UMEÅ UNIVERSITET
Institutionen för datavetenskap
Material till Unix-intro för TDV 2001

17 maj 2002

Introduktion till Unix

För tekniska datavetare vid Umeå Universitet

Per Nordlinder, per@cs.umu.se
Marcus Bergner, bergner@cs.umu.se
Jon Hollström, jon@cs.umu.se

Sammanfattning

Vi ska i detta häfte försöka förklara och klargöra de mest grundläggande delarna av den datavetenskapliga institutionens Unix-system. Detta är mest tänkt att hjälpa dig att komma igång samt att användas som en liten repetition av vad som sagts på introduktionen. Ett annat syfte är att hindra den ovane Unix-användaren från att bli avskräckt över något som till en början kan verka lite krångligt och svårt men som med tiden blir både roligt, smidigt och användbart.

Innehåll

★ = Överkursavsnitt

1 Konton, lösenord och hemkataloger

Ett Unix-system bygger till skillnad från t ex DOS-baserade system på att varje användare har ett eget *användarkonto* med ett tillhörande *lösenord*. Detta betyder att det första man måste göra på ett Unix-system är att identifiera sig med sitt användarnamn och lösenord för att sedan kunna börja jobba. Det betyder också att det sista man måste göra innan man lämnar systemet är att logga ut. Detta görs enklast här på skolan genom att trycka på knappen ”Logout”.

Vår institution ger sina tekniska datavetare användarnamn enligt mönstret c[år][första bokstaven i förnamnet][första bokstaven i efternamnet][sista bokstaven i efternamnet]. Detta innebär att exempel-eleven Adam Botovic som börjar på programmet i år skulle få användarnamnet `c01abc`, vilket innebär att hans epost-adress blir `c01abc@cs.umu.se`.

Du kommer att få ett eget användarkonto på institutionens system som kommer att heta något i stil med exemplet ovan. Ditt lösenord kommer att vara en kombination av tecken som systemadministratörerna i förväg har satt på hela c01-gruppen. Det du då bör göra när du loggar in första gången är att *byta detta lösenord* till något specifikt som bara du vet om och kommer ihåg. För att byta lösenord så kör du vid prompten i det terminalfönster som finns på skärmen kommandot `passwd` och följer instruktionerna.

Med din användare följer också en hemkatalog (en katalog som bara är din), där du kan spara dina filer etc. Den katalogen står du normalt i när du tar upp ett nytt terminalfönster. Den fullständiga sökvägen till din hemkatalog är `/home/c01/användarnamn` så vår gode vän Adams filer ligger alltså under `/home/c01/c01abc/`. Du har till att börja med 5 Mb utrymme att spara saker på i din hemkatalog och den ligger alltså på en diskserver (`oxe`) så att du kommer åt den och dina inställningar vilken dator du än sitter vid. En mer detaljerad beskrivning av katalogstruktuen och filsystemet hittar du längre in i detta häfte.

Det grafiska användargränssnittet under Unix heter X och kan liknas vid Windows. Här på CS så används X på samtliga Unix-maskiner.

2 Filsystem, program och tab completion

Filsystemet i Unix är uppbyggt som ett enda stort träd av kataloger och filer. Detta utgör en tydlig skillnad mot t ex DOS/Windowssystem som har olika enheter (A:, C: o s v). *Rotkatalogen* i Unixsystem heter `/`. För att komma till rotkatalogen skriver man i ett terminalfönster `cd /`. I den numera knappt använda DOS-Prompten, som finns i Windows, skriver man som du kanske vet `cd \` för att komma till rotkatalogen. Detta är en skillnad som gäller i *alla lägen*. Då man i Windows använder tecknet \ använder man i Unix `/`. Filnamnen i ett Unix-filsystem kan vara upp till 256 tecken långa.

I ett typiskt Unix-filsystem finns förutom filer och kataloger även så kallade *länkar*. Länkar är helt enkelt genvägar från en plats i systemet till en annan. Du kan t ex ha skapat en länk från din hemkatalog till någon komplicerad sökväg i

systemet. Om du nu vill komma till den komplicerade sökvägen så kan du helt enkelt gå via länken som om det vore en katalog genom att skriva `cd länknamn`.

Några av de De viktigaste katalogerna på systemet ser du nedan:

`~`	=	Din hemkatalog. Den kan du alltid nå genom att bara skriva `cd` och trycka enter.
`~c01abc`	=	Adam Botovic's hemkatalog.
`/home/c01`	=	Här finns alla c01:ors hemkataloger.
`/home`	=	Här finns alla klasser (c01, c98, dva99 o s v.).
`/pkg`	=	Här finns de flesta program som systemet erbjuder. (t ex Netscape, MatLab, Maple m fl)
`/tmp`	=	Här lagras alla dina och alla andras temporära filer.

Det bästa sättet att lära sig hitta i filsystemet är att sätta sig ned vid en dator och titta sig omkring i katalogstrukturen. För att se var du befinner dig så kan du använda kommandot `pwd`. Ett annat bra hjälpmedel är programmet `where`. Om du undrar var ett program finns så kan du skriva `where [programnamn]` och får då (förhoppningsvis) fram sökvägen. När du sedan hittat programmet och skall starta det så är det en bra ide att starta programmet med ett `&`-tecken efter. Det betyder att programmet lägger sig i bakgrunden och du kan fortsätta att jobba med din terminal. I annat fall är terminalen "låst" så länge du har programmet igång. Ex: `>xemacs myfile.c &`.

Ett väldigt användbart hjälpmedel när man skall bläddra runt i filsystemet är en finess som heter *TAB completion*. Detta innebär att du kan skriva en del av t ex ett filnamn eller en katalog och sedan trycka på TAB. När du gör detta så fyller systemet på med resten av namnet. Om systemet inte kan hitta ett entydigt namn så fylls så många tecken som möjligt på utan att tvetydigheter uppstår. Nedan följer ett litet exempel:

I den aktuella katalogen antas följande filer finnas:
`index.html`, `index.html.bak` och `indian.jpg`

Om nu användaren skriver följande i ett terminalfönster:
`>vim in`

och där efter trycker på TAB (innan ENTER) så fyller systemet på till:
`>vim ind`

Om man nu skriver ett `e` så har man:
`>vim inde`

Om man nu trycker på TAB så fyller systemet på till:
`>vim index.html`

Vi rekommenderar att du lär dig använda denna funktion, den kommer bespara dig många tangenttryckningar. En annan tidsbesparande finess är kommandohistoriken. Om du vill upprepa ett kommando du nyss kört kan uppåtpil användas för att bläddra fram nyligen givna kommandon.

De finns ett par servers på institutionen vilka kan vara bra att känna till. **oxe** heter diskservern där din hemkatalog och alla andras hemkataloger finns. **peppar** heter en kraftfull server som du kan logga in på via `ssh` om du t ex vill kolla din e-post hemifrån.

Samtliga servers, och datorer också för den delen, har fullständiga namn och kan alla nås utifrån. De har namn enligt mönstret `datornamn.cs.umu.se`, där datornamn t ex kan vara `peppar` eller `madicken`.

3 De vanligaste kommandona i Unix

Här nedan har vi gjort en sammanställning av de allra viktigaste kommandona i ett Unix-system, vad de har för parametrar samt deras eventuella motsvarighet i DOS/Windows.

3.1 Konventioner i detta avsnitt

I kommande avsnitt presenteras flera kommandon som kan köras i ett terminalfönster. Kommandona kommer att presenteras på ett sätt som ser ut ungefär så här:

```
mv fil fil
mv fil... katalog
```

Först anges kommandonamnet, `mv`, följt av ett antal parametrar, `källa` och `mål`. Parametrar kan vara av ett antal olika typer varav de vanligaste visas nedan.

`[-flaggor]`	Noll eller flera flaggor kan anges. Föregås av ett `-`.
fil	Ett filnamn som måste anges.
[*fil*]	Ett filnamn som kan, men inte måste, anges.
fil...	Ett eller flera filer anges som parametrar.
[*fil...*]	Noll eller flera filer kan anges som parametrar.
{a|b|c}	Antingen `a`, `b` eller `c` måste anges.
[a|b|c]	Antingen `a`, `b`, eller `c` kan anges.

Sammanfattningsvis gäller att parametrar som beskrivs som omslutna i hakparenteser behöver ej anges. Till de flesta kommandon som tas upp i kommande avsnitt kan en mera utförlig beskrivning hittas m h a `man kommando`. Då det gäller flaggor så kan dessa *oftast* anges på två olika sätt. T ex så utför följande kommandon samma sak.

```
ls -al *.txt
ls -la *.txt
ls -a -l *.txt
ls -l -a *.txt
```

I samtliga fall så visas alla filer som slutar med `.txt` i en lång lista där även dolda filer visas, som slutar på `.txt` alltså. Kommandot `ls` beskrivs i avsnittet *Navigationskommandon*.

3.2 Få hjälp om kommandon

Det är ofta fallet, speciellt för nybörjare, att man vet vad man vill göra men inte hur det ska göras. För att lösa det problemet så finns det ett par kommandon (se Tabell 1) som underlättar.

Namn	Anropssätt	Beskrivning
`man`	`man` *namn...*	Visar hjälp om kommandon
`apropos`	`apropos` *nyckelord...*	Söker kommandon som har med nyckelordet att göra

Tabell 1: Hjälpkommandon

Se även: `man {man|apropos}`

Det finns även en grafisk variant av `man` som heter `xman`. Den kan användas om man så önskar. Den kan dock vara krångligare än vanliga `man` om man redan vet vilket kommando man vill ha hjälp om, men något enklare om man inte vet det. Indelningen i kategorier för `xman` är dock inte så väldigt enkel. Om man vet vad kommandot man vill ha hjälp om gör för något så kan man istället använda `apropos` för att hitta rätt. T ex så kan man skriva `apropos shell` vid en prompt och får då upp t ex `sh`, `csh` och `tcsh`, som är olika kommandoskal. (Ett kommandoskal är det program som körs i terminalfönstret och tolkar det du skriver osv).

3.3 Navigation i filsystemet

För att ha någon nytta av ett filsystem, oavsett om det är ett Unix system eller inte, så måste man kunna navigera i filsystemet. Det rör sig då om att byta aktuell katalog, visa innehållet i en katalog och ta reda på var man befinner sig i filsystemet. I Tabell 2 visas de kommandon som används för detta syfte i Unix.

Namn	Anropssätt	Beskrivning	Flaggor
`cd`	`cd` *katalog*	Byter aktuell katalog	
`pwd`	`pwd`	Skriver ut aktuell katalog.	
`ls`	`ls` [*-flaggor*] [*fil...*]	Listar filer. Om inga filer/kataloger anges visas aktuell katalog.	`-a` *All.* Visa alla filer, även gömda, dvs sådana som börjar med `.` `-d` *Directory.* Om en katalog anges så lista endast katalogen, ej innehållet. `-F` *Family.* Markera kataloger med `/`, körbara filer med `*` mm. `-l` *Long.* Skriv en lång lista med mycket information. Innehåller ägare, storlek, rättigheter mm. `-R` *Recursive.* Skriv ut innehållet i underliggande kataloger. `-p` Markera kataloger med `/`. `-1` Skriv ut 1 filnamn/rad.

Tabell 2: Navigationskommandon

Se även: `man {cd|pwd|ls}`

3.4 Filhantering

Filhantering hör till vardagen för alla datoranvändare. Det kan röra sig om att flytta filer eller skapa kataloger för att organisera diverse filer. Nedan följer en tabell med de vanligaste filhanteringskommandona.

Namn	Anropssätt	Beskrivning	Flaggor
`cp`	`cp` [*-flaggor*] *källa mål*	Kopierar filer	`-r` *Recursive*. Dvs alla underliggande filer och kataloger.
`mv`	`mv` [*-flaggor*] *källa mål*	Flyttar och döper om filer	
`rm`	`rm` [*-flaggor*] *fil* [*fil...*]	Tar bort filer	`-r` *Recursive*. Se ovan `-f` *Force*. Ta bort filer utan meddelanden, även om de är skrivskyddade. `-i` *Interactive*. Bekräfta borttagning.
`mkdir`	`mkdir` *katalog*	Skapar kataloger	
`rmdir`	`rmdir` *katalog*	Tar bort *tomma* kataloger	

Tabell 3: Filhanteringskommandon

Se även: `man {cp|mv|rm|mkdir|rmdir}`

3.5 Länkar och lite övrigt

På Unix-system finns det förutom vanliga filer och kataloger en del andra saker i filsystemet. En av dessa är länkar. En länk är *en genväg från ett ställe till ett annat* i systemet. Det finns två typer av länkar, hårda länkar och mjuka länkar, som i bland kallas symboliska länkar. För att skapa båda typerna av länkar används kommandot `ln`.

Två kommandon som inte direkt passade in i någon av de kategorier som kommandona delats upp i är `clear` som rensar terminalfönstret och `echo` som kan användas för att skriva ut text till terminalen, eller till filer. En vanlig användning av `echo` är att se hur parametrar kommer att tolkas. T ex så skriver `echo *.[Tt]xt` ut namnet på alla filer som slutar på `.txt` eller `.Txt`. Just detta kan naturligtvis åstadkommas med `ls` också, men `echo` kan användas i andra sammanhang. Om man t ex skriver `echo "\033[7m"`, vilket ser ganska underligt ut, så kommer text och bakgrund att växla färg med varandra (dvs texten ser ut att vara markerad). Kul och meningslöst! Detta är ett exempel på en speciell styrkod som terminalen förstår. Det finns massor av sådana koder, men de är bara de mest inbitna Unix-hackarna som använder sig av dem. En av de vanligaste användningsområdena är för dessa koder är att *göra en personlig prompt*. Detta tas upp i överkursavsnittet *Konfiguration*.

Se även: `man {ln|clear|echo}`

Namn	Anropssätt	Beskrivning	Flaggor
`ln`	`ln` [*-flaggor*] *källa mål*	Skapar en länk från källa till mål	`-s` *Soft.* Skapa en mjuk länk
`clear`	`clear`	Rensar terminalen	
`echo`	`echo` [*-flaggor*] [*text...*]	Skriver ut text	`-n` Skriv ej linefeed

Tabell 4: Skapa länkar och två andra vanliga kommandon

Endast mjuka länkar kan skapas till kataloger. Om man står i sin hemkatalog så kan man t ex skapa en genväg (länk) till katalogen där de flesta programmen ligger genom kommandot `ln -s /pkg program`. Nu har man en länk i sin hemkatalog som heter `program` och man använder länken som om det vore en katalog, genom att skriva `cd program`. Ett litet problem med mjuka länkar är att om filen/katalogen som de länkar till tas bort så kan man stöta på en del konstiga felmeddelanden. För att ta bort, kopiera, flytta eller döpa om en länk så görs detta som om det vore en vanlig fil, d v s med kommandona `rm`, `cp` och `mv`. Observera att detta gäller även om länken avser en katalog.

3.6 Arbeta med filer

Att arbeta med filer på något sätt är vanligt förekommande i Unix-världen. Gemensamt för de olika Unix-varianterna är att alla innehåller ett stort antal små kommandon som gör olika saker. Ibland kanske man vill göra något som inte ett enskilt kommando klarar av, men då finns det möjlighet att kombinera kommandon.

Här presenteras ett antal kommandon. Vissa av dem kan vid första anblick verka lite meningslösa medan andra är omöjliga att undgå. För samtliga kommandon se även motsvarande `man` sida för detaljer.

I detta avsnitt används förkortningen STDOUT. Denna står för *standard output* och avser vanligtvis skärmen. Det finns även två andra vanliga förkortningar, STDIN, *standard input*, som normalt är kopplad till tangentbordet, och STDERR, *standard error*, som normalt är kopplad till skärmen.

Kommandona nedan skriver till STDOUT, och de flesta kommandon som kan ta noll eller flera filer som parametrar läser oftast från STDIN om inga filer anges. I Tabell 5 visas ett antal kommandon.

Se även: `man` sidan för respektive kommando.

3.7 Kommandon för diskutrymme, lösenord, processer och utskrifter

Precis som namnet på avsnittet säger så tas här diverse systemkommandon upp. De rör utskrifter, diskutrymme, lösenord mm. Ett kommando som man omedelbart *ska* använda när man fått sitt riktiga användarkonto är `passwd` för att byta lösenord. Skriv då bara `passwd` och följ sedan instruktionerna. Man bör, av säkerhetsskäl, ha *olika* lösenord på Unix-systemen och NT-systemen. I

Namn	Anropssätt	Beskrivning	Flaggor
`cat`	`cat` [*-flaggor*] [*fil...*]	Skriver ut filers innehåll.	`-n` Radnumrering
`grep`	`grep` [*-flaggor*] *text* [*fil...*]	Söker efter en text i filer och skriver ut matchande rader.	`-i` *Ignore case* Små och stora bokstäver behandlas lika. `-v` Skriv ut rader som ej matchar
`egrep`	`egrep` [*-flaggor*] *uttryck* [*fil...*]	En kraftfullare `grep`	
`cut`	`cut` -{b|c|f} *lista* [*fil...*]	Klipper ut kolumner ur filer.	
`sort`	`sort` [*-flaggor*] [*fil...*]	Sorterar data i filer.	`-n` *Numeric* Sortera tal `-r` *Reverse* Omvänd ordning
`more`	`more` [*fil...*]	Visar filer en skärm åt gången	
`less`	`less` [*fil...*]	Kraftfullare än `more`.	
`head`	`head` [*-flaggor*] [*fil...*]	Visar början av filer.	`-n` Antal rader som skall visas
`tail`	`tail` [*-flaggor*] [*fil...*]	Visar slutet av filer.	`-n` Se `head`
`wc`	`wc` [*-flaggor*] [*fil...*]	Räknar ord, rader och tecken i filer.	
`spell`	`spell` [*-flaggor*] [*fil...*]	Utför rättstavning på filer.	
`chmod`	`chmod` [*-flaggor*] *mode fil...*	Ändrar filrättigheter.	
`diff`	`diff` [*-flaggor*] *fil fil*	Jämför två filer.	
`tee`	`tee` [*-flaggor*] *fil*	Skickar STDIN till en fil och till STDOUT.	
`where`	`where` *fil*	Snabbsökning efter filer	
`find`	`find` *katalog...* [*-flaggor*] *uttryck*	Söker efter filer	

Tabell 5: Kommandon för att arbeta med filer

Tabell 6 visas ett antal systemkommandon, varav vissa är mycket vanliga.

Se även: `man` sidan för respektive kommando. Vissa av dem saknar dock sådan.

4 Grupper och filrättigheter

Ett Unix-system hanterar förutom användare också *grupper*. En eller flera användare kan vara medlem i en grupp och en användare kan vara medlem i flera grupper, varav en primär grupp. Alla användare är medlem i någon grupp. Som ett exempel tillhör alla c01or en och samma grupp, nämligen gruppen `c01`.

Varje fil och katalog på ett Unix-system har specifika *rättigheter*. Systemet skiljer på tre olika kategorier av användare; ägaren av filen/katalogen, alla användare som tillhör samma grupp som ägaren och alla andra användare. Systemet håller reda på om varje kategori skall få läsa, skriva och exekvera (köra) filen/katalogen. För att *kontrollera* rättigheterna på en fil skriver man `ls -l [filnamn]`. Ex:

```
peppar:~> ls -l foo
-rw-r--r--   1 per        193 Aug 22 14:45 foo
```

Kommando	Anropssätt	Beskrivning	Flaggor
`df`	`df`	Visar hur mycket ledigt diskutrymme som finns tillgängligt i systemet.	
`quota`	`quota` [-*flaggor*] [-*användare*]	Visar hur mycket diskutrymme du kan använda.	`-v` Visa quota för alla filsystem där användaren har diskquota
`du`	`du` [-*flaggor*]	Går igenom ett katalogträd och visar hur mycket disk-utrymme olika delar tar upp.	`-a` *All* Dvs alla filer `-k` *Kilobyte* Visa storlek i kB `-s` *Summary* Dvs endast slut-summan
`date`	`date` [-*flaggor*]	Visar datum och tid.	
`who`	`who` [-*flaggor*]	Visar alla användare som är inloggade.	
`top`	`top` [-*flaggor*]	Visar vilka processer (program) som kräver mest systemresurser.	`-U` *User* Ange en användare
`w`	`w` [-*flaggor*] [-*användare*]	Visar diverse system-information.	
`ps`	`ps` [-*flaggor*]	Visar systemets processer.	
`kill`	`kill` [-*flaggor*] id	Dödar en process.	
`passwd`	`passwd`	Byter lösenord.	
`fg`	`fg` [%jobb]	Säger åt ett jobb att köra i förgrunden (i terminalfönstret).	
`bg`	`bg` [%jobb]	Säger åt ett jobb att köra i bakgrunden.	
`jobs`	`jobs` [-*flaggor*]	Visar alla jobb för aktuell terminal.	
`pquota`	`pquota` [*användare*]	Visar hur många sidor som skrivits ut.	
`lpr`	`lpr` [-*flaggor*] [*fil...*]	Skriver ut filer på skrivare.	`-P` *Printer* Ange skrivarnamn `-#` Antal kopior
`lpq`	`lpq` [-*flaggor*]	Visar information om skrivare.	`-P` *Printer* Se ovan
`lprm`	`lprm` [-*flaggor*] [*id*]	Tar bort dokument från skrivarkön.	`-P` *Printer* Se ovan
`stty`	`stty` [-*flaggor*] [*mode*]	Visar/ändrar diverse inställningar.	`-a` Visa alla inställningar
`finger`	`finger` [-*flaggor*] [*användare*]	Visar information om användare.	

Tabell 6: Diverse systemkommandon

Här kan vi då se att ägaren till filen har rättigheter att läsa och skriva till filen, medan gruppen och alla andra bara har rättigheter att läsa filen.
Ex:

```
peppar:~> ls -l bar
-rwx---r-x   1 per         45 Aug 22 14:46 bar
```

Den här filen har ägaren rätt att både läsa, skriva till och exekvera medan gruppen inte har några rättigheter alls och alla andra får läsa och exekvera den.

För att *ändra* rättigheterna på en fil (som du är ägare till) så använder du kommandot `chmod` med följande syntax: `chmod [rättighet för ägare] [rättighet för grupp][rättighet för alla andra] [filnamn]`. De olika rättigheterna anges med siffror enligt tabellen nedan.

Rättighet	**Betyder**	**Visas**
7	läsa, skriva och exekvera	`rwx`
6	läsa, och skriva	`rw-`
5	läsa och exekvera	`r-x`
4	läsa	`r-`
3	skriva och exekvera	`-wx`
2	skriva	`-w-`
1	exekvera	`-x`
0	inga rättigheter	`--`

Om du då vill ändra så att bara du själv och gruppen kan skriva och läsa filen foo så skriver du alltså:

```
peppar:~> chmod 660 foo
peppar:~> ls -l foo
-rw-rw----   1 per        193 Aug 22 14:45 foo
```

Ett litet exempel till:

```
peppar:~> chmod 321 bar
peppar:~> ls -l bar
--wx-w---x   1 per         45 Aug 22 14:46 bar
```

När du inom en snar framtid skall börja *laborera* så skall alla laborationer sparas under en katalog vid namn `edu/[kursnamn]` i din hemkatalog. Den första ML-laborationen skall t ex sparas under katalogen `edu/ml/lab1/`. Då är det jätteviktigt att har *rätt rättigheter* på dessa filer så att labrättaren kan läsa filen och så att dina kursare *inte* kan det. Ta för vana att alltid köra kommandot chmod 705 [filnamn] på alla dina lösningar till laborationer om inte annat anges. Det medför att du har fulla rättigheter på filen, att dina kursare (som ju tillhör samma grupp) inte har några rättigheter på filen och att labrättaren kan läsa och provköra filen!

5 Pine – för att läsa epost

Om (eller kanske när) du vill kolla om du fått någon ny epost så är programmet Pine ett utmärkt val. Det är smidigt och lättanvänt. Du startar det genom

att vid en prompt köra kommandot `pine`. Detta är någonting du bör ta som vana att göra *minst en gång om dagen* under terminerna då nästan all skriftlig information till oss studenter kommer den vägen. För den intresserade läsaren kan nämnas att Pine står för *Pine Is Not Elm*. Känner du dig begränsad av Pine föreslår vi att du provar en annan klient som heter `mutt`.

6 Ssh och multi-user-system

Ett Unix-system har den fördelen att du har precis lika stor kontroll över en dator om du sitter framför den som om du loggar in på den via nätverk. Det innebär att du kan använda skolans datorer till att t ex labba från din hemdator. Det enklaste sättet att göra det är via ett program som heter `ssh` där trafiken mellan datorerna är krypterad och inte kan avlyssnas av en tredje part. Det finns ett gratisprogram för Windows som heter Putty och som kan laddas hem från bla: `ftp.acc.umu.se/Public/putty`. Om man sitter på en Unix-burk och vill logga in på t ex `peppar` är syntaxen för att använda `ssh` följande:

```
ssh användarnamn@peppar.cs.umu.se
```

Att man kan logga in på dessa datorer via nätverket innebär också att *någon annan* kan sitta och arbeta på just den dator du sitter framför i labbet. Därför gäller en viktig regel i samtliga Unix-lab:

> Stäng *aldrig* av en dator i ett Unix-lab!

De skall alltid vara påslagna. Logga bara ut och lämna sedan datorn och arbetsytan som du önskar finna den...

7 Några vanliga editorer

Under de flesta Unix-system finns det en hel bunt med *editorer*. Dessa använder man för att skriva och ändra i filer om man t ex programmerar eller skriver ett dokument. En av de allra enklaste editorerna som även finns på de flesta system heter Pico. Den är lämplig om man bara skall göra någon liten ändring i en fil då den är liten och snabb, men den är ingenting att rekommendera om man t ex skall programmera.

Då är de tyngre och kraftfullare editorerna som t ex Emacs eller XEmacs att föredra då de innehåller en mängd finesser som gör arbetet enklare. För den riktigt insatte Unix-användaren finns även editorer som både är snabba och smidiga men också fulla av finesser. Två namn på sådana är `vi` och `vim`. Dessa har dock en högre inlärningströskel och kräver lite träning innan de kan användas effektivt.

För den vane Windows användaren finns på institutionens Unix-system en editor vid namn NEdit, vilken har stora likheter med flera Windows-editorer. Problemet med denna editor är att den inte är någon standard-editor och när du kommer till ett nytt system så är det inte säkert att den finns där.

Nedan presenterar vi nämnda editorer lite närmare. Vi rekommenderar att du lär dig åtminstone en editor bra, så att du känner dig van att arbeta med den. Det kommer att underlätta för dig i framtiden!

7.1 Emacs och XEmacs

Emacs är en kraftfull editor för all typ av programmering. Den har utvecklats under lång tid och av många människor vilket fått till följd att den kräver mer systemresurser än övriga nämnda editorer. Emacs erbjuder bland annat fina möjligheter för användaren att skriva egna macron.

Emacs startas enkelt med kommandot `emacs`. Det finns även en distribution av Emacs som heter XEmacs. Dennna startas med kommandot `xemacs`. Mycket av dokumentationen till Emacs finns inbyggd och kan kommas åt med kommandot `Ctrl+H Ctrl+I`, hoppa runt med `tab`.

7.2 VI, VIM och GVIM

Vilken editor som finns på vilket system varierar ofta kraftigt, men en sak kan man nästan alltid vara säker på, det finns någon variant av `vi`. Det är den klassiska editorn för Unix-system och finns numer i ett antal olika smaker.

En populär variant av `vi` heter `vim` vilket står för VI Improved. Den har förutom alla kraftfulla finesser från klassiska `vi` även stöd för bland annat färgglad kod (*syntax highlighting*).

En ännu nyare variant är `gvim` som är en grafisk editor, till skillnad från `vi` och `vim` som båda är terminalbaserade editorer. Den enklaste att börja med är `gvim`, eftersom man då kan använda lite menyer innan man har lärt sig de vanligaste kommandona. En liten detalj med `gvim` är att, trots det faktum att den är grafisk, *inte* ska startas i bakgrunden. Alltså ska kommandoraden *inte* avslutas med ett `&`-tecken.

Det finns tre olika sätt att starta `vi`:

`vi` *`fil`*	Öppnar *fil* för redigering
`vi` *`+n fil`*	Öppnar *fil*, börja på rad *n*
`vi` *`+/text fil`*	Öppnar *fil*, börja vid *text*

När `vi` startas står vi normalt i *kommandoläge*. I detta läge är många av de vanliga tangenterna bundna till kommandon för att utföra olika redigeringar och förflyttningar. För att t ex flytta markören uppåt, neråt, till vänster och till höger används tangenterna `k j h` och `l`. Kommandot `dd` klipper ut aktuell rad, `yy` kopierar aktuell rad och `p` klistrar in kopierad text. Vill man utföra ett kommando flera gånger görs det enkelt genom att kommandot föregås av ett tal

för antalet gånger kommandot utförs. Vill vi ta bort 24 rader trycker vi `24dd`.

Det finns ett antal olika sätt att komma till *insättningsläget*, (läget där man kan skriva in ny text).

i	Sätter in text på aktuell position
I	Sätter in text i början på raden
a	Sätter in text efter markören
A	Sätter in text sist på raden
o	Öppnar en ny rad under aktuell rad
O	Öppnar en ny rad före aktuell rad

För att sedan komma tillbaka till kommandoläget trycker man `Esc`. Spara aktuell fil kan göras med kommandot `:w`. Avsluta `vi` görs med kommandot `:q`. Vill man avsluta `utan` att spara gjorda ändringar trycker man `:q!`. För att både spara och avsluta trycker man `:x` eller `ZZ`.

`vim` kan konfigureras m h a filen *~/.vimrc*. Exempel på en sådan fil finns på *~jon/.vimrc*. För en mer komplett förteckning över kommandon, se bilaga. Mer information om `vim` finns på `http://www.vim.org`.

7.3 NEdit

NEdit är en grafisk editor med många kraftfulla verktyg. Den startas genom att skriva `nedit` i ett terminalfönster. För den som tidigare sysslat med Windows så kommer mycket att vara sig likt. De vanligaste snabbkommandona, som t ex klippa och klistra text samt öppna och spara filer görs på samma sätt som i Windows. Om man tittar på menyerna så står det vilka snabbkommandon som används så det är bara att se efter. Vilka är då de viktigaste egenskaperna som gör NEdit till bra editor för programmering?

1. Suverän syntax highlighting, som kan konfigureras i all oändlighet.
2. Automatisk indentering, som är konsekvent till skillnad från XEmacs automatiska indentering.
3. Möjlighet att spela in och spela upp tangentbordsmakron.
4. Möjlighet att markera text *utan* att använda musen.
5. Bra, inbyggd, och lättåtkomlig hjälp.
6. Alla standardfunktioner såsom sök, ersätt, klipp och klistra mm.

Vad har då NEdit för brister? Ett par saker kan vara värda att nämna.

1. Den anses inte vara en standard editor för alla Unix system.
2. Kan endast köras grafiskt, d v s kan ej köras i ett terminalfönster som t ex VIM.

Sammanfattningsvis så är NEdit en bra editor, förutsatt att man använder en tämligen ny version av den. Om du vill veta mer om, och eventuellt ladda hem ett eget exemplar, av NEdit se `http://www.nedit.org`.

7.4 Pico

Denna editor, som startas genom att skriva `pico` i ett terminalfönster, är en enkel och terminalbaserad editor. Den bör absolut inte användas för programmering, eftersom den saknar många av de funktioner som en bra programmeringseditor bör ha. Denna editor används av mailprogrammet `pine` och vissa av snabbkommandona fungerar även vid terminalprompten. Vad har då Pico att erbjuda? Inte alltför mycket, men ett par saker finns att tillgå. Samtliga snabbkommandon visas längst ned i terminalfönstret när Pico startats. T ex så står det `^W Where`, vilket innebär att snabbkommandot för att söka efter text är Ctrl+W.

För att få utförlig hjälp om Picos kommandon, starta Pico och tryck `Ctrl+G`.

8 Utskrifter och utskriftsquota

För att *skriva ut* på institutionens Unix-system så använder man sig av kommandot `lpr -P skrivarnamn filnamn`. Skrivarna här på CS är namngivna efter den sal de står i, med ändelsen ps. Skrivaren i MA436 heter således `ma436ps`. Om man nu har en postscriptfil vid namn `foo.ps` och sitter i MA436 så kör man kommandot `lpr -P ma436ps foo.ps` för att få ut den på rätt sätt och på rätt skrivare. Skulle man av någon anledning vilja skriva ut på skrivaren i MA446 (trots att man sitter i MA436) anger man på samma sätt det skrivarnamnet istället.

Man kan givetvis även skriva ut saker direkt ifrån program. Om man t ex vill skriva ut en sida i Netscape, väljer man ”Print” och anger `lpr -P skrivarnamn` som ”print command”. De flesta program fungerar ungefär på samma sätt när det gäller utskrifter.

Vi studenter har inte rätt att skriva ut obegränsat antal sidor. Varje användare har en viss *kvot* som hon/han får skriva ut. Denna kvot utökas med 20 sidor för varje poäng man läser på Datavetenskap. På TDV börjar man med att läsa Programmeringsmetodik 8p, vilket innebär att man till att börja med har 160 sidor på sin kvot. Dessa ackumuleras dock till nästa kurs om man inte använder upp alla och det är ett hett tips att inte göra några onödiga utskrifter då dessa kan behövas senare.

För att kolla hur många sidor du har kvar på din kvot, eller quota som det heter på engelska, så skriver du `pquota`.

9 Kopiera och klistra samt lite ML

X, d v s det grafiska gränssnitt vi använder, har ett speciellt sätt att hantera kopiering och inklistring av text. I många program kan detta bara göras med musen. Man gör helt enkelt på följande sätt:

1. Markera text i ett program med musen genom att hålla den vänstra knappen nedtryckt och dra musen över den önskade texten.
2. Ta fram det program där du vill klistra in texten.
3. Flytta textmarkören till den position där du vill klistra in texten.
4. Klicka på den mittersta musknappen.

Detta kan tyckas som ett jobbigt sätt att kopiera text, men faktum är att i vissa fall är det väldigt behändigt att på ett enkelt sätt kunna kopiera text från ett program till ett annat. Det är väldigt smidigt när man programmerar ML. I Moscow ML (den ML-version vi använder här på CS) skriver man in sin kod direkt till den kompilatorn, man har ingen egentlig editor att tillgå. Detta är väldigt krångligt då man måste skriva om alltihop om man råkar göra ett litet fel. Lösningen på problemet är att man i en separat editor, förslagsvis någon av de som nämns ovan, skriver in sin kod för att sedan kopiera och klistra in den i Moscow ML och testköra. Blir något fel så ändrar man bara i editorn och kopierar en gång till. Gör gärna följande övning så får du känna på lite av vad det innebär att programmera i ML:

1. Starta valfri editor.
2. Starta Moscow ML genom att skriva `mosml -P nj93` och trycka på enter i ett terminalfönster.
3. Skriv in följande kodavsnitt i editorn:

```
fun nFac 0 = 1
 |  nFac n = n * nFac(n - 1);
```

Om man nu markerar texten i editorn och går över till fönstret för Moscow ML och klickar på den mittersta musknappen så läser Moscow ML in koden. Nu kan du i ML-fönstret t ex skriva `nFac 7;` och få ut svaret `5040` som ju är precis vad 7! (7-fakultet, d v s $1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7$) är. Om man vill läsa in hela filen från editorn kan man i Moscow ML, istället för att kopiera all kod och klistra in den enligt ovan, använda kommandot `use("filnamn");`, där `filnamn` är filen du vill läsa in. Om du vill avsluta Moscow ML så skriv `quit();` och tryck enter.

10 Man-sidor, datorhandboken och support

Om man behöver *hjälp* med något på ett Unix-system, t ex vilket syntax ett program har eller hur man konfigurerar det, så finns det till de allra flesta seriösa

program något som kallas man-sidor. Det är en slags hjälpfiler till programmen där det står mycket ”matnyttigt”. Ta för vana att alltid försöka lösa dina problem genom att läsa dessa sidor samt README-filer etc. Detta är väldigt lärorikt.

Du startar ett programs man-sidor med kommandot `man programnamn`.

För att söka i en man-sida skriver man bara ett slash (/) följt av sökordet och enter. Du kommer då till första träffen på sidan. För att komma till nästföljande matchning av sökordet, trycker du bara ett slash följt av enter. Du avslutar en man-sida genom att trycka `q`.

Behöver du hjälp eller bara är intresserad av något som rör ämnet datorer så kan vi rekommendera universitetets dataförening ACC's (Academic Computer Club, http://www.acc.umu.se/) datorhandbok. Den behandlar allt från sekundärminnen till Internet och är bra uppstrukturerad. Boken har dock några år på nacken och vissa avsnitt är därför inte helt aktuella.
Du hittar datorhandboken på `http://www.acc.umu.se/~handbok/`.

Vår institution (datavetenskap) har också en hel del information tillgänglig via dess hemsidor. Du hittar dessa på `http://www.cs.umu.se/` och `http://support.cs.umu.se/` .

Om du nu slutligen har kört fast totalt så att inga hjälpfiler eller handböcker i världen kan hjälpa dig så kan du höra av dig institutionens supportavdelning.

Du når dem på adressen `support@cs.umu.se` eller i rum B439 på plan 4 i MIT-huset.

11 Ordbehandling och rapportskrivande ★

Ordbehandling under Unix är något annorlunda än vad de flesta är vana vid och kan närmast liknas vid programmering. Detta dokument är typsatt i LaTeX, som är ett kraftfullt verktyg för vetenskapliga skrifter. Att arbeta med LaTeX innebär att skriva kod, som kan liknas vid HTML-kod, i någon editor. Sedan körs denna kod genom en LaTeX-kompilator, `latex`. Kompilatorn genererar en fil som kan beskådas med programmet `xdvi`. På så sätt kan man se hur dokumentet kommer att se ut på papper utan att skriva ut det. När man är färdig så använder man kommandot `dvips` för att få en PostScript fil som sedan kan skrivas ut. Man kan även omvandla till andra format än PostScript. För den som är intresserad av att lära sig mer om detta utmärkta sätt att t ex skriva laborationsrapporter rekommenderas *The Not So Short Introduction to LaTeX2ε* som kan hämtas från `http://www.ctan.org/tex-archive/info/lshort/`. För de som vill finns även StarOffice som liknar Microsoft Office.

12 Konfigurering ★

Detta är ett väldigt stort område när det gäller Unix-system. Man kan egentligen säga att *allt* går att konfigurera (ställa in) efter eget tycke. Det är även ett ganska krångligt område som kräver mycket läsning av man-sidor etc för att lista ut hur saker och ting skall skrivas i inställningsfilerna. Här nedan tar vi bara upp några små enkla inställningar som görs vid inloggningen och som kan göra ditt Unix-användande lite smidigare, resten överlåter vi till dig att utforska.

12.1 Inställningar av kommandoskalet

I detta avsnitt så tas några inställningssaker upp som kan vara värda att känna till om du vill ägna lite mer tid åt Unix. En mycket vanlig inställning som användare leker med är prompten. I sitt standardutseende så ser prompten ut som *datornamn:* >. På servern Peppar blir det då `peppar: >`. Din prompt sätts i en fil som heter `.tcshrc`. Denna fil ligger i din hemkatalog. Lägg till något i stil med `set prompt="%m:%~> "` enligt önskad prompt. Nedan följer några speciella sekvenser som ger en del värdefull information i prompten.

`%/`	Ger absolut sökväg från systemets rotkatalog
`%~`	Som ovan, men ersätter din hemkatalog med `~` om den ingår
`%m`	Ger namnet på den dator du använder
`%n`	Ditt användarnamn
`%T`	Nuvarande tid och dag

Sök sedan efter ordet `prompt` på man-sidan för att hitta rätt så snabbt som möjligt. Det finns möjlighet att även ändra det visuella utseendet på din prompt. Om du vill ha en prompt med fet stil som innehåller datornamnet och namnet på den katalog du för tillfället befinner dig i kan du prova att lägga till följande i din `.tcshrc`-fil: `set prompt="%{\033[1m%}%m:%~>%{\033[m%}"`, vilket inte är speciellt lätt att begripa, men det är inte varje dag man ändrar utseende på sin prompt.

För lite mer information om styrkoder och diverse terminalkonfiguration se `http://www.cs.umu.se/~bergner/xterm.htm`.

12.2 Vanliga inställningar i X

För att ställa in X så att det blir lite hemtrevligare kan filen `/pub/env/X/fvwm-0/fvwm2rc` kopieras till `fvwm2rc` i katalogen `.X/`, som finns i din hemkatalog. Några rader som kan vara av intresse är:

DeskTopSize anger hur många virituella skrivbord du vill ha, samt hur dessa skall placeras. `DeskTopSize 3x2` betyder att du får sex st skrivbord i tre rader och två kolumner.

AddToFunc InitFunction är en funktion som körs varje gång X startas, d v s varje gång du loggar in.

+ "I" Exec exec xv +noresetroot -root -quit

/pub/X11/pixmaps/datavetskrov.png & Denna rad sätter bakgrunden i X. Vill du byta denna bakgrund kan du titta i katalogen `/pub/X11/pixmaps/` eller `/pub/X11/backgrounds` där en hel uppsjö av bakgrunder finns att välja på. När du sedan bestämt dig för en ny, byter du bara ut `datavetskrov.png` mot den nya filen.

13 Några övningar

Till sist har vi gjort några övningar så att du kan se om du lärt dig något. Om du kör fast så se avsnittet om kommandon eller det om editorer om du inte vet namnet på någon editor. Du behöver naturligtvis inte göra alla om du inte vill. Gör de som du tror att du har mest nytta av.

13.1 Terminalfönstret - Filhantering

1. Starta ett terminalfönster.
2. Vilken katalog står du i med avseende på systemets rotkatalog?
3. Byt katalog till `/`.
4. Hur många poster finns det i den katalogen?
5. Gå till din hemkatalog.
6. Skapa en katalog med valfritt namn t ex "dir1".
7. Skapa ytterligare en katalog, denna gång en underkatalog "dir1" till t ex "dir2".
8. Skapa en textfil som du namnger till "foobar" med valfritt innehåll, du kan t ex starta Pico och skriva några rader. Spara den i dir2
9. Gör en underkatalog förslagsvis "dir3" till din hemkatalog och kopiera filen som du nyss skapat hit. Byt sedan namn på filen till "foobar2".
10. Vilka tal motsvarar filrättigheterna `rwx-xr-x`? `rw-r-r-`? `rwx-w--x`?
11. Ändra foobar2:s rättigheter så att den sista av ovanstående filrättigheter erhålls.
12. Flytta foobar2 till dir3, genom att använda ett enda kommando en gång.
13. Använd less för att läsa filen foobar2.
14. Gör en länk från din hemkatalog till foobar2.
15. Ta bort den skapade katalogen och dess innehåll med valfri metod.
16. Prova att kopiera text från ett terminalfönster till ett annat.

13.2 E-Post

1. Ta reda på vilken identitet du har.
2. Starta Pine genom att skriva `pine`.
3. Välj `Compose Message`.
4. Skicka ett mail till dig själv med rubriken "`Jag kan maila`".
5. Avsluta Pine.
6. Starta pine på nytt.
7. Välj `Folder Index`.
8. Läs brevet som du skickat.
9. Ta bort brevet. (Se nedre delen av fönstret då du står i listan för mail.)

13.3 Editorer

Prova på de du känner för, titta lite på menyer mm.

1. Starta Pico genom att skriva `pico` och avsluta genom att trycka `Ctrl+X`.
2. Starta Emacs genom att skriva `emacs` och avsluta Emacs med hjälp av menyn.
3. Starta XEmacs genom att skriva `xemacs` och avsluta XEmacs med hjälp av menyn.
4. Gå till katalogen `/pkg/nedit/bin`. Starta NEdit genom att skriva `nedit` och avsluta NEdit genom att använda menyn.
5. Starta VIM genom att skriva `vim` och avsluta genom att trycka `ESC` följt av `:q` och `ENTER`.

13.4 Man-sidor

1. Ta fram Man sidan för följande kommandon: `ls, cat, printf`.
2. Starta programmet XMan genom att skriva `xman`.
3. Upprepa uppgift 1 fast med hjälp av XMan.

13.5 X - Det grafiska gränssnittet

1. Tryck ned vänster musknapp då pekaren inte står ovanför något fönster.
2. Titta på menyn som kommer fram.
3. Upprepa steg 1 och 2 fast med den mittersta och den högra musknappen.
4. Titta på panelen till höger.

5. Klicka på de rutor som finns längst ned i panelen. Vad är de till för och hur fungerar de?

A XEmacs Reference Card

B VI Quick Reference