

Osa 1: Esittely ja perusteet

Editorit

Editorien evoluutio

Vim

Vi:n perusasiat

Tekstinkäsittely

Editorit

Editorien evoluutio

Vim

Vi:n perusasiat

Tekstinkäsittely

Editori, mitä ne ovat?

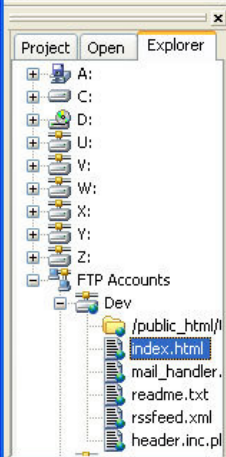
- ▶ Editori on työvälineohjelma, jolla muokataan ja kirjoitetaan puhdasta tekstiä.

Editori, mitä ne ovat?

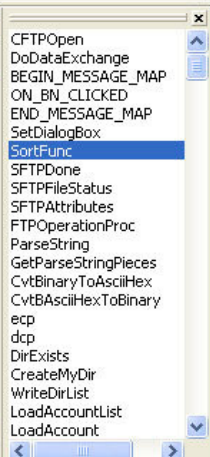
- ▶ Editori on työvälineohjelma, jolla muokataan ja kirjoitetaan puhdasta tekstiä.
- ▶ Esimerkiksi Windowsin *Muistio*, maksullinen *UltraEdit* ja DOSin *Edit*.
- ▶ *Microsoft Word* ei ole editori, vaan tekstinkäsittelyohjelma.



randomizer.js abs_space.java cftopen.cpp config.cgi datasec.c formmail.pl index.html



```
233
234 typedef struct threadStruct {
243 // Sort callback function
244 static int CALLBACK SortFunc(LPARAM lParam1, LPARAM
245
246 CFTPOpen* pDialog = (CFTPOpen*)lParamSort;
247 CListCtrl* pListCtrl = &pDialog->m_lFtpList;
248 LV_FINDINFO findInfo;
249 char szBuf1[MAX_PATH];
250 char szBuf2[MAX_PATH];
251
252 LV_ITEM lvstruct1, lvstruct2;
253
254 // initial checks for first item (dir up cmd)
255 if (lParam1 == 0)
256     return -1;
257
258 if (lParam2 == 0)
259     return +1;
```



Mihin editoreja käytetään?

- ▶ Tekstimuotoinen data on pääasiallinen käyttökohde. Kuitenkin myös isommat ohjelmakokonaisuudet sisältävät pienimuotoisempia editoreja
- ▶ Käytännössä esimerkiksi *Word* koostuu editorista, asetteluohjelmasta, kielioppitarkistimesta, ...

Mihin editoreja käytetään?

- ▶ Tekstimuotoinen data on pääasiallinen käyttökohde. Kuitenkin myös isommat ohjelmakokonaisuudet sisältävät pienimuotoisempia editoreja
- ▶ Käytännössä esimerkiksi *Word* koostuu editorista, asetteluohjelmasta, kielioppitarkistimesta, ...
- ▶ Vanhaan hyvään aikaan editori oli työmiehen ainut työkalu. Sen parissa kirjoitettiin koodi, tehtiin raportit ja viimeisteltiin työpäiväkirja.
- ▶ Editorin virittelyyn ja hienosäätöön kulutettiin aikaa, koska monissa tapauksissa koko työpäivän kaikki näyttöpäätetyöskentely tehtiin yhden ohjelman sisällä, editorin.

Notepad2.txt - Notepad2

File Edit View Settings ?

22
23
24 Features
25
26 - Syntax highlighting: HTML, XML, PHP, ASP (JS, VBS), CSS,
27 JavaScript, VBScript, C/C++, C#, Resource Script, Makefiles, Java,
28 Visual Basic, Pascal, Assembly, SQL, Perl, Python, Configuration
29 Files, Apache Config Files, Batch Files, Diff Files
30 - Drag & drop text editing inside and outside Notepad2
31 - Basic regular expression search and replace
32 - Useful word, line and block editing shortcuts
33 - Rectangular selection (Alt+Mouse)
34 - Brace matching, auto indent, long line marker, zoom functions
35 - Support for Unicode, UTF-8, Unix and Mac text files
36 - Open shell links
37 - Mostly adjustable
38
39
40 New in version 2.1.19 (released April 10, 2008)
41
42 - "Line Comment" (Ctrl+Q) and "Stream Comment" (Ctrl+Shift+Q)
43 - "Title Case" (Ctrl+Alt+T) and "Sentence Case" (Ctrl+Alt+S)
44 - "Compress whitespace" (Alt+P) command reduces spaces and tabs
45 - Original Notepad LOG feature

Editorit

Editorien evoluutio

Vim

Vi:n perusasiat

Tekstinkäsittely

Editorien ajanlaskun alussa: rivieditorit

- ▶ Kun näyttötila oli kallista, ulkoverkkoja ei ollutkaan ja silti näppäimistöpainallusta seurasi sekuntien viive, oli monesti tehokkainta käyttää sellaista editoria, joka ei turhia räikeile hienouksilla

Editorien ajanlaskun alussa: rivieditorit

- ▶ Kun näyttötila oli kallista, ulkoverkkoja ei ollutkaan ja silti näppäimistöpainallusta seurasi sekuntien viive, oli monesti tehokkainta käyttää sellaista editoria, joka ei turhia räikeile hienouksilla
- ▶ ... hienouksilla, kuten esimerkiksi asiakirjan teksti. Tiedostosta luettiin pyytämällä rivejä lyhyillä kryptisillä komennoilla kuten 2p ja sinne lisättiin tekstiä vaikkapa komentojen kuten 5i tapaan.

Unix-ajan vörmyt

Kultaisella 70- ja 80-luvulla Unix-mainframeilla työskennelleillä oli valittavanaan kahdenlaista editoria, joista kumpikin edusti omanlaista suunnittelufilosofiaansa.

Vi

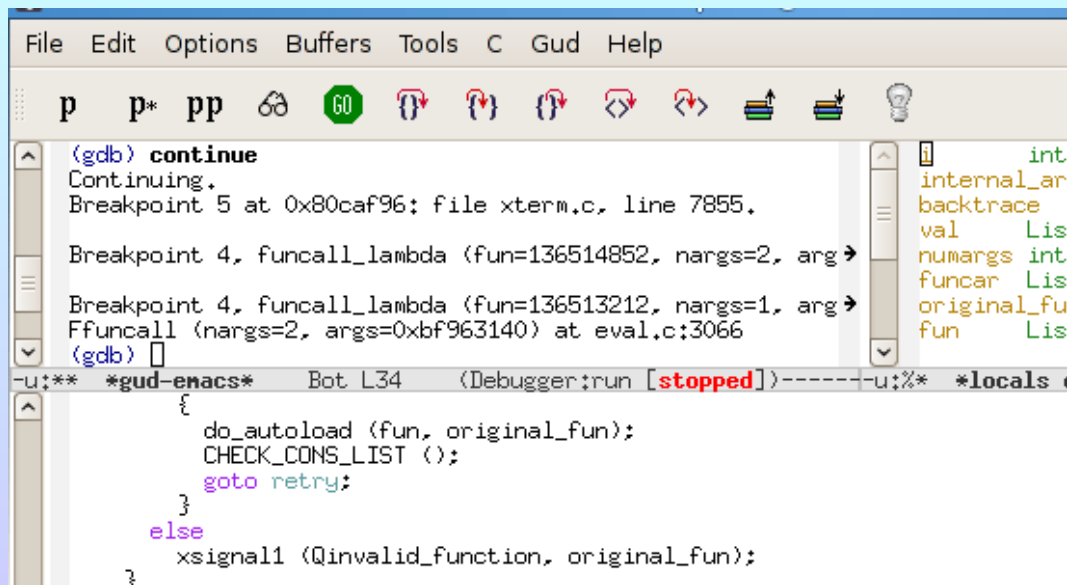
- ▶ Kevyt
- ▶ Vain tärkeimmät perustoiminnot
- ▶ Nopeille kirjailijoille

Emacs

- ▶ Raskas
- ▶ Monipuolinen
- ▶ Todella monipuolinen

Me keskitymme nyt tässä tämän editorikatsauksen toiseen osapuoleen, Vi:hin, ja erityisesti sen kehittyneeseen nykyversioon.

Palanen Emacsia



```
(gdb) continue
Continuing.
Breakpoint 5 at 0x80caf96: file xterm.c, line 7855.

Breakpoint 4, funcall_lambda (fun=136514852, nargs=2, arg=>
Breakpoint 4, funcall_lambda (fun=136513212, nargs=1, arg=>
Ffuncall (nargs=2, args=0xbf963140) at eval.c:3066
(gdb)

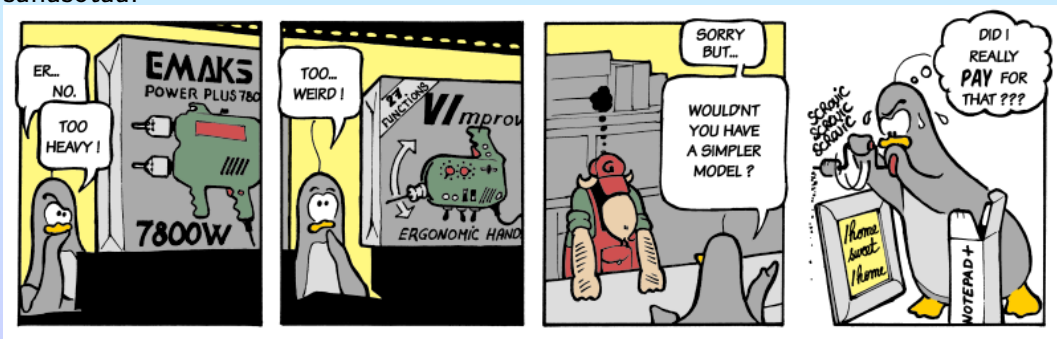
int
internal_ar
backtrace
val Lis
numargs int
funcar Lis
original_fu
fun Lis
```

```
-u:*** *gud-emacs* Bot L34 (Debugger:run [stopped])----- -u:*** *locals*
```

```
{
    do_autoload (fun, original_fun);
    CHECK_CONS_LIST ();
    goto retry;
}
else
    xsignal1 (Qinvalid_function, original_fun);
}
```

Vi ja Emacs

Nämä kaksi ovat olleet keskenään "sodassa" aina 70-luvulta lähtien. Nykypäivän graafinen sukupolvi ei enää näytä välittävän aiheesta tarpeeksi lietsotakseen sanasotaa.



Editorit

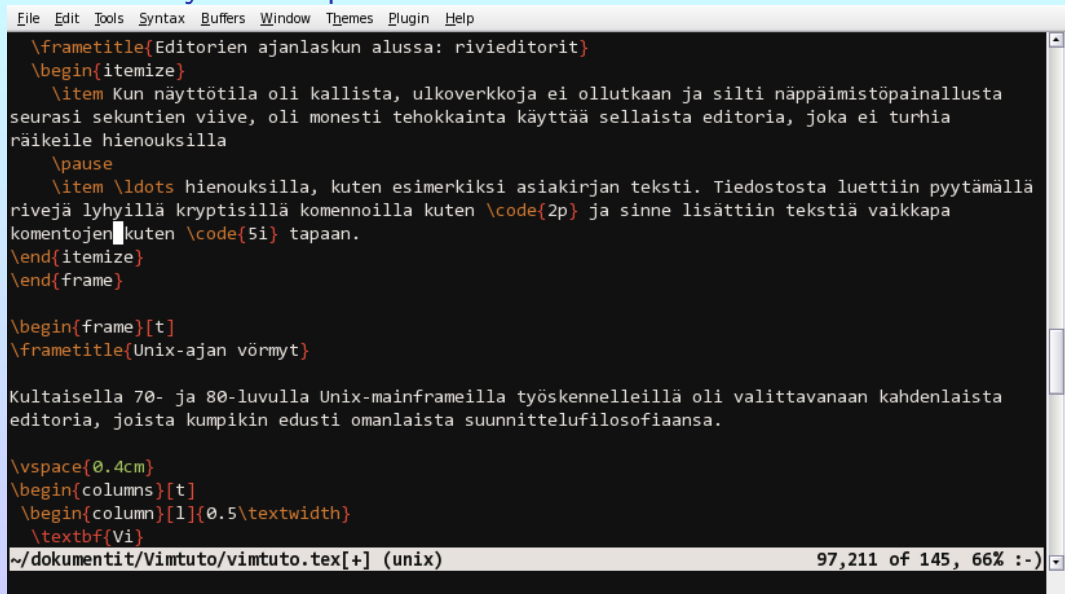
Editorien evoluutio

Vim

Vi:n perusasiat

Tekstinkäsittely

Vim tämän työn kimpussa



The screenshot shows a Vim editor window with a menu bar at the top containing 'File', 'Edit', 'Tools', 'Syntax', 'Buffers', 'Window', 'Themes', 'Plugin', and 'Help'. The main editing area has a black background with text in white and yellow. The text is a mix of LaTeX commands and rendered text. The first part is enclosed in a frame. The second part is a paragraph of text. The third part is enclosed in a frame with a table structure. The status bar at the bottom shows the file path and the current line and percentage.

```
\frametitle{Editorien ajanlaskun alussa: rivieditorit}
\begin{itemize}
  \item Kun näyttötila oli kallista, ulkoverkkoja ei ollutkaan ja silti näppäimistöpainallusta
seurasi sekuntien viive, oli monesti tehokkainta käyttää sellaista editoria, joka ei turhia
räikeile hienouksilla
  \pause
  \item \ldots hienouksilla, kuten esimerkiksi asiakirjan teksti. Tiedostosta luettiin pyytämällä
rivejä lyhyillä kryptisillä komennoilla kuten \code{2p} ja sinne lisättiin tekstiä vaikkapa
komentojen kuten \code{5i} tapaan.
\end{itemize}
\end{frame}

\begin{frame}[t]
\frametitle{Unix-ajan vörmyt}

Kultaisella 70- ja 80-luvulla Unix-mainframeilla työskennelleillä oli valittavanaan kahdenlaista
editoria, joista kumpikin edusti omanlaista suunnittelufilosofiaansa.

\vspace{0.4cm}
\begin{columns}[t]
  \begin{column}[l]{0.5\textwidth}
    \textbf{Vi}
```

~/dokumentit/Vimtuto/vimtuto.tex[+] (unix) 97,211 of 145, 66% :-)

Mikä tekee Vi:stä erikoisen?

- ▶ Lyhyesti sanottuna: sen modaalisuus
- ▶ Toimintavarmuus ja keveys olivat alkuperäisen Vi:n toiset vahvuudet
- ▶ ...ja sen jatkaja Vim lisäsi joukkoon laajennettavuuden ja monipuolisuuden

Mikä tekee Vi:stä erikoisen?

- ▶ Lyhyesti sanottuna: sen modaalisuus
- ▶ Toimintavarmuus ja keveys olivat alkuperäisen Vi:n toiset vahvuudet
- ▶ ...ja sen jatkaja Vim lisäsi joukkoon laajennettavuuden ja monipuolisuuden
- ▶ Alkuperäinen Vi muodostaa ydintoiminnallisuuden, jonka parissa nyky-Viminkin parissa työskennellään 90% ajasta, vaikka ne käsittävätkin vain 10% kaikista Vimin toiminnoista

Modaalisuus, mitä?

- ▶ Vi koostuu kahdesta pääasiallisesta toimintamoodista: normaali- ja lisäysmoodista

Modaalisuus, mitä?

- ▶ Vi koostuu kahdesta pääasiallisesta toimintamoodista: normaali- ja lisäysmoodista
 - ▶ normaalimoodissa liikutaan asiakirjassa erilaisten näppäinten avulla

Modaalisuus, mitä?

- ▶ Vi koostuu kahdesta pääasiallisesta toimintamoodista: normaali- ja lisäysmoodista
 - ▶ normaalimoodissa liikutaan asiakirjassa erilaisten näppäinten avulla
 - ▶ lisäysmoodissa lisätään uutta tekstiä haluttuun kohtaan. Puhdasoppiset vi-käyttäjät eivät tee liikettäkään normaalimoodin ulkopuolella, vaikka se olisikin mahdollista

Modaalisuus, mitä?

- ▶ Vi koostuu kahdesta pääasiallisesta toimintamoodista: normaali- ja lisäysmoodista
 - ▶ normaalimoodissa liikutaan asiakirjassa erilaisten näppäinten avulla
 - ▶ lisäysmoodissa lisätään uutta tekstiä haluttuun kohtaan. Puhdasoppiset vi-käyttäjät eivät tee liikettäkään normaalimoodin ulkopuolella, vaikka se olisikin mahdollista

Modaalisuus, mitä?

- ▶ Vi koostuu kahdesta pääasiallisesta toimintamoodista: normaali- ja lisäysmoodista
 - ▶ normaalimoodissa liikutaan asiakirjassa erilaisten näppäinten avulla
 - ▶ lisäysmoodissa lisätään uutta tekstiä haluttuun kohtaan. Puhdasoppiset vi-käyttäjät eivät tee liikettäkään normaalimoodin ulkopuolella, vaikka se olisikin mahdollista
- ▶ Koska kovin harva massojen suosima editori käyttää enää modaalisuutta, on se taiteenlaji redusoitunut lähinnä tänne vi-maailmaan.

Modaalisuus, mitä?

- ▶ Vi koostuu kahdesta pääasiallisesta toimintamoodista: normaali- ja lisäysmoodista
 - ▶ normaalimoodissa liikutaan asiakirjassa erilaisten näppäinten avulla
 - ▶ lisäysmoodissa lisätään uutta tekstiä haluttuun kohtaan. Puhdasoppiset vi-käyttäjät eivät tee liikettäkään normaalimoodin ulkopuolella, vaikka se olisikin mahdollista
- ▶ Koska kovin harva massojen suosima editori käyttää enää modaalisuutta, on se taiteenlaji redusoitunut lähinnä tänne vi-maailmaan.
- ▶ Modaalisuuden edut ovat kuitenkin selvät verrattuna de facto -standardiksi ajateltavaan *modifier*-malliin

Editorit

Editorien evoluutio

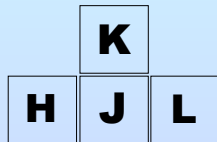
Vim

Vi:n perusasiat

Tekstinkäsittely

Tekstissä liikkuminen 1

- ▶ Jo pitkään käytetty nuolinäppäimistö on itse asiassa hyvin epäergonominen
- ▶ Vi käyttää ergonomisia näppäimiä normaalimoodissa, joilla tehdään perusliikkumista tekstissä
- ▶ Vaikka nykyisin voi käyttää nuolinäppäimiäkin liikehdintään, on hyvä edes kokeilla (ja huomata HJKL paljon mukavammaksi) vaihtoehtoista systeemiä



Käynnistys ja lopetus

- ▶ Käynnistys: Unix-piireissä komennolla `vi [tiedosto.tex]`, Windowseissa tyypillisesti klikkaillaan kuvakkeesta
- ▶ Sammutus: normaalimoodissa yleisin tapa on `:q` (lyh. quit)
- ▶ Avaus: normaalimoodissa `:e` (lyh. edit) avaa tiedoston. Nimeä on osattava tavata, tai voi käyttää tekstigraafista selausohjelmaa. Windowsissa standardi avausdialogi
- ▶ Tallennus: normaalimoodissa `:w` (lyh. write). Tallenna ja lopeta: joko `:wq` tai ytimekkäämpi `:x` tahi ZZ (kaksi kertaa iso Z-kirjain)

Editorit

Editorien evoluutio

Vim

Vi:n perusasiat

Tekstinkäsittely

Tekstin kirjoittaminen

- ▶ Editorin yleisin tehtävä; vi tarjoaa kymmeniä tapoja lisäillä tekstiä.
- ▶ Lisäysvaiheessa siirrytään normaalimoodista lisäysmoodiin: näppäimet (kuten h, j, k, l) tuottavatkin nyt kirjaimia näytölle "kuten pitäisikin"

Tekstin kirjoittaminen

- ▶ Editorin yleisin tehtävä; vi tarjoaa kymmeniä tapoja lisäillä tekstiä.
- ▶ Lisäysvaiheessa siirrytään normaalimoodista lisäysmoodiin: näppäimet (kuten h, j, k, l) tuottavatkin nyt kirjaimia näytölle "kuten pitäisikin"
- ▶ Kun valmista, palataan normaalitilaan painamalla <Esc>

Tekstin kirjoittaminen

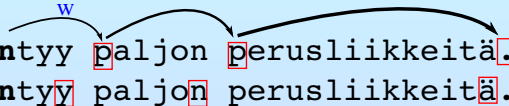
- ▶ Editorin yleisin tehtävä; vi tarjoaa kymmeniä tapoja lisäillä tekstiä.
- ▶ Lisäysvaiheessa siirrytään normaalimoodista lisäysmoodiin: näppäimet (kuten h, j, k, l) tuottavatkin nyt kirjaimia näytölle "kuten pitäisikin"
- ▶ Kun valmista, palataan normaalitilaan painamalla <Esc>
- ▶ Keskeinen lisäyskomento **i** (lyh. insert): aloittaa tekstin lisäämisen täsmälleen kursorin kohdalle
- ▶ Samanhenkinen komento on **a** (lyh. append): vastaa samaa kuin komento **li**

Tekstissä liikkuminen 2

Hyvin perustavat komennot w , e , b , 0 ja $\$$:

Lyhykäinen lause, jossa esiintyy paljon perusliikkeitä.

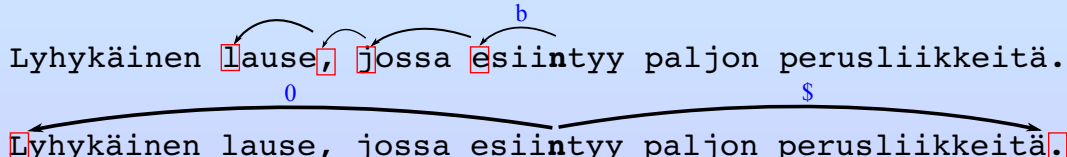
Lyhykäinen lause, jossa esiintyy paljon perusliikkeitä.



The diagram illustrates the 'w' (word) and 'e' (end of line) commands. In the first sentence, 'w' jumps from the end of 'esiintyy' to the start of 'paljon', and 'e' jumps from the end of the line to the start of the second sentence. In the second sentence, 'e' jumps from the end of 'paljon' to the start of 'perusliikkeitä'.

Lyhykäinen lause, jossa esiintyy paljon perusliikkeitä.

Lyhykäinen lause, jossa esiintyy paljon perusliikkeitä.



The diagram illustrates the 'b' (block) and '0' (zero) commands. In the first sentence, 'b' jumps from the end of 'esiintyy' to the start of 'paljon'. In the second sentence, '0' jumps from the end of 'lause,' to the start of 'Lyhykäinen', and '\$' jumps from the end of 'paljon' to the end of the line.

Tekstissä liikkuminen 3

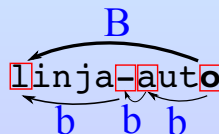
- ▶ Edellisen kalvon komennot `w`, `e` ja `b` käyttävät hyväkseen termiä "sana", joka tarkoittaa yhtä aakkosnumeerista rykelmää, jonka molemmin puolin on jokin erikoismerkki tai rivin alku/loppu

Tekstissä liikkuminen 3

- ▶ Edellisen kalvon komennot `w`, `e` ja `b` käyttävät hyväkseen termiä "sana", joka tarkoittaa yhtä aakkosnumeerista rykelmää, jonka molemmiin puolin on jokin erikoismerkki tai rivin alku/loppu
- ▶ Tämä voi olla työlästä, jos sanojen seassa on paljon erikoismerkkejä, kuten pilkkuja, kysymysmerkkejä ynnä muuta sellaista

Tekstissä liikkuminen 3

- ▶ Edellisen kalvon komennot **w**, **e** ja **b** käyttävät hyväkseen termiä "sana", joka tarkoittaa yhtä aakkosnumeerista rykelmää, jonka molemmin puolin on jokin erikoismerkki tai rivin alku/loppu
- ▶ Tämä voi olla työlästä, jos sanojen seassa on paljon erikoismerkkejä, kuten pilkkuja, kysymysmerkkejä ynnä muuta sellaista
- ▶ Ratkaisuna on erillinen käsite "*big word*" joka on suomalaisittain helppo omaksua: yhdyssana *linja-auto* koostuu kahdesta pienestä sanasta, ja muodostaa yhden ison sanan
- ▶ Helppo muistaa: korvaa pieni komento isolla kirjaimella



Tekstin lisäily 2

- ▶ Vaikka käyttämällä komentoa `i` joka paikkaan pääseekin hyvin pitkälle, on vimissä tarkoituksella hyvin monia keinoja päästä lisäälemään tekstiä uusiin paikkoihin, kaikki joustavuuden nimissä
- ▶ Komennot `I` ja `A`: lisää rivin alkuun ja loppuun tekstiä. Hyvin näppärä.

Tekstin lisäily 2

- ▶ Vaikka käyttämällä komentoa `i` joka paikkaan pääseekin hyvin pitkälle, on vimissä tarkoituksella hyvin monia keinoja päästä lisäälemään tekstiä uusiin paikkoihin, kaikki joustavuuden nimissä
- ▶ Komennot `I` ja `A`: lisää rivin alkuun ja loppuun tekstiä. Hyvin näppärä.
- ▶ Komennot `O` ja `o`: tee tyhjä rivi rivin yläpuolelle / alapuolelle ja lisää sinne

Tekstin poistaminen

- ▶ Tekstin poistaminen on joskus tarpeen. Vim tukee komentoa **d** (lyh. delete), joka ottaa kaverikseen *liikkumakomennon*. Liikkumakomennon kursorista hyppykohtaan oleva pala poistetaan
- ▶ Esimerkkejä: olkoon tekstinä Linja-autolla **m**atkustaminen on mukavaa. Ja kursori punaisessa kohdassa. Komentamalla **dw** poistettaisiin teksti matkustaminen. Komentamalla **d0** vastaavasti mennään taaksepäin, ja poistettaisiin teksti "Linja-autolla ".

Tekstin poistaminen

- ▶ Tekstin poistaminen on joskus tarpeen. Vim tukee komentoa **d** (lyh. delete), joka ottaa kaverikseen *liikkumakomennon*. Liikkumakomennon kursorista hyppykohtaan oleva pala poistetaan
- ▶ Esimerkkejä: olkoon tekstinä Linja-autolla **m**atkustaminen on mukavaa. Ja kursori punaisessa kohdassa. Komentamalla **dw** poistettaisiin teksti matkustaminen. Komentamalla **d0** vastaavasti mennään taaksepäin, ja poistettaisiin teksti "Linja-autolla".
- ▶ Koko rivin voi poistaa nopealla oikopolulla **dd**.
- ▶ Teksti palautettavissa joko perumalla **u** (lyh. undo) tai liittämällä teksti rekisteristä komennolla **p** (lyh. put)

Tekstin korvaaminen

- ▶ Tekstiä voi korvata muutamalla tavalla. Kaikki korvausmenetelmät ovat oikopolkuja sille, että ensin poistetaan sana ja siirrytään lisäysmoodiin. Kaikki menetelmät ovat hyvin käytettyjä sopivissa paikoissa
- ▶ Keskeinen komento on `c` (lyh. change), joka ottaa deleten tapaan suunnan komennon jälkeen. Esimerkiksi `c2w` poistaisi kursorin edestä kaksi sanaa, ja siirtyisi sitten lisäystilaan.
- ▶ Komentamalla nopealla oikopolulla `cc` tai vaihtoehtoisesti `S` (lyh. substitute) korvataan koko nykyinen rivi alusta alkaen

Tekstin korvaaminen

- ▶ Tekstiä voi korvata muutamalla tavalla. Kaikki korvausmenetelmät ovat oikopolkuja sille, että ensin poistetaan sana ja siirrytään lisäysmoodiin. Kaikki menetelmät ovat hyvin käytettyjä sopivissa paikoissa
- ▶ Keskeinen komento on `c` (lyh. `change`), joka ottaa deleten tapaan suunnan komennon jälkeen. Esimerkiksi `c2w` poistaisi kursorin edestä kaksi sanaa, ja siirtyisi sitten lisäystilaan.
- ▶ Komentamalla nopealla oikopolulla `cc` tai vaihtoehtoisesti `S` (lyh. `substitute`) korvataan koko nykyinen rivi alusta alkaen

Tekstin korvaaminen

- ▶ Tekstiä voi korvata muutamalla tavalla. Kaikki korvausmenetelmät ovat oikopolkuja sille, että ensin poistetaan sana ja siirrytään lisäysmoodiin. Kaikki menetelmät ovat hyvin käytettyjä sopivissa paikoissa
- ▶ Keskeinen komento on `c` (lyh. change), joka ottaa deleten tapaan suunnan komennon jälkeen. Esimerkiksi `c2w` poistaisi kursorin edestä kaksi sanaa, ja siirtyisi sitten lisäystilaan.
- ▶ Komentamalla nopealla oikopolulla `cc` tai vaihtoehtoisesti `S` (lyh. substitute) korvataan koko nykyinen rivi alusta alkaen

Pikkukomentoja ja korvauksia

- ▶ Poista yksi merkki kursorin alta: **x**

Pikkukomentoja ja korvauksia

- ▶ Poista yksi merkki kursorin alta: **x**
- ▶ Korvaa yksi merkki kursorin kohdalta: **r**, jonka jälkeen korvaava merkki:
- ▶ Esimerkiksi lause Linja-a**a**to, näppäily **ru** muuttaa sanan oikeinkirjoitetuksi

Pikkukomentoja ja korvauksia

- ▶ Poista yksi merkki kursorin alta: **x**
- ▶ Korvaa yksi merkki kursorin kohdalta: **r**, jonka jälkeen korvaava merkki:
- ▶ Esimerkiksi lause Linja-a**a**to, näppäily **ru** muuttaa sanan oikeinkirjoitetuksi
- ▶ Yksi tyylikäs tapa korvata tekstiä on **R**, joka heittää editorin lisäysmoodiin, mutta siten, että lisätty teksti korvaa aiemman. Vastaa tavallisten editorien Insert/Replace -toimintaa

Vi-komentojen anatomia

- ▶ On olemassa kahdenlaisia komentoja:

Vi-komentojen anatomia

- ▶ On olemassa kahdenlaisia komentoja:
 - ▶ Sellaisia, jotka ottavat liikkeen argumentikseen
 - ▶ Sellaisia, jotka eivät ota mitään

Vi-komentojen anatomia

- ▶ On olemassa kahdenlaisia komentoja:
 - ▶ Sellaisia, jotka ottavat liikkeen argumentikseen
 - ▶ Sellaisia, jotka eivät ota mitään
- ▶ Lisäksi komennot voivat ottaa numeroarvoja ikään kuin kertoimeksi eteensä.

Vi-komentojen anatomia

- ▶ On olemassa kahdenlaisia komentoja:
 - ▶ Sellaisia, jotka ottavat liikkeen argumentikseen
 - ▶ Sellaisia, jotka eivät ota mitään
- ▶ Lisäksi komennot voivat ottaa numeroarvoja ikään kuin kertoimeksi eteensä.
- ▶ Jos `dw` poistaa yhden sanan, simppelisti `d3w` poistaakin kolme sanaa.
Kerroinluku voi sijaita myös aivan edessä: `3dw = d3w`

Vi-komentojen anatomia

- ▶ On olemassa kahdenlaisia komentoja:
 - ▶ Sellaisia, jotka ottavat liikkeen argumentikseen
 - ▶ Sellaisia, jotka eivät ota mitään
- ▶ Lisäksi komennot voivat ottaa numeroarvoja ikään kuin kertoimeksi eteensä.
- ▶ Jos dw poistaa yhden sanan, simppelisti $d3w$ poistaakin kolme sanaa.
Kerroinluku voi sijaita myös aivan edessä: $3dw = d3w$
- ▶ Yleisesti: jos a on komento, m on liikekomento ja $n \in \mathbb{N}$ on kerroin, niin kaikki suuntia ottavat vi-komennot toimivat näin: $a \ n \ m$ tai $n \ a \ m$

Osa 2: Tekstinkäsittelyä

Pari kehittyntä liikekomentoa

Tekstin ”maalaus” – visual-moodi

Rekisterit – Vimin vastine leikepöydälle

Etsi...

...ja korvaa

Pari kehittynyttä liikekomentoa

Tekstin "maalaus" – visual-moodi

Rekisterit – Vimin vastine leikepöydälle

Etsi...

...ja korvaa

Rivillä liikkuminen

- ▶ Hyppääminen rivillä haluttuun merkkiin: **f** ja **F**, joita seuraa haluttu kirjain.
- ▶ Vim etsii annettua merkkiä vain senhetkiseltä riviltä. Löydettyään kursori siirtyy sinne

Rivillä liikkuminen

- ▶ Hyppääminen rivillä haluttuun merkkiin: `f` ja `F`, joita seuraa haluttu kirjain.
- ▶ Vim etsii annettua merkkiä vain senhetkiseltä riviltä. Löydettyään kursori siirtyy sinne
- ▶ Esimerkki: `Lause`, jossa on jotain turhaa välissä, jatkuu normaalisti.
- ▶ Komennetaan `f`, `df`,

Rivillä liikkuminen

- ▶ Hyppääminen rivillä haluttuun merkkiin: `f` ja `F`, joita seuraa haluttu kirjain.
- ▶ Vim etsii annettua merkkiä vain senhetkiseltä riviltä. Löydettyään kursori siirtyy sinne
- ▶ Esimerkki: Lause, jossa on jotain turhaa välissä, jatkuu normaalisti.
- ▶ Komennetaan `f,df,`

Rivillä liikkuminen

- ▶ Hyppääminen rivillä haluttuun merkkiin: `f` ja `F`, joita seuraa haluttu kirjain.
- ▶ Vim etsii annettua merkkiä vain senhetkiseltä riviltä. Löydettyään kursori siirtyy sinne
- ▶ Esimerkki: Lause, jossa on jotain turhaa välissä, jatkuu normaalisti.
- ▶ Komennetaan `f,df,`

Rivillä liikkuminen

- ▶ Hyppääminen rivillä haluttuun merkkiin: `f` ja `F`, joita seuraa haluttu kirjain.
- ▶ Vim etsii annettua merkkiä vain senhetkiseltä riviltä. Löydettyään kursori siirtyy sinne
- ▶ Esimerkki: Lause, jossa on jotain turhaa välissä, jatkuu normaalisti.
- ▶ Komennetaan `f,df`,
- ▶ Tuloksena: Lause jatkuu normaalisti.

Isompia liikkeitä

- ▶ Olemme oppineet perusliikkeitä, kuten liikehdinnän rivi tai merkki kerrallaan
- ▶ Myös hieman mielekkäämpiä komentoja, kuten hyppelyt sanojen välillä hallussa → vähentää kovasti tarvetta `h:n` ja `l:n` käyttöön

Isompia liikkeitä

- ▶ Olemme oppineet perusliikkeitä, kuten liikehdinnän rivi tai merkki kerrallaan
- ▶ Myös hieman mielekkäämpiä komentoja, kuten hyppelyt sanojen välillä hallussa → vähentää kovasti tarvetta `h:n` ja `l:n` käyttöön
- ▶ Siirtyminen virkkeiden välillä: `(` ja `)`. Vim tunnistaa virkkeet pisteen, huuto- tai kysymysmerkin avulla. Helppo aloittaa vaikka kesken oleva lause alusta komennolla `c(`.
- ▶ Siirtyminen kappaleiden välillä: `{` ja `}`. Tyhjä rivi, tai useampi, tekstirivien välissä toimii kappale-erottimena.

Isompia liikkeitä 2

- ▶ Ruudulla näkyvässä sisällössäkin on nopeata liikkua kolmella komennolla:
- ▶ **H** (head) – hyppää ruudun yläreunaan
- ▶ **M** (middle) – hyppää keskelle ruutua
- ▶ **L** (low) – hyppää ruudun alareunaan

Isompia liikkeitä 2

- ▶ Ruudulla näkyvässä sisällössäkin on nopeata liikkua kolmella komennolla:
- ▶ **H** (head) – hyppää ruudun yläreunaan
- ▶ **M** (middle) – hyppää keskelle ruutua
- ▶ **L** (low) – hyppää ruudun alareunaan
- ▶ Hyppääminen tiedoston alkuun: **gg**
- ▶ Hyppääminen tiedoston loppuun: **G**
- ▶ Hyppääminen halutulle riville: **:#**, missä **#** rivinumero

Yhteenvetoa keskeisistä komennoista

Liikkuma-, lisäys- ja muokkauskomendoja:

wW	seur. sanan alkuun	i	Lisää nyk. kohtaan	d	Poista ¹
bB	nyk. sanan alkuun	a	Lisää seur. ruutuun	dd	Poista rivi
eE	nyk. sanan loppuun	I	Lisää rivin alkuun	c	Korvaa teksti ¹
O	rivin alkuun	A	Lisää rivin loppuun	cc	Korvaa rivi
\$	rivin loppuun	o	Avaa uusi rivi alas	x	Poista merkki
gg	tiedoston alkuun	O	Avaa uusi rivi ylös	r	Korvaa merkki
G	tiedoston loppuun	fa	Etene merkkiin a	R	dyn. korvaus
:#	riville #	Fa	Peräänny merkkiin a	{	Edellinen kappale
(virkkeen alkuun)	virkkeen loppuun	}	seuraava kappale

¹vaatii liikekomennon

Pari kehittyynyttä liikekomentoa

Tekstin ”maalaus” – visual-moodi

Rekisterit – Vimin vastine leikepöydälle

Etsi...

...ja korvaa

Visual-moodi

- ▶ Tyypillinen vim-operaatio, vaikkapa tekstin poisto, koskettaa usein isoa tai muuten epämääräistä lohkoa tekstiä
- ▶ Tämmöistä lohkoa on usein hankala tai mahdotontakin valita järkevästi *yhdellä* liikekomennolla, vaan liikkeitä tarvitaan useampia

Visual-moodi

- ▶ Tyypillinen vim-operaatio, vaikkapa tekstin poisto, koskettaa usein isoa tai muuten epämääräistä lohkoa tekstiä
- ▶ Tämmöistä lohkoa on usein hankala tai mahdotontakin valita järkevästi *yhdellä* liikekomennolla, vaan liikkeitä tarvitaan useampia
- ▶ Ratkaisuna on visual-moodi! Moodiin pääsee painamalla **v**, ja sen jälkeen käytetäänkin tavallisia liikekomentoja aivan tavalliseen tapaan
- ▶ Teksti maalautuu komentojen mukana, joten ihmisen on helppo havaita, mitä ollaan valitsemassa. Toiminta näyttää aivan kuin graafisten ohjelmien kanssa hiirellä maalaillessa tekstiä

Visual-moodi 2

- ▶ Siinä missä tavallinen *komento-kerroin-liike* toimii ilman visual-moodia, tarvitaan nyt hieman toisenlainen lähestymistapa
- ▶ Visual-moodissa ensin valitaan alue, sitten sille jokin komento, usein pelkkä **d** tai **c**. Visual-moodin valintojen kanssa voi käyttää myös erikoisia komentoja, kuten vaikkapa **r**:ää.

Visual-moodi 2

- ▶ Siinä missä tavallinen *komento-kerroin-liike* toimii ilman visual-moodia, tarvitaan nyt hieman toisenlainen lähestymistapa
- ▶ Visual-moodissa ensin valitaan alue, sitten sille jokin komento, usein pelkkä **d** tai **c**. Visual-moodin valintojen kanssa voi käyttää myös erikoisia komentoja, kuten vaikkapa **r**:ää.
- ▶
 1. **v**: visualize
 2. liiku ja valitse haluamasi alue
 3. aseta maalatulle alueelle komento

Visual-moodi 3

- ▶ Pikku-**v**een lisäksi joissain tilanteissa näppärä visual-moodin erikoistapaus on iso **V**, riveittäinen valinta
- ▶ Käytännössä parin rivin valinta käyttäen (toivottavasti) selkäytimestä tulevilla näppäimillä **j** ja **k**, **V** on hyvin nopea verrattuna **v**:hen.

Visual-moodi 3

- ▶ Pikku-**v**een lisäksi joissain tilanteissa näppärä visual-moodin erikoistapaus on iso **V**, riveittäinen valinta
- ▶ Käytännössä parin rivin valinta käyttäen (toivottavasti) selkäytimestä tulevilla näppäimillä **j** ja **k**, **V** on hyvin nopea verrattuna **v**:hen.
- ▶ Esimerkki: muunna rivi tekstiä – kaikki kirjaimet pisteiksi:
- ▶ **Vr.** tekee homman: **V** valitsee oletuksena jo aktiivisen rivin kokonaan, se riittää. Komento **r** toimii visuaalimoodin kanssa siten, että se käy jokaisen merkin läpi, ja muuntaa sen tässä tapauksessa pisteeksi ".".

Tekstiobjektit

- ▶ Visual-moodissa voi valita tekstiä muillakin tavoilla kuin vain liikehtimällä tuttujen komentojen avulla. Tässä moodissa on määritelty joitain näppäimiä uusiksi tarjoten varsin sulavaa joustavuutta askareita varten

Tekstiobjektit

- ▶ Visual-moodissa voi valita tekstiä muillakin tavoilla kuin vain liikehtimällä tuttujen komentojen avulla. Tässä moodissa on määritelty joitain näppäimiä uusiksi tarjoten varsin sulavaa joustavuutta askareita varten
- ▶ Esimerkki: `vap`: *visualize a paragraph*, valitsee koko kappaleen, sekä sitä seuraavan tyhjän rivin. Komentamalla `vapd` poistetaan se kappale, jonka sisällä kursori sattuu olemaan.

Tekstiobjektit

- ▶ Visual-moodissa voi valita tekstiä muillakin tavoilla kuin vain liikehtimällä tuttujen komentojen avulla. Tässä moodissa on määritelty joitain näppäimiä uusiksi tarjoten varsin sulavaa joustavuutta askareita varten
- ▶ Esimerkki: `vap`: *visualize a paragraph*, valitsee koko kappaleen, sekä sitä seuraavan tyhjän rivin. Komentamalla `vapd` poistetaan se kappale, jonka sisällä kursori sattuu olemaan.
- ▶ Tekstiobjekteja voi joissain paikoissa käyttää myös ilman visual-moodia: edellinen komento sujuisi myös näin: `dap`: *delete a paragraph*

Tekstiobjektit 2

- ▶ Tekstiobjekteja on monenlaisia, mutta jokaista niistä yhdistää alkukirjain **a** (engl. a) tai **i** (engl. inner)
- ▶ ero on hiuksenhieno: **i** jättää mahdolliset välit rauhaan. Kappaleiden tapauksessa tyhjät rivit, lauseiden tapauksessa välit.

Tekstiobjektit 3

Mitä objekteja on sitten käytettävissä? Tehdään lyhyt listaus käytetyimpiin.

kirjain	a:n kanssa	i:n kanssa
x	ax	ix
w/W	sana/isosana ja sitä seuraava väli	pelkkä sana
p	kappale + tyhjä rivi	pelkkä kappale
s	virke + väli	pelkkä virke
(tai)	sulut sisältöineen	pelkkä sisältö
”, ’ tai ‘	“lainatun tekstin” valinta hipsuineen	pelkkä sisältö

Pari kehittynyttä liikekomentoa

Tekstin ”maalaus” – visual-moodi

Rekisterit – Vimin vastine leikepöydälle

Etsi...

...ja korvaa

Perusleikepöytäily

- ▶ Keskeiset komennot ovat yhtä helppoja omaksua kuin graafisissa ohjelmissakin:
- ▶ **d**: *delete*, joka toimii samalla myös leikkaustoimintona
- ▶ **y**: *yank*, kopioi poistamatta tekstiä
- ▶ **p**: *put* tai *paste*. Liittää tekstin alkaen kursoria seuraavasta ruudusta
- ▶ **P**: kuten edellä, mutta liittää tekstin alkaen juuri senhetkisestä ruudusta

Rekisterit ovat vimin leikepöytä

- ▶ Vimissä on 27 rekisteriä käyttäjän omille teksteille, joihin voi joko sijoittaa tai lisätä tekstiä tiedostosta. Lisäksi on erikoisrekistereitä, joista ehkä tuonnempana
- ▶ Rekisteriin viitataan aakkosella, ja merkitään eteen lainausmerkki " , jota seuraa rekisterin nimi

Rekisterit ovat vimin leikepöytä

- ▶ Vimissä on 27 rekisteriä käyttäjän omille teksteille, joihin voi joko sijoittaa tai lisätä tekstiä tiedostosta. Lisäksi on erikoisrekistereitä, joista ehkä tuonnempana
- ▶ Rekisteriin viitataan aakkosella, ja merkitään eteen lainausmerkki `"`, jota seuraa rekisterin nimi
- ▶ Esimerkiksi haluamme leikata sanan rekisteriin a: `"adw`
- ▶ Nyt tämän sanan voi liittää haluamaansa paikkaan komentamalla `"ap`
- ▶ Nopeata on hakea viimeisin poistettu tai kopioitu pätkä. Silloin riittää käyttää liitä-komentoa `p` ilman rekisteriä

Rekisterit ovat vimin leikepöytä 2

- ▶ Kuten jo nähtiin, rekisteriin voi sijoittaa uutta tekstiä käyttämällä "x-notaatiota
- ▶ Se kuitenkin tyhjentää entisen sisällön.

Rekisterit ovat vimin leikepöytä 2

- ▶ Kuten jo nähtiin, rekisteriin voi sijoittaa uutta tekstiä käyttämällä "x-notaatiota
- ▶ Se kuitenkin tyhjentää entisen sisällön.
- ▶ Nyt, tekstin lisääminen ennestään olevan tekstin perään rekisterissä onnistuu siten, että rekisterin nimi korvataan isolla kirjaimella: "X

Rekisterit ovat vimin leikepöytä 2

- ▶ Kuten jo nähtiin, rekisteriin voi sijoittaa uutta tekstiä käyttämällä `"x` -notaatiota
- ▶ Se kuitenkin tyhjentää entisen sisällön.
- ▶ Nyt, tekstin lisääminen ennestään olevan tekstin perään rekisterissä onnistuu siten, että rekisterin nimi korvataan isolla kirjaimella: `"X`
- ▶ Kaikki käytössä olevat rekisterit voi aina tarkastaa komentamalla `:registers`

Erikoisrekisterit

- ▶ Käyttäjän omien rekisterien lisäksi on olemassa muutama erikoisrekisteri
- ▶ Tärkein ja ehkäpä peruskurssille ainut maininnan arvoinen on nimetty rekisteri "+", joka säilyttää tietokoneen yleisessä muistissa olevan leikepöydän sisältöä

Erikoisrekisterit

- ▶ Käyttäjän omien rekisterien lisäksi on olemassa muutama erikoisrekisteri
- ▶ Tärkein ja ehkäpä peruskurssille ainut maininnan arvoinen on nimetty rekisteri "+", joka säilyttää tietokoneen yleisessä muistissa olevan leikepöydän sisältöä
- ▶ Esimerkiksi windowsissa voisi Muistiosta kopioida tekstiä leikepöydälle ja vimissä komentaa sitten "+ p", ja saada kopioitu teksti vimiin.
- ▶ Rekisteriin voi myös tallentaa, kuten odottaa saattaa. Kertauksen vuoksi: vaikkapa koko tiedoston kopioiminen Windowsin leikepöydälle: gg"+yG

Pari kehittyynyttä liikekomentoa

Tekstin ”maalaus” – visual-moodi

Rekisterit – Vimin vastine leikepöydälle

Etsi...

...ja korvaa

Tiedostosta etsiminen

- ▶ Editorin keskeisiä tehtäviä on etsiä tiedostosta jotain tekstiä
- ▶ Siinä missä Windows-maailmassa näppäinyhdistelmä **Ctrl-F** on *de-factó*, on Unix- ja Linux-piireissä komento **/** kovaa huutoa
- ▶ Käyttö on yksinkertaista: **/hakusana** hakee eteenpäin, ja komento **?hakusana** hakee taaksepäin

Tiedostosta etsiminen

- ▶ Editorin keskeisiä tehtäviä on etsiä tiedostosta jotain tekstiä
- ▶ Siinä missä Windows-maailmassa näppäinyhdistelmä **Ctrl-F** on *de-factó*, on Unix- ja Linux-piireissä komento **/** kovaa huutoa
- ▶ Käyttö on yksinkertaista: **/hakusana** hakee eteenpäin, ja komento **?hakusana** hakee taaksepäin
- ▶ Lisäksi helpottamaan hakuja, komento **n** toistaa hakua samaan suuntaan kuin mihin viimeksi on haettu
- ▶ Toisaalta **N** hakee myös samalla hakusanalla, mutta vastakkaiseen suuntaan

Etsintää

- ▶ Vim luonnollisesti antaa käyttäjälle mahdollisuuden käyttää etsintää liikekomentona. Esimerkiksi komento `d/Nyt<CR>` poistaisi kaiken tekstin, mitä on sillä hetkellä kursorin alla ja aina ensimmäiseen Nyt-sanaan asti.
- ▶ Mitään ei valita (tai esimerkissämme poisteta) jos hakusanaa ei löydy
- ▶ Kirjainkoon on täsmättävä. Tähän on asetuksia.

Komento	Suunta
/foo	→
n	→
N	←
n	→
/fob	→
?bar	←
n	←
N	→

Etsintään liittyviä asetuksia

- ▶ `hlsearch`: värjääkö vim hakutulokset tiedostosta
- ▶ `incsearch`: hakeeko vim sanoja sitä mukaa kuin sitä kirjoittaa, vai vasta enteriä painettua
- ▶ `ignorecase`: vim ei välitä kirjainkoosta
- ▶ `smartcase`: hakusanojen kirjainkoolla ei väliä, jos kaikki kirjoitettu pienellä. Jos yksikin kapitaali mukana, koon on täsmättävä täysin

Säännöllisistä lausekkeista

- ▶ Säännölliset lausekkeet ovat hyvin tehokas työkalu Unix-maailmassa
- ▶ Niillä suoritetaan hahmontunnistusta tekstistä, käytännössä monimutkaisia hakulauseita ja korvauksia. Koodin käyttö on vaikeaselkoista ja kielioppiin on erilaisia murteita

Säännöllisistä lausekkeista

- ▶ Säännölliset lausekkeet ovat hyvin tehokas työkalu Unix-maailmassa
- ▶ Niillä suoritetaan hahmontunnistusta tekstistä, käytännössä monimutkaisia hakulauseita ja korvauksia. Koodin käyttö on vaikeaselkoista ja kielioppiin on erilaisia murteita
- ▶ Tällä kurssilla emme puutu asiaan enempää kuin tarpeen: esittelemme jokerit
- ▶ . vastaa yhtä, mitä tahansa kirjainta tai numeroa
- ▶ esimerkiksi haku `a...` löytäisi sanan "auto" tekstistä, tai minkä tahansa nelikirjaimisen kirjainkokonaisuuden, jossa on pieni a-kirjain ja vähintään kolme muuta merkkiä perässä

Säännöllisistä lausekkeista

- ▶ Säännölliset lausekkeet ovat hyvin tehokas työkalu Unix-maailmassa
- ▶ Niillä suoritetaan hahmontunnistusta tekstistä, käytännössä monimutkaisia hakulauseita ja korvauksia. Koodin käyttö on vaikeaselkoista ja kielioppiin on erilaisia murteita
- ▶ Tällä kurssilla emme puutu asiaan enempää kuin tarpeen: esittelemme jokerit
- ▶ . vastaa yhtä, mitä tahansa kirjainta tai numeroa
- ▶ esimerkiksi haku `a...` löytäisi sanan "auto" tekstistä, tai minkä `tahansa` nelikirjaimisen kirjainkokonaisuuden, jossa on pieni a-kirjain ja vähintään kolme muuta merkkiä perässä

Säännöllisistä lausekkeista 2

- ▶ Yksinään jokerimerkin käyttö voi olla hidasta toistaa, siksi sen kaveriksi esitellään säännöllisten lausekkeiden kerroinsysteemi
- ▶ Kunkin kirjaimen tai jokerin perään voidaan laittaa symboli, joka kertoo, paljonko edeltävää merkkiä saa löytyä.
- ▶ Esimerkkejä: "Just aaask" täsmäisi hakuun `/a{3}`, mutta myös hakuihin `/a*` ja `/s.*s`

Kerroin	Lkm
<code>?</code>	0 – 1
<code>+</code>	1 tai enemmän
<code>*</code>	0 tai enemmän
<code>{m}</code>	tasan m kpl
<code>{,m}</code>	enintään m kpl
<code>{m,}</code>	vähintään m kpl
<code>{m,n}</code>	$m - n$ kpl

Säännöllisistä lausekkeista 2

- ▶ Yksinään jokerimerkin käyttö voi olla hidasta toistaa, siksi sen kaveriksi esitellään säännöllisten lausekkeiden kerroinsysteemi
- ▶ Kunkin kirjaimen tai jokerin perään voidaan laittaa symboli, joka kertoo, paljonko edeltävää merkkiä saa löytyä.
- ▶ Esimerkkejä: "Just aaask" täsmäisi hakuun /a{3}, mutta myös hakuihin /a* ja /s.*s

Kerroin	Lkm
?	0 – 1
+	1 tai enemmän
*	0 tai enemmän
{m}	tasan m kpl
{,m}	enintään m kpl
{m,}	vähintään m kpl
{m,n}	$m - n$ kpl

Säännöllisistä lausekkeista 2

- ▶ Yksinään jokerimerkin käyttö voi olla hidasta toistaa, siksi sen kaveriksi esitellään säännöllisten lausekkeiden kerroinsysteemi
- ▶ Kunkin kirjaimen tai jokerin perään voidaan laittaa symboli, joka kertoo, paljonko edeltävää merkkiä saa löytyä.
- ▶ Esimerkkejä: "Just aaask" täsmäisi hakuun /a{3}, mutta myös hakuihin /a* ja /s.*s

Kerroin	Lkm
?	0 – 1
+	1 tai enemmän
*	0 tai enemmän
{m}	tasan m kpl
{,m}	enintään m kpl
{m,}	vähintään m kpl
{m,n}	$m - n$ kpl

Säännöllisistä lausekkeista 2

- ▶ Yksinään jokerimerkin käyttö voi olla hidasta toistaa, siksi sen kaveriksi esitellään säännöllisten lausekkeiden kerroinsysteemi
- ▶ Kunkin kirjaimen tai jokerin perään voidaan laittaa symboli, joka kertoo, paljonko edeltävää merkkiä saa löytyä.
- ▶ Esimerkkejä: "Just **aaask**" täsmäisi hakuun `/a{3}`, mutta myös hakuihin `/a*` ja `/s.*s`

Kerroin	Lkm
?	0 – 1
+	1 tai enemmän
*	0 tai enemmän
{ <i>m</i> }	tasan <i>m</i> kpl
{ <i>m</i> , <i>m</i> }	enintään <i>m</i> kpl
{ <i>m</i> ,}	vähintään <i>m</i> kpl
{ <i>m</i> , <i>n</i> }	<i>m</i> – <i>n</i> kpl

Pari kehittyynyttä liikekomentoa

Tekstin ”maalaus” – visual-moodi

Rekisterit – Vimin vastine leikepöydälle

Etsi...

...ja korvaa

Manuaalista korvausta

- ▶ Mikäli on määrä korvata tiedostossa jotain merkkijonoja toisilla merkkijonoilla, näppärin tapa voisi olla hakea haettavaa merkkijonoa ensin haulla, sitten komentaa `cwkorvattava`
- ▶ Näin kun ensimmäinen kohta on korvattu, voi edetä hyppien seuraavaan kohteeseen `n`-komentoa käytellen, ja komentamalla `.` toistaa edellinen komento, joka sattumoisin oli sananvaihtaja

Manuaalista korvausta

- ▶ Mikäli on määrä korvata tiedostossa jotain merkkijonoja toisilla merkkijonoilla, näppärin tapa voisi olla hakea haettavaa merkkijonoa ensin haulla, sitten komentaa `cwkorvattava`
- ▶ Näin kun ensimmäinen kohta on korvattu, voi edetä hyppien seuraavaan kohteeseen `n`-komentoa käytellen, ja komentamalla `.` toistaa edellinen komento, joka sattumoisin oli sananvaihtaja
- ▶ Jos kuitenkin halutaan korvata tiedoston kaikki ilmentymät kerralla, on siihen hyvä apuväline, `:substitute` tai lyhyemmin `:s`

Substitute

- ▶ Komennon syntaksi on seuraava: `:s/haku/korvaus/asetuksia`
- ▶ `:s` tekee tehtävänsä aina aktiiviselle riville. Kattaakseen koko tiedoston kerralla, lisää `%` kaksoispisteen ja ässän väliin

Substitute

- ▶ Komennon syntaksi on seuraava: `:s/haku/korvaus/asetuksia`
- ▶ `:s` tekee tehtävänsä aina aktiiviselle riville. Kattaakseen koko tiedoston kerralla, lisää `%` kaksoispisteen ja ässän väliin
- ▶ **Haku** on samanlainen kuin tavallisesti etsittäessä: säännölliset lausekkeet kelpaavat
- ▶ **Korvaus** on tavallinen merkkijono, jolla korvataan löydettyt osumat
- ▶ **Asetukset** ovat merkkejä, joilla ohjeistetaan Vimiä hieman tarpeen vaatiessa

Substitute 2 – asetukset ja esimerkit

- ▶ Pari asetusta, joita useimmiten tarvitaan:
- ▶ i: ignore case, eli älä välitä kirjainkoosta
- ▶ g: käy koko rivi läpi, eli älä lopeta korvaamista ensimmäiseen löytyneeseen osumaan

Substitute 2 – asetukset ja esimerkit

- ▶ Pari asetusta, joita useimmiten tarvitaan:
- ▶ `i`: ignore case, eli älä välitä kirjainkoosta
- ▶ `g`: käy koko rivi läpi, eli älä lopeta korvaamista ensimmäiseen löytyneeseen osumaan
- ▶ Esimerkki: `:%s/While/Whilst/g` käy koko tiedoston läpi muuttaen While-sanat Whilsteiksi. Vain isolla kirjaimella aloitettu While korvataan.
- ▶ Esimerkki: `:s/Ke.+o/Pie/g` muuntaa aktiivisella rivillä kaikki merkkijohdistelmät, jotka alkavat yhdistelmällä "Ke" ja päättyvät oohon, Pieksi.
- ▶ Esimerkki: `:s` toistaa edellisen korvauksen sellaisenaan uudestaan. Siirry uudelle riville, komenna `:s`, etene, jne ...

Osa 3: Usean tiedoston kanssa työskentely

Sanatäydennykset

Asetuksia

Puskurit

Ikkunat

Sanatäydennykset

Asetuksia

Puskurit

Ikkunat

Täydennä sanat valmiiksi

- ▶ Vim ei pelkästään vähennä näppäilyjen määrää komentojen suhteen, vaan myös tekstin lisäysmoodissa on useita kirjoittamista nopeuttavia kikkoja

Täydennä sanat valmiiksi

- ▶ Vim ei pelkästään vähennä näppäilyjen määrää komentojen suhteen, vaan myös tekstin lisäysmoodissa on useita kirjoittamista nopeuttavia kikkoja
- ▶ Yleisimmin käytetty täydennyskomento on lisäysmoodissa tehtävä **C-n** ja **C-p** : ne hakevat sanoja aukiolevista tiedostoista vimin syövereissä.
- ▶ Erityisesti pitkien sanojen naputtelu helpottuu, kun riittää näppäillä pari kirjainta sanan alusta ja painaa sitten **C-n**, jolloin Vim täydentää sanan loppuun. Jos samalla alulla löytyy useita eri sanoja, Vim näyttää listan ehdokkaista. Listaa selataan samoilla näppäimillä

Sanakirjatäydennys

- ▶ Jos käyttäjä asettaa itselleen sanakirjan (yksinkertaisuudessaan "sanakirja" on tekstitiedosto, joka sisältää vaikkapa kaikki suomen kielen sanat yksi rivillään), voi hän käyttää komentoa `C-x C-d` täydentääkseen sanoja sanakirjasta
- ▶ Mikäli ehdokkaita on useita, listaa selataan käyttäen komentoja `C-n` ja `C-p`
- ▶ Unixien mukana tulee kohtalaisen kattava englannin kielen sanalista. Windowseissa tämä lista täytyy hakea erikseen, esimerkiksi Kotus tarjoaa² suomen kielen sanalista
- ▶ Sanakirjatiedosto määrätään asetuksella `dictionary`

²<http://kaino.kotus.fi/sanat/nykysuomi/>,
http://www.ohjelmointiputka.net/tiedostot/kotus_sanat.zip

Muita täydennyksiä

- ▶ Vaikka vain tehokäyttäjien suosiossa, **C-x** -valikon takaa löytyy monia vaihtoehtoja, joihin Vim osaa antaa täydennys- ja kirjoitusapua:

Muita täydennyksiä

- ▶ Vaikka vain tehokäyttäjien suosiossa, **C-x** -valikon takaa löytyy monia vaihtoehtoja, joihin Vim osaa antaa täydennys- ja kirjoitusapua:
- ▶ **C-x C-f**: täydennä tiedostonimiä. Näppärä vaikka \LaTeX -tiedostossa hakea kuvatiedostoja

Muita täydennyksiä

- ▶ Vaikka vain tehokäyttäjien suosiossa, **C-x** -valikon takaa löytyy monia vaihtoehtoja, joihin Vim osaa antaa täydennys- ja kirjoitusapua:
- ▶ **C-x C-f**: täydennä tiedostonimiä. Näppärä vaikka \LaTeX -tiedostossa hakea kuvatiedostoja
- ▶ **C-x C-l**: täydennä kokonaisia rivejä. Jos muistat, kuinka joku tietty rivi alkoi, niin kopiointi-liittämisen sijaan voi olla näppärää tehdä kokonainen rivitäydennys.

Muita täydennyksiä

- ▶ Vaikka vain tehokäyttäjien suosiossa, **C-x** -valikon takaa löytyy monia vaihtoehtoja, joihin Vim osaa antaa täydennys- ja kirjoitusapua:
- ▶ **C-x C-f**: täydennä tiedostonimiä. Näppärä vaikka \LaTeX -tiedostossa hakea kuvatiedostoja
- ▶ **C-x C-l**: täydennä kokonaisia rivejä. Jos muistat, kuinka joku tietty rivi alkoi, niin kopiointi-liittämisen sijaan voi olla näppärää tehdä kokonainen rivitäydennys.
- ▶ **C-x C-o**: nk. *omni completion* ei oletuksena tee mitään, mutta siihen voi kehittynyt käyttäjä kirjoittaa oman funktion, joka täydentää vaikkapa \TeX -tageja tai muuta. Vain taivas on rajana

Kielioppitarkastus

- ▶ Vim tukee myös sanojen oikeinkirjoitusta. Sanalistoiksi riittävät sanakirjojen tapaiset listaukset, jotka asetetaan asetukselle `spellfile`
- ▶ Toisaalta jos kielet on asennettu Vimin mukana vakiopaikoille, riittää asettaa komento `spelllang` kielen valitsemiseksi

Kielioppitarkastus

- ▶ Vim tukee myös sanojen oikeinkirjoitusta. Sanalistoiksi riittävät sanakirjojen tapaiset listaukset, jotka asetetaan asetukselle `spellfile`
- ▶ Toisaalta jos kielet on asennettu Vimin mukana vakiopaikoille, riittää asettaa komento `spelllang` kielen valitsemiseksi
- ▶ Kun tarkastus on päällä, Vim värjää korostaen virheelliset tekstit. Komentoja oikolukuun:

<code>]s</code>	Hyppää seuraavaan virheeseen
<code>[s</code>	Hyppää edelliseen virheeseen
<code>zg</code>	Merkitse sana hyväksi
<code>zw</code>	Merkitse sana huonoksi
<code>zug,zuw</code>	Peru merkintä
<code>z=</code>	Lista ehdotuksista sanan korjaamiseksi

Sanatäydennykset

Asetuksia

Puskurit

Ikkunat

Ohjelma-asetuksia

- ▶ Aiemmissa osioissa on jo noussut esille erilaisia asetuksia, joita muuttamalla Vimin toimintaan voi vaikuttaa
- ▶ Komennot asetetaan komentamalla `:set asetus:arvo`
- ▶ Esimerkiksi säädetään Vim oikolukemaan englantia: `:set spelllang=en`

Ohjelma-asetuksia

- ▶ Aiemmissa osioissa on jo noussut esille erilaisia asetuksia, joita muuttamalla Vimin toimintaan voi vaikuttaa
- ▶ Komennot asetetaan komentamalla `:set asetus:arvo`
- ▶ Esimerkiksi säädetään Vim oikolukemaan englantia: `:set spelllang=en`
- ▶ Voimassaolevan asetuksen voi tarkastaa näin: `:set spelllang?`. Vim tulostaa alas asetuksen arvon.

Asetustiedosto

- ▶ Asetuksia säätämällä ei kuitenkaan saavuteta pysyviä tuloksia. Ohjelman sulkeuduttua vaihdetut asetukset katoavat.
- ▶ Ratkaisuna on muokata Vimin käynnistyksessä luettavaa asetustiedostoa *vimrc*
- ▶ Tiedosto löytyy Windowsissa joko Vimin asennuskansioista tai käyttäjän kotihakemiston ylähakemistosta, Unixeissa käyttäjän kotihakemistossa
- ▶ Vimistä käsin on tosin helppo päästä siihen käsiksi: komentamalla `:e $HOME/` ja valitsemalla oikeannäköisen tiedoston – Linuxissa `.vimrc` ja Windowsissa `_vimrc`

Mitä vimrc voi sisältää?

- ▶ Kaikkia asetuksia, käytännössä kaikkia komentoja, joihin laitetaan kaksoispistettä eteen
- ▶ Kuitenkin valtaosin set-komentoja. Myös esimerkiksi väriteeman (`:colorscheme`) valinta luontuu näppärästi
- ▶ Edistyneemmät käyttäjät hyötyvät lyhennelmien ja omien näppäinkomentojen lisäämisestä
- ▶ Esitellään joitain tavallisimpia asetuksia lyhyesti

Yleisimmät asetukset

Ohjeissa on käsitelty kattavasti kutakin näistä asetuksista: selaile vapaasti komentamalla `:help asetus`

<code>nocompatible</code>	Toimiiko Vim alkeis- vai laajennetussa moodissa
<code>tabstop</code>	Tabulaattorin pituus
<code>textwidth</code>	Tekstirivin leveys (asetta 0, jotta Vim ei katko automaattisesti)
<code>linebreak</code>	Vim katkoo pitkät rivit ruudulle näkyväksi
<code>number</code>	Rivinumerot
<code>ruler</code>	Näytä ruudun alareunassa nykyinen rivi ja sarake
<code>hidden</code>	Puskureiden (tulossa) välillä on näppärämpi hyppiä
<code>syntax</code>	Syntaksivärjäys päälle (hyödyllinen koodien kanssa)

Sanatäydennykset

Asetuksia

Puskurit

Ikkunat

Puskurit ovat ikkunoita tiedostoihin

- ▶ Kun Vimissä avataan (`:edit`-komennolla) uusi tiedosto käsittelyyn, vanha ei suinkaan sulkeudu, vaan jää taustalle auki
- ▶ Avonaisia tiedostoja kutsutaan Vimin termein puskureiksi (engl. buffers), ja puskurit vastaavat nykyisten välilehdellisten editorien tabeja: on vain yksi puskuri/tiedosto kerralla näkösällä, per ikkuna

Puskurit ovat ikkunoita tiedostoihin

- ▶ Kun Vimissä avataan (`:edit`-komennolla) uusi tiedosto käsittelyyn, vanha ei suinkaan sulkeudu, vaan jää taustalle auki
- ▶ Avonaisia tiedostoja kutsutaan Vimin termein puskureiksi (engl. buffers), ja puskurit vastaavat nykyisten välilehdellisten editorien tabeja: on vain yksi puskuri/tiedosto kerralla näkösällä, per ikkuna
- ▶ Puskurit ovat eräänlaisia portteja kiintolevyillä sijaitseviin tiedostoihin, kuten muissakin ohjelmissa tapaa olla

Tiedostoja

Vim

Puskureita

Puskureilla navigointi

- ▶ Vim tarjoaa liudan komentoja, joilla puskureissa voi liikkua:

`:edit` Avaa uusi tiedosto (ja uusi puskuri)
`:bn` (Buffer next) Siirry seuraavaan puskuriin
`:bp` (Buffer prev) Siirry edelliseen puskuriin
`:buffers` Listaa kaikki avoinna olevat puskurit
`:bd` (Buffer delete) Sulje nykyinen puskuri

- ▶ Asetus `hidden` kannattaa asettaa päälle joustavaa selailua varten.

Puskureilla navigointi

- ▶ Vim tarjoaa liudan komentoja, joilla puskureissa voi liikkua:

`:edit` Avaa uusi tiedosto (ja uusi puskuri)
`:bn` (Buffer next) Siirry seuraavaan puskuriin
`:bp` (Buffer prev) Siirry edelliseen puskuriin
`:buffers` Listaa kaikki avoinna olevat puskurit
`:bd` (Buffer delete) Sulje nykyinen puskuri

- ▶ Asetus `hidden` kannattaa asettaa päälle joustavaa selailua varten.
- ▶ Me lisäksi sujuvoitamme puskureissa liikkumista luomalla kaksi uutta pikanäppäintä: `C-j` ja `C-k`

Puskureiden kanssa toimimista

- ▶ Kaikki peruskäyttöön tarvittava on puskureista jo käyty, joten otetaan muutama lisätoiminto
- ▶ Komennon `:bufdo` avulla voi jokaiselle aukiolevalle puskurille tehdä saman komennon, esimerkiksi korvauksen

Puskureiden kanssa toimimista

- ▶ Kaikki peruskäyttöön tarvittava on puskureista jo käyty, joten otetaan muutama lisätoiminto
- ▶ Komennon `:bufdo` avulla voi jokaiselle aukiolevalle puskurille tehdä saman komennon, esimerkiksi korvauksen
- ▶ Esimerkki: korvaa sana kaikissa aukiolevissa puskureissa: `:bufdo %s/Ear/Air/g`

Puskureiden kanssa toimimista

- ▶ Kaikki peruskäyttöön tarvittava on puskureista jo käyty, joten otetaan muutama lisätoiminto
- ▶ Komennon `:bufdo` avulla voi jokaiselle aukiolevalle puskurille tehdä saman komennon, esimerkiksi korvauksen
- ▶ Esimerkki: korvaa sana kaikissa aukiolevissa puskureissa: `:bufdo %s/Ear/Air/g`
- ▶ Puskureilla voi olla omia asetuksia. Silloin `:bufdo` on hyödyksi

Sanatäydennykset

Asetuksia

Puskurit

Ikkunat

Vim

Puskureita

Ikkunat

Ikkunat vimissä

- ▶ Vimin termi "ikkuna" ei aivan tarkoita samaa kuin Windowsin oma määritelmä:
- ▶ Kun Vimissä haljotaan ruutua osiin, kukin osaruutu toimii *ikkunana* (engl. *window* tai kuvaavammin *viewport*) johonkin puskuriin

Ikkunat vimissä

- ▶ Vimin termi "ikkuna" ei aivan tarkoita samaa kuin Windowsin oma määritelmä:
- ▶ Kun Vimissä haljotaan ruutua osiin, kukin osaruutu toimii *ikkunana* (engl. *window* tai kuvaavammin *viewport*) johonkin puskuriin
- ▶ Kahdella ikkunalla voi käyttäjä siis tarkastella kahta puskuria samanaikaisesti: jokainen ikkuna saa näyttää siis tasan yhtä puskuria
- ▶ Kaksi eri ikkunaa saa silti näyttää myös samaakin puskuria
- ▶ Kullakin ikkunalla on omat näyttöasetukset. Esimerkiksi rivinumeroita ei aina tarvita joka ikkunassa

Ikkunoiden käyttö

Ikkunoihin liittyy paljon näppäimiä ja komentoja, koska niiden ulkoasuakin voi luonnollisesti muuttaa aika monella tavalla. Onneksi tärkeimmät muistaa johdonmukaisuuden ansiosta.

<code>:sp</code>	(Split) jaa nykyinen ikkuna kahdeksi pienemmäksi
<code>:sp <i>tnimi</i></code>	Jaa ja avaa <i>tnimi</i> toisessa
<code>C-w j</code>	Hyppää alempaan ikkunaan
<code>C-w k</code>	Hyppää ylempään ikkunaan (ja vastaavasti <code>C-w h</code> ja <code>C-w l</code>)
<code>C-w w</code>	Siirry seuraavaan (syklinen)
<code>C-w J</code>	Siirrä tämä ikkuna alemmaksi (vastaavasti muut isolla kirjaimella)
<code>C-w v</code>	(Vertical split) jaa ikkuna vertikaalisesti
<code>:q</code>	Sulje ikkuna (jos vain yksi ikkuna, Vim sulkeutuu)

Ikkunoiden koonmuuttelu

Kun ikkunoita on saatu auki, joustavan käytön takana on kyky nopeasti hallita ikkunoiden kokoa, ja vaikka isontaa aina päädokumentille varattua ikkunaa.

- C-w +/- Nosta ikkunankarmia ylemmäs/alemmas (lisää kerroin eteen)
- C-w </> Siirrä ikkunankarmia oikealle/vasemmalle (lisää kerroin)
- C-w _ Pienennä muut ikkunat yhden rivin korkuisiksi (maksimoi pysty)
- C-w | Kavenna muut ikkunat yhden rivin levysiksi (maksimoi vaaka)
- C-w = Jaa ikkuna-ala tasaisesti kaikille

Ikkunoiden käyttö

- ▶ Ikkunoita voi siis siirrellä ja pyöritellä miltei miten haluaa.
- ▶ Tärkeintä on havaita, että ikkunat eivät ole sidoksissa puskureihin, vaan ikkunaa säätämällä vain valitaan haluamiaan asetuksia puskurien tarkasteluun nähden

Ikkunoiden käyttö

- ▶ Ikkunoita voi siis siirrellä ja pyöritellä miltei miten haluaa.
- ▶ Tärkeintä on havaita, että ikkunat eivät ole sidoksissa puskureihin, vaan ikkunaa säätämällä vain valitaan haluamiaan asetuksia puskurien tarkasteluun nähden
- ▶ Ikkunoihin löytyy myös oma komento `:windo`, joka tekee komennon kullekin ikkunalle

Osa 4: Kehittyneitä, sekalaisia menetelmiä

Digraafit

Lyhennelmät

Makrot

Pikanäppäimet

Muuta sekalaista

Mitä jäi käsittelemättä?

Digraafit

Lyhennelmät

Makrot

Pikanäppäimet

Muuta sekalaista

Mitä jäi käsittelemättä?

Erikoismerkit näppärällä menetelmällä

- ▶ Vim tarjoaa eksoottisille merkeille oikopolun tuottaa niitä. Homman nimi on "digraafi"
- ▶ Esimerkiksi tuottaakseen Copyright-merkin ©, Vim-käyttäjä kirjailee lisäysmoodissa `C-k Co`, ja merkki ilmestyy tekstiin. Ei tietenkään suositeltava tapa \LaTeX -dokumentteihin, mutta noin muuten
- ▶ Listan kaikista merkeistä saa komentamalla `:digraphs`, merkkien "translitterointia" on pyritty pitämään johdonmukaisena ja selkeänä

Digraafit

Lyhennelmät

Makrot

Pikanäppäimet

Muuta sekalaista

Mitä jäi käsittelemättä?

Lyhennelmistä on moneksi

- ▶ Lyhennelmät, engl. abbreviations, sopivat Vimin suunnittelufilosofiaan kuin nyrkki silmään
- ▶ Vim toteuttaa lyhennyksen "täydentämisen" siten, että lisäysmoodissa kirjaillun lyhennelmän tulee olla täsmälleen määritelty, ja tyhjän välin on seurattava

Lyhennelmistä on moneksi

- ▶ Lyhennelmät, engl. abbreviations, sopivat Vimin suunnittelufilosofiaan kuin nyrkki silmään
- ▶ Vim toteuttaa lyhennyksen "täydentämisen" siten, että lisäysmoodissa kirjaillun lyhennelmän tulee olla täsmälleen määritelty, ja tyhjän välin on seurattava
- ▶ Komento oman lyhennelmän tekoon on simppele: `:abbr <lyhenne> <täysi versio>`
- ▶ Esimerkiksi `:abbr MP Mikael Puhakka`

Lyhennelmistä

- ▶ Lyhennelmä täydentyy pitkään muotoonsa myös komentorivillä (siis :-merkillä alkavien komentojen kanssa)
- ▶ Lyhennelmän saa poistettua komentamalla `:unabbrev` ja perään lyhennelmä. Se tulee täydentymään pitkäksi, mutta Vim osaa johtaa halutun lyhennelmän siitä
- ▶ Tallenna halutut, useinkäytetyt lyhennelmät *vimrc*-tiedostoosi, jotta ne ovat aina käytettävissä

Lyhennelmistä mallineisiin

- ▶ Lyhennelmissä voi käyttää myös erikoismerkkejä, joilla saa Vimin simuloimaan näppäinpainalluksia. Tämä voi osoittautua hyvin tehokkaaksi työkaluksi

Lyhennelmistä mallineisiin

- ▶ Lyhennelmissä voi käyttää myös erikoismerkkejä, joilla saa Vimin simuloimaan näppäinpainalluksia. Tämä voi osoittautua hyvin tehokkaaksi työkaluksi
- ▶ Esimerkki: `:abbrev EM \emph{ }<Left><BS>`. Kirjoittaessaan lyhenteen EM tekstiin, se täydentyä `emph`-komennoksi, ja siirtää kursorin yhden ruudun vasemmalle, sulkujen sisään

Lyhennelmistä mallineisiin

- ▶ Lyhennelmissä voi käyttää myös erikoismerkkejä, joilla saa Vimin simuloimaan näppäinpainalluksia. Tämä voi osoittautua hyvin tehokkaaksi työkaluksi
- ▶ Esimerkki: `:abbrev EM \emph{ }<Left><BS>`. Kirjoittaessaan lyhenteen EM tekstiin, se täydentyy `emph`-komennoksi, ja siirtää kursorin yhden ruudun vasemmalle, sulkujen sisään
- ▶ Edellisessä käytetään lisäksi yhden kerran backspacea, koska yleensä lyhenne täydennetään painamalla välilyöntiä. `<BS>` poistaa sen luonnollisesti.

Lyhennelmistä mallineisiin 2

- ▶ Edellä nähtiin muutamaa erikoisnämiskää käytettävän. Vim toteuttaa niitä aivan kuin käyttäjä itse painaisi näppäimistöltä
- ▶ Muita hyödyllisiä näppäimiä ovat ainakin <ESC> ja <CR> (enter). Myös tabulaattori tulisi naputella näppäimen muodossa <Tab>.

Lyhennelmistä mallineisiin 2

- ▶ Edellä nähtiin muutamaa erikoisnämiskää käytettävän. Vim toteuttaa niitä aivan kuin käyttäjä itse painaisi näppäimistöltä
- ▶ Muita hyödyllisiä näppäimiä ovat ainakin <ESC> ja <CR> (enter). Myös tabulaattori tulisi naputella näppäimen muodossa <Tab>.
- ▶ Listat aktiivista lyhennelmistä nähtävillä komentamalla `:abbr` ilman argumentteja

Digraafit

Lyhennelmät

Makrot

Pikanäppäimet

Muuta sekalaista

Mitä jäi käsittelemättä?

Makron nauhoitus ja toisto

- ▶ Makrothan ovat määritelmänsä nojalla tarkkoja työselosteita, jotka käyttäjä nauhoittaa ja mahdollisesti muokkaa. Tämän jälkeen tietokone voi suorittaa makroa omatoimisesti säästäten paljon aikaa
- ▶ Vimissä makro nauhoitetaan näppäilemällä qa , missä a on vapaavalintainen aakkonen. Itse asiassa se on mielivaltainen rekisteri

Makron nauhoitus ja toisto

- ▶ Makrothan ovat määritelmänsä nojalla tarkkoja työselosteita, jotka käyttäjä nauhoittaa ja mahdollisesti muokkaa. Tämän jälkeen tietokone voi suorittaa makroa omatoimisesti säästäten paljon aikaa
- ▶ Vimissä makro nauhoitetaan näppäilemällä `qa` , missä `a` on vapaavalintainen aakkonen. Itse asiassa se on mielivaltainen rekisteri
- ▶ Nauhoitus lopetetaan painamalla näppäintä `q` kerran.
- ▶ Makroa voi suorittaa komentamalla `@a`
- ▶ Edellisen toistetun makron voi toistaa painamalla `@@`

Makron muokkaaminen

- ▶ Makrot nauhoitetaan siis tavalliseen rekisteriin, samaan paikkaan mihin leikkaukset ja kopioinnitkin
- ▶ Nyt valitussa rekisterissä on selväkielisenä tekstinä makro. Tämän voi liittää tekstin sekaan tavalliseen tapaan: "a**p**
- ▶ Makron voi kirjoittaa suoraankin ja sitten kopioida rekisteriin

Makron muokkaaminen

- ▶ Makrot nauhoitetaan siis tavalliseen rekisteriin, samaan paikkaan mihin leikkaukset ja kopioinnitkin
- ▶ Nyt valitussa rekisterissä on selväkielisenä tekstinä makro. Tämän voi liittää tekstin sekaan tavalliseen tapaan: "a**p**
- ▶ Makron voi kirjoittaa suoraankin ja sitten kopioida rekisteriin
- ▶ Esimerkiksi pieni tekstiä käyvä makro voisi näyttää tällaiselta: v{gUuiFoo^[
- ▶ Osaava vim-silmä näkee, että siellä on käytetty ensin komentoa **gU** (make upper case) ja sitten on peruttu sen toiminta **u**:lla (undo), joten ne kaksi komentoa voi poistaa, jos haluaa hienosäätää

Digraafit

Lyhennelmät

Makrot

Pikanäppäimet

Muuta sekalaista

Mitä jäi käsittelemättä?

Omien pikanäppäinten luominen

- ▶ Vimissä toki riittää näppäimiä joka lähtöön, mutta silti on usein tarvetta lisääillä tai peräti muutella vanhoja näppäimiä uusiksi
- ▶ Vim tekee erityisesti tästä asiasta hyvin käytännönläheistä. Jos pystyy omaksumaan sen, että aakkosta painettaessa tuleeekin komento, eikä tekstiä, ollaan jo makroissa ja sitä myötä pikanäppäimissä

Omien pikanäppäinten luominen

- ▶ Vimissä toki riittää näppäimiä joka lähtöön, mutta silti on usein tarvetta lisäillä tai peräti muutella vanhoja näppäimiä uusiksi
- ▶ Vim tekee erityisesti tästä asiasta hyvin käytännönläheistä. Jos pystyy omaksumaan sen, että aakkosta painettaessa tuleeekin komento, eikä tekstiä, ollaan jo makroissa ja sitä myötä pikanäppäimissä
- ▶ Alustava esimerkki: `:map <C-s> :w<CR>` tuo tallennuksen Ctrl-ässän perään

Makrot näppäimen alle

- ▶ Makron voi nauhoittaa, tai sitten ihan kirjailla tarkkaan näppäimet alas
- ▶ Kommentosarjan saa yksinkertaisesti ujutettua sitten näppäinkomennon perään: juuri helpommaksi ei voisi enää mennä

Makrot näppäimen alle

- ▶ Makron voi nauhoittaa, tai sitten ihan kirjailla tarkkaan näppäimet alas
- ▶ Komentosarjan saa yksinkertaisesti ujutettua sitten näppäinkomennon perään: juuri helpommaksi ei voisi enää mennä
- ▶ Esimerkki: luo pikanäppäin välilyönnistä: laita se hyppäämään lause eteenpäin:
- ▶ `:map <Space>)`
- ▶ Mitä kirjoittaisit suoraan Vim-ikkunalle, nyt kanavoit komennoksi

Pikanäppäinten käyttöalue

- ▶ Kuten oletusnäppäimilläkin, pitää omillakin pikanäppäimille välillä määritellä, mihin moodiin ne on tarkoitettu:

Pikanäppäinten käyttöalue

- ▶ Kuten oletusnäppäimilläkin, pitää omillakin pikanäppäimille välillä määritellä, mihin moodiin ne on tarkoitettu:
- ▶ Komento `:imap` luo pikanäppäimen lisäysmoodiin
- ▶ Komento `:nmap` luo pikanäppäimen normaalimoodiin

Pikanäppäinten käyttöalue

- ▶ Kuten oletusnäppäimilläkin, pitää omillakin pikanäppäimille välillä määritellä, mihin moodiin ne on tarkoitettu:
- ▶ Komento `:imap` luo pikanäppäimen lisäysmoodiin
- ▶ Komento `:nmap` luo pikanäppäimen normaalimoodiin
- ▶ Lisäksi `:vmap` visuaalimoodia ja `:cmap` komentoriviä varten
- ▶ Pelkkä `:map` tekee suunnilleen kaikkiin muihin paitsi lisäysmoodiin

Tallennusesimerkin jatkojalostus

- ▶ Aikaisempi esimerkki, `:map <C-s> :w<CR>`, toimii siis vain normaalimoodissa

Tallennusesimerkin jatkojalostus

- ▶ Aikaisempi esimerkki, `:map <C-s> :w<CR>`, toimii siis vain normaalimoodissa
- ▶ Ehkä vastoin Vimin periaatteita, esitellään sopiva ratkaisu lisäysmoodiin toimivaksi:
- ▶ `:imap <C-s> <ESC>:w<CR>i`

Tallennusesimerkin jatkojalostus

- ▶ Aikaisempi esimerkki, `:map <C-s> :w<CR>`, toimii siis vain normaalimoodissa
- ▶ Ehkä vastoin Vimin periaatteita, esitellään sopiva ratkaisu lisäysmoodiin toimivaksi:
- ▶ `:imap <C-s> <ESC>:w<CR>i`
 - ▶ Ctrl-S näppäimeksi

Tallennusesimerkin jatkojalostus

- ▶ Aikaisempi esimerkki, `:map <C-s> :w<CR>`, toimii siis vain normaalimoodissa
- ▶ Ehkä vastoin Vimin periaatteita, esitellään sopiva ratkaisu lisäysmoodiin toimivaksi:
- ▶ `:imap <C-s> <ESC>:w<CR>i`
 - ▶ Ctrl-S näppäimeksi
 - ▶ Poistutaan lisäysmoodista

Tallennusesimerkin jatkojalostus

- ▶ Aikaisempi esimerkki, `:map <C-s> :w<CR>`, toimii siis vain normaalimoodissa
- ▶ Ehkä vastoin Vimin periaatteita, esitellään sopiva ratkaisu lisäysmoodiin toimivaksi:
- ▶ `:imap <C-s> <ESC>:w<CR>i`
 - ▶ Ctrl-S näppäimeksi
 - ▶ Poistutaan lisäysmoodista
 - ▶ Komennetaan `:w` ja enteriä perään

Tallennusesimerkin jatkojalostus

- ▶ Aikaisempi esimerkki, `:map <C-s> :w<CR>`, toimii siis vain normaalimoodissa
- ▶ Ehkä vastoin Vimin periaatteita, esitellään sopiva ratkaisu lisäysmoodiin toimivaksi:
- ▶ `:imap <C-s> <ESC>:w<CR>i`
 - ▶ Ctrl-S näppäimeksi
 - ▶ Poistutaan lisäysmoodista
 - ▶ Komennetaan `:w` ja enteriä perään
 - ▶ Takaisin lisäysmoodiin

Lisäysmoodissa voi olla outojakin yhdistelmiä

- ▶ Lisäysmoodissakaan ei tarvitse käyttää epäergonomista Ctrl-näppäintä, jos ei halua
- ▶ Seuraava esimerkki on esimerkiksi täysin laillinen:
- ▶ `:imap jj <ESC>`

Lisäysmoodissa voi olla outojakin yhdistelmiä

- ▶ Lisäysmoodissakaan ei tarvitse käyttää epäergonomista Ctrl-näppäintä, jos ei halua
- ▶ Seuraava esimerkki on esimerkiksi täysin laillinen:
- ▶ `:imap jj <ESC>`
- ▶ Tällöin lisäysmoodissa kaksi kertaa jiitä painettuaan (nopeahkossa tahdissa) vim rekisteröi sen komennoksi, ja poistuu lisäysmoodista
- ▶ Jos jiiden välissä pitää pienen tauon, saa ne kirjailtua tekstiin

Lisäysmoodissa voi olla outojakin yhdistelmiä

- ▶ Lisäysmoodissakaan ei tarvitse käyttää epäergonomista Ctrl-näppäintä, jos ei halua
- ▶ Seuraava esimerkki on esimerkiksi täysin laillinen:
- ▶ `:imap jj <ESC>`
- ▶ Tällöin lisäysmoodissa kaksi kertaa jiitä painettuaan (nopeahkossa tahdissa) vim rekisteröi sen komennoksi, ja poistuu lisäysmoodista
- ▶ Jos jiiden välissä pitää pienen tauon, saa ne kirjailtua tekstiin
- ▶ Periaatteessa lyhennelmät (`:abbr`) toimivat tällä mekaniikalla, mutta on suositeltavaa käyttää lyhennelmiä sen sijaan, että kokeilisi jotain tämmöistä:
`:imap P Puu`, vaikka se onnistuisikin

Digraafit

Lyhennelmät

Makrot

Pikanäppäimet

Muuta sekalaista

Mitä jäi käsittelemättä?

Vaihda kahden olion paikkaa keskenään

- ▶ Johtuen Vimin tekemistä oletuksista, kahden rivin paikkaa on helppo vaihtaa leikkaamalla rivi ja heti liittämällä se takaisin: `ddp` tekee tempun

Vaihda kahden olion paikkaa keskenään

- ▶ Johtuen Vimin tekemistä oletuksista, kahden rivin paikkaa on helppo vaihtaa leikkaamalla rivi ja heti liittämällä se takaisin: `ddp` tekee tempun
- ▶ Vastaava onnistuu yksittäisille kirjaimille: `xp`

Vaihda kahden olion paikkaa keskenään

- ▶ Johtuen Vimin tekemistä oletuksista, kahden rivin paikkaa on helppo vaihtaa leikkaamalla rivi ja heti liittämällä se takaisin: `ddp` tekee tempun
- ▶ Vastaava onnistuu yksittäisille kirjaimille: `xp`
- ▶ Sanoille joutuu tekemään enemmän työtä:
- ▶ `dWE1p` kuitenkin toimii. Ja senhän voi vaikka asettaa sopivan näppäinkombon alle

Nopea otsikon alleviivaus

- ▶ Oletetaan, että haluat tehdä tekstitiedostossa nätin otsikon nopeasti

Nopea otsikon alleviivaus

- ▶ Oletetaan, että haluat tehdä tekstitiedostossa nätin otsikon nopeasti
- ▶ Ensin voisi kirjailla sen ylös: ggOnättiä otsikkoa tässä<ESC>

Nopea otsikon alleviivaus

- ▶ Oletetaan, että haluat tehdä tekstitiedostossa nätin otsikon nopeasti
- ▶ Ensin voisi kirjailla sen ylös: ggOnättiä otsikkoa tässä<ESC>
- ▶ Sitten huomaat, että pahkeinen, jäi kapsit käyttämättä. Ei hätää, komento gU auttaa. Valitse koko rivi ensin V:llä ja sitten kaikki isoiksi.

Nopea otsikon alleviivaus

- ▶ Oletetaan, että haluat tehdä tekstitiedostossa nätin otsikon nopeasti
- ▶ Ensin voisi kirjailla sen ylös: ggOnättiä otsikkoa tässä<ESC>
- ▶ Sitten huomaat, että pahkeinen, jäi kapsit käyttämättä. Ei hätää, komento gU auttaa. Valitse koko rivi ensin V:llä ja sitten kaikki isoiksi.
- ▶ Alleviivaus: monista otsikkorivi (esimerkiksi yyp), siirry alemmalle riville ja komenna Vr-. Nyt kaikki merkit korvautuvat yksitellen viivoiksi. Syntyvä viiva on tasan yhtä pitkä kuin otsikkokin

Työn kääntäminen

- ▶ Aika usein editorin kanssa pitää säännöllisesti käännellä työn alla olevaa tiedostoa jollain erillisellä ohjelmalla

Työn kääntäminen

- ▶ Aika usein editorin kanssa pitää säännöllisesti käännellä työn alla olevaa tiedostoa jollain erillisellä ohjelmalla
- ▶ Esimerkiksi $\text{T}_{\text{E}}\text{X}$ -tiedostojen kanssa `pdflatex` on aktiivikäytössä. Vim voi helpottaa asiaa monella tavalla:

Työn kääntäminen

- ▶ Aika usein editorin kanssa pitää säännöllisesti käännellä työn alla olevaa tiedostoa jollain erillisellä ohjelmalla
- ▶ Esimerkiksi T_EX-tiedostojen kanssa pdf_latex on aktiivikäytössä. Vim voi helpottaa asiaa monella tavalla:
- Käyttämällä GNU Make -ohjelmaa ja yksinkertaista *makefile*ttä voi Vim hoitaa käännökset yksinkertaisella komennolla `:make`

Työn kääntäminen

- ▶ Aika usein editorin kanssa pitää säännöllisesti käännellä työn alla olevaa tiedostoa jollain erillisellä ohjelmalla
- ▶ Esimerkiksi T_EX-tiedostojen kanssa pdf_latex on aktiivikäytössä. Vim voi helpottaa asiaa monella tavalla:
- ☐ Käyttämällä GNU Make -ohjelmaa ja yksinkertaista *makefile*tä voi Vim hoitaa käännökset yksinkertaisella komennolla `:make`
- ☐ oman komennon voi kirjoittaa (lyhennelmän avulla) tekemään saman tempun ilman ulkopuolisia työkaluja.

Työn kääntäminen

- ▶ Aika usein editorin kanssa pitää säännöllisesti käännellä työn alla olevaa tiedostoa jollain erillisellä ohjelmalla
- ▶ Esimerkiksi T_EX-tiedostojen kanssa pdf_latex on aktiivikäytössä. Vim voi helpottaa asiaa monella tavalla:
- Käyttämällä GNU Make -ohjelmaa ja yksinkertaista *makefile*ttä voi Vim hoitaa käännökset yksinkertaisella komennolla `:make`
- oman komennon voi kirjoittaa (lyhennelmän avulla) tekemään saman tempun ilman ulkopuolisia työkaluja.
- ▶ Esimerkki: `:cabbr MAK !pdflatex %|!pdflatex %` aikaansaisi sen, että komentamalla `:MAK` pdf_latex-ohjelma ajettaisiin kahdesti, argumenttina senhetkinen tiedosto

Työn kääntäminen automaattisesti

- ▶ Vim voisi myös kääntää tiedoston aina kun käyttäjä tallentaa tiedoston tavallisella `:w` -komennolla

Työn kääntäminen automaattisesti

- ▶ Vim voisi myös kääntää tiedoston aina kun käyttäjä tallentaa tiedoston tavallisella `:w` -komennolla
- ▶ Tämän kehittyneen tekniikan taustalla on Vim-komento `:autocmd` (lyh. `:au`)
- ▶ Emme käy automaattitoimintoja tässä läpi, vaan ilmaisemme esimerkin avulla, että mitä on mahdollista tehdä

Työn kääntäminen automaattisesti

- ▶ Vim voisi myös kääntää tiedoston aina kun käyttäjä tallentaa tiedoston tavallisella `:w` -komennolla
- ▶ Tämän kehittyneen tekniikan taustalla on Vim-komento `:autocmd` (lyh. `:au`)
- ▶ Emme käy automaattitoimintoja tässä läpi, vaan ilmaisemme esimerkin avulla, että mitä on mahdollista tehdä
- ▶ Esimerkki: `:au BufWritePost * :!pdflatex %`

Digraafit

Lyhennelmät

Makrot

Pikanäppäimet

Muuta sekalaista

Mitä jäi käsittelemättä?

Komento g osaa monet temput

- ▶ gj , gk , ... : siirry seuraavaan paikkaan visuaalisesti

Komento g osaa monet temput

- ▶ `gj`, `gk`, ... : siirry seuraavaan paikkaan visuaalisesti
- ▶ `g;` hyppää paikkaan, jossa on viimeksi tehty muutoksia

Komento g osaa monet temput

- ▶ `gj`, `gk`, ... : siirry seuraavaan paikkaan visuaalisesti
- ▶ `g;` hyppää paikkaan, jossa on viimeksi tehty muutoksia
- ▶ `gf` avaa se tiedosto, jonka nimi on kursorin alla

Komento g osaa monet temput

- ▶ `gj`, `gk`, ... : siirry seuraavaan paikkaan visuaalisesti
- ▶ `g;` hyppää paikkaan, jossa on viimeksi tehty muutoksia
- ▶ `gf` avaa se tiedosto, jonka nimi on kursorin alla
- ▶ `gu`, `gU`: muunna teksti pieniksi tai isoiksi kirjaimiksi

Komento g osaa monet temput

- ▶ `gj`, `gk`, ... : siirry seuraavaan paikkaan visuaalisesti
- ▶ `g;` hyppää paikkaan, jossa on viimeksi tehty muutoksia
- ▶ `gf` avaa se tiedosto, jonka nimi on kursorin alla
- ▶ `gu`, `gU`: muunna teksti pieniksi tai isoiksi kirjaimiksi
- ▶ `g?` "salaa" teksti ROT13-suojauksella

Komento g osaa monet temput

- ▶ `gj`, `gk`, ... : siirry seuraavaan paikkaan visuaalisesti
- ▶ `g;` hyppää paikkaan, jossa on viimeksi tehty muutoksia
- ▶ `gf` avaa se tiedosto, jonka nimi on kursorin alla
- ▶ `gu`, `gU`: muunna teksti pieniksi tai isoiksi kirjaimiksi
- ▶ `g?` "salaa" teksti ROT13-suojauksella
- ▶ `gq` muotoile valittu alue sopivanlevyiseksi (yleensä 78 merkkiä)

Komento g osaa monet temput

- ▶ `gj`, `gk`, ... : siirry seuraavaan paikkaan visuaalisesti
- ▶ `g;` hyppää paikkaan, jossa on viimeksi tehty muutoksia
- ▶ `gf` avaa se tiedosto, jonka nimi on kursorin alla
- ▶ `gu`, `gU`: muunna teksti pieniksi tai isoiksi kirjaimiksi
- ▶ `g?` "salaa" teksti ROT13-suojauksella
- ▶ `gq` muotoile valittu alue sopivanlevyiseksi (yleensä 78 merkkiä)
- ▶ `g C-g`: anna tietoja kursorin sijainnista (sekä sanamäärä)

Vim osaa tempun jos toisenkin

- ▶ Paikan merkkkaus: `ma` ja siihen palaaminen: `'a`

Vim osaa tempun jos toisenkin

- ▶ Paikan merkkkaus: `ma` ja siihen palaaminen: `'a`
- ▶ Korostus eli `:match`

Vim osaa tempun jos toisenkin

- ▶ Paikan merkkkaus: `ma` ja siihen palaaminen: `'a`
- ▶ Korostus eli `:match`
- ▶ Tiedoston taivuttelu eli *folding*

Vim osaa tempun jos toisenkin

- ▶ Paikan merkkkaus: `ma` ja siihen palaaminen: `'a`
- ▶ Korostus eli `:match`
- ▶ Tiedoston taivuttelu eli *folding*
- ▶ Valmiit tiedostopohjat: esimerkiksi luomalla uuden tiedoston, jonka päätte on `tex`, voi Vim heittää siihen valmiiksi pari perusriviä

Vim osaa tempun jos toisenkin

- ▶ Paikan merkkkaus: `ma` ja siihen palaaminen: `'a`
- ▶ Korostus eli `:match`
- ▶ Tiedoston taivuttelu eli *folding*
- ▶ Valmiit tiedostopohjat: esimerkiksi luomalla uuden tiedoston, jonka pääte on `tex`, voi Vim heittää siihen valmiiksi pari perusriviä
- ▶ Ulkoisten komentojen komentaminen ja syötteen lukeminen tiedostoon

Vim osaa tempun jos toisenkin

- ▶ Paikan merkkkaus: `ma` ja siihen palaaminen: `'a`
- ▶ Korostus eli `:match`
- ▶ Tiedoston taivuttelu eli *folding*
- ▶ Valmiit tiedostopohjat: esimerkiksi luomalla uuden tiedoston, jonka pääte on `tex`, voi Vim heittää siihen valmiiksi pari perusriviä
- ▶ Ulkoisten komentojen komentaminen ja syötteen lukeminen tiedostoon
- ▶ `:sort` lajittelee koko tiedoston aakkosjärjestykseen, tai vain osan Visual-moodin kanssa