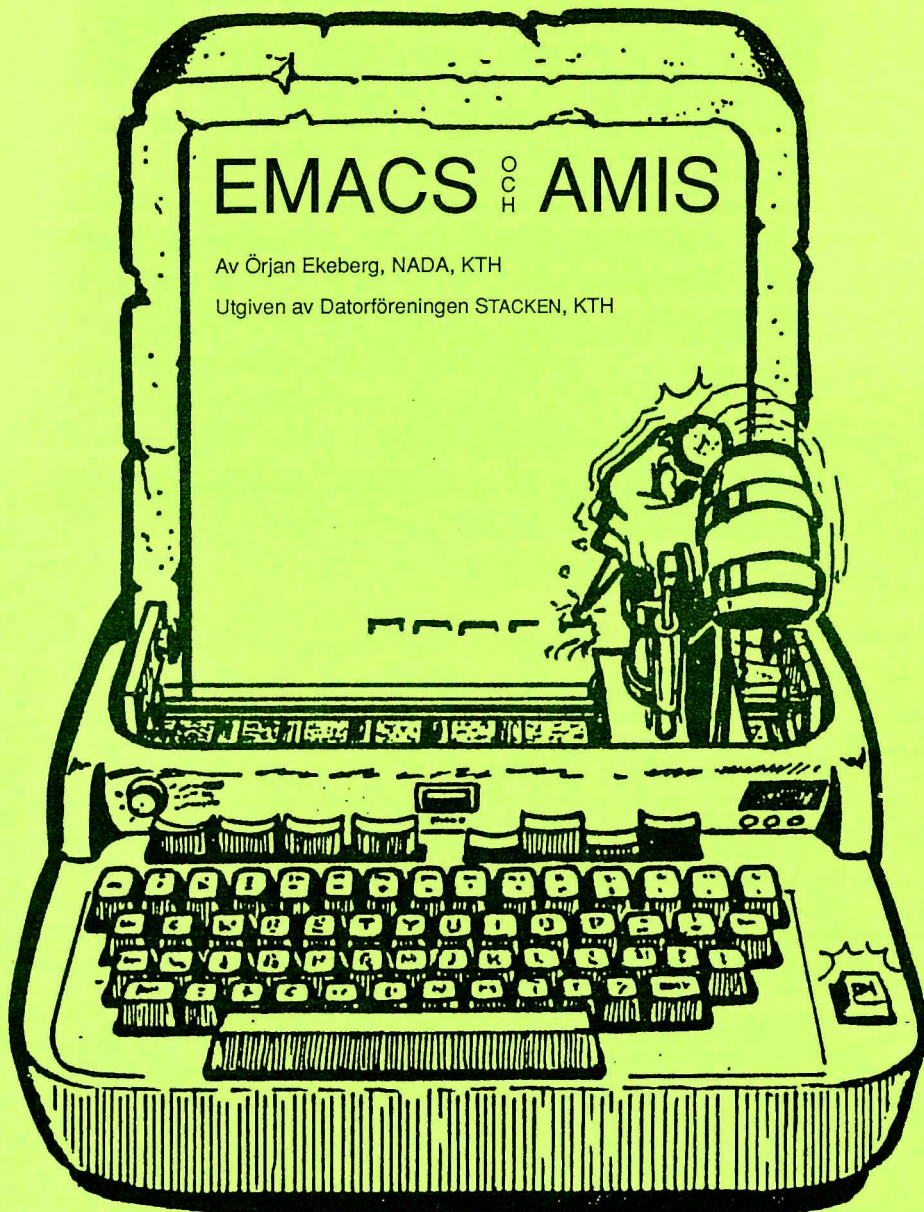


EMACS OCH AMIS

Av Örjan Ekeberg, NADA, KTH

Utgiven av Datorföreningen STACKEN, KTH



Datorföreningen STACKEN

STACKEN är datorföreningen på KTH. Vi har en mängd intressanta verksamheter på gång. Dock hänger det ytterst på den enskilde medlemmen att avgöra vad han eller hon vill göra för föreningen. STACKEN är en idéell förening, där intresse för datorer är den gemensamma faktorn. Sedan föreningen grundades 1978 har vi (bland annat) åstadkommit:

- samköp av mikrodatorbyggsatser
- diverse kurser och föredrag
- studiebesök på intressanta ställen
- AMIS, den portabla EMACS-kompatibla editorn för TOPS-10, VMS, PRIME, NORD, ...
- STACKPOINTER, föreningens tidning
- en egen datorhall för våra stordatorer
- en egen DEC-10:a (se nedan)

Vi har sedan ett år tillbaka en egen DEC-10, som vi installerat, felsökt och kör på. Det är en gammal modell med KA10-processor. Vi kallar henne KATIA. Hon står i vår nya klubblokal och maskinhall "B30", på Brinellvägen 30 (V-sektionen) på gaveln mot Lill-Jansskogen (där vi har en egen ingång). En våning ovanför finns en hörsal, där vi håller till vid större möten.

Ordinarie möten är första torsdag i varje månad kl. 19 i antingen lokalen eller hörsalen. Är Du intresserad av föreningen, är Du välkommen till något av våra möten. Vill Du sedan bli medlem, lämnar Du in en skriftlig ansökan till styrelsen eller skickar den till vår postadress:

Datorföreningen STACKEN
c/o NADA
KTH
100 44 STOCKHOLM

*Computers have more fun!
Computare necesse est!*

Innehåll

1	Introduktion	4
2	Begrepp	7
3	Grundläggande kommandon	10
4	Starta Emacs	17
5	Numeriskt argument	21
6	Hur man skriver kommandon	22
7	Hur man tar sig ut	25
8	Hjälp och dokumentationskommandon	27
9	Flytta arbetspunkten	30
10	Märket och området	32
11	Radering och flyttning av text	34
12	Sökning	37
13	Utbyte av text	39
14	Huvudtillstånd, undertillstånd och bibliotek	40
15	Speciella text-kommandon	45
16	Flera buffertar	49
17	Filhantering	51
18	Två fönster	53
19	Variabler	54
20	Kommandomakron	58
21	Sagan om Amis (av Per Lindberg)	59
22	Amis på Katia (av Jan Michael Rynning)	65
23	Sakregister	67

Synpunkter på detta häftes innehåll och utformning är välkomna. Skriv ner dem på en lapp adresserad till *NADAs systemgrupp* och lämna den till någon handledare eller lärare vid NADA. Du kan också skicka lappen till *Systemgruppen, NADA, KTH, 100 44 Stockholm* eller lägga den i postfacket märkt *Systemgruppen* i NADAs posttrum.

1

Introduktion

Emacs

Emacs är en avancerad skärmorienterad texteditor. En texteditor är ett program man använder för att redigera text med. Den har ursprungligen utvecklats av Richard M Stallman vid MIT i USA och kan användas under operativsystemen TOPS-20, Tenex och ITS på Digital Equipments datorer i serien PDP-10.

Emacs har kallats för en avancerad, självdokumenterande, anpassningsbar, utvidgbar och skärmorienterad realtidseditor.

Skärmorienterad därför att man utnyttjar moderna skärmterminalers möjligheter att visa den text du arbetar med för att du ska få en så bra överblick som möjligt.

Realtidseditor därför att skärmen uppdateras omedelbart då du gör en ändring för att du ska slippa hålla i huvudet vad du gjort. Normalt gör varje tangenttryckning att texten och därmed även skärmen uppdateras.

Avancerad därför att Emacs har betydligt fler och mer kraftfulla kommandon än många andra editorer. Som nybörjare kan du klara dig med ett mycket litet antal kommandon men allt eftersom du blir mer van kan du börja använda speciella funktioner som passar för just din typ av textredigering.

Självdokumenterande därför att Emacs innehåller speciella kommandon för att nå all dokumentation om Emacs. Det finns möjlighet att få beskrivningar av speciella kommandon, söka efter kommandon inom ett visst ämne, få hjälp då man håller på att skriva ett kommando och att läsa allmänna beskrivningar av både grundläggande och avancerade funktioner i Emacs.

Anpassningsbar därför att användaren har stora möjligheter att anpassa funktionen hos Emacs för egna ändamål och önskemål genom att ändra värden på speciella *variabler*. Detta kan göras antingen direkt med kommandon eller i en initieringsfil som läses varje gång man startar Emacs.

Utvidgbar därför att nya Emacskommandon kan definieras direkt då man kör Emacs. Eftersom avancerad Emacsutvidgning kräver kunskaper i programmering i det mycket speciella programspråket TECO är det inte något som den vanlige användaren gör. Däremot är det mycket enkelt att använda en utvidgning som någon annan har skrivit i form av ett bibliotek. En mängd sådana bibliotek finns färdiga att användas av den som har nytta av dem.

Amis

Ett flertal Emacsliknande editorer har utvecklats för att kunna användas under andra operativsystem, bl a Zwei (LispMachine), Mince (CP/M-mikrodatorer), Pepper (PERQ), Fine (DEC-10/TOPS-10), Amis (VAX/VMS, DEC-10/TOPS-10, PDP-11/RSTS-E, Prime/Primos, NORD-500, m fl) och flera andra, där vissa kallas Emacs efter förebilden.

Amis är skriven av en grupp personer i Datorföreningen STACKEN vid KTH. Amis är skriven i programspråket Pascal med undantag för några få specialrutiner i maskinnära språk. Detta har gjort det möjligt att flytta Amis till ett flertal olika datorer.

Amis har alla vanliga kommandon från Emacs, men saknar Emacs möjligheter att kunna programmeras och anpassas genom speciella bibliotek.

Hur man läser detta häfte

Detta häfte är i första hand avsett som en användarhandledning för dig som använder Emacs på Vera. Skillnaden mot andra TOPS-20 system är dock minimal. Även om du använder Amis ska du kunna utnyttja häftet. De funktioner som saknas eller är annorlunda i Amis är därför speciellt markerade i texten (oftast som fotnoter).

Börja med att läsa igenom kapitel 2 och 3 som beskriver viktiga begrepp resp grundläggande kommandon. Därefter kan du tillräckligt mycket för att börja använda Emacs. Provkör några gånger och prova gärna också det självinstruerande programmet Teach-Emacs¹ som låter dig succesivt prova olika kommandon. Teach-Emacs startas med kommandot

```
@TEACH-EMACS
```

Kapitlen 4 till 8 beskriver allmängiltiga principer som du har nytta av att känna till, t ex hur man får hjälp, hur man avbryter felaktiga kommandon och hur man ger mer avancerade kommandon.

Kapitlen 9 till 20 går sedan igenom olika grupper av kommandon och beskriver samtidigt begrepp som hör samman med dessa. Dessa kapitel kan du gärna skumma igenom för att få en bild av vad som finns att lära. Använd dem sedan som uppslagsbok då du börjar använda nya kommandon. Även grundläggande kommandon beskrivs här, men på ett utförligare sätt än i kapitel 3.

På baksidan av häftet finns en sammanställning av kommandon ordnade efter funktionen. Ha den gärna uppslagen bredvid terminalen när du arbetar med Emacs.

Om du vill eller behöver veta mer än det som beskrivs här finns det mycket mer

1 För Amis heter motsvarande program ATeach.

dokumentation att tillgå. All dokumentation går att nå genom hjälpsystemet INFO i Emacs. Kapitel 8 beskriver hur du använder detta.

2

Begrepp

Vi kommer i detta kapitel beskriva några grundläggande begrepp och principer som du bör känna till då du använder Emacs. Vi går först igenom vad arbetspunkten är för något och hur man arbetar med den i bufferten. Därefter följer en beskrivning av hur olika delar av skärmen används.

Bufferten och arbetspunkten

Bufferten är det arbetsutrymme i datorns minne där texten lagras medan du redigerar den. Då man ska ändra i en fil läser man in texten från filen till bufferten, utför ändringen och skriver ut buffertinnehållet på filen igen. Du bestämmer själv när en fil ska läsas in och när buffertinnehållet ska skrivas ut genom att ge speciella kommandon för detta. Emacs håller automatiskt reda på vilken fil du läst in i bufferten.

Arbetspunkten är den position i bufferten där du arbetar för tillfället. Nästan alla kommandon utför sina operationer vid arbetspunkten. Nya tecken stoppas in där, raderingskommandon tar bort tecken där o s v. Andra kommandon används för att flytta arbetspunkten i bufferten. Arbetspunkten befinner sig alltid mellan två tecken (eller före första eller efter sista tecknet) i bufferten.

Emacs tillåter också att man har flera samtidiga buffertar med olika innehåll, normalt inläst från olika filer. Emacs håller då reda på vilken fil som hör ihop med vilken buffert och man har en separat arbetspunkt i varje buffert. Användning av flera buffertar beskrivs i kapitel 16.

Fönstret

När du kör Emacs upptas större delen av skärmen av det s k **fönstret**. De nedersta raderna används för andra ändamål, mer om detta i nästa avsnitt. Fönstret visar alltid en del av bufferten runt den plats där arbetspunkten befinner sig. När du ger ett kommando som gör att arbetspunkten flyttas så att den inte längre finns med i fönstret flyttas automatiskt fönstret i bufferten så att den nya arbetspunkten blir synlig.

När Emacs väntar på inmatning från terminalen placeras terminalens markör (den blinkande fyrkanten eller strecket) på tecknet omedelbart efter arbetspunkten i fönstret. Kom ihåg att arbetspunkten står mellan två tecken. Markören gör att du enkelt kan se var arbetspunkten finns.

I denna handledning använder vi symbolen  för att markera arbetspunktens position. Rutan motsvarar terminalens blinkande markör medan punkten i vänsterkanten symboliserar arbetspunktens läge.

Ibland är rader i bufferten så långa att de inte ryms på en rad på skärmen. Emacs visar sådana rader genom att placera ett utropstecken längst till höger på skärmen och sedan fortsätta raden på nästa skärmrad. Observera att detta bara har att göra med hur raden visas i fönstret. I bufferten och filen kan raderna vara hur långa som helst.

Modraden och ekoraderna

Modraden är raden omedelbart under fönstret på skärmen. Den inleds normalt med ordet `EMACS` och innehåller viktig information om den aktuella bufferten. Under Mod-radens finns en eller flera så kallade **ekorader**. Här skrivs korta meddelanden och frågor till användaren och här listas de kommandon du skriver (i vissa fall).

Ett typiskt utseende för modraden är:

```
EMACS (huvud under) Main: PS:<filkatalog>fil.typ  --17%-- *
```

Ordet `EMACS` talar om att du är inne i Emacs, dvs kör programmet Emacs. Ibland står det något annat namn här och då fungerar inte Emacs kommandon som vanligt. I Amis står det naturligtvis `AMIS` istället för `EMACS`.

(*huvud under*) talar om vilket huvudtillstånd Emacs befinner sig i och eventuella under-tillstånd som är aktiva. Tillstånden betyder att Emacs är speciellt anpassad för redigering av viss typ av text. Huvudtillståndet styrs normalt av filtypen för den inlästa filen.

`Main:` är namnet på den buffert som syns i fönstret. Om du bara använder en buffert heter den alltid just **Main**.

`PS:<filkatalog>fil.typ` är namnet på den fil som lästs in i bufferten. Om du sparar buffertinnehållet utan att ange något annat filnamn används detta namn automatiskt.

`--17%--` talar om var fönstret är placerat i bufferten. `17%` betyder att 17 procent av hela texten är ovanför fönstret. Ibland står det `--TOP--` som betyder att fönstret är i buffertens början, eller `--BOT--` då buffertens slut är med i fönstret. När texten i bufferten är så kort att hela texten ryms i fönstret är denna information överflödigt och finns inte med på modraden alls.

Asterisken i slutet av modraden talar om att någon ändring har gjorts i bufferten sedan den senast sparades på fil. Denna information är extra viktig för den som har många samtidiga buffertar.

Överskrivningsmeddelanden

Korta meddelanden skrivs som sagt ut på ekoraderna. Längre meddelanden och hjälptexter ryms dock inte på dessa få rader. Därför används en speciell metod för att visa dem: **Överskrivning**.

Överskrivning innebär att meddelandet skrivs över fönstret på skärmen. Det försvinner igen om du trycker på mellanslagstangenten eller ger något kommando.

Ofta är överskrivningsmeddelandet så långt att det inte ens ryms på fönstrets plats på skärmen. Då skrivs texten -MORE- i modradens högra del för att markera att det finns mer att läsa. Mellanslag gör att du får läsa vidare i meddelandet. RUBOUT (eller något kommando) gör att du hoppar över resten. Ordet FLUSHED skrivs då i modradens högra del.

3

Grundläggande kommandon

Här ska vi gå igenom några få grundläggande kommandon som du säkert behöver. Vi går igenom hur man stoppar in och tar bort text, flyttar sig runt i bufferten och slutligen sparar resultatet på fil.

Detta kapitel innehåller många framåtsreferenser till senare kapitel där viktiga ämnesområden beskrivs utförligare. Om du har läst och förstått det förra kapitlet bör du kunna börja köra Emacs samtidigt som du läser detta kapitel.

Start av Emacs

För att starta Emacs använder du EXEC-kommandot

```
@EMACS (FILE) filnamn
```

som dessutom gör att Emacs läser in angiven fil i bufferten.¹ Man kan också starta Emacs på andra sätt, t ex utan att läsa in någon fil. Mer om detta i kapitel 4.

Hur man startar Amis beror på vilken dator och vilket operativsystem som används. Det vanligaste sättet är att man skriver ett kommando

```
AMIS filnamn
```

som motsvarar EMACS-kommandot. Se även kapitel 4.

Stoppa in text

Det behövs inget speciellt kommando för att stoppa in text vid arbetspunkten. Det finns ingen "Insert Mode" som många andra editorer har, utan alla vanliga tecken du skriver stoppas in i bufferten direkt.

Den text du skriver skjuts in vid arbetspunkten. Om bufferten t ex innehåller texten:

```
Woozle
```

¹ Detta är en lokal ändring för Vera. Om du använder något annat datorsystem bör du först kontrollera hur det fungerar där.

och arbetspunkten är placerad före z (markören på skärmen blinkar vid z) och du trycker på **A**-tangents blir resultatet:

```
WooAzle
```

med arbetspunkten fortfarande före z.

Radbyten stoppas in genom att du trycker på **RETURN**-tangents. Detta använder du både för att skapa nya blanka rader (när arbetspunkten står i slutet av en rad) och för att dela av en rad (när arbetspunkten står inne på raden). Om du fortsätter i exemplet med att trycka på **RETURN** så blir nu resultatet:

```
WooA
zle
```

Om du skriver fel när du skriver in text kan du använda **RUBOUT**-tangents (märkt **DEL** på vissa terminaler) för att radera de tecken du just skrev. **RUBOUT** är nämligen ett Emacs-kommando som raderar tecknet omedelbart före arbetspunkten. Även radbyten kan raderas på detta sätt. Om arbetspunkten står omedelbart efter ett radbyte, d v s först på en rad, så innebär **RUBOUT** att radbytet tas bort.

Skriva kommandon

Du har nu lärt dig hur man skriver in text och raderar enstaka tecken. Du måste också behärska några kommandon för att flytta arbetspunkten. Först ska vi kort beskriva hur man ger enkla kommandon.

Emacs stoppar ju in alla vanliga tecken som vi skriver i bufferten. Kommandon måste alltså bestå av ovanliga tecken eller åtminstone inledas med ett ovanligt tecken. "Ovanliga tecken" i denna mening är styrtecknen som produceras genom att du håller tangents märkt **CTRL** nedtryckt samtidigt som du trycker på någon annan tangent. Vissa styrtecken har speciella tangents. Viktigast är **ESCAPE**-tangents, ofta märkt **ESC**.

De två vanligaste typerna av kommandon är **CTRL**- och **META**-kommandon. En utförligare beskrivning finns i kapitel 6.

CTRL-kommandon (uttalas kontroll-) åstadkommer du helt enkelt genom att hålla **CTRL**-tangents nedtryckt samtidigt som du trycker på en annan tangent. Normalt används det förkortade skrivsättet **C-A** för Ctrl-A, **C-B** för Ctrl-B o s v. Alla mycket vanliga kommandon är **CTRL**-kommandon för att de är enkla att skriva. Man kan därför betrakta **CTRL**-tangents som ett slags kommando-skift. Med den tangents nedtryckt ger man kommandon, annars skriver man in text.

META-kommandon kan man ge på några olika sätt. Om din terminal har en **META**-tangent används den precis som **CTRL**-tangents. Annars får du ge **META**-komman-

don genom att först trycka på **[ESCAPE]** och därefter ett tecken till. Normalt används det förkortade skrivsättet **[M-A]** för Meta-A, **[M-B]** för Meta-B o s v. För att t ex ge kommandot **[M-F]** trycker du alltså först på **[ESCAPE]** och sedan på **[F]**.

Nu ska vi gå vidare och titta på några CTRL- och META-kommandon som du säkert kommer att behöva.

Flytta arbetspunkten

[C-F]	(Forward)	Flytta arbetspunkten ett steg framåt
[C-B]	(Backward)	Flytta arbetspunkten ett steg bakåt
[C-N]	(Next)	Flytta arbetspunkten rakt ner till nästa rad
[C-P]	(Previous)	Flytta arbetspunkten rakt upp till föregående rad
[C-E]	(End)	Flytta arbetspunkten till slutet av raden
[C-A]		Flytta arbetspunkten till början av raden
[C-V]		Gå en skärmsida framåt
[M-V]		Gå en skärmsida bakåt

Antag att din buffert innehåller texten

```
Första raden
Andra raden
Sista raden
```

Arbetspunkten är placerad före r i ordet Andra. Kommandot **[C-F]** flyttar arbetspunkten till efter r, **[C-P]** flyttar den till före s i ordet Första, **[C-A]** flyttar den till före A i Andra, o s v. Genom att upprepa dessa kommandon tillräckligt många gånger, kan du placera arbetspunkten var du vill i bufferten.

Kommandot **[C-V]** kan du använda för att bläddra framåt i en text. **[M-V]** bläddrar på motsvarande sätt bakåt.

Radera text

Vi har nu gått igenom hur man flyttar sig runt i bufferten, skriver in ny text och raderar enstaka tecken (med **[RUBOUT]**). Detta räcker för att göra elementär redigering. Vi ska nu titta lite mer på hur man gör för att radera text.

[C-D]	(Delete)	Radera tecknet efter arbetspunkten
[RUBOUT]		Radera tecknet före arbetspunkten
[C-K]	(Kill line)	Radera resten av raden

Både kommandot **[C-D]** och **[RUBOUT]** raderar ett tecken, men de har lite olika användningsområden. **[RUBOUT]** används främst när man skrivit fel då man skriver in ny text. Antag igen att bufferten innehåller texten

Woo~~z~~le

och arbetspunkten är placerad före z. Efter ett **[RUBOUT]** blir resultatet

Woo~~z~~le

men om du använde **[C-D]** istället för **[RUBOUT]** blir det istället

Woo~~l~~e

[C-K] används för att ta bort hela rader. **[C-K]** tar bort texten mellan arbetspunkten och nästa radbyte. Själva radbytet lämnas dock kvar. Om arbetspunkten står omedelbart före ett radbyte så tas radbytet bort istället. Om bufferten innehåller texten

Woo~~z~~le
Wizzle

med arbetspunkten före z på första raden så gör ett **[C-K]** att resultatet blir

Woo~~z~~
Wizzle

med arbetspunkten efter ordet Woo. Ytterligare ett **[C-K]** ger resultatet

Woo~~W~~izzle

eftersom du nu tog bort radbytet.

När man ska ta bort en hel rad placerar man sig i början av raden och trycker på **[C-K]** två gånger. Första gången för att ta bort texten fram till radbytet och andra gången för att ta bort själva radbytet.

Se vidare i kapitel 11 om hur man kan ta bort text på andra sätt och hur man gör för att flytta och kopiera textavsnitt.

Filer

Vi har nu kommit så långt att du hjälpligt kan redigera texten i bufferten, men det är ganska meningslöst så länge du inte kan spara resultatet på en fil. Filhanteringskommandon använder man inte så ofta att de har fått några egna CTRL- eller META-tecken. Istället används en följd av två tecken där det första alltid är **[C-X]** (X:et står för *eXtended command*).

C-X	C-S
C-X	C-V

(Save file)
(Visit file)

Skriv ut texten i bufferten på filen
Läs in en annan fil i bufferten

C-X C-S gör att texten i bufferten skrivs ut igen på filen. Använd detta kommando innan du lämnar Emacs samt då och då under ett längre redigeringspass för att inte förlora resultatet vid ett eventuellt maskinfel.

C-X C-V gör att texten från en fil läses in i bufferten så att man kan börja redigera den. Emacs frågar efter namnet på filen. Skriv det och avsluta med **RETURN**. Detta kommando utförs automatiskt när du startar Emacs genom kommandot

@EMACS (FILE) *filnamn*

För att skapa en ny fil gör du precis som du hade gjort om du hade redigerat en redan existerande. Börja alltså med **C-X C-V** eller skriv det nya filnamnet i EMACS-kommandot för att försöka läsa in filen. Om filen inte fanns så får du meddelandet (New File) på ekraden. När du sedan sparar filen på vanligt sätt med **C-X C-S** så skapas filen automatiskt.

Detta var de vanligaste kommandona som har med filer att göra. Se vidare i kapitel 17.

Att lämna Emacs

Man kan lämna Emacs på flera olika sätt som beskrivs i kapitel 7. Det bästa för normala situationer är att använda kommandot **C-X C-Z**. Glöm inte att spara bufferten på fil innan du går ut! Den normala kommandosekvensen för att spara bufferten och lämna Emacs är alltså **C-X C-S C-X C-Z**.

Att ta sig ur besvärliga situationer

Ibland kan man råka hamna i situationer där man inte vet vad man ska göra och ibland ångrar man sig när man har påbörjat ett kommando. Standardmetoden för att avbryta i sådana lägen är att trycka på **C-G** en eller flera gånger. Detta gör att Emacs avbryter det som pågår och återgår till att tolka kommandon som vanligt igen.

C-G kan inte ta dig ur alla situationer som kan inträffa. I kapitel 7 beskrivs vilka alternativ som finns och hur du kan avgöra i vilken situation du befinner dig.

Ett vanligt misstag är att man råkar trycka på **ESCAPE** två gånger. Man har då givit kommandot **M-ESCAPE** som innebär att man skapar en s k minibuffert där man kan skriva TECO-kommandon, något som den vanlige Emacs-användaren inte behärskar.²

² Kommandot **M-ESCAPE** finns inte i Amis.

Avbryt med ett par `[C-G]`.

Hjälpkommandot

Som nämnts tidigare finns det stora möjligheter att få hjälp i Emacs. Här ska vi bara titta på två varianter. Resten beskrivs i kapitel 8.

Det finns ett speciellt hjälptecken i Emacs, nämligen `[C-]`. Tyvärr måste man på vissa terminaler använda andra tangenter för att generera hjälptecknet. Om du har problem med `[C-]` så prova istället `[C-?]` och `[C-/]`.

`[C-]` ger olika hjälp beroende på situationen du befinner dig i. Du kan därför använda `[C-]` när du får frågor du inte förstår eller då Emacs verkar reagera konstigt för dina kommandon. Du får då oftast en förklaring av vad Emacs förväntar sig.

I det normala läget skriver Emacs på ekoraden

```
Doc (? for help):
```

när du trycker på `[C-]`. Du ska då skriva ett tecken till för att specificera vilken sorts hjälp du vill ha. `[?]` eller `[C-]` ger en lista över alternativen.

Här ska bara nämnas varianten `[C-][C]` (C står för Character) som ger information om hur ett kommando fungerar. Du skriver kommandot som ska beskrivas omedelbart efter `[C-][C]`. Exempelvis `[C-][C][C-F]` beskriver kommandot `[C-F]`. Informationen får du som överskrivningsmeddelande. Tryck på mellanslag när du läst färdigt.

Använd blanka rader

När man ska stoppa in några nya rader mitt i en existerande text kan man göra på flera olika sätt. Antag att vi har texten

```
Wizzle
Woozle
```

och vill stoppa in raden

```
Pooh
```

mellan dessa båda rader. Ett bra sätt är då att först stoppa in en blank rad med kommandot `[C-O]` och sedan fylla i texten på raden. `[C-O]` stoppar in ett radbyte efter arbetspunkten och öppnar därför en ny blank rad om man står först på en rad ("O" står förresten för "Open Line").

Om du placerar arbetspunkten före ordet `Woozle` och trycker på `[C-O]` så blir resul-

tatet

Wizzle



Woozle

med arbetspunkten kvar på den nu tomma raden. Därefter skriver du in texten Pooh på den nya raden med resultatet

Wizzle

Pooh

Woozle

Man kan också använda `[C-U][C-U][C-O]` för att stoppa in många blanka rader (beskrivs i kapitel 5). Du behöver inte vara rädd för att stoppa in för många blanka rader. Kommandot `[C-X][C-O]` gör det lätt att ta bort onödiga blanka rader. `[C-X][C-O]` tar bort alla tomma rader kring arbetspunkten, utom den där du står.

4

Starta Emacs

I detta kapitel ska vi gå igenom några olika sätt att starta Emacs. Det som beskrivs gäller inte säkert för alla datorer där Emacs används, utan är anpassat efter hur Emacs fungerar på Vera. När det gäller Amis är det ännu mer varierande hur man bär sig åt. Sista avsnittet i detta kapitel diskuterar kort hur det fungerar för Amis på några olika system.

Terminaltypen

Emacs är ett program som använder många avancerade egenskaper hos terminalen. Därför måste Emacs veta vilken terminal du använder. Det är därför extra viktigt att du angivit rätt terminaltyp med kommandot

```
@TERMINAL TYPE terminaltyp
```

För vissa terminaltyper sätts detta automatiskt rätt när du loggar in. Om du råkat starta Emacs med felaktig terminaltyp så måste du lämna Emacs, sätta terminaltypen och återstarta med START (se nedan om återstart).

Första redigeringen

Vi skiljer på om du startar Emacs för första gången under ett terminalpass, eller om du går tillbaka till Emacs. Anledningen är att första gången Emacs startas måste en hel del initieringar göras. När du lämnat Emacs för att köra något annat program har du sedan möjlighet att snabbt återstarta Emacs. En återstart av Emacs går mycket fortare och drar mindre resurser än en total omstart.

För att Emacs inte ska förstöras när du kör ett annat program emellan krävs det att Emacs sparas i en speciell gren (engelska: Fork). På Vera sker detta automatiskt om du använder kommandot EMACS.

Man startar Emacs med kommandot

```
@EMACS
```

Nästan varje gång man går in i Emacs vill man också läsa in en fil i bufferten. Därför kan man skriva ett filnamn som

@EMACS (FILE) *filnamn*

Emacs gör automatiskt "Visit File" på den angivna filen, d v s om filen finns läses den in i bufferten och om den inte finns initieras bufferten för att skapa den.

Återstart av Emacs

För att återstarta Emacs gör du precis som när du startar Emacs första gången, d v s använder kommandot EMACS. Förutsatt att du lämnade Emacs med `[C-X][C-Z]` eller `[C-M-Z]` kommer du in i Emacs precis som du lämnade den.

Om du inte anger något filnamn efter kommandot när du återstartar Emacs fortsätter du att redigera den buffert du har kvar från förra gången. Observera att om du har ändrat filen utanför Emacs måste du läsa in den nya versionen av filen till bufferten.

Om du anger ett filnamn när du återstartar Emacs kommer den filen automatiskt att läsas in i bufferten. Om du glömde att spara bufferten förra gången upptäcker Emacs det och ger dig en sista chans att spara bufferten innan den nya filen läses in.

Problem vid återstart

Ibland kan det vara problem att återstarta Emacs. Det kan t ex bero på att du inte lämnade Emacs på normalt sätt eller att terminaltypen inte var riktigt satt. Om du får problem kan du först prova med att gå ut ur Emacs och ge kommandot

@START

Det innebär att Emacs gör vissa initieringar, men du har ändå kvar innehållet i bufferten.

Om inte START hjälper kan du ta till grövre medel. Om du istället ger kommandot

@RESET (FORK) EMACS

så försvinner grenen med Emacs. Därefter kan du som vanligt starta Emacs från början igen.

Kommandot EDIT, ett alternativ

Det finns ett alternativt kommando för att starta Emacs, nämligen

@EDIT

men för att EDIT ska starta Emacs måste du ha definierat det logiska namnet `Editor:`. Det gör du med kommandot

```
@DEFINE (LOGICAL NAME) EDITOR: (AS) EMACS:EMACS
```

Denna definition kan du stoppa in i din `LOGIN.CMD`-fil så att namnet automatiskt definieras varje gång du loggar in.

EDIT-kommandot kan också användas för att starta andra editorer genom att man definierar `Editor:` till något annat. Annars fungerar kommandot precis som EMACS-kommandot.

Att definiera `Editor:` får också en del andra effekter. I vissa program, t ex MM (brevsystemet på Vera), finns ett lokalt EDIT-kommando. Detta kommando startar då den editor man valt genom att definiera `Editor:`.

Starta Amis

De flesta system där Amis används saknar möjligheten att återstarta samma Amis efter det att man kört något annat program. Det innebär att du alltid måste komma ihåg att spara bufferten innan du lämnar Amis. Annars är den hjälplöst förlorad.

På nästan alla datorer där Amis finns startas den med kommandot

```
AMIS filnamn
```

där *filnamn* är namnet på den fil som ska läsas in i bufferten. Amis måste också informeras om vilken terminaltyp du använder, och det sker på olika sätt beroende på vilket datorsystem du använder.

Här är några råd för speciella systemtyper:

Under operativsystemet **TOPS-10** (t ex på datorerna Nadjä och Oden) använder Amis den terminaltyp du satt med monitorkommandot:

```
.TERMINAL TYPE terminaltyp
```

Om inte kommandot AMIS fungerar kan du normalt starta Amis med

```
.R PUB:AMIS
```

eller

```
.EDIT filnamn
```

om du har en rad i din `SWITCH.INI`-fil med innehållet:


```
EDIT /RUN:PUB:AMIS
```

Så länge inget annat program har körts kan Amis återstartas med kommandot `START`.

På **Prime**-system måste terminaltypen sättas med kommandot

```
TERMDEF
```

innan man startar Amis.

På **VAX**-system frågar Amis efter terminaltypen första gången.

5

Numeriskt argument

Till alla Emacs-kommandon kan man ge ett numeriskt argument. Detta argument tolkas olika beroende på vilket kommando det gäller.

Många kommandon uppfattar argumentet som en repetition av kommandot. T ex kommandot `[C-F]` (gå framåt ett tecken) med argumentet 17 betyder "gå framåt 17 tecken".

Flera kommandon bryr sig bara om ifall man givit något argument alls. T ex kommandot `[M-Q]` gör ombrytning av ett stycke så att raderna fylls ut maximalt. Om man ger ett argument till kommandot får man dessutom justering till rak högermarginal. Här spelar det alltså ingen roll vilket värde argumentet har.

Andra kommandon behandlar argumentet på speciella sätt. T ex `[C-K]` som med ett argument tar bort så många rader som argumentet anger. Utan argument krävs ju två `[C-K]` för att ta bort en icke-blank rad. Ett annat exempel är kommandot `[C-V]`. Utan argument betyder det att fönstret flyttas en skärmsida framåt i bufferten. Med ett argument så anger det hur många rader fönstret ska flyttas.

Det grundläggande sättet att ge ett numeriskt argument är med kommandot `[C-U]` följt av de siffror som utgör argumentet. Exempelvis `[C-U]17[C-F]` utför `[C-F]` med argumentet 17. Även negativt argument är tillåtet och då skriver man ett minustecken före siffrorna.

Om man inte skriver några siffror alls så antas värdet 4. Detta är användbart för att t ex få några stycken blanka rader (`[C-U][C-O]`), gå ner några rader (`[C-U][C-N]`) o s v. Det är också bekvämt för de kommandon som bara bryr sig om ifall man givit något argument överhuvudtaget.

Argument till `[C-U]` är multiplikativt så att ett ev argument före `[C-U]` multipliceras med argumentet i själva `[C-U]` kommandot. Det innebär också att `[C-U][C-U]` betyder 16, `[C-U][C-U][C-U]` ger 64, o s v.

Lämpliga användningar är t ex

[C-U]	[C-U]	[C-F]
[C-U]	[C-U]	[C-B]
[C-U]	[C-U]	[C-N]
[C-U]	[C-U]	[C-U][C-O]

gå en bit framåt längs raden (16 tecken),
gå en bit bakåt längs raden (16 tecken),
gå ner en knapp skärmsida (16 rader),
stoppa in många blanka rader (64 st)

6

Hur man skriver kommandon

Det finns flera olika kategorier av kommandon till Emacs. Vi har redan stött på CTRL- och META-kommandon och några andra varianter. Nu ska vi mer systematiskt gå igenom vilka principer som gäller.

CTRL och META

Terminaler som är ideala för Emacs har både en CTRL- och en META-tangent och dessa kan kombineras med alla andra tangenter. Med dessa tangenter kan vi åstadkomma kommandon som `C-A` (`CTRL` nere när man trycker på `A`), `M-A` (`META` nere), `C-M-A` (både `CTRL` och `META` nere) och vanligt `A` (varken `CTRL` eller `META` nere).

På vanliga terminaler så saknas `META`-tangent och `CTRL` kan inte kombineras med alla tecken. Därför kan man istället använda speciella prefixtecken som kan ersätta `CTRL`- och `META`-tangenterna.

Tecknet `C-^` fungerar som `CTRL`-prefix och kan användas då man vill ge CTRL-kommandon som inte kan ges på ett vanligt tangentbord, t ex `C-RETURN`. Dessa kommandon är dock ganska ovanliga.

Tecknet `ESC`, som har en egen tangent på alla vanliga tangentbord, fungerar som `META`-prefix. Detta används mycket ofta. Kommandot `M-A` ger man alltså genom att först trycka på `ESC` och sedan på `A`.

Tecknet `C-Z` fungerar som CTRL-META-prefix. Om man t ex ska ge kommandot `C-M-A` så ger man först `C-Z` och därefter `A`.

Utvidgning till fler kommandon

Kombinationerna `C-`, `M-` och `C-M-` ger många kommandon, men det är ändå inte på långa vägar tillräckligt för att rymma alla Emacs-kommandon. Man har därför valt att låta `C-X` och `M-X` vara inledningar till flerteckenskommandon. Tänk på "X" som eXtend (utvidga till fler kommandon).

`C-X` ska följas av ytterligare ett tecken. `M-X` ska följas av en textrad som avslutas med `RETURN`. Mer om detta i nästa avsnitt.

Varje grupp av kommandon har vissa gemensamma drag. Det gör det mycket lättare att komma ihåg dem. CTRL-kommandon är kommandon som man använder ofta. META-kommandon är ofta en kraftigare variant av motsvarande CTRL-kommando. Exempelvis betyder [C-F] ”gå ett tecken framåt” och [M-F] ”gå ett ord framåt”. CTRL-META-kommandon är mer specialiserade och används ofta för programspråksanpassade operationer. [C-X]-gruppen är kommandon som man bara ger några gånger, bl a de som har med filhantering att göra.

Namngivna kommandon

Alla kommandon har ett namn. Namnen brukar bestå av några ord som mycket kort beskriver vad kommandot gör. Typiska exempel är `Replace String`, `Edit Tab Stops` och `Save All Files`.

Som du redan vet använder man [C-], [M-], [C-M-] eller [C-X] kombinationer för att ge vanliga kommandon. Detta kallas att kommandot ”sitter på en tangent”. Även dessa kommandon har alltså namn, men man behöver nästan aldrig använda dem. Det finns också många kommandon som inte ”sitter på någon tangent”, d v s som inte kan skrivas med [C-], [M-], [C-M-] eller [C-X] kombinationer. Dessa kan man bara ge genom att skriva deras namn. Det är framför allt dessa som vi är intresserade av här.

Med [M-X] kan man skriva hela namnet på ett kommando som man vill ha utfört. Efter [M-X] skriver man namnet och avslutar med [RETURN]. Det är tillåtet att förkorta kommandonamnen ord för ord så länge de är entydiga. Om man förkortar orden fyller Emacs automatiskt i det man hoppat över när man trycker på mellanslag. Om man trycker på [ESCAPE] försöker Emacs fylla i hela kommandonamnet.

Kommandon som man normalt inte skriver som namngivna kommandon har namn som inleds med ^R. Anledningen till detta är att man inte vill att de i onödan ska kollidera med de vanliga namnen när man förkortar dem. Namn med ^R i början ser man då och då i hjälpinformation och dokumentation.

Till ett namngivet kommando kan man också ge ett eller flera textargument. Man kan antingen skriva argumenten direkt efter kommandonamnet eller låta Emacs fråga efter dem. När man skriver argumenten efter kommandonamnet gör man det innan man trycker på [RETURN]. Efter kommandonamnet och mellan argumenten trycker man på [ESCAPE]. [ESCAPE]-tecknet skrivs som \$ både i dokumentationen och på ekoraden när man ger kommandot. Om man inte skriver några argument efter kommandonamnet ställer Emacs frågor på ekoraderna efter de argument som erfordras.¹

Ett exempel på namngivet kommando som tar textargument är `Replace String`

1 Detta är en lokal modifiering av Emacs vid Vera. Utan denna måste man använda C-M-X istället för M-X för att få frågor om argumenten.

som byter en text mot en annan. Om man ger detta kommando som namngivet kommando och skriver argumenten direkt ser det ut som

```
(M-X) Replace String$woozle$wizzle
```

där \$-tecknen fylls i automatiskt när du trycker på `ESCAPE`. Om du istället låter Emacs fråga efter argumenten skriver du bara

```
(M-X) Replace String
```

och sedan frågar Emacs efter först

```
Replace:
```

där du svarar `woozle` och sedan

```
Replace woozle with:
```

där du svarar `wizzle`. Båda svaren avslutar du med `RETURN`.

7

Hur man tar sig ut

I detta kapitel ska vi beskriva hur man tar sig ut ur olika situationer. Det omfattar både att lämna Emacs och att gå ut ur olika delar av Emacs inre gömmor.

Lämna Emacs med C-X C-Z

När du vill lämna Emacs och återgå till det program som startade Emacs (normalt EXEC) använder du kommandot `[C-X][C-Z]`. Glöm inte att spara texten i bufferten på fil innan du lämnar Emacs!

Du kan också använda `[C-M-Z]` som tar dig ur den nivå av Emacs där du är. Normalt innebär det att du lämnar Emacs. Skillnaden beskrivs på nästa sida. `[C-M-Z]` skrivs enklast på terminalen som `[C-Z][Z]`.

Avbryt med C-G

När du vill avbryta Emacs använder du `[C-G]`. Exempel på användning är:

- Du har tryckt på `[ESCAPE]` men ångrar dig och vill inte ge något META-kommando.
- Du ger ett sökkommando men ångrar dig.
- Du råkade trycka på `[ESCAPE]` två gånger och hamnade i något konstigt läge (se nedan).

Om du råkat ge ett kommando som förstörde väldigt mycket, t ex tog bort all text, kan du ofta få det gjort om du omedelbart efteråt ger kommandot

`[M-X]` Undo

Läs på modraden

När Emacs inte bär sig åt som vanligt bör du titta på modraden för att se om du hamnat i något specialsystem. När modraden inleds med ordet EMACS är du inne i Emacs som vanligt. Om det istället står något annat ord som t ex INFO eller Dired har du hamnat i ett undersystem till Emacs. De flesta av dessa system kan man lämna med kommandot `[Q]`.

Om modraden omges av parenteser som t ex

(Minibuffer)

eller

(Query Replace)

är du mitt inne i något kommando. Använd `[C-]` för att få hjälp eller `[C-G]` för att avbryta.

Om modraden omges med `Ä` och `Å` t ex

ÄEdit OptionsÅ

så har något kommando anropat Emacs rekursivt. Se vidare i nästa avsnitt.

Rekursivt anrop av Emacs

Några avancerade kommandon låter dig temporärt redigera någon annan text än den du egentligen arbetar med, t ex kommandot `[M-X]` Edit Tab Stops som låter dig redigera en text som innehåller tabulatorstopparna.¹ När du ändrat tabulatorstopparna återgår du till att redigera den vanliga texten igen. Detta kallas att man gör en rekursiv redigering.

Vid en rekursiv redigering fylls skärmen med det som du temporärt ska redigera och du kan använda alla Emacs-kommandon för den ändringen. En rekursiv redigering känner man igen på att modraden inleds med `Ä` och avslutas med `Å`.

En rekursiv redigering kan man lämna på två olika sätt. `[C-M-Z]` är det normala sättet när man har ändrat färdigt. Det är alltså samma kommando som man kan använda för att lämna Emacs. Det andra sättet är med `[C-A]` som betyder att kommandot som gjorde den rekursiva redigeringen avbryts. `[C-G]` kan inte användas för att komma ut ur en rekursiv redigering.

Om man använder `[C-X][C-Z]` i en rekursiv redigering kommer man direkt ut till EXEC istället för att bara gå ur den rekursiva redigeringen.

1 Kommandot M-X Edit Tab Stops finns inte i Amis.

8

Hjälp och dokumentationskommandon

I kapitel 3 beskrev vi hur man kan använda hjälptecknet `C-h` för att få hjälp. När man är inne i någon speciell situation ger `C-h` en beskrivning av den situationen. På den s k toppnivån måste man skriva ett tecken till för att välja vilken sorts hjälp man vill ha. Emacs frågar efter det andra tecknet genom att

```
Doc (? for help):
```

skrivs på ekoraden.

I detta kapitel ska vi gå igenom de viktigaste alternativen.

Character: C-_ C

`C-h C` ger information om ett kommando. Skriv kommandot efter `C-h C` så får du se vad kommandot heter och en beskrivning av vad det gör. Exempelvis `C-h C M-F` ger en beskrivning av kommandot `M-F`.

Describe: C-_ D

`C-h D` hittar beskrivningen för ett namngivet kommando eller en Emacsvariabel. Du skriver namnet på kommandot eller variabeln när Emacs ställer frågan `Describe:` och avslutar med `RETURN`.

Apropos: C-_ A

`C-h A` använder man för att söka reda på ett visst kommando som man har glömt namnet på. Apropos-funktionen listar alla kommandon som har en viss given textsträng i sitt namn. Du skriver själv in söktexten då Emacs frågar med `Apropos:` och avslutar med `RETURN`.

Du måste välja texten så att den på lämpligt sätt avspeglar vad du söker efter. Namnen på Emacs' kommandon är oftast systematiskt valda så att ett väl valt sökord ger precis det du letar efter.

Lämpliga sökord är character, line, word, sentence, paragraph, region, page, buffer, screen, window, bounds, file, dir, beginning, end, case, mode, forward, backward,

next, previous, up, down, search, kill, delete, mark, fill, indent, change.

Ge inte upp om du inte får napp första gången. Försök igen med ett annat sökord.

Emacs Info: C- I

C- I startar informationssystemet Info.¹ Info är ett system som man kan använda för att hitta och titta på dokumentationsfiler och hjälpfiler av olika slag. Info innehåller inte bara information om Emacs, utan även beskrivningar av andra program och programsystem, bl a beskrivningar för alla EXEC-kommandon.

Emacs Info kan också anropas direkt från EXEC under namnet XINFO.

Informationen i Info är organiserad i form av noder med text och referenser mellan dessa noder. Många noder innehåller en meny med alternativ och då man väljer ett av dessa alternativ så kommer man vidare till en ny nod.

Några Info-kommandon:

M <i>menyalternativ</i> RETURN	(Menu)	Välj ett menyalternativ
G <i>nodnamn</i> RETURN	(Go)	Gå direkt till en nod
Q	(Quit)	Lämna INFO
U	(Up)	Gå upp till den nod som refererar hit i sin meny
SPACE		Bläddra framåt i nodtexten
BACKSPACE		Bläddra bakåt i nodtexten
?		Lista INFO-kommandona
D	(Directory)	Återgå till första noden (innehållsförteckningen)
F <i>notnamn</i> RETURN	(Footnote)	Följ en korsreferens via en fotnot som markeras med *Note: <i>notnamn</i> :... i texten
L	(Last)	Gå tillbaka till föregående nod

Beskrivning av Emacskommandon hittar du under huvudalternativet Emacs. Under detta alternativ finns sedan en mängd noder med både grundläggande och mycket avancerad information om Emacs.

Kommandot **H** i Info gör att du kommer till en nod i Info som beskriver hur man använder Info. Prova det någon gång!

¹ Kommandot C- I finns ännu inte i Amis

Last Commands: C-_ L

Ibland ställer man sig plötsligt frågan: "Vad gjorde jag nu?". Svaret får du om du ger kommandot `C-_L` som då listar de 60 senaste tangenttryckningarna du gjorde.² Detta är mycket användbart även för en handledare som ska försöka återställa en text du förstört med några felaktiga kommandon.

Hjälp om C-_

`C-_C-_` ger naturligtvis hjälp om vilka alternativ som finns till `C-_` kommandot.

2 C-_L finns ännu inte i Amis.

9

Flytta arbetspunkten

I detta kapitel ska vi gå igenom olika kommandon för att flytta arbetspunkten i bufferten. Man kan också använda sökkommandon som beskrivs i kapitel 12.

Flyttning mindre sträckor

Dessa kommandon kan man använda för att förflytta sig framåt och bakåt längs raden. Man kan också komma över till intilliggande rader, för radbytet betraktas som ett tecken precis som alla andra.

C-F	Framåt teckenvis
C-B	Bakåt teckenvis
M-F	Framåt ordvis
M-B	Bakåt ordvis

Här följer några kommandon som har med rader att göra. Kommandona **C-N** och **C-P** försöker gå till samma kolumn på nästa resp föregående rad, men om raden är för kort så placeras man så nära det går.

C-P	Uppåt radvis
C-N	Nedåt radvis
C-A	Till början av raden
C-E	Till slutet av raden
M-RETURN	Till radens början men efter inledande blanka

Flyttning relativt meningar och stycken

För att dessa kommandon ska fungera så krävs det att man skriver meningar och stycken som Emacs förväntar sig. En sats anses sluta med ett skiljetecken (".", ":", ";", "!", " eller "?"") följt av minst två blanka eller ett radbyte. Ett stycke anses sluta när man stöter på en tom rad eller en rad som inte börjar i första positionen på raden. När man redigerar programtext så är det bara tomma rader som bryter "stycken".

M-A	Bakåt en mening
M-E	Framåt en mening
M-Ä	Till början av stycket
M-Å	Till slutet av stycket

Flyttning större sträckor

Slutligen några kommandon för att flytta sig större sträckor i bufferten. Kommandoparet [C-V] och [M-V] är lämpligt för att bläddra i en text. Sidbyten är detsamma som tecknet ^L. Det kan du stoppa in med [C-Q][C-L].

[C-V]	Gå en skärmsida framåt
[M-V]	Gå en skärmsida bakåt
[C-X] [Å]	Till början av sidan
[C-X] [Ä]	Till slutet av sidan
[M-<]	Till början av hela texten
[M->]	Till slutet av hela texten

Om du har en stor .fil så kan det vara bättre att använda sökkommandon för att komma till något ställe mitt i.

10

Märket och området

Många kommandon opererar inte bara vid arbetspunkten utan på en större del av bufferten. I sådana fall måste man ha någon möjlighet att tala om vilken del av bufferten man menar. För att markera en del av bufferten så placerar man arbetspunkten i ena änden och något som kallas **märket** (engelska: *mark*) i den andra. Texten mellan märket och arbetspunkten kallar vi för **området** (engelska: *region*). Det spelar ingen roll om märket står i början och arbetspunkten i slutet av området eller tvärt om.

Att flytta märket

Man kan se märket som ett slags andra arbetspunkt. Ett problem med märket är att man inte kan se var det är placerat. Därför brukar man placera märket omedelbart innan det ska användas, så att man inte hinner glömma bort var det är.

Det finns många kommandon för att flytta arbetspunkten, men mycket få för att flytta märket. Man använder istället metoden att flytta arbetspunkten till den plats där man vill ha märket och ger kommandot `C-@`. Detta kommando sätter märket på samma plats som arbetspunkten befinner sig på.

`C-@`

Sätt märket vid arbetspunkten
(alternativt skrivsätt för samma sak)

`C-SPACE``M-@`

Sätt märket ett ord framåt

`C-X C-X`

Byt märket och arbetspunkten mot varandra

Att sätta området

När man ska sätta området gör man normalt så här: Först placerar man arbetspunkten i ena änden av det önskade området och ger där kommandot `C-@` för att sätta märket. Därefter flyttar man sig till andra änden och området är klart. Kommandot `C-X C-X` är trevligt för att kontrollera märkets position utan att förstöra området. Det byter ju bara plats på märket och arbetspunkten.

Det finns också några kommandon som direkt sätter området runt någonting. De flyttar alltså både arbetspunkten och märket.

`M-H`

Sätt området kring stycket

`C-X C-P`

Sätt området kring sidan

`[C-X][H]`

Sätt området kring hela bufferten

Andra användningar av märket

Vissa kommandon som flyttar arbetspunkten långt sätter märket vid startpositionen så att man enkelt ska kunna hitta tillbaka igen. Ett typiskt exempel är sökkommandona `[C-S]` och `[C-R]` som sätter märket vid startpositionen om sökningen flyttade arbetspunkten mer än 500 tecken.

Kommandon som stoppar in ett stycke text, t ex vid en flyttning eller inkopiering från en annan fil, sätter området runt den insatta texten. Det gör det mycket enkelt att plocka bort den igen om det skulle vara fel.

Gamla märken

Om man tar bort eller stoppar in text före märket så driver märket iväg. Det beror på att märket behåller sitt avstånd från buffertens början. Därför bör man alltid sätta märket strax innan det används.

När man använder märket för att hitta tillbaka i en stor fil så gör det dock inte så mycket om märket drivit iväg något. Man kommer ändå i närheten av rätt plats. Då kan det också vara intressant att hitta tillbaka till näst sista märket o s v. Det är möjligt genom att Emacs håller kvar en lista på de 16 senaste märkena. Om man ger argumentet 4 till `[C-@]` så får det en ny betydelse. Det flyttar arbetspunkten till märket, märket till föregående märke o s v. Flera `[C-U][C-@]` gör alltså att man successivt går tillbaka till äldre och äldre märken.

`[C-U][C-@]`

Gå till märket, återställ gamla märken

11

Radering och flyttning av text

Vi ska här beskriva kommandon för att ta bort och flytta text. Anledningen till att vi tar dessa funktioner tillsammans är att man till viss del använder samma kommandon.

Radering

I Emacs skiljer man mellan två sorters raderingskommandon. Den ena är de kommandon som raderar enstaka tecken, tar bort blanka rader, följer av blanktecken o s v. Den andra är de kommandon som tar bort hela ord, rader, meningar eller andra större områden. På engelska kallas den första gruppen för *Delete* och den andra för *Kill*. Text som raderats med ett *Delete*-kommando är borta för alltid. En text som däremot raderats med ett *Kill*-kommando sparas i en speciell buffert, raderbufferten. Denna text kan sedan sättas tillbaka i texten, antingen på den ursprungliga platsen eller någon annanstans, eller båda. Detta är det enklaste sättet att flytta och kopiera text.

Delete-kommandon

Dessa kommandon tar bort text utan att spara det borttagna i raderbufferten.

C-D
RUBOUT
M-Ö
C-X C-O
M-^

Radera nästa tecken
Radera föregående tecken
Radera blanka (och Tab-tecken) runt arbetspunkten
Radera tomma rader runt aktuell rad
Sätt ihop raden med raden före

Kill-kommandon

Dessa kommandon tar bort text och sparar samtidigt det borttagna i raderbufferten. Om man ger flera *Kill*-kommandon i omedelbar följd så sparas inte bara det sista, utan hela det borttagna tillsammans.

C-K
C-W
M-D
M-RUBOUT

Tag bort till radslut
Tag bort området
Tag bort nästa ord
Tag bort föregående ord

C-X **RUBOUT**
M-K

Tag bort från meningens början
 Tag bort till meningens slut

Det nomala sättet att ta bort en rad är att man placerar sig i radens början och ger kommandot **C-K**. Raden blir då tom och man kan med ett extra **C-K** ta bort även radbytet. När man ska ta bort flera rader kan man antingen trycka på **C-K** flera gånger i följd eller ge ett numeriskt argument till **C-K** som då anger hur många rader som ska tas bort.

Man kan också använda **C-W** för att ta bort större eller mindre delar av texten. Man sätter då först området kring den aktuella texten och ger sedan **C-W**.

När du ska ta bort ett eller flera ord använder du **M-D** eller **M-RUBOUT**. Kommandot **M-RUBOUT** passar bra för att ta bort ett ord som man just skrivit in. Om man börjat skriva en mening men ångrar sig så kan **C-X** **RUBOUT** vara lämpligt för att ta bort det man skrev.

Flytta och kopiera text

För att flytta en text använder man först något av ovanstående *Kill*-kommandon för att ta bort texten och sedan **C-Y** för att återinsätta den på den nya platsen. Metoden brukar kallas för "Klippa och klistra" (engelska: *Cut and Paste*). På detta sätt kan man alltså flytta ord, rader, meningar och godtyckliga områden.

C-Y
M-Y
M-W
C-M-W

Sätt in senast borttagna text vid arbetspunkten
 Byt återskapad text mot ännu äldre
 Kopiera området till raderbufferten
 Låt nästa *Kill*-kommando lägga till i raderbufferten

För att kopiera någonting gör man precis som vid flyttning, men man sätter dessutom tillbaka det borttagna där det ursprungligen fanns. Detta är så vanligt att man har gjort ett speciellt kommando för det. Med **M-W** läggs innehållet i området in i raderbufferten utan att tas bort. Det fungerar alltså som ett **C-W** omedelbart följt av ett **C-Y**. Med **C-Y** kan man sedan sätta in denna text på en eller flera andra platser.

Vi har tidigare sagt att Emacs sparar den sist borttagna texten i raderbufferten. Egentligen sparas de 8 senast borttagna texterna. Om du använder **C-Y** för att sätta tillbaka en borttagen text och upptäcker att du egentligen skulle ha en äldre borttagning så ger du kommandot **M-Y**. Den nyss insatta texten ersätts då med den äldre. **M-Y** ska bara ges på detta sätt efter **C-Y**. Ytterligare ett **M-Y** ger en ännu äldre borttagen text o s v.

Q-register

Nackdelen med raderbufferten är att den bara finns kvar ett tag efter det att raderin-

gen gjordes. Q-register är platser där man kan spara undan textdelar under lite längre tid. Beteckningen Q-register har sina rötter i kommandot "Q" i texteditorn TECO med ungefär samma funktion. Det finns flera Q-register med olika namn bestående av en bokstav A-Z.

$\begin{bmatrix} \text{C-X} & \text{X} \\ \text{C-X} & \text{G} \end{bmatrix} \alpha$

Kopiera området till Q-register α
Sätt in texten från Q-register α

Med $\begin{bmatrix} \text{C-X} & \text{X} \\ \text{C-X} & \text{G} \end{bmatrix} \alpha$, där α är någon av bokstäverna A till Z, kopieras texten i området in i Q-registret. För att sedan använda texten man sparar ger man kommandot $\begin{bmatrix} \text{C-X} & \text{G} \end{bmatrix}$ α som då sätter in det sparade vid arbetspunkten.

Byta-plats-kommandon

$\begin{bmatrix} \text{C-T} \\ \text{M-T} \\ \text{C-X} & \text{C-T} \\ \text{C-X} & \text{T} \end{bmatrix}$

Byt plats på två tecken
Byt plats på två ord
Byt plats på två rader
Byt plats på området och gamla området

För att använda dessa kommandon placerar man sig mellan de saker som ska bytas och ger bytkommandot. För rader gäller att man byter raden där man står mot föregående rad. Efter bytet placeras arbetspunkten före nästa tecken, ord resp rad. Det gör att om man upprepar kommandot så "drar man" tecknet, ordet resp raden med sig framåt.

Kommandot $\begin{bmatrix} \text{C-X} & \text{T} \end{bmatrix}$ fordrar en extra förklaring. Detta kommando används för att byta texten i två angivna områden med varandra. För att markera två olika områden används dels det vanliga området och dels området mellan förra märket och förrförra märket. Antag att vi vill byta texten mellan två punkter i bufferten som vi kallar *a* och *b* med texten mellan två andra punkter *c* och *d*. Vi går då först till *a* och sätter märket, sen till *b* och sätter märket, sen till *c* och sätter märket igen och slutligen till *d* och ger kommandot $\begin{bmatrix} \text{C-X} & \text{T} \end{bmatrix}$.

12

Sökning

När texten du arbetar med börjar bli stor så blir det svårare att hitta till rätt plats. Emacs har kommandon för sökning både framåt och bakåt som gör det möjligt att snabbt hitta rätt.

Stegvis sökning

C-S
C-R

Stegvis sökning framåt
Stegvis sökning bakåt

Kommandot `[C-S]` startar en stegvis sökning framåt. `[C-R]` gör att man istället söker bakåt. När man startat sökningen så har man dels möjlighet att skriva in en text att söka efter och dels att ge underkommandon för att styra sökningen.

Samtidigt som du skriver in söktext och ger underkommandon kan du i fönstret se vart arbetspunkten kommer att flyttats av sökningen. Denna sorts sökning kallar vi *stegvis sökning* (engelska: *Incremental Search*). När du hittat rätt plats trycker du på `[ESCAPE]`-tangenter för att avsluta sökningen. Arbetspunkten flyttas då till den nya platsen.

På ekoraden står texten `I-Search: resp Reverse I-Search:` medan sökningen pågår. Efter kolonet står den aktuella söksträngen, d v s den text man söker efter. Om du skriver in text så stoppas den in i söksträngen och samtidigt flyttas arbetspunkten fram till första förekomsten av strängen.

Underkommandon

Dessa kommandon kan man ge medan den stegvisa sökningen pågår:

C-S
C-R
RUBOUT
ESCAPE

Sök efter nästa förekomst framåt
Sök efter nästa förekomst bakåt
Radera sist skrivna tecknet
Avsluta sökningen

Om man vid sökning framåt inte kommer till rätt plats så kan man antingen skriva in en lite längre söksträng eller trycka på `[C-S]` en eller flera gånger för att gå vidare till nästa förekomst av samma sträng. Ifall man inte hittade det man sökte vid sökning framåt så kan man använda `[C-R]` för att vända och söka bakåt efter samma sträng.

Ofta vill man söka efter samma sträng som man använde vid föregående sökning. Genom att trycka på ett extra **[C-S]** (resp **[C-R]**) innan du skrivit in någon söksträng så får du direkt tillbaka den sträng som användes förra gången.

Om sökningen flyttade arbetspunkten mer än 500 tecken så sätts automatiskt märket vid startpositionen. För att meddela detta så skrivs texten `^^@` på ekoraden. Detta märke kan man använda om man vill återvända strax efteråt.

När man inte hittar strängen

Om Emacs inte hittar strängen du söker efter så skrivs `Failing I-Search:` respektive `Failing Reverse I-Search:` på ekoraden. Är du nöjd med den plats du kommit till ändå så kan du avsluta sökningen med **[ESCAPE]**, trots att du inte kommit ända fram. Om anledningen till att strängen inte finns är att du råkat skriva fel så går det bra att radera de sist inskrivna tecknen med **[RUBOUT]**.

Om en sökning framåt misslyckas så kan det bero på att det du letade efter egentligen fanns tidigare i texten. Du kan då helt enkelt vända sökriktningen med **[C-R]** för att söka efter samma sträng bakåt.

Om du avbryter sökningen med **[C-G]** så återställs allting som det var före sökningens början. Det betyder bl a att arbetspunkten flyttas tillbaka till startläget.

13

Utbyte av text

Ibland har man behov av att byta ut en textsträng mot en annan överallt. I andra fall vill man inte byta riktigt överallt utan bara på vissa ställen. För båda operationerna kan man använda kommandot för selektivt utbyte (engelska: *Query Replace*).

M-%

Selektivt utbyte

När du givit detta kommando får du två frågor. Dels efter strängen som ska bytas ut och dels efter den som ska stoppas in stället.

Emacs söker från den aktuella arbetspunkten till buffertens slut. Om strängen finns så flyttas arbetspunkten dit och du får med en enteckenskod tala om vad som ska göras. Dina svar bör vara något av:

SPACE
RUBOUT
ESCAPE
.
|

Byt och sök vidare efter nästa förekomst

Byt inte här, men sök vidare

Sluta det selektiva utbytet

Byt här och sluta sen

Byt här och i resten av bufferten också

Det finns ytterligare några alternativ som vi inte tar upp här.

Oftast placerar man arbetspunkten i buffertens början innan utbytet börjar. På så sätt byter man i hela texten. Medan selektiva utbytet pågår så trycker man på mellan-slagstangenten för att byta och **RUBOUT** för att låta bli. Om man vill byta överallt, utan att få en fråga för varje ställe, så svarar man med ett utropstecken.

Om du skrivit både söksträngen och utbytessträngen med enbart små bokstäver så försöker Emacs bevara det utbytta ordets form: små bokstäver, stora bokstäver eller stor begynnelsebokstav.

14

Huvudtillstånd, undertillstånd och bibliotek

I detta kapitel ska vi se på hur Emacs med enkla medel kan anpassas för olika typer av text. Vilken huvudtyp av text som Emacs är anpassad för för tillfället anges med huvudtillståndet. En användare har dessutom möjlighet att slå på och av speciella funktioner som kallas undertillstånd.

Detaljegenskaper av olika slag kan ofta styras genom att man ändrar värdet på s k variabler. Detta beskrivs i kapitel 19.¹

Genom att ladda speciella bibliotek kan man få tillgång till helt nya kommandon med nya egenskaper. I slutet på detta kapitel går vi igenom några av de bibliotek som finns tillgängliga.²

Huvudtillstånd

När man redigerar texten i en buffert så befinner sig Emacs i något huvudtillstånd (engelska: *Major Mode*). Vilket huvudtillstånd som gäller ser man som det första ordet i parentes på modraden. Normalt väljer Emacs huvudtillstånd efter filtypen.

Exempel på huvudtillstånd är Fundamental, Text, Simula³, Pascal, Lisp och Macro. Fundamental är ett grundtillstånd utan några speciella finesser. Text passar bra för redigering av vanlig text. Tillstånden för olika programspråk gör att Emacs anpassas till hur man brukar skriva program.

För att växla till ett annat huvudtillstånd ger man ett namngivet kommando bestående av tillståndets namn följt av ordet Mode. Exempel:

```
M-X Text Mode
M-X Simula Mode
```

Man kan också få Emacs att automatiskt sätta önskat huvudtillstånd varje gång man läser in en viss fil, trots att Emacs inte känner igen filtypen. Det gör man genom att någonstans på första raden ha en text på formen `--tillstånd--*, t ex --Text--`

1 Variabler finns inte i Amis.

2 I Amis kan man inte ladda in några bibliotek.

3 Simula Mode finns inte i Amis. Använd Algol Mode istället.

* -

Texttillstånd

I texttillstånd är Emacs anpassad för redigering av engelsk text. Det innebär bl a att kommandon som behandlar ord betraktar A-Z som bokstäver. På nästa sida beskrivs hur man kan få även Å, Ä och Ö att vara bokstäver.

Indragna rader betraktas som början på ett nytt stycke. `[TAB]`-tangentsen används för att gå till nästa tabulatorstopp. Tabulatorstopparna kan flyttas med hjälp av kommandot

```
M-X Edit Tab stops
```

Texttillstånd använder man normalt för all text som inte är programtext av något slag. Se vidare i kapitel 15 om kommandon som är speciellt lämpade för text.

Programspråkstillstånd

I de olika programspråkstillstånden är Emacs anpassad för redigering av programtext skriven i just det språket. Det innebär bl a att de tecken som är tillåtna i identifierare i språket behandlas korrekt.

Det finns några kommandon för att sätta in och ta bort kommentarer. Dessa kommandon sätter automatiskt in kommentarer på rätt form för det aktuella språket. Kommandona är gjorda för att man ska skriva kommentarer i slutet på rader, prydligt under varandra.

<code>[M-;]</code>	Sätt in kommentar efter raden
<code>[C-M-;]</code>	Tag bort kommentar
<code>[C-X] [;]</code>	Sätt önskad kolumn för kommentarer
<code>[M-N]</code>	Sätt in kommentar efter nästa rad
<code>[M-P]</code>	Sätt in kommentar efter föregående rad
<code>[M-LINEFEED]</code>	Fortsätt kommentar på nästa rad

En annan speciell egenskap hos programtext är att man ofta vill ha en mycket speciell indragning vid vänsterkanten (engelska: *Indentation*). `[TAB]`-kommandot försöker dra in raden lagom mycket. `[LINEFEED]` avslutar en rad och drar in nästa.

För att få reda på vilka speciella egenskaper ett tillstånd har så kan du använda `[C-] [D]`-kommandot (Describe). Exempel:

```
[C-] [D] Text Mode [RETURN]
```

Undertillstånd

Undertillstånd (engelska: *Minor Mode*) används för att slå på eller av funktioner som man kan vilja ha oberoende av vilket huvudtillstånd man använder. Även undertillstånden kan man avläsa på modraden, efter namnet på huvudtillståndet. För att slå på eller slå av ett undertillstånd så skriver man tillståndets namn som ett namngivet kommando. Exempel:

☐ M-X Swedish Mode

☐ M-X Auto Fill Mode

Svensk text

Vid arbete med svensk text kan det vara lämpligt att slå på "Swedish Mode"-tillståndet. Det medför att tecknen Å, Ä och Ö betraktas som bokstäver i kommandon som behandlar ord o dyl och att små och stora bokstäver behandlas korrekt även för dessa tecken, t ex vid sökning.

Överskrivning

Om man slår på "Overwrite Mode" så innebär det att de tecken man skriver inte skjuts in vid arbetspunkten, utan skriver över de som redan stod där. Detta tillstånd kan vara användbart när man har en text med en speciell längd och vill ersätta den med en annan av samma längd. Glöm inte att slå av detta tillstånd igen, för det är lätt att råka skriva över något som man egentligen ville ha kvar.

Automatiskt radbrott

När man skriver in löpande text kan det vara bekvämt att slå på "Auto Fill Mode". Detta innebär att när man skriver in ord som hamnar utanför högermarginalen så flyttas ordet automatiskt ner till nästa rad. Om man håller på att skriva en kommentar och Emacs är i lämpligt programspråktillstånd så försöker Emacs se till att kommentaren fortsätter på rätt sätt även på den nya raden.

Automatisk skyddskopia

I Emacs får man normalt själv avgöra om och när man på fil vill spara undan de ändringar man har gjort. Vid maskinfel o dyl finns bara det man sparat på fil kvar. Om man har lätt för att glömma att spara texten så kan man låta Emacs automatiskt spara buffertens innehåll på fil då och då. Detta får man genom att slå på "Auto

Save Mode”.⁴

Skyddskopiorna förstör inte originalfilen, utan skrivs ut på en fil med ett namn av typen `_SAV00.MAIN` (Main är namnet för bufferten). Du kan använda kommandot

(M-X) Revert File

för att återställa bufferten i den form som senast sparats på fil.

Bibliotek

Emacs kan anpassas för speciella önskemål genom att man programmerar upp nya funktioner och kommandon och lägger dessa i ett bibliotek.⁵ Detta kräver dock att man skriver sina program i det mycket speciella språket TECO och det är förhållandevis svårt att lära sig. Att använda ett bibliotek som redan finns är dock mycket enkelt och lyckligtvis finns det en hel del bibliotek att välja på.

För att kunna använda funktionerna i ett bibliotek måste det först laddas med kommandot

(M-X) Load Library

För att få reda på vilka funktioner som finns i ett bibliotek så kan du använda kommandot

(M-X) List Library

Några exempel på tillgängliga bibliotek:

BBlock	Hjälp för att skriva stora blockbokstäver
Chess	Hjälp för att rita schackbräden
Columns	Hantering av flerspaltig text
Dired	Paket för att titta på och ändra i sin filkatalog
Draw	Paket för att rita linjer m m
Tags	Hjälp för att hitta procedurer o dyl i flera filer
Page	Annorlunda hantering av sidindelade filer
Phrase	Rutiner för hantering av delar av meningar
Picture	Hjälpmedel för att rita figurer
Renum	Rutiner för att numrera om formler, kapitel o dyl
Slowly	Anpassning för långsamma terminaler
Sort	Diverse rutiner för sortering av rader/sidor/stycken...
System	Rutiner för information om systemet
VT100	Utnyttjar funktionstangenterna på VT100-terminal
Wordab	Automatisk expansion av förkortade ord

4 Auto Save Mode finns inte i Amis.

5 Bibliotek finns inte för Amis.

WP (Word Processing) Rutiner för ordbehandling

Dessutom finns det många paket som laddas automatiskt när man sätter på ett huvud- eller undertillstånd första gången.

15

Speciella textkommandon

I detta kapitel ska vi gå igenom några kommandon som är speciellt användbara när man arbetar med löpande text. Redan tidigare har vi beskrivit kommandon som hanterar ord, meningar och stycken, men vi repeterar dem här för fullständighetens skull.

När man redigerar text bör man använda Text Mode. För att få Emacs att fungera bra för svensk text bör man dessutom sätta på Swedish Mode. Om du arbetar med marginalställd text bör du dessutom ladda in biblioteket "WP" (Word Processing).

M-F	Framåt ordvis
M-B	Bakåt ordvis
M-D	Tag bort nästa ord
M-RUBOUT	Tag bort föregående ord
M-@	Sätt märket ett ord framåt
M-T	Byt plats på två ord
M-A	Bakåt en mening
M-E	Framåt en mening
M-K	Tag bort till meningens slut
C-X RUBOUT	Tag bort från meningens början
M-A	Till början av stycket
M-A	Till slutet av stycket
M-H	Sätt området kring stycket

Byte mellan gemena och versaler

När man ändrar i en text händer det ofta att man vill ersätta ett ord skrivet med stor begynnelsebokstav med samma ord men med små bokstäver eller vice versa. För sådana ändringar är dessa kommandon smidiga:

M-U	Ändra nästa ord till stora bokstäver
M-L	Ändra nästa ord till små bokstäver
M-C	Ändra nästa ord till stor begynnelsebokstav

Arbetspunkten sätts efter ordet som ändrats, så ett upprepanande av kommandot ändrar successivt nästa och nästa ord. Med ett numeriskt argument kan flera ord ändras samtidigt. Argumentet 1 gör att föregående ord ändras utan att arbetspunkten

flyttas. Det är mycket användbart för att korrigera skrivfel (glömd skifttangente).

Dessutom finns två kommandon som ändrar alla bokstäver i hela området.

C-X **C-U**
C-X **C-L**

Ändra alla bokstäver i området till versaler
Ändra alla bokstäver i området till gemena

Stavningskontroll för engelsk text

Emacs kan anropa programmet ISPELL för att kontrollera om ett ord är ett korrekt stavat engelskt ord.¹ ISPELL har en stor ordlista och kontrollerar om ditt ord finns med där.

M-\$

Kontrollera stavningen av ordet före arbetspunkten

Om ordet finns i ordlistan så svarar Emacs "Found it". Är ordet oigenkännligt så får man istället meddelandet "Couldn't find it". Är ordet mycket likt ett eller flera ord i ordlistan så listar Emacs dessa ord med en siffra 0-9 före varje. Skriver du en siffra så byts ditt ord ut mot motsvarande ord i ordlistan. Trycker du på mellanslagstangenten så sker inget automatiskt utbyte.

Indragen text och tabulatorstopp

TAB
M-S
M-TAB
LINEFEED
M-^
M-I
C-X **TAB**
C-M-O

Drag in raden "lagom" mycket
Centrera raden
Sätt in ett Tab-tecken
Som **RETURN** följt av **TAB**
Sätt ihop raden med raden före
Gå till nästa tabulatorstopp
Flytta texten i området i sidled
Drag in alla rader i området "lagom" mycket

När man ska skriva en rad som ska vara indragen använder man normalt **TAB**-tangente i början på raden. Vilken effekt detta får beror på vilket huvudtillstånd man befinner sig i. Är man i texttillstånd så betyder det att man går till första tabulatorstoppen. Även i andra tillstånd kan man använda tabulatorstopparna, men då får man ge kommandot **M-I** istället. När man skriver flera rader som ska vara indragna efter varandra är det bekvämt att använda **LINEFEED** efter varje rad.

Kommandot **M-S** centrerar raden mellan kolumn noll (Obs: ej vänstermarginalen!) och högermarginalen. Med hjälp av kommandot **C-X** **TAB** kan man flytta hela området som ett block i sidled. Ett numeriskt argument talar om hur många steg det ska flyttas. Ett positivt argument ger flyttning åt höger, negativt åt vänster.

1 Stavningskontroll finns inte i Amis.

Tabulatorstopparna² kan man flytta med kommandot

[M-X] Edit Tab Stops

som låter dig använda Emacs för att redigera en text som innehåller tabulatorstopparna. Tabulatorstopparna är markerade med kolon, och du kan ta bort, flytta och sätta in kolon på de platser där du vill ha dina stoppar. När du är klar lämnar du denna redigering med **[C-M-Z]** (som ju skrivs **[C-Z][Z]** på terminaler utan Metatangent).

Marginalställning

Innan du ger några kommandon som har med marginalställning av text att göra bör du ladda in biblioteket "WP".³ Vissa av kommandona finns även om WP inte laddats, men de fungerar inte på riktigt samma sätt.

[C-X][F]	Sätt högermarginalen
[C-X][.]	Sätt vänstermarginalen
[M-Q]	Marginalställ till nästa blanka rad
[M-G]	Marginalställ området
[C-M]-	Avstava ordet och marginalställ igen

För att tala om för Emacs var du vill ha din högermarginal flyttar du arbetspunkten till den positionen och ger kommandot **[C-X][F]** (engelska: *Fill Column*). Du kan också ge önskad kolumn som numeriskt argument till kommandot. Om du vill ha vänstermarginalen indragen måste du också sätta vänstermarginalen. Det gör du med kommandot **[C-X][.]** på motsvarande sätt.

När du skriver in löpande text kan du få Emacs att automatiskt bryta av raderna när nästa ord hamnar utanför högermarginalen. Denna funktion får du om du sätter på undertillståndet "Auto Fill Mode". Då behöver du inte trycka på **[RETURN]** medan du skriver in styckets text. Använd **[RETURN]** för att avsluta stycket och **[LINEFEED]** för att påbörja ett nytt indraget stycke.

När du har ändrat i ett stycke så att marginalerna inte längre stämmer kan du justera om stycket med hjälp av kommandot **[M-Q]**. Placera arbetspunkten före stycket, ge kommandot **[M-Q]**. Texten fram till nästa blanka rad blir då marginalställd. Om du t ex har flera stycken utan blanka rader emellan kan du ange hur många rader som ska justeras som ett numeriskt argument till **[M-Q]**. Alternativt kan du sätta märket i slutet på området som ska justeras och ge kommandot **[M-G]**.

Avstavningar ställer till lite speciella problem när man marginalställer text. WP innehåller därför speciella hjälpmedel för att ta bort och sätta in avstavningar. Efter att

² Amis har inga flyttbara tabulatorstoppar.

³ Amis har endast en enklare form av marginalställning utan hantering av avstavade ord och med sämre understöd för hantering av indragen text. Auto Fill Mode finns dock.

du justerat ett stycke med **[M-Q]** placeras arbetspunkten på den första möjliga avstavningspunkten i stycket. Du får själv bestämma vad som är en lämplig avstavningspunkt genom att placera arbetspunkten där och ge kommandot **[C-M-]**. Ordet blir då avstavat och resten av stycket omjusterat igen.

När du justerar om ett stycke som redan innehåller avstavningar får du en fråga för varje sådant ord. Du får själv bestämma om bindestrecket ska plockas bort (engelska: *Remove*), vara kvar (engelska: *Keep*) eller behållas som två ord (engelska: *Split*) genom att svara med **[R]** (eller ekvivalent: **[SPACE]**), **[K]** resp **[S]**.

Marginalställning med WP ger normalt en taggig högermarginal, men det är också möjligt att få rak högermarginal genom kommandot:

[M-X] Right Margin Mode

16

Flera buffertar

Tidigare har vi bara arbetat med en text i taget med Emacs. Denna text har legat i den s k bufferten. Ibland har man anledning att arbeta med flera olika texter på en gång. Det kan t ex vara när man håller på och flyttar delar av en text från en fil till en annan, när man parallellt ändrar i två olika filer eller när man läser en manualtext samtidigt som man skriver ett program.

I dessa fall kan man utnyttja att Emacs kan ha flera buffertar på gång samtidigt. Man läser in en fil i varje buffert, precis som man normalt gör i sin enda buffert. Därefter kan man mycket snabbt växla fram och tillbaka mellan sina olika buffertar. Man behöver alltså inte spara innehållet i en buffert på fil bara för att växla till en annan buffert. Man måste dock komma ihåg att gå runt till alla buffertar och spara dem innan man lämnar Emacs för gott. Detta kan man antingen göra med `(C-X)(C-S)` i varje buffert som vanligt, eller med kommandot

`(M-X)` Save All Files

Dessa kommandon behöver man för att arbeta med flera buffertar:

`(C-X)(C-B)`
`(C-X)(B)`
`(C-X)(K)`
`(C-X)(C-F)`

Lista vilka buffertar vi har
 Växla till en annan buffert
 Tag bort en buffert
 Finn eller skapa buffert med angiven fil

Alla buffertar har ett namn. Den buffert man har från början har namnet "Main". När man listar buffertarna med kommandot `(C-X)(C-B)` får man reda på var buffertarna heter och vilka filer som lästs in i dem. När man växlar buffert med kommandot `(C-X)(B)` frågar Emacs efter namnet på den buffert som önskas. Om man skriver ett nytt namn så skapas en ny buffert med det namnet.

Med kommandot `(C-X)(C-F)` läser Emacs in en fil i en ny buffert, men om filen redan fanns i en buffert så växlar man bara över till den direkt. Detta kommando är alltså ett bekvämt och effektivt sätt att växla mellan olika filer som man arbetar med. När `(C-X)(C-F)` skapar en ny buffert blir namnet automatiskt samma som filens huvudnamn.

Ibland vill man plocka bort buffertar som man inte längre har någon användning för. Då kan man använda kommandot `(C-X)(K)`.

Observera att det också finns möjligheter att samtidigt se innehållet i två olika buffertar på skärmen genom att använda två fönster. Hur detta går till beskrivs i

kapitel 18.

17

Filhantering

Redan tidigare har vi beskrivit några kommandon för hantering av filer. Här ska vi gå igenom dem och några till lite ordentligare.

C-X **C-V**
C-X **C-F**
C-X **C-S**
C-X **C-W**
C-X **C-Q**
M-~

Läs in en fil i bufferten
 Finn en fil i någon buffert
 Spara buffertens innehåll på filen
 Skriv ut bufferten på någon fil
 Ändra skrivskyddet för filen/bufferten
 Tag bort markeringen av bufferten som ändrad

Med kommandot **C-X** **C-V** läser man in en ny fil i bufferten. Emacs ställer frågan

Visit File:

och du ska då svara med filnamnet. Om du tänker växla fram och tillbaka mellan två eller flera filer kan det vara bra att läsa in filerna i egna buffertar. Du använder då **C-X** **C-F** istället.

När du är färdig med dina ändringar, eller vill spara undan dem för att vara säker på att de inte ska gå förlorade, använder du kommandot **C-X** **C-S**. Buffertens innehåll sparas då på den fil som anges på modraden. Om du vill skriva ut resultatet av redigeringen på en annan fil använder du istället kommandot **C-X** **C-W**. Emacs ställer då frågan

Write File:

och väntar sig att du svarar med önskat filnamn.

Om du använder många buffertar kan det vara besvärligt att behöva gå runt till alla buffertar för att ge kommandot **C-X** **C-S** överallt. Du kan då istället ge kommandot

M-X Save All Files

som för alla icke skrivskyddade filer där någon ändring gjorts, frågar om bufferten ska sparas.

Med kommandot

M-X Insert File

kan man kopiera in texten från en annan fil in i den buffert där man arbetar. Filens innehåll stoppas naturligtvis in vid arbetspunkten.

Skrivskydd

Om man inte avser att ändra i filen, utan bara vill läsa den, kan det vara lämpligt att läsa filen skrivskyddad (engelska: *Read Only*). Detta gör man genom att ge ett numeriskt argument till `[C-X][C-V]` eller `[C-X][C-F]`. Om argumentet är positivt så betraktar Emacs filen som skrivskyddad och hindrar att man senare sparar ändringar man gjort i bufferten (om man inte speciellt säger till). Om argumentet är negativt så blir texten i bufferten skrivskyddad. Emacs säger då ifrån så fort man försöker ändra i texten.

Man kan också ändra skrivskyddet för bufferten/filen med kommandot `[C-X][C-Q]` som ska ha samma argument som beskrivits ovan.

Radnumrerade filer

Filer som har redigerats med vissa andra editorer (typiskt SOS, EDIT eller motsvarande) efterlämnar ibland en radnumrering i den resulterande filen. Om man vill redigera en sådan fil med hjälp av Emacs bör man, som det första man gör, plocka bort dessa radnummer. Detta kan man göra i Emacs med hjälp av kommandot

`[M-X]` Strip SOS Line Numbers

18

Två fönster

Emacs har inte bara möjligheter att ha flera buffertar aktuella samtidigt. Man kan också samtidigt titta i två av dem genom att använda två fönster. Två fönster innebär att Emacs delar av skärmen med ett vågrätt streck och att de två fönstren oberoende av varandra kan flyttas omkring i olika buffertar, eller vara på olika platser i samma buffert.

C-X	1
C-X	2
C-X	3
C-X	O
C-X	^
C-M-V	

Ett fönster

Två fönster

Två fönster, men stanna i det första

Växla till andra fönstret

Gör aktuellt fönster större

Bläddra fram det andra fönstret

Med C-X 2 får du Emacs att visa två fönster. Du kan bara arbeta i ett åt gången, men du kan byta aktuellt fönster med C-X O (bokstaven O som i *Other Window*). När du vill gå tillbaka till att använda bara ett fönster igen ger du kommandot C-X 1.

För att arbeta med två olika filer i var sitt fönster ger du först kommandot C-X 2 för att få två fönster. Du får då ett andra fönster med en ny tom buffert. Du ger därefter kommandot C-X C-F för att växla till en buffert med den önskade filen. Detta fungerar vare sig den önskade filen redan finns i någon buffert eller ej. Nästa gång du använder C-X 2 får du tillbaka samma fönster med samma buffert.

Eftersom raderbuffert och Q-register är gemensamma för alla buffertar är det mycket enkelt att kopiera och flytta textavsnitt mellan olika filer. Med två fönster blir det också lätt att se vad man gör. Man kan alltså ta bort text i ett fönster, ge kommandot C-X O för att komma över i det andra fönstret och där sätta in samma text med C-Y.

En annan användning av två fönster är när man vill titta i en annan fil än den man arbetar med. Det kan vara en manual eller hjälpfil eller kanske ett annat liknande program som man vill kika på. I detta fall kan det vara lämpligt att markera den andra bufferten som skrivskyddad genom att ge ett negativt argument till C-X C-F när man läser in filen.

19

Variabler

Emacs¹ har ett stort antal variabler som styr funktionen hos många kommandon. Genom att ändra dessa variabler kan användaren anpassa funktionen till egna önskemål.

Variabler har flerordsnamn, precis som kommandon. Med kommandot

(M-X) View Variable

kan du få reda på vad en variabel har för värde. Med kommandot

(M-X) Set Variable

kan värdet ändras. För att ge ett numeriskt värde på variabeln ger du det som argument till kommandot. Ger du inget numeriskt argument så frågar Emacs efter ett textvärde som variabeln ska få.

Ett trevligare sätt att ändra variablers värden är dock att använda funktionen

(M-X) Edit Options

När du ger detta kommando frågar Emacs efter

Pattern:

Du svarar då med variabelns namn, eller en del av variabelns namn. Emacs låter dig sedan redigera en text som innehåller namn och värden för alla de variabler som matchar den text du skrev. Genom att variablernas namn har valts så att variabler som hör ihop har namn som delvis är lika får man automatiskt alla aktuella variabler tillgängliga. När du ändrat värdet på de variabler du ville avsluta du med kommandot **(C-M-Z)** (som ju kan skrivas **(C-Z)(Z)**).

Filen EMACS.VARS

Genom att man har en egen fil med namnet EMACS.VARS kan man där specificera vilka värden man själv önskar ha på sina variabler. Här är ett exempel på hur en sådan fil skulle kunna se ut:

1 Amis har inga variabler som kan ändras av användaren

```

Comment Column: 70
Comment Begin: "-"
Tab Stop Definition: " : : : : : : "

```

Man skriver alltså variabelns namn följt av ett kolon. Därefter skriver man det värde som man vill ha. För de variabler som ska ha strängar som värden skriver man citationstecken kring strängarna.

Lokala värden för en fil

Emacs har möjlighet att låta vissa variablers värden vara lokala för en buffert. Ett sätt att få sådana lokala värden är att ange variabelvärden i slutet på en fil i en s k *Local Modes List*. Dessa värden kommer då alltid att gälla automatiskt när man redigerar just den filen. På så sätt kan man t ex lagra tabulatorstoppar, högermarginal o dyl direkt i den redigerade filen.

Listan ska placeras på en egen sida sist i filen och ha ett utseende på formen:

```

; Local Modes:
; Auto Fill Mode:1
; Comment Column:32
; Left Margin: 6
; Right Margin: 70
; Tab Stop Definition: " : : : : "
; End:

```

Semikolonet i början på varje rad får bytas ut mot någon annan sträng, men det måste vara samma sträng som inleder alla rader. Listan ska börja med en rad där det står `Local Modes:` och avslutas med en rad `End:`.

Några variabler

Här är ett urval av alla de variabler som man kan vilja ändra på. Inom parentes efter variabelns namn anges vilket värde som gäller om man inte sagt något annat.

Auto Push Point Option (500)

Antal tecken som en sökning måste flytta arbetspunkten för att märket ska sättas i början.

Auto Save Default (0)

Annat värde än noll gör att Auto Save Mode används automatiskt.

Auto Save Filenames (" SAV00")

Filnamnet som används vid Auto Save Mode.

Auto Save Interval (500)

Hur många tangenttryckningar man ska göra mellan Auto Save.

Auto Save Visited File (0)

Om denna variabel är 1 sparar Auto Save filen under sitt riktiga namn.

Autoarg Mode (0)

Om denna variabel är 1 tolkas siffror och minustecken som numeriskt argument om de skrivs före kommandon.

Case Replace (1)

Om 1 så försöker utbyteskommandon bevara ordens form med stora och små bokstäver.

Case Search (1)

Om 1 så betraktas stora och små bokstäver som lika vid sökningar.

Comment Begin

Den text som inleder kommentarer som Emacs sätter in.

Comment Column

Position där kommentarerna ska placeras.

Comment End

Den text som avslutar kommentarer.

Comment Start

Den text som inleder kommentarer som Emacs söker efter.

Display Matching Paren (-1)

Om icke-noll så visar Emacs var motsvarande vänsterparentes befinner sig när man skriver en högerparentes.

Display Mode Line Inverse (0)

Om icke-noll så visas modraden med omvänd svärtning (ifall du använder en terminal som kan detta).

Echo Area Height (3)

Antalet ekorader som Emacs använder.

Error Messages in Echo Area (0)

Om 1 så skrivs felmeddelanden på ekoraderna istället för med överskrivning.

Even Right Margin (0)

Marginalställning ger rak högermarginal om denna variabel har ett värde skiljt från noll (endast om biblioteket WP laddats).

Fill Extra Space List ("?!")

De tecken som ska följas av dubbla blanktecken.

Indent Tabs Mode (-1)

Om noll så används inga TAB-tecken för indragningar, utan bara blanka.

Left Margin (0)

Vänstermarginalen (endast om biblioteket WP laddats).

Page Delimiter ("^L")

Sträng som används för att söka efter sidbyten.

Permit Unmatched Paren

Om 1 så är det tillåtet att sätta in högerparenteser som inte har någon matchande vänsterparentes.

Region Query Size (5000)

Vissa kommandon som gör stora ändringar på området frågar användaren ifall området innehåller fler än så här många tecken.

Right Margin

Högermarginalen (endast om biblioteket WP laddats).

Space Indent Flag (0)

Om icke-noll så gör Auto Fill Mode att nya rader dras in.

Tab Stop Definitions (" : : : : ...")

Denna variabel innehåller en sträng med tabulatorstopparna.

Visit File Save Old (1)

Om -1 så sparas filen automatiskt när man läser in en ny. Om 1 så frågas användaren först.

20

Kommandomakron

Ibland står man inför uppgiften att upprepa samma sekvens av tangenttryckningar många gånger. Det kan då vara ide att skapa ett s k *kommandomakro* för detta. Ett kommandomakro innebär att man utför sekvensen av tangenttryckningar en gång samtidigt som Emacs lagrar dem. Man kan därefter utföra samma kommandon igen, en eller många gånger.

C-X	[
C-X	]
C-X	E
C-X	Q

Börja definiera ett kommandomakro
 Avsluta definitionen av kommandomakrot
 Utför kommandomakrot
 Villkor i kommandomakro

När man vill definiera ett kommandomakro ger man kommandot `C-X[`, följden av tangenttryckningar och slutligen kommandot `C-X]`. Samtidigt som du definierar makrot utförs alla kommandon du ger så att du kan se att du tänkt rätt.

När du väl definierat makrot kan du använda det genom kommandot `C-XE`. Om du ger ett numeriskt argument till kommandot så utförs makrot flera gånger. Om argumentet är noll kommer det att fortsätta utföras tills något fel inträffar, t ex att en sökning misslyckas.

Du kan också lägga in ett villkor i makrot genom att använda `C-XQ` (engelska: *Query*). Detta kommando gör ingenting alls när makrot definieras, men när det sedan används frågas användaren om makrot ska fortsätta eller inte. Användaren kan då bl a svara

SPACE
RUBOUT
ESCAPE
.

Ja, fortsätt med makrot
 Nej, hoppa över resten av makrot
 Nej, sluta utföra detta makro
 Ja, utför makrot färdigt, men sluta sedan

Ytterligare några andra alternativ finns. Skillnaden mellan `RUBOUT` och `ESCAPE` märks bara om man givit ett numeriskt argument till `C-XE`. `RUBOUT` avbryter bara denna repetition av makrot medan `ESCAPE` avbryter hela `C-XE` kommandot.

21

Sagan om Amis eller Hur man utvecklar en editor

av Per ^{Tf Ma}_{Pogammer} Lindberg

Detta är berättelsen om sju hackers som ville göra en editor. Alla här förekommande personnamn är fullständigt ofingerade, och rejält insyltade. Berättelsen är f.ö. skriven med AMIS själv, så allt du läser är sant!

Det hela började en gång för länge, länge sedan, i datorernas barndom. Någon hade kommit på att man kunde redigera sina källprogram direkt på datorn (eller "elektronhjärnan", som man sa på den tiden). Han skrev sig ett program, med vars hjälp han kunde redigera ett stycke text, vilket som helst, innan han körde det genom kompilatorn, eller vad han nu ville göra med det han skrev. "En strålande idé!", sade alla hans kollegor, och stal den meddetsamma. Alla hade naturligtvis sina egna idéer om hur EDITORn, som programmet kallades, skulle fungera. Så de skrev om och la till, kopierade, plagierade och gjorde om. Och på så sätt kom idén om en editor att göra sitt segertåg världen över. När datorerna blev allt bättre och bättre, kom även editörerna att förbättras, och redan innan BILDSKÄRMSTERMINALERNA gjort den sista SKRIVANDE PAPPERTERMINALEN arbetslös, kom någon på att man borde göra en editor som höll en del av texten konstant uppritad på skärmen, så att man kunde åka omkring i texten och redigera. Den första SKÄRMEDITORN var född. Precis som när den första editorn kom till världen, spred sig idén som en löpeld, och alla försökte göra en egen skärmeditor.

Och sådan var situationen för ungefär ett år sedan, när Lars-Henrik försökte planka den nya fantastiska super-editorn EMACS. Denna EMACS var större, snabbare, bättre och gladare än någon annan hittills existerande skärmeditor. EMACS kom från datorernas Mecka, USA, och inte nog med det, utan till och med från detta Meckas Kaba, MIT! Till Lars-Henriks förtvivlan fanns inte EMACS tillgänglig på våran dator, NADJA. (Få av oss andra hade vid den tiden hört talas om EMACS, än mindre provat den). Men Lars-Henrik, eller KRSNA som vi kallade honom, var frälst, och ville göra en EMACS som gick att köra på Nadja.

Så småningom fick han ihop något som likande en editor. Skärmuppdateringen var av en ny och revolutionerande typ: Han skrev om HELA SKÄRMEN varje gång något enstaka tecken hade blivit förändrat. I 1200 baud. Men den fungerade, och

vi andra samlades runt Krsnas terminal och skrattade rätt.

Men nu hade vi blivit nyfikna, och ville lära oss vad det var för något fantastiskt som Krsna hade försökt göra. Så vi samlades tidigt en söndagmorgon och åkte i ett par bilar till Uppsala, där den dator på vilken Krsna hade lärt sig EMACS stod. (För att förstå vår entusiasm bör poängteras att det var något alledeles enastående att se detta gäng vaket så tidigt — och på en söndag, dessutom!) Den officiella anledningen var dock att vi skulle köra INTERLISP och det nya CFS (Computer Fantasy Simulation)-spelet ZORK. EMACS, INTERLISP, och ZORK kunde nämligen bara köras under operativsystemet TOPS-20, och på Nadja fanns bara TOPS-10. Inom kort var vi alla övertygade om att EMACS nog var rätt bra, och något som vi ville kunna köra även på NADJA. Så gick några veckor, och vi gick och tränade, och längtade tills nästa gång vi skulle åka till Uppsala.

Under tiden hade något annat fantastiskt hänt: ett program som heter KOM hade blivit tillgängligt på Nadja. KOM är ett "telekonferenssystem", det vill säga att i KOM kan man skriva brev till varandra, och inte nog med det, man kan ha "konferenser" där man gör "inlägg" och "kommentarer". Det hela fungerar ungefär som en insändarspalt, där alla som är "medlem" i en viss konferens kan läsa vad andra skrivit, kommentera, och själv skriva inlägg. Nu började vi alla med stor entusiasm köra KOM minst 2 gånger om dagen, och producera en astronomisk mängd totalt onyttiga nonsensinlägg.

Men så en dag skrev Krsna ett ilsket inlägg där han förklarade att han var utled på att inte ha en bra skärmeditor på Nadja! Vi skulle skriva en egen EMACS!! Om det bara samlades ett gäng hackers vid Nadja nästa söndag och hackade frenetiskt några timmar, borde vi kunna länka ihop våra moduler någon gång vid midnatt, och en första testversion av vår egen EMACS vara ett faktum! Allt skulle skrivas i Pascal och lite assembler. Samling klockan 12!

En total entusiasm utbröt, och eftersom det var ett par dagar kvar till söndag ägnade vi våra KOM-inlägg åt att smida planer. Hur skulle vi dela upp projektet mellan oss? Vad ska editorn heta? Skulle den vara en trogen EMACS-kopia, eller skulle vi göra en egen, och kanske ännu bättre editor? Eftersom vi via KOM hade kontakt med ett gäng likasinnade i Linköping, fick vi tillskott av en kille till som var intresserad av att skriva en editor, Anders Ström, av alla kallad ASTROM eftersom han inte använde Ö utan O i sitt user-namn när han körde på TOPS-20 i Linköping. (Han hade alltså också kört EMACS, och insett att allt annat är skräp).

Några förberedelser inför Den Stora Dagen gjordes. Bl.a. fick vi av NADA (Numerisk Analys och Datalogi, institutionen som har hand om datorn Nadja) ett PPN (konto) på Nadja där vi kunde arbeta och lägga upp våra filer. Det beslöts att projektet skulle utföras i STACKENs regi. (Stacken är datorföreningen på KTH.)

Så blev då klockan 12 denna efterlängtna söndag. Efter n (ett stort heltal) antal akademiska kvartar, var vi sju stycken hackers samlade: Lars-Henrik "Krsna" Eriksson, bröderna Per och Björn Danielsson, Erik "Captain Zilog" Forsberg, Johnny

Eriksson, Anders "Astrom" Ström (upprest från Linköping enkom för detta), och jag själv, Per Lindberg (kallad TMPSA eller TMP, efter min ovana att signera mina program "The Mad Programmer Strikes Again"). Och nu bröt en vild diskussion ut! En av de första allvarliga frågorna som raskt avhandlades, var när och var vi skulle äta middag. Den lokala kineskrogen "Minus Ett" sattes som default, tillsammans med ett klockslag. Så småningom enades vi efter mycket vilda (och stundom handgripliga) diskussioner om ett par punkter:

1. Editorn skulle vara en trogen delmängd av EMACS. Inga egna påhitt.
2. Ordförande och projektledare Krsna må, om så krävs, använda livremmen för att få dom andra att hålla klaffen.
3. En skärmuppdateringsmodul skulle anropa en logisk skärmhanterare som i sin tur anropar en fysisk dito.
4. Erik håller klaffen, och slutar prata om ovidkommande saker.
5. Ett subset EMACS-kommandon valdes ut.
6. Övriga moduler utkristalliserades och delades ut.

Efter en stormig middag, satte sig Krsna vid en terminal, och, ivrigt påhejad av oss andra, skrev en lista på vilka rutiner som skulle finnas i respektive modul.

Nu hade klockan blivit så mycket att Astrom måste åka med sista tåget tillbaka till Linköping, och vi andra satte oss utmattade framför var sin terminal och läste dagens KOM-inlägg. Klockan blev midnatt, och någon editor hade det inte blivit. Det var nog inte så lätt att göra en editor, iallafall.

Veckan som följde började vi skriva våra moduler och vilt diskutera vårt projekt i KOM. I KOM löste vi namnfrågan, som efter omröstning (I KOM, såklart!) blev mitt eget förslag: AMIS efter Anti-MISär, ett symboliskt namn på vårt hopp om en bättre framtid. "Editor X" var visserligen ett invariant namn, men, trots häftigt motstånd från Björn, ströks det från listan på möjliga namn på editorn. Per Daniels-son kontrade dock med en alternativ utläsning av AMIS: "Anti - Mung Interface for Suckers". Krsna ville kalla den TMACS eller TEMACS (ur Tiny EMACS) eller XEDIT. Det senare förslaget visade sig redan vara upptaget av en hemsk editor på CDC-datorerna. Återstod så bara att få honom att ge upp TMACS, vilket lyckades efter en smula elak övertalning. För att ytterligare öka förvirringen kom någon med en ny uttolkning av AMIS: "A Major Innovation In Software".

Så blev det söndag igen, och alla samlades för att gå igenom och revidera våra moduler. Inte heller nu blev vi färdiga, men kinesmaten var lika god som förra gången. Söndagarna passerade revy.

Krsna hade under denna period börjat få för sig att man även kan äta vårrullen med pinnar, och gjorde tafatta försök att skära upp den i små bitar med sina chopsticks. (Han är numera proffs på detta trick!) Ett annat inside-grepp var att man skulle lägga maten i risskålen, och äta med pinnarna i ena handen, och risskålen i andra, nära munnen. Det var under denna period som jag slutade fumla med pinnarna och började använda dom som om det vore den naturligaste sak i världen.

Så en mörk natt när jag satt hemma vid min skärm och körde KOM via telefonlinjen, fick jag syn på ett inlägg av Per Danielsson, som hackade DISKIO, det vill säga den modul som skötte in- och utmatningen på fil. "Break-even point nådd!" För första gången hade en modul i AMIS editerats med AMIS själv. Entusiasmen, som aldrig varit särskilt låg, steg nu, och jag själv började faktiskt tro att AMIS skulle bli något användbart. Det var ungefär vid denna tidpunkt som åtminstone jag började inse att vad som sagts från några i Linköping (som inte var med i AMIS-projektet) om Astrom, var sant. Nämligen att han hade 2 st. Lysares kapacitet när det gällde att skriva program. (LYSATOR är datorföreningen i Linköping). Hans modul, den som sköter skärmuppdateringen, visade sig nämligen vara i stand till de mest vansinniga trix: den skrev t.ex. ALDRIG om någon del av skärmen där det (av en ren tillfällighet) redan stod vad som skulle vara där.

Andra finesser och featuers diskuterades: bl.a. vad "kill-bufferten" skulle innehålla innan man själv tagit bort någon text? Ett tag innehöll denna buffer (som går att återkalla, därigenom kan man alltså flytta omkring textbitar) innan man tagit bort något innehålla "The Gettysburg Adress", ett känt amerikanskt tal från nationens barndom. Efter en hetsig debatt beslöts att den skulle bort, det skulle bara vara förvirrande för nybörjarna som ville lära sig AMIS. (Vi tänkte naturligtvis att alla elever på NADA skulle läras AMIS som första editor, en profetia som faktiskt delvis slog in.) Ett nytt talesätt myntades: "Därför att EMACS gör så!!!!!!!" Denna dräpande replik visade sig vara ytterst användbar i alla diskussioner om AMIS, och avslutade snabbt varje dispyt som kunde uppstå. Uttrycket blev så spritt (framförallt i KOM) att det till sist kom att användas i en vidare mening, där varken AMIS eller EMACS behövde vara med. Johnny tillverkade en skylt med de bevingade orden, och satte upp till allmän munterhet.

Projektgruppen hade också fått tillökning i personalen: Jan Michael Rynning, kallad JMR, och Örjan Ekeberg, kallad Örjan. JMR tyckte att vi var lite söliga med att skriva sökrutinerna så han hackade snabbt ihop en egen modul innehållande sökrutiner till AMIS. Örjan kunde inte heller hålla fingrarna i styr, utan hjälpte till med att göra en lathund och hackade också in förkortning av s.k. "Meta-X" - kommandon. Numera är JMR fullt insyltad i projektet, och Örjan är vår femte-kolonnare på NADA.

Nu hade vi en editor som gick att använda, även om mycket återstod att göra. Riktigt HUR mycket hade vi väl inte riktigt klart för oss vid den tiden, annars skulle nog många av oss tappat sugen totalt. Det var nämligen nu som dom VERKLIGA problemen började. Ryktet om AMIS hade spritt sig rejält, och nu ville alla titta på den lustiga lilla apen. Det var alltså dags för en RELEASE!

Den Stora Dagen var som sig bör en söndag, och vi samlades rätt tidigt, vill jag minnas. En ryslig sak slog oss: vi hade ingen MANUAL. Snabbt kasta ihop en sådan, alltså. En total genomgång av samtliga moduler visade att några behövde avlusas lite. Den fruktade "sista buggen" skulle bort innan vi vågade visa upp något. (Fotnot: den sista buggen definieras i hacker-kretsar som den bugg man tar bort före den sista buggen...) Klockan gick, och vi slet med manual och buggar. Dessutom

uppstod vissa organisatoriska problem: vi behövde en speciell version av ALLTING som skulle sparas som en särskild "release-version", och en speciell area att lägga dessa på. Klockan blev midnatt, och mycket återstod att göra. Så småningom kunde vi ändå vara ense om en relativ bugfrihet, och började sammanställa de färdiga modulerna, och kompilera dessa. I den allmänna röran (vi hade börjat bli lite sömniga allihop) uppstod diverse missförstånd, och klockan gick. När det började ljusna över taken (det var mitt i vintern då!!) kunde vi till sist meddela Tommy Ericson som satt som systemprogrammerare på QZ, Stockholms Datorcentral, att han kunde kopiera rubbet. Ytterligare missförstånd: Manualen var skriven med svensk teckenstandard, och alla HELP-filer o.dyl. skulle vara konverterade till "QZ-standard", vilket innebar att vi (d.v.s. de få av oss som var kvar och hållit oss vakna) fick köra dem igenom diverse konverteringsprogram. (Denna konvertering är nu avskaffad på Nadja, och senare fick vi vi konvertera tillbaka...) Efter ett antal försök lyckades vi, och releasen var ett faktum. Hem och kasta sig på sängen! Naturligtvis var det inte riktigt rätt nu heller, och under måndagen som följde krävdes ytterligare räddningsaktioner för att ställa saker till rätta.

Och nu började cirkusen ta fart på riktigt! En massa synpunkter och felrapporter började strömma in, och ett speciellt möte i KOM skapades, där synpunkter framfördes och kommenterades. (Ännu i dag diskuteras olika finesser livligt i KOM.) Det gamla tillhygget "—Därför att EMACS gör så!!!!!" användes flitigt. AMIS spreds med ljusets hastighet på det nät av DEC-maskiner som Nadja hänger på, ANF-10. Nu hade dessutom flera fått upp ögonen för att något var i görningen. Grabbarna på ELVIRA, undervisningsdatorn vid E-sekt. på KTH, en PDP-11, var snabbt på plats, och ville få en kopia av källkoden på AMIS för att kompilera med sin egen Pascal. Rapporter om inkompatibiliteter mellan våra olika Pascaler började hagla. Meningen var ju från början att vi skulle skriva Editor X i Pascal för att 1. få den någorlunda snabb, och 2. för att få den flyttbar till andra system. Nu visade det sig att vi utnyttjat saker i vår Pascal som inte är standard. Mer problem. Och sedan ville Jan Åman och dom andra på Fysikums VAX-11 göra sammaledes. Samma problem igen. Dessa klarades dock upp efter några möten med alla inblandade.

Föga anade vi vid denna tid vilka provningar vi hade framför oss. I så fall hade det nog aldrig blivit någon editor, vilket ju vore synd. Den stora smällen kom när vi skulle SALJA editorn.

Jodå, efter inte fullt ett års hackande kom det besked från en firma med anknytningar till KTH: "Hur mycket vill ni ha för AMIS?" Vi kände oss visserligen smickrade, men såg framför oss ändlösa diskussioner med advokater som med hjälp av finstiltat kontrakt på någon pervers avart av svenska kräver att vi ska supporta COBOL-mode åt någon bokföringsfirma. Vi började försiktigt fråga höga vederbörande på NADA om vi egentligen fick sälja vår editor, och efter en del funderingar kom vi fram till att vi inte kunde ta emot pengar för AMIS, eftersom vi gjort utvecklingen med gratis maskintid som inte var tänkt till kommersiell användning. Men någon fann på råd: STACKEN, som är en ideell förening kunde sälja AMIS. Sagt och gjort, snart började vi använda vår hemhackade editor till att författa kontrakt — en användning som vi aldrig i våra vildaste drömmar tänkt på.

Det var nu som stämningen började bli lite otrevlig. Så länge alla jobbar gratis för sitt höga nöjes skull, är allt frid och fröjd. Men när det finns en stor summa PENGAR att slåss om — akta er! Alla hade förstås synpunkter, och några tog t.o.m. till skällsord. Efter oändliga manglingar kunde vi till sist sätta våra namn på en utskrift av den 4711:e versionen av kontraktet, som dessutom fått ett "barn" i form av tilläggskontrakt för VAX-versionen. Ett fridfullt lugn lågrades över alla inblandade. Och Stacken tackade för tillskottet i klubbkassan med en brakmiddag som saknade like! Per Danielsson gav sig inte förrän han fått en fjärde efter rätt.

Men Sagan om AMIS är inte slut ännu! Fler installationer har frågat efter (och fått) AMIS. Den första internationella ordern kom (via diverse datornät) från Knut Smaaland på UiO (Universitetet i Oslo:s DEC-10), och ryktet har t.o.m. sipprat ut på det berömda ARPA-nätet i USA. Ett flertal företag som vill skaffa en skärmeditor har frågat efter AMIS. (Idén med skärmeditorer har först efter ett tag spridits till Verkligheten, alias industrin.) Alla som har en bildskärmsterminal, vill naturligtvis ha AMIS modifierad så att den även går att köra på just hans terminalmodell. Och mitt upp i alltihop sitter en liten grupp hackers och hackar förtvivlat på nästa version av AMIS, som ska innehålla ytterligare en delmängd av pappa EMACS. Den färdiga versionen är fortfarande lika avlägsen...

22

Amis på Katia

av Jan Michael Rynning

Fyra dagar efter Peter hade fått igång TOPS-10 på Katia nere i CCCC, så tyckte jag det var dags att försöka få en fungerande AMIS till henne. STACKENS egen editor på STACKENS egen dator ...

Eftersom jag visste att det skulle krävas en del ändringar i AMIS, så skickade jag ett KOM-brev till Johnny och bad honom kopiera över källkoden från Oden till Kicki. Åkte hem och sov. Vaknade, åkte tillbaka till CCCC. Ingen källkod.

Jag hade ingen lust att vänta, så jag bestämde mig för att gå den hårda vägen — patcha direkt i maskinkoden. Dessutom såg jag det som en rolig utmaning att få igång AMIS på det sättet.

Tre olika typer av problem väntade mig:

Katia har en KA10-processor, som inte klarar av alla de maskininstruktioner som finns på de modernare KI10-, KL10- och KS10-processorerna. Det problemet hade jag redan löst, genom att skriva en emulator för DMOVE och DMOVEM — de två mest använda av de instruktioner som inte finns på KA10.

Minneshanteringen på KA10 skiljer sig från den på de andra processorerna. KA10 delar in användarprogrammets adressrymd i två segment: lågsegmentet, som börjar på adress 0₈, och högsegmentet, som börjar mitt i adressrymden, på adress 400000₈. För vart och ett av segmenten finns ett bas- och ett längdregister. Basregistren pekar ut var i det fysiska minnet segmenten börjar och längdregistren är till för att hindra användarprogrammet från att läsa och skriva i minnesadresser utanför segmenten.

På de modernare processorerna består den logiska adressrymden av 512 sidor à 512 ord. Översättningen från logiska till fysiska adresser görs med hjälp av en tabell, som anger var i det fysiska minnet de olika sidorna börjar. TOPS-10 simulerar indelningen i låg- och högsegment genom att lägga in lämpliga värden i tabellen, men högsegmentet behöver inte börja på 400000₈.

AMIS högsegment börjar på 600000₈, för att man ska kunna ha ett större lågsegment och därmed kunna editera större filer. För att kunna köra AMIS på Katia var jag tvungen att flytta ner högsegmentet till 400000₈. Det gjorde jag genom att skriva ett litet PASCAL-program, som gick igenom AMIS.EXE ord för ord, kollade om

värdet i höger halvord kunde vara en adress i högsegmentet och isåfall subtraherade 200000₈ från den. Jag kunde inte vara säker på att allt jag ändrade var adresser — det kunde ju vara texter eller någonting annat — men det var värt ett försök. Till sist patchade jag EXE-filsdirectoryt med FILDDT, så monitorn skulle ladda in högsegmentet på rätt adress.

Dags att provköra. ”?Illegal memory reference”. En tabell med adresser i vänster halvord visade sig vara boven. Fixade till den med FILDDT. Nu gick den patchade AMIS-versionen på Kicki — fast en del texter hade fått stryk. Det stod ”Fundamental” i st f ”Fundamental” på modraden, t ex.

Återstod att flytta över AMIS till Katia och övervinna det tredje hindret. Den modernaste version av TOPS-10 som går att köra på KA10 är 6.03A. I senare versioner av operativsystemet har det tillkommit en del saker, bl a några terminalparametrar som AMIS försöker läsa av.

Första försöket resulterade i att AMIS skrev ut att dittan-dattan terminalparameter inte gick att läsa av och gjorde EXIT. Grrr! Morrr! Frässs! Fel som man inte kan göra någonting åt och som inte har någon betydelse ska man ignorera! I synnerhet som det som är fel är rätt i äldre versioner av operativsystemet. Felkontrollerna försvann, en efter en, tills AMIS gav med sig och lät sig köras på Katia.

Och än i dag körs denna version av AMIS på Katia, fast Hazze Slöberg har ändrat tillbaka till ”Fundamental”.

```

PROGRAM kaamis;

TYPE
  word = PACKED RECORD
    rh, lh : 0..777777B;
  END;

VAR
  i, o : FILE OF WORD;
  w : WORD;

BEGIN
  reset(i, 'PUB:AMIS.EXE'); rewrite(o, 'AMIS.EXE');
  WHILE NOT eof(i) DO
    BEGIN
      read(i, w);
      IF (w.rh >= 600000B) AND (w.rh <= 670777B) THEN
        w.rh := w.rh - 200000B;
      write(o, w);
    END;
  close(i); close(o);
END.
AMIS (Fundamental) Main: DSKB:KAAMIS.PASÁ10,302,EMPTYÅ<055>

```

23

Sakregister

Algol Mode (i Amis)	40
Amis	5, 19
Apropos, Leta efter kommandon	27
Arbetspunkten	7
Argument, Numeriskt	21
Argument, Text	23
ATeach	5
Auto Fill Mode	42
Auto Push Point Option	55
Auto Save Default	55
Auto Save Filenames	55
Auto Save Interval	56
Auto Save Mode	42
Auto Save Visited File	56
Autoarg Mode	56
Automatisk skyddskopia	42
Automatiskt radbrott	42
Avstavning	48
Backup, automatisk skyddskopia	42
BBlock, Hjälp för att skriva stora blockbokstäver	43
Bibliotek	43
Blanka rader	11, 15
Borttagning av SOS radnummer	52
Buffertar, Flera	49
Bufferten	7
Byta plats	36
 CTRL-kommandon	11, 22
 Sätt märket vid arbetspunkten	32
 Hjälpkommandot	15, 27
 Prefix för CTRL-kommandon	22
 Flytta arbetspunkten till början av raden	12, 30
 Flytta arbetspunkten ett steg bakåt	12, 30
 Radera tecknet efter arbetspunkten	12, 34
 Flytta arbetspunkten till slutet av raden	12, 30
 Flytta arbetspunkten ett steg framåt	12, 30
 Avbryt	14, 25
 Radera resten av raden	12, 34
 CTRL-META-kommandon	22

C-M-	Avstava ordet och marginalställ igen	47
C-M;	Tag bort kommentar	41
C-M-V	Bläddra fram det andra fönstret	53
C-M-W	Låt nästa Kill-kommando lägga till i raderbufferten	35
C-M-X	Alternativ till M-X	23
C-M-Z	Lämna Emacs	25
C-N	Flytta arbetspunkten rakt ner till nästa rad	12, 30
C-O	Öppna en blank rad	15
C-P	Flytta arbetspunkten rakt upp till föregående rad	12, 30
C-R	Stegvis sökning bakåt	33, 37
C-S	Stegvis sökning framåt	33, 37
C-SPACE	Sätt märket vid arbetspunkten	32
C-T	Byt plats på två tecken	36
C-U	Numeriskt argument	21
C-U C-@	Gå till märket, återställ gamla märken	33
C-V	Gå en skärmsida framåt	12, 31
C-W	Tag bort området	34
C-X	Prefix för ytterligare kommandon	13
C-X [	Börja definiera ett kommandomakro	58
C-X]	Avsluta definitionen av kommandomakrot	58
C-X .	Sätt vänstermarginalen	47
C-X ^	Gör aktuellt fönster större	53
C-X 1	Ett fönster	53
C-X 2	Två fönster	53
C-X 3	Två fönster, men stanna i det första	53
C-X :	Sätt önskad kolumn för kommentarer	41
C-X B	Växla till en annan buffert	49
C-X C-B	Lista vilka buffertar vi har	49
C-X C-F	Finn eller skapa buffert med angiven fil	49
C-X C-F	Finn en fil i någon buffert	51
C-X C-L	Ändra alla bokstäver i området till gemena	46
C-X C-O	Radera tomma rader runt aktuell rad	16, 34
C-X C-P	Sätt området kring sidan	32
C-X C-Q	Ändra skrivskyddet för filen/bufferten	51
C-X C-S	Skriv ut texten i bufferten på filen	13, 51
C-X C-T	Byt plats på två rader	36
C-X C-U	Ändra alla bokstäver i området till versaler	46
C-X C-V	Läs in en annan fil i bufferten	13, 51
C-X C-W	Skriv ut bufferten på någon fil	51
C-X C-X	Byt märket och arbetspunkten mot varandra	32
C-X C-Z	Lämna Emacs	14, 25
C-X E	Utför kommandomakrot	58
C-X F	Sätt högermarginalen	47
C-X G α	Sätt in texten från Q-register α	36
C-X H	Sätt området kring hela bufferten	32
C-X K	Tag bort en buffert	49
C-X O	Växla till andra fönstret	53

C-X Q Villkor i kommandomakro	58
C-X RUBOUT Tag bort från meningens början	34, 45
C-X T Byt plats på området och gamla området	36
C-X X α Kopiera området till Q-register α	36
C-X A Till slutet av sidan	31
C-X A Till början av sidan	31
C-Y Sätt in senast borttagna text vid arbetspunkten	35
C-Z Prefix för CTRL-META-kommandon	22
C-Z C-Z Detsamma som C-M-Z	25
C-A Avbryt en rekursiv redigering	26
Case Replace	56
Case Search	56
Case, Stora och små bokstäver	45
Centrering	46
Character, Hjälp om ett kommando	27
Chess, Hjälp för att rita schackbräden	43
Columns, Hantering av flerspaltig text	43
Comment Begin	56
Comment Column	56
Comment End	56
Comment Start	56
CTRL-kommandon	11, 22
Datorföreningen STACKEN	2, 5
Definiera kommandomakro	58
Delete-kommandon	34
Describe, Hjälp om ett namngivet kommando	27
Dired, Paket för att titta på och ändra i sin filkatalog	25, 43
Display Matching Paren	56
Display Mode Line Inverse	56
Doc	15
Dokumentation	15, 27
Draw, Paket för att rita linjer m m	43
Echo Area Height	56
Edit Options	54
EDIT radnummer	52
Edit Tab Stops	46
EDIT-kommandot för att starta Emacs	18
Editor:, logiskt namn för editorn	19
Ekorader	8
Emacs Info	28
EMACS-kommandot för att starta Emacs	17
EMACS.VARS	54
Error Messages in Echo Area	56
ESCAPE Prefix för META-kommandon	22

Failing I-Search:	38
Felstavning, kontroll	46
Filer	13
Filer, Flera	49
Fill	47
Fill Column	47
Fill Extra Space List	57
Find File	51
Flera buffertar	49
FLUSHED	9
Flytta arbetspunkten	30
Flytta text	34
Flytta text mellan olika filer	53
Flyttning av arbetspunkten	12
Fortsättningsrader	8
Fundamental Mode	40
Fönstret	7
 Gamla märken	 33
Gemena	45
 Hjälpkommandon	 15, 27
Huvudtillstånd	40
Hyphenation	48
Högermarginalen	47
 Incremental Search	 37
Indent Tabs Mode	57
Indentering	41
Indragen text	41
INFO	25
Initieringsfil	54
Insert File	51
Insättning av text	10
ISpell	46
 Justering	 47
Justify	47
Jämn högermarginal	48
 Keep (vid avstavningar)	 48
Keyboard macro	58
Kill-buffer	34
Kill-kommandon	34
Kommandomakron	58
Kommandon, hur man skriver dem	22
Kommentarhantering	41

Kopiera text	34
Last Commands	28
Left Margin	57
Library	43
Lisp Mode	40
List Library	43
Load Library	43
Local Modes	55
Lokala variabelvärden för en fil	55
Långa rader	8
Lämna Emacs	14, 25
[M-] META-kommandon	11, 22
[M-$\\$] Kontrollera stavningen av ordet före arbetspunkten	46
[M-<] Till början av hela texten	31
[M-%] Selektivt utbyte	39
[M->] Till slutet av hela texten	31
[M-@] Sätt märket ett ord framåt	32, 45
[M-~] Tag bort markeringen av bufferten som ändrad	51
[M-^] Sätt ihop raden med raden före	34
[M-$\cdot$] Sätt in kommentar efter raden	41
[M-A] Bakåt en mening	30, 45
[M-B] Bakåt ordvis	30
[M-C] Ändra nästa ord till stor begynnelsebokstav	45
[M-D] Tag bort nästa ord	34
[M-E] Framåt en mening	30, 45
[M-F] Framåt ordvis	30
[M-G] Marginalställ området	47
[M-H] Sätt området kring stycket	32, 45
[M-K] Tag bort till meningens slut	34, 45
[M-L] Ändra nästa ord till små bokstäver	45
[M-LINEFEED] Fortsätt kommentar på nästa rad	41
[M-N] Sätt in kommentar efter nästa rad	41
[M-P] Sätt in kommentar efter föregående rad	41
[M-Q] Marginalställ till nästa blanka rad	47
[M-RETURN] Till radens början men efter inledande blanka	30
[M-RUBOUT] Tag bort föregående ord	34, 45
[M-S] Centra raden	46
[M-T] Byt plats på två ord	36
[M-U] Ändra nästa ord till stora bokstäver	45
[M-V] Gå en skärmsida bakåt	12, 31
[M-W] Kopiera området till raderbufferten	35
[M-X] Namngivna kommandon	22
[M-Y] Byt återskapad text mot ännu äldre	35
[M-A] Till slutet av stycket	30, 45
[M-Å] Till början av stycket	30, 45

[M-O] Radera blanka (och Tab-tecken) runt arbetspunkten	34
Macro Mode	40
Major Mode	40
Makron	58
Marginalställning	47
Mark	32
Meddelanden	9
Meningar	30
META-kommandon	11, 22
META-tangent	22
Minibufferten	14
Minor Mode	41
Mode	40
Modraden	8, 25
-MORE-	9
Märken, Gamla	33
Märket	32
Nadja, Att starta Amis på Nadja	19
Namn på buffertar	49
Nämngivna kommandon	23
Numeriskt argument	21
Ny fil	14
Oden, Att starta Amis på Oden	19
Området	32
Ordbehandling	43
Overwrite Mode	42
Page Delimiter	57
Page, Annorlunda hantering av sidindelade filer	43
Paragrafer	30
Pascal Mode	40
Permit Unmatched Paren	57
Phrase, Rutiner för hantering av delar av meningar	43
Picture, Hjälpmedel för att rita figurer	43
Prefixtecken	22
Prime, Att starta Amis på Prime	20
Programspråkstillstånd	41
Q-register	35
Query Replace	39
Radbyten, insättning av	11
Radera enstaka tecken	11
Radera text	12, 34
Raderbufferten	34

Radnumrerade filer	52
Radutfyllnad	47
Rak högermarginal	48
Read Only	52
Region	32
Region Query Size	57
Rekursivt anrop av Emacs	26
Remove (vid avstavningar)	48
Renum, Rutiner för att numrera om formler, kapitel o dyl	43
Repetera ett kommando flera gånger	21
Replace	39
Replace String	23
RESET (FORK) EMACS	18
Right Margin	57
Right Margin Mode	48
<u>[RUBOUT]</u> Radera tecknet före arbetspunkten	12, 34
Rättstavning, kontroll	46
 Sammansatta kommandon	58
Satser	30
Save All Files	49
Selektivt utbyte	39
Set Variable	54
Sidbyten	31
Simula Mode	40
Skapa ny fil	14
Skrivskydd	52
Skyddskopia	42
Slowly, Anpassning för långsamma terminaler	43
Sluta, hur man lämnar Emacs	14, 25
Små bokstäver	45
Sort, Sortering av rader/sidor/stycken.....	43
SOS radnummer	52
Space Indent Flag	57
Spell	46
Split (vid avstavningar)	48
STACKEN, Datorföreningen	2, 5
Starta Amis	19
Starta Emacs	17
Stavningskontroll	46
Stegvis sökning	37
Stora bokstäver	45
Strip SOS Line Numbers	52
Stycken	30
Substituering	39
Svensk Text	42
Swedish Mode	42

System, Rutiner för information om systemet	43
Sökning	37
Tab Stop Definitions	57
Tabulatorstoppar	46
Taggig högermarginal	48
Tags, Hjälp för att hitta procedurer o dyl i flera filer	43
Tangentbordsmakron	58
Teach-Emacs	5
TERMDEF, Terminaltyper på Primesystem	20
Terminaltyp	17
Terminaltyp, fel	18
Text Mode	40
Textargument	23
Textkommandon	45
Tillstånd	40
TOPS-10, Att starta Amis under TOPS-10	19
Transpose	36
Två buffertar	49
Två filer	49
Två Fönster	53
Undertillstånd	41
Undo, kommando för att återställa	25
Upprepa samma kommando	21
Ut, hur man lämnar Emacs	14, 25
Utbyte	39
Variabler	54
VAX, Att starta Amis på VAX	20
Versaler	45
View Variable	54
Visit File	51
Visit File Save Old	57
VT100, Utnyttjar funktionstangenterna på VT100-terminal	43
Vänstermarginalen	47
Wordab, Automatisk expansion av förkortade ord	43
WP, (Word Processing) Rutiner för ordbehandling	43, 47
Write File	51
XINFO	28
Återstart av Emacs	18
Överskrivning	9
Överskrivningstillstånd	42

Plats för egna anteckningar

Prefixtecken

C-U	Numeriskt argument
C-^	C-prefix
ESCAPE	M-prefix
C-Z	C-M-prefix
M-X	Namngivna kommandon

Lämna och återställa

C-G	Avbryt
C-X C-Z	Lämna Emacs eller Amis
C-M-Z	Lämna nivå
C-A	Avbryt rekursiv nivå
M-X Undo	Gör någonting o gjort

Flytta arbetspunkten

C-F	Framåt teckenvis
C-B	Bakåt teckenvis
M-F	Framåt ordvis
M-B	Bakåt ordvis
C-P	Uppåt radvis
C-N	Nedåt radvis
C-A	Början av raden
C-E	Slutet av raden
M-A	Bakåt en mening
M-E	Framåt en mening
M-A	Början av stycket
M-A	Slutet av stycket
C-V	En skärmsida framåt
M-V	En skärmsida bakåt
C-X A	Början av sidan
C-X A	Slutet av sidan
M-<	Början av hela texten
M->	Slutet av hela texten

Märket och området

C-@	Sätt märket
C-SPACE	Sätt märket
C-X C-X	Byt plats med märket
C-U C-@	Återställ märket

Radera och flytta text

C-D	Radera nästa tecken
RUBOUT	Radera föregående tecken
M-D	Tag bort nästa ord
M-RUBOUT	Tag bort nästa ord
C-K	Tag bort till radslut
C-W	Tag bort området
M-K	Tag bort till meningens slut
C-X RUBOUT	Tag bort från meningens början
C-Y	Återskapa senast borttagna text
M-Y	Byt återskapad text mot äldre
M-W	Kopiera området till raderbufferten
M-O	Radera blanktecken
C-X C-O	Radera tomma rader
C-X X α	Kopiera området till Q-register α
C-X G α	Sätt in texten från Q-register α
C-T	Byt plats på två tecken
M-T	Byt plats på två ord
C-X C-T	Byt plats på två rader
C-X T	Byt plats på områden

Sökning och utbyte

C-S	Stegvis sökning framåt
C-R	Stegvis sökning bakåt
M-%	Selektivt utbyte

Indragen text och marginalställning

TAB	Drag in raden "lagom"mycket
M-TAB	Sätt in ett TAB-tecken
LINEFEED	Som RETURN följt av TAB
M-^	Sätt ihop raden med raden före
M-I	Gå till nästa tabulatorstopp
C-X TAB	Flytta texten i området i sidled
M-S	Centrera raden
C-M-O	Drag in alla rader i området
M-Q	Marginalställ ett stycke
M-G	Marginalställ området
C-M--	Avstava efter marginalställning
C-X F	Sätt högermarginalen
C-X _	Sätt vänstermarginalen

Stora och små bokstäver

M-U	Ändra nästa ord till stora bokstäver
M-L	Ändra nästa ord till små bokstäver
M-C	Ändra nästa ord till stor begynnelsebokstav
C-X C-U	Ändra alla bokstäver i området till stora
C-X C-L	Ändra alla bokstäver i området till små

Kommentarhantering

M-;	Sätt in kommentar efter raden
C-M-;	Tag bort kommentar
C-X ;	Sätt önskad kolumn för kommentarer

Filer

C-X C-S	Spara bufferten på fil
C-X C-V	Läs in en fil i bufferten
C-X C-F	Finn en fil i någon buffert
C-X C-W	Skriv ut bufferten på någon fil
C-X C-O	Ändra skrivskyddet
M-X	Save All Files
M-X	Insert File

Flera buffertar

C-X C-B	Lista vilka buffertar vi har
C-X B	Växla till en annan buffert
C-X K	Ta bort en buffert

Flera fönster

C-X 1	Ett fönster
C-X 2	Två fönster
C-X 3	Två fönster, stanna kvar
C-X O	Växla till andra fönstret
C-X ^	Gör aktuellt fönster större
C-M-V	Bläddra det andra fönstret

Kommandomakron

C-X (	Börja definiera ett makro
C-X)	Avsluta definition av ett makro
C-X E	Utför kommandomakrot
C-X Q	Villkor i kommandomakron

Hjälpkommandon

C-	Hjälpkommandot
C- C	Character
C- D	Describe
C- A	Apropos
C- I	INFO
C- L	Last commands