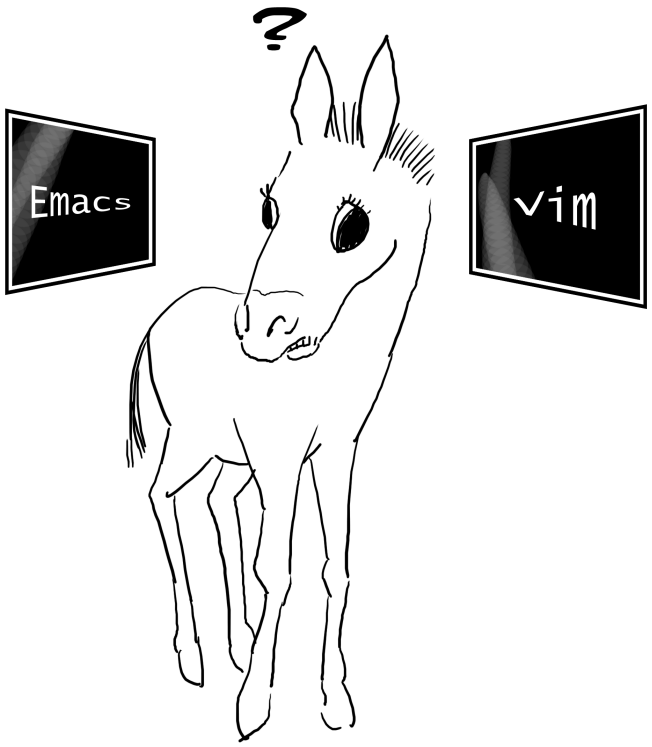
Kapitel 4

Editorer: Emacs eller Vim?



4.1 Editorer

En editor, ett program för att redigera text, är en systemadministratörs viktigaste verktyg. Det finns två stora: Emacs och Vim. Man måste kunna grunderna i båda, men vilken man väljer att använda är upp till var och en. Innan du väljer kan det kännas som åsnan mellan två lika stora hötappar, men prova båda och välj en.

Emacs kommer ursprungligen från Richard Stallman, han som även startade GNU-projektet. I Emacs använder man ctrl- och alt-tangenterna mycket.

Vi uttalas ”vi-aj”. Den skapades redan 1976 av Bill Joy (som sen blev medgrundare till Sun Microsystems). Numera bör man använda varianten Vim (vi improved) som kan mycket mera än den äldre vi.

Det finns ofta bestämda uppfattningar om vilken editor som är bäst. Själv lärde jag mig Emacs först (1996) men övergick senare (98–99) till att mest använda vi. Tobias Hagberg kallar båda för ”Världens bästa textredigerare” i sin bok *Effektivare Linux*. Jag tycker att man måste veta hur man sparar och avslutar i båda så man kan använda system där bara den ena finns. Jag kommer skriva mest om Vim, men börjar med att berätta lite om Emacs. Jag avslutar med att nämna några andra alternativ.

Hela denna bok skriver jag i Vim, med hjälp av typsättningsprogrammet LaTeX.

4.2 Emacs

Emacs kan nästan allt. Den är till stor del skriven i LISP och går att modifiera obegränsat. Det går att surfa på webben, läsa mejl, få terapi och kompilera program utan att lämna editorn.

I dokumentation för Emacs ser man ofta förkortningar som **C-x** och **M-y** (där x och y är olika tecken). **C-x** betyder håll nere Ctrl och tryck x, **M-y** betyder håll nere Alt och tryck y. (På äldre datorer kallades tangenten ofta Meta.) Det går även att trycka ESC istället för Alt, men då ska man inte hålla nere den när man trycker på nästa tangent. Till exempel för att spara dokumentet är kommandot **C-x C-s**, det betyder håll nere Ctrl och tryck först x och sen s utan att släppa Ctrl. För att avsluta Emacs trycker man **C-x C-c**. När du vill

skriva text i ett dokument är det bara att skriva. Du kan flytta markören med piltangenterna som vanligt, och kan lära dig några kommandon i taget. Om du försöker avsluta utan att ha sparat får du en fråga om du vill göra det. Det finns mycket inbyggd hjälp och dokumentation i Emacs.

C-a hoppar till början av en rad, **C-e** till slutet. Med **C-s** kan du söka inkrementellt (tecken för tecken) framåt i texten. **C-r** gör samma sak bakåt. Terapeuten får du fram genom att trycka **M-x doctor**.

För att ångra ändringar trycker du **C-x u** (u betyder undo). Vill du öppna en ny fil trycker du **C-x C-f filnamn**. Det går bra att använda TAB-komplettering för både filnamn och kommandon. Med **C-g** avbryter du ett kommando.

Vill du läsa mera om Emacs finns boken *Learning GNU Emacs* av Cameron et al från O'Reilly förlag.

4.3 Vim

Vim finns till alla GNU/Linux-distributioner. Ibland kan man behöva lägga till aliaset `vi=vim` i sin `.bashrc`, men på de flesta system blir det Vim som startas när man skriver `vi` vid prompten.

Vim kan mycket mera än den traditionella vi, till exempel kan man ha flera fönster med olika filer och man kan backa (undo) obegränsat antal ändringar.

4.3.1 Lägen

För att förstå vi och Vim så måste man lära sig att dessa textredigerare har två olika lägen. Kommandoläge (command mode) och insättningsläge (insert mode). I kommandoläget använder man bokstäver för att flytta markören, ta bort ord och rader, klistra in osv. Inget man "skriver" kommer med i filen utan allt är kommandon till editorn. Däremot i insättningsläge kommer all text du skriver som text i filen. Det är nästan som att ha två olika tangentbord. När du startar Vim, till exempel med kommandot `vi fil.txt` så kommer du först i kommandoläget. Enklaste sättet att komma till insättningsläget är att trycka **i** som ger dig möjlighet att skriva vidare där markören är. För att återvända till kommandoläge trycker du Escape **<ESC>**. Vim (men inte den ursprungliga vi) visar på nedersta raden vilket läge du är

i. Eller rättare sagt så står det – INFOGA – när du är i insättningsläge och inget alls i kommandoläge. Är du osäker på vilket läge du är i kan du alltid komma till kommandoläge med **<ESC>**. Det finns även tre visuella lägen (visual mode) i Vim, men det kan vara överkurs.

4.3.2 Spara och avsluta

Vissa kommandon i kommandoläge börjar med ett kolon, dessa är ursprungligen från editorn `ex` (som i sin tur byggde vidare på `ed`). De vanligaste av dessa kolonkommandon används för att spara och avsluta. Jag sammanfattar dem i tabell 4.1.

Tabell 4.1: Spara och avsluta

:w	spara (write)
:q	avsluta (quit)
:wq	spara och avsluta
:q!	avsluta och ignorera ändringar
:x	spara om filen ändrats, avsluta
:n	gå till nästa fil
:b [nummer|namn]	byt fil

Om du öppnar flera filer, t.ex. `vi *.sh` kan du alltså byta till nästa fil med **:n**, eller med **:b2** till den andra i listan, eller **:b backup.sh** där du anger namnet. Det funkar med TAB-komplettering av både filnamn och kommandon i Vim.

ZZ gör samma som **:x**. Skillnaden med **ZZ** och **:x** jämfört med **:wq** är att filens modifieringstid bara ändras om filen ändrats, det kan göra skillnad vid kompilering (`make` behöver enbart kompilera det nya). **:w** ska du trycka ofta, visserligen sparar Vim automatisk en säkerhetskopia, men det är ändå bra att regelbundet spara sin text. Vill du spara den med ett annat namn så anger du det efter **:w**, alltså **:w kopia.txt**. Det kan vara bra att veta om du märker att du inte har rättighet att skriva till filen som du öppnat. Då kan du spara med annat namn i din hemkatalog eller /tmp och sen flytta med `sudo mv`.

4.3.3 Kommandoläget

I kommandoläget har nästan alla bokstäver, och vissa andra tecken, på tangentbordet en speciell funktion. Jag nämner bara vissa, de som jag själv brukar använda.

För att komma till insättningsläge kan du trycka **i** (insert) eller **a** (append). Skillnaden är att med **a** kommer markören hamna efter det tecken du står vid och med **i** blir det före tecknet. Vill du börja skriva på en ny rad nedanför så trycker du **o** (open new line). Vill du ha ny rad ovanför trycker du **O** (stort O).

För att radera en rad trycker du **dd**. Vill du radera fem rader trycker du **5dd**. Det du senast har raderat (klippt ut) kan du klistra in på nytt ställe genom att trycka **p** (put). Kopiera heter ”yank” i vi, för att kopiera en hel rad trycker du **yy**, och klistra in gör du med **p** som vanligt. Även **yy** kan föregås av ett antal, alltså för att kopiera nuvarande rad och tre nedanför så kan du trycka **4yy**.

Du kan flytta dig runt i texten med piltangenterna, eller med **h**, **j**, **k**, **l** (vänster, ner, upp, höger). Jag använder oftast pilarna, men vana vi-användare föredrar att inte flytta fingrarna för mycket. I ”gammal-vi” fungerar inte alltid piltangenterna i insättningsläge, utan man måste trycka **<ESC>** innan man kan flytta markören. Detta problem har man inte i Vim.

När du vill hoppa till början av en rad trycker du **0** (noll), för slutet av en rad **$**. För att söka efter en sträng, framåt i texten, trycker du / och skriver strängen. Det kan vara ett praktiskt sätt att få markören på rätt ställe. Vill du hoppa till nästa förekomst av strängen trycker du **n** (next). Med standardinställningarna börjar sökningen om från toppen om man når botten av texten (detta går att stänga av).

Om du gör fel och vill ångra så trycker du **u** (undo). Vim har obegränsat antal nivåer av undo (traditionell vi enbart en nivå, nästa u blir redo). För redo (gör om) i Vim trycker du **ctrl-r**.

Vill du radera ett tecken trycker du **x**. Även den kan ha ett antal framför sig: **7x**.

4.3.4 Insättningsläget

I insättningsläget kommer de flesta tangenttryckningar direkt i filen. Du kan flytta runt med pilarna i Vim, men

inte alltid i vi. Det finns några kommandon som fungerar i ”insert mode”, ett exempel är **ctrl-n** som ger dig en lista av ord att lägga till.

4.3.5 Interagera med filer och skal

Vill du köra ett enstaka skalkommando utan att lämna Vim så trycker du **:!** följt av kommandot. Till exempel **:!ls**. När du trycker enter så kommer du tillbaka till Vim. Vill du köra flera kommandon är det enklare att suspendera Vim med **ctrl-z**. Då får du en vanlig prompt och återvänder till Vim med `fg`. Ett annat sätt är att skriva **:sh** i kommandoläget.

Du kan även läsa in text från en fil till editorns text. Ställ markören där du vill infoga filen och tryck **:r filnamn**. Till exempel **:r /etc/redhat-release** (det funkar bra med TAB-komplettering).

Vill du istället infoga utdata från ett program skriver du **:r!kommando**, till exempel `:r!date` för att få dagens datum.

Vill du öppna en helt ny fil skriver du **:e filnamn**.

4.3.6 gvim

Om du föredrar grafiska program, och kör X, så kan du testa gvim. Det är som vanliga Vim, men med menyer och mus. Det kan vara ett bra sätt att lära sig vanliga Vim, eftersom kortkommandon står i menyerna. I Red Hat heter paketet du behöver vim-X11. I Debian kan du välja mellan vim-gnome och vim-gtk.

4.3.7 Fönster

Även i den textbaserade Vim (men inte i ”gammal-vi”) kan du ha flera fönster. Alla kommandon för att skapa och byta fönster börjar med **ctrl-W** (ctrl förkortas ofta till ^, och observera att det är stort W). Med **ctrl-Ws** (eller **:split**) delar du skärmen eller terminalen i två (eller flera) fönster horisontellt. Vill du öppna nytt fönster utan att öppna samma fil så trycker du **ctrl-Wn** (eller **:new**). Du växlar fönster neråt med **ctrl-Wj** och uppåt med **ctrl-Wk**. För att stänga aktuellt fönster trycker du **ctrl-Wc**.

4.4 Andra alternativ

I GNOME kan man använda gedit. Den började som en enkel textredigerare, men har fått flera funktioner, dock inte lika många som Vim eller Emacs.

En annan enkel editor är nano, eller dess föregångare pico. Pico följde från början med mejlklienten Pine, och nano är i princip samma program fast med bättre licens. Den är ganska begränsad, men lätt att lära sig. Kommandon inleds med ctrl, och de vanligaste finns listade längst ner i fönstret.

4.5 $EDITOR

Skalvariabeln $EDITOR bestämmer ofta vilken editor som ska användas. T.ex. för `visudo` (som jag nämnde i 3.5) även för `vipw`, `vigr` och `crontab -e`. Ibland kan man även använda variabeln $VISUAL. På många distributioner är EDITOR=vi, men på Ubuntu är den nano. Det kan vara lite förvirrande för kommandon som börjar med ”vi”. Lägg till `export EDITOR=vim` i t.ex. `.bashrc` för att ändra det.

4.6 Lästips

- För Vim finns en bra bok från O’Reilly: Robbins, Arnold & Elbert, Hannah & Lamb, Linda (2008). *Learning the vi and Vim Editors*: O’Reilly
- För Emacs finns: Cameron, Debra & Rosenblatt, Bill & Elliot, James (2004). *Learning GNU Emacs*: O’Reilly