier

- Lansăm Vim în execuție cu `vim filename`

Cum edităm un fișier

- Lansăm Vim în execuție cu `vim filename`
- Pentru a salva schimbările, `:w`

Cum edităm un fișier

- Lansăm Vim în execuție cu `vim filename`
- Pentru a salva schimbările, `:w`
 - “save as” – `:w new-filename`

Cum edităm un fișier

- Lansăm Vim în execuție cu `vim filename`
- Pentru a salva schimbările, `:w`
 - “save as” – `:w new-filename`
- Ieșim din Vim cu `:q`

Cum edităm un fișier

- Lansăm Vim în execuție cu `vim filename`
- Pentru a salva schimbările, `:w`
 - “save as” – `:w new-filename`
- Ieșim din Vim cu `:q`
 - ne “răstim” pentru a nu salva modificările: `:q!`

Cum edităm un fișier

- Lansăm Vim în execuție cu `vim filename`
- Pentru a salva schimbările, `:w`
 - “save as” – `:w new-filename`
- Leșim din Vim cu `:q`
 - ne “răstim” pentru a nu salva modificările: `:q!`
 - salvăm și închidem dintr-o singură comandă: `:wq`

Cum edităm un fișier

- Lansăm Vim în execuție cu `vim filename`
- Pentru a salva schimbările, `:w`
 - “save as” – `:w new-filename`
- Ieșim din Vim cu `:q`
 - ne “răstim” pentru a nu salva modificările: `:q!`
 - salvăm și închidem dintr-o singură comandă: `:wq`

Exercițiu: creați un fișier cu numele `test-01` și scrieți numele cursului la care participați în el. Exersați salvând în alt fișier și renunțând la unele modificări mai puțin utile la ieșirea din Vim.

- Ne deplasăm cu h j k l (*în modul Normal*)

Basic movement and first aid

- Ne deplasăm cu `hjkl` (*în modul Normal*)
 - `h` și `l` – stânga și dreapta

Basic movement and first aid

- Ne deplasăm cu `hjk1` (*în modul Normal*)
 - `h` și `l` – stânga și dreapta
 - `j` și `k` – sus și jos

Basic movement and first aid

- Ne deplasăm cu `hjk1` (*în modul Normal*)
 - `h` și `l` – stânga și dreapta
 - `j` și `k` – sus și jos
 - mai rapid – suntem pe “home row”

Basic movement and first aid

- Ne deplasăm cu `hjk1` (*în modul Normal*)
 - `h` și `l` – stânga și dreapta
 - `j` și `k` – sus și jos
 - mai rapid – suntem pe “home row”
 - `j` are un semn pe unele tastaturi

Basic movement and first aid

- Ne deplasăm cu `hjkl` (*în modul Normal*)
 - `h` și `l` – stânga și dreapta
 - `j` și `k` – sus și jos
 - mai rapid – suntem pe “home row”
 - `j` are un semn pe unele tastaturi
- `:help` – de departe cea mai utilă comandă Vim

Basic movement and first aid

- Ne deplasăm cu `hjkl` (*în modul Normal*)
 - `h` și `l` – stânga și dreapta
 - `j` și `k` – sus și jos
 - mai rapid – suntem pe “home row”
 - `j` are un semn pe unele tastaturi
- `:help` – de departe cea mai utilă comandă Vim
 - folosiți `:q` pentru a închide fereastra pe care o deschide `:h`

More advanced movement

Cum suntem eficienți în modul Normal?

More advanced movement

Cum suntem eficienți în modul Normal?

- ^ și \$ – home și end

More advanced movement

Cum suntem eficienți în modul Normal?

- ^ și \$ – home și end
- w și b – sărim peste un cuvânt la dreapta / la stânga (*word / back*)

More advanced movement

Cum suntem eficienți în modul Normal?

- ^ și \$ – home și end
- w și b – sărim peste un cuvânt la dreapta / la stânga (*word / back*)
- gg și G – prima / ultima linie din fișier

More advanced movement

Cum suntem eficienți în modul Normal?

- ^ și \$ – home și end
- w și b – sărim peste un cuvânt la dreapta / la stânga (*word / back*)
- gg și G – prima / ultima linie din fișier
 - :42 sau 42G ne duce la linia 42

More advanced movement

Cum suntem eficienți în modul Normal?

- ^ și \$ – home și end
- w și b – sărim peste un cuvânt la dreapta / la stânga (*word / back*)
- gg și G – prima / ultima linie din fișier
 - :42 sau 42G ne duce la linia 42
- { și } – paragraful precedent / paragraful următor

More advanced movement

Cum suntem eficienți în modul Normal?

- ^ și \$ – home și end
- w și b – sărim peste un cuvânt la dreapta / la stânga (*word / back*)
- gg și G – prima / ultima linie din fișier
 - :42 sau 42G ne duce la linia 42
- { și } – paragraful precedent / paragraful următor
- fx și Fx – ne poziționează pe prima apariție a caracterului x de pe linia curentă, căutând spre dreapta / stânga

More advanced movement

Cum suntem eficienți în modul Normal?

- ^ și \$ – home și end
- w și b – sărim peste un cuvânt la dreapta / la stânga (*word / back*)
- gg și G – prima / ultima linie din fișier
 - :42 sau 42G ne duce la linia 42
- { și } – paragraful precedent / paragraful următor
- fx și Fx – ne poziționează pe prima apariție a caracterului x de pe linia curentă, căutând spre dreapta / stânga
 - `int main(char argc, char **argv)`

More advanced movement

Cum suntem eficienți în modul Normal?

- ^ și \$ – home și end
- w și b – sărim peste un cuvânt la dreapta / la stânga (*word / back*)
- gg și G – prima / ultima linie din fișier
 - :42 sau 42G ne duce la linia 42
- { și } – paragraful precedent / paragraful următor
- fx și Fx – ne poziționează pe prima apariție a caracterului x de pe linia curentă, căutând spre dreapta / stânga
 - `int main(char argc, char **argv)`
 - fc, apoi cw pentru a schimba char în int

More advanced movement

Cum suntem eficienți în modul Normal?

- ^ și \$ – home și end
- w și b – sărim peste un cuvânt la dreapta / la stânga (*word / back*)
- gg și G – prima / ultima linie din fișier
 - :42 sau 42G ne duce la linia 42
- { și } – paragraful precedent / paragraful următor
- fx și Fx – ne poziționează pe prima apariție a caracterului x de pe linia curentă, căutând spre dreapta / stânga
 - `int main(char argc, char **argv)`
 - fc, apoi cw pentru a schimba char în int
- % mă duce la perechea parantezei pe care mă aflu

More advanced editing (1)

Ștergere:

- x șterge caracterul de sub cursor

More advanced editing (1)

Ștergere:

- x șterge caracterul de sub cursor
- d șterge caracterele date de mișcarea care îl urmează

More advanced editing (1)

Ștergere:

- x șterge caracterul de sub cursor
- d șterge caracterele date de mișcarea care îl urmează
 - d1 este echivalent cu x

More advanced editing (1)

Ștergere:

- x șterge caracterul de sub cursor
- d șterge caracterele date de mișcarea care îl urmează
 - dl este echivalent cu x
 - dj șterge linia curentă și pe cea de sub ea

More advanced editing (1)

Ștergere:

- x șterge caracterul de sub cursor
- d șterge caracterele date de mișcarea care îl urmează
 - dl este echivalent cu x
 - dj șterge linia curentă și pe cea de sub ea
 - d% șterge o pereche de paranteze

More advanced editing (1)

Ștergere:

- x șterge caracterul de sub cursor
- d șterge caracterele date de mișcarea care îl urmează
 - dl este echivalent cu x
 - dj șterge linia curentă și pe cea de sub ea
 - d% șterge o pereche de paranteze
 - dd șterge o linie (shortcut)

More advanced editing (1)

Ștergere:

- x șterge caracterul de sub cursor
- d șterge caracterele date de mișcarea care îl urmează
 - dl este echivalent cu x
 - dj șterge linia curentă și pe cea de sub ea
 - d% șterge o pereche de paranteze
 - dd șterge o linie (shortcut)

Copy and Paste:

More advanced editing (1)

Ștergere:

- x șterge caracterul de sub cursor
- d șterge caracterele date de mișcarea care îl urmează
 - dl este echivalent cu x
 - dj șterge linia curentă și pe cea de sub ea
 - d% șterge o pereche de paranteze
 - dd șterge o linie (shortcut)

Copy and Paste:

- y (yank) copiază caracterele date de mișcarea care îl urmează

More advanced editing (1)

Ștergere:

- `x` șterge caracterul de sub cursor
- `d` șterge caracterele date de mișcarea care îl urmează
 - `d1` este echivalent cu `x`
 - `dj` șterge linia curentă și pe cea de sub ea
 - `d%` șterge o pereche de paranteze
 - `dd` șterge o linie (shortcut)

Copy and Paste:

- `y` (yank) copiază caracterele date de mișcarea care îl urmează
- `p` (put) este echivalentul lui *paste*

More advanced editing (1)

Ștergere:

- x șterge caracterul de sub cursor
- d șterge caracterele date de mișcarea care îl urmează
 - dl este echivalent cu x
 - dj șterge linia curentă și pe cea de sub ea
 - d% șterge o pereche de paranteze
 - dd șterge o linie (shortcut)

Copy and Paste:

- y (yank) copiază caracterele date de mișcarea care îl urmează
- p (put) este echivalentul lui *paste*
 - P puts *before* the character under cursor

More advanced editing (1)

Ștergere:

- x șterge caracterul de sub cursor
- d șterge caracterele date de mișcarea care îl urmează
 - dl este echivalent cu x
 - dj șterge linia curentă și pe cea de sub ea
 - d% șterge o pereche de paranteze
 - dd șterge o linie (shortcut)

Copy and Paste:

- y (yank) copiază caracterele date de mișcarea care îl urmează
- p (put) este echivalentul lui *paste*
 - P puts *before* the character under cursor
- **Atenție:** d pune ce șterge în “clipboard”

More advanced editing (1)

Ștergere:

- x șterge caracterul de sub cursor
- d șterge caracterele date de mișcarea care îl urmează
 - dl este echivalent cu x
 - dj șterge linia curentă și pe cea de sub ea
 - d% șterge o pereche de paranteze
 - dd șterge o linie (shortcut)

Copy and Paste:

- y (yank) copiază caracterele date de mișcarea care îl urmează
- p (put) este echivalentul lui *paste*
 - P puts *before* the character under cursor
- **Atenție:** d pune ce șterge în “clipboard”
 - dacă ștergem o linie goală înainte de a face p, tocmai am pierdut conținutul “clipboard-ului”

More advanced editing (1)

Ștergere:

- x șterge caracterul de sub cursor
- d șterge caracterele date de mișcarea care îl urmează
 - dl este echivalent cu x
 - dj șterge linia curentă și pe cea de sub ea
 - d% șterge o pereche de paranteze
 - dd șterge o linie (shortcut)

Copy and Paste:

- y (yank) copiază caracterele date de mișcarea care îl urmează
- p (put) este echivalentul lui *paste*
 - P puts *before* the character under cursor
- **Atenție:** d pune ce șterge în “clipboard”
 - dacă ștergem o linie goală înainte de a face p, tocmai am pierdut conținutul “clipboard-ului”
 - **Soluția:** mai multe registre: "ry, "rp

More advanced editing (2)

- `rx` – înlocuiește caracterul de sub cursor cu `x`

More advanced editing (2)

- `rx` – înlocuiește caracterul de sub cursor cu `x`
- `c` (change) – șterge caracterele de sub mișcarea următoare și intră în modul Insert pentru a introduce text

More advanced editing (2)

- `rx` – înlocuiește caracterul de sub cursor cu `x`
- `c` (change) – șterge caracterele de sub mișcarea următoare și intră în modul Insert pentru a introduce text
 - `C` – echivalent cu `c$`

More advanced editing (2)

- `rx` – înlocuiește caracterul de sub cursor cu `x`
- `c` (change) – șterge caracterele de sub mișcarea următoare și intră în modul Insert pentru a introduce text
 - `C` – echivalent cu `c$`
 - textul șters se duce în registrul implicit

Modul Visual:

- putem selecta o regiune complexă pentru a opera asupra ei

Modul Visual:

- putem selecta o regiune complexă pentru a opera asupra ei
- v pentru a intra în modul Visual

Modul Visual:

- putem selecta o regiune complexă pentru a opera asupra ei
- v pentru a intra în modul Visual
- ne mișcăm pentru a selecta text

Modul Visual:

- putem selecta o regiune complexă pentru a opera asupra ei
- v pentru a intra în modul Visual
- ne mișcăm pentru a selecta text
- d, y, etc. pentru a opera asupra lui și a ieși din modul Visual

Modul Visual:

- putem selecta o regiune complexă pentru a opera asupra ei
- v pentru a intra în modul Visual
- ne mișcăm pentru a selecta text
- d, y, etc. pentru a opera asupra lui și a ieși din modul Visual

Variațiuni pe aceeași temă:

Modul Visual:

- putem selecta o regiune complexă pentru a opera asupra ei
- v pentru a intra în modul Visual
- ne mișcăm pentru a selecta text
- d, y, etc. pentru a opera asupra lui și a ieși din modul Visual

Variațiuni pe aceeași temă:

- V – Visual Line mode; selectăm numai linii întregi

Modul Visual:

- putem selecta o regiune complexă pentru a opera asupra ei
- v pentru a intra în modul Visual
- ne mișcăm pentru a selecta text
- d, y, etc. pentru a opera asupra lui și a ieși din modul Visual

Variațiuni pe aceeași temă:

- V – Visual Line mode; selectăm numai linii întregi
- Ctrl-V – Visual Block mode; selectăm “un dreptunghi” de text

- u și Ctrl-R – undo și redo

Loose ends

- u și Ctrl-R – undo și redo
- /text caută text
 - n și N – next și previous

Loose ends

- u și Ctrl-R – undo și redo
- /text caută text
 - n și N – next și previous
- :s/old_pattern/new_pattern/[options] – substituie old_pattern cu new_pattern

Loose ends

- u și Ctrl-R – undo și redo
- /text caută text
 - n și N – next și previous
- :s/old_pattern/new_pattern/[options] – substituie old_pattern cu new_pattern
 - operează implicit asupra liniei curente

Loose ends

- u și Ctrl-R – undo și redo
- /text caută text
 - n și N – next și previous
- :s/old_pattern/new_pattern/[options] – substituie old_pattern cu new_pattern
 - operează implicit asupra liniei curente
 - folosim Visual pentru a selecta ușor mai multe linii

Loose ends

- u și Ctrl-R – undo și redo
- /text caută text
 - n și N – next și previous
- :s/old_pattern/new_pattern/[options] – substituie old_pattern cu new_pattern
 - operează implicit asupra liniei curente
 - folosim Visual pentru a selecta ușor mai multe linii
 - sau folosim % între : și s pentru a selecta întregul fișier

- u și Ctrl-R – undo și redo
- /text caută text
 - n și N – next și previous
- :s/old_pattern/new_pattern/[options] – substituie old_pattern cu new_pattern
 - operează implicit asupra liniei curente
 - folosim Visual pentru a selecta ușor mai multe linii
 - sau folosim % între : și s pentru a selecta întregul fișier
 - opțiunea g înlocuiește toate aparițiile de pe o linie

- u și Ctrl-R – undo și redo
- /text caută text
 - n și N – next și previous
- :s/old_pattern/new_pattern/[options] – substituie old_pattern cu new_pattern
 - operează implicit asupra liniei curente
 - folosim Visual pentru a selecta ușor mai multe linii
 - sau folosim % între : și s pentru a selecta întregul fișier
 - opțiunea g înlocuiește toate aparițiile de pe o linie
 - :%s/old/new/g ar înlocui toate instanțele lui old cu new

- descărcați și dezarhivați arhiva `http://rosedu2.cs.pub.ro/~ddvlad/vim_example.tar.bz2`
- trebuie să faceți fișierul `myfile` să fie identic cu `myfile.original`, folosind unele dintre comenzile de editare prezentate
- **nu rămâneți în Insert mai mult decât este necesar**
- pentru a vedea cele două fișiere unul lângă altul, deschideți `myfile.original`, apoi scrieți `:vsp myfile`
- vă deplasați între cele două jumătăți cu `Ctrl-W Ctrl-W` (da, de două ori)
- **maximizați terminalul înainte de a începe**

1 Introducere

2 Editare

3 Opțiuni Vim

4 Editarea mai multor fișiere

5 ...and beyond

Ce sunt opțiunile

- comportamentul Vim poate fi modificat de opțiuni

Ce sunt opțiunile

- comportamentul Vim poate fi modificat de opțiuni
- folosim `:set` pentru a modifica sau interoga opțiunile

Ce sunt opțiunile

- comportamentul Vim poate fi modificat de opțiuni
- folosim `:set` pentru a modifica sau interoga opțiunile
- modificăm folosind `:set optiune=valoare`

Ce sunt opțiunile

- comportamentul Vim poate fi modificat de opțiuni
- folosim `:set` pentru a modifica sau interoga opțiunile
- modificăm folosind `:set optiune=valoare`
 - pentru help despre o opțiune, `:h 'opțiune'` (observați apostroafele)

Ce sunt opțiunile

- comportamentul Vim poate fi modificat de opțiuni
- folosim `:set` pentru a modifica sau interoga opțiunile
- modificăm folosind `:set optiune=valoare`
 - pentru help despre o opțiune, `:h 'opțiune'` (observați apostroafele)
- aflăm valoarea unei opțiuni cu `:set optiune?`

Ce sunt opțiunile

- comportamentul Vim poate fi modificat de opțiuni
- folosim `:set` pentru a modifica sau interoga opțiunile
- modificăm folosind `:set optiune=valoare`
 - pentru help despre o opțiune, `:h 'opțiune'` (observați apostroafele)
- aflăm valoarea unei opțiuni cu `:set optiune?`
- unele opțiuni sunt de tip boolean; convenția este `:set opt` și `:set noopt`

Ce sunt opțiunile

- comportamentul Vim poate fi modificat de opțiuni
- folosim `:set` pentru a modifica sau interoga opțiunile
- modificăm folosind `:set optiune=valoare`
 - pentru help despre o opțiune, `:h 'opțiune'` (observați apostroafele)
- aflăm valoarea unei opțiuni cu `:set optiune?`
- unele opțiuni sunt de tip boolean; convenția este `:set opt` și `:set noopt`
- foarte multe opțiuni au și forme prescurtate

Exemple de opțiuni

- `textwidth` (tw) – la dimensiunea liniei curente de mai mult de tw caractere, linia este spartă; $tw=0$ înseamnă că nu există limită;

Exemple de opțiuni

- `textwidth` (`tw`) – la dimensiunea liniei curente de mai mult de `tw` caractere, linia este spartă; `tw=0` înseamnă că nu există limită;
- `filetype` (`ft`) – tipul fișierului curent, așa cum îl “vede” Vim; poate fi sursă (C, bash, etc.), Makefile, fișier de configurare, sau chiar mesaj e-mail;

Exemple de opțiuni

- `textwidth` (`tw`) – la dimensiunea liniei curente de mai mult de `tw` caractere, linia este spartă; `tw=0` înseamnă că nu există limită;
- `filetype` (`ft`) – tipul fișierului curent, așa cum îl “vede” Vim; poate fi sursă (C, bash, etc.), Makefile, fișier de configurare, sau chiar mesaj e-mail;
- `softtabstop` (`sts`) și `shiftwidth` (`sw`) – controlează cu câte spații se indentează codul când apăsăm pe Tab;

Exemple de opțiuni

- `textwidth` (`tw`) – la dimensiunea liniei curente de mai mult de `tw` caractere, linia este spartă; `tw=0` înseamnă că nu există limită;
- `filetype` (`ft`) – tipul fișierului curent, așa cum îl “vede” Vim; poate fi sursă (C, bash, etc.), Makefile, fișier de configurare, sau chiar mesaj e-mail;
- `softtabstop` (`sts`) și `shiftwidth` (`sw`) – controlează cu câte spații se indentează codul când apăsăm pe Tab;
- `autoindent` (`ai`) și `cindent` (`cin`) – opțiuni care controlează modul în care textul este (sau nu) indentat automat; încercați `cin` pentru cod C;

Exemple de opțiuni

- `textwidth` (`tw`) – la dimensiunea liniei curente de mai mult de `tw` caractere, linia este spartă; `tw=0` înseamnă că nu există limită;
- `filetype` (`ft`) – tipul fișierului curent, așa cum îl “vede” Vim; poate fi sursă (C, bash, etc.), Makefile, fișier de configurare, sau chiar mesaj e-mail;
- `softtabstop` (`sts`) și `shiftwidth` (`sw`) – controlează cu câte spații se indentează codul când apăsăm pe Tab;
- `autoindent` (`ai`) și `cindent` (`cin`) – opțiuni care controlează modul în care textul este (sau nu) indentat automat; încercați `cin` pentru cod C;
- `wildmenu` – face tab-completion în Command să funcționeze asemănător cu bash.

Putem automatiza și personaliza Vim folosind fișierul vimrc.

Putem automatiza și personaliza Vim folosind fișierul vimrc.

- vimrc local este în `$HOME/.vimrc`

Putem automatiza și personaliza Vim folosind fișierul vimrc.

- vimrc local este în `$HOME/.vimrc`
- conține comenzi care sunt executate de Vim la începutul sesiunii
 - fără prefixul :

Putem automatiza și personaliza Vim folosind fișierul vimrc.

- vimrc local este în `$HOME/.vimrc`
- conține comenzi care sunt executate de Vim la începutul sesiunii
 - fără prefixul :
- putem pune comenzi set în vimrc

- 1 Introducere
- 2 Editare
- 3 Opțiuni Vim
- 4 Editarea mai multor fișiere**
- 5 ...and beyond

- Vim poate edita mai multe fișiere odată

Mai multe buffere

- Vim poate edita mai multe fișiere odată
- vedem ușor: `vim file1 file2 ... fileN`

Mai multe buffere

- Vim poate edita mai multe fişiere odată
- vedem ușor: `vim file1 file2 ... fileN`
- `:next` și `:Next` – următorul fişier / fişierul precedent

Mai multe tab-uri

- Hello, Firefox!

Mai multe tab-uri

- Hello, Firefox!
- tab-uri – indiciu vizual permanent cu privire la fişierele deschise

Mai multe tab-uri

- Hello, Firefox!
- tab-uri – indiciu vizual permanent cu privire la fişierele deschise
- deschidem un tab nou cu `:tabnew filename`

Mai multe tab-uri

- Hello, Firefox!
- tab-uri – indiciu vizual permanent cu privire la fişierele deschise
- deschidem un tab nou cu `:tabnew filename`
- `gt` și `gT` ciclează prin taburi

Mai multe tab-uri

- Hello, Firefox!
- tab-uri – indiciu vizual permanent cu privire la fişierele deschise
- deschidem un tab nou cu `:tabnew filename`
- `gt` și `gT` ciclează prin taburi
- `ngt` ne duce la al n-lea tab

Mai multe tab-uri

- Hello, Firefox!
- tab-uri – indiciu vizual permanent cu privire la fişierele deschise
- deschidem un tab nou cu `:tabnew filename`
- `gt` și `gT` ciclează prin taburi
- `ngt` ne duce la al n-lea tab
- închidem un tab cu `:q`

Mai multe tab-uri

- Hello, Firefox!
- tab-uri – indiciu vizual permanent cu privire la fişierele deschise
- deschidem un tab nou cu `:tabnew filename`
- `gt` și `gT` ciclează prin taburi
- `ngt` ne duce la al n-lea tab
- închidem un tab cu `:q`
- `:qa`, `:qa!`, `:wqa` – operează asupra tuturor tab-urilor

- 1 Introducere
- 2 Editare
- 3 Opțiuni Vim
- 4 Editarea mai multor fișiere
- 5 ...and beyond**

Make și ctags

Make:

- putem rula `make` direct din Vim – `:make`

Make și ctags

Make:

- putem rula `make` direct din Vim – `:make`
- în cazul eventualelor erori, folosim `:cnext` pentru a merge la linia care conține eroarea următoare

Make și ctags

Make:

- putem rula `make` direct din Vim – `:make`
- în cazul eventualelor erori, folosim `:cnext` pentru a merge la linia care conține eroarea următoare

ctags:

Make și ctags

Make:

- putem rula `make` direct din Vim – `:make`
- în cazul eventualelor erori, folosim `:cnext` pentru a merge la linia care conține eroarea următoare

ctags:

- fișier care conține informații despre definirea funcțiilor C

Make și ctags

Make:

- putem rula `make` direct din Vim – `:make`
- în cazul eventualelor erori, folosim `:cnext` pentru a merge la linia care conține eroarea următoare

ctags:

- fișier care conține informații despre definirea funcțiilor C
- `ctags -aR *` pentru a genera fișierul

Make și ctags

Make:

- putem rula `make` direct din Vim – `:make`
- în cazul eventualelor erori, folosim `:cnext` pentru a merge la linia care conține eroarea următoare

ctags:

- fișier care conține informații despre definirea funcțiilor C
- `ctags -aR *` pentru a genera fișierul
- opțiunea `tags` din Vim ține numele fișierului în care căutăm definițiile

Make și ctags

Make:

- putem rula `make` direct din Vim – `:make`
- în cazul eventualelor erori, folosim `:cnext` pentru a merge la linia care conține eroarea următoare

ctags:

- fișier care conține informații despre definirea funcțiilor C
- `ctags -aR *` pentru a genera fișierul
- opțiunea `tags` din Vim ține numele fișierului în care căutăm definițiile
- `:tag tagname, Ctrl-], Ctrl-T`

Ce problemă rezolvă un modeline?

- vreau să îți dau un fișier

Ce problemă rezolvă un modeline?

- vreau să îți dau un fișier
 - dar vreau să îl vezi așa cum vreau eu

Ce problemă rezolvă un modeline?

- vreau să îți dau un fișier
 - dar vreau să îl vezi așa cum vreau eu
 - și știi că folosești vim :-)

Ce problemă rezolvă un modeline?

- vreau să îți dau un fișier
 - dar vreau să îl vezi așa cum vreau eu
 - și știu că folosești vim :-)
- un modeline conține comenzi pe care vim trebuie să le execute când citește fișierul

Ce problemă rezolvă un modeline?

- vreau să îți dau un fișier
 - dar vreau să îl vezi așa cum vreau eu
 - și știi că folosești vim :-)
- un modeline conține comenzi pe care vim trebuie să le execute când citește fișierul
- `/* vim: set tw=72 sts=4 sw=4 cindent: */`

Ce problemă rezolvă un modeline?

- vreau să îți dau un fișier
 - dar vreau să îl vezi așa cum vreau eu
 - și știi că folosești vim :-)
- un modeline conține comenzi pe care vim trebuie să le execute când citește fișierul
- `/* vim: set tw=72 sts=4 sw=4 cindent: */`
- utilizatorul trebuie să aibă modeline setat

Ce problemă rezolvă un modeline?

- vreau să îți dau un fișier
 - dar vreau să îl vezi așa cum vreau eu
 - și știi că folosești vim :-)
- un modeline conține comenzi pe care vim trebuie să le execute când citește fișierul
- `/* vim: set tw=72 sts=4 sw=4 cindent: */`
- utilizatorul trebuie să aibă modeline setat
 - `set modeline` e un candidat bun pentru orice vimrc

More awesome goodness

- rulăm comenzi de shell cu `:!comanda`

More awesome goodness

- rulăm comenzi de shell cu `:!comanda`
- autocompletion

More awesome goodness

- rulăm comenzi de shell cu `:!comanda`
- autocompletion
- registrul `+` este clipboard-ul sistemului: `"+y` și `"+p`

Random Goodness for your vimrc

- `imap jj <Esc>`

Random Goodness for your vimrc

- `imap jj <Esc>`
- `autocmd FileType c set tw=72 cindent`

Întrebări