

# Vejledning i brug af EDB-systemet på ØL

---

af Allan Bo Hansen & Frederik Treue  
Bogen er en stærkt redigeret udgave af DIKU's Kursusbog 2,  
skrevet af Jesper Holm Olsen & Søren Debois

August 2004

Denne bog er sat med  $\text{\LaTeX}$ , documentclass book udvidet med (bl.a.) pakkerne Charter (10pt), Euler, a4wide og frontpage.

Reproduktion af denne bog eller dele af dens indhold er tilladt med tilhørende kildeangivelse, samt besked til forfatterne:

Frederik Treue ([treue@fys.ku.dk](mailto:treue@fys.ku.dk))  
Allan Bo Hansen ([abh@fys.ku.dk](mailto:abh@fys.ku.dk))  
Jesper Holm Olsen ([dunkel@diku.dk](mailto:dunkel@diku.dk))  
Søren Debois ([debois@diku.dk](mailto:debois@diku.dk))

# Indhold

<b>I</b>	<b>Basale funktioner</b>	<b>ix</b>
<b>1</b>	<b>Konto på Ørsted Laboratoriet</b>	<b>1</b>
1.1	Datalokaler . . . . .	1
1.2	Log ind . . . . .	1
1.2.1	Tekstbaseret login . . . . .	2
1.2.2	Grafisk login . . . . .	2
1.3	Log ud . . . . .	3
1.4	Pauseskærm . . . . .	3
1.5	Password . . . . .	3
1.6	Netværkstrukturen . . . . .	4
1.6.1	Quota . . . . .	5
1.6.2	Drev . . . . .	6
<b>2</b>	<b>Xwindows</b>	<b>7</b>
2.1	Displaymanager og windowmanager . . . . .	7
2.2	KDE . . . . .	8
2.3	Brugerfladen . . . . .	8
2.3.1	Genvejstaster . . . . .	9
2.3.2	Cut'n'paste . . . . .	9
<b>3</b>	<b>Microsoft Windows</b>	<b>11</b>
3.1	Log ind og log ud . . . . .	11
3.2	Brug af terminalserveren . . . . .	11

<b>4</b>	<b>Kommandofortolkeren</b>	<b>13</b>
4.1	Filsystemet . . . . .	13
4.2	Kommandoer . . . . .	14
4.2.1	Fil- og katalogmanipulering . . . . .	15
4.2.2	Særlige kataloger . . . . .	15
4.2.3	Usynlige filer . . . . .	15
4.2.4	Mønstre . . . . .	16
4.3	Filrettigheder . . . . .	16
4.3.1	Ændring af rettigheder . . . . .	17
4.3.2	Standardrettigheder . . . . .	18
4.4	Elektronisk dokumentation . . . . .	18
<b>5</b>	<b>Tekstredigeringsværktøjer</b>	<b>21</b>
5.1	Hvad er et tekstredigeringsværktøj? . . . . .	21
5.2	VI(M) . . . . .	22
5.2.1	Navigation . . . . .	22
5.2.2	Gem og luk . . . . .	23
5.2.3	Editeringskommandoer . . . . .	23
5.2.4	Klip, Klister og Kopiér . . . . .	23
5.2.5	Søg og erstat . . . . .	23
5.2.6	Kildetekstredigering . . . . .	24
5.2.7	Tips og tricks i vi . . . . .	25
5.2.8	Yderligere hjælp . . . . .	25
5.3	Emacs . . . . .	25
5.3.1	Koncepter i Emacs . . . . .	25
5.3.2	Taster . . . . .	28
5.3.3	Interne funktioner . . . . .	29
5.3.4	Håndtering af buffere . . . . .	29
5.3.5	Klip, klister og kopiér . . . . .	30
5.3.6	Filhåndtering . . . . .	31
5.3.7	Søg og erstat . . . . .	31
5.3.8	Fortryd . . . . .	32
5.3.9	Kildetekstredigering . . . . .	32
5.3.10	Hjælp . . . . .	33

<b>6 Elektronisk post</b>	<b>35</b>
6.1 Mutt . . . . .	35
6.1.1 Hvordan starter man mutt? . . . . .	35
6.1.2 Hvordan sender man elektronisk post fra mutt? . . . . .	36
6.1.3 Hvordan svarer jeg på elektronisk post i mutt? . . . . .	36
6.1.4 Diverse detaljer . . . . .	36
6.2 Andre programmer . . . . .	37
6.3 Spam . . . . .	37
<b>7 Tilgang til ØLs system udefra</b>	<b>39</b>
7.1 Logge ind på systemet udefra . . . . .	39
7.1.1 Microsoft Windows . . . . .	39
7.1.2 Un*x . . . . .	40
7.2 Kopiere filer til og fra brugerkonti på ØL . . . . .	40
7.2.1 Microsoft Windows . . . . .	41
7.2.2 Un*x . . . . .	42
<b>8 WWW på ØL</b>	<b>43</b>
8.1 ØLs websystem . . . . .	43
8.1.1 Webmail . . . . .	43
8.1.2 Administration af konto . . . . .	44
8.2 Personlig website . . . . .	44
8.2.1 Scripts . . . . .	44
<b>9 Printersystemet</b>	<b>45</b>
9.1 Hvilke printere er til rådighed? . . . . .	45
9.2 Hvordan udskriver man? . . . . .	45
9.3 Hvad er der i printer køen? . . . . .	46
9.4 Hvordan sletter man sit printjob fra printer køen? . . . . .	46
9.5 Hvordan skifter man papir/toner i printeren? . . . . .	46

<b>10 Tekstbehandling og <math>\text{\LaTeX}</math></b>	<b>47</b>
10.1 Et kort $\text{\LaTeX}$ dokument . . . . .	48
10.2 Oversættelse og udskrift af $\text{\LaTeX}$ -dokumenter . . . . .	48
10.3 Grundlæggende teknikker . . . . .	50
10.3.1 Typografiske spidsfindigheder . . . . .	52
10.3.2 $\text{\LaTeX}$ og matematik . . . . .	52
10.4 Tips til rapportskrivning . . . . .	53
10.4.1 Indholdsfortegnelser . . . . .	53
10.4.2 Titelblad . . . . .	53
10.4.3 Krydsreferencer . . . . .	54
10.4.4 Fodnoter . . . . .	54
10.4.5 Skriftsnit . . . . .	54
10.4.6 Lister . . . . .	54
10.4.7 Tabeller . . . . .	55
10.4.8 Figurer . . . . .	56
10.4.9 Stavekontrol . . . . .	57
10.5 Referencer . . . . .	58
<b>11 Projekter</b>	<b>59</b>
11.1 Arbejdsgang . . . . .	59
11.1.1 Projektmapper . . . . .	59
11.1.2 Valg af programmer . . . . .	60
11.1.3 Valg af filer . . . . .	61
11.2 Grafikprogrammer . . . . .	61
11.2.1 xv . . . . .	62
11.2.2 GIMP . . . . .	62
11.2.3 xfig . . . . .	63
11.3 Viewere . . . . .	64
11.3.1 dvi-filer & xdvi . . . . .	65
11.3.2 ps-filer & gv . . . . .	65
11.3.3 pdf-filer & Acrobat Reader . . . . .	65

<b>12 Diverse</b>	<b>67</b>
12.1 Hjælp! . . . . .	67
12.2 Edbafdelingens hjemmeside . . . . .	67
12.3 Kort om sikkerhed . . . . .	67
12.4 Brug af bærbar på systemet . . . . .	68
12.5 HJÆLP – jeg kan ikke logge ind! . . . . .	68
12.6 Trådløst netværk på HCØ . . . . .	68
12.7 Personoplysninger . . . . .	69
12.8 UNIX på sin egen PC . . . . .	69
12.9 UNIX programmer under Microsoft Windows . . . . .	70
<b>II Avancerede funktioner</b>	<b>71</b>
<b>13 Kommandofortolkeren</b>	<b>73</b>
13.1 Indtastning af kommandoer . . . . .	73
13.1.1 Redigering på kommandolinien . . . . .	73
13.1.2 Kommandohistorie . . . . .	73
13.1.3 Automatisk udfyldning . . . . .	74
13.1.4 Alias . . . . .	74
13.2 Ind- og uddata til programmer . . . . .	75
13.3 Jobstyring . . . . .	76
13.3.1 Processer i baggrunden . . . . .	76
13.3.2 Processer i forgrunden . . . . .	76
13.3.3 Jobkontrol . . . . .	77
13.4 Søgning . . . . .	77
13.4.1 Find . . . . .	77
13.4.2 Grep . . . . .	78
13.5 Pipes . . . . .	79
13.5.1 Eksempler på pipes . . . . .	79
13.6 Kommandofiler . . . . .	80
13.7 Konfiguration . . . . .	81
13.8 Referencer . . . . .	81

<b>14 Tekstredigeringsværktøjer</b>	<b>83</b>
14.1 Avanceret brug af Emacs . . . . .	83
14.1.1 Tilpasning af Emacs . . . . .	83
14.1.2 Smarte redigeringsfaciliteter . . . . .	83
14.1.3 $\text{\LaTeX}$ -mode . . . . .	84
14.1.4 I-spell . . . . .	85
14.2 Referencer . . . . .	86
<b>15 Versionskontrol</b>	<b>87</b>
15.1 Introduktion til versionskontrol . . . . .	87
15.2 CVS, en introduktion . . . . .	87
15.3 Grundliggende CVS . . . . .	88
15.3.1 Oprettelse af et repository . . . . .	88
15.3.2 Udtræk fra repository . . . . .	88
15.3.3 Tilføjelse af filer til projektet . . . . .	89
15.3.4 Ændringer af filer i projektet . . . . .	89
15.3.5 Opdatering . . . . .	90
15.3.6 Sletning af filer fra repository . . . . .	90
15.3.7 Konflikter i forbindelse med ændringer i filer . . . . .	90
15.4 Brug af CVS på systemet . . . . .	91
15.4.1 Repository . . . . .	91
15.4.2 Oprettelse af ssh-nøgle . . . . .	91
15.4.3 Adgang til cvs-repositories på andre konti . . . . .	92
15.5 Avanceret brug af CVS . . . . .	92
15.5.1 cvs diff . . . . .	92
15.5.2 Udtræk af ældre revisioner af en fil . . . . .	93
15.6 Opsætning . . . . .	93
15.7 Adgang til repository på systemet over SSH . . . . .	93
<b>A Kommandoreference</b>	<b>95</b>
A.1 Kommandoer opdelt efter emne . . . . .	95
A.2 Kommandoer i alfabetisk orden . . . . .	95
<b>B Regler for edb-systemet</b>	<b>101</b>
B.1 Regler . . . . .	101
B.2 Systemadministratorens rettigheder . . . . .	102
<b>C Emacs tastaturreference</b>	<b>103</b>
<b>D Vi tastaturreference</b>	<b>107</b>
<b>Litteratur</b>	<b>109</b>



# Forord

Dette er manualen til computersystemet som anvendes af Ørsted Laboratoriet og dele af Kemisk Institut. Meningen med den er dels at opfylde et længe forsømt behov for dokumentation af systemet, dels at give brugerne en chance for at lære at bruge systemet.

Bogen er opdelt i to dele, en grundlæggende og en avanceret. I den grundlæggende del beskrives ting fra det helt basale, såsom at logge ind på systemet, op til at kunne producere en afleveringsopgave på systemet fra start til slut. Denne del er ment som det basale niveau, som man er nødt til at kende for at kunne bruge systemet iløbet af en uddannelse på KU. Den avancerede del beskriver omvendt ting, som man fint kan overleve uden, men som kan gøre systemet urimeligt meget nemmere at anvende effektivt, hvis man bruger det meget.

Vi har ikke lavet små øvelser eller opgaver undervejs, men har til gengæld forsøgt at eksemplificere, hvor vi har fundet det nødvendigt. Det anbefales at man læser bogen igennem og undervejs (eller bagefter) sætter sig foran en terminal og prøver tingene i praksis – det er den eneste rigtige måde at lære det på, og desuden er terminalrummene et glimrende sted at finde hjælpsomme personer, der kan forklare tingene, hvis man har et problem.

Specielt i de første kapitler vil der være en del referencer fremad i bogen, særligt til afsnittet om kommandofortolkeren, eller xtermen (afsnit 4): Dette er en konsekvens af, at mange ting på linux kræver en hvis basal viden, og indtil man har den, kan man ikke ret meget, men når man når et vist niveau, bliver det meste pludseligt ret nemt: Hold derfor fast i de første par afsnit, og lad dig ikke slå ud af, at du ikke forstår alt i første omgang - det vil (forhåbentligt) blive klart senere.

## Kort om notation

Kommandoer som man selv skal indtaste skrives i teksten som f.eks:

```
datamat > lpr master.dvi
```

“datamat” er navnet på den datamat man bruger, > angiver, at brugeren kan indtaste kommandoer og den tekst, der er sat med dette skriftsnit angiver brugerens inddata.

Tastetryk angives undervejs omkranset af  $\langle \rangle$ , som i  $\langle c \rangle$  for “tryk på c-tasten”. *Kombinerede tastetryk* (hvor man holder første tast nede, mens man trykker på anden tast) angives som f.eks.  $\langle \text{ctrl-c} \rangle$  for kombinationen “hold Control-tasten nede og tryk på c-tasten”,  $\langle \text{ctrl-alt-c} \rangle$  for “hold Control-tasten, og samtidigt alt-tasten nede og tryk på c-tasten”, eller  $\langle \text{ctrl-c a} \rangle$  for “hold Control-tasten nede, tryk på c, slip Control-tasten og tryk på a”.

På side 109 findes en litteraturliste med de bøger vi refererer til undervejs – dette skrives i teksten som “se [x]”, hvor “x” er et nummer i litteraturlisten.

Mange af de programmer og kommandoer der bruges på systemet, har deres egen manual liggende på systemet i form af *manualsider*. Disse er opdelt i forskellige *sektioner*, så når vi refererer til

en kommando, vil vi ofte angive hvilken sektion kommandoens manualsider kan findes på (hvis ikke der er mere end én manualsider med det navn, behøver man ikke angive sektion). Feks. vil vi skrive: “se kommandoen `ls(1)`”, hvor man så kan se manualsiden med kommandoen

```
datamat > man ls
```

Se mere herom i afsnit 4.4.

## Opfordring til feedback

Generelt vil vi meget gerne have feedback på netop *din* oplevelse af denne bog: Var den for nem? For svær? For overfladisk? Manglede der emner? Har du forslag til bedre eksempler eller bedre opbygning? Kommentarer, rettelser og forslag kan sendes til emailadressen `fejlfys.ku.dk`.

Kursusbogen ligger på systemets hjemmeside, hvor man kan finde nyeste udgave. Se:

```
http://www.fys.ku.dk/edb/manual
```

## Bogens oprindelse

Denne manual er tilblevet ved at stjæle med arme og ben i DIKU's kursusbog 2, 3. udgave, for derefter at tilrette den til Ørsted Laboratoriets system. Således er visse kapitler næsten kopieret herfra, mens andre er bygget fra bunden. DIKU's edbafdeling, og i særdeleshed de to oprindelige forfattere<sup>1</sup> har gjort sig fortjent til stor tak i denne sammenhæng, men vær opmærksom på at denne udgave af bogen vedligeholdes separat af systemadministrationen på Ørsted Laboratoriet - alle henvendelser om den bedes derfor rettet til `fejlfys.ku.dk`.

---

<sup>1</sup>Søren Debois og Jesper Holm Olsen

**Del I**

# **Basale funktioner**



# Kapitel 1

## Konto på Ørsted Laboratoriet

De fleste, der læser denne manual, vil allerede på forhånd have en konto på Ørsted Laboratoriets computer-systemet (ØLs system); men hvis du ikke har en, eller er i tvivl skal du henvende dig på studiekontoret på 1. sal i D-bygningen (flytter ned i sydenden af vandrehallen). Husk at medbringe studiekort.

Men hvad er egentligt en "konto"? En konto består af tre dele: retten til at bruge systemet (login og password), en mængde diskplads<sup>1</sup> og en emailadresse<sup>2</sup>. Personer, der har en konto, betegnes *brugere*, og på ØL-systemet har alle brugere som udgangspunkt de samme rettigheder.

### 1.1 Datalokaler

Datalokaler På HCØ og NBI er der opstillet computere i datalokaler til de studerende. Datalokalerne benyttes også til undervisning og på disse tidspunkter må de øvrige brugere benytte et af de andre datalokaler. Undervisningstidspunkterne er opslået uden for datalokalerne. Computerne i de forskellige lokaler er ikke ens og egenskaberne ved de forskellige datalokaler ses af nedenstående skema:

navn	placering	antal computere	åbningstider	drev
a-lokalet	A112, vandrehallen	16 maskiner	døgnåben	cd + brænder, disketter, usb
s-lokalet	D315, d-bygningen	14 maskiner	8-18	cd + brænder, usb
n-lokalet	HA1, NBI	16 maskiner	8-22	cd, disketter
k-lokalet	A108, vandrehallen	15 maskiner	primært undervisning	cd, disketter

Figur 1.1: Det skal bemærkes at der er to typer computere i a-lokalet, og derfor er det ikke alle computerne, der har de angivne drevtyper.

Hvis der skal udføres beregninger, der kræver mange computere samtidig, må kun computerne i s-lokalet benyttes, og processerne skal *nices*<sup>3</sup>. Desuden skal beregningerne udføres så de ikke generer de daglige brugere. Hvis der hærsker tvivl kontakt da [fej1@fys.ku.dk](mailto:fej1@fys.ku.dk).

### 1.2 Log ind

Brugere på ØL-systemet har mulighed for at *logge ind* på samtlige linux-maskiner i D-bygningen samt på alle computerne i de datalokaler, der er nævnt i sidste afsnit. Der er ingen begrænsninger

<sup>1</sup>Til at starte med får hver bruger 150 MB diskplads.

<sup>2</sup>Både post, der sendes til [brugernavn@fys.ku.dk](mailto:brugernavn@fys.ku.dk) og [brugernavn@nano.ku.dk](mailto:brugernavn@nano.ku.dk), modtages af brugerne.

<sup>3</sup>Kør kommandoen "man nice" for at få mere at vide om nice.

på hvor mange, der kan logge ind på en computer samtidig.

På hver computer er der 7 konsoller man kan logge ind på. En grafisk og resten tekstbaseret. Ved at trykke på <ctrl-alt-F>1 kommer den første tekstbaseret konsol frem. <ctrl-alt-F>2, <ctrl-alt-F>3, <ctrl-alt-F>4, <ctrl-alt-F>5 og <ctrl-alt-F>6 giver de resterende tekstbaseret konsoller. Det er muligt at være logget ind på flere tekstbaseret konsoller samtidig, det skal blot huskes at logge ud fra *alle* konsollerne når man er færdig med at benytte computeren.

Der er også mange, der logger ind over netværket for at lave udregninger på flere maskiner samtidig. Så selvom der ikke sidder nogen foran en computer kan den godt være i brug. For ikke at brugerne generer hinanden er det blevet vedtaget, at det ikke er tilladt at genstarte eller slukke for computerne.

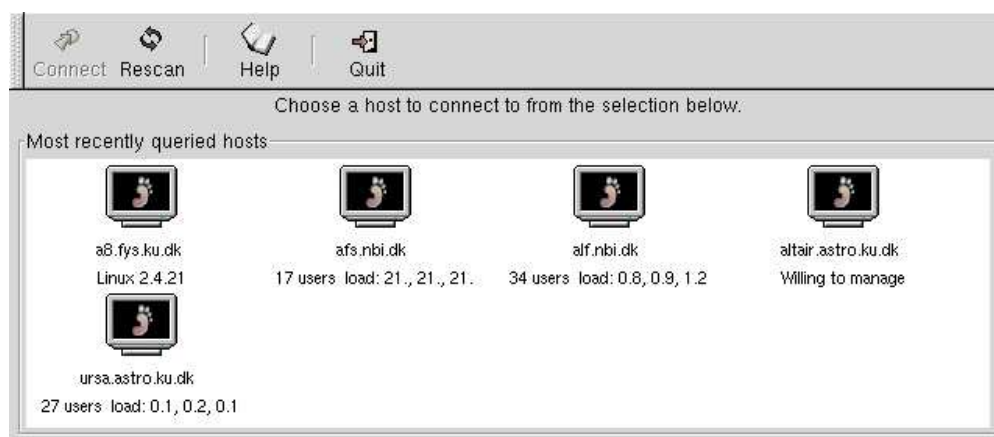
### 1.2.1 Tekstbaseret login

For at logge ind på en af de tekstbaseret konsoller skal der blot indtastes login efterfuldt af <enter> og password efterfuldt af <enter>, bemærk at man ikke kan se at man skriver noget mens man taster sit password ind.

### 1.2.2 Grafisk login

Der er kun én grafisk konsol og den fås frem ved at trykke <ctrl-alt-F>7 <sup>4</sup>.

For at logge ind grafisk skal man gennem to login skærme. Den første skærm giver en mulighed for at vælge hvilken computer man ønsker at logge ind på. Som udgangspunkt skal man altid logge ind på den computer man sidder foran. Navnet på den computer man sidder ved står på et klistermærke på computeren. Computerne i s-lokalet hedder s?, hvor "?" er et tal fra 1 til 14, og maskinerne i a-, k-, og n-lokalerne er navngivet tilsvarende. For at vælge en maskine skal man enten dobbeltklikke på ikonet, eller klikke en gang på ikonet og derefter trykke connect på knappen.



Figur 1.2.2: Her ses den første loginskærm.

Herefter vil loginskærm nummer to komme frem. Her skal du indtaste dit login og password. Der er desuden menuer, der er beregnet til at vælge sprog og windowmanager. Undlad at vælge noget i windowmanagermenuen, da det til tider har utilsigtede resultater<sup>5</sup>. Hvis der ikke ændres i disse menuer vil engelsk blive sproget og kde vil blive display-manageren, hvilket er det vi anbefaler.

<sup>4</sup>Til tider skal F8, F9, F10, F11 eller F12 benyttes istedet.

<sup>5</sup>I kapittel 2 vil det blive beskrevet hvordan der skiftes windowmanager.

## 1.3 Log ud

Det er *meget* vigtigt at logge ud når man er færdig med at benytte en computer, hvis man glemmer det kan fremmede læse ens mail eller slette det der ligger i ens hjemmekatalog.

For at logge ud fra de tekstbaseret konsoller skal der skrives kommandoen `exit` eller trykkes `<ctrl-d>`.

For at logge ud fra den grafiske konsol skal logout-knappen i k-menuen benyttes. Hvis skærm-billedet er frosset kan der trykkes `<ctrl-alt-backspace>`, hvorefter man bliver logget ud. Sidst nævnte metode at logge ud på virker også hvis andre har logget ind og har sat pauseskærm på. Vis dog hensyn, og log kun andre brugere ud, hvis det er nødvendigt.

## 1.4 Pauseskærm

Generelt skal man logge sig ud hver gang man forlader computerne; men hvis det kun er en kortere pause kan pauseskærmen benyttes. Bemærk at der findes to typer pauseskærme: en der kræver password for at slå den fra, og en der ikke kræver password. Det er *kun* den pauseskærm, der kræver password, som må benyttes når computeren forlades kortvarigt.

Der er to måder at aktiverer pauseskærmen med lås på.

- Vælg ikonet `Lock Screen` i k-menuer.
- Skriv kommandoen `lock` i en terminal.

For at låse pauseskærmen op skal man indtaste sit password efterfulgt af `enter`.

## 1.5 Password

Umiddelbart efter at havde modtaget sin konto skal man ændre sit password. Passwordet må *ikke* være et ord. Det skal være en tilfældig kombination af tal, specialtegn<sup>6</sup>, store og små bogstaver.

Dit password er strengt personligt og må ikke skrives ned. Hvis man glemmer sit password skal man henvende sig på studiekontoret på ØL, hvor de kan udlevere et nyt.

Når man har haft et password i 240 dage udløber det og skal udskiftes til noget nyt. Dette vil man blive informeret om en gang dagligt via mail når der er mindre en 60 dage til at passwordet udløber.

Der er to måder at skifte password på.

- Via en `xterm`<sup>7</sup> kan kommandoen `passwd` benyttes. Man skal intaste det gamle password en gang efterfulgt af `enter`. Herefter skal det nye password indtasten to gange hver gang efterfulgt af `enter`.
- Via internettet på hjemmesiden <http://www.fys.ku.dk/password>.

Begge ovenstående metoder til at skifte password på resulterer i at man overalt på linuxsystemet skal benytte det nye password. Bemærk at passwordet man benytter i windows (på terminalserveren) også er blevet ændret.

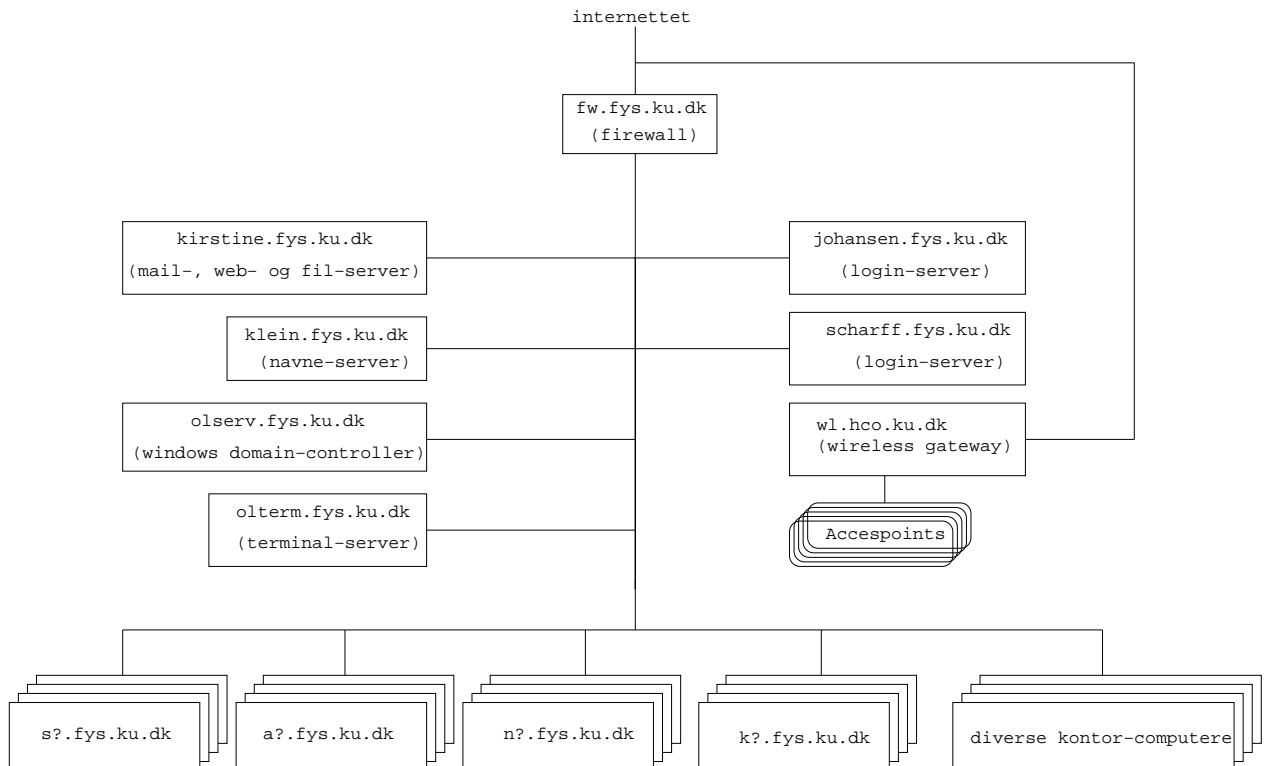
---

<sup>6</sup>Specialtegn er fx `!`, `%`, `&`

<sup>7</sup>Se afsnit 2.3

## 1.6 Netværkstrukturen

Computersystemet på ØL er sammensat af servere og *klienter* <sup>8</sup>. Mellem serverne og klienterne er der et *netværk*, hvilket vil sige en masse kabler, der forbinder alle computerne og printerne.



Figur 1.6:Oversigt over netværket på ØL.

Idet alle computerne er forbundet er det muligt at have alle brugerfilerne liggende på en central server. Filerne kan stadig tilgås fra alle klienterne og det er ikke muligt at se om en fil ligger på den computer man sidder ved eller ej.

Som bruger kan man gemme sine filer tre steder:

- `~bruger/`  
Her ligger ens *hjemmekatalog*. Hjemmekatalogerne ligger på en server så uanset hvilken computer man logger ind på vil hjemmekataloget altid være tilgængeligt. Som udgangspunkt må der ligge 150MB; men der kan gives ekstra diskplads. For at finde ud af hvor meget der ligger i en hjemmekatalog benyttes quotasystemet, se afsnit 1.6.1.
- `/tmp/` eller `/net/computernavn/tmp/`  
Her kan alle gemme midlertidige filer. Filerne ligger lokalt på den computer man sidder ved når stien `/tmp/` benyttes. For at få adgang til `/tmp/` på en anden computer benyttes `/net/computernavn/tmp/`. Hvis man fx sidder ved computeren `s2` og ligger noget i `/tmp/`, og derefter sætter sig ved computeren `s5`, da vil filerne ligge i `/net/s2/tmp/`. Bemærk at når en computer bliver genstartes vil indholdet af `/tmp/` blive slettet.

<sup>8</sup>En klient er de maskiner som brugerne har fysisk adgang til.



- /var/mail/bruger

Her har alle brugere én fil med samme navn som deres login. Dette er den enkelte brugers emailbox. Filen må ikke være større end 250 MB og hver mail må ikke være større end 10 MB. Filen ligger på mail-serveren så uanset hvilken computer man logger ind på vil mailfilen altid være tilgængelig.

### 1.6.1 Quota

diskkvote (eng. *quota*) *quota-systemet* holder styr på hvor meget hver enkelt bruger har liggende. Filer der ligger i /tmp/ og mailfilen (/var/mail/brugernavn) tæller ikke med; men *alt* andet gør.

Der benyttes to typer grænser. Første grænse kaldes *quota* og siger hvor meget man maksimalt *bør* have liggende - denne grænse kan man godt overskride. Næste grænse hedder *limit*, denne grænse fortæller hvor meget det er muligt at have liggende som yderste maksimum. Det er ikke muligt at gemme mere når man når denne grænse. Man må kun overskride sin quota 7 dage af gangen, og hvis man overskrider sin quota længere tid vil man ikke kunne gemme filer indtil man har slettet noget og kommer ned under ens quota igen. Der vil på intet tidspunkt blive slettet noget automatisk på folks kontoer uanset hvor længe og med hvor meget de overskrider deres quotaer.

Når man overskrider sin quota vil man blive informeret via email en gang dagligt, så man kan rydde op i sit hjemmekatalog.

For selv at finde ud af hvor meget man har liggende benyttes kommandoen *quota* i en xterm (se afsnit 4). Herefter vil følgende output komme.

```

o3:~> quota
Disk quotas for user testuser (uid 3181):
  Filesystem  blocks    quota   limit   grace   files   quota   limit   grace
kirstine:/bdisk/00      3896  150000  300000             704      0      0
kirstine:/bdisk/98      3896  150000  300000             704      0      0
kirstine:/bdisk/03      3896  150000  300000             704      0      0
kirstine:/bdisk/04      3896  150000  300000             704      0      0
o3:~> 

```

Figur 1.2.2: Her ses hvad brugeren testuser fik frem efter at havde skrevet *quota* i en xterm.

Øverst på figur 1.6.1 ses det at brugeren hedder testuser og derunder er der en række søjler med information. Søjlen der hedder *filesystem* kan man ignorere og informationerne på linierne med forskellige filesystemer betyder alle det samme. Søjlen *blocks* viser hvor meget brugeren har liggende i "blocks" (1024 blocks = 1MB), i dette tilfælde drejer det sig om ca. 4 MB. Søjlen til venstre hedder *quota* og viser brugerens quota, for denne bruger ca. 150 MB. Søjlen efter viser *limit*, her er limit ca. 300 MB, hvilket er standard for nye brugere. Søjlen *grace* fortæller hvor mange dage man må overskride sin quota. Når man ligger under quotaen vises der 0 og ellers vises en nedtælling fra 7 til 0, alt efter hvor langt tid man har overskredet quota grænsen. De sidste søjler er uden betydning på dette system.

Hvis man ønsker at få forhøjet sin quota skal man sende en mail til [fejl@fys.ku.dk](mailto:fejl@fys.ku.dk).

### 1.6.2 Drev

På trods af at mange af de nye maskiner har cd-brændere og usb-porte kan dette ikke benyttes endnu - edbafdelingen har begrænsede resourcer, men det vil blive sat op så snart det er muligt.

#### cd-drev

Der er to måder at få adgang til sit cd-drev på:

- Der klikkes på cd-ikonet i KDE på skrivebordet. Herefter vil der komme et vindue frem, der viser indholdet af den cd, som sidder i cd-drevet. Når man er færdig med at benytte cd'en eller ønsker at sætte en ny i, skal man sørge for at man ikke har nogle vinduer eller xterm'er, der viser indholdet af cd'en. Herefter højreklikkes på cd-ikonet og der vælges `unmount`. Hvis man ikke `unmount`er cd-drevet kan man *ikke* åbne drevet og få cd'en ud.
- xterm'er (beskrives i afsnit 4) kan også benyttes til at tilgå cd-drevet. Først sikrer man sig at det aktive katalog ikke er `/cdrom/` eller et underkatalog af dette. Herefter skrives `mount /cdrom`. Da vil indholdet af cd'en være at finde i `/cdrom`. Når man er færdig med at benytte cd'en går man ud af `/cdrom` og skriver `umount /cdrom`, og herefter kan cd-drevet igen åbnes.

#### diskette-drev

Der er tre måder at benytte diskette-drevet på:

- Der klikkes på floppy-ikonet i KDE på skrivebordet. Herefter vil der komme et vindue frem, der viser indholdet af disketten. Når man er færdig med at benytte disketten eller ønsker at sætte en ny i, skal man sørge for at man ikke har nogle vinduer eller xterm'er, der viser indholdet af disketten. Herefter højreklikkes på diskette-ikonet og der vælges `unmount`. Hvis man ikke `unmount`er diskettedrevet kan diskette-drevet ikke benyttes efterfølgende.
- xterm'er kan også benyttes til at tilgå diskette-drevet. Først sikrer man sig at man ikke står i `/floppy/`. Herefter skrives `mount /floppy`. Da vil indholdet af disketten være at finde i `/floppy`. Når man er færdig med at benytte disketten går man ud af `/floppy` og skriver `umount /floppy`.
- *mtools* er en række værktøjer, der kan benyttes til at tilgå disketter, som er i MS-DOS format. Dette er et værktøj, som bruges fra en xterm (se 4), og anvendelsen af det kan vises ved følgende tre eksempler:

For at se hvad, der ligger på disketten benyttes `mdir`

For at kopierer alt fra diskettedrevet ned på ens konto benyttes: `mcopy a:/* ~/`

For at kopierer alt, der ligger i biblioteket `~/test/` ned på en diskette benyttes: `mcopy ~/test/* a:/`

For en fuld beskrivelse af *mtools* henvises til *man*-siden<sup>9</sup>.

---

<sup>9</sup>For at se *man*-siden køres komandoen "*man mtools*" i en xterm.

# Kapitel 2

## Xwindows

Tekstbaserede konsoller er særdeles nyttige; men de fleste, der arbejder med computere er dog også interesseret i at have en grafisk brugerflade til rådighed. Der findes styresystemer, der kun tilbyder én brugerflade<sup>1</sup>, som så er grafisk; men i linux er der flere valg.

### 2.1 Displaymanager og windowmanager

En grafisk brugerflade i linux består af flere lag. Det nederste lag er *Xwindows*, hvilket giver en samlet betegnelse for den grafiske brugerflade i linux. Det næste lag er displaymanageren. Her på ØL benyttes GnomeDisplayManager (gdm); men til dagligt har det næsten ingen praktisk betydning hvilken displaymanager, der benyttes. Det eneste tidspunkt man ser displaymanageren på er når der logges ind, hvor displaymanageren står for de to logindskærme.

Laget oven på displaymanageren hedder windowmanageren hvilket er det man ser på hver gang man er logget ind. På ØL's system kan man vælge mellem *KDE*, *gnome* og *fvwm95* som windowmanagere. Denne bog beskriver hvordan KDE benyttes; men det er også muligt at benytte en anden windowmanager, hvis man ønsker det.

- *KDE* Windowmanageren, som vi anbefaler. På forhånd har systemadministrationen lavet en fornuftig default-opsætning. KDE minder meget om Microsoft Windows, og har grafiske vinduer til opsætning. Dog er KDE noget langsom i forhold til mere simple windowmanagere, hvilket på maskinerne i datalokalerne ikke kan ses når først man er logget ind.
- *gnome* Minder meget om KDE; men der er ikke redigeret i opsætningen, så dette skal man afsætte noget tid til, hvis man vælger gnome.
- *fvwm95* Meget simpel windowmanager. Er ikke lige så flot som KDE og gnome; men kører hurtigere, fx er opstartstiden  $\frac{1}{6}$  af KDE's. Alt opsætning sker i konfigurationsfiler, der tager noget tid at gennemskue. Hvis man ikke er interesseret i andet end at kunne åbne grafiske programmer og ellers udfører alt andet i en xterm, så er fvwm95 et godt valg.

For at skifte mellem windowmanagerne skal man editere<sup>2</sup> filen `~/.xsession`. Til at starte med bør der stå `exec startkde` i filen, hvilket starter KDE op. For at starte gnome op i stedet skal `startkde` udskiftes med `gnome-session`. Og endelig skal der i `~/.xsession` skrives `exec fvwm95` for at starte fvwm95 op. Bemærk at de opsætninger man laver af de enkelte windowmanagere bliver ikke slettet fordi man starter en anden windowmanager op.

---

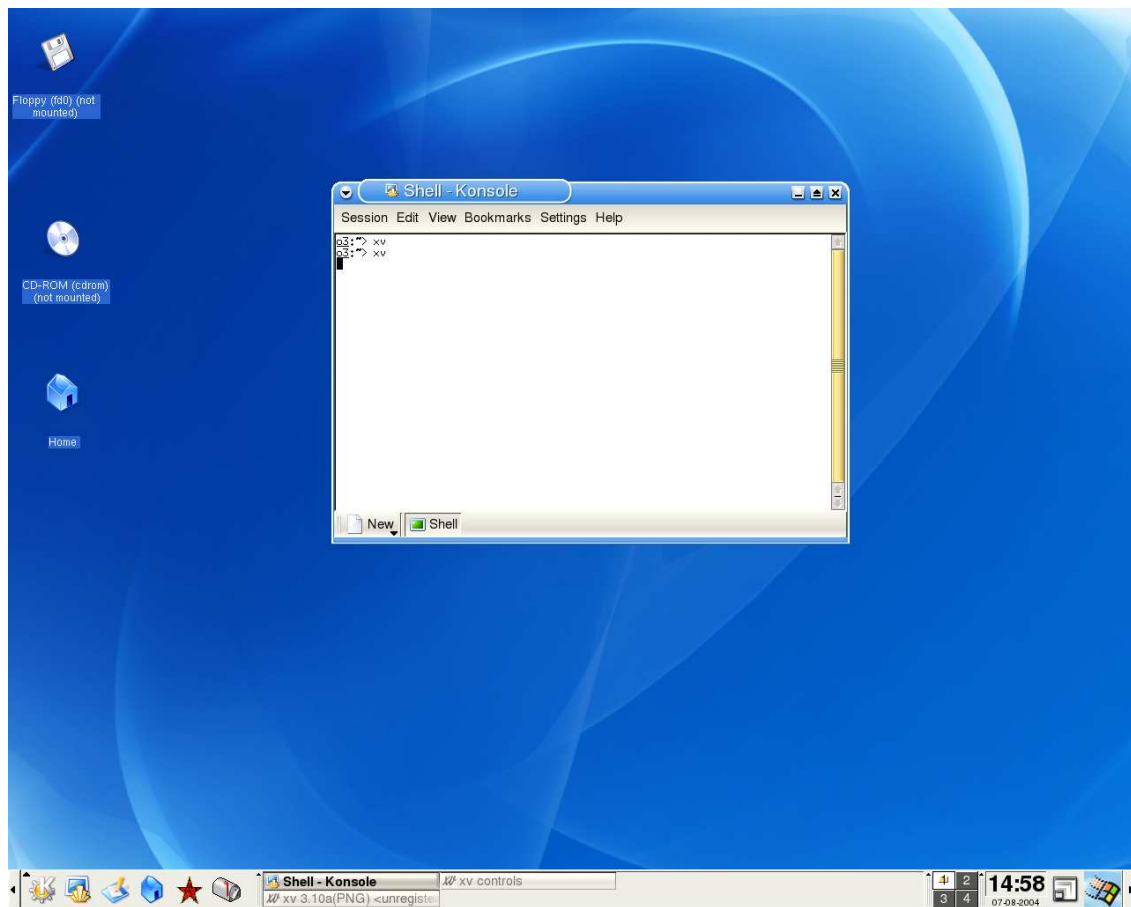
<sup>1</sup>Microsoft Windows er et eksempel herpå

<sup>2</sup>se afsnit 5

## 2.2 KDE

KDE minder til forveksling om Microsoft Windows med et *skrivebord*, ikoner, startmenu og hvad der ellers plejer at være. Så når man er i tvivl om hvordan noget virker er det ofte succesfuldt at prøve sig frem.

## 2.3 Brugerfladen



Figur 2.3: Her ses brugerfladen i KDE.

Herefter vil de enkelte dele af brugerfladen blive gennemgået:

- *skrivebordet* På skrivebordet kan der placeres filer. Desuden er der genveje til cdrom-drevet og deskette-drevet<sup>3</sup>. Ikonet home viser indholdet af en hjemmekatalog.
- *menubjælken*, der befinder sig i bunden af skærmen kan skjules ved at klikke på en af de to pile, der er i enderne af den. Fra venstre mod højre indeholder menubjælken følgende.
  - *K-menuen*, der giver adgang et udvalg af programmerne på ØLs system. Specielt skal programmet Control Center under Settings fremhæves. Her er det muligt at ændre på *alle* indstillinger i KDE.

<sup>3</sup>Brugen af disse er beskrevet i afsnit 1.6.2.

- *Shell - Konsole*, med et ikon af en skærm med en musling foran. Shell-Konsole giver et vindue til at skrive komandoer i. Tidligere i teksten er et sådan vindue blevet kaldet en *xterm*, hvilket er et lignende vindue. Vi har valgt at benytte navnet *xterm* da det ikke kan forveksles med de grafiske konsoller (se afsnit 1.2), i praksis er der ingen forskel på *xterm*er, Shell-Konsole'er eller lignende vinduer.
- *skrivebord-ikonet*, der viser en blyant, der skriver på et stykke papir. Ved at klikke her lukkes alle vinduer, der er åbne, således at man får adgang til selve skrivebordet. Ved at klikke på ikonet igen åbnes de vinduer, som blev lukket.
- *home-ikonet*, der viser et blå hus. Ved at klikke her fås et vindue frem, der viser indholdet af ens hjemmekatalog.
- *mozilla-ikonet*, en rød stjerne. Dette ikon starter webbrowseren mozilla op (se afsnit 8).
- *mutt-ikonet*. Ikonet er en postkasse og ved at klikke på den kommer mailprogrammet mutt frem (se afsnit 6.1).
- *Vindue-ikoner* placeres på det midterste af menubjælken. Alle vinduer, der åbnes, får deres eget ikon, således at man kan komme hen til et vindue ved at klikke på det tilhørende ikon.
- *4-desktops-ikonet*, som forestiller fire små desktops. De fire desktops kan benyttes samtidig, og man skifter imellem den, ved at klikke på en desktop.

Det sidste stykke af menubjælken er et ur og ikoner til terminalserveren, se afsnit 3.

### 2.3.1 Genvejstaster

genvejstaster i KDE Det er ikke nødvendigt at benytte genvejstaster for at udføre handlinger i KDE; men det går meget hurtigere hvis man gør. Genvejstasterne sættes op i *K-menuen* under *Settings* i *Control Center*. Inde i *Control Center* skal man gå ind under *Accessibility* og ind under *Keyboard Shortcuts*. Vi har valgt at fremhæve følgende tre tastaturgenveje:

**<ctrl-alt-b>**

- *ackspace Process Table* startes. Her er det muligt at lukke programmer, der er "frosset", og få overblik over hvilke processer, der kører på maskinen man sidder ved.
- **<ctrl-alt-piletast>** Herved skiftes til den desktop der befinder sig i pilens retning.

**<alt-F>**

- 2 Der kommer et lille vindue op hvor der kan instastes en kommando. Dette kan benyttes til at starte et program. Det lille vindue lukker af sig selv når man taster **<enter>**

### 2.3.2 Cut'n'paste

En af de smarte ting ved en grafisk brugerflade, er muligheden for at kopiere tekst fra et vindue over i et andet, en ting som de fleste nok er prøvet med Microsoft Windows. Dette kan man naturligvis også i xwindows, men det foregår på en lidt anden måde: Først markerer man den tekst, som man gerne vil kopiere, ved at holde venstre museknap nede. Derefter kan man indsætte denne tekst i et andet vindue, ved at flytte tekst-markøren til det sted, hvor teksten ønskes indsat, og trykke på midterste museknap<sup>4</sup>. Ønskes teksten indsat flere steder, kan man blot trykke på midterste knap igen - den indsætter simpelthen det sidste, som blev markeret.

<sup>4</sup>hvis man har en mus med kun to knapper, kan man opnå samme effekt ved at trykke begge knapper samtidigt



## Kapitel 3

# Microsoft Windows

Langt de fleste programmer, der er tilgængelige til Windows findes også til linux, dog er der undtagelser. For at løse dette problem er der på ØL-netværket en windows-terminalserver.

Terminalserveren gør det muligt fra en linuxmaskine at starte windows op i et vindue og derved at køre de programmer, som kun kan køre i Microsoft Windows.

### 3.1 Log ind og log ud

Nede i højre hjørne af skærmen er der to ikoner, der giver adgang til terminalserveren.



Figur 3.1: Der er to pile, der peger på de ikoner, som starter Windows.

Ikonet, der ligner et Windowsflag starter Windows op i en fuldskærm version, mens ikonet, der forestiller to vinduer inden i hinanden, starter Windows op i et vindue<sup>1</sup>.

Når Windows er startet kommer der en login-skærm frem, hvor der skal benyttes samme login og password som på linuxmaskinerne.

Efter at havde benyttet terminalserveren skal man huske at logge ud, hvorefter man havner i linux igen. Primært vil terminalserveren blive benyttet til Officepakken, og det frarådes at benytte terminalserveren unødvendigt, da det går meget langsomt, hvis for mange brugere benytter den samtidig.

### 3.2 Brug af terminalserveren

Terminalserveren virker som en almindelig Windows-computer med undtagelse af at hverken diskette- eller cdrom-drev virker.

---

<sup>1</sup>Størrelsen af vinduet kan ikke ændres.

Når der skal gemmes filer skal det ske i username on 'samba.fys.ku.dk' under My Computer, hvilket er en genvej til ens hjemmekatalog. Filer kan også placeres på skrivebordet i Windows hvorefter de kommer til at ligge i kataloget `~/winprofile/Desktop`.

Desuden opfører tastaturet sig lidt underligt. <Alt Gr> tasten virker ikke og istedet skal <ctrl - alt> benyttes.



## Kapitel 4

# Kommandofortolkeren

Det vigtigste program på computerne på ØLs system er uidentivl kommandofortolkeren, der som standart er sat til `xterm(1)` . Den startes op med det ikon nr. 2 fra venstre i bunden af skærmen, og vinduet som startes ved at klikke på dette, vil indeholde en linie, som ser således ud:

```
datamat >
```

Denne tekst kaldes kommandoprompten, og den skal forstås som “kommandofortolkeren er klar til at modtage kommandoer, der vil blive udført på datamaten datamat”. I de fleste tilfælde (alt efter hvad man valgte i login vinduet), vil “datamat” være den computer, man sidder ved.

I en tid, hvor peg-og-klik har gjort datamaten til et værktøj, der benyttes af lægfolk, synes det måske forældet eller ligefrem arkaisk at have en tekstbaseret grænseflade. Forhåbentlig vil det længere inde i dette kapitel blive klart, at netop den tekstbaserede grænseflade giver fleksibilitet og muligheder, som man ikke med rimelighed kan håbe på at opnå i en grafisk brugergrænseflade. Men det kommer vi til; lad os først gøre os bekendt de mest brugte kommandoer. I det følgende vil vi komme til at strejfe en del emner og koncepter, der har konsekvenser for langt mere end blot kommandofortolkeren, men som det ligger ligefor at introducere i dette kapitel.

Som ny UNIX-bruger kan man sagtens klare sig, uden at sætte sig grundigt ind i kommandofortolkeren. Gør man det ikke, afskærer man sig dog fra *mange* af UNIX-miljøets styrker. Læsere der føler, at de ikke behøver at vide mere end det allermest rudimentære, kan nøjes med at læse dette kapitel, men vi opfordrer til, at man også orienterer sig i kapitlet 13

For ikke at komplicere fremstillingen mere end højst nødvendigt, vil vi i det følgende undertiden tillade os lemfældig omgang med sandheden.

### 4.1 Filsystemet

Mange kommandoer tager filnavne som argumenter. Filsystemet i UNIX består af filer og kataloger organiseret i et træ. Under kørsel af kommandofortolkeren vil et bestemt katalog altid være udpeget til *arbejdskatalog*. Når kommandofortolkeren starter, vil det aktive katalog altid være brugerens *hjemmekatalog*, dvs. det katalog, der indeholder brugerens egne filer.

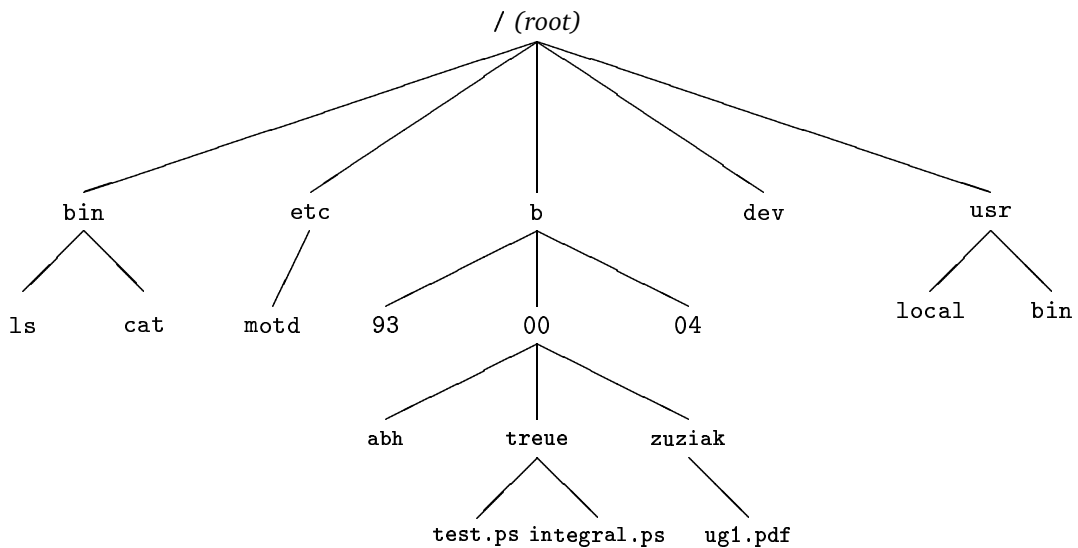
Man refererer til en fil ved at angive dens stinavn (eng. *pathname*), som består af navnene på de kataloger, man skal igennem for at komme frem til filen, samt til slut filens navn. Navnene adskilles af /. Stinavne kan være enten *absolutte* eller *relative*. Et absolut stinavn indeholder navnene på alle

kataloger mellem filsystemets rod (der i UNIX kaldes /) og filen selv. Et absolut stinavn begynder altid med /. Et relativt stinavn indeholder navnene på katalogerne mellem det aktive katalog og filen.

Eksempelvis refererer det absolutte stinavn `/etc/motd` (eng. message of the day) til filen `motd` i kataloget `etc`, som igen befinder sig i rodkataloget `/`. Hvis det aktive katalog f. eks. er `/b/00`, refererer stien `treue/test.ps` til filen med det absolutte stinavn

`/b/00/treue/test.ps`

Filsystemet kan illustreres sådan:



Bemærk, at ovenstående diagram kun skitserer *dele* af ØLs system.

Filsystemet i UNIX er versalfølsom (eng. *case-sensitive*). Det medfører at der er forskel på store og små bogstaver, når man refererer til kataloger og filer.

## 4.2 Kommandoer

Lad os forsøge en kommando:

```
datamat > pwd
/usr/local/sbin
```

Vi har her angivet den tekst, som brugeren tænkes at have skrevet, med dette skriftsnit, og kommandofortolkerens tekst med dette skriftsnit.

Kommandoen `pwd` angiver det aktive katalog ("`pwd`" står for **p**rint **w**orking **d**irectory). Når man udsteder en kommando, der arbejder på en eller flere filer, vil kommandoen lede efter disse filer i det aktive katalog, med mindre andet er angivet.

Lad os prøve at kopiere en fil; hvis du netop har fået din konto, har du nok ikke nogle filer liggende, som vi har lyst til at pille ved. Vi henter derfor en fil fra et andet katalog på systemet:

```
datamat > cp /usr/doc/FAQ/Linux-FAQ.gz foo
datamat >
```

Formatet for kommandoen `cp` er `cp <fra> <til>`, så vi kopierer her filen `Linux-FAQ.gz` fra kataloget `/usr/doc/FAQ/` til filen `foo` i det aktive katalog. Læg mærke til, at der ikke kommer noget egentligt svar fra kommandoen. Kommandoprompten kommer blot tilbage uden nogle beskeder à la `1 file(s) copied`. Under UNIX giver langt de fleste kommandoer kun uddata, hvis noget gik galt. Dette kaldes på engelsk *silent accept*.

### 4.2.1 Fil- og katalogmanipulering

Her er en liste over de mest almindelige kommandoer. En længere liste findes i appendiks A.

<code>pwd</code>	<b>print working dir.</b>	Viser det aktive katalog.
<code>cd &lt;katalog&gt;</code>	<b>change dir.</b>	Skifter det aktive katalog.
<code>ls</code>	<b>list</b>	Viser en liste af filer i det aktive katalog.
<code>ls -l</code>	<b>list long</b>	Viser en detaljeret liste af filer i det aktive katalog.
<code>rm &lt;filnavne ...&gt;</code>	<b>remove</b>	Sletter de angivne filer.
<code>cp &lt;fra&gt; &lt;til&gt;</code>	<b>copy</b>	Kopierer filen <fra> til filen <til>.
<code>mv &lt;fra&gt; &lt;til&gt;</code>	<b>move</b>	Flytter filen <fra> til filen <til>.
<code>mkdir &lt;navn&gt;</code>	<b>make dir.</b>	Opretter et nyt katalog ved navn <navn>
<code>rmdir &lt;navn&gt;</code>	<b>remove dir.</b>	Sletter kataloget ved navn <navn>.
<code>cat &lt;filnavne ...&gt;</code>	<b>concatenate</b>	Viser indholdet af <filnavne>.
<code>less &lt;filnavne ...&gt;</code>	<b>(efterfølger more)</b>	Bladrer i indholdet af <filnavne>.

Langt de fleste kommandoer er dokumenteret elektronisk. Den elektroniske dokumentation er inddelt i forskellige afsnit. Der er tradition for, at man, når man præsenterer en ny kommando, samtidigt angiver hvilken sektion den er dokumenteret i, så det vil vi også gøre. For fremtiden vil vi præsentere kommandoer som `cp(1)`, i stedet for blot `cp`. Vi kommer tilbage til den elektroniske dokumentation i afsnit 4.4.

### 4.2.2 Særlige kataloger

Der findes forkortelser for kataloger, som man ofte har behov for at tilgå:

<code>.</code>	(punktum)	Det aktive katalog
<code>..</code>	(2×punktum)	Folderen over det aktive katalog
<code>~</code>	(tilde)	Den aktive brugers hjemmekatalog
<code>~&lt;bruger&gt;</code>	(tilde-bruger)	Brugeren “bruger”s hjemmekatalog

Hvis man til eksempel ønsker at skifte det aktive katalog til brugeren `treue`’s hjemmekatalog, udsteder man kommandoen

```
datamat > cd ~treue
```

### 4.2.3 Usynlige filer

Kommandoen `ls` viser, som nævnt, en liste af filnavne. I UNIX betragtes filer, hvis navn starter med ‘.’, som “usynlige”, og vil *ikke* blive vist af `ls`. Sådanne filer kaldes på engelsk *dot-files* eller *hidden-files*, og de bruges mest af programmer, der behov for at gemme konfigurationsfiler eller lignende i ens hjemmekatalog. Til eksempel gemmer `mozilla(1)` ens `bookmarks` osv. i kataloget `.mozilla`. Hvis man ønsker at se alle filer, inklusive de usynlige, kan man give `ls` argumentet `-a` for `all`.

#### 4.2.4 Mønstre

Ønsker man at udpege en række filnavne, f. eks. som argumenter til en kommando, kan man gøre brug af forskellige “mønstre”. Mønstret `*` betyder “en vilkårlig tegnfølge” (inklusive den tomme) og mønstret `?` står for “netop ét vilkårligt tegn”. Kommandoen `ls foo*` vil således producere en liste af filnavne, der begynder med `foo`, mens `ls foo?` vil vise de af filnavne, der består af `foo` og et ekstra tegn. Brugen af mønstre kaldes på engelsk *globbing*.

Kommandoen `cp` ovenfor kan også kopiere flere filer ad gangen; hertil kan man anvende `cp` som

```
cp <fra1> <fra2> ... <katalog>
```

Da vil alle de første filer blive kopieret ind i den sidste, der så helst skal være en katalog. Hvis vi f.eks. har lavet en rapportopgave, der består af en masse forskellige filer, der ligger i kataloget `rapport`, kan vi ved hjælp af mønstre og dette særlige format for `cp` lave os en sikkerhedskopi:

```
datamat > mkdir rapport-backup
datamat > cp rapport/* rapport-backup
```

Hvis `rapport`-kataloget indeholder underkataloger, kan vi bede `cp` om at tage dem med ved at tilføje parameteren `-R` (**R**ecursive):

```
datamat > cp -R rapport/* rapport-backup
```

Flyt-kommandoen `mv` accepterer det samme format, så hvis vi vil lægge `tex`-kildeteksterne<sup>1</sup> til rapporten i et katalog for sig selv, kunne vi forsøge med

```
datamat > mkdir rapport-latex
datamat > mv rapport/*.tex rapport-latex
```

Med mønstre ser vi for første gang en facilitet, der kun vanskeligt lader sig forene med en grafisk brugerflade. Selvom visse moderne grafiske brugerflader har ligeså fleksible metoder til på en gang at udvælge mange filer, falder disse metoder aldrig flydende ind i resten af grænsefladen. I et tekstbaseret system bliver de derimod en del af den måde, som man udtrykker sig på.

### 4.3 Filrettigheder

Alle systemets brugere arbejder på det samme UNIX-system, så der er naturligvis behov for nogen beskyttelse. F.eks. skal studerende helst ikke kunne læse et kommende eksamenssæt, blot fordi de er lumske nok til at lede efter det i en forelærsers hjemmekatalog.

For at kunne give selektivt adgang til filer, har UNIX såkaldte rettigheder (eng. *permissions*). Vi har tidligere set kommandoen `ls`, der giver en oversigt over hvilke filer, der er i et katalog. Giver man `ls` flaget `-l`, får man detaljerede oplysninger om de enkelte filer. Lad os prøve at se på mit hjemmekatalog:

```
datamat > ls -l
-rwxr-xr-x   1 treue   users      1372 May 26   2003 livekeys
-rw-r--r--   1 treue   users     13824 Jan 27   2003 propaganda.sdw
-rw-----   1 treue   users    9334881 Jun 14   2003 treue20030614
[...]
```

<sup>1</sup>Der kommer mere om `.tex`-filer, `TEX` og `LATEX` i kapitel 10

Alle filer har under UNIX en ejer; ejerens brugerid (her “treue”) står i den tredje kolonne. Ligesom alle filer har en ejer, har hver fil også en gruppe; grupper lader udvalgte brugere deles om filer og kataloger, på systemet vil alle kontoer have den samme gruppe (nemlig “users”). EDB-afdelingen kan oprette nye grupper, men vi gør det kun, hvis det er *absolut* nødvendigt, hvad det næsten aldrig er. Gruppenavnet findes i fjerde kolonne. Herefter følger filens størrelse, dato og tidspunkt for sidste ændring og tilsidst filnavnet.

Filrettighederne kan læses ud af den første kolonne. Det første bogstav angiver, om filen er et katalog (d) eller en almindelig fil (-). De efterfølgende bogstaver angiver rettigheder. Rettighederne er grupperet tre og tre, således at de første tre bogstaver angiver *ejerens* rettigheder, de “midterste” tre *gruppens* rettigheder, og de sidste tre *alle andres* rettigheder. Et r betyder, at læsning er tilladt (read), et w at skrivning (og herunder sletning) er tilladt (write), og et x at filen må afvikles, hvis den er et program, eller hvis filen er et katalog, at man må gå ind i kataloget (execute). Feks. er filen treue20030614 en backup af mine emails. Emails er en privat sag, så ingen andre end jeg kan læse eller skrive denne fil.

Filen propaganda.sdw indeholder en politisk brandtale fra et af de valg, som jeg har stillet op til. Jeg selv kan både læse og skrive i den, og jeg har ikke noget imod at andre læser den (det var jo netop ideen), så alle andre har lov til at læse filen. Jeg tillader imidlertid ikke andre at skrive til den, da nogen ved et uheld kunne komme til at slette den.

### 4.3.1 Ændring af rettigheder

Filrettigheder kan ændres med programmet `chmod(1)`, der har følgende syntaks:

```
chmod [-R] {ugo}{+-}{rwx} <filnavne>
```

Den første klamme angiver hvis rettigheder der ændres: ejerens (user), gruppens (group) eller andres (other). Fortegnet angiver om rettighederne skal gives (+) eller trækkes tilbage (-), og den sidste klamme angiver hvilke rettigheder det drejer sig om.

Så hvis jeg f.eks. ønsker at lade en god bekendt ændre i min brandtale, kunne jeg midlertidigt give “andre” skriverettigheder:

```
datamat > chmod o+w propaganda.sdw
datamat >
```

Jeg vil måske gerne sikre mig, at der rent faktisk skete noget:

```
datamat > ls -l propaganda.sdw
-rw-r--rw-  1 treue  users      13824 Jan 27  2003 propaganda.sdw
```

Tilføjer man -R til `chmod`-kommandoen vil den rekursivt arbejde sig ned igennem kataloger. Feks. har jeg på min konto et katalog ved navn `admin`, der indeholder underkatalogerne `linux-2.4.25`, `Mathematica-5.0` og `ghostscript-7.05`, som er under udvikling, eller som jeg er i gang med at teste. Idet de ikke er klar til “normale” brugere er der selvfølgelig ingen andre, der kan læse kataloget, eller nogle af dets underkataloger eller filer. Det kunne imidlertid godt tænkes, at jeg havde behov for at lade en anden administrator kigge disse kataloger igennem. Vi kunne så oprette en gruppe for alle administratorerne, og ved at ændre grupperettighederne kan jeg tillade medadministratorer men ikke studerende læsning. Jeg kan bruge -R-parameteren til at ordne både `admin`-kataloget og alle dens filer og underkataloger i et hug:

```
datamat > chmod -R g+r admin
```

### 4.3.2 Standardrettigheder

Laver man en ny fil, kommer man automatisk selv til at eje den, og filen får rettighederne

```
-rw-r--r--
```

Man kan ændre standardrettighederne med kommandoen `umask`. Denne er dokumenteret i `man`-siden for `tcsh(1)`, og det særlige oktale rettighedsformat i `man`-siden for `chmod(1)`. Vi behandler `man`-sider i næste afsnit.

## 4.4 Elektronisk dokumentation

I virkeligheden er de fleste kommandoer vi har set ovenfor ikke indbygget i kommandofortolkeren. I stedet er kommandoerne selvstændige programmer, som kommandofortolkeren finder og kører for en, når man angiver deres navne. Kommandofortolkeren leder efter filer, der kan gøre det ud for programmer, i nogle på forhånd fastsatte kataloger. Nogle få kommandoer, f.eks. `pwd` håndterer kommandofortolkeren direkte. Disse kommandoer kaldes *built-ins*.

Langt de fleste programmer er dokumenteret online ved hjælp af `man(1)`. Man finder dokumentation af et program med kommandoen `man <program>`. Man bladrer i dokumentationen for et program med følgende taster:

```
<space>  Bladrer fremad.
<d>      Bladrer fremad (down).
<u>      Bladrer tilbage (up).
</>     Søger fremad.
<?>     Søger tilbage.
<n>      (Efter søgning) finder næste.
<N>      (Efter søgning) finder forrige.
<g>      Hop til toppen.
<G>      Hop til slutningen.
<q>      Afslut.
```

Efter begge søgetasterne skal man indtaste det ord man søger efter, og afslutte med `<return>`. I virkeligheden benyttes `less(1)` til at vise dokumentation; se dokumentationen af `less` for flere muligheder.

Vil jeg f.eks. vide mere om `ls`, der til trods for sin simple funktionalitet har virkeligt mange parametre, prøver jeg

```
datamat > man ls
```

der resulterer i

```
LS(1)                                FSF                                LS(1)

NAME
    ls - list directory contents

SYNOPSIS
    ls [OPTION]... [FILE]...
```

## DESCRIPTION

List information about the FILEs (the current directory by default). Sort entries alphabetically if none of `-cftuSUX` nor `--sort`.

`-a, --all`  
do not hide entries starting with `.`

`-A, --almost-all`  
do not list implied `.` and `..`

`-b, --escape`  
print octal escapes for nongraphic characters

[...]

Den elektroniske dokumentation dækker ikke kun programmer, men også C-standardbiblioteket, standardbiblioteker for en mængde andre programmeringssprog samt en masse andet. Hvis man skal vide noget, om et eller andet der har et navn, kan man som regel finde noget dokumentation med `man <navn>`. Til tider er der desværre overlap imellem f.eks. et programnavn og et funktionsnavn i et eller andet standardbibliotek, så `man`-siderne er delt ind i afsnit. De fleste programmer befinder sig i afsnit 1. Når man slår op i `man`-siderne ved hjælp af `man`-programmet kan man afgive, hvilken del af dokumentationen man ønsker at slå op i:

```
datamat > man 1 chmod
```

Som nævnt er der tradition for, at man i UNIX-dokumentation angiver et programs `man`-afsnit sammen med dets navn, når man introducerer det.

En `man`-side har tilknyttet nogle nøgleord, og på systemets klienter kan man få en liste af `man`-sider, der dækker et bestemt nøgleord med programmet `man -k`, og man kan få en en-linies beskrivelse af et program med programmet `whatis(1)`, f.eks.:

```
datamat > whatis ln
ln (1)                - make links between files
datamat > man -k permissions
chmod (1)             - change file access permissions
chmod, fchmod (2)     - change permissions of a file
ioperm (2)            - set port input/output permissions
datamat >
```

Læg her mærke til, at `chmod` optræder både som program (1) og C-funktionskald (2).





## Kapitel 5

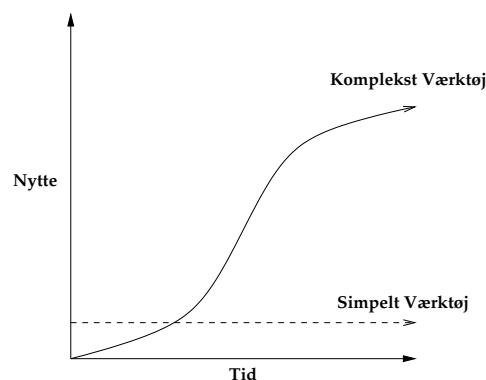
# Tekstredigeringsværktøjer

### 5.1 Hvad er et tekstredigeringsværktøj?

Et tekstredigeringsværktøj (eng. *editor*) er et program der bruges til at redigere tekstfiler med. Dette er absolut et af de mest benyttede programmer, da mange af de ting man laver på systemet kræver, man redigerer i tekstfiler – programmering, rapportskrivning og tilpasning af opsætningsfiler<sup>1</sup>. Der findes mange forskellige tekstredigeringsværktøjer, og det er naturligvis en smagssag, hvilken man bedst kan lide. Fra systemadministrationens side anbefaler vi, at man bruger tid på at lære sig et ordentlig værktøj, idet det vil spare en for utrolige mængder af besvær og frustration senere hen.

Dette kapitel er derfor opbygget således, at vi gennemgår de to klart mest benyttede tekstredigeringsværktøjer — `emacs` eller `vi`, idet disse to editorer på sigt tillader en at arbejde langt mere effektivt end de forskellige mindre avancerede alternativer.

Vi kunne have valgt kun at beskrive en enkelt avanceret teksteditor, men da ca. 50% foretrækker `emacs` og ca. 50% foretrækker `vi` beskriver vi begge. Ideen er så, at man selv kan vælge den af editorerne man synes er lettest at anvende. Efterhånden som opgaverne bliver større og mere komplicerede, vil et simpelt værktøj blive en hæmsko, da det ikke tilbyder tilpas avancerede faciliteter. Et avanceret værktøj vil derimod vokse med opgaven, og spare en for tid og frustrationer, som vist på følgende figur:



<sup>1</sup>Ofte kaldet dot-filer, da konfigurationsfiler under UNIX typisk starter med “.” (se afsnit 4.2.3).

## 5.2 VI(M)

Det oprindelige grafiske tekstredigeringsværktøj på UNIX er `vi(1)` (**vi**sual editor). Hvilket udtales ”vee eye”. Den nyeste udgave af `vi` hedder `vim(1)` (**vi** improved). `vi` findes som standard på stort set alle UNIX maskiner og det er derfor et værktøj som er rart at kunne bare et minimum af kommandoer til. Man starter `vi` ved at skrive kommandoen `vi <filnavn>` i kommandofortolkeren. Hvis filnavnet findes i forvejen åbnes den eksisterende fil, ellers skabes et nyt tomt dokument. Det er vigtigt at vide at `vi` er en *tilstandseditor* og har tre forskellige tilstande:

1. Normaltilstand
2. Indsættelsestilstand
3. Kolontilstand

Når `vi` starter op befinder den sig altid i normaltilstanden. Man kommer i indsættelsestilstand, når man enten skriver **A** (**A**ppend), **I** (**I**nsert) eller **O** (**O**pen blank line). Hvis man vil tilbage til normaltilstanden trykker man på ESC. Ofte skriver `vi -- INSERT --` i nederste venstre hjørne for at indikere at man er i indsættelsestilstand.

Kommando	Beskrivelse
I	Indsæt først på linien
i	Indsæt foran markøren
A	Indsæt i slutningen af linien
a	Indsæt efter markøren
O	Åben helt ny linie over eksisterende
o	Åben helt ny linie efter eksisterende

Tabel 5.1: `vi` indsættelseskommandoer

### 5.2.1 Navigation

`vi` er ikke beregnet til at man navigerer rundt i sin fil med piletasterne. I stedet er det meningen at man benytter fire almindelige taster (h, j, k og l) til navigation. Dette er i høj grad historisk betinget, idet man ikke altid har haft piletaster på et tastatur. Man kan dog som regel bruge piletasterne i nyere udgaver af `vi`.

	k navigerer opad
^	
k	
<h    l>	h navigerer til venstre l navigerer til højre
j	
v	
	j navigerer nedad

Alligevel har man valgt at beholde denne navigationsform, da den er utrolig fiks. Hvis man f.eks. ønsker at bevæge sig fem linier op, kan man nøjes med at skrive `5k`, i stedet for at trykke fem gange på `k` tasten eller fem gange på en op-piletast.

### 5.2.2 Gem og luk

Der er flere forskellige måder at gemme og lukke på i vi. Den nemmeste kommando er ZZ. En mere sigende måde er ved hjælp af kolontilstanden. Alle kommandoer i denne tilstand starter med et kolon. Hvis man vil gemme sin fil skrives :w (write) og hvis man vil afslutte programmet skrives :q (quit). Kommandoerne kan også kombineres til :wq (write and quit). Hvis man ved en fejltagelse er kommet til at åbne vi og gerne vil ud af programmet igen - vel at mærke uden at lave ulykker - er det en god ide at skrive kommandoen :q!, hvilket laver en tvungen lukning af filen, hvor eventuelle ændringer ikke gemmes. Dette er den magiske kommando som mange førstegangs vi-brugere desperat mangler.

### 5.2.3 Editeringskommandoer

Hvis man vil slette et enkelt tegn bruges x i normaltilstanden, hvis man i stedet ønsker at erstatte et enkelt tegn med et andet bruges r (replace character) eller cw (change word) for at ændre i et helt ord. Kommandoen cc (change current line) ændrer i hele linien. Ønsker man at slette en hel linie, skriver man dd.

Hvis man er kommet til at slette noget ved en fejl, er det altid rart at kunne fortryde sin handling. Til dette benyttes kommandoen u (undo).

### 5.2.4 Klip, Klistre og Kopiér

Man kan ofte spare redigeringstid ved at kopiere dele af allerede eksisterende tekst. Med kommandoerne y (yank) og p (put) kan man optimere sin brug af editoren. Yank kopierer den valgte tekst ind i speciel buffer, hvor den opbevares indtil man kalder yank igen, eller man vælger at slette noget tekst i hvilket tilfælde bufferen bliver overskrevet med det man lige har slettet. Kommandoen yy er gældende for en hel linie, på samme måde som dd og cc. Man bruger kommandoen p til at indsætte den tekst man lige har kopieret.

Hvis man ønsker at gentage den samme kommando flere gange, kan man med fordel benytte kommandoen . (repeat), som gentager den sidst anvendte kommando.

### 5.2.5 Søg og erstat

Kommandoerne til søgning i vi er nemme og det er iøvrigt værd at bemærke at disse også fungerer i mange andre UNIX-programmer som f.eks. less. / efterfulgt af et søgeord, vil søge fremad/nedad i filen (på samme måde som ved manualsider). ? efterfulgt af et søgeord, vil søge bagud/opad i filen. Man kan gentage den samme søgning ved blot at skrive søgekommandoen uden søgeordet eller ved at bruge gentagelseskommandoen n (next).

Markøren bevæger sig automatisk hen til det sted i filen hvor søgeordet befinder sig, og man kan herefter redigere ordet på normal vis. Ønsker man at bevæge sig til en specifik linie i sin fil, kan man blot skrive :<linienummer>. Hvis man vil til toppen af sin fil, kan man derfor blot skrive :0. Hvis man i stedet ønsker at hoppe ned til bunden af sin fil, kan man skrive G (go to).

Har man behov for at lave en global søg-og-erstat kan man benytte kommandoen :%s/old/new/g. Denne lidt kryptiske linie erstatter ordet old med ordet new globalt i hele filen.

**Bemærk** at alt søgning er versalfølsom (case sensitiv).

### 5.2.6 Kildetekstredigering

De fleste studerende vil på et eller andet tidspunkt komme til at arbejde med kildekode. Derfor er det fordelagtigt at benytte et tekstredigeringsværktøj, som understøtter funktioner til understøttelse af dette arbejde. Mange af disse funktioner er tilgængelige via kolontilstanden i `vi`.

Generelt gælder det at disse funktioner slås til med kommandoen `:set option` og slås fra igen med kommandoen `:set nooption`. Man kan til enhver tid se, hvilke valgmuligheder der er slået til med kommandoen `:set all`.

**Bemærk** at disse options typisk skal slås til/fra før du begynder at skrive din tekst, da de ofte først er gældende fra der hvor markøren er placeret i filen! Mange af disse kolonkommandoer kan med fordel sættes ind i filen `.vimrc` i dit hjemmekatalog. Denne vil da blive læst ved opstart af programmet.

#### Parring af parenteser

I mange sprog (f.eks. C++) er det utroligt vigtigt at have styr på sine parenteser. Denne mulighed slås til i `vi` med kommandoen `:set showmatch` og slås fra igen med kommandoen `:set noshowmatch`, hvilket også kan forkortes 'nosm'.

Man kan også benytte kommandoen `%`. Hvis man placerer markøren ovenpå den pågældende parentes og trykker `%` vil markøren kortvarigt hoppe til den tilsvarende åbnings- eller slutningsparentes.

#### Automatisk ombrydning af tekst

Når man ikke arbejder med kildekode er det fordelagtigt at editoren automatisk sørger for at ombryde tekst til en ny linie. Kommandoen `:set wrapmargin=10` sørger netop for dette, mens du skriver. En typisk værdi er mellem 7 og 15. Hvis man skriver kildekode kan det være en god ide at slå denne funktionalitet fra. Dette gøres ved at sætte margin til 0, altså `:set wrapmargin=0`.

Man kan også bruge kommandoen `:set textwidth=72` (forkortet 'tw'), som sørger for at ombryde teksten til en ny linie, når man når til det 72'ende tegn. Den kan tilsvarende slås fra igen med kommandoen `:set notextwidth`.

#### Automatisk indrykning

Automatisk indrykning sørger for at starte din linie på samme indrykningsniveau som den forrige. Dette sparer en for mange tastetryk, når der er tale om pæn struktureret kode. Denne funktion slås til med kommandoen `:set autoindent` og slås fra igen med kommandoen `:set noautoindent`, hvilket kan forkortes 'noai'.

#### Visning af linienumre

Hvis man skulle blive i tvivl om hvilken linie man er på i sin kode, kan man altid slå linienumering til. Dette gøres med kommandoen `:set number` og slås fra igen med kommandoen `:set nonumber`, hvilket kan forkortes 'nonu'.

### 5.2.7 Tips og tricks i vi

Hvis man ønsker at ændre et enkelt bogstav fra stort til småt eller omvendt, kan man benytte kommandoen `~`. Det skifter størrelsen på bogstavet.

Hvis man er kommet til at lave en slåfejl og derved bytte rundt på to bogstaver, kan man blot placere markøren oven på det første bogstav og trykke `xp` (transpose).

Ønsker man at få slået to linier sammen kan kommandoen `J` bruges.

Man kan lave en shell indefra `vi`, ved at skrive `:sh`. Når man vil tilbage til `vi` kan man blot skrive `exit`. Du kan også udføre alm. shell kommandoer indefra `vi`. Dette gøres ved at skrive `!:<kommando>`, f.eks. `!:ls -ll` for at få en katalogliste.

### 5.2.8 Yderligere hjælp

Hvis man har brug for yderligere hjælp til at komme igang, findes der et rigtig godt indlæringsprogram ved navn `vimtutor`. Dette startes ved at kalde `vimtutor` fra kommandolinien og derefter følge lektionerne i filen. Desuden er der bagerst i denne bog inkluderet en tastaturreference til `vi` (se D).

## 5.3 Emacs

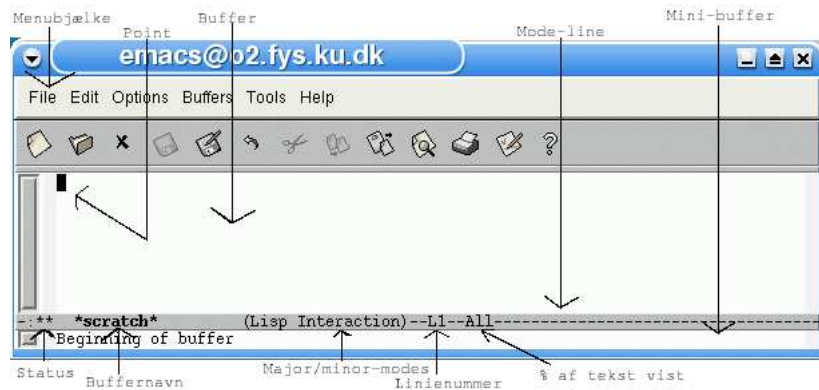
Før vi fortsætter vil vi endnu en gang opfordre til at hænge ved! Læs dette afsnit igennem og prøv tingene af imens. Hvis du stadig finder det vanskeligt, er det måske værd at udforske nogle af referencerne sidst i kapitlet. Et rigtig godt råd er at begynde at bruge emacs i det daglige. Selvom det måske i starten føles lidt akavet og tager lidt længere tid, vil det efterhånden vende, og man vil føle sig rigtig godt tilpas.

Emacs er et ekstremt kraftfuldt værktøj med mange forskellige faciliteter. Basalt set er det et tekstredigeringsværktøj, men idet man selv kan programmere udvidelser, kan emacs meget meget mere. Programmet har mange år på bagen og der er derfor lavet et væld af udvidelser til det. Man kan faktisk gøre stort set alt inde fra emacs – tekstredigering, kald af andre programmer, filhåndtering, debugging, email, news og spil. Dette betyder omvendt også, at emacs *er* ret kompliceret, men det skal ikke afskrække en fra at komme igang – man kan sagtens nøjes med at udnytte de mere basale faciliteter; efterhånden vil man opdage de mere avancerede.

### 5.3.1 Koncepter i Emacs

Før vi kan komme rigtig igang med at beskrive de forskellige faciliteter i emacs, er der nogle koncepter og begreber der skal forklares. Det er vigtigt at man har styr på disse, da det ellers kan være forvirrende at læse resten af dette kapitel.

Man starter emacs med kommandoen `emacs` og følgende vindue viser sig:



## Menubjælken

Her kan man vælge forskellige emacs-funktioner med musen, hvis ikke man vil bruge tastaturet. Menuen `(Buffers)` kan bruges til hurtigt at skifte mellem åbne buffere. Bemærk at menuen ændrer sig, alt efter hvilken *mode* emacs er i (se senere).

## Point

Det sted cursoren er placeret, kaldes i emacs for *point*.

## Bufferen

En *buffer* er det sted i emacs, hvor man redigerer i teksten. Denne kan indeholde forskellige ting: Det kan være en fil man er ved at redigere, eller det kan være en *besked-buffer* som emacs bruger til at give brugeren meddelser, men som altså ikke er tilknyttet en fil på disken. Man kan betragte buffere som en slags vinduer inde i emacs. (Se afsnit 5.3.4 for mere om buffere).

## Mode-line

Den sorte linje i bunden af emacs-vinduet kaldes på engelsk for *mode-line* og bruges til at give brugeren forskellige praktiske informationer:

- *Status*: Hvis filen er ændret i forhold til kopien på disk, vises der her to stjerner: \*\*. Hvis en fil er skrivebeskyttet vises to procenttegn: %%.
- *Buffernavn*: Dette er navnet på den buffer man er i. Dette kan være et filnavn, eller et navn på en beskedbuffer (disse har typisk navnet omgivet af \*). Den første buffer man står i når man starter emacs hedder *\*scratch\**, men ændrer navn, så snart man gemmer bufferen på disk under et nyt navn.
- *Major/minor-modes*: Emacs har forskellige *modes* alt efter hvilken slags fil man redigerer og hvilke indstillinger der er aktive: Dette er smart, da emacs så kan tilpasse og tilbyde specielle faciliteter til forskellige typer af filer.  
*Major-modes* afgør hvilken filtype man redigerer (f.eks. HTML, C++, C eller bare ren tekst). Standardopsætningen sørger for, at emacs selv skifter *major-mode*, når man henter en fil ind (dette afgør den på *endelsen* af filen – altså om filen ender på `.html`, `.cc`, `.tex` osv).

*Minor-modes* er forskellige tilpasninger, der kan påvirke enten den lokale buffer eller hele emacs. Et par eksempler er *Fill* der automatisk knækker en linje, når cursoren når ud til kanten, eller *Ovwr* (overwrite) der aktiveres når man trykker på `<insert>`-tasten og overskriver det eksisterende indhold i bufferen i stedet for at indsætte tekst.

- *Linjenummer*: Viser hvilket linjenummer i bufferen, som cursoren er placeret på.
- *% af tekst vist*: Viser hvor stor en procentdel af hele bufferen, som man kan se i vinduet.

### Mini-bufferen

Den nederste linje i emacs-vinduet kaldes for *mini-bufferen* og bruges til i forbindelse med at hente og gemme filer, til at give beskeder til brugeren undervejs og meget mere. (Alt hvad der skrives i mini-bufferen gemmes i besked-bufferen *\*Messages\**).

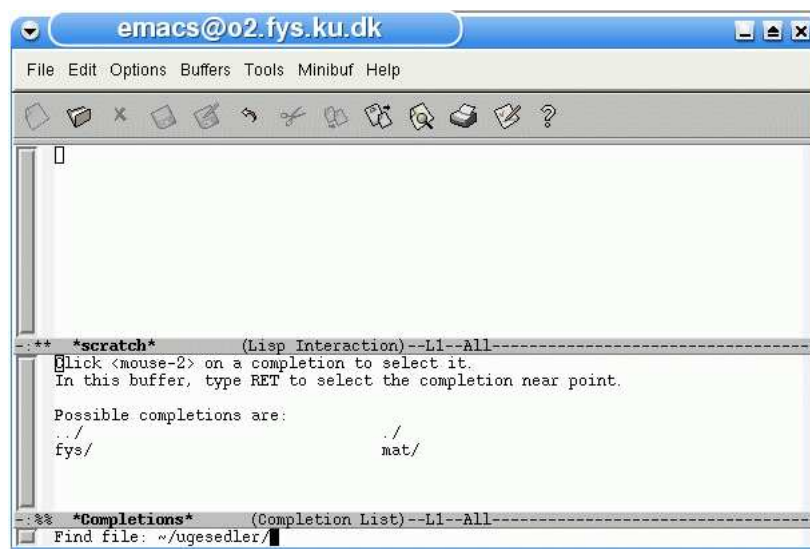
Når der f.eks. i starten står `Loading font-lock..done` er det fordi opstartsfilerne har bedt emacs om at slå `font-lock minor-mode` til (mere om det senere) og emacs svarer tilbage, at det nu er ok.

Emacs kan stille ja/nej-spørgsmål i mini-bufferen på to måder: Hvis det er et spørgsmål der kan have store konsekvenser (f.eks. at afslutte uden at gemme ændringer), skal man skrive `yes` eller `no` efterfulgt af `<enter>`, men hvis det er et mindre fatalt spørgsmål, taster man `<y>` eller `<n>`.

Når man skal vælge forskellige ting via minibufferen, kan den hjælpe en til at finde det rigtige navn. Når man f.eks. skal åbne en fil, sker dette på følgende måde: Man trykker `<ctrl-x>` `<ctrl-f>` for at åbne en fil, og minibufferen skifter til følgende:



Her kan man så taste et filnavn ind. Hvis man ikke kan huske det udenad, kan man starte på det og dernæst taste `<tab>` – herefter vil emacs åbne en midlertidig buffer, der viser hvilke mulige filer eller kataloger der er i den sti, man indtil videre har indtastet. F.eks:



Hvis der er flere filer, end der er plads til i det midlertidige buffer-vindue (det kan man se på scrollbaren yderst til venstre), kan man igen bruge `<tab>` til at vise resten. Desuden kan man hele tiden bruge piletaster, `<backspace>` og andre redigeringstaster i minibufferen undervejs. Hvis man ikke kan finde den fil man søger, eller ikke vil åbne en fil alligevel, kan man afslutte interaktionen med minibufferen ved at taste `<ctrl-g>` - denne kommando kan specielt i starten være nyttig, idet man *altid* kan komme ud af minibufferen med den.

### 5.3.2 Taster

Emacs har sin egen notation for at beskrive tastetryk, der afviger fra konventionen brugt i denne bog. Emacs' notation er dog ganske udbredt, så udover at man skal kunne den i emacs, kan man også få brug for den mange andre steder. Den gør sig bare ikke så godt på tryk. I den resterende del af dette kapitel vil vi derfor bruge emacs' notation.

En lille spidsfindighed er `<meta>`-tasten. På nogle producenters tastaturer (bla. Sun Microsystems) er der placeret en tast mellem `<alt>` og `<space>` – denne kaldes `<meta>`. På tastaturer der ikke har en meta-tast, kan `<alt>` som regel bruges i stedet. Dette er vigtigt at vide, da emacs som regel referer til `<meta>`-tasten. På grund af denne forvirring har man vedtaget et tastetryk for `<meta>`, som *altid* virker, nemlig først at trykke på `<esc>` efterfulgt af den tast, man gerne vil have `<meta>` versionen af. Emacs-notationen for tastetryk er som følger:

Tastekombination:	Noteres som:
<code>&lt;a&gt;</code>	a
<code>&lt;shift-a&gt;</code>	S-a
<code>iesc̣ iạ</code>	M-a
<code>&lt;ctrl-a&gt;</code>	C-a
<code>&lt;ctrl-a b&gt;</code>	C-a b
<code>iesc̣ &lt;ctrl-a&gt;</code>	C-M-a

Når man bruger kombinerede tastetryk, skriver emacs i minibufferen hvilke taster man har trykket. Hvis man f.eks. taster `<ctrl-x>` skriver emacs C-x- i minibufferen for at illustrere, at der ikke er bundet en funktion til `<ctrl-x>`, så der forventes endnu en tast. Følgende tabel viser nogle af de mest almindelige redigeringstastetryk i emacs-notation:

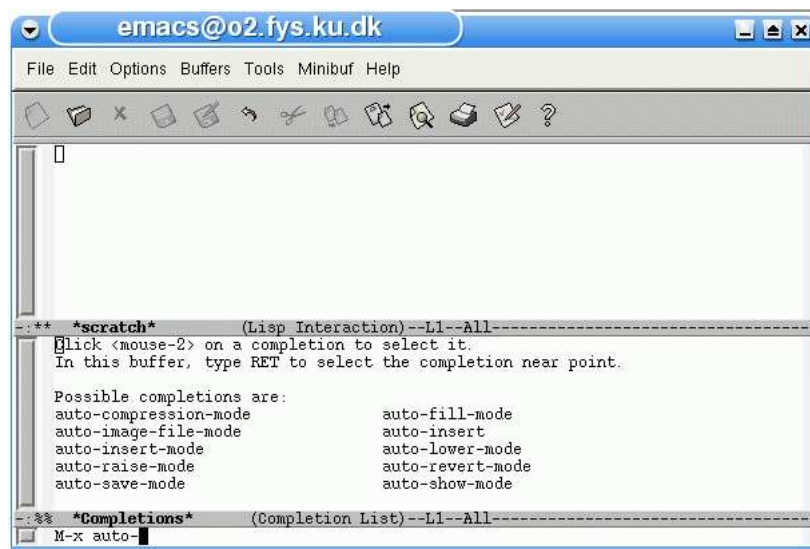


Tastekombinationen:	Har følgende funktionalitet:
C-g	Afbryder den nuværende handling – f.eks. minibufferen, en tastekombination eller andet. Generelt er dette tastetryk en rigtig god ting at kende.
C-a	Går til begyndelsen af linjen (samme som tasten <home>).
C-e	Gar til enden af linjen (samme som tasten <end>).
C-d	Sletter tegnet til højre for cursoren (sammen som tasten <del>).
C-_	Fortryder (undo). Denne kan gentages efter behov.
C-home	Går til toppen af bufferen.
C-end	Går til bunden af bufferen.
M-g	Hop til en bestemt linje.
M-f	Flytter cursoren ét helt ord til venstre.
M-b	Flytter cursoren ét helt ord til højre.

I de følgende afsnit vil der desuden løbende blive introduceret flere tastekombinationer i forbindelse med gennemgangen af de forskellige faciliteter i emacs. Se desuden bilag C for en hurtig oversigt over nogen af de mest almindelige tastekombinationer.

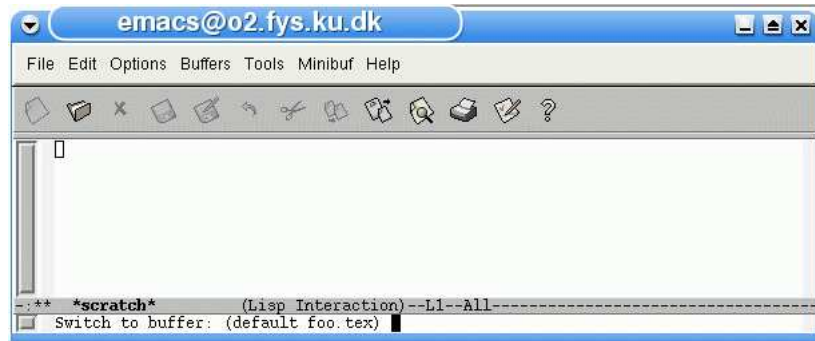
### 5.3.3 Interne funktioner

Emacs har et væld af interne funktioner. Nogle af disse aktiveres ved tastetryk (f.eks. udfører C-e funktionen `end-of-line`), men en del er ikke bundet til nogen tast. Disse kan aktiveres ved at aktivere minibufferen til at modtage kommandoer. Dette gøres med M-x. Herefter kan man indtaste kommandoer og tab viser (ligesom i filhåndtering) hvilke muligheder der er. (Se desuden afsnit 5.3.10). Følgende eksempel viser kommandolinjen i minibufferen hvor der er tastet `auto` og trykket `tab`:



### 5.3.4 Håndtering af buffere

Idet det er muligt (og praktisk) at have mange buffere åbne samtidig, er der naturligvis mulighed for at skifte rundt imellem dem. Dette kan enten gøres fra menubjælken ((Buffers)) eller vha. taster. Tastekombinationen C-x b aktiverer et bufferskift og mini-bufferen spørger hvilken buffer man vil skifte til (her kan man få en liste ved at bruge tab). Standardvalget er den sidst besøgte buffer, men eller indtaster man navnet på den buffer man vil skifte til (her virker tab også).



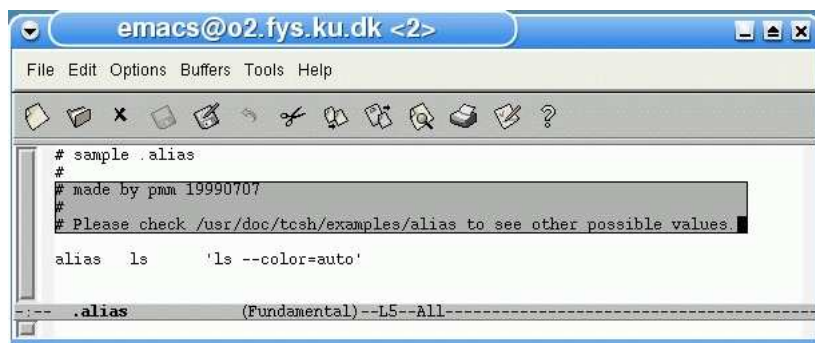
Ovenstående eksempel viser et skift fra bufferen *\*scratch\** til den tidligere besøgte buffer, *foo.tex*. Man kan desuden få emacs til at vise flere buffere på én gang, ved at bruge tastekombinationerne `C-x 2` (dele vinduet i to buffere på tværs), `C-x 3` (dele vinduet i to buffere på langs). Man skifter mellem flere buffere i vinduet med `C-x o` og man maksimerer den aktive buffer med `C-x 1`.

Man lukker en buffer med `C-x k`.

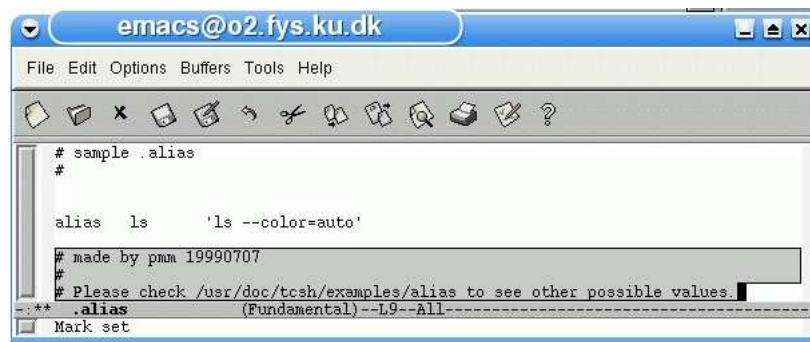
### 5.3.5 Klip, klister og kopiér

Når man redigerer i en buffer i emacs, er der forskellige metoder til at klippe, klistre og kopiere tekst i vinduet. Man kan enten benytte standard X-metoden (se afsnit 2.3.2) eller man kan bruge emacs' indbyggede faciliteter. Dette er lidt anderledes end man ser det i Microsoft Windows-programmer, men er ikke mere kompliceret:

For at markere et stykke tekst til kopiering eller klipning, sætter man først et *mærke* (eng. *mark*) ved at placere cursoren (point) der, hvor man vil markere *fra* og taster `C-space`. Dernæst flyttes cursoren til der hvor man vil markere *til*. Området mellem mærket og cursorens nuværende position (point) kaldes en *region*. Følgende billede illustrerer dette (regionen er her for illustrationens skyld markeret med gråt):



Når man har markeret en *region* kan man enten kopiere den med tastekombinationen `M-w` eller klippe den med `C-w`. Hvis man herefter vil indsætte den kopierede/klippede tekst, placeres cursoren et passende sted og teksten indsættes `C-y`. Følgende billeder viser situationen hvor teksten blev klippet og sat ind et par linjer nede:



Emacs gemmer desuden tidligere kopierede/klippede tekster, så efter kan har tastet C-y kan man taste M-y for at cykle rundt imellem tidligere tekster<sup>2</sup>. En anden smart feature ved *mærket* er, at man kan bytte cursorens nuværende placering med det – det er smart, hvis man er ét sted og lige skal hen og kigge et andet sted i filen: Først sætter man mærket med C-space, dernæst navigerer man rundt i filen indtil man har set hvad man ville. Når man så vil tilbage til mærket, taster man blot C-x C-x.

### 5.3.6 Filhåndtering

For at hente og gemme filer bruges mini-bufferen som tidligere beskrevet. Når man henter en fil ind får den sin egen buffer, så det er muligt (og meget praktisk) at have mange filer åbne samtidig. Følgende tabel giver de mest almindelige filbehandlings-operationer:

Tast:	Funktion:
C-x C-f	Åbner en fil. Hvis filen ikke eksisterer i forvejen, oprettes den.
C-x C-s	Gemmer den aktive buffer i en fil. Hvis bufferen endnu ikke er gemt under et filnavn, spørges efter dette.
C-x C-w	Gemmer den aktive buffer under et andet navn (save-as).
C-x i	Henter en fil ind i den aktive buffer.

Emacs gemmer desuden som standard en sikkerhedskopi af bufferen hvert 5 minut.

### 5.3.7 Søg og erstat

Emacs har adskillige faciliteter til at søge i en buffer. De benytter dog alle stort set den samme fremgangsmåde, som vi illustrerer i dette afsnit gennem de mest anvendte søgefaciliteter.

#### Søgning

Denne funktion søger i bufferen efter cursorens position. Forlæns søgning aktiveres ved C-s og baglæns søgning aktiveres ved C-r. Herefter skriver emacs I-search: i mini-bufferen og man kan her taste et søgeord (eller en del af det). Efterhånden som man taster i minibufferen søger emacs til det første sted, der matcher søgeordet. Hvis man vil videre til næste forekomst, tastes C-s igen. Bemærk at man kan redigere i sit søgeord undervejs – fokus under søgning er altså i minibufferen og *ikke* i selve bufferen. Hvis man finder et sted, man gerne vil rette, bruges piletasterne til at stoppe søgningen og stille cursoren på det nye sted, men mærket sættes på det sted, hvor søgningen startede. Det betyder, at når man er færdig med at redigere, kan man hoppe tilbage ved at taste C-x C-x, som før beskrevet. Emacs husker det sidst søgte ord, så C-s C-s gentager forrige søgning.

<sup>2</sup>Det sted hvor emacs gemmer alle klippede ting, kaldes en *kill-ring*.

### Søg og erstat

Denne funktion er praktisk, hvis man f.eks. konsekvent har stavet et ord forkert – så kan man erstatte forekomster af én tekst med en anden. Søg og erstat aktiveres med `M-%` og minibufferen spørger først hvilken tekst, der skal søges på, og dernæst hvad der skal erstattes med. Herefter finder emacs den første forekomst af søgeordet og spørger brugeren hvad der skal gøres. Her er forskellige muligheder: Man kan interaktivt taste `y` eller `n` for at bestemme om den pågældende match skal ændres, eller man kan taste `!` for at alle forekomster skal ændres. (Bemærk iøvrigt at `C-g` også her virker som afbryd.)

### 5.3.8 Fortryd

Emacs husker (op til en vis grænse) hvilke ændringer, der er foretaget i hver buffer. Det betyder, at man kan fortryde (undo) stort set alle ændringer (eller gentage (redo) den, hvis ikke man ville fortryde alligevel).

Fortryd kaldes med tastekombinationen `C-_` og kan gentages efter behov. Så snart man foretager sig andet end at fortryde, vil funktionen “vende” og næste gang man fortryder, er det fortrydelsen fra før, man gendanner. Forvirret? Se følgende eksempel; først taster:

```
a
b
c
d
```

Herefter taster fortryd (`C-_`) fire gange:

```
a
b
```

Dette fortryder (i rækkefølge) tastesequensen `enter`, `d`, `enter`, `c`. Herefter taster for eksemplets skyld `e` efterfulgt af `enter`.

```
a
b
e
```

Dette afbryder fortrydelsessekvensen, så når man nu trykker fortryd 5 gange sker følgende:

```
a
b
c
d
```

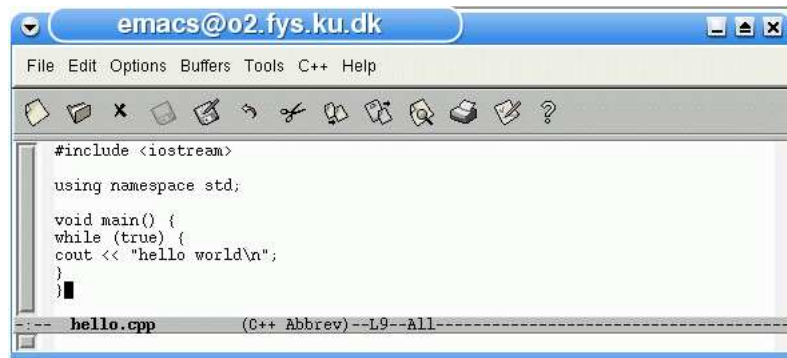
Fortryd fortrød her først indsættelsen af `e`, og herefter den tidligere fortrydelse.

### 5.3.9 Kildetekstredigering

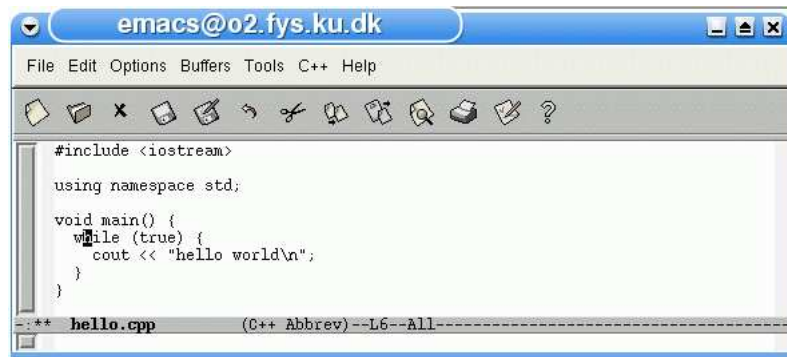
Idet emacs har major-modes til de fleste programmeringssprog, kan programmet hjælpe en med mange ellers trivielle redigeringsopgaver, hvilket typisk gør, at man laver færre fejl. Hvis man åbner en fil i emacs med en standartendelse for et givet programmeringssprog (f.eks. `.cpp` for C++ programmer), vil emacs automatisk starte i den tilsvarende major-mode. Det kan her iøvrigt bemærkes, at en major-mode ændrer menubjælken, så nu er der et menupunkt med funktioner relevante for C++.

### Automatisk indrykning

Til mange af Emacs' major-modes, er der defineret hvordan indrykning (eng. *indentation*) skal være for det pågældende programmeringssprog. Dette er en ekstrem stor hjælp når man programmerer, da man så kan lade emacs gøre det automatisk. Følgende eksempel illustrerer hvad indrykning er, og hvordan emacs gør det for en. På det første billede ses at koden ikke er indrykket – alle linjer starter i kolonne 1:



Dette gør store programmer fuldstændigt ulæselige, så derfor laver man indrykning. Når man skriver programmer i emacs, kan man trykke på tab et vilkårligt sted på en linje og emacs vil lave passende indrykning:



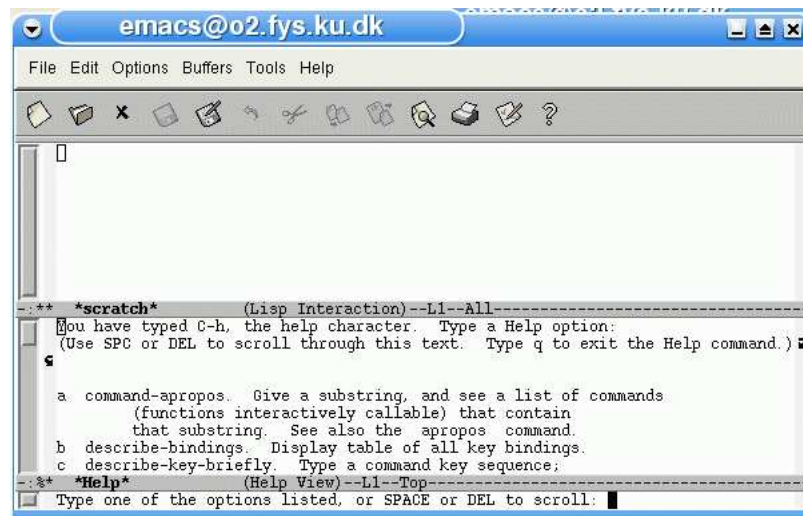
Dette kan også gøres for en *region*, ved at markere regionen og taste C-M-\<sup>3</sup>.

Automatisk indrykning er også en stor hjælp til syntaktisk korrekthed, da man kan se hvis indrykningen er helt i skoven, hvilket typisk betyder at der glemte en parentes eller lignende.

#### 5.3.10 Hjælp

Emacs har et omfattende indbygget hjælpesystem, der kan hjælpe med informationer om taste-kombinationer, interne funktioner osv. Hvis man taster C-h C-h får man følgende buffer med hjælpefunktioner, der så kan aktiveres ved at taste den relevante tast:

<sup>3</sup>Et lille tip: C-x h markerer hele bufferen som en region.



De vigtigste af disse er:

- a - Søg efter en funktion ud fra navnet
- b - Viser en liste over gældende tastekombinationer
- c - Beskriver hvilken funktion et bestemt tastetryk udfører
- i - Info-systemet. Dette indeholder den fulde dokumentation til Emacs. Tast herefter h for at få introduktionen til info-systemet.

## Kapitel 6

# Elektronisk post

Dette kapitel omhandler elektronisk post. Vi gennemgår programmet `mutt` og hvorledes dette bruges på systemet, hvorefter vi vil komme ind på diverse detaljer mht. email.

Fra enhver konto på systemet kan man sende elektronisk post (email). Ens egen adresse er navnet på ens konto `konto@fys.ku.dk` eller `konto@nano.ku.dk`<sup>1</sup>

### 6.1 Mutt

Det anbefalede program til at læse elektronisk post med er `mutt` (se afsnit 6.2 for alternativer). Det er et tekstbaseret program, men det skal man ikke lade sig snyde af. `Mutt` kan virkelig mange ting og er faktisk ikke så svært at bruge. Denne introduktion giver et kort overblik og viser hvordan man kommer igang, men ellers er det bare om at begynde at bruge det.

#### 6.1.1 Hvordan starter man mutt?

`Mutt` startes med kommandoen `mutt`<sup>2</sup>.

Når `mutt` åbnes, viser den automatisk brugerens inbox. I øverste linie af skærmen ses forskellige kommandoer, såsom `mail`, `reply` osv. Netop disse to er beskrevet nedenfor (pga. deres vigtighed), men ellers kan man anvende hjælp funktionaliteten (ved at trykke `?`) til at få en total liste af alle kommandoer i `mutt`, med enliniers beskrivelser.

At læse en email gøres simpelthen ved at vælge filen fra startvinduet - man kan herefter køre rundt i mailen med `<pg-up>` og `<pg-down>`, og skifte fra mail til mail med pil op og ned. Man kan markere mails til sletning ved at trykke `<d>` mens man læser dem, eller mens man har markeret dem i mailboxen. Man lukker for `mutt` ved at trykke `<q>`, hvorefter man vil blive spurgt, om man virkelig ønsker at slette de mails, som man evt. har markeret til sletning.

---

<sup>1</sup>faktisk har alle konti begge emailadresser, men er man tilknyttet nanoteknologi, anbefaler vi selvfølgelig at man giver nano-adressen ud, og omvendt, er man tilknyttet NBIfAFG giver man fys-adressen ud.

<sup>2</sup>Første gang man starter `mutt` stiller den et spørgsmål om man vil oprette biblioteket `Mail` i ens hjemmekatalog - det er en god ide at gøre det, så tast `<enter>`

### 6.1.2 Hvordan sender man elektronisk post fra mutt?

For at sende et nyt brev, taster man `<m>`, indtaster modtager adressen og et emne for emailen, hvorefter ens standart editor startes. Her kan man saa skrive mailen, og når man er færdig, gemme den og lukke for editoren - mutt åbner så for et vindue, hvor man kan indtaste forskellige andre ting (såsom flere modtagere), vedhæfte en fil eller andet. Når mailen er tilfredsstillende, taster man `<y>` for at sende den, og mutt returnerer til mailboxen.

### 6.1.3 Hvordan svarer jeg på elektronisk post i mutt?

Når en besked er markeret med bjælken i inbox-oversigten, eller hvis man er ved at læse den, kan man svare på den ved at trykke `<r>` (reply). Herefter fungerer mutt næsten, som hvis man var ved at sende en ny mail, bortset fra, at mutt foreslår afsenderen af den mail, man svarer på, som modtager af svaret, ligesom mailen som standart får en fornuftig emne-linie sat. Mutt spørger også, om man vil citere den mail man svarer på, hvilket kan være rart hvis mailen f.eks. indeholder en række spørgsmål, som man ønsker at svare på i rækkefølge, på denne måde:

```
> <spørgsmål 1>
<svaer på spørgsmål 1>
> <spørgsmål 2>
<svaer på spørgsmål 2>
> <spørgsmål 3>
<svaer på spørgsmål 3>
...
```

### 6.1.4 Diverse detaljer

Her kommer til sidst et par ekstra detaljer, der kan være gode at vide. For mere information anbefales det at gå "på opdagelse" i mutt og benytte sig af den indbyggede hjælp.

**Foldere** Udover din "inbox" kan du kan også selv oprette foldere, hvis du vil sortere din post. En folder oprettes vha. kommandofortolkeren (se 4.2.1), som et underkatalog til kataloget Mail i brugerens hjemmekatalog. Man skal, for at mutt forstår at kataloget er en mailfolder, gå ind i kataloget og køre kommandoen `touch .mh_sequences`.

**Placering af post** Ny post (inboxen) ligger i kataloget `/var/mail/<kontonavn>` og andre foldere ligger normalt under kataloget `~/Mail`. Her skal man være opmærksom på, at inboxen ikke må overstige 250 MB, samt at foldere i `~/Mail` "tæller med" under beregningen af ens quota - det er med andre ord en god ide at rydde op i ens mail en gang imellem.

**Signaturer** En signatur er et par linjer i bunden af dit brev med oplysninger om dig selv, din adresse eller måske et sjovt citat. Grundreglen er her, at en signatur starter med linjen: *linjeskift minus minus mellemrum linjeskift* og derefter højst fylder 4 linjer. Eksempelvis:

```
--
test@fys.ku.dk
"Light thinks it travels faster than anything but it is wrong.
No matter how fast light travels it finds the darkness has always
got there first, and is waiting for it." -- Terry Pratchett
```



## 6.2 Andre programmer

Der findes også andre mailprogrammer end mutt. Et alternativ er pine, der er meget udbredt (også blandt ældre studerende), men fra edb-afdelingen råder vi imod det: det er langsomt, og brugerinterfacet kan, efter vores mening, være meget obskurt ind i mellem.

Netscape/mozilla har også et emailprogram, men da det er notorisk ustabilt vil vi ikke anbefale at bruge det og det er under alle omstændigheder på eget ansvar (ensbetydende med *“ingen support fra edb-afdelingen”*). Edb-afdelingen understøtter officielt mutt, så det anbefales at man bruger en af den.

## 6.3 Spam

Begrebet “spam” handler om uopfordret email. Den moderne teknologi giver desværre mulighed for meget nemt at nå ud til mange mennesker, hvilket mange benytter sig af. Dette betyder, at man til tider får emails med vittigheder, reklamer eller jobtilbud sendt uopfordret. Systemet har en mulighed for automatisk at frasortere en stor del af den spam man for tilsendt - se 8.1.2.

Derudover er det naturligvis selvskrevet at man selv holder sig langt væk fra at sende spam-post, idet det er *meget* dårlig tone og opfattes som misbrug af systemet.



## Kapitel 7

# Tilgang til ØL's system udefra

Det kan en gang imellem være nyttigt at kunne tilgå ens konto, uden at man sidder på HCØ - f.eks. kan det være rart at kunne læse sin email hjemmefra, eller man kan ønske at hente ens seneste rapport ned fra systemet til ens hjemmecomputer så man ikke behøver at tage ind på instituttet bare for at læse korrektur på den. Måden man opnår disse ting på, er forskellig alt efter hvorvidt man bruger en eller anden version af Microsoft Windows, eller om man har sin egen Un\*x installation (f.eks. linux) installeret. Det må dog tilrådes at man skimmer "det andet" afsnit igennem også - selv om man ikke bruger det fra sin hjemmecomputer kan man stadig risikere at skulle bruge det fra computere på andre institutioner (RISØ, DTU, CERN osv. osv.).

### 7.1 Logge ind på systemet udefra

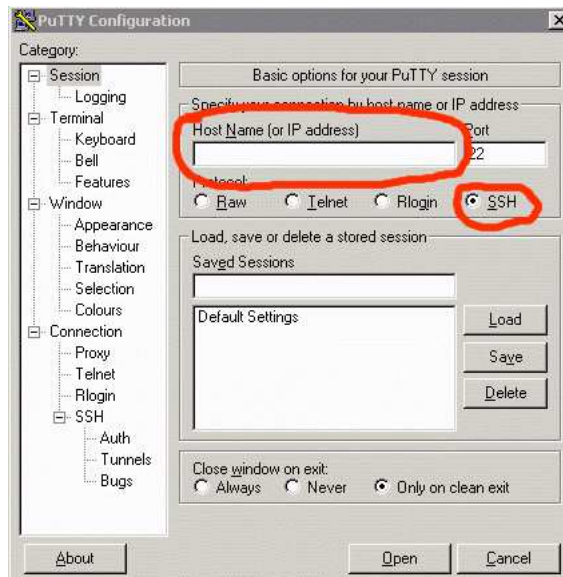
Den mest normale situation vil formentligt være, at man ønsker at logge ind på ØL's system udefra, dvs. at få noget tilsvarende til en xterm på ØL's system, blot fra en computer et andet sted i verden. For at opnå dette bruger man et program som hedder `ssh(1)`<sup>1</sup> Ssh er implementeret til både MS Windows og Un\*x, og jeg vil i det følgende vise hvordan man fra sin egen computer kan tilgå ØL's system vha. disse implementioner

#### 7.1.1 Microsoft Windows

Vi vil her bruge den implementation som hedder putty . Fra EDB-afdelingens webside (<http://www.fys.ku.dk/edb>) linker vi til et sted, hvorfra man kan downloade den nyeste version af putty - den fil man downloader kører man umiddelbart ved at dobbeltklikke på den, og man får så et vindue, som det vist på figur 7.1 op på skærmen. I feltet "Host name or IP address" skriver man "fys.ku.dk" og sørger for, at "SSH" er valgt under "protocol" lige nedenunder. Derefter trykker man ok, hvorefter putty muligvis vil spørge om man ønsker at stole på serveren for fremtiden - her vil man normalt kunne svare "OK". Herefter vil man blive bedt om at indtaste login og derefter password (på samme måde, som hvis man sad på HCØ), og hvis disse indtastes korrekt, vil man derefter kunne bruge putty på samme måde som en xterm (se afsnit 4) - blot vil man ikke kunne starte grafiske programmer, dvs. programmer som resulterer i andet en tekst udskrift på skærmen<sup>2</sup>.

<sup>1</sup>De læsere som på forhånd har kendskab til at logge ind på andre computere vil måske kende programmet `telnet(1)` , der kan bruges som alternativ til `ssh(1)` . Det er i skrivende stund ikke besluttet hvornår adgangen til at telnet'te til ØL's system bliver afskaffet, men det er besluttet, at den skal afskaffes, hvorfor man bør lære `ssh(1)` uanset om man har erfaring med `telnet(1)` .

<sup>2</sup>f.eks. vil man ikke kunne starte `mozilla(1)` , da den forsøger at vise grafik, men man vil godt kunne starte `mutt(1)` da den har et rent tekst interface.



Figur 7.1: Putty i aktion

### 7.1.2 Un\*x

Næsten alle Un\*x distributioner har som standart ssh(1) installeret. For at logge ind på fysik bruges komandoen:

```
hjemmedatamat > ssh -l <login> fys.ku.dk
```

hvor <login> udskiftes med ens brugernavn. Man vil herefter blive afkrævet et password, og når dette er tastet ind, har man fået en prompt fulstændigt tilsvarende en xterm (se afsnit 4). Ligesom det var tilfældet for putty i windows ovenfor, kan man (normalt) ikke starte grafiske programmer, men dette problem kan løses for Un\*x - det kræver to ting:

1. ssh skal være kaldt fra en xterm på en xwindows-server på den maskine man forsøger at logge ind fra.
2. Man skal tilføje -X (husk: det skal være et *stort* X) til komandolinien ovenfor.

Er disse to betingelser opfyldt, skulle man kunne få grafiske programmer vist, selvom de køres på ØLs servere. Vær dog opmærksom på, at det bruger temmeligt meget båndbredde at udnytte denne facilitet - med mindre du sidder på et relativt godt netværk (f.eks. en anden forskningsinstitution som RISØ osv.), kan det vise sig reelt at være ubrugbart.

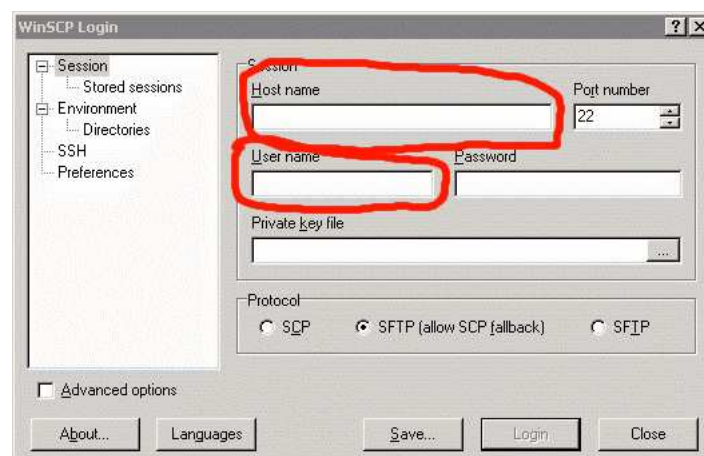
## 7.2 Kopiere filer til og fra brugerkonti på ØL

En anden ting, som det er rart at kunne, er at kopiere filer til/fra ens konto på ØL. Til dette kan man bruge programmet sftp(1), der er en slags søsterprogram til ssh(1)<sup>3</sup>. Ligesom ssh(1) er sftp(1) implementeret for de fleste kendte operativsystemer, og det vil her blive forklaret hvorledes det bruges i hhv. Microsoft Windows og Un\*x:

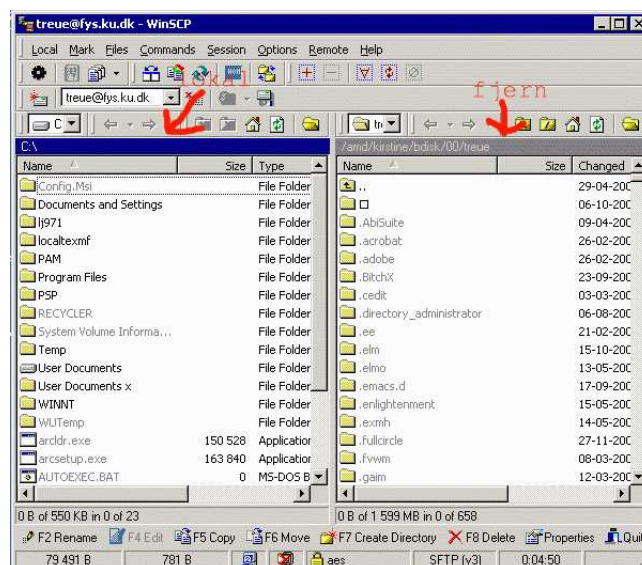
<sup>3</sup>nogle brugere kender muligvis ftp(1). Hvad dette angår gælder samme situation som for telnet(1): der vil blive lukket for det indenfor en overskuelig fremtid.

### 7.2.1 Microsoft Windows

Vi anbefaler at man bruger WinSCP under Microsoft Windows, som bl.a. har sftp(1) inbygget - der er linket til den fra edbafdelingens hjemmeside (<http://www.fys.ku.dk/edb>). Man skal her køre et installations program for at kunne bruge programmet, men standart instillingerne vil være udemærkede for de fleste. Når man starter WinSCP får man et billede op som det på figur 7.2, hvor man indtaster "fys.ku.dk" i feltet "host name" og ens brugernavn i feltet "login", og trykker OK. Man vil muligvis blive spurgt om man stoler serveren - her kan man blot svare "yes", hvorefter man bliver afkrævet et password. Når det er indtastet, får man en todelt skærm op (se fig. 7.3), hvor den venstre del af skærmen repræsenterer ens egen maskine, og højre del af skærmen repræsenterer ens brugerkonto på ØL. Man kan nu på ganske almindelig vis flytte filer mellem de to - stort set som hvis det var mellem to foldere på ens egen maskine.



Figur 7.2: Winscp login vindue



Figur 7.3: Den lokale og fjerne maskines filer

### 7.2.2 Un\*x

Som for `ssh(1)` gælder det, at de fleste installationer har `sftp(1)` installeret som standart. Man starter det med

```
hjemmedatamat > sftp <login>@fys.ku.dk
```

hvor `<login>` udskiftes med ens brugernavn. Man vil herefter blive afkrævet et password, og derefter vil man få en prompt meget lig den som `ftp(1)` bruger. For en dokumentation af, hvad `sftp(1)` kan bruge af kommandoer, se

```
hjemmedatamat > man sftp
```

## Kapitel 8

# WWW på ØL

En af de ting, man som bruger får brug for, er at tilgå websider på Internettet. Til dette formål har ØLs system installeret browseren mozilla, som er en udløber af netscape browseren, der var den første browser som blev anvendt i større omfang. Den startes med kommandoen mozilla(1) i en xterm(1) (se afsnit 4) eller vha. det ikon, som ser ud som en rød stjerne.

Det hænder, at mozilla istedet for at starte op, beder brugeren om at vælge en profil, da den normale allerede er i brug. Dette skyldes at mozilla allerede er startet et eller andet sted fra en brugerkonto, eller at mozilla ikke blev lukket rigtigt ned sidst man brugte den. I det sidste tilfælde, hvor man altså ikke bare er kommet til at starte mozilla to gange, skal man, inden man kan begynde at bruge mozilla igen, køre følgende kommando:

```
datamat > rm ~/.mozilla/<brugernavn>/*/lock
```

hvor <brugernavn> skal være det brugernavn man er logget på som. Herefter kan man starte mozilla .

### 8.1 ØL's websystem

ØL har et relativt veludviklet websystem, som det ville føre for vidt at forklare totalt i denne bog. Der er dog to ting som har særlig interesse for brugere på ØLs system, nemlig webmail og kontoadministration.

#### 8.1.1 Webmail

Webmail dækker over faciliteter til at læse ens email via en webside, i modsætning til at læse den med f.eks. mutt(1) . Fordelen ved det er, at man ikke behøver at være logget ind på ØLs computere for at anvende det, hvorfor det kan bruges fra næsten alle computere i verden, bare de er koblet til Internettet. På ØLs system er der to forskellige programmer til webmail, nemlig twig og squirrelmail. Hvilken en, man bruger, er en smagssag, men de kan tilgås via webadresserne [www.fys.ku.dk/twig](http://www.fys.ku.dk/twig) og [www.fys.ku.dk/squirrelmail](http://www.fys.ku.dk/squirrelmail) - vi har links til dem fra edb-afdelingens hjemmeside. For begge systemer gælder det, at man til at starte med logger ind med sit brugernavn og password, og derefter kommer man til et system, hvorfra man kan læse ens email og sende emails til andre.

### 8.1.2 Administration af konto

Fra edbafdelingens hjemmeside er der links til forskellige administrations sider for brugerne af ØLs system. I skrivende stund inkluderer det:

**Ændring af password.** Hvis man ønsker at ændre sit password, eller hvis man har fået en mail om at man *skal* ændre sit password, så skal man gøre det vha. en website.

**Ændring af shell.** Hvis man ønsker en anden kommandofortolker end standarden kan det sættes fra denne side.

**Aktivering af spamfilter.** ØLs system har indeholder mulighed for at frasortere uønsket, kommerciel mail (spam) fra ens mailkonto. Det kræver dog, at man har forstået visse egenskaber ved filtret, hvorfor man selv skal aktivere filtret. Læs mere om spam i afsnit 8.1.2

**Videreforsendelse af email til anden adresse.** Denne facilitet er rar at have, hvis man for en tid eksempelvis skal til et universitet i udlandet, og der får en anden email adresse. Istedet for hele tiden at læse fra to emailkontoer, kan man videresende alle mails fra den ene konto til den anden - dette gøres vha. denne side på ØLs system.

**Automatisk svarbesked på emails.** Hvis man på et tidspunkt tager på ferie eller andetsteds hvor man ikke ønsker eller har mulighed for at læse sine emails, kan man på denne siden bede systemet om at sende en meddelelse til alle, der sender en emails om, at man ikke vil læse dem før man kommer tilbage fra ferie.

## 8.2 Personlig website

Alle brugere af ØLs system har adgang automatisk en hjemmeside på deres konto: Alt, hvad der ligger i `~/public_html/` kan tilgås på hjemmesiden `www.fys.ku.dk/~<brugernavn>/`, hvor `<brugernavn>` er ens login på systemet. Webserveren forsøger automatisk at vise siden `index.html` i kataloget - hvis den ikke eksisterer, vises en oversigt over filer i kataloget.

### 8.2.1 Scripts

Ovenstående hjemmeside tillader ikke andet end almindelig html, dvs. man kan ikke anvende php eller cgi scripts. Der er dog en server på ØLs netværk, som tillader at man anvender disse faciliteter, nemlig `php.fys.ku.dk`. Denne server tillader at man anvender disse faciliteter, men man skal bruge lidt mere tid på at sætte en hjemmeside op på den. Det kræver principielt to ting: Man skal have en konto på `php.fys.ku.dk`, og man skal uploade sin hjemmeside til denne konto. `php.fys.ku.dk` administreres *ikke* af edb-afdelingen, så for at få en konto, skal man henvende sig til `php@fys.ku.dk` - man behøver ikke anden "kvalifikation" end at være bruger på ØLs system, men for at have lidt styr på hvem der bruger `php.fys.ku.dk` skal man lige maile administratorerne af den. Når man har fået sin konto, uploades siden til den vha. `sftp(1)` - login/password er det samme som til resten af systemet. For at tilgå den gennem ØLs webserver, skal man lægge en automatisk henvisning i sin `index.html`-fil til `http://php.fys.ku.dk/~<brugernavn>/`.



## Kapitel 9

# Printersystemet

I systemets terminalrum er der placeret printere til de studerende, ligesåvel som der forskellige steder på HCØ er placeret printere til de ansatte. "Printersystemet" er det system, der holder styr på:

- De forskellige printere på instituttet (se 9.1).
- Hvilken type dokument der sendes til printeren (se afsnit 9.2).
- Printer køer til hver printer. Hvis der er flere printjobs til én printer, bliver de resterende jobs sat midlertidigt i kø og udskrevet efter tur (se afsnit 9.3).

### 9.1 Hvilke printere er til rådighed?

For studerende er der en laserprinter til rådighed i hvert terminalrum, navngivet med terminalrummets bogstav efterfulgt af ps (for **p**ostscript) - dvs. `sps`, `aps`, `kps` og `nps` i hhv. s-lokalet på 3. sal i D-bygningen, rum 315 (for fysikere og nanoer), a-lokalet på 1. sal i vandrehallen, rum 112 (for alle), k-lokalet på 1. sal i vandrehallen, rum 108 (undervisningsloale for kemikere), og n-lokalet på NBI, rum HA1 (for fysikere). For en total liste af printere på systemet kan man gå ind på printere i edb-afdelingens hjemmeside. Ønsker man at udskrive på begge sider af papiret (nyttigt ved meget lange dokumenter), skal man anvende printernavnet med et d (**d**uplex) bagefter, dvs. `spsd` er printeren i D315 med duplex. Hvis ikke man angiver en printer, udskrives normalt til printeren i det rum man printede fra, dvs. f.eks. `kps`, hvis man sidder i rum A108.

### 9.2 Hvordan udskriver man?

UNIX-kommandoen til udskrivning hedder `lpr` (**l**ine-**p**rinter). Den bruges på følgende måde:

```
lpr -P<printernavn> <filnavn>
```

<printernavn> er en af printerne fra afsnit 9.1, mens <filnavn> er navnet på den fil der skal udskrives.

Hvis man eksempelvis vil udskrive filen `foo.txt` til printer `aps`, bruges følgende kommando:

```
datamat > lpr -Pps foo.txt
```

Det er værd at bemærke at printersystemet har et “magisk filter”. Det betyder, at det automatisk kan genkende forskellige filtyper og sørge for at udskrive dem korrekt. F.eks. tekst, postscript, gif eller dvi. Bemærk også, at man *ikke* kan udskrive Microsoft Windows printerfiler eller Microsoft Word filer! (En løsning kan dog være, at installere en *postscript-printerdriver* – f.eks. “Apple LaserWriter II NT” – og så printe til en fil, man derefter tager med på HCØ og skriver ud med *lpr*). Man kan skrive *Postscript*-filer ud med programmet *ghostview(1)* og Adobe Acrobat (*pdf*) filer ud med programmet *acroread*.

### 9.3 Hvad er der i printer køen?

For at se indholdet af printer køen for en given printer bruges *lpq* (**line-printer-queue**):

```
lpq -P<printernavn>
```

F.eks. vil kommandoen *lpq -Psps* vise en oversigt over køen for printer *sps* i stil med:

```
datamat > lpq -Psps
Printer: studps@kirstine (originally sps) 'D315, HP Laserjet 2200DN, duplex, postscript'
Queue: 2 printable jobs
Server: pid 29583 active
Unspooler: pid 29585 active
Status: waiting for subserver to exit at 13:27:52.156
```

Rank	Owner/ID	Class	Job	Files	Size	Time
active	treue@datamat+185	A	185	foo.ps	56942	13:27:51
2	treue@datamat+186	A	186	bar.ps	56942	13:27:52

Her ses at brugeren “treue” har udskrevet filen “foo.ps” fra maskinen “datamat” og det pågældende printjob har fået nummer 185 i printer køen. “Rank” for dette job er “active”, hvilket betyder at det aktivt er ved at blive udskrevet. Brugeren har også udskrevet filen “bar.ps”, men den venter på at “foo.ps” bliver færdig og har derfor fået “Rank” 2.

### 9.4 Hvordan sletter man sit printjob fra printer køen?

Kommandoen til at slette et printjob fra en printer kø hedder *lprm* (**line-printer-remove**):

```
lprm -P<printernavn> <jobid>
```

<jobid> kan være to ting: Enten et brugernavn, hvilket i så fald vil fjerne alle den pågældende brugers jobs fra printer køen for printeren <printernavn>, eller nummeret på printjobbet fra *lpq*. Hvis man eksempelvis vil slette printjobbet “bar.ps” fra eksemplet i afsnit 9.3, køres følgende:

```
datamat > lprm -Psps 930
```

### 9.5 Hvordan skifter man papir/toner i printeren?

Hvis nogle af printerne på HCØ mangler papir eller toner, kan man sende en mail til fejl@fys.ku.dk - vi vil så hurtigst muligt komme med nye forsyninger. På printeren på NBI står der en email adresse som kan kontaktes for papir/toner til printeren nps.

## Kapitel 10

# Tekstbehandling og L<sup>A</sup>T<sub>E</sub>X

Systemet tilbyder to principielt forskellige tekstbehandlingsværktøjer: L<sup>A</sup>T<sub>E</sub>X og officeprogrammet StarOffice. Mange brugere vil sikkert allerede være bekendt med Microsoft Office<sup>TM</sup> - StarOffice er en implementation af den samme ide, lavet til linux af Sun Microsystems. Brugere af Microsoft Office vil formentligt uden nogen problemer kunne anvende StarOffice, da dette til dels er lavet med det formål at ligne Microsoft Office mest muligt - derfor vil de ikke blive gennemgået nøjere her. Det er dog IT-afdelingens erfaring, at man hurtigt kommer i problemer, hvis man forsøger at klare sig igennem et naturvidenskabeligt studie udelukkende med Office programmer - ingen af dem er specielt gode til at håndtere mange formler i et dokument osv. Vi anbefaler derfor, at man gør sig selv den tjeneste at lære at bruge L<sup>A</sup>T<sub>E</sub>X som vi vil bruge resten af dette kapitel på at give en kort, men (for begyndere) tilstrækkelig, introduktion til.

Bruger man WYSIWYG<sup>1</sup>-tekstbehandlingsværktøjer (f.eks. Microsoft Word<sup>TM</sup>), oplever man måske, at man bruger uforholdsmæssigt lang tid på at sætte teksten pænt op, i forhold til hvor lang tid man bruger på at formulere sig.

L<sup>A</sup>T<sub>E</sub>X (udtales "latek") søger at komme dette misforhold til livs ved at ordne al opsætning automatisk, så brugeren kan koncentrere sig fuldstændigt om *indholdet* af den tekst, han er ved at producere. Brugeren skal blot fortælle, hvad funktion en del af teksten har (om det er en overskrift, brødtekst, et citat, osv.), så vil L<sup>A</sup>T<sub>E</sub>X selv sætte teksten pænt op. Brugeren giver L<sup>A</sup>T<sub>E</sub>X en tekstfil, og L<sup>A</sup>T<sub>E</sub>X producerer så udfra denne tekstfil et nydeligt typesat dokument. Eksempelvis ser starten af dette kapitel således ud, før det har været igennem L<sup>A</sup>T<sub>E</sub>X:

```
\chapter{Tekstbehandling og \LaTeX}
```

```
Systemet tilbyder to principielt forskellige tekstbehandlingsværktøjer:  
\LaTeX\ og officeprogrammet StarOffice.
```

Ord der starter med \ er direktiver til L<sup>A</sup>T<sub>E</sub>X, så den første linie er en instruktion til L<sup>A</sup>T<sub>E</sub>X om at her begynder et nyt kapitel med titlen L<sup>A</sup>T<sub>E</sub>X – direktivet \LaTeX sætter L<sup>A</sup>T<sub>E</sub>X-logoet. Herefter begynder selve teksten, der snart afbrydes af en fodnote.

Umiddelbart kan det virke noget hæmmende at overlade al kontrol over layout til L<sup>A</sup>T<sub>E</sub>X, men vænner man sig først til det, ser man det snarere som en befrielse. Når jeg skriver denne tekst, kan jeg fokusere udelukkende på at bringe mit budskab igennem og være trygt forvisset om, at L<sup>A</sup>T<sub>E</sub>X vil få min tekst til at tage sig godt ud. Design og layout af dokumenter er en kunstart i sig selv (se f.eks. [2]), så det er egentlig rimeligt at antage, at L<sup>A</sup>T<sub>E</sub>X, der (til en vis grad) er lavet

---

<sup>1</sup>What You See Is What You Get

af professionelle og dermed trækker på snart 500 års layout- og trykkertradition, kan gøre det væsentlig bedre end jeg selv kan.

Medaljen har selvfølgelig en bagside. Dels har L<sup>A</sup>T<sub>E</sub>X, som de fleste andre værktøjer vi her har præsenteret, en indlæringskurve der er noget stejlere end et WYSIWYG værktøj. Dels kan det være vanskeligt at tvinge L<sup>A</sup>T<sub>E</sub>X til bestemte layout-beslutninger, hvis man insisterer på at ens dokument skal se anderledes ud.

Dette kapitel introducerer kort grundlæggende L<sup>A</sup>T<sub>E</sub>X-begreber, og forsøger at give tips til at bruge L<sup>A</sup>T<sub>E</sub>X i forbindelse med rapportopgaver. L<sup>A</sup>T<sub>E</sub>X er imidlertid et stort system, så interesserede bør skaffe sig en eller flere af de lidt længere introduktioner, der refereres til sidst i dette kapitel. Flere af disse introduktioner er frit tilgængelige på det Internettet. Emacs har en vældig fiks L<sup>A</sup>T<sub>E</sub>X-mode; læs mere herom i afsnit 14.1.3.

## 10.1 Et kort L<sup>A</sup>T<sub>E</sub>X dokument

Et L<sup>A</sup>T<sub>E</sub>X-dokument falder i to dele: et *preamble* og selve teksten. I *preamble* instruerer man L<sup>A</sup>T<sub>E</sub>X om hvilken type dokument man ønsker at forfatte, f.eks. en artikel, et brev, en rapport osv., og man specificerer enkelte designbeslutninger. Lad os se et kort L<sup>A</sup>T<sub>E</sub>X-dokument:

```
\documentclass{article}

\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[danish]{babel}

\begin{document}
Dette er et en-linies \LaTeX-dokument!
\end{document}
```

Den årvågne læser vil have gættet, at selve teksten er den linie, der befinder sig imellem `\begin` hhv. `\end{document}`. Før teksten kommer et indledende *preamble*, hvor man angiver dokumentets overordnede layout samt eventuelle udvidelser man ønsker at benytte.

Det overordnede layout skal altid angives som det allerførste med direktivet `\documentclass`. Her laver vi en artikel, men vi kunne også have valgt en af klasserne `book` eller `report`.

Udvidelser til L<sup>A</sup>T<sub>E</sub>X giver typisk brugeren flere direktiver eller ændrer dokumentets layout indenfor rammerne af den valgte dokumentklasse. Udvidelser hentes med direktivet `\usepackage`. I dokumentet ovenfor anvendes udvidelserne `inputenc` og `fontenc`, der sætter L<sup>A</sup>T<sub>E</sub>X istand til at håndtere danske bogstaver, og `babel`, der vælger danske betegnelser som `kapitel`, `bibliografi` og `indholdsfortegnelse` i stedet for de engelske `chapter`, `bibliography` og `table-of-contents`. Læg mærke til at pakken angives i krølleparentes, og argumenter til pakken angives i firkantede parenteser (i dette tilfælde instruerer vi `babel` om at vælge danske betegnelser; vi kunne også have valgt f.eks. `tyske`).

## 10.2 Oversættelse og udskrift af L<sup>A</sup>T<sub>E</sub>X-dokumenter

Lad os antage, at vi har ovenstående tekst i filen `doku1.tex`; vi vil nu gerne oversætte dokumentet, se hvordan det tager sig ud (preview, om man vil), og til sidst udskrive det. Først kører vi `doku1.tex` igennem `latex(1)`:

```

datamat > latex doku1.tex
This is TeX, Version 3.14159 (Web2C 7.3.1)
(doku1.tex
LaTeX2e <1999/12/01> patch level 1
Babel <v3.6Z> and hyphenation patterns for american, british,
danish, nohyphenation, loaded.
(/usr/local/stow/share/texmf/tex/latex/base/article.cls
Document Class: article 1999/09/10 v1.4a Standard LaTeX document
class
(/usr/local/stow/share/texmf/tex/latex/base/size10.clo))
(/usr/local/stow/share/texmf/tex/latex/misc/isolatn1.sty
ISO-latin-1 input coding, version 0.9 of 1-Jun-1992.)
(/usr/local/stow/share/texmf/tex/generic/babel/babel.sty
(/usr/local/stow/share/texmf/tex/generic/babel/danish.ldf
(/usr/local/stow/share/texmf/tex/generic/babel/babel.def)))
No file doku1.aux.
[1] (doku1.aux) )
Output written on doku1.dvi (1 page, 328 bytes).
Transcript written on doku1.log.

```

L<sup>A</sup>T<sub>E</sub>X producerer en frygtelig masse uddata, men man kan trygt ignorere det meste af det. Resultatet af kørslen er en såkaldt DVI-fil<sup>2</sup>, i dette tilfælde filen `doku1.dvi`. Vi kan nu se hvordan dokumentet tager sig ud med programmet `xdvi(1)`

```
datamat > xdvi doku1.dvi
```

Man behøver ikke at starte `xdvi` forfra hvis man oversætter sit L<sup>A</sup>T<sub>E</sub>X-dokument igen, for `xdvi` holder selv øje med om `dvi`-filen ændres, og genindlæser den om nødvendigt (man vil dog normalt skulle klikke i `xdvi` vinduet, før dette opdateres).

Vi kan selvfølgelig også udskrive filen. Dette kræver dog normalt at `dvi`-filen oversættes til POSTSCRIPT, en art interlingua for printere. Dette gøres med programmet `dvips(1)`, f.eks. ville man i ovenstående tilfælde anvende kommandoen

```
datamat > dvips doku1.dvi -o doku1.ps
```

Den årvågne læser vil have gættet at dette resulterer i en fil med navnet `doku1.ps` (det var det, vi specificerede med `-o doku1.ps`). Denne kan nu printes i sin helhed via `lpr` (se afsnit 9.2), eller den kan vises med programmet `gv` (se afsnit 11.3.2), som også indeholder faciliteter til at printe udvalgte dele af dokumentet.

Det hænder selvsagt at man laver fejl i sine L<sup>A</sup>T<sub>E</sub>X-dokumenter; Typisk vil man glemme at sætte en krølleparantes `'` eller lignende. Skulle dette ske, vil L<sup>A</sup>T<sub>E</sub>X standse og give en fejlmeddelelse. Hvis vi f.eks. fjerner den højre krølleparantes fra `\begin{document}` får vi følgende resultat:

```

datamat > latex doku1-med-fejl.tex
This is TeX, Version 3.14159 (Web2C 7.3.1)
(doku1-med-fejl.tex
LaTeX2e <1999/12/01> patch level 1
Babel <v3.6Z> and hyphenation patterns for american, british,
danish, nohyphenation, loaded.

```

---

<sup>2</sup>DVI står for DeVice Independent, der refererer til, at DVI-filer ikke er rettet mod nogen bestemt printer.

```
(/usr/local/stow/share/texmf/tex/latex/base/article.cls
Document Class: article 1999/09/10 v1.4a Standard LaTeX document
class
(/usr/local/stow/share/texmf/tex/latex/base/size10.clo))
(/usr/local/stow/share/texmf/tex/latex/misc/isolatin1.sty
ISO-latin-1 input coding, version 0.9 of 1-Jun-1992.)
(/usr/local/stow/share/texmf/tex/generic/babel/babel.sty
(/usr/local/stow/share/texmf/tex/generic/babel/danish.ldf
(/usr/local/stow/share/texmf/tex/generic/babel/babel.def))))
Runaway argument?
{document Dette er et en-linies \LaTeX -dokument! \end {document}
! File ended while scanning use of \begin.
<inserted text>
\par
<*> doku1-med-fejl.tex

?
```

Her kan man enten trykke `<return>`, og L<sup>A</sup>T<sub>E</sub>X ignorerer fejlen, eller man kan trykke `<x>` eller `<ctrl-c>` for at afbryde. Emacs' L<sup>A</sup>T<sub>E</sub>X-mode kan håndtere oversættelse af L<sup>A</sup>T<sub>E</sub>X-dokumenter automatisk, se 14.1.3.

L<sup>A</sup>T<sub>E</sub>X har reserveret visse tegn til specielt brug. Det drejer sig om tegnene

`\ { } $ & # ^ _ % ~`

En meget almindelig fejl er at komme til at bruge `'&'` eller `'_'` i brødteksten. På nær tegnet `'\'` kan man bruge tegnene alligevel, man skal blot sætte en `'\'` foran. Eksempelvis sættes tegnet `'&'` med direktivet `\&`. Hvis man har brug for at sætte et andet specialtegn, kan man bruge direktivet `\verb`; således sætter `\verb+\+` tegnet `'\'`.

## 10.3 Grundlæggende teknikker

Al tekst opfattes af L<sup>A</sup>T<sub>E</sub>X som brødtekst, med mindre der er udstedt et direktiv, der siger noget andet. L<sup>A</sup>T<sub>E</sub>X sørger selv for at bryde brødteksten ned i linier, idet den deler ord hvor det er nødvendigt. Her er et eksempel; først den tekst der står i filen `eks1.tex` som jeg har tastet den ind<sup>3</sup>, herefter teksten som den ser ud, når den kommer ud af L<sup>A</sup>T<sub>E</sub>X.

Siden Galileo's berømte eksperiment med frit faldende legemer har man vidst, at alle legemer, på det samme sted på jordkloden, falder med den samme acceleration, når man ser bort fra luftmodstand. Siden har talrige eksperimenter gjort dette til et veletableret, videnskabeligt faktum.

Siden Galileo's berømte eksperiment med frit faldende legemer har man vidst, at alle legemer, på det samme sted på jordkloden, falder med den samme acceleration, når man ser bort fra luftmodstand. Siden har talrige eksperimenter gjort dette til et veletableret, videnskabeligt faktum.

---

<sup>3</sup>frit oversat fra [3]

Læg mærke til, at  $\LaTeX$  har ikke har ombrudt linierne som angivet i kildeteksten. Ønsker man at starte et afsnit uden automatisk indrykning kan man udstede direktivet `\noindent` før det første bogstav i afsnittet.

Sædvanligvis vil man gerne dele selve brødteksten ind i afsnit af 5–6 linier, adskilt ved indrykning af den første linie i hver klump.  $\LaTeX$  starter et nyt afsnit, hver gang der læses en tom linie. Hvis vi fortsætter eksemplet fra før:

Siden Galileo's berømte eksperiment med frit faldende legemer har man vidst, at alle legemer, på det samme sted på jordkloden, falder med den samme acceleration, når man ser bort fra luftmodstand. Siden har talrige eksperimenter gjort dette til et veletableret, videnskabeligt faktum.

Efter at Newton havde formuleret sine bevægelseslove, fik Galileo's observationer en dybere mening: I Newtonsk mekanik anses kraften, som et objekt udsættes for, for at være *\emph{årsagen}* til objektets acceleration. Objektets masse er et kvantitativt mål for det faktum, at objektet er trægt. De bevægelseslove, som galileo formulerede, burde kunne forklares udfra Newtons love.

Siden Galileo's berømte eksperiment med frit faldende legemer har man vidst, at alle legemer, på det samme sted på jordkloden, falder med den samme acceleration, når man ser bort fra luftmodstand. Siden har talrige eksperimenter gjort dette til et veletableret, videnskabeligt faktum.

Efter at Newton havde formuleret sine bevægelseslove, fik Galileo's observationer en dybere mening: I Newtonsk mekanik anses kraften, som et objekt udsættes for, for at være *årsagen* til objektets acceleration. Objektets masse er et kvantitativt mål for det faktum, at objektet er trægt. De bevægelseslove, som galileo formulerede, burde kunne forklares udfra Newtons love.

`\emph`-direktivet fremhæver (**emphasize**) tekst.

Som udgangspunkt adskiller  $\LaTeX$  afsnit ved indrykning. Man kan styre hvor stor indrykning der skal være, og evt. hvor stor vertikal afstand der skal være imellem afsnit med direktiverne `\parindent` hhv. `\parskip`. Denne bog er sat med direktiverne:

```
\parindent=0pt
\parskip=8pt plus 2pt minus 4pt
```

Den sidste linie angiver, at afstanden imellem afsnit helst skal være 8 punkter, men at  $\LaTeX$  efter behov kan udvide afstanden med op til 4 punkter eller mindske den med op til 2 punkter.

Den allermest fundamentale typografiske teknik er opdelingen af en tekst i kapitler, afsnit, underafsnit og så videre. I  $\LaTeX$  inddeler man med direktiverne `\chapter`, `\section`, `\subsection`, `\subsubsection`. Direktiverne tager en titel som argument, så ønsker jeg at påbegynde et nyt underafsnit med titlen "Moderne krypteringsalgoritmer" udsteder jeg direktivet

```
\subsection{Moderne krypteringsalgoritmer}
```

Inddelingsdirektiverne vil få forskelligt typografisk udtryk afhængigt af hvilken `\documentclass` man har valgt. I dokumentklassen `article` kan man ikke bruge direktiverne `\chapter` og `\part`.

### 10.3.1 Typografiske spidsfindigheder

Gåseøjne bør altid vende ind mod ordet. Sammenlign f.eks. “funktion” med ”funktion”. De korrekte gåseøjne opnås med enkeltplinger i den rigtige retning:

De ‘‘korrekte’’ gåseøjne ...

I Emacs L<sup>A</sup>T<sub>E</sub>X-mode kan man imidlertid bare trykke " når man egentlig mener ‘‘ eller ’’, så gætter Emacs sig til hvilken man mente. Se 5.3.1.

Til tider vælger L<sup>A</sup>T<sub>E</sub>X at bryde en linie imellem to ord, der hænger så tæt sammen, at det virker forstyrrende, f.eks. i en sidehenvisning imellem ordet “side” og sidetallet. Man kan angive et mellemrum i hvilket en linie ikke må brydes med ~:

... se mere herom side~28.

Man kan anbefale L<sup>A</sup>T<sub>E</sub>X at lave et linie- eller sideskift med direktivet \goodbreak. L<sup>A</sup>T<sub>E</sub>X giver point til hvert potentielle lineskift, så det at den anbefaler et sted betyder blot, at man tildeler det sted, man udsteder direktivet, lidt flere point.

Man kan tvinge et lineskift henholdsvis et sideskift igennem med direktiverne \\ henholdsvis \newpage.

### 10.3.2 L<sup>A</sup>T<sub>E</sub>X og matematik

Den enkeltfacilitet, der har gjort L<sup>A</sup>T<sub>E</sub>X til en de facto-standard i næsten alle fag med matematisk islæt er (stor overraskelse) at L<sup>A</sup>T<sub>E</sub>X er enormt god til at sætte matematik. Vi vil ikke her gå dybt ind i disse faciliteter, primært pga. pladshensyn, men interesserede læsere vil med fordel kunne anvende referencerne i slutningen af dette kapitel til at gøre sig fortrolige med L<sup>A</sup>T<sub>E</sub>X's matematik faciliteter.

Fra tid til anden kan det dog være rart at skrive et x eller et α, så vi forklarer her meget kort det allermest nødvendige. Ønsker man at udtrykke matematiske symboler, sætter man disse imellem to dollartegn. Bogstaver bliver nu sat med et skriftsnit der passer til variable, tal ser en anelse anderledes ud, og almindeligt forekommende symboler som +, −, ×, ÷, = osv. giver anledning til ekstra afstand. Samtidigt overtager L<sup>A</sup>T<sub>E</sub>X al kontrol med mellemrum.

Multiplikation skrives \times, division \div (for en skråstreg) eller \frac{tæller}{nævner} for en brøk. Eksempelvis resulter  $\frac{x+1}{2} + 3 \times y$  i  $\frac{x+1}{2} + 3 \times y$ . Fodtegn eller løftede tegn angives med henholdsvis \_ og ^, og det der skal sænkes eller løftes samles imellem klammer, hvis det er mere end et bogstav:  $x_i + y^{2+i}$  giver  $x_i + y^{2+i}$ . De fleste almindeligt forekommende funktioner findes som direktiver: \lg n + 2\sin\pi giver  $\lg n + 2 \sin \pi$ . Ønsker man en formel skrevet på en linie for sig selv, benytter man to dollartegn istedet for et:

$$\sum_{k=0}^n \frac{1}{k} = O(\lg n)$$

giver

$$\sum_{k=0}^n \frac{1}{k} = O(\lg n)$$

Se dette kapitals sidste afsnit (10.5) for at finde en oversigt over hvilke direktiver der frembringer hvilke matematiske symboler.



## 10.4 Tips til rappportskrivning

I dette afsnit gennemgås tricks og teknikker, som man typisk vil ønske sig i forbindelse med rapportopgaver på studiet. Det anbefales, at læseren blot skimmer dette afsnit for at danne sig overblik over mulighederne, og så vender tilbage når det bliver aktuelt at benytte en af teknikkerne herfra. Læseren vil måske finde det værd at tage et blik på kapitlets allersidste afsnit 10.5.

### 10.4.1 Indholdsfortegnelser

Hvis man bruger inddelingsdirektiverne (`\section` osv.) kommer titlen med i  $\LaTeX$ s indholdsfortegnelse. Indholdsfortegnelsen bliver placeret det eller de steder i ens dokument man udsteder direktivet `\tableofcontents`. Ønsker man noget andet end afsnittets titel i indholdsfortegnelsen, kan man angive dette “andet” som ekstra argument:

```
\section[Kvantegravitation]{Min formulering af kvantegravitationsteorien}
```

Ønsker man ikke at få afsnittet i indholdsfortegnelsen skriver man en `*` før titlen:

```
\subsubsection*{RSA}
```

Til tider er det nødvendigt at køre et dokument igennem  $\LaTeX$  to gange for at få indholdsfortegnelsen eller andre krydsreferencer korrekt;  $\LaTeX$  skriver hvis det er nødvendigt. Dette er ingen programfejl, det er et udtryk for at  $\LaTeX$  på nogen punkter er et *meget* gammeldags stykke programmel.

### 10.4.2 Titelblad

En skriftlig fremstilling får en mere afsluttet karakter, hvis den har et titelblad. Dette gøres ved at lave følgende ændringer i ens *preamble*:

```
\documentclass[titlepage]{...}
...
\title{Overvejelser om kvantecomputere}
\author{T. Ilfældig Fysiker \and J. Random Chemist}
\date{\today}
...
\begin{document}
\maketitle
...
```

Den første linie gør, at titelbladet får en side for sig selv (i nogle dokumenttyper som f.eks. `article` sker dette ikke automatisk). Direktiverne `\title`, `\author`, `\date` og `\today` er vist oplagte; bemærk direktivet `\and`, der fortæller  $\LaTeX$  hvornår et nyt forfatternavn begynder. Tilsidst skal man, som med indholdsfortegnelsen, angive, at titelbladet skal indsættes med direktivet `\maketitle`.

### 10.4.3 Krydsreferencer

Ofte har man lyst til at referere frem eller tilbage i teksten, se f.eks. afsnit 10.4.3 side 54. I L<sup>A</sup>T<sub>E</sub>X udsteder man direktivet `\label` for at markere en position i ens dokument, som man så senere kan referere til, enten ved afsnitsnummer, med direktivet `\ref`, eller ved sidetal, med direktivet `\pageref`. Eksempelvis ser starten af dette (under-)afsnit således ud:

```
\subsection{Krydsreferencer}
\label{latex:xref}
Ofte har man lyst til at referere frem eller tilbage i teksten,
se f.eks. afsnit~\ref{latex:xref}, side~\pageref{latex:xref}.
```

### 10.4.4 Fodnoter

Man kan lave en fodnote med direktivet `\footnote`<sup>4</sup>:

```
Man kan lave en fodnote med direktivet~\lcmdI{footnote}%
\footnote{Man kan dog ikke bruge fodnoter inden i tabeller.}:
```

Procenttegnet sidst i første linie er et kommentartegn, der får L<sup>A</sup>T<sub>E</sub>X til at ignorere resten af linien. Det er nødvendigt her, for at forhindre L<sup>A</sup>T<sub>E</sub>X i at sætte et mellemrum imellem `\footnote` og fodnotemarkøren. Hvis procenttegnet ikke havde været der, ville L<sup>A</sup>T<sub>E</sub>X have opfattet lineskiftet imellem de to linier som et mellemrum.

### 10.4.5 Skriftsnit

I L<sup>A</sup>T<sub>E</sub>X kan man ikke umiddelbart vælge hvilket skriftsnit man ønsker at benytte. Det kan virke som en noget voldsom restriktion, men det giver god mening: For at ens dokument skal være bare nogenlunde at se på, skal der være en vis harmoni imellem f.eks. skriftsnittet til brødteksten og skriftsnittet til overskrifter.

Istedet for at vælge enkelte skriftsnit kan man derfor istedet vælge hele pakker med skriftsnit, der definerer skriftsnit til brødtekst, overskrifter, indhold mv. Skriftsnit pakkerne opfører sig fulstændigt som alle mulige andre pakker; man anvender dem ved at bruge `\usepackage` direktivet.

De mest interessante af de tilgængelige pakker på systemet er `charter`, `palatino`, `utopia`, `bookman` og `times`. Skriftsnit til matematik vælges separat fra resten, her findes bl.a. `euler`. Pakken `palatino` fungerer udmærket med begge.

Denne bog benytter `charter`:

```
\usepackage{charter}
```

### 10.4.6 Lister

- Nogle gange kan man lette overblikket ved at sætte ting op punktvis i lister.
- Andre gange bliver det bare forvirrende.

---

<sup>4</sup>Man kan dog ikke bruge fodnoter inden i tabeller.

- $\text{\LaTeX}$  har flere forskellige slags lister:
  1. Lister med numre (som denne indlejrede liste),
  2. Punktlister (som den ydre liste)
  3. Beskrivelser (som den næste liste)
- Alle disse lister anvendes på (ca) samme måde.

En beskrivelse er en liste, der istedet for punkter eller tal bruger fremhævet tekst som overskrift. Feks. hedder de tre listetyper

**itemize** benytter blot • som punktoverskrift,

**enumerate** benytter tal,

**description** benytter markerede ord. Denne listeform ser klart bedst ud, hvis nogle punkter har tekst, der fylder mere end blot en enkelt linie.

De tre fremhævede ord er præcis titlen på de listetyper  $\text{\LaTeX}$  genkender. Man angiver en punktlister med

```
\begin{itemize}
  \item Første punkt.
  \item Andet punkt, dette punkt er der mere tekst ved,
        faktisk så meget, at jeg skal skift linie.
  ...
\end{itemize}
```

Ønsker man en nummereret liste benytter man blot `enumerate` istedet for `itemize`; resten er ens, idet  $\text{\LaTeX}$  selv tæller punkterne op. Beskrivelser fås med `description`, og her angiver man det fremhævede ord i klammer:

```
...
\item[enumerate] benytter tal,
...
```

### 10.4.7 Tabeller

Ofte er det mest overskueligt at opstille resultater i en tabel. Feks. resultaterne fra et fysisk forsøg. Her er et eksempel:

```

\begin{center}
\begin{tabular}{|l|rr|p{6.5cm}|}
\hline
Forsøg      & Tid(sek.)      & afstand(meter) & Kommentar \\
\hline
A           & 8,5            & 4,33           & \\
B           & 73,4           & 2,3           & 
        Det ene gruppemedlem glemte at
        slukke for stopuret. Bør ikke
        medtages i databehandling\\
C           & 12,4           & 6,42           & \\
D           & 3,7           & 1,2           & Stærk modvind\\
\hline
\end{tabular}
\end{center}

```

Forsøg	Tid(sek.)	afstand(meter)	Kommentar
A	8,5	4,33	
B	73,4	2,3	Det ene gruppemedlem glemte at slukke for stopuret. Bør ikke medtages i databehandling
C	12,4	6,42	
D	3,7	1,2	Stærk modvind

Den anden klamme i anden linie, `{|l|rr|p{6.5cm}|}`, angiver tabellens format. Hvert tegn i klammen angiver en kolonne. En lodret streg betyder, at kolonnen blot består af en lodret streg (dvs. adskildelsen mellem de andre kolonner). Et 'l', 'c', 'r' eller 'p' angiver justering af teksten i kolonnen, henholdsvis left, center right eller paragraph. Den sidste betyder, at man har til hensigt at skrive almindelig tekst der skal linieombrydes, og man skal som parameter angive bredden af feltet.

Herefter følger selve tabellen. En kolonne angives, ved at man blot skriver almindelig  $\text{\LaTeX}$ -brødtekst adskilt af &-tegn. Et tvungent lineskift (`\\`) afslutter kolonnen. Der må gerne være færre kolonner end angivet i starten af tabellen, og &-tegnene springer automatisk over de lodrette streger. Direktivet `\hline` sætter en vandret streg igennem hele tabellen. Mellemrum omkring &-tegnene er ligegyldige;  $\text{\LaTeX}$  retter selv tabellen til.

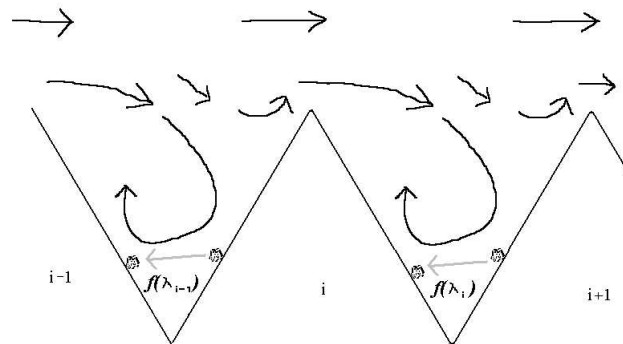
Direktivet `\center` gør, at det direktivet omgiver bliver centreret horisontalt; sædvanligvis er det sådan, man gerne vil have tabeller sat.

Det er selvsagt muligt at lave væsentlig mere komplicerede tabeller, men det er *yderst* sjældent at man har en fornuftig grund til at gøre det. I langt de fleste tilfælde gør delte eller dobbelte celler blot en tabel langt mere uoverskuelig. Hvis der absolut ikke er nogen vej udenom, så kan man opnå mere komplicerede tabeller med `\multicolumn` direktivet; læs om det i f.eks. [6].

For en omfattende gennemgang af hvad der er god og dårlig stil mht. tabeller (eller i det hele taget) henvises til [2, kapitel 12].

### 10.4.8 Figurer

Komplekse systemer beskrives ofte nemmere med en figur end med ord. Figur 10.1 er f.eks. en figur, som illustrerer bølgers transport af sand.



Figur 10.1: Illustration fra en (aldeles ulæselig!) rapport som forfatterne lavede på første år af fysikstudiet

$\text{\LaTeX}$  har indbyggede direktiver til at tegne, men i næsten alle tilfælde er det mere bekvemt at anvende et grafisk værktøj. Det mest udbredte sådanne værktøj er `xfig(1)`, som forklares i afsnit 11.2.3. Når man har tegnet sin figur, vælger man menupunktet `(file→export)` og eksporterer diagrammet som *encapsulated POSTSCRIPT*. Benytter man pakken `epsfig` kan man nu få diagrammet medtaget i ens dokument med direktivet `\epsfig`.

Har man behov for at inkludere egentlige billeder, f.eks. i gif-, bpm- eller jpg-format, er det nemmest at konvertere billederne til indkapslet POSTSCRIPT, og så inkludere dem som ovenfor ved hjælp af pakken `epsfig`. Programmet `convert(1)` (se afsnit A) kan konvertere fra mange populære billedformater til indkapslet POSTSCRIPT.

Meget rutinerede  $\text{\LaTeX}$ -brugere med aggressivt behov for diagrammer anvender ofte pakken `pstricks`. Matematikere med hang til kommutative diagrammer kan tage et kig på `xypic`. Dokumentation af begge disse pakker findes elektronisk, se 10.5.

Store tabeller eller store diagrammer bør “flyde” rundt imellem siderne, så der ikke kommer store hvide områder. Dette betyder tilgængæld at man ikke altid kan styre hvor billeder, tabeller eller diagrammer ender henne.  $\text{\LaTeX}$  har sofistikerede metoder til at opnå denne effekt på, men at udnytte disse metoder rækker udover denne introduktion. Vi henviser til en af de lidt længere introduktioner, f.eks. [6] eller en af de andre der henvises til i afsnit 10.5.

### 10.4.9 Stavekontrol

Talløse studerende er tilsyneladende ikke klar over eksistensen af stavekontrol eller deres behov for samme. Skulle man være en af de stadigt færre der staver upåklageligt, vil der alligevel uvægerligt indsnige sig slåfejl i ens opgaver. Heldigvis findes der også under UNIX en stavekontrol.

Lad os antage, at jeg har skrevet en rapport i  $\text{\LaTeX}$ , og at den er fordelt over filerne `del1.tex`, `del2.tex` og `del3.tex`. Idet jeg erkender min fejlbarlighed, kører jeg stavekontrollen `ispell(1)`:

```
datamat > ispell -d dansk -t -C del?.tex
```

Herefter startes den interaktive stavekontrol. Her er en forklaring på den lidt hemmelige kommandolinie:

- d dansk** vælger den danske ordbog. Vi kunne istedet have valgt **english**.
- t** sørger for, at L<sup>A</sup>T<sub>E</sub>X-direktiver ikke bliver opfattet som stavfejl.
- C** gør, at sammensatte ord (Compound words) ikke betragtes som stavfejl.
- del?.tex** vælger alle relevante filer, se afsnit 4.2.4.

Emacs kan stavkontrollere ens tekst med `ispell`, se afsnit 14.1.4.

## 10.5 Referencer

Der findes rigtigt gode, let tilgængelige gratis introduktioner til L<sup>A</sup>T<sub>E</sub>X på Internettet, f.eks. [6] eller [9].

Bruger man L<sup>A</sup>T<sub>E</sub>X meget kan det imidlertid godt betale sig at købe et mere omfattende opslagsværk, f.eks. [5]. Før eller siden køber langt de fleste L<sup>A</sup>T<sub>E</sub>X-brugere [1], det mest omfattende opslagsværk overhovedet; til gengæld forudsætter denne bog et vist forhåndskendskab, så begyndere vil måske finde den uoverskuelig snarere end hjælpsom.

L<sup>A</sup>T<sub>E</sub>X kompenserer for sin manglende brugervenlighed ved at være usædvanlig godt dokumenteret; dette gælder især ekstra pakker. Er man i tvivl om hvad en pakke kan, eller hvordan den aktiveres, kan man konsultere dokumentationen for L<sup>A</sup>T<sub>E</sub>X pakke på

<http://www.ctan.org/tex-archive/help/Catalogue/catalogue.html?action=/index.html>

Her er *alle* L<sup>A</sup>T<sub>E</sub>X-pakker grundigt dokumenteret i dvi-format. Dokumentationsfilerne indeholder typisk både dokumentation af hvordan en pakke bruges og af hvordan den er implementeret; som regel kan man ignorere implementeringsdelen.

Kan man ikke undvære WYSIWYG, findes der et WYSIWYG-tekstbehandlingsprogram, der benytter L<sup>A</sup>T<sub>E</sub>X som underliggende motor. Programmet hedder `lyx(1)`, og er installeret på systemet, men `lyx` er *ikke* understøttet af IT-afdelingen - man bruger det altså på eget ansvar. Læg desuden mærke til, at vi kun *nævner* programmet, vi anbefaler det ikke.

L<sup>A</sup>T<sub>E</sub>X er i virkeligheden ikke et selvstændigt program, men Leslie Lamports makropakke til Donald E. Knuths generiske typesætningssystem T<sub>E</sub>X. T<sub>E</sub>X er beskrevet i [4]<sup>5</sup>. Man kan finde meget mere information om T<sub>E</sub>X og L<sup>A</sup>T<sub>E</sub>X på

<http://www.ctan.org/search>

L<sup>A</sup>T<sub>E</sub>X findes i en udgave til Windows-systemer; denne udgave findes på ovenstående hjemmeside. Langt de fleste UNIX-distributioner kommer med L<sup>A</sup>T<sub>E</sub>X.

---

<sup>5</sup>Kort fortalt blev Knuth tilbage i halvfjerdserne gal over, at hans daværende forelægger ikke satte hans iøvrigt klassiske værk om algoritmer op, og påbegyndte derfor designet af T<sub>E</sub>X. T<sub>E</sub>X bærer kraftigt præg af at være designet i den programmerings barndom, hvilket måske især afspejles i den rædsesfulde syntaks.

# Kapitel 11

## Projekter

Linux er opbygget af en stor mængde småprogrammer, og for at løse større opgaver såsom projekter vil det være nødvendigt at benytte mange programmer og ikke blot et enkelt program eller programpakke for at opnå et optimalt resultat. Ved sjusket brug bliver arbejdsprocessen rodet og besværlig. Alle programmer bruger hvert deres filformat, og det er ikke svært at opnå en situation, hvor intet passer sammen til sidst.

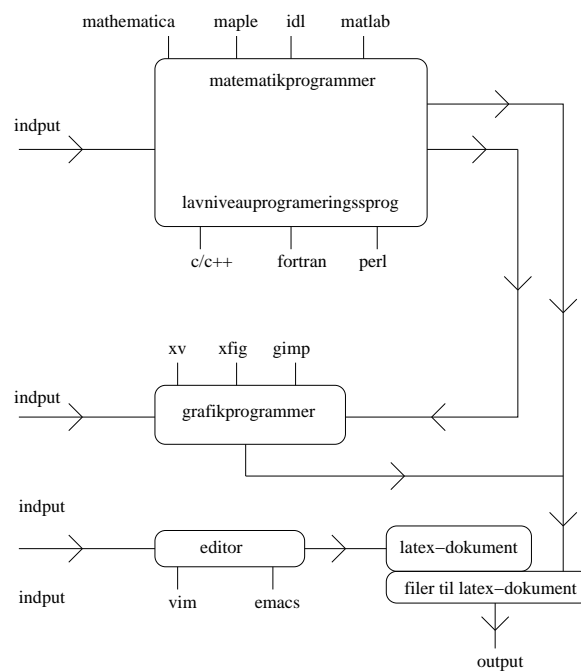
### 11.1 Arbejdsgang

Dette afsnit skal forsøge at give et overblik over samspillet mellem de programmer, der benyttes ved projekter og lignende, således at de muligheder der ligger i at kunne vælge mange programmer til at løse en opgave, kan udnyttes.

#### 11.1.1 Projektmapper

Som det senere vil fremgå af dette kapittel bliver der genereret et enormt antal filer i løbet af et projekt. Derfor anbefales det at oprette nye biblioteker og om nødvendigt underbiblioteker til hver projekt, da det ellers kan blive meget tidskrævende at finde relevante filer og iøvrigt stort set umuligt at rydde op i sit hjemmekatalog. Se afsnit 4.2 om oprettelse af biblioteker og flytning af filer mv.

### 11.1.2 Valg af programmer



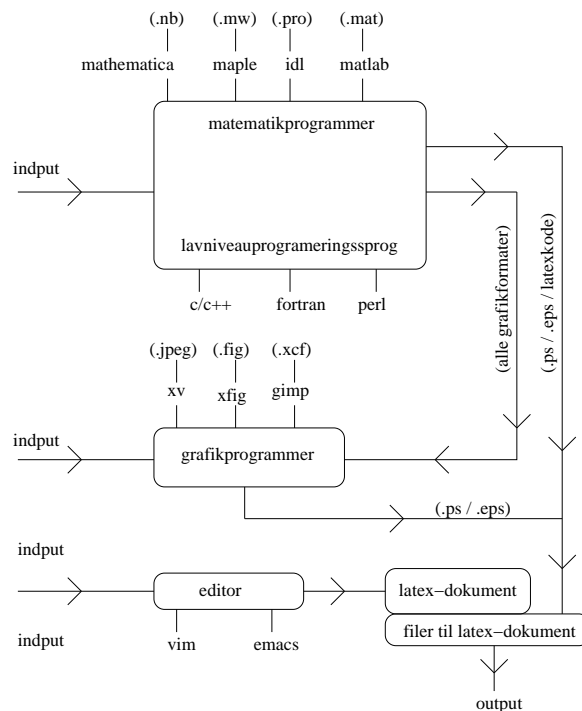
Figur 11.1.2.

Figur 11.1.2 viser hvordan man kommer fra det indput, der er til et projekt, og hen til én endelig fil (Output). Input dækker over at skabe noget nyt enten fra bunden eller ud fra noget givet data-materiale. Hver firkantede kasse dækker over en gruppe programmer, og ud fra kassen er der skrevet de programmer som findes her på ØL's system <sup>1</sup>. Pilene på figur viser den tidlige udvikling. Man bør ikke bevæge sig mod pilenes retning.

<sup>1</sup>Kun de programmer på computersystemet, som er vigtigst i hver katagori er medtaget.



### 11.1.3 Valg af filer



Figur 11.1.3.

Figur 11.1.3 viser et bud på hvor hvilke filformater skal benyttes i arbejdsprocessen. I praksis vil man ikke altid kunne følge figur 11.1.3; men det vil være muligt at benytte det som retningslinie. Figur 11.1.3 skal læses på samme måde som figur 11.1.2, dog er der i parenteser tilføjet filnavne til liniestykkerne. De steder hvor noget skal gemmes eller overføres som tekst-fil, er der ikke skrevet noget filformat.

Mellem programmerne i firkanterne kan der udveksles informationer i et hav af filformater; men når en firkant forlades bør man *exportere*<sup>2</sup> sine filer i de angivet formater. Det er muligt at benytte andre filformater end dem der er angivet; men det giver ingen fordele.

Bemærk at man i mange af programmerne bør gemme sit arbejde i programmets eget filformat. Dette skyldes at der kan gå information tabt ved export til et andet filformat. Fx når der er lavet nogle matematiske beregninger i maple, der ender med en graf. Her vil grafen skulle exporteres som en billedfil for at kunne indsættes i et latex-dokument. Men billedet af grafen indeholder ikke udregningerne, der ligger til grund for billedet, og derfor er det vigtigt at gemme maple-arket også som *Maple Worksheet* .mw-fil.

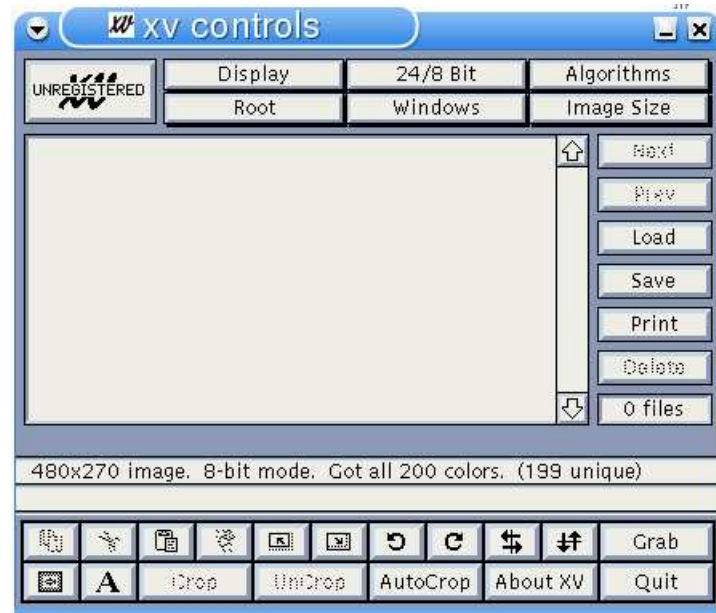
## 11.2 Grafikprogrammer

Det tre mest anvendte grafikprogrammer på ØLs system er beskrevet her. Det skal bemærkes at programmerne kan meget mere end det der beskrives her.

<sup>2</sup>Export af filer betyder at gemme filen under et andet format, hvor muligvis ikke alt informationen fra den oprindelige fil kommer med.

### 11.2.1 xv

xv er et mindre grafikprogram, der egner sig bedst til at tage *Screen Shots* (som fx figur 11.2.1). Efter at have startet programmet xv kommer der et vindue frem, hvor der står xv på. Ved at klikke på dette vindue med højre mussetast kommer det egentlige program frem.



Figur 11.2.1 Her ses programmet xv.

Der kan tages to typer af Screen Shot's.

- *Autograp* Klik på *Grap*-knappen. Herefter vælges et antal sekunder og der klikkes på *Autagrap*-knappen. Dernæst kan man klikke på det vindue man ønsker et billede af<sup>3</sup>. Når tidsintervallet er gået bliver billeder taget og man kan se det i et nyt vindue, der kommer frem.
- *grap* Klik på *Grap*-knappen. Herefter klikkes på *Grap*-knappen. Det første vindue man klikker på herefter bliver der taget et billed af.

Når man ønsker at gemme et Screen Short klikkes der på *Save*-ikonet. Herefter kommer der et vindue frem, hvor der vælges en filplacering og et filformat. Oftest er det bedst at gemme i PostScript-format. Der kommer et vindue frem med mulige indstillinger for PostScript-filen, dette behøver der ikke at blive ændret på før der vælges OK.

### 11.2.2 GIMP

*GIMP* står for *Gnu Image Manipulation Program*. Programmet er godt til at redigere i billeder men ikke til at lave dem fra bunden.

Programmet består af en lille vindue med værktøjer og derudover et vindue til hvert billede der åbnes. For at åbne et billede vælges punktet *open* i menuen *Filer* i værktøjsvinduet. Herfter kan billedet redigeres efter behov. I værktøjsvinduet er der en *Help*-menu med udførlige beskrivelser af hvordan *GIMP* virker.

<sup>3</sup>Hvis der klikkes et bart sted på skrivebordet får man et billede af hele skærmen.


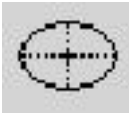


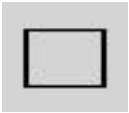
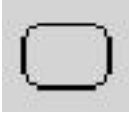

Ved at højreklikke på et af de åbne billeder kommer der en lille menu frem. Via denne er det muligt at vælge de funktioner, der er til rådighed for billedet. Her findes også *Save* og *Save as . . .* under *File*. Når der gemmes kan der altid benyttes xcf-format uden at billedet bliver forringet. Hvis der benyttes andre filformater skal billedet måske exporteres<sup>4</sup>.

### 11.2.3 xfig

*xfig* er et tegneprogram, der er beregnet til tekniske tegninger. Programmet startes med kommandoen *xfig*, der kommer et vindue med en mængde ikoner.

#### Drawing

Her findes værktøjerne til at tegne med. Følgende er en beskrivelse af nogle af dem.






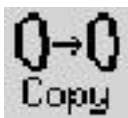
-  Tegner en cirkel med centrum ved der hvor man klikker med musen første gang. Klig med venstre mussetast for at starte og slutte.
-  Tegner en cirkel med periferi der hvor man klikker med musen første gang. Klig med venstre mussetast for at starte og slutte.
-  Tegner lukkede splines. Klik med venstre mussetast for at starte og midtertasten for at slutte.
-  Tegner åbne splines. Klik med venstre mussetast for at starte og midtertasten for at slutte.
-  Tegner firkanter. Klig med venstre mussetast for at starte og slutte.
-  Tegner firkanter med runde hjørner. Klig med venstre mussetast for at starte og slutte.
-  Benyttes til at skrive tekst.

---

<sup>4</sup>Se afsnit 11.1.3.

## Editing

Her findes værktøjerne til at redigere i det eksisterende.

- 
 Laver en samling, således at flere streger kommer til at hænge sammen. Venstre klik på de objekter der skal samles, derefter højreklikkes.
- 
 Bryder en samling. Klik med venstre musseknep på en samling og den går i opløsning.
- 
 Scalering af objekter. Der klikkes med venstre mussetast på et hjørne af et objekt, herefter flyttes musen til størrelsen er rigtig og derefter klikkes der med venstre mussetast igen.
- 
 Flytte et punkt i et objekt. Der klikkes med venstre mussetast på et hjørne af et objekt, herefter flyttes musen til hjørnet er flyttet og derefter klikkes der med venstre mussetast igen.
- 
 Flytte objekter. Der klikkes med venstre mussetast på et hjørne af et objekt, herefter flyttes objektet og derefter klikkes der med venstre mussetast igen.
- 
 Kopiering af objekter. Der klikkes med venstre mussetast på et objekt, hvorefter der laves en kopi. Denne flyttes med musen og lægges ved at klikke med venstre mussetast igen.

For at se mere om hvordan xfig virker se i menuen `Help` øverst til højre i programmet.

## Save

Når man har lavet en figur skal det huskes at gemme den som *.fig*-fil, ellers kan man ikke redigere i billedet længere. Hvis man skal benytte billedet i et andet filformat må eksportere billedet. Hvis det tegnede skal indsættes i latex anbefales det at eksportere til *.eps*-format.

## 11.3 Viewere

Flere af de filformater, der benyttes til dagligt kan ikke redigeres umiddelbart. Det drejer sig om *.ps*, *.pdf* og *.dvi* filer. Disse filer skal altså laves ud fra nogle andre filer, og man kan kun ændre i *ps/pdf/dvi*-filen ved at rette i de filer den er lavet af.

### 11.3.1 dvi-filer & xdvi

*dvi* står for *Device Independent*, og det betyder at indholdet af filen er ens på alle computere. Det lyder måske som noget, der burde blive taget for givet; men fx word-dokumenter behøver *langt fra* at se ens på alle computere.

dvi-filer kan ses i programmet *xdvi*. Den nemmeste metode at åbne en dvi-fil i *xdvi* på er ved i en xterm at skrive `xdvi ~/detbibliotekfilenliggeri/filnavn`. Hvis man fx har en dvi fil, der ligger `~/dokumenter/projekt1/stjerner.dvi` kan man åbne filen i *xdvi* ved kommandoen `xdvi ~/dokumenter/projekt1/stjerner.dvi`.

Man vil dog ofte vælge at konvertere sin dvi-fil over til enten en ps-fil eller en pdf-fil. Kommandoen `dvips indputfil -o outputfil` vil konvertere indputfilen fra dvi-format til ps-format og navngive den nye ps-fil `outputfil`. Kommandoen `dvipdf` virker helt tilsvarende, blot omformes filen til en pdf-fil.

### 11.3.2 ps-filer & gv

*ps* står for *Post Script*. ps-filer vil ikke nødvendigvis se ens ud på alle computere, tilgængeligheden kan de udskrives. Faktisk er ps det format som de fleste programmer i linux sender deres udskrift i til printeren. Så hvis man udskriver til en fil i stedet for til en printer, får man en ps-fil.

Programmet *gv* kan benyttes til at åbne ps-filer. Ved i en xterm at skrive `gv ~/detbibliotekfilenliggeri/filnavn` vil filen blive åbnet i ps. Hvis man fx har en ps fil, der ligger `~/dokumenter/projekt1/stjerner.ps` kan man åbne filen i *gv* ved kommandoen `gv ~/dokumenter/projekt1/stjerner.ps`.

### 11.3.3 pdf-filer & Acrobat Reader

*pdf* står for *Portable Document Format* og er et filformat, der decideret er lavet så det er velegnet til at kunne distribuerer dokumenter i. Filerne ser altså ens ud på alle computere og desuden fylder de noget mindre end ps-filer, hvilket er praktisk når man skal give andre en kopi.

pdf-filer kan åbnes i programmet *Acrobat Reader*. Hvis man fx har en pdf fil, der ligger `~/dokumenter/projekt1/stjerner.pdf` kan man åbne filen i Acrobat Reader ved kommandoen `acroread ~/dokumenter/projekt1/stjerner.pdf`.



# Kapitel 12

## Diverse

Dette kapitel indeholder diverse oplysninger, der er gode at have, men som ikke rigtigt passede ind i nogle af de øvrige kapitler.

### 12.1 Hjælp!

Brugen af EDB-systemer er altid problematisk. Før eller siden får man brug for hjælp til at løse et eller andet problem.

Mange overser den første og ofte mest tilfredsstillende løsning: opsøg dokumentation! Edbafdelingens hjemmeside indeholder dokumentation til dele af systemet, så det er værd at se her efter hjælp ([www.fys.ku.dk/edb](http://www.fys.ku.dk/edb)). UNIX' `man`-sider (se 4.4) er også sædvanligvis storartet dokumentation. GNU projektet har på det seneste valgt at gå væk fra `man`-sider, og benytter i stedet deres eget dokumentationssystem ved navn `info`. Man tilgår `info`-systemet enten ved hjælp af `info`-programmet, eller ved hjælp af GNU Emacs kommandoen `C-h i`.

Hvis man ikke finder svar på sit spørgsmål i dokumentationen, kan man prøve at spørge tilfældige, haj-agtigt udseende personer i terminalrummet. Sædvanligvis finder UNIX-hajer stor tilfredsstillelse i at demonstrere deres store viden; dette kan udnyttes.

Herudover står EDB-afdelingen selvfølgelig til rådighed for spørgsmål omkring systemet og problemer med hardware og programmel, men *ikke* decideret support af brugen af programmer. Edbafdelingen kan kontaktes via email på [fejl@fys.ku.dk](mailto:fejl@fys.ku.dk).

### 12.2 Edbafdelingens hjemmeside

Edbafdelingen udarbejder vejledninger til forskellige dele af systemet. Disse vejledninger, samt diverse web-baserede tjenester, kan forefindes på [www.fys.ku.dk/edb](http://www.fys.ku.dk/edb).

### 12.3 Kort om sikkerhed

EDB-afdelingen gør hvad den kan, for at ondskabsfulde personer ikke skal kunne læse eller ødelægge brugere af systemets data, og for iøvrigt at modvirke fejl & katastrofer der kan medføre tab/

beskadigelse af data. Desværre er implementering af EDB-sikkerhed altid en afvejning imellem hvor sikkert et system skal være, og hvor nemt det skal være at bruge; jo sikrere det er, jo mere irriterende er det også. Derfor er systemet ikke et “sikkert” system.

Hvis du har data, som du ikke ønsker at andre skal læse eller skrive, er det første skridt at sørge for at filrettighederne er sat passende, se afsnit 4.3. Det er imidlertid ikke altid nok; uanset hvad du sætter filrettighederne til, kan EDB-afdelingens medlemmer *altid* læse *alle* dine filer. Ansatte i EDB-afdelingen er selvfølgelig under ansvar: EDB-afdelingen skal have en fornuftig grund til at snuse i folks konti. Hvis du har mistanke om, at nogle har læst eller pillet ved dine private filer, skal du straks kontakte EDB-afdelingen; skriv til [fejl@fys.ku.dk](mailto:fejl@fys.ku.dk).

## 12.4 Brug af bærbar på systemet

Studerende, der vil arbejde på deres egne medbragte labtops, kan enten anvende den trådløse netforbindelse i vandrehallen og ØL på HCØ (se 12.6), eller anvende en af de netkabler, som ligger i terminalrummene og er markeret til brug for labtops - man skal anvende automatisk tildeling af IP-adresse for at komme på nettet. Lad være med at tage kablerne ud af terminalerne og bruge dem - det gør, at terminalen ikke kan bruges.

## 12.5 HJÆLP – jeg kan ikke logge ind!

Det sker fra tid til anden, at en bruger ikke kan logge ind. Det er som regel selvforskyldt, så her kommer nogen af de mest gængse årsager, samt hvordan man løser dem:

- *Der angives forkert login eller password.*  
Løsning: Find det rigtige. I UNIX er login og passwords versalfølsomme (eng. *case-sensitive*). Der er altså forskel på store og små bogstaver. Hvis man stadig ikke kan logge ind, kan man henvende sig på studenterkontoret på 1. sal i D-bygningen.
- *Kontoen er lukket.*  
Dette kan f.eks. skyldes misbrug af systemet, overskridelse af diskkvota i mere end syv dage, eller at man ikke har skiftet sit password da man fik mails om dette. Løsning: Undlad først og fremmest at blive fanget i et af disse tilfælde, men ellers henvend dig til [fejl@fys.ku.dk](mailto:fejl@fys.ku.dk).
- *Kontoens diskkvote er overskredet.*  
Løsning: Log ind vha. tekstbaseret login (se afsnit 1.2), og slet filer nok til at du kommer ned under din quota. Her kan det være en hjælp at læse afsnit 1.6.1 om quota, og desuden kan kommandoen `du -h` være behjælpelig: Viser hvor mange bytes hver katalog/fil i hjemmekataloget fylder. Herefter kan man overveje, hvad man evt. kan slette. En kendt synder er Mozilla, der ofte gemmer kopier af sider man har besøgt. Dette gemmes under kataloget `.mozilla`, som forklaret i 8 Hvis man mener at man har god grund til at have højere diskkvote end de 150 MB som er standard, kan man skrive en mail til [fejl@fys.ku.dk](mailto:fejl@fys.ku.dk) - så ser vi på om det virker relevant nok til, at diskkvoten skal forhøjes

## 12.6 Trådløst netværk på HCØ

I vandrehallen, auditorierne 1,2,3 og 4, dele af E-bygning samt stue, 1. og 3. sal i D-bygningen, er der etableret et fælles trådløst netværk. Dette kan bruges hvis man er bruger på enten matematikernes eller vores system, og for videre reference henviser vi til [wl.hco.ku.dk](http://wl.hco.ku.dk), som forklarer hvordan man kobler sig på netværket



## 12.7 Personoplysninger

En gang imellem har man brug for at finde emailadressen på en personen, personen bag en emailadresse eller bare en almindelig adresse på en person. Dette kan gøres på to forskellige måder, alt efter hvad man præcist har brug for:

**get** Programmet *finger* er i stand til at oplyse navn, userid, og et par andre ting for systemets brugere. Programmet kaldes som:

```
datamat > finger <brugernavn>
```

eller

```
datamat > finger <navn>
```

hvor navn kan være enten fornavn, efternavn eller begge dele (husk citationstegn omkring navne med mellemrum i)

**Interne telefonnumre** Københavns Universitet har et internt telefonsystem. En online telefonbog kan findes på Internetadressen:

```
http://www.ku.dk/vejviser/
```

## 12.8 UNIX på sin egen PC

Har man en PC, kan man installere Linux, en gratis UNIX-klon med kildeteksten offentligt tilgængeligt. Til trods for at Linux er gratis, er det på ingen måde et legetøjssystem; forklaringen på, at nogle forærer højkvalitets-programmel væk, er ideologisk; se evt. en af

```
http://www.opensource.org  
http://www.gnu.org  
http://www.linux.org
```

Ønsker man at forsøge sig med Linux, skal man være forberedt på, at Linux er et UNIX-lignende operativsystem, og som sådan er noget vanskeligere at styre end Microsoft Windows<sup>TM</sup>. Linux kan findes mange steder på Internettet, f.eks.

```
http://www.redhat.com  
http://www.linuxworld.com  
http://www.debian.org
```

Med mindre man har en meget hurtig forbindelse kan det bedre betale sig at købe en Linux-distribution. De fås i Bogladen og i velassorterede boghandler til ganske rimelige priser; en CD burde kunne erhverves for 50–100kr.

## 12.9 UNIX programmer under Microsoft Windows

For dem der ikke umiddelbart har mod på at installere Linux på deres egen PC, men i stedet bruger Microsoft Windows, er der stadig mulighed for at bruge en del af programmerne omtalt i denne bog. Specielt er det muligt at få f.eks. emacs,  $\text{\LaTeX}$  og en ordentlig shell (f.eks. tcsh).

Følgende links kan være af interesse:

- <http://www.gnu.org/software/emacs/windows/ntemacs.html>  
Emacs til Windows.
- [http://kefk.net/Open\\_Source/MiKTeX/](http://kefk.net/Open_Source/MiKTeX/)  
En Windows-implementation af  $\text{\LaTeX}$ .
- <http://sources.redhat.com/cygwin/>  
Her kan man finde Windows-udgaver af en lang række UNIX-programmer.

## **Del II**

# **Avancerede funktioner**



## Kapitel 13

# Kommandofortolkeren

Kommandofortolkeren er et ekstremt avanceret værktøj, faktisk næsten et programmeringssprog, så det vil være umuligt at komme omkring alle detaljer i denne bog. Der er dog nogle ting, som vi skønner er så smarte, at det vi vil gå lidt nærmere ind på dem. Brugere, som måtte ønske endnu mere information vil kunne hente inspiration til videre læsning i afsnit 13.8.

### 13.1 Indtastning af kommandoer

Det er selvfølgelig altid simplere at bruge et grafisk værktøj, som f.eks. Microsofts Explorer, end at bruge en tekstbaseret kommandofortolker. Bruger man grafiske værktøjer behøver man ikke at huske lange kommandoer, eller møjsommeligt sidde og taste lange stinavne ind. Moderne kommandofortolkere gør imidlertid hvad de kan, for at være behagelige at arbejde med, til trods for deres tekstbaserede natur.

#### 13.1.1 Redigering på kommandolinien

Selve kommandolinien opfører sig omtrent som man skulle forvente. Hvis man er vant til et vMicrosoft Windows miljø, vil man måske alligevel blive en smule overrasket; tasterne er de samme som i GNU Emacs (se afsnit 5.3):

←	Flyt markøren til venstre
→	Flyt markøren til højre
<ctrl-a>	Flyt markøren til begyndelsen af linien
<ctrl-e>	Flyt markøren til enden af linien
<backspace>	Slet tegnet på markørens venstre side.
<ctrl-d>	Slet tegnet under markøren.
<ctrl-k>	Slet til linieslut (gem i udklipsbuffer).
<ctrl-y>	Kopier udklipsbuffer til markørposition.

#### 13.1.2 Kommandohistorie

Ofte kommer man i den situation, at man har brug for atter at udføre en kommando, som man udførte for 10 minutter siden. Kommandofortolkeren husker tidligere udførte kommandoer, så

i stedet for møjsommeligt at indtaste kommandoen igen, kan man bladere igennem de tidligere kommandoer med ↑ og ↓. Denne facilitet kaldes historie (eng. *history*).

Ydermere kan man få kommandofortolkeren til at finde den sidst udførte kommando der starter med en bestemt tegnsekvens ved at skrive tegnsekvensen og trykke ⟨esc-p⟩. Så hvis jeg f.eks skriver `lp⟨esc-p⟩`, finder kommandofortolkeren den tidligere kommando

```
lpr -Pm1c gruppearbejde.tex
```

Hvis jeg mente en endnu tidligere kommando kan jeg blot endnu en gang trykke ⟨esc-p⟩, og kommandofortolkeren giver mig nu muligheden `lpracct` og så videre.

### 13.1.3 Automatisk udfyldning

På bare halvstore UNIX-installationer skal er man ofte nødt til at skrive meget lange stinavne, for at nå frem til den fil, man har brug for. F.eks. er den fulde sti for den fil, der ligger bag den tekst du læser netop nu

```
/amd/kirstine/00/treue/kursusbog2/shells/main.tex
```

hvad der i længden bliver temmelig træls at skrive. Heldigvis implementerer kommandofortolkeren automatisk udfyldning (eng. *completion*), der betyder, at man nøjes med at indtaste en del af en kommando eller et filnavn, hvorefter man ved at taste ⟨tab⟩ beder kommandofortolkeren gøre det færdigt. Eksempelvis vil `ls /usr/loc⟨tab⟩` blive udfyldt til `ls /usr/local/`.

Ofte er der selvsagt flere muligheder for udfyldning. I så fald vil kommandofortolkeren fylde så mange tegn ind som muligt og signalere ved et bip, at der fandtes flere muligheder. Man kan få en liste over alle muligheder ved at taste ⟨ctrl-d⟩. Kommandofortolkeren genskriver kommandolinien bagefter. Hvis jeg f.eks. skriver

```
datamat > cd /amd/kirstine/00/kursusbog2/⟨ctrl-d⟩
OEL/          bibliography.bib  master.tex~      shell/
Makefile      front/           preamble.tex      unix/
OLDKB2/       latex/          preamble.tex~
back/         master.tex       print/
```

Automatisk udfyldning virker som nævnt ikke kun på filnavne men også på kommandoer:

```
datamat > lp⟨ctrl-d⟩
lp      lpq      lpr      lprm      lpstat
```

### 13.1.4 Alias

Et alias er en forkortelse for en kommando. Hvis man f. eks. tit anvender kommandoen `ls -a`, kan man oprette aliaset `la` for den. Herefter vil kommandoen `la katalog` blive opfattet som `ls -a katalog`. Aliaset oprettes med kommandoen `alias la ls -a`. Alias, der er oprettet på kommandolinien, gemmes ikke når der logges ud. Vil man have en varig oprettelse af et alias, skal oprettelsen af aliaset indsættes i konfigurationsfilen `.alias`, der er placeret i ens hjemmekatalog.

Ovenstående eksempel var simpelt, fordi argumentet `katalog` blot skulle indsættes til sidst i kommandoen. Man kan sagtens forestille sig, at det er ønskeligt at få placeret argumenterne et andet sted. Her anvendes tegnfølgen `!*` som forkortelse for “argumenterne, der efterfølger navnet på aliaset”. Antag, at aliaset `find` er defineret ved

```
alias find 'grep !* | less'
```

Kommandoen `find hej *` bliver da fortolket som

```
grep hej * | less
```

Man kan se sine alias ved at give kommandoen `alias` uden argumenter og fjerne aliaser med `unalias`.

## 13.2 Ind- og uddata til programmer

Et UNIX-program, der startes fra kommandofortolkeren, har to muligheder for at kommunikere med brugeren: Enten kan det lave sig et nyt vindue, eller det kan overtage kommandofortolkerens. De fleste små programmer til filmanipulering, f.eks. `ls(1)` og `mv(1)`, overtager kommandofortolkerens vindue. Når man udfører programmet `ls`, så standser kommandofortolkeren midlertidigt, idet den overlader vinduet til programmet `ls`. Når `ls` er færdig, dvs. når den har vist en liste af filer, så kører kommandofortolkeren videre i vinduet.

Programmer som `ls` der “overtager vinduet” kender i virkeligheden slet ikke til vinduer. De har blot en “kanal”, som de forventer at kunne skrive uddata på. Kommandofortolkeren sørger for at koble denne kanal til dens eget vindue. Denne kanal kaldes `stdout` (**standard output**).

Når man starter et program, kan man bede kommandofortolkeren om at binde programmets `stdout` til en fil istedet for kommandofortolkerens vindue; dette kaldes på engelsk *redirection*. Man gør dette ved at skrive `> filnavn` efter kommandoen. Så hvis jeg gerne vil have en detaljeret oversigt over hvilke filer, der ligger i mit hjemmekatalog, og jeg gerne vil have oversigten placeret i filen `mine-filer.txt`, så bruger jeg kommandoen

```
ls -l > mine-filer.txt
```

Bemærk, at hvis filen `mine-filer.txt` allerede eksisterer, vil ovenstående kommando overskrive den. Man kan også tilføje til enden af en fil istedet. Det gør man med to større-end tegn:

```
ls -l >> mine-filer.txt
```

Programmer har også en kanal til inddata, som kommandofortolkeren almindeligvis binder til dens eget vindue. På den kommer tastetryk igennem vinduet og ind til programmet. F.eks. kan man tvinge `rm(1)` til at spørge om lov, så man ikke ved et uheld kommer til at slette vigtige filer, ved at give den argumentet `-i` (for interactive):

```
datamat > rm -i frygtelig-vigtig-fil.tex
rm: remove frygtelig-vigtig-fil.tex: ? (y/n)
```

Denne “indkanal” kaldes `stdin` (**standard input**), og også denne kan bindes om. De fleste programmer kan ikke se forskel på, om de får inddata fra tastaturet eller fra en fil. Man binder `stdin` om ved at skrive `< filnavn` efter kommandoen. Man kan på engang bruge både `<` og `>`, hvis man vil omdirigere både `stdin` og `stdout`:

```
datamat > kommando < inddata > uddata
```

Programmer har også en tredje ud-kanal, `stderr` (**standard error**), til hvilken de skriver egentlige fejlmeddelelser. Denne kan omdirigeres med `2>`. Man kan koble `stderr` til `stdout`, og sende begge dele ud i en fil med `>&`.

## 13.3 Jobstyring

Et kørende program kaldes i UNIX en proces. Et UNIX system *multitasker*, dvs. kører flere processer på en gang. I realiteten kan en datamat med kun een processor ikke gøre mere end een ting ad gangen, men ved at skifte imellem de aktive processer f.eks. hvert millisekund giver datamaten indtryk af at gøre flere ting på en gang.

Når man instruerer kommandofortolkeren om at udføre en kommando, stopper kommandofortolkeren midlertidigt, og en ny proces startes til kommandoen. Denne nye proces overtager `stdin`, `stdout` og `stderr` fra kommandofortolkeren. Når kommandoen terminerer, genstartes kommandofortolkeren, der samtidigt får sine kanaler igen.

### 13.3.1 Processer i baggrunden

Selvom man har omdirigeret alle en proces' kanaler, venter kommandofortolkeren alligevel. Til tider er man ikke interesseret i at vente, f.eks. hvis det program, man har sat i gang skal stå og regne i lang tid, eller hvis programmet ikke benytter nogle af de tre kanaler, men istedet laver sit eget vindue. En proces der startes, uden at kommandofortolkeren venter på den, siges at køre i baggrunden. En proces startes i baggrunden ved at tilføje `&` i slutningen af kommandolinien. Lad os f.eks. starte kommandoen `xdvi(1)` (mere om den i afsnit 11.3.1):

```
datamat > xdvi &
[3] 28462
datamat >
```

En proces har et såkaldt proces-id, et tal der identificerer processen. Dette tal er i ovenstående eksempel 28462. For nemhedens skyld nummererer kommandofortolkeren de processer den har startet fortløbende fra 1 — i ovenstående eksempel er den nået til nummer 3.

Når en baggrundsproces er kørt færdig (f.eks. fordi man har lukket kommandoens vindue), gør kommandofortolkeren opmærksom på det:

```
[3] + done      xdvi
```

### 13.3.2 Processer i forgrunden

Har man startet et program i forgrunden, dvs. uden at sætte det i baggrunden med `&`, kan man suspendere programmet midlertidigt og vende tilbage til kommandofortolkeren ved at trykke `<ctrl-z>`:

```
datamat > xdvi
<ctrl-z>
Suspended
datamat >
```

Har man fortrudt kan man også standse programmet helt med `<ctrl-c>`:

```
datamat > mosml -quietdec < /usr/lib/words > words.encrypted
<ctrl-c>
datamat >
```



### 13.3.3 Jobkontrol

Kommandoen `jobs` viser en liste af programmer, kommandofortolkerne har startet for en. For eksempel:

```
datamat > xdvi &
[1] 28710
datamat > mozilla
(ctrl-z)
Suspended
datamat > jobs
[1] - Running      xdvi
[2] + Suspended   mozilla
datamat >
```

Job nummer 1, `xdvi(1)`, kører i baggrunden (derfor ampersanden), mens job nummer 2, `mozilla`, er suspenderet (eng. *suspended*), dvs. har fået frataget sine kanaler, og er sat til at vente.

Hvis en proces er suspenderet, kan programudførelsen genoptages enten i forgrunden (med kommandoen `fg`, *foreground*) eller i baggrunden (med kommandoen `bg`, *background*). Ud for hver kommando viser `jobs` et jobnummer, der med tegnet `%` som prefix, kan bruges som argument til `fg`, `bg` og til kommandoen `kill(1)`, som slår processen ihjel. Processer har mulighed for til en hvis grad at vægre sig ved at blive slået ihjel. Man kan insistere ved at give `kill` parameteren `-9`. Ville jeg standse `xdvi`-processen kunne jeg gøre dette ved

```
datamat > kill -9 %1
```

## 13.4 Søgning

I det følgende præsenteres kort nogle af de vigtigste værktøjer, som UNIX stiller til rådighed i forbindelse med søgning efter eller i filer. Værktøjerne gennemgås ikke i detaljer; dette afsnit tjener mest til at gøre opmærksom på deres eksistens. Vi håber at læseren ved senere lejligheder vil tænke f.eks. "ah, det kunne man da vist med dette-eller-hint værktøj" og så selv ved hjælp af man-systemet vil kunne hitte en løsning.

### 13.4.1 Find

Især hvis man programmerer, har man fra tid til anden brug for at finde filer med bestemte karakteristika, f.eks. at de har samme efternavn. Ligger filerne i samme katalog kan man bruge mønstre. Er de derimod spredt over flere kataloger tyr man sædvanligvis til værktøjet `find(1)`:

```
find <start> [betingelser]
```

Søgningen starter i kataloget `<start>`, og går rekursivt ned igennem alle underkataloger. Hvis der ikke angives nogle `[betingelser]`, skriver `find` på `stdout` alle de filnavne den møder. Angivelsen af `[betingelser]` er en lille videnskab i sig selv, så vi nøjes her med at give et enkelt eksempel; interesserede henvises til man-siden. Ønsker jeg at finde alle filer med efternavn `.tex` i det aktive katalog og dets underkataloger skriver jeg blot:

```

datamat > find . -name \*.tex
./dat/BilagC.tex
./dat/BilagF.tex
./dat/afproevinddata.tex
./dat/afproevning.tex
./dat/afproevtest.tex
[...]

```

Der *skal* være en skråstreg før `*.tex`, ellers vil kommandofortolkeren opfatte asterisken som et mønster den skal lave om til et antal filnavne og give til `find`; her er vi imidlertid interesserede i at give asterisken selv som parameter.

### 13.4.2 Grep

Tilsvarende kan man nogle gange have brug for at søge efter en bestemt tekststreng *inden i* en eller flere filer. Lad os f.eks. tænke os, at jeg var i færd med at skrive en rapport. Lidt større rapporter fordeler man gerne på flere forskellige filer, så lad os sige, at mit program er spredt over en hel masse forskellige filer, der alle sammen hedder `.tex` til efternavn:

```

datamat > ls *.tex
BilagA.tex      afproevning.tex      problemorienteretanalyse.tex
BilagB.tex      afproevtest.tex      programbeskrivelse.tex
BilagC.tex      brugervejledning.tex programmeringsovervejelser.tex
BilagF.tex      litteraturliste.tex  rapport.tex
afproevinddata.tex objektorienteretdesign.tex sammenfatning.tex

```

Hvis jeg nu vil finde den fil, der indeholder teksten `Indsamlet data`, kan jeg bruge programmet `grep(1)`:

```

datamat > grep "Indsamlet data" *.tex
BilagA.tex:      Indsamlet data er præsenteret her:
datamat >

```

Hvis jeg vil have linienummeret med, så tilføjer jeg blot `-n`:

```

datamat > grep -n "Indsamlet data" *.tex
BilagA.tex:10:      Indsamlet data er præsenteret her:
datamat >

```

Værktøjet `grep` har generelt formatet

```
grep <regexp> [filnavne]
```

hvor `<regexp>` er et såkaldt regulært udtryk (eng. *regular expression*). Alle strenge<sup>1</sup> er regulære udtryk, så i eksemplet ovenfor finder `grep` alle forekomster af strengen `"Indsamlet data "` i alle filer, hvis efternavn er `.tex`. Regulære udtryk kan imidlertid meget, meget mere end blot at finde faste strenge. Interesserede henvises til man-siden for `grep`, Emacs' online hjælp til regulære udtryk eller [8].

---

<sup>1</sup>for de teoretisk inklinerede: alle *endelige* strenge.

## 13.5 Pipes

Ligesom vi kan binde en proces' `stdin` til en fil, kan vi også binde den til en anden proces' `stdout`. Man hælder så at sige uddata fra et program ind i et andet. Man foretager bindingen ved at skrive to kommandoer adskilt af en lodret streg '|'.

Med pipes kan man, ved at kombinere programmer der hver især gør meget simple ting, udføre komplicerede kommandoer. Dette er en af de absolut allerstærkeste *features* ved kommandofortolkeren og er i allerhøjeste grad i tråd med UNIX-filosofien, så vi demonstrerer med et antal eksempler.

Pipes i sig selv er selvsagt ikke rigtigt interessante før man sætter dem imellem programmer, så vi introducerer nogle kommandoer undervejs. Alle de følgende eksempler har jeg gjort brug af; visse bruger jeg dagligt. Der er altså på ingen måde tale om konstruerede eller "tænkte" eksempler.

### 13.5.1 Eksempler på pipes

Vi kaster os hovedkuls ud i det:

```
datamat > grep kommandofortolker *.tex | wc -l
38
```

Kommandoen `wc(1)` læser fra `stdin`, indtil der ikke er mere at læse, og udskriver (givet parameteren `-l`) antallet af læste linier på `stdout`. Den første del af kommandoen, `grep ...`, udskriver for hver fil hvis navn slutter med `.tex` alle de linier i filen, der indeholder strengen "kommandofortolker". Så ovenstående tæller (ca.) hvor mange gange ordet kommandofortolker optræder i kildeteksten til dette kapitel.

Måske jeg istedet har lyst til at finde ud af, ca. hvor mange linier kursusbog vi har skrevet ialt, ikke kun dette kapitel. Hvert kapitel ligger i et katalog for sig, så jeg samler sammen med `find`:

```
datamat > find . -name \*.tex | xargs wc -l
3254
```

Den middel-obskure `xargs(1)` kommando tager en anden kommando, i dette tilfælde `wc`, som parameter, og giver alt hvad den læser på `stdin` som parametre til denne anden kommando. Så `wc` får altså alle filer der slutter på `.tex` som kommandolinie-parameter.

Vi tager et `xargs` eksempel til. Lad os tænke os tilbage til `grep`-eksemplet i afsnit 13.4.2, men lad os denne gang antage, at alle `.tex` filerne ligger i *forskellige* underkataloger af det aktive katalog. Vi kan finde alle `tex`-filerne med `find`, og herefter bruge `xargs` til at give `grep` disse filer som kommandolinie-parametre:

```
datamat > find . -name \*.tex | xargs grep -n "Indsamlet data"
```

Til det næste eksempel benytter vi programmet `su(1)`, der skifter brugerid. Man skriver simpelthen

```
datamat > su treue
password:
```

Hvis man indtaster det rigtige password, ville man ved ovenstående “blive til treue”, dvs. få treue’ rettigheder mht. filer osv.

Hvis man blot ønsker at udføre en enkelt kommando under en påtaget identitet, kan gøre dette med argumentet `-c <kommando>`. Ønsker man ved samme lejlighed at skifte til den nye brugers hjemmekatalog og få den nye brugers opsætning, tilføjer man argumentet `-`. Vi kan udnytte dette til at kopiere en fil fra en konto til en anden uden at behøve at fedte med filrettigheder. Antag, at vi ønsker at kopiere filen `top-hemmelig-fil` fra brugeren `treues` hjemmekatalog til det aktive katalog:

```
datamat > su - treue -c "cat top-hemmelig-fil" > top-hemmelig-fil
password:
```

Vi tager et sidste eksempel: Lad os tænke os, at jeg har skrevet et program, hvis afviklingshastighed på afgørende vis afhænger af størrelsen konstanten `block`. Jeg ønsker eksperimentelt at bestemme, hvor stor denne blok skal være, for at programmet kører hurtigst muligt, så jeg skriver programmet en lille smule om, så det prøver alle blokstørrelser fra 2 til 691. Programmet giver uddata på følgende format:

```
(seed is 677938, blocksize 2)
block=2 scramble      60.02
block=2 accumulate    0.87
block=2 2000000 elements, 23084460 bytes, 11.5422 bytes/element
block=2 (340 bytes reserved but unused)
(seed is 10814, blocksize 3)
block=3 scramble      51.6
block=3 accumulate    0.68
block=3 2000000 elements, 19626624 bytes, 9.81331 bytes/element
block=3 (384 bytes reserved but unused)
[...]
```

Lad os finde den blokstørrelse, der giver den bedste `scramble`-tid. Hertil benytter vi programmet `sort(1)` der sorterer inddata. Først piller vi med `grep` de linier ud, der indeholder `scramble`-tiden. Herefter sorterer vi dem, og tilsidst udskriver de 5 første (dette sidste er funktionaliteten af `head(1)`):

```
datamat > grep scramble times.dat | sort -k3 | head -5
block=22      scramble    36.89
block=26      scramble    36.89
block=20      scramble    36.93
block=30      scramble    36.94
block=28      scramble    37.02
datamat >
```

Parameteren til `sort` vælger den tredje kolonne; interesserede henvises (atter!) til `man`-siden.

## 13.6 Kommandofiler

Kommandoer kan samles i en fil og udføres samlet med kommandoen

```
source filnavn
```

En kommandofil kaldes gerne et *script*. Kommandofortolkeren har faktisk et helt lille programmeringssprog indbygget, der gør det muligt at få selv meget avancerede operationer foretaget automatisk. Kommandofortolkerens indbyggede sprog ligger dog noget udover denne bogs mål. Interesserede henvises til *man*-siderne for kommandofortolkeren, `tcsh(1)`.

## 13.7 Konfiguration

Kommandofortolkeren har variable. En variabel er en tekststreng, der har tilknyttet en værdi. Man kan inspicere variables værdier med kommandoen `echo`, og man kan tildele en værdi til en variabel med kommandoen `setenv`:

```
datamat > echo $FOO
FOO: undefined variable
datamat > setenv FOO foo
datamat > echo $FOO
foo
```

Kommandofortolkeren går ud fra, at strenge der starter med et '\$'-tegn er variable.

Variable bruges typisk til at konfigurere kommandofortolkeren eller programmer man kører, eller til at indeholde basal brugerinformation. F.eks. indeholder variabelen `$USER` navnet på den nuværende bruger, og `$HOME` stien til ens hjemmekatalog:

```
datamat > echo $USER
treue
datamat > echo $HOME
/b/00/treue
```

Enkelte variable har særlig betydning for kommandofortolkeren. Variabelen `$PATH` indeholder f.eks. stien til alle de steder, kommandofortolkeren skal lede efter programmer, mens variabelen `$MANPATH` indeholder stien til systemets *man*-sider.

Alle nødvendige variable sættes automatisk op, når man logger ind. Finder man det nødvendigt at ændre nogle af standardopsætningerne, kan man ændre i filen `.tcshrc`. Når kommandofortolkeren starter, læser den denne fil, og udfører alt hvad der står i den som kommandoer. Man skal være lidt forsigtig med at pille i `.tcshrc`, for piller man de forkerte steder, kan man pludselig komme i den situation, at *intet* virker. Hvis man får pillet så meget, at man slet ikke kan logge ind, kan man henvende sig til administrationen på [fejl@fys.ku.dk](mailto:fejl@fys.ku.dk) - så kan resette `.tcshrc`-filen.

## 13.8 Referencer

Kommandofortolkeren `tcsh(1)` er blot en blandt mange forskellige. Den udspringer af `csh(1)`, der igen udspringer `sh(1)`, den "oprindelige" kommandofortolker. Vil man vide mere om kommandofortolkere, er *man*-siderne for disse et godt sted at starte.

Ønsker man istedet en egentlig bog om at benytte UNIX, er [8] en omfattende og nogenlunde tilgængelig gennemgang af de mange, mange muligheder der er i UNIX.

På Internettet findes talløse UNIX-introduktioner. Et godt sted at starte er <http://www.geek-girl.com>.

På systemet er alle de mest almindelige kommandofortolkere installeret, og brugerne kan (hvis de er tilstrækkeligt "eventyrlystne") selv skifte deres kommandofortolker vha. en af systemets administrations sider på Internettet- se afsnit 8.1.2.



## Kapitel 14

# Tekstredigeringsværktøjer

I dette kapittel kommer vi ind på visse af de mere avancerede dele af emacs, som uden tvivl er den editor, som kan mest - eller som fanatiske vim-brugere siger det: emacs er et udemærket operativsystem - blot mangler den en ordentlig editor. Dette gør desværre visse af dens mere avancerede facilliteter temmeligt besværlige at bruge, hvorfor vi vil forklare de mest brugte i dette kapittel.

### 14.1 Avanceret brug af Emacs

#### 14.1.1 Tilpasning af Emacs

Emacs tilpasses gennem filen `.emacs`, der som standard ligger i hjemmekataloget. Herefter er det blot om at dykke ned i emacs-dokumentationen, spørge på nyhedsgrupper eller læse nogen bøger omkring alle de ting (og vi mener *alle*), man kan ændre ved emacs (se afsnit 14.2).

#### 14.1.2 Smarte redigeringsfaciliteter

I dette afsnit vil vi blot hurtigt nævne nogle af de mere smarte redigeringsfaciliteter.

- Tastekombinationen `M-/` prøver at matche det ord, som cursoren står ved, med alle muligheder i alle åbne buffere. Dette kan i praksis bruges, hvis man har enormt lange variabelnavne. Eksempel: I et C-program bruges variabelen `pokkerstillangtnavnforenenkeltvariabel`, hvilket vil være ret trist at skulle skrive hver gang, udover at det sandsynligvis vil blive forkert skrevet et antal gange. I stedet skriver man blot de første par bogstaver `pokk` og bruger `M-/` til at finde ord der passer (man kan blive ved med at taste, indtil der kommer det rigtige ord).
- I de fleste major-modes kan man udkommentere en region med tastekombinationen `C-c ;`.
- I stedet for kun at klippe hele linjer, kan det også gøres med *rektangler* – dette kan være smart, hvis f.eks. man vil fjerne de første 3 kolonner fra et antal linjer. For at bruge rektangler, sættes mærket som sædvanligt med `C-space` og point placeres der hvor rektanglet skal udspændes. Herefter bruges tastekombinationen: `C-x r k` (rectangle kill). Hvis man vil indsætte teksten bruges `C-x r y` (rectangle yank).

- Emacs kan sættes til at huske hvilke buffere der var åbne, da man afsluttede det sidst – på en måde en slags “skrivebord”, hvor man ikke rydder papirene op inden man går hjem. Dette konfigureres ved at placere følgende i `.emacs`:

```
;; Enable save desktop
(load "desktop")
(desktop-load-default)
(desktop-read)
```

Derudover skal man oprette filen `.emacs.desktop` (dette gøres med kommandoen `touch(1)`).

- Hvis man arbejder med mange buffere, kan det være værd at prøve `iswitchb`, der gør buffer-skift mere komfortable. Dette kan gøres med følgende i `.emacs`:

```
(require 'iswitchb)
(iswitchb-default-keybindings)
```

### 14.1.3 $\LaTeX$ -mode

Kapitel 10 omhandler  $\LaTeX$ , som emacs har en fortræffelig major-mode til kaldet `AUC-TeX`<sup>1</sup>. Det anbefales at man først læser kapitel 10 og først derefter vender tilbage hertil og ser, hvor meget emacs kan hjælpe en med rapportskrivningen osv.

#### Almindelige tastekombinationer

Som beskrevet i kapitel 10 har  $\LaTeX$  en række kommandoer til opstilling af lister, centrering osv. Disse kan `AUC-tex` indsætte for en, så man ikke behøver skrive en masse hele tiden. Tast `C-c C-e` og skriv i minibufferen hvad du vil indsætte (`<tab>` virker også her). Feks. vil `C-c C-e itemize` indsætte følgende:

```
\begin{itemize}
\item
\end{itemize}
```

`M-ret` indsætter et nyt `\item`. Tastekombinationen `C-c C-s` indsætter en ny `section` (vælg niveau i minibufferen).

Hvis man retter meget i et afsnit og formateringen bliver rodet, kan `M-q` rette op på dette ved at lave korrekt indrykning af det pågældende afsnit (tekst mellem to tomme linjer).

Generelt kan man se i menupunktet `(LaTeX)` hvilke tastekombinationer der er tilgængelige.

#### Sådan oversætter man sit dokument

I stedet for at skifte mellem emacs og et `xterm`-vindue for manuelt at afvikle `latex`-kommandoen, kan man taste `C-c C-c`. Hvis ikke filen er gemt spørges om dette, eller spørges om hvilket program, der skal bruges til at oversætte filen (standard er `latex`, så bare tast `enter` (hvis man vil se, hvad  $\LaTeX$  skriver mens den oversætter, kan man herefter taste `C-c C-l`). Hvis ens dokument er `up-to-date` vil default være `xdvi`, der bruges til at se resultatet med. Så man taster altså typisk `C-c C-c enter` (vent) `C-c C-c enter` for at se sit dokument<sup>2</sup>.

<sup>1</sup>Stammer fra Aalborg Universitets Center.

<sup>2</sup>Hvis man splitter dit dokument op i flere filer, er det en god ide at konsultere `AUC`-dokumentationen (se afsnit 14.2).



## Fejlhåndtering

Hvis der er fejl i ens dokument, vil man få dette at vide i minibufferen med en linje á la:

```
LaTeX errors in: <file> Use C-c ' to display
```

Herefter kan man så bruge C-c ' til at lade emacs hoppe til det sted (til den fil), hvor der var en fejl. Man kan gentage tastesekvensen, for at hoppe til næste fejl<sup>3</sup>.

### 14.1.4 I-spell

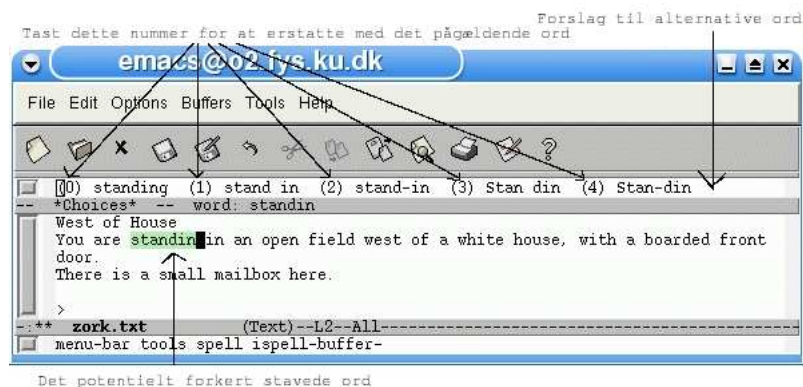
Når man benytter emacs og L<sup>A</sup>T<sub>E</sub>X til rapportskrivning er det ønskværdigt (og anbefalelsesværdigt), at benytte en stavekontrol. Til UNIX findes programmet `ispell`, som er emacs naturligvis integrerer.

Først vælges hvilket sprog ordbogen skal have fra menuen `<tools→spell checking>`. Herefter kan man stavekontrollere ved f.eks. `<tools→spell checking→spell check buffer>`.

Stavekontrol med `ispell` i emacs foregår på følgende måde:

- Ispell finder et ord, som den mener er stavet forkert. Dette markeres med grønt.
- Hvis ikke ordet er forkert, tastes blot `space` for at gå videre til næste ord.
- Hvis Ispell kan finde et potentielt forkert stavet ord, foreslår den alternativer i en linje (på samme måde som minibufferen) i toppen af bufferen – hvert ord har et nummer, som man taster for at skifte ordet ud.
- Hvis Ispell finder et ord, og det er forkert, men ingen af dens forslag i toppen af bufferen er korrekte, kan man selv redigere ordet – dette gøres ved at taste `'r`.
- I minibufferen skriver Ispell desuden, hvilke tastetryk der er tilgængelige (prøv ?).
- Ispell kan stoppes når som helst med C-g.

En typisk session med Ispell ser således ud:



<sup>3</sup>Her er det forøvrigt rart at vide, at C-x 1 maksimerer den buffer man står i, så man efter at have rettet fejl, kan slippe for bufferen med fejlbeskeder).

## 14.2 Referencer

vims officielle hjemmeside findes på

`http://www.vim.org`

Den oprindelige far til emacs – Richard Stallman — har skrevet en manual, der kan findes på:

`http://www.gnu.org/manual/emacs/index.html`

Bilag C indeholder en kort oversigt over de mest benyttede tastekombinationer.

Der er for nyligt skrevet en glimrende introduktionsbog til Emacs (endda af en dansker). Denne indeholder en masse gode tips til at komme igang, samt mere avancerede aspekter (Ispell,  $\LaTeX$ , programmeringssprog osv) – se [7].

På Aalborg Universitets Center kan man finde dokumentationen til AUC-TeX:

`http://sunsite.auc.dk/auctex/doc/`

## Kapitel 15

# Versionskontrol

### 15.1 Introduktion til versionskontrol

I forbindelse med større projekter opstår der som regel en række problemer med vedligeholdelse af filer:

**Editering af samme fil:** 2 eller flere brugere kan ikke editere i den samme fil. Det er den bruger der sidst gemmer filen, hvis ændringer der er aktuelle. De andre brugeres ændringer går tabt.

**Sletning af forkert indhold i fil:** Når der udvikles programmel gennemgår dette som regel en række rettelser. Ved et uheld kan væsentlige dele af en fil slettes, fordi det skønnes at være unødvendigt. Dispositionen kan dog vise sig at være forkert med det resultat, at væsentlige data er gået tabt. I nogle tilfælde har man endda slettet hele filen ved et uheld.

**Sammenlægning af forskellige opdateringer til en fil:** Når to personer laver forskellige rettelser til en fil, er man nødt til at sammenkøre disse rettelser. Denne affære er temmelig tidskrævende og man kan hurtigt overse vigtige dele i en sådan proces.

En del af disse problemer kan afhjælpes manuelt. Man kan eksempelvis aftale hvem der editere i hvad og hvornår. På denne måde undgår man det første problem. Man kan også foretage sikkerhedskopier med jævne mellemrum, så man begrænser tabet ved sletning af data.

Versionskontrol er en automatisering af ovenstående, således at opgaven flyttes fra brugerne til systemet. Dette er selvsagt sikrere og en værdifuld hjælp under rapportskrivning. Vi vil i det følgende beskrive versionkontrollsystemet CVS kort og fortælle hvorledes det anvendes.

Versionskontrol er ikke et krav, når man skriver artikler, rapporter mv. — Men det kan være en stor hjælp. Jo større det man skriver er, jo mere hjælper versionskontrol. Det anbefales, at man tester CVS før man anvender det i forbindelse med en rapport.

### 15.2 CVS, en introduktion

En løsning på ovenstående problem er at lade hver enkelt person arbejde på sin egen lokale kopi af projektet. Det er netop den løsning der er valgt i CVS. Da der eksisterer en individuel lokal kopi, kan ens arbejde ikke interferere med andres.

CVS arbejder med en hovedkopi, der kaldes repository. I repository opbevares samtlige ændringer, der er foretaget på projektets filer. Hver enkelt ændring har sit eget unikke versionsnummer, så det er muligt at skelne ændringer fra hinanden.

Personer arbejder med projektet ved at udtrække lokale kopier fra repository, rette i dem og føre deres ændringer tilbage til repository. Hver ændring, der føres tilbage, tildeles et versionsnummer med en tilhørende ændringsbeskrivelse (i en log).

Man kan på ethvert tidspunkt synkronisere sin lokale kopi med indholdet i repository, så man har de seneste ændringer. CVS er med andre ord en “klient-server” model, hvor repository er “serveren” og den lokale kopi er “klienten”.

I det følgende gennemgås de grundliggende CVS kommandoer. Vi vil ikke gå i dybden med værktøjet, da det er temmelig omfattende. I stedet vil vi begrænse os til at forklare hvorledes CVS kan anvendes på systemet.

## 15.3 Grundliggende CVS

### 15.3.1 Oprettelse af et repository

Skal ens projekt styres af CVS, skal man have et repository til projektet. Repositories oprettes i sit eget katalog med kommandoen:

```
cvs -d <repositorykatalog> init
```

Det antages at repositorykataloget allerede eksisterer.

**Eksempel:** Vi vil oprette et repository til vores projekt “Nano42”. Man kan gøre dette ved først at oprette et katalog, og dernæst oprette et repository i dette katalog:

```
mkdir $HOME/Nano42-repository  
cvs -d $HOME/Nano42-repository init
```

Vores repositorykatalog er nu initialiseret som repository. Husk på at miljøvariablen \$HOME altid peger på ens hjemmekatalog.

### 15.3.2 Udtræk fra repository

Udtræk af kopier fra repository foretages med kommandoen `checkout`. Syntaksen er:

```
cvs [-d repositorykatalog] checkout <sti>
```

I ovenstående er *repositorykatalog* placeringen af repository. *sti* er en sti relativt til repositorykataloget. Man får udtrukket alle underkataloger fra og med *sti*. Derfor udtrækker “.” alt fra repository.

**Eksempel:** Vi vil udtrække en kopi af vores “Dat42G” projekt. Først opretter vi et katalog til kopien og dernæst udtrækker vi den til kataloget:

```
mkdir $HOME/Nano42
cd $HOME/Nano42
cvs -d $HOME/Nano42-repository checkout .
```

Man vil se, at der udtrækkes et katalog ved navn *CVSROOT*. *CVSROOT* er kontrolkataloget for et repository. Vi vil ikke komme nærmere ind på dette katalog, men blot bemærke at man kan konfigurere sit repository ved at ændre på filerne i kontrolkataloget.

Hvert enkelt katalog i kopien indeholder et katalog ved navn *CVS*. Dette katalog benyttes af CVS-systemet til at styre kopien med. Når man en gang har lavet udtræk af en kopi, gemmes lokaliteten af repository i den udtrukne kopi. Det betyder, at man ikke længere behøver at angive et repositorykatalog når man arbejder med kopien.

### 15.3.3 Tilføjelse af filer til projektet

En fil kan tilføjes til projektet, hvis filen er placeret i ens lokale kopi. Kommandoen er:

```
cvs add ['filnavne eller kataloger'...]
```

Det er muligt at specificere flere filer på samme linie. Som beskrevet i 15.3.2 er det ikke nødvendigt at specificere lokationen af repository, når man arbejder i en kopi. Tilføjer man et katalog, oprettes det med det samme i repository. Tilføjelse af filer er at betragte som ændringer. De skal efterfølgende gøres bindende. Dette gøres med *commit* kommandoen (se afsnit 15.3.4).

**Eksempel** Lad os oprette nogle kataloger og nogle filer i vores repository:

```
cd $HOME/Nano42
mkdir data rapport
cvs add data rapport
cd data
echo '1.43521e-09 1.8723e-09 4.63463e-09' >>strangedata.dat
cvs add strangedata.dat
```

Tilbage er der så at gøre tilføjelsen af *strangedata.dat* bindende. Dette gøres med *commit* kommandoen (se afsnit 15.3.4).

### 15.3.4 Ændringer af filer i projektet

De ændringer man foretager på sin kopi er individuelle. Ønsker man at publishere sine ændringer til andre, skal de føres tilbage til repository. Til dette anvendes *commit* kommandoen:

```
cvs commit [-m logbesked] [filnavn...]
```

Operationen binder ændringerne i *filnavn* til repository og øger filens versionsnummer med 1. Samtidigt gemmes ens *logbesked* som kommentar til ændringen. Undlader man at anvende “-m” parameteren, startes ens valgte editor så man kan skrive en passende logbesked<sup>1</sup>.

<sup>1</sup>Editoren findes ved at kigge på miljøvariablene *\$CVSEEDITOR*, *\$VISUAL* samt *\$EDITOR* i nævnte rækkefølge.

**Eksempel:** Vi vil gøre tilføjelsen af filen `strangedata.dat` fra det tidligere eksempel bindende:

```
cd $HOME/Nano42/data
cvs commit -m 'Initiel tilføjelse' strandedata.dat
```

Nu vil vi så tilføje noget ekstra data til filen og publishere ændringen:

```
echo '1.43213451e-09' >>strangedata.dat
cvs commit -m 'Tilføjede næste data-værdi' strandedata.dat
```

### 15.3.5 Opdatering

Når man gerne vil synkronisere sin kopi med repository anvendes *update* kommandoen. Kommandoen skal kaldes inde fra ens egen kopi:

```
cvs update [-d] [-P]
```

Herefter vil kopien være synkroniseret med repository og have de nyeste versioner af filer. Opdatering foregår kun for filer i det nuværende katalog og samtlige underkataloger. På denne måde kan man undgå at skulle opdatere hele kopien hver gang men kan nøjes med en delmængde.

Et *update* er nondestruktivt i den forstand, at de ændringer man selv har foretaget forsøges bibeholdt. I det tilfælde hvor ændringer fra repository overlapper med ens egne ændringer opstår der en konflikt. Hvordan de løses kan læses i afsnit 15.3.7.

Der er 2 parametre angivet i syntaksen. “-d” vil automatisk oprette nye kataloger, hvis disse tilføjes til repository. “-P” vil medføre, at tomme kataloger automatisk slettes fra ens kopi.

### 15.3.6 Sletning af filer fra repository

Hvis en fil skal slettes fra repository må man først slette den fra ens egen kopi. Derefter kan filen slettes fra repository med:

```
cvs rm <filnavn>
```

Filen slettes ikke totalt fra repository, og ældre udgaver af filen vil stadig kunne udtrækkes. Man kan derfor ikke miste data ved sletning. Filen vil blive slettet fra andre kopier når der udføres en *update* operation på disse.

### 15.3.7 Konflikter i forbindelse med ændringer i filer

I visse tilfælde vil en *update* føre til konflikt. Det sker når de ændringer man selv har foretaget, overlapper med ændringer andre har foretaget. De overlappende områder i de to filer bliver markeret på følgende måde:

```
<<<<<<<<<<<<
  <version 1>
=====
  <version 2>
>>>>>>>>>>>>
```

Konflikter løses ved at finde ovenstående markeringer. Dernæst sletter man markeringerne og det, der ikke skal være der. Man kan slette den ene eller den anden version. Man kan blande versionerne som man vil, eller skrive noget helt andet.

Til sidst udfører man en *commit*, så der oprettes en ny version i repository med konflikten løst.



### 15.4.3 Adgang til cvs-repositories på andre konti

Når man skal tilgå et repository på en anden konto angiver man ikke stien til repository som normalt. Istedet angiver man følgende:

```
:ext:<repositorykonto>@<host>:<fuld sti til repository>
```

Her følger en forklaring af de forskellige felter:

**repositorykonto** Den konto hvorpå repository befinder sig.

**host** Den computer, hvorpå repositorykontoen befinder sig.

**fuld sti til repository** Dette er den fulde sti til repository på den pågældende maskine. Man kan finde den fulde sti ved at skifte katalog til repositorykataloget og anvende `pwd(1)` kommandoen.

**Eksempel:** Lad os sige, at vi sidder på kontoen “testuser” og skal tilgå repository på kontoen “cvsuser”. Vi har givet adgang til testuser på cvsuser’s katalog som beskrevet i 15.4.2. Vi vil nu udtrække repository, der befinder sig i kataloget

```
/b/04/cvsuser/project-repository
```

Det gøres på følgende måde:

```
cvs -d :ext:cvsuser@fys.ku.dk:/b/04/cvsuser/project-repository checkout .
```

## 15.5 Avanceret brug af CVS

### 15.5.1 cvs diff

`cvs diff` kan lave sammenligninger mellem forskellige versioner af filer i cvs. Det kan være et nyttigt redskab, da man kan se de aktuelle ændringer foretaget i en fil. Syntaksen er:

```
cvs diff [-u] [-r rev1] [-r rev2] [filer...]
```

Sammenligningen af filerne foregår normalt mod den seneste version i repository, men det er muligt at specificere en anden version med “-r” parameteren. Med 2 “-r” parametre kan man sammenligne forskellige versioner i repository. “-u” parameteren udskriver i såkaldt “unified” format, hvilket kan være en del lettere at læse.

**Eksempel** Vi vil gerne se forskellene mellem version 1.20 og version 1.24 af filen `rapport.tex`. Vi vil gerne have output i unified format. Da vi forventer der er mange ændringer sender vi resultatet til `less(1)` gennem en pipe:

```
cvs diff -u -r 1.20 -r 1.24 rapport.tex | less
```



### 15.5.2 Udtræk af ældre revisioner af en fil

Ældre versioner af en fil kan udtrækkes ved at give en “-r” parameter til *checkout* kommandoen. Det anbefales, at man ikke udtrækker filen oven i ens oprindelige repository. Istedet anbefales det, at man udtrækker filen separat. Man kan efterfølgende tage relevante dele af filen og overføre til sin kopi.

**Eksempel** Vi vil gerne udtrække filen `rapport.tex` i version 1.20 fra et givent repository. Filen befinder sig i kataloget “src” (set relativt i forhold til roden af repository).

```
cvcs -d /b/04/cvsuser/project-repository checkout -r 1.20 src/rapport.tex
```

## 15.6 Opsætning

Når man har anvendt CVS et stykke tid, finder man som regel en række parametre, man altid ønsker at give *cvcs*. Eksempeltvist kunne vi ønske en opsætning hvor *checkout* altid har “-P”, *update* altid har “-dP” og *diff* altid har “-u” parametrene sat. Dette kan gøres ved at oprette filen `.cvsrc` i ens hjemmekatalog og skrive følgende ind i den:

```
checkout -P
update -dP
diff -u
```

CVS holder øje med en række miljøvariable. De væsentligste er:

**\$CVSEEDITOR** Den editor der anvendes når man skriver logbeskeder. Hvis **\$CVSEEDITOR** variabelen ikke kan findes, ledes først efter **\$VISUAL** og dernæst efter **\$EDITOR** variablene.

**\$CVSROOT** Lokaliteten af repository. Hvis man kun har et enkelt repository kan det være en fordel at sætte denne variabel til at pege herpå.

**\$CVS\_RSH** Angiver metoden for at kommunikere mellem conti (f.eks. ssh).

## 15.7 Adgang til repository på systemet over SSH

Man kan tilgå et repository på systemet over SSH. Metoden er som følger:

1. Fra maskinen der skal tilgå repository skal miljøvariablen **\$CVS\_RSH** sættes til at være “ssh”:

```
export CVS_RSH=ssh
```

Og for tcsh-lignende shells:

```
setenv CVS_RSH ssh
```

2. Kildekoden kan nu udtrækkes som var det tilgang fra en anden konto på systemet. Vil vi eksempelvis udtrække et repository der befinder sig hos brugeren “cvsuser” og lokaliteten af repository er `/b/04/cvsuser/cvsrep`, gøres følgende:

```
cvcs -d :ext:cvsuser@fys.ku.dk:/b/04/cvsuser/cvsrep checkout .
```



## Bilag A

# Kommandoreference

Dette bilag er en liste over de mest almindeligt brugte UNIX kommandoer, samt et kort forklaring. Listen kan give et overblik over hvad der findes, men man opfordres til at finde yderligere information (f.eks. ved brug af `man`-kommandoen).

### A.1 Kommandoer opdelt efter emne

Filhåndtering	<code>chmod</code> , <code>cp</code> , <code>mkdir</code> , <code>mv</code> , <code>rm</code> , <code>rmdir</code> .
Filmanipulering	<code>cat</code> , <code>grep</code> , <code>gzip</code> , <code>head</code> , <code>less</code> , <code>sort</code> , <code>tail</code> , <code>wc</code> .
Jobstyring	<code>bg</code> , <code>fg</code> , <code>jobs</code> , <code>kill</code> , <code>ps</code> .
Udskrivning	<code>lpr</code> , <code>lprm</code> , <code>lpq</code> .

### A.2 Kommandoer i alfabetisk orden

**bg** Afvikler job i baggrunden, bruges typisk efter `suspend` (`<ctrl-z>`).

```
bg [job_id]
```

**cat** Viser filer. En nyttig parameter er `-n`, der tilføjer linienummerering. Kan bruges til sammensætning af filer:

```
cat file1 file2 fileN > ny_sammensat_fil
```

**cd** Skifter katalog.

```
cd [katalog]
```

Hvis der ikke angives et argument skiftes der til hjemmekataloget.

**chmod** Sætter rettighederne på en fil. Der er basalt to måder at bruge `chmod` på: symbolsk og absolut, her vil kun blive omtalt førstnævnte metode. Parameteren `-R` kan benyttes til at udføre kommandoen rekursivt. Rettigheder sættes for user/ejer `u`, group `g`, others `o` eller alle `a`, der kan bruges `+`, `-` og `=`, til henholdsvis at tilføje, fjerne eller sætte (alle) rettigheder for de forskellige grupper: Alle må læse fil:

```
chmod a+r fil
```

Andre end gruppen og ejeren må ikke noget som helst:

```
chmod o= fil
```

Gruppen må læse og afvikle fil:

```
chmod g=rx fil
```

Brug `ls -l` til efterfølgende at se rettighederne. Se også afsnit 4.3.

**clear** Renser skærmen.

```
clear
```

**convert** converterer mellem forskellige billedformater, f.eks.

```
convert picture.bmp picture.jpeg
```

for at konvertere billedet `picture` fra `bmp` til `jpeg` format. For en liste af understøttede formater henvises til man siden for `convert`

**cp** Kopierer filer.

```
cp fra_fil til_fil
```

**compress, uncompress** Pakker hhv. udpakker filer pakket i det *gamle* kommercielle unix-format der kendes på endelserne: `.z` og `.Z`.

**date** Udskriver tid og dato.

**df** Udskriver diskforbrug på diske mountet på den aktuelle server.

**diff** Viser forskellen på to filer.

```
diff fil1 fil2
```

**du** Viser diskforbrug fordelt på underkataloger fra kataloget kommandoen afvikles i. Parameteren `-h` bruges på mange systemer til at få størrelser i menneskevenligt (**h**uman readable) format.

**exit** Afslutter kommandofortolker.

**fg** Bringer job i forgrunden. Bruges ofte i forbindelse med `suspend` (`<ctrl-z>`).

```
fg [job_id]
```

**find** Finder filer der opfylder visse angivne prædikater. Fx kan `-name` bruges til at angive et mønster for navne.

```
find start_katalog udtryk
```

**ftp** Overfører filer imellem fysiske maskiner. Desværre er `ftp` usikker; det anbefales i stedet at anvende `sftp`.

**grep** Finder linier i filer der matcher et angivet udtryk. Meget anvendelige parametre er `-i`, for at angive at der ikke skal skelnes mellem store og små tegn (gælder `a-z`), samt `-v` for at vise linier der ikke matcher udtrykket.

```
grep -i 'bar' fil1 fil2 filN
```

**groups** Viser hvilke grupper en bruger er med i.

```
groups
```

**gzip, gunzip** Bruges til at pakke og udpakke filer med. Pakkede filer får endelsen `.gz`. Når filen pakkes eller udpakkes, slettes den gamle fil, således at der kun vil være en fil før og en fil efter operationen. Filen `hat.tar` pakkes:

```
gzip foo.tar
```

Dette vil bevirke at `foo.tar` forsvinder og der vil i stedet være en fil `foo.tar.gz`.

**head** Viser begyndelsen af en fil. Parameteren `-n` benyttes til at angive hvor mange linier der skal vises.

**jobs** Viser information om kørende jobs.

**kill** Bruges til at slå processer ihjel med; brug evt. `ps` til at få fat i proces-id (pid).

```
kill pid1 pid2 pidN
```

Parameteren `-9` kan bruges til genstridige processer – den tager ingen fanger, så brug den kun ved processer der ikke kan afsluttes på normal vis.

**less** Viser filer; benyttes bl.a. af `man`. Normal syntaks:

```
less fil
```

Se afsnit 4.4 for en oversigt over taster i `less`.

**ln** Laver et link i filsystemet.

**lpq** Viser printerkøen. Parameteren `-P` bruges til at angive printer med.

```
lpq [-Pprinter]
```

**lpr** Bruges til udskrivning, evt. med parametren `-P` til angivelse af printer;

**lprm** Fjerner job fra printerkøen. Enten kan man angive hvilke jobs der skal fjernes (med job-id fra `lpq`) eller undlade angivelse for at fjerne alle jobs man ejer på printeren.

**ls** Udskriver indholdet af et katalog. Parameteren `-l` angiver at der ønskes fuld information, `-a` at der ønskes medtaget filer der begynder med `'.'`, `-R` angiver at visningen skal foretages rekursivt.

**man** Viser manualsiden. Der angives evt. hvilken sektion siden skal tages fra.

```
man [NX] program
```

Ofte angives programmer eller biblioteker med noget i `"()`". Fx `crypt(3)`. I parenteser er angivet hvilken sektion manualen findes i. Dette har typisk kun betydning hvor der findes flere kommandoer eller biblioteker med samme navn. Manualen til `crypt(3)` vises med:

```
man 3 crypt
```

**mkdir** Opretter af katalog.

```
mkdir katalog
```

**mv** Omdøber (flytter) filer og kataloger.

```
mv gammel_fil ny_fil
```

**nohup** Program til start af processer der ikke skal afsluttes når man logger af systemet.

```
nohup kommando
```

Eksempelvis:

```
nohup wget ftp://ftp.xemacs.org/pub/xemacs/xemacs-19.14.tar.gz
```

**passwd** Ændrer password.

**ping** Afgør, om der kan oprettes forbindelse til en host.

```
/usr/sbin/ping slashdot.org
```

**ps** Viser hvilke processer der er kører på systemet. Hvis der ikke gives argumenter vises processer der er startet fra den aktuelle shell.

**pwd** Viser hvor man står i filsystemet:

**quota** Viser hvor meget diskplads man bruger i forhold til det tilladte:

```
quota -v
```

**rlogin** Forbinder den aktuelle server til en fremmed server:

```
rlogin [-l bruger] server
```

**rm** Sletter filer. *Slettede filer er forsvundet for altid!*. Brug **-i** for interaktiv mode, dvs. der spørges om hver enkelt fil. Parameteren **-R** sletter rekursivt filer og kataloger, men vær forsigtig!

```
rm filer
```

**rmdir** Sletter kataloger. Disse skal være tomme.

```
rmdir katalog
```

**uptime** Viser status for serverne i netværket.

```
uptime
```

**rwho** Identificerer brugere på det lokale net.

```
rwho
```

**sort** Sorterer. Parameteren **-r** angiver omvendt sortering.

**tail** Viser slutningen af en fil. Parameteren **-n N** benyttes til at angive hvor mange linier der skal vises; **-f** kan bruges på filer der vokser.

```
tail fil
```

**tar** Bruges til at behandle .tar-filer:

```
tar xvzf foo.tar.gz
```

Visning af indhold af arkivet foo.tar.gz:

```
tar tvzf foo.tar.gz
```

Pakning af aktuelle bibliotek til arkivet foo.tar.gz:

```
gtar cvzf foo.tar.gz .
```

Udpakning af arkivet foo.tar:

```
tar xvf foo.tar.gz
```

Visning af indhold af arkivet foo.tar:

```
tar tvf foo.tar.gz
```

Pakning af aktuelle bibliotek til arkivet foo.tar:

```
tar cvzf foo.tar .
```

**telnet** Forbinder til en fremmed host. Er ligesom ftp usikker, og vi anbefaler derfor at man bruger ssh.

```
telnet server
```

**scp** Krypteret kopiering over netværk.

**ssh** Opretter en krypteret forbindelse til en fremmed host. Som bonus bliver ens X-display sat op, så man uden videre kan åbne vinduer etc.

```
ssh [-l bruger] server
```

**su** Skifter bruger.

```
su [-] bruger
```

- angiver at brugerens omgivelser skal benyttes.

**wc** Tæller antallet af linier eller ord i en fil. Nedenstående tæller linierne i en fil:

```
wc -l fil
```

**wget** Henter filer via http eller ftp uden brug af browser eller ftp-klient.

**which** Viser hvor et program findes.

**who** Identificerer bruger der pt. er logget ind

```
who
```

**whoami** Identificerer brugeren der er logget ind.

```
whoami
```

**zcat** Benyttes til at vise (det ukomprimerede) indhold af en fil pakket med gzip. Se cat(1).

**zip, unzip** Benyttes til at pakke og udpakke zip-arkiver.





## Bilag B

# Regler for edb-systemet

Velkommen til computerfaciliteter ved Niels Bohr institutet for astronomi, fysik og geofysik  
Systemadministrationen fejl@fys.ku.dk

### B.1 Regler

Følgende regler gælder for adgang til computerfaciliteterne ved Ørsted Laboratoriet:

1. Til hver bruger udleveres en personlig kode (løsen/password) til brug for adgang til systemet. Denne kode er strengt personlig og må under ingen omstændigheder udleveres til andre, heller ikke systemadministratorer eller lignende. INGEN! Har en bruger formodning om, at andre har kendskab til vedkommendes eller andres personlige kode, skal systemadministratoren straks underrettes herom. Skift af koden skal ske mindst hvert halvår.
2. Det er ikke tilladt at hindre systemets daglige drift eller påvirke det på en måde, der medfører gene for de andre brugere af systemet. Herunder hører, at det ikke er tilladt at belaste netværket eller serverne uforholdsmæssigt meget, hvad enten denne belastning skyldes relevant arbejde eller ej.
3. Det er ikke tilladt at forsøge at skaffe sig adgang til information, man ikke har retmæssig adgang til. Det vil bl.a. sige, at det ikke er tilladt:
  - at søge adgang til andres personlige post, kataloger eller computere uden deres samtykke.
  - at overvåge netværkstrafikken.
  - at forsøge at skaffe sig uretmæssig adgang til system- eller logfiler på servere.
  - at forsøge at skaffe sig uretmæssig adgang til ikke-offentlige oplysninger på Internettet.
4. Brugeren skal straks underrette systemadministratoren ved mistanke om uautoriseret adgang, sikkerhedsfejl, virus og andet der kan være truende for systemets drift.
5. Brugeren skal vise hensyn til andre brugere, bl.a. ved ikke at optage terminalplads til useriøse formål.
6. Brugeren skal straks underrette systemadministratoren ved viden om andres brud på disse regler.
7. Brud på disse regler kan i forhold til forseelsens alvor medføre

- Tab af adgang til specifikke services.
  - Tab af adgang til systemet.
  - Udsmidelse fra universitetet.
  - Politianmeldelse.
8. Beslutninger om sanktioner og vurderinger i forbindelse med misbrug og lignende foretages af systemadministratoren i samråd med ansvarlige VIP'ere.

## **B.2 Systemadministratorens rettigheder**

Systemadministratoren har ret til, for at kunne udføre sit arbejde, at læse enhver fil på systemet heri inkluderet personlig post.



## Bilag C

# Emacs tastaturreference

## GNU Emacs Reference Card

(for version 20)

### Starting Emacs

To enter GNU Emacs 20, just type its name: **emacs**

To read in a file to edit, see Files, below.

### Leaving Emacs

suspend Emacs (or iconify it under X) **C-z**  
exit Emacs permanently **C-x C-c**

### Files

**read** a file into Emacs **C-x C-f**  
**save** a file back to disk **C-x C-s**  
**save all** files **C-x s**  
**insert** contents of another file into this buffer **C-x i**  
**replace** this file with the file you really want **C-x C-v**  
**write** buffer to a specified file **C-x C-w**  
**version control** checkin/checkout **C-x C-q**

### Getting Help

The help system is simple. Type **C-h** (or F1) and follow the directions. If you are a first-time user, type **C-h t** for a **tutorial**.

remove help window **C-x l**  
scroll help window **C-M-v**  
apropos: show commands matching a string **C-h a**  
show the function a key runs **C-h c**  
describe a function **C-h f**  
get mode-specific information **C-h m**

### Error Recovery

**abort** partially typed or executing command **C-g**  
**recover** a file lost by a system crash **M-x recover-file**  
**undo** an unwanted change **C-x u** or **C-\_**  
**restore** a buffer to its original contents **M-x revert-buffer**  
**redraw** garbaged screen **C-l**

### Incremental Search

search forward **C-s**  
search backward **C-r**  
regular expression search **C-M-s**  
reverse regular expression search **C-M-r**  
select previous search string **M-p**  
select next later search string **M-n**  
exit incremental search **RET**  
undo effect of last character **DEL**  
abort current search **C-g**

Use **C-s** or **C-r** again to repeat the search in either direction. If Emacs is still searching, **C-g** cancels only the part not done.

© 1997 Free Software Foundation, Inc. Permissions on back, v2.2

### Motion

entity to move over	backward	forward
character	<b>C-b</b>	<b>C-f</b>
word	<b>M-b</b>	<b>M-f</b>
line	<b>C-p</b>	<b>C-n</b>
go to line beginning (or end)	<b>C-a</b>	<b>C-e</b>
sentence	<b>M-a</b>	<b>M-e</b>
paragraph	<b>M-{</b>	<b>M-}</b>
page	<b>C-x [</b>	<b>C-x ]</b>
sexp	<b>C-M-b</b>	<b>C-M-f</b>
function	<b>C-M-a</b>	<b>C-M-e</b>
go to buffer beginning (or end)	<b>M-&lt;</b>	<b>M-&gt;</b>
scroll to next screen	<b>C-v</b>	
scroll to previous screen	<b>M-v</b>	
scroll left	<b>C-x &lt;</b>	
scroll right	<b>C-x &gt;</b>	
scroll current line to center of screen	<b>C-u C-l</b>	

### Killing and Deleting

entity to kill	backward	forward
character (delete, not kill)	<b>DEL</b>	<b>C-d</b>
word	<b>M-DEL</b>	<b>M-d</b>
line (to end of)	<b>M-O C-k</b>	<b>C-k</b>
sentence	<b>C-x DEL</b>	<b>M-k</b>
sexp	<b>M-- C-M-k</b>	<b>C-M-k</b>

**kill region** **C-y**  
**copy region to kill ring** **M-w**  
**kill through next occurrence of char** **M-z char**  
**yank back last thing killed** **C-y**  
**replace last yank with previous kill** **M-y**

### Marking

**set mark here** **C-@** or **C-SPC**  
**exchange point and mark** **C-x C-x**  
**set mark *arg* words away** **M-@**  
**mark paragraph** **M-h**  
**mark page** **C-x C-p**  
**mark sexp** **C-M-@**  
**mark function** **C-M-h**  
**mark entire buffer** **C-x h**

### Query Replace

interactively replace a text string **M-/\_**  
using regular expressions **M-x query-replace-regexp**  
Valid responses in query-replace mode are  
**replace** this one, go on to next **SPC**  
**replace** this one, don't move **,**  
**skip** to next without replacing **DEL**  
**replace** all remaining matches **!**  
**back up** to the previous match **-**  
**exit** query-replace **RET**  
**enter recursive edit (C-M-c to exit)** **C-r**

### Multiple Windows

When two commands are shown, the second is for "other frame."

delete all other windows	<b>C-x 1</b>
split window, above and below	<b>C-x 2</b> <b>C-x 5 2</b>
delete this window	<b>C-x 0</b> <b>C-x 5 0</b>
split window, side by side	<b>C-x 3</b>
scroll other window	<b>C-M-v</b>
switch cursor to another window	<b>C-x o</b> <b>C-x 5 o</b>
select buffer in other window	<b>C-x 4 b</b> <b>C-x 5 b</b>
display buffer in other window	<b>C-x 4 C-o</b> <b>C-x 5 C-o</b>
find file in other window	<b>C-x 4 f</b> <b>C-x 5 f</b>
find file read-only in other window	<b>C-x 4 r</b> <b>C-x 5 r</b>
run Dired in other window	<b>C-x 4 d</b> <b>C-x 5 d</b>
find tag in other window	<b>C-x 4 .</b> <b>C-x 5 .</b>
grow window taller	<b>C-x ^</b>
shrink window narrower	<b>C-x {</b>
grow window wider	<b>C-x }</b>

### Formatting

indent current line (mode-dependent)	<b>TAB</b>
indent region (mode-dependent)	<b>C-M-\</b>
indent sexp (mode-dependent)	<b>C-M-q</b>
indent region rigidly <i>arg</i> columns	<b>C-x TAB</b>
insert newline after point	<b>C-o</b>
move rst of line vertically down	<b>C-M-o</b>
delete blank lines around point	<b>C-x C-o</b>
join line with previous (with arg, next)	<b>M-^</b>
delete all white space around point	<b>M-\</b>
put exactly one space around point	<b>M-SPC</b>
fill paragraph	<b>M-q</b>
set fill column	<b>C-x f</b>
set prefix each line starts with	<b>C-x .</b>
set face	<b>M-g</b>

### Case Change

uppercase word	<b>M-u</b>
lowercase word	<b>M-l</b>
capitalize word	<b>M-c</b>
uppercase region	<b>C-x C-u</b>
lowercase region	<b>C-x C-l</b>

### The Minibuffer

The following keys are defined in the minibuffer.

complete as much as possible	<b>TAB</b>
complete up to one word	<b>SPC</b>
complete and execute	<b>RET</b>
show possible completions	<b>?</b>
fetch previous minibuffer input	<b>M-p</b>
fetch later minibuffer input or default	<b>M-n</b>
regex search backward through history	<b>M-r</b>
regex search forward through history	<b>M-s</b>
abort command	<b>C-g</b>

Type **C-x ESC ESC** to edit and repeat the last command that used the minibuffer. Type F10 to activate the menu bar using the minibuffer.

GNU Emacs Reference Card

Buffers

select another buffer	C-x b
list all buffers	C-x C-b
kill a buffer	C-x k

Transposing

transpose <b>characters</b>	C-t
transpose <b>words</b>	M-t
transpose <b>lines</b>	C-x C-t
transpose <b>sexps</b>	C-M-t

Spelling Check

check spelling of current word	M-\$
check spelling of all words in region	M-x ispell-region
check spelling of entire buffer	M-x ispell-buffer

Tags

find a tag (a definition)	M-.
find next occurrence of tag	C-u M-.
specify a new tags file	M-x visit-tags-table
regex search on all files in tags table	M-x tags-search
run query-replace on all the files	M-x tags-query-replace
continue last tags search or query-replace	M-,

Shells

execute a shell command	M-!
run a shell command on the region	M-
filter region through a shell command	C-u M-
start a shell in window *shell*	M-x shell

Rectangles

copy rectangle to register	C-x r r
kill rectangle	C-x r k
yank rectangle	C-x r y
open rectangle, shifting text right	C-x r o
blank out rectangle	C-x r c
prefix each line with a string	C-x r t

Abbrevs

add global abbrev	C-x a g
add mode-local abbrev	C-x a l
add global expansion for this abbrev	C-x a i g
add mode-local expansion for this abbrev	C-x a i l
explicitly expand abbrev	C-x a e
expand previous word dynamically	M-/

Regular Expressions

any single character except a newline	.	(dot)
zero or more repeats	*	
one or more repeats	+	
zero or one repeat	?	
quote regular expression special character <i>c</i>	\c	
alternative ("or")		
grouping	\( ... \)	
same text as <i>n</i> th group	\n	
at word break	\b	
not at word break	\B	
<b>entity</b>	<b>match start</b>	<b>match end</b>
line	^	\$
word	\<	\>
buffer	\'	\'
<b>class of characters</b>	<b>match these</b>	<b>match others</b>
explicit set	[ ... ]	[^ ... ]
word-syntax character	\w	\W
character with syntax <i>c</i>	\sc	\Sc

International Character Sets

specify principal language	M-x set-language-environment
show all input methods	M-x list-input-methods
enable or disable input method	C-\
set coding system for next command	C-x RET c
show all coding systems	M-x list-coding-systems
choose preferred coding system	M-x prefer-coding-system

Info

enter the Info documentation reader	C-h i
find specified function or variable in Info	C-h C-i

Moving within a node:

scroll forward	SPC
scroll reverse	DEL
beginning of node	. (dot)

Moving between nodes:

<b>next</b> node	n
<b>previous</b> node	p
move <b>up</b>	u
select menu item by name	m
select <i>n</i> th menu item by <b>number</b> (1-9)	n
follow cross reference (return with 1)	f
return to last node you saw	l
return to directory node	d
go to any node by name	g

Other:

run Info <b>tutorial</b>	h
<b>quit</b> Info	q
search nodes for regexp	M-s

Registers

save region in register	C-x r s
insert register contents into buffer	C-x r i
save value of point in register	C-x r SPC
jump to point saved in register	C-x r j

Keyboard Macros

<b>start</b> defining a keyboard macro	C-x (
<b>end</b> keyboard macro definition	C-x )
<b>execute</b> last-defined keyboard macro	C-x e
append to last keyboard macro	C-u C-x (
name last keyboard macro	M-x name-last-kbd-macro
insert Lisp definition in buffer	M-x insert-kbd-macro

Commands Dealing with Emacs Lisp

eval sexp before point	C-x C-e
eval current <b>defun</b>	C-M-x
eval <b>region</b>	M-x eval-region
read and eval minibuffer	M-:
load from standard system directory	M-x load-library

Simple Customization

customize variables and faces	M-x customize
Making global key bindings in Emacs Lisp (examples):	
(global-set-key "\C-cg" 'goto-line)	
(global-set-key "\M-#" 'query-replace-regexp)	

Writing Commands

(defun <i>command-name</i> ( <i>args</i> ) " <i>documentation</i> " (interactive " <i>template</i> ") <i>body</i> )
An example:
(defun this-line-to-top-of-window (line) "Reposition line point is on to top of window. With ARG, put point on line ARG." (interactive "P") (recenter (if (null line) 0 (prefix-numeric-value line))))

The interactive spec says how to read arguments interactively. Type C-h f interactive for more details.

Copyright © 1997 Free Software Foundation, Inc.  
v2.2 for GNU Emacs version 20, June 1997  
designed by Stephen Gildea

Permission is granted to make and distribute copies of this card provided the copy-  
right notice and this permission notice are preserved on all copies.  
For copies of the GNU Emacs manual, write to the Free Software Foundation, Inc.,  
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA





## Bilag D

# Vi tastaturreference

### Vi Reference Card

#### Modes

Vi has two modes: insertion mode, and command mode. The editor begins in command mode, where cursor movement and text deletion and pasting occur. Insertion mode begins upon entering an insertion or change command. [ESC] returns the editor to command mode (where you can quit, for example by typing :q!). Most commands execute as soon as you type them except for "colon" commands which execute when you press the return key.

#### Quitting

exit, saving changes :x  
quit (unless changes) :q  
quit (force, even if unsaved) :q!

#### Inserting text

insert before cursor, before line i , I  
append after cursor, after line a , A  
open new line after, line before o , O  
replace one char, many chars r , R

#### Motion

left, down, up, right h , j , k , l  
next word, blank delimited word w , W  
beginning of word, of blank delimited word b , B  
end of word, of blank delimited word e , E  
sentence back, forward ( , )  
paragraph back, forward { , }  
beginning, end of line ^ , \$  
beginning, end of file 0 , \$  
line n 1G , G  
forward, back to char c nG or :n  
forward, back to before char c fc , Fc  
top, middle, bottom of screen H , M , L

#### Deleting text

Almost all deletion commands are performed by typing d followed by a *motion*. For example dw deletes a word. A few other deletions are:

character to right, left x , X  
to end of line D  
line dd  
line :d

#### Yanking text

Like deletion, almost all yank commands are performed by typing y followed by a *motion*. For example y\$ yanks to the end of line. Two other yank commands are:

line yy  
line :y

#### Changing text

The change command is a deletion command that leaves the editor in insert mode. It is performed by typing c followed by a *motion*. For example cw changes a word. A few other change commands are:

to end of line C  
line cc

#### Putting text

put after position or after line P  
put before position or before line p

#### Registers

Named registers may be specified before any deletion, change, yank, or put command. The general prefix has the form "c where c may be any lower case letter. For example, "adw deletes a word into register a. It may thereafter be put back into the text with an appropriate put command, for example "ap.

#### Markers

Named markers may be set on any line of a file. Any lower case letter may be a marker name. Markers may also be used as the limits for ranges.

set marker c on this line mc  
goto marker c 'c  
goto marker c first non-blank 'c

#### Search for strings

search forward /string  
search backward ?string  
repeat search in same, reverse direction n , N

#### Replace

The search and replace function is accomplished with the :s command. It is commonly used in combination with ranges or the :g command (below).

replace pattern with string :s/pattern/string/flags  
flags: all on each line, confirm each g , c  
repeat last :s command &

#### Regular expressions

any single character except newline . (dot)  
zero or more repeats \*  
any character in set [...] [^...]  
any character not in set [^...]  
beginning, end of line ^ , \$  
beginning, end of word < , >  
grouping \ ( , \ )  
contents of nth grouping \n

#### Counts

Nearly every command may be preceded by a number that specifies how many times it is to be performed. For example 5dw will delete 5 words and 3fe will move the cursor forward to the 3rd occurrence of the letter e. Even insertions may be repeated conveniently with this method, say to insert the same line 100 times.

#### Ranges

Ranges may precede most "colon" commands and cause them to be executed on a line or lines. For example :3,7d would delete lines 3–7. Ranges are commonly combined with the :s command to perform a replacement on several lines, as with :.,\$s/pattern/string/g to make a replacement from the current line to the end of the file.

lines n-m :n,m  
current line :.  
last line :\$  
marker c :'c  
all lines :%  
all matching lines :g/pattern/

#### Files

write file (current file if no name given) :w file  
append file (current file if no name given) :w >>file  
read file after line :r file  
read program output :r !program  
next file :n  
previous file :p  
edit new file :e file  
replace line with program output :.!program

#### Other

toggle upper/lower case ~  
join lines J  
repeat last text-changing command .  
undo last change, all changes on line u , U



# Litteratur

- [1] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The L<sup>A</sup>T<sub>E</sub>X Companion*. Addison–Wesley, 1994.
- [2] John Grossman, editor. *The Chicago Manual of Style*. The University of Chicago Press, fourteenth edition, 1993.
- [3] J.M. Knudsen and P.G. Hjorth. *Elements of Newtonian Mechanics*. Springer, second edition, 1996.
- [4] Donald E. Knuth. *The T<sub>E</sub>Xbook*, volume A of *Computers & Typesetting*. Addison–Wesley, 1986.
- [5] Helmut Kopka and Patrick W. Daly. *A Guide to L<sup>A</sup>T<sub>E</sub>X*. Addison–Wesley, third edition, 1999.
- [6] Tobias Oetiker, Hubert Partl, Irene Hyna, and Elisabeth Schlegl. *The Not So Short Introduction to L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>*, 1996.
- [7] Jesper Pedersen et al. *Teach Yourself Emacs in 24 Hours*. Sams Publishing, 1999.
- [8] Jerry Peek, Tim O'Reilly, and Mike Loukides. *UNIX Power Tools*. O'Reilly & Associates, second edition, 1997.
- [9] Jon Warbrick et al. *Essential L<sup>A</sup>T<sub>E</sub>X++*, 1994.

# Indeks

- ., 15
- .., 15
- .emacs, 83
- .tcshrc, 81
- >, 75
- >>, 75
- >&, 75
- %, 77
- &, 76
- L<sup>A</sup>T<sub>E</sub>X-mode, 84
- |, 79
- ~, 15
- 2>, 75
  
- alias, 74
- \and, 53
- \author, 53
  
- bærbar, 68
- bg, 77, 95
- billedmanipulation, 62
- bookman, 54
- buffer, 26
- built-ins, 18
  
- cat, 15, 95
- cd, 15, 95
- cdrom, 6
- \center, 56
- \chapter, 51
- charter, 54
- chmod, 17, 95
- clear, 96
- completion, 74
- compress, uncompress, 96
- convert, 57, 96
- cp, 15, 96
- CVS, 87
- cvs
  - add, 89
  - checkout, 88
  - commit, 89
  - diff, 92
  - init, 88
  - konflikter i, 90
  - på systemet, 91
  - rm, 90
  - SSH adgang, 93
  - update, 90
- \$CVS\_RSH, 93
- \$CVSEDITOR, 89, 93
- \$CVSR00T, 93
  
- Datalokaler, 1
- date, 96
- \date, 53
- description, 55
- df, 96
- diff, 96
- disketter, 6
- disketter fra MS Windows, 6
- diskkvote, 5
- \div, 52
- \documentclass, 48
- dot-filer, 21
- du, 96
- dvi, 49
- dvi-filer, visning af, 65
- dvips, 49
  
- echo, 81
- \$EDITOR, 89, 93
- elektronisk post, 35
- emacs, 25
- email, 35
- \emph, 51
- enumerate, 55
- epsfig, 57
- \epsfig, 57
- euler, 54
- exit, 96
  
- fg, 77, 96
- fil
  - ejer, 17
  - gruppe, 17
  - rettigheder, 16, 18
  - usynlig, 15
- file
  - dot-, Se fil, usynlig

- group, *Se* fil, gruppe
- hidden-, *Se* fil, usynlig
- owner, *Se* fil, ejer
- filer fra terminalserveren, 11
- find, 77, 96
- findfind, 79
- finger, 69
- fontenc, 48
- \footnote, 54
- forward af email, 44
- \frac, 52
- ftp, 96
- genvejstaster i KDE, 9
- globbing, *Se* mønstre
- grep, 79, 80
- grep, 78, 96
- groups, 97
- gunzip, 97
- gzip, 97
- head, 97
- historie, 74
- history, *Se* historie
- Hjælp, 67
- hjemmekataloger, 4
- \hline, 56
- \$HOME, 81, 88
- info, 67
- inputenc, 48
- ispell, 85
- itemize, 55
- jobs, 77, 97
- kill, 77, 97
- kill-ring, 31
- klippe-klistre
  - emacs, 30
- kommandoer
  - alfabetisk, 95
  - almindelige, 15
  - efter emne, 95
  - indtastning af, 73
- kommandofortolker
  - ændring af, 44
- kommandoprompten, 13
- komprimering, 96, 99
- Konto, 1
- \label, 54
- \LaTeX, 47
- latex, 48
- L<sup>A</sup>T<sub>E</sub>X
- afsnit, 51
- aritmetiske operatorer, 52
- bøger om, 58
- beskrivelse, 55
- diagrammer, 56
- direktiver, 47
- dokumentation af, 58
- fejl, 49
- gåseøjne, 52
- inddelingsdirektiver, 51
- indholdsfortegnelse, 53
- kapitel, 51
- linieskift, 52
- lister, 54
- preamble, 48
- punkter, *Se* L<sup>A</sup>T<sub>E</sub>X, lister
- reserverede tegn, 50
- sideskift, 52
- stavekontrol og, 57
- tabeller, 55
- titlepage, 53
- Windows og, 58
- less, 15, 18, 97
- Linux, 69
- ln, 97
- logge ind
  - hjemmefra, 39
  - i terminalrummene, 1
- logge ud, 3
- lpq, 46, 97
- lpr, 45, 97
- lprm, 46, 97
- ls, 15, 97
- lyx, 58
- mønstre, 16
- mail over web, 43
- major-mode, 26
- \maketitle, 53
- man, 18, 97
- \$MANPATH, 81
- manualsider, vii
- mark, 30
- mini-buffer, 27
- mkdir, 15, 97
- mode-line, 26
- \multicolumn, 56
- mutt, 35
- mv, 15, 98
- nohup, 98
- \noindent, 51
- omdirigering, *Se* redirection

- \pageref, 54
- palatino, 54
- \parindent, 51
- \parskip, 51
- passwd, 98
- passwordskift, 44
- \$PATH, 81
- pathname, *Se* stinavn
- pauseskærm, 3
- pdf-filer, visning af, 65
- permissions, *Se* rettigheder
- pine, 37
- ping, 98
- pipe, 79
- point, 26
- Postscript, 46
- postscript, 49, 57
- printer
  - standard, 45
- proces, 76
  - i baggrunden, 76
  - i forgrunden, 76
  - id, 76
  - kørende, 76
- projektfiler, 61
- ps, 98
- ps-filer, visning af, 65
- pstricks, 57
- pwd, 15, 98
- quota, *Se* diskkvote
- quota, 98
- redirection, 75
- \ref, 54
- region, 30
- regulært udtryk, 78
- regular expression, *Se* regulært udtryk
- rettigheder, 16
- rlogin, 98
- rm, 15, 98
- rmdir, 15, 98
- ruptime, 98
- rwho, 98
- søgning
  - efter filer, *Se* find(1)
  - i filer, *Se* grep(1)
- scp, 99
- screen shots, 62
- script, 81
- \section, 51
- setenv, 81
- shell-variables, 81
- signatur, 36
- sikkerhed, 67
- silent accept, 15
- skifte password, 3
- sort, 98
- spam, 37
- spamfilter, 44
- ssh, 99
- standardrettigheder, *Se* fil, standardrettigheder
- stavekontrol, 57, 85
- stderr, 75
- stdout, 75
- stinavn, 13
- su, 99
- \subsection, 51
- \subsubsection, 51
- suspended, *Se* suspenderet
- suspenderet, 77
- \tableofcontents, 53
- tail, 98
- tar, 99
- tekniske tegninger, 63
- telnet, 99
- terminal serveren, 11
- TeX, 58
- times, 54
- \times, 52
- \title, 53
- \today, 53
- udfyldning
  - automatisk, 74
- umask, 18
- unalias, 75
- unzip, 99
- usb, 6
- \usepackage, 48, 54
- \$USER, 81
- utopia, 54
- variable
  - i kommandofortolkeren, 81
- \verb, 50
- versionskontrol, 87
- vi
  - gem, 23
  - hjælp, 25
  - klip, 23
  - kopier, 23
  - luk, 23
  - søg og erstat, 23
  - tilstande, 22

vim, 22  
vimtutor, 25  
\$VISUAL, 89, 93  
  
wc, 79, 99  
webscripts, 44  
wget, 99  
whatis, 19  
which, 99  
who, 99  
whoami, 99  
Windowmanagere, 7  
WYSIWYG, 47  
  
xdvi, 49  
xfig, 57  
xypic, 57  
  
zcat, 99  
zip, 99