Indhold

1 UNIX filsystemet

I det følgende beskrives simple dele af operativsystemet UNIX. De fleste af de beskrevne kommandoer er – selvom de hedder (eller kan forkortes til) det samme som standard UNIX kommandoer – specielt tilrettede til Dat1 systemet. Man kan altså ikke være sikker på, at disse kommandoer virker på helt samme måde, som de tilsvarende kommandoer i et "ægte" UNIX system.

1.1 Hvad er en fil?

En fil en en følge af ASCII tegn, der kan behandles som en sammenhængende enhed. Til en fil hører følgende:

- Et *navn*. Filers navne er sekvenser af tegn. Ofte vil navnet være af formen ≪filnavn≫.≪extension≫ (bemærk punktummet). Filens *extension* kan man tænke på som et efternavn; det angiver, hvordan indholdet tænkes brugt. Fx vil extension `tri` betyde, at filen indeholder et TRINE program.
- En *ejer*. En fil ejes af den bruger, der har skabt den.
- En *dato*. Man kan se dato og klokkeslet for den sidste ændring af filen.
- En *længde*. Filens længde angiver, hvor mange tegn den indeholder.
- En *beskyttelse*. En fil kan være beskyttet, så kun ejeren kan læse den.

Filers indhold kan opfattes som tekster, programmer, relationer og meget andet.

1.2 Hvad er et directory?

Et *directory* er et sted i UNIX-systemet, hvor filer ligger. Hver bruger har et *home directory* til opbevarelse af sine egne filer. Et directory kan desuden indeholde andre directories, såkaldte *(sub)directories*, der igen kan

indeholde filer og flere directories. På denne måde kan man organisere sine filer i en træstruktur. Tænk på filer som *dokumenter* og på directories som *foldere* eller *mapper*, der kan indeholde dokumenter og andre foldere.

1.3 Hvad er en path?

Hver fil og hvert directory kan angives med en entydig *path*. Home directory for brugeren ≪bruger≫ hedder `/users/`≪bruger≫. En fil ≪fil≫ i et directory med path ≪path≫ har selv path ≪path≫`/`≪fil≫; et subdirectory ≪sub≫ har tilsvarende selv path ≪path≫`/`≪sub≫. Fx vil filen `opg.tri` i home directory for brugeren `mis` have path

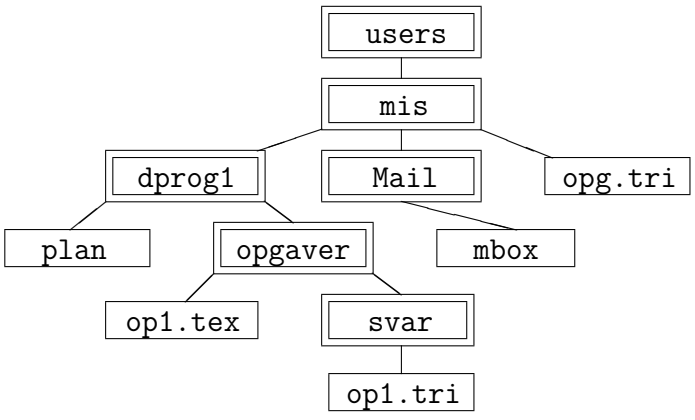
`/users/mis/opg.tri`

Hvis dette home directory også indeholder et subdirectory med navn `dprog1`, der igen indeholder filen `plan`, så har denne fil path

`/users/mis/dprog1/plan`

Når man arbejder i systemer, så er man altid positioneret i et *aktuelt directory*. Filer og directories kan enten angives ved deres fulde path, som beskrevet ovenfor, eller ved en *relativ path* ud fra det aktuelle directory. Hvis dette indeholder filer eller subdirectories, så kan de angives blot med deres navn (uden en foranstillet skråstreg). Det unikke directory ovenover det aktuelle kan, hvis det findes, angives med path `..` (to prikker).

Betragt følgende (fiktive) system af filer og directories, tegnet som et træ. Filer er angivet i enkeltstregede kasser, directories i dobbeltstregede.



Antag, at det aktuelle directory er `opgaver`. Så er

```
            ..  =  /users/mis/dprog1
        op1.tex  =  /users/mis/dprog1/opgaver/op1.tex
     ../../Mail  =  /users/mis/Mail
   svar/op1.tri  =  /users/mis/dprog1/opgaver/svar/op1.tri
     ../opgaver  =  /users/mis/dprog1/opgaver
```

Man kan skifte det aktuelle directory til ≪path≫ med kommandoen

`changedir` ≪path≫ (eller kort `cd` ≪path≫)

Det aktuelle directory starter altid med at være brugerens home directory. I UNIX-vinduet bliver det aktuelle directory's path altid skrevet først på hver kommandolinje. Hvis man "farer vild", så kan man komme hjem til sit home directory med kommandoen `cd` (uden argument).

1.4 Hvad kan man gøre med filer og directories?

Med kommandoen

`list` (eller kort `ls`)

kan man se indholdet af det aktuelle directory. Det skrives på følgende form

```
mis          2048 Apr  8 13:12 opgaver/
mis         75228 Jul 17 11:16 op1.tri
mis          1024 Jul 18 11:17 op1*
```

Første søjle er brugernavnet på ejeren. Anden søjle er længden i antal tegn. Tredje til femte søjle er dato og klokkeslet for sidste ændring. Sjette søjle er navnet. Hvis navnet efterfølges af `/` så angiver det et directory, ellers en fil. Hvis navnet på en fil efterfølges af `*`, så indeholder den et oversat program, der kan køres. Bemærk, at hverken `/` eller `*` er en del af navnet. Kommandoen

`remove` ≪path≫ (eller kort `rm` ≪path≫)

fjerner den angivne fil. Man skal dog først bekræfte, at det skal ske. Kommandoen

`removedir` ≪path≫ (eller kort `rmdir` ≪path≫)

fjerner det angivne directory, der i forvejen skal være tomt. Kommandoen

`copy` ≪path1≫ ≪path2≫ (eller kort `cp` ≪path1≫ ≪path2≫)

laver en kopi af filen med path ≪path1≫; kopien får path ≪path2≫. Hvis en sådan fil allerede findes, så bliver den overskrevet. Kommandoen

`makedir` ≪navn≫ (eller kort `mkdir` ≪navn≫)

skaber i det aktuelle directory et nyt tomt subdirectory med navn ≪navn≫. Dog fåes en fejl hvis der allerede findes et subdirectory eller en fil med samme navn.

1.5 Hvem kan rode med mine filer?

I princippet kan alle brugere læse hinandens filer. Man kan dog forhindre dette med kommandoen `private` ≪path≫. Herefter kan den angivne fil eller directory kun læses af ejeren. Man kan ophæve denne beskyttelse med kommandoen `public` ≪path≫. Man kan også sende filer til andre brugere ved hjælp af elektronisk post. En nyskabt fil er altid beskyttet. Det er altid kun ejeren, der kan ændre eller slette en fil.

1.6 Eksempler

I hver brugers home directory er der et specielt subdirectory: `eksempler`, hvor der ligger forskellige RASMUS og TRINE eksempler beskrevet i noterne. Man er ikke selv ejer af hverken dette directory eller de filer, som det indeholder. Det er således ikke muligt at ændre dem eller slette dem, og man kan heller ikke oprette nye filer der. Hvis man vil arbejde med eksemplerne, så skal man først kopiere dem over i et andet directory.

1.7 Hvad er en disk-kvote?

Plads er en begrænset ressource. Den samlede længde af en Datalogi 1 studerendes filer må højst være 1000000 tegn, eller 1000Kb. Hvis man overskrider dette over et vist tidsrum, så vil man af UNIX-systemet automatisk modtage en advarsel. Hvis denne ignoreres, så vil ens brugernavn blive spærret. Hvis dette sker, så skal man forklare sig over for driftsstaben, personificeret ved Birger Nielsen (kontor R0.34). Med kommandoen `DQinfo` kan man se, hvor meget plads man bruger.

Der er ikke i forbindelse med kurset nogen risiko for, at man bliver nødt til at overskride sin kvote. Det er ikke nødvendigt at gemme kopier af indholdet af `eksempler`; det vil altid være tilgængeligt der. Bemærk, at udførbare programmer (der mærkes med `*` af `ls` kommandoen) let kan blive meget store. De bør derfor med mellemrum fjernes, for de kan jo altid genskabes udfra TRINE-programmerne. De udførbare filer der hører til TRINE-programmet `program.tri` fjernes på en gang med kommandoen `trirm program`.

2 Udskrifter og printere

2.1 Hvad kan man skrive ud?

Man kan udskrive to slags ting: tekstfiler og grafikfiler. Tekstfiler kan være TRINE programmer, RASMUS værdier, og meget andet. Grafikfiler laves fra kørsler af TRINE programmer ved hjælp af `dumpgraphics` kommandoen. Man skal ikke prøve at udskrive udførbare filer (der mærkes med `*` af `ls` kommandoen); det fører kun til kostbart papirspild. Man udskriver en tekstfil med path ≪path≫ ved at bruge kommandoen

`printascii` ≪path≫ (eller kort `pa` ≪path≫)

Man udskriver en grafikfil med path ≪path≫ ved at bruge kommandoen

`printgraphics` ≪path≫ (eller kort `pg` ≪path≫)

Når man har sendt en fil til udskrift, så anbringes den i en kø ved den pågældende printer. Man kan undersøge status af denne kø med kommandoen

`printqueue` (eller kort `pq`)

Man kan på denne måde også se, hvor lang køen er, *inden* man sender en fil afsted.

2.2 Hvilke printere er der?

Der er flere tilgængelige printere. Man vælger en *aktuel* printer med kommandoen

`printer` ≪printernavn≫

Alle printkommandoer er rettet mod denne aktuelle printer. Følgende tabel fortæller om printere på DAIMI.

Navn	Placering
necs9	Sk.41
nec9	Sk.16
alw9	Sk.25
alw0	R0 ved trappen

Som standard er alw9 den aktuelle printer.

2.3 Hvad kan der gå galt?

Printere kan mangle papir, toner eller farvebånd, og papiret kan sætte sig fast. Desuden kan deres styreprogram gå i stå. Man kan selv fylde mere papir i; ved alle andre problemer skal man sende elektronisk post til programmørvagten med brugernavnet `pvagt`.

Hvis en fil ikke kan blive printet, eller man blot har fortrudt, at den blev sendt afsted, så kan man fjerne den igen med kommandoen

`printremove` (eller kort `prm`)

Når man tager sine udskrifter fra papirbakken, så skal man være forsigtig med ikke at tage (dele af) andre folks udskrifter med ved samme lejlighed. Alt ligger i den samme bunke, så kig lige efter, at du har taget det rigtige.

2.4 Hvem skal nu betale?

Som nævnt koster laserprint penge. Til hvert brugernavn er der en *printerkonto*, hvor disse beløb hæves. Man kan således kun lave laserprint, når der er penge på kontoen. Man fylder penge på sin konto ved hjælp af et FinCard, der købes i Informationkontoret ved Vandrehallen. Man kan derefter overføre penge til sin printerkonto ved hjælp af en kortlæser, der står på R0 ved trappen. Følg den angivne vejledning. Man kan få et kontoudtog med kommandoen `printaccount`.

3 X-vinduer

3.1 Hvad er et X-vindue?

På maskinerne kører vinduessystemet X. Et *X-vindue* er en lille skærm i sig selv. Man kan have mange X-vinduer igang på samme tid, svarende til at man kører mange programmer på een gang. Kun et enkelt X-vindue er *aktivt*, hvilket vil sige, at man kan skrive i det med tastaturet. Et X-vindue gøres aktivt ved, at man anbringer musepilen i det. Programmerne i de ikke-aktive vinduer vil dog køre videre, uanset om deres vinduer er aktive. X-vinduer skabes af programmer, som fx EMACS og oversatte TRINE programmer.

3.2 Hvad kan man med et X-vindue?

Et X-vindue har øverst et *titelfelt*, der f.eks. kan se således ud:

Fra venstre mod højre indeholder titelfeltet følgende:

- En X-knap: **X**
- Programmets navn
- Et aktivt felt, der er gråt, når vinduet er aktivt, og ellers er hvidt
- Fire knapper med udseende: , , og

Hvis man klikker med musen på *menu-knappen*  , så fås en menu med følgende indgange

- *Refresh Window*: vinduet gentegnes
- *(De)Iconify*: et vindue kan "pakkes sammen" og blot vises som en lille såkaldt *icon* i skærmens øverste højre; vinduet pakkes også ud igen med denne indgang
- *Move Window*: vinduet hænger fast i musepilen og kan flyttes på skærmen; det sættes ned med et museklik
- *Zoom Window*: vinduet gives maksimal højde
- *FullZoom Window*: vinduet gives maksimal højde og bredde
- *Resize Window*: vinduet kan gives en vilkårlig størrelse; man vælger med musen en kant, der skal flyttes, og sætter den på plads med et museklik
- *Raise Window*: vinduet anbringes over eventuelle overlappende vinduer
- *Lower Window*: vinduet anbringes under eventuelle overlappende vinduer
- *Destroy Window*: vinduet (og dets program) lukkes ned

Man kan skyde genvej til visse af disse handlinger, ved at klikke med musen i det aktive felt. Et klik med musens venstre knap udfører *Raise Window*, et klik med den midterste knap starter *Move Window*, et klik på  svarer til *(De)Iconify*, et klik på svarer til *FullZoom Window*, et klik på  svarer til *Zoom Window*, og et klik på  svarer til *Resize Window*.

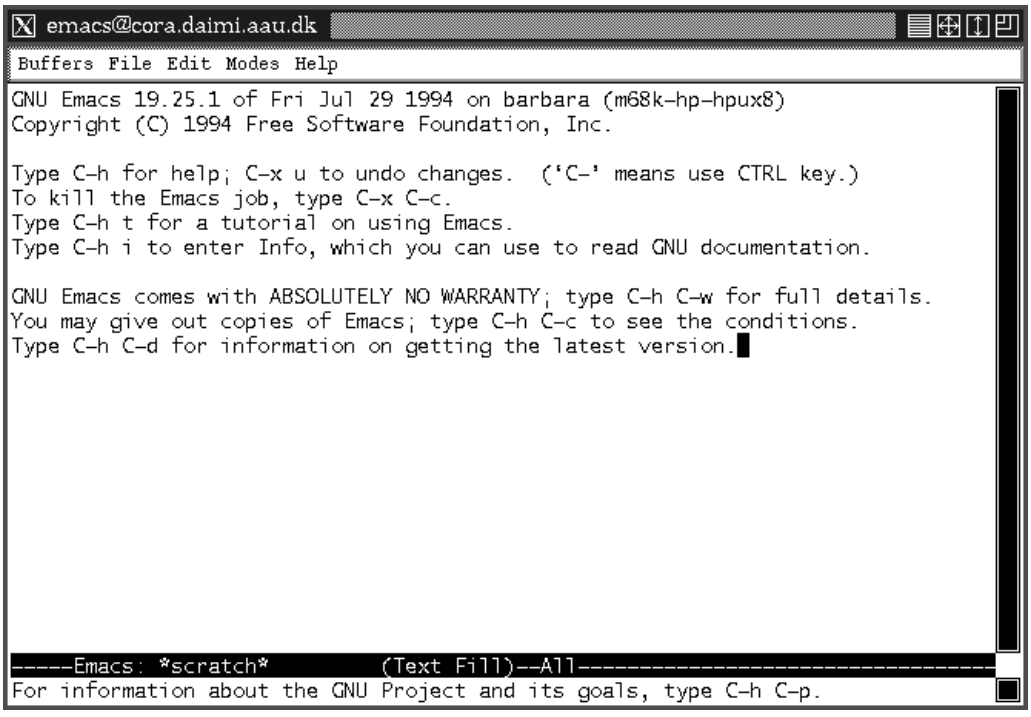
4 Emacs

4.1 Hvad er Emacs?

Emacs er DAIMI's standard editor, altså et program der kan bruges til at redigere filer. Emacs er et meget avanceret program, med en meget bred vifte af funktioner. Dog er der lagt en del begrænsninger ind i den opsætning, som bruges på Datalogi 1. Dette er gjort for at gøre det mere overskueligt og sikkert at lære Emacs at kende.

4.2 Det basale

Emacs startes automatisk og vil præsentere sig som et X-vindue, der minder meget om nedenstående figur:



Dette X-vindue kaldes i Emacs terminologi for en *frame*. Man kan bemærke 4 vigtige detaljer. Øverst er der en *menubar* med en række *pulldown-menuer* (`Buffers`, `File`, `Edit`, `Modes` og `Help`) med kommandoer. De to nederste linier optages af *status-linien* og *mini-bufferen*. Endelig er der en firkantet blok, kaldet *cursor* eller *point*, som angiver hvor tekst vil blive indsat.

EMACS har ikke nogen øvre grænse for hvor meget tekst må fylde, hverken i højden eller i bredden. Derfor vil en del af teksten oftest være usynlig. For at gøre usynlig tekst synlig må man flytte cursoren. Dette er nærmere i afsnittet om navigation (afsnit 4.6).

Menuer bruges ved at man placerer musen oven på den relevante titel i menubaren, hvorefter man trykker på den venstre museknap. Herved kommer selve menuen til syne. Så kan man trække musen nedover menuen (*pull down*) mens man holder den venstre knap nede. Når man har fundet den rigtige indgang, slipper man blot den venstre knap, hvorved den valgte kommando udføres. Hvis man helt fortryder sit forehavende, trækkes musen blot uden for menuen og knappen slippes. Herefter forsvinder den, uden at man har udført noget.

Nogle menuer har undermenuer, hvilket kan ses af en lille pil til højre for menu-indgangen. Når musen når ned til en sådan undermenu, vil den nye menu dukke op ved siden af hovedmenuen, og man kan nu trække musen videre hen over indgangene i undermenuen.

Visse kommandoer (f.eks. cut-and-paste operationerne i `Edit` menuen) opererer på områder af tekst, en *region*, og behøver derfor et sådant område udpeget inden kommandoen udføres. Dette gøres ved at placere musen oven på det tegn der starter (eller slutter) regionen, trykke venstre museknap ned og trække musen hen til det tegn der slutter regionen. Herved bliver regionen grå og man kan nu udfører sin kommando. Hvis man udvælger en region og herefter udfører en kommando, der *ikke* bruger regionen, vil regionen blive *inaktiv* og det grå område forsvinder.

4.3 Buffers

Al tekst i EMACS tilhører en *buffer*. Hvis man ønsker at redigere i en fil, må man først hente den ind i en buffer (se afsn. om filer nedenfor).

EMACS kan have mange buffere og kan dermed også redigere flere filer på en gang. Men der er altid kun én *aktiv* buffer i en frame. Ønsker man at redigere i en anden buffer end den aktive, må man bruge `Buffers` menuen til at gøre den relevante buffer aktiv. `Buffers` har to undermenuer: `Buffers` og `Frames` som er en liste over henholdsvis alle buffere og alle frames som

EMACS bestyrer. Ved at køre musen hen på den ønskede buffer i `Buffers` undermenuen, får man den gjort aktiv.

En forudsætning for alt dette er at man kan kende forskel på de forskellige buffere. Dertil bruger man bufferens navn. En buffer har *altid* et navn, som også fremgår af status-linien (helt til venstre). Vores EMACS-frame viser altså en buffer der hedder `*scratch*`.

Minibufferen (altså den nederste linie i EMACS framen) er en særlig buffer som EMACS bruger, når den skal have yderligere oplysninger.

4.4 Filer

Hvis man ønsker at arbejde på en fil, vælger man `Open file...` i `File` menuen. EMACS vil nu bede dig indtaste et filnavn i minibufferen. Hvis man taster navnet på en eksisterende fil, vil denne fil nu dukke op i en buffer, der vil hedde det samme som filen. Hvis filen ikke eksisterede i forvejen vil man stadigvæk få en (tom) buffer svarende til den nye fil (og med samme navn), men bemærk: filen eksisterer ikke på disken endnu! Det vil den først gøre når man gemmer den igen. Brug `Save Buffer` fra `File menuen` eller `Save Buffer As...` hvis man ønsker at gemme den under et andet navn.

Når man gemmer en ny version af en fil, vil emacs gemme en kopi af den gamle version i en *backup-fil*. Backup-filens navn fåes ved at hæfte tegnet ~ efter den rigtige fils navn.

4.5 Modes

En buffer har altid én *major mode*, og denne vil fremgå imellem parenteserne på status liniens midte, imellem parenteserne. Hvis der står mere end et ord, angiver det første ord major mode. Vores `*scratch*` buffer har altså major mode `Text`.

En mode begrænser ikke, hvad man kan gøre med en fil, men har primært betydning for, om der er ekstra ting man kan gøre. F.eks. er `text-mode` beregnet for generel text, så der er ikke nogle særlige finesser. `Trine-mode`

derimod er beregnet til redigering af TRINE programmer. F.eks. er betydningen af `TAB` tasten ændret fra almindelig tabulering som på en skrivemaskine til korrekt indentering af en TRINE program linie (mere herom i afsnit 5.12).

En buffer kan også have en eller flere *minor modes*. Det vil så blive angivet på status-linien ved ekstra angivelser imellem de to parenteser. Vores `*scratch*` buffer har altså yderligere en minor mode der er angivet ved `Fill`, eller som den egentlig hedder: `auto-fill-mode`.

Dette er en af de mest nyttige minor modes. Dens mission her i livet er at sørge for, at linier ikke bliver for lange. Hver gang man taster et mellemrum, vil `auto-fill-mode` checke liniens nuværende længde, og hvis den er nået ud over en fast grænse (70 tegn), indsættes et linieskift.

Et andet eksempel på en minor mode kunne være `overwrite-mode`. Hvis man taster `Ins` vil man se at der står `Ovwrt` i status-linien, og man har nu slået `overwrite-mode` til. Som navnet antyder, medfører det, at det der skrives vil erstatte den tekst der findes i forvejen, fremfor at indsætte tegn og rykke den eksisterende tekst fremad.

De her nævnte modes er dog kun et lille udpluk af det meget store udvalg der findes, både af major og minor modes, hver tilpasset en speciel situation eller filtype.

Major modes vælges automatisk på grundlag af en fils *type* og er derfor slået til, når en fil er blevet hentet ind i en buffer. Minor modes skal som regel slåes til og fra eksplicit. Dog vil visse major modes automatisk slå en bestemt minor mode til, f.eks. vil en `text-mode` buffer altid starte med at have `auto-fill-mode` slået til, modsat en `trine-mode` buffer.

En buffers modes kan manipulers eksplicit via `Modes` menuen, med den forskel mellem major og minor mode, at en major mode vil *erstatte* den gældende major mode, mens en minor mode vil *skifte* status på en eksisterende minor mode, altså slå den fra, hvis den er slået til og omvendt.

4.6 Navigation

Man kan bevæge sig i en buffer ved hjælp af piletasterne som normalt. Et tryk på en piletast vil flytte cursoren et tegn eller en linie. Man kan bruge `select` tasten hvis man ønsker at flytte sig et større stykke:

↑	en linie op
select ↑	en side op
select select ↑	til bufferens første linie
↓	en linie ned
select ↓	en side ned
select select ↓	til bufferens sidste linie
→	et tegn til højre
select →	et ord til højre
select select →	til liniens slutning
←	et tegn til venstre
select ←	et ord til venstre
select select ←	til liniens begyndelse

Med en side forståes tekst svarende til højden af vinduet.

Man kan også bruge musen til at flytte sig, ikke bare op og ned men også til siderne.

Hvis man placerer musen et sted i en buffer og trykke på den venstre knap, vil cursoren blive flyttet derhen.

For at køre op eller ned med sin tekst, kan man bruge den *scroll-bar* der sidder på højre side af X-vinduet. Man kan enten klikke med den højre (for ned) eller den venstre (for op) museknap. Man kan også "tage fat" i det sorte felt ved at placere musen i scroll-baren og så trykke den midterste museknap ned. Så længe man holder den nede, kan trække op og ned i teksten.

Status-linien indikerer hvor man befinder sig i højde-retningen. Til højre

på linien ses enten en procentangivelse eller ordene `Top` (hvis man er ved bufferens start) eller `Bot` (hvis man er ved bufferens slutning) eller `All` hvis al tekst er vist.

Hvis en linie er for lang i bredde-retningen, indikeres det med et `$` tegn, der hvor linien løber ud over kanten. Man kan så bruge status-linien til at gå frem og tilbage. Hvis man klikker med den højre museknap på henholdsvis den venstre eller den højre side af statuslinien, vil den tekst der går ud over den pågældende kant, komme til syne.

4.7 Danske tegn

Et særligt problem vedrører hvorledes man kan indsætte danske tegn, når man nu kun har et amerikansk tastatur.

Til dette formål benyttes gruppen med 4 funktions taster der er placeret øverst til højre på tastaturet. De tre første vil indsætte henholdsvis `æ`, `ø` og `å`. Nu ligger disse taster ret langt væk fra de øvrige taster, så det kunne være rart hvis man kunne lave sit tastatur om, så man fik de danske tegn til at ligge sammen med resten. Det kan man ved at taste på den 4. funktionstast, altså den helt ude til højre. Herved aktiveres `dk-mode` som vil gøre `[`,`]` og `\` til henholdsvis `æ`, `ø` og `å`. Hvis man i `dk-mode` ønsker at indsætte det oprindelige tegn, f.eks. et `[`, kan det ske ved taste to gange på `æ`.

`Dk-mode` slåes fra igen ved endnu et tast på den 4. funktionstast, eller ved at vælge `dk-mode` i `Modes` menuen. Herved vender tastaturet tilbage til sin amerikanske verdensopfattelse.

5 Trine

5.1 Hvordan oversætter man et program?

Et TRINE program kan komponeres ved hjælp af EMACS og gemmes på en fil. Filer med TRINE programmer bør altid have extension `tri`. Programmet er dog på dette tidspunkt blot en tekstfil. For at programmet skal kunne køres, skal det først oversættes til instruktioner som maskinen kan forstå. Det sker med TRINE oversætteren, der genererer en udførbar fil. Et program skal således kun oversættes en enkelt gang for at kunne udføres mange gange. Antag, at programmet ligger på filen `program.tri`; så startes oversætteren med UNIX-kommandoen

```
trine program
```

Dette producerer en udførbar fil med navn `program` (uden extension). Programmet kan nu gentagent udføres med kommandoen `program`. Under oversættelsen vil man se et skærmbillede i stil med det følgende

```
 TRINE Compiler UNIX Ver.3 Rel.0 DAIMI 1993

* Source file: program

* Syntax analysis
   Including file System.tri
* Symbol analysis
* Type analysis
* Code generation
* Code compilation
  ... (1)(2)(3)(4)
```

De mange linjer skrives hovedsageligt for at underholde brugeren.

Foruden filen `program` får man for hver proces med navn `P` en udførbar fil med navn `program.P`. Dette er en hjælpefil, som man ikke direkte skal benytte. Hvis programmet kun indeholder en enkelt, unavngiven proces, så hedder hjælpefilen `program.program`. Hvis oversætteren startes med

```
trine -c program
```

så skelner den ikke mellem små og store bogstaver i programmet.

5.2 Hvad kan man oversætte?

Man kan oversætte sætninger, processer, systemer og samlinger af erklæringer. Man kan både oversætte og udføre programmer med ubestemte stumper, der ikke er færdigdefinerede. Hvis man under udførelsen rammer en udefineret ubestemt stump, så må programmet dog naturligvis give op.

5.3 Hvad hvis programmet indeholder fejl?

Et TRINE program skal overholde mange forskellige regler for at være lovligt. Hvis dette ikke er tilfældet, så vil oversætteren generere *fejlmeddelelser* på en fil med navn `program.lst`. En oversættelse med fejl kan se ud som følger

```
 TRINE Compiler UNIX Ver.3 Rel.0 DAIMI 1993

* Source file: fejl

* Syntax analysis
   Including file System.tri
* Symbol analysis
* The program contains 3 errors
   Writing file fejl.lst
```

Bagefter har filen `fejl.lst` dette indhold

```
  7     xy:=87
*****    ^91
*  91: Variable name not defined

  8     pip(xyz)
*****     ^93
*  93: Proc name not defined

  9     write(j)
*****         ^91
*  91: Variable name not defined
```

Oversætteren forsøger at give hjælpsomme fejlmeddelelser, men hvis programmet indeholder mange fejl, så kan den let blive forvirret. Det kan derfor anbefales, at man løbende oversætter det delvist færdige program, så man ikke til sidst får alt for mange fejl på een gang.

5.4 Hvordan inkluderer man filer?

Alle steder i et TRINE program kan man *inkludere* indholdet af en anden fil ved at skrive

`@"`≪path≫`"`

Filen angivet af ≪path≫ skal naturligvis eksistere. Man må gerne have flere niveauer af inkluderede filer, men man skal undgå cykler.

Hvis oversætteren ikke kan finde en inkluderet fil, der er angivet kun med navn og uden path, så leder den i et bestemt system directory. Her finder man blandt andre filerne `System.tri`, `graphics.tri`, `sequence.tri` og `alphanum.tri`.

5.5 Hvordan kører man et program?

Når `program.tri` er oversat, så har man en udførbar fil med navn `program`. Når UNIX-kommandoen `program` afgives, så vil UNIX-systemet starte et

X-vindue for hver proces i programmet. Hvis programmet benytter grafik, så startes der også et specielt grafikvindue. Man kan skrive inddata til en given proces ved først at gøre dens X-vindue aktivt. Når en proces er termineret, så venter vinduet på, at man skriver `Return`; derefter lukkes det. Hvis man er utilfreds med et programs opførsel, fx fordi det ikke terminerer, så kan man afbryde det i utide ved i dets vindue at holde `Control` eller `CTRL` tasten nede og taste `C` i processens vindue.

5.6 Hvad hvis programmet giver fejl?

Et program kan give fejl under udførslen, fx hvis man prøver at dividere med nul. Hvis det sker, så vil programmet standse med en fejldiagnose. Man får en kort beskrivelse af fejlen, en udskrift af den skyldige linje i programmet, samt en liste af de aktive procedurekald.

5.7 Hvordan dokumenterer man en kørsel?

Ved at starte et oversat program med

```
program -l
```

får man for hver proces en kopi af al skærmkommunikation. Filen for processen `P` i programmet `program` hedder `program.P.log`. Hvis man kun har en enkelt, unavngiven proces, så hedder filen `program.program.log`. Filerne indeholder tekster og kan udskrives med kommandoer `pa`.

5.8 Hvad er pladsgenbrug?

Oversatte programmer vil normalt frådse med pladsen på maskinen og blot håbe, at det går godt. Det er imidlertid et problem, hvis man derved bruger mere plads, end man har til rådighed. Hvis man starter oversætteren som

```
trine +g program
```

så vil det oversatte program omhyggeligt søge at genbruge pladsen.

5.9 Hvad er trace?

Under udførelsen af et TRINE program kan man på skærmen følge udvalgte procedurekald og variabler. Af hensyn til fejlfinding eller af almindelig nysgerrighed kan det være interessant at følge sådanne detaljer.

Oversætteren accepterer forekomster af nøgleordet `trace` efter nøgleordet `Proc` i procedurer

```
Proc trace Search ···
```

og sætninger af formen

```
trace[v1,v2,...,vn](t)
```

hvor $\texttt{v}_1\texttt{,v}_2\texttt{,...,v}_n$ er variabelnavne og `t` er et udtryk af type `Text`; begge slags parametre er iøvrigt valgfrie. Disse `trace` konstruktioner vil normalt blive ignoreret af oversætteren, men hvis den startes som

```
trine +t program
```

så får de effekt.

Hver gang en procedure mærket med `trace` bliver kaldt, så vil den på skærmen skrive hvor i programmet, der er blevet kaldt fra, samt hvilke værdier dens parametre har. Når proceduren afsluttes, bemærkes dette også, og værdierne af variabelparametrene udskrives. Efter enhver sådan udskrift standser programudførelsen, indtil man trykker på `Return`.

Et eksempel er følgende program

```
(+ Proc trace Search[v: Vector,f: Bool](x: Int)
     (+ var i: Int
        i,f:=0,false
        do i < |v| ->
           trace[i,f]("inside do")
           i,f:=i+1,f or (v.(i)=x)
        od
     +)
   end Search

   var w: Vector
   var b: Bool
   w:=Vector(17,9,14)
   Search[w,b](9)
+)
```

hvis udførelse giver anledning til denne udskrift

```
TRACE: Proc Search called from line 14
[] v = LIST(17,9,14)
[] f = ?-BOOL
() x = 9

TRACE: inside do, line 5
i = 0
f = FALSE

TRACE: inside do, line 5
i = 1
f = FALSE

TRACE: inside do, line 5
i = 2
f = TRUE

TRACE: Proc Search terminated
[] v = LIST(17,9,14)
[] f = TRUE
```

Variabelparametre er ved procedurekald angivet med `[]` og værdiparametre med `()`.

5.10 Hvad er indrykning?

Et korrekt og fungerende TRINE program kan nemt være en præsentationsmæssig fiasko. Et eksempel er følgende program

```
PROCess gcd (+ var x,
y: INt reAd[x,
    y] dO x >
y->x  :=x -y|y >x->y
:=y-x Od write(
x)+) EnD
gcd
```

der næppe kan betegnes som læseligt. Hvis programmet ligger på filen `program.tri`, så vil UNIX-kommandoen

```
indent program
```

indrykke programmet, og dermed ændre indholdet til

```
Process gcd
  (+ Var x,y: Int
     read[x,y]
     do x > y -> x:=x-y
      | y > x -> y:=y-x
     od
     write(x)
  +)
end gcd
```

Dette sker kun, hvis programmet ligner et TRINE program tilstrækkeligt. I modsat fald genereres en fil `program.lst` med en meddelelse om den første afvigelse.

Hvis indrykningen gik godt, så vil der også blive foretaget en analyse af de navne, der optræder i programmet. Man får en liste over alle de navne, der bliver brugt men ikke defineret noget sted. Hvis det er muligt, gives der også et forslag til, hvordan fejlen kan rettes. Af denne grund er det en god ide altid at indrykke et program, inden man forsøger at oversætte det. Programmet

```
(+ Proc Pip(i: Int)
     write(i,eol)
   end Pip

   Var xyz: Int

   xy:=87
   pip(xyz)
   write(j)
+)
```

giver anledning til dette skærmbillede

```
 TRINE Indenter UNIX Ver.3 Rel.0 DAIMI 1993

* Source file: fejl

  Analyzing identifiers

  Identifiers used but never defined:  Possible corrections:

  [j.............] at line 9
  [pip...........] at line 8          [Pip...........] at line 1
  [xy............] at line 7          [xyz...........] at line 5

* Indention complete
```

5.11 Hvad er ekspansion?

Et TRINE program med ubestemte stumper kan oversættes som det er. Man har dog også mulighed for at *ekspandere* disse definitioner med UNIX-kommandoen

```
expand program
```

Det gamle program vil blive gemt på en fil med samme navn som det oprindelige forlænget med tilstrækkeligt mange `~` tegn til at danne et nyt filnavn. Man skal kun forsøge at ekspandere et program, der kan accepteres af oversætteren.

5.12 Trine-mode i Emacs

Hvis man i EMACS har en fil, der slutter på `.tri`, så findes der en ekstra menu-indgang ved navn `Trine`. Den har fire indgange:

- `compile buffer`: gemmer og oversætter indholdet.
- `indent buffer`: indrykker indholdet.
- `next error`: viser cyklisk den næste fejl i indholdet.
- `first error`: hopper tilbage til den første fejl i indholdet.

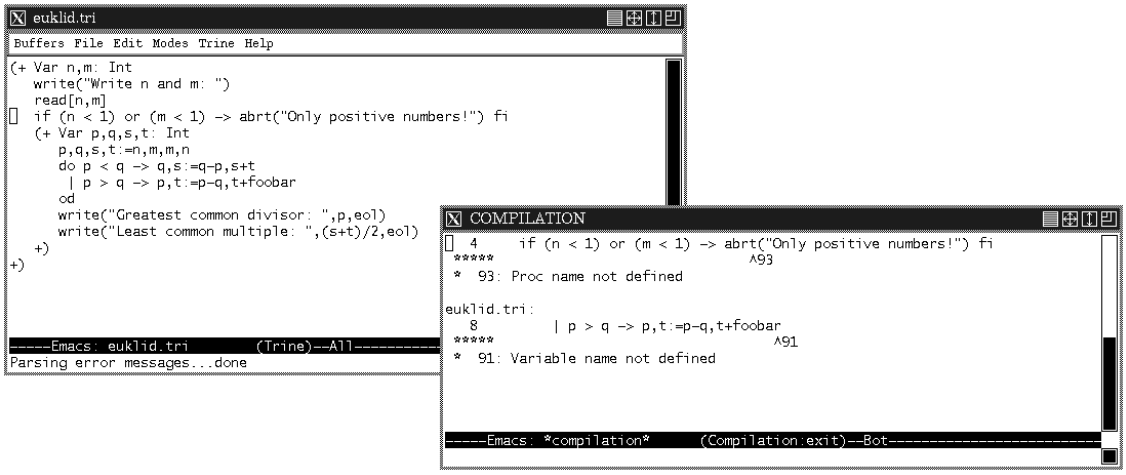
Denne menu kan lette arbejdsgangen en smule.

Når man vælger `compile buffer` vil EMACS prøve at oversætte den fil der hører til den aktive buffer. Først vil EMACS præsentere den nødvendige UNIX-kommando i minibufferen, så man har en mulighed for at angive direktiver til oversætteren. Når man er tilfreds med kommandoen, tastes `Return` og oversættelsen går igang.

EMACS vil åbne en frame (kaldet `COMPILATION`), som løbende viser output fra oversætteren (i buffer `*compilation*`), således at man kan følge med i hvordan det går.

Hvis programmet indeholder fejl, vil disse blive listet i `*compilation*` bufferen, efter oversættelsen er overstået. Man kan så bruge `next error` til at undersøge de enkelte fejl. Når man har valgt `next error` vil man se cursoren i `*compilation*` bufferen blive placeret ved den næste fejlmeddelelse, mens cursoren i filbufferen rykkes hen til den linie som oversætteren fandt fejlen i.

Følgende figur angiver en typisk situation. Bufferen `euklid.tri` er blevet oversat, og oversætteren fandt nogle fejl som kan ses i `*compilation*` bufferen. Efter at have valgt `next error` i `Trine` menuen, kan man på figuren se, hvorledes cursoren er placeret i de to frames ved henholdsvis fejlbeskeden og den pågældende linie:



Hvis man vælger `indent buffer` i `Trine` menuen, vil man få udført UNIX-kommandoen `indent` på bufferens fil (som beskrevet i afsnit 5.10). Når `indent` er færdig, udskiftes indholdet i bufferen med det nye indhold af filen. `indent` programmets udgydelser kan følges i `*compilation*` bufferen som ved oversættelse.

Man kan også løbende sørge for at holde sit program korrekt indrykket, nemlig ved hjælp af TAB tasten. Taster man TAB i en buffer, der er i `trine-mode`, vil EMACS justere indrykning af den aktuelle linie, så den er korrekt. Dog skal man være opmærksom på at EMACS ikke betragter hele bufferen, sådan som `indent` programmet gør. Hvis der er linier højere oppe i bufferen der ikke står korrekt, er det ikke sikkert at den aktuelle linie vil komme til at gøre det.

6 Elektronisk post

6.1 Hvad er elektronisk post?

Hvert brugernavn har en tilknyttet *postkasse*, som andre brugere kan sende *elektroniske breve* til. Et elektronisk brev er blot en tekstfil. Det kan bruges til mange nyttige ting, så som: at aftale møder for læsegrupper, at udveksle programmer, at henvende sig til sin instruktor, og at rapportere fejl ved UNIX-systemet. Desuden vil man på denne måde modtage forskellige beskeder om praktiske forhold på kurset.

For at sende et elektronisk brev til en person, så skal man kende dennes brugernavn. Kommandoen

`userid` ≪tekst≫

udskriver den del af en stor tabel over personnavne og brugernavne hvori ≪tekst≫ forekommer. Fx vil `userid fred` udskrive

```
brf         Benny Frederiksen
fred        Fred Mosekjaer
nikki       Nicolette Frederiksen
```

Der er også et antal brugernavne for specielle *grupper* af brugere.

- `pvagt`: programmørvagten; her sender man beskeder om generelle problemer med UNIX-systemet, printere og så videre.
- `dat1-prog`: her sender man beskeder om problemer med de specielle Datalogi 1 programmer, som fx RASMUS og TRINE.
- `dat1-alle`: alle med tilknytning til Datalogi 1; det er over 200 personer, så tænk dig godt om først!
- `dat1-hold-n`: alle på Datalogi 1 øvelseshold nummer `n`; denne liste administreres af den pågældende instruktor.

6.2 Hvordan kan jeg bruge det?

I EMACS `File`-menuen er der to indgange: `read mail` og `send mail`. Hvis man vælger `read mail`, så vil det først ankomne ulæste brev blive det nye indhold af EMACS. Herefter kan man læse brevet, eventuelt ændre i det, eller gemme på en fil – ganske som sædvanligt. Hvis man vælger `send mail`, så vil man blive spurgt om et brugernavn. Herefter vil det aktuelle indhold af EMACS blive sendt som et elektronisk brev til denne bruger. Man kan også angive en liste af brugernavne adskilte med blanke; de vil så alle modtage en kopi af brevet.

I skærmens øverste hjørne kan man altid se en tegning af en postkasse. Hvis den er *fed*, så er der breve i postkassen. Med kommandoen

`finger` ≪bruger≫

kan man (blandt andet) se, hvornår en given bruger sidst har læst sin post.

7 Fejlrapportering

De systemer, der kan benyttes, er under konstant udvikling og mutation. Der kan derfor opstå fejl og uhensigtsmæssigheder i dem. Hvis man tror, at en del af UNIX-systemet ikke virker som det skal, så skal man henvende sig med elektronisk post til brugernavnet `pvagt` (med fejl i det generelle system) eller til brugernavnet `dat1-prog` (med fejl i det specielle Datalogi 1 programmel). En fejlrapport er dog kun til gavn, hvis den indeholder meningsfulde oplysninger. Dette er et eksempel på en *ubrugelig* fejlrapport

```
Det er noget galt, når jeg kører mit program! Det virkede
i går aftes!! Du kan selv prøve!!! Og jeg skal aflevere i
forgårs!!!!!!! Nu bliver jeg aldrig datalog!!!!!!!!!!!!!
```

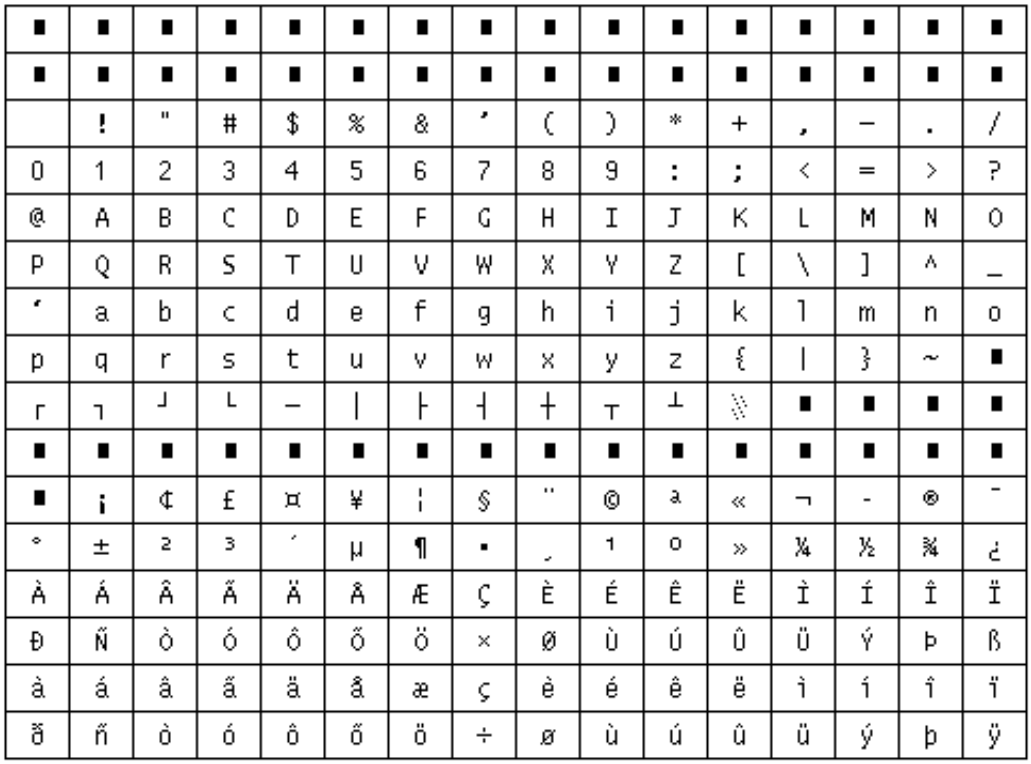
hvorimod dette er et eksempel på en *nyttig* fejlrapport

```
Naar jeg forsøger at oversætte programmet paa filen:
   /users/u960007/afl6.tri
saa faar jeg dette paa skærmen:
    TRINE Compiler UNIX Ver.3 Rel.0 DAIMI 1993

   * Source file: afl6

   * Syntax analysis
      Including file System.tri
   * Symbol analysis
   segmentation fault (core dumped)
og oversætteren stopper. En tidligere version paa filen
   /users/u960007/afl6.old.tri
kan derimod godt oversættes.

                  venlig hilsen
                       Maja
```

Nyttige fejlrapporter vil føre til en hurtig udbedring af den eventuelle fejl, eller i hvert fald til en forklaring på, hvad der foregår.

8 ASCII tabel

Følgende tabel viser de 256 ASCII-tegn. Det øverste venstre hjørne svarer til nummer 0, og det nederste højre hjørne svarer til nummer 255.



9 På egen hånd

Når man starter som dProg1-bruger, så får man en *standardopsætning* af UNIX-systemet, der fungerer som beskrevet ovenfor. Der er utallige muligheder for at lave om på dette, hvilket I sikkert vil blive gjort opmærksomme på af andre studerende. Reglerne for dette er som følger:

- Det er tilladt at ændre på opsætningen, så meget som man har lyst til.
- Afdelingens medarbejdere vil ikke bruge tid på at hjælpe med dette.
- Hvis man piller ved standardopsætningen, så er der rige muligheder for at dumme sig, så væsentlige dele af systemet ikke længere virker efter hensigten.
- Afdelingens medarbejdere vil ikke bruge tid på at afhjælpe sådanne skader.
- Hvis standardopsætningen ikke virker, og man derfor ikke kan overholde en afleveringsfrist, så er det vores ansvar, og passende hensyn vil blive taget.
- Hvis man ændrer ved standardopsætningen og derfor ikke kan overholde en afleveringsfrist, så er det ens eget ansvar.