Praktisk Linux

Av: Marcus Rejås <marcus@rejas.se>

Jag vill tacka Jenny för att hon alltid ställer upp på mig, mitt jobb och mina tidskrävande intressen.

Denna presentation i ett format lämpligt för redigering finns på http://rejas.se/docs/praktisk-linux.sxi (OpenOffice)

Marcus Rejås,
Norrtälje 2004-07-04

Praktisk Linux

Eskilstuna 2004-07-21

Marcus Rejås
<marcus@rejas.se>

Svenska Linuxföreningen

- Även känd som SeLinux
- Ideell förening som verkar för främjandet av fri och öppen programvara och operativsystem
- > 1740 direkta medlemmar (jul 2004)
- Kostnadsfritt medlemskap, aktiv förening
- http://se.linux.org

Om Marcus Rejås

- Driver ett litet företag specialiserat på fri programvara
- Utför bland annat utredningar om när och var fri programvara passar i en verksamhet
- Inför och underhåller fri programvara
- Utbildar
- http://www.rejas.se

Inledning

- Bitvis kan det bli väldigt tekniskt. **Säg till om ni inte förstår**
- Avbryt med frågor när som helst (jag kan inte svara på allt :-))
- Vi kommer att hoppa över bilder, ni kommer att få tillgång till alla sedan om ni vill gå tillbaka
- Vi kommer att testa det mesta i praktiken, det är det bästa sättet att lära sig

Att lära sig Linux

- Ha ett öppet sinne, det är inte Windows, MacOS eller vad ni nu kan innan
- Om något verkar konstigt, ta reda på varför.
- Sätt upp konkreta mål
- För en loggbok
- Fråga om hjälp när du kör fast, men försök själv först och fråga på rätt sätt

Ämnen

Eftersom det finns fler ämnen än tid kan det vara bra att prioritera ...

- Linuxsystemet (kort)
- Uppstartsproceduren
- Filsystemet
- Användare
- Filrättigheter (vanliga och avancerade)
- Fönsterhanterare
- Kommandoskalet
- Processer
- Pipes
- Skalprogrammering (kort)
- Schemalägga med cron
- Installera program

Linuxsystemet

- Linux består av en mängd olika delar som fungerar oberoende av varandra
- Kärnan hålls så liten som möjligt
- Kärnan kan ändras under körning, med hjälp av moduler
- Varje program skall göra en sak, och göra den väl (Unixfilosofi)
- Allt är filer (Unixfilosofi)

Uppstart av ett Linuxsystem

- Så snart man blir lite varm i kläderna vill man veta vad som händer när systemet startar
- Vi beskriver här lite kort hur det går till i sysvinit (baserat på System V's init-design). Denna används i Debian, RedHat och alla derivat från dem
- Den andra skolan heter BSD som har en lite annorlunda startsekvens

Kärnan laddas (PC)

- Det första som sker är att BIOS-systemet på maskinen letar upp ett boot-block på hårddisken, disketten eller cd-rom (eller vad det nu kan vara)
- Där finns (nästan alltid) en boot-loader, till exempel Lilo eller Grub.
- Bootloadern vet var kärnan finns och laddar den i minnet och exekverar den.

Kärnan startas

- När kärnan exekveras gör den en massa saker.
  - Kollar vilken hårdvara som finns
  - Laddar rot-filsystemet (styrs via parametrar)
  - När allt är klart startas **init** (styrs via parametrar)

Init

- Initprocessen har alltid PID 1
- Initprogrammet heter oftast “init”
- Init är ett vanligt program med en konfigurationsfil som heter /etc/inittab
- Init är urförälder till alla processer i ett Linuxsystem, om en process dör före sina barn tas barnen om hand av init.
- Init gör olika saker beroende på vilken “runlevel” (körnivå) som gäller.

Körnivåer, Runlevels

- 6 olika körnivåer
  - 0 är halt
  - 1 är “Single-user mode”
  - 2-5 är “Multi-user mode”
  - 6 är reboot
  - 7 och 8 finns men skall inte användas
- Körnivån specas i /etc/inittab eller som argument till init.

/etc/inittab

- Konfigurationsfil till init
- Bestämmer vad init skall göra i olika körnivåer
- Vanligtvis skall init starta getty för att man skall kunna logga in vid konsolen
- Init startar vanligtvis även skripten som tar upp resten av systemet
- Titta på inittab!

Startskripten

- Vanligtvis i /etc/rc*
  - /etc/rcS.d/* Körs först
  - /etc/rc?.d/* Körs sedan, ? är körnivån
  - Skript med namn på S* körs med parametern “start” när systemet skartas och skript med namn på K* körs med parametern “stop” när systemet tas ned
  - Skripten körs i bokstavsordning så S05keymap körs före S35quota
  - Ofta samlar man alla skript i en katalog och symlänkar dit

Filsystemet

- Ett filsystem i Linux är väldigt standardiserat och ser ungefär likadant i alla stora distributionerna
- Filesystem Hierarchy Standard är en standard som de flesta distributioner följer
- I Linux finns inga “enheter” A:, C:, osv
- Man behöver bara veta ett filnamn och inte på vilken disk en fil finns
- Ett filsystem spänner oftast över flera diskar/partitioner

Filtyper

- Man kan urskilja 4 olika typer av filer (man kan särskilja filer på fler sätt men detta sätt är bra när man talar om filsystem)

	Statiska	Icke statiska
Delbara	/usr	/home
Icke delbara	/etc	/var

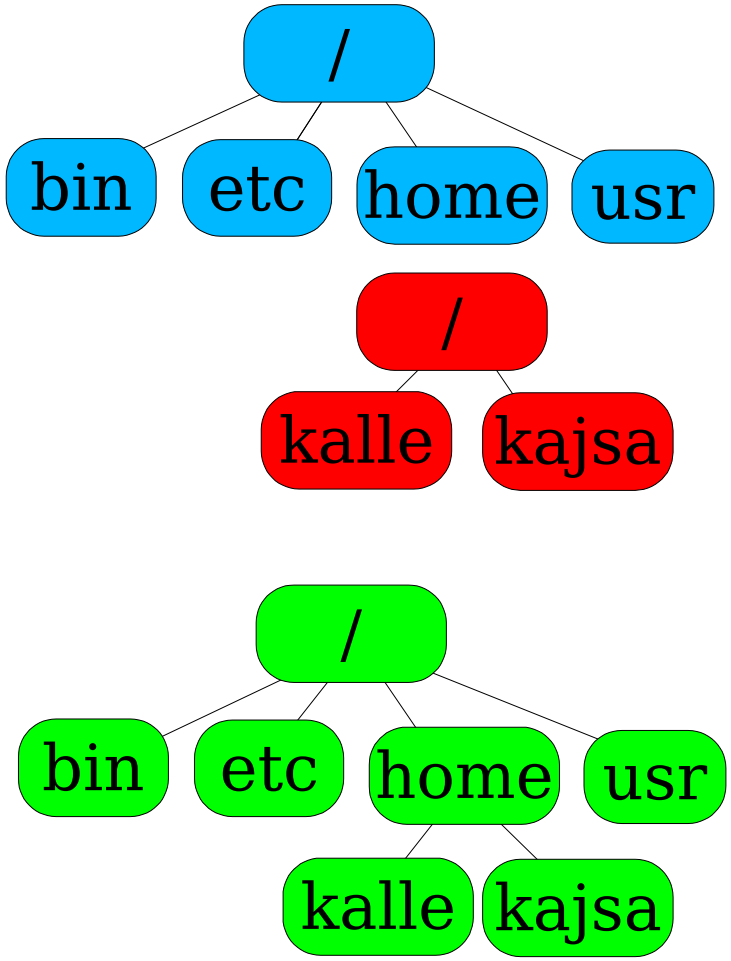
- Detta kan vara bra att ha i åtanke när man planerar sitt filsystem så att man har filer i samma kategori på samma filsystem

Montering

- Ett filsystem “monteras ihop” av flera olika små filsystem
- En katalog i ett filsystem blir en monteringspunkt för ett annat filsystem
- Alla delar i filsystemet utgår från roten “/” och kataloger åtskiljs med snedstreck “/”
- Ett exempel på ett filnamn är: /home/rejas/praktisk-linux.xsi

Montering, princip

- Det blå är rotfilsystemet
- Det röda är ett filsystem på en annan disk
- Det gröna är filsystemet efter att det röda systemet monterats på /home
- /home kallas mount-point (monteringspunkt)
- Monterar gör man med mount och umount



Vanliga kataloger

rotfilsystemet

- /bin (viktiga systemkommandon)
- /etc (systemkonfiguration)
- /dev (Enhetsfiler)
- /lib (delade bibliotek och moduler till kärnan)
- /sbin (systemkommandon)

/usr

- Oftast på en stor volym
- Delbar och statisk
- Ungefär samma struktur som rotfilsystemet
  - /usr/bin (Program som inte måste ligga i /bin)
  - /usr/sbin (Systemkommandon som inte måste ligga i /sbin)
  - /usr/lib (delade bibliotek)
  - /usr/share (Platformsoberoende)

/usr/local

- Är till för lokala program
- Skall vara tom efter det att operativsystemet installerats
- Din plats som systemadministratör
- Innehåller bin, sbin, lib osv.
- Man skall inte skapa /usr/local/program
- Glöm inte att backa denna

/opt

- Ett filsystem för tilläggsprogram
- Varje program skall ligga i /opt/program
- Kommandot för att starta programmet bör ligga i /opt/program/bin/
- Man kan skapa länkar till /opt/bin/program
- /opt/bin skall i övrigt vara tom

/var

- Här samlas all föränderlig data (förutom hemmakataloger)
- Tänk på att göra den tillräckligt stor eftersom e-post, loggar, mm hamnar här
- /var/log (systemloggar)
- /var/mail (e-postfiler)
- /var/spool (skrivarköer och annat som köas)
- /var/run, /var/lock (Används av körande program för att hålla reda på pid, lås, mm)

/home

- Denna katalog är helt frivillig. Har man inga användare så behövs den inte ...
- Detta filsystem blir oftast väldigt stort
- Beroende på antalet användare så blir det olika komplext.
  - /home/användarnamn
  - /home/a/användarnamn
  - /home/grupp/användarnamn
  - ... (upp till dig)

Filsystem

- I ett Linuxsystem kan man blanda flera olika filsysten
- Linux stödjer massor av filsystem, de vanligaste man använder är
  - **Ext2**, linux standard-filsystem
  - **Ext3**, den nyare versionen av ext2 med bland annat journalföring
  - **ReiserFS**, journalförande filsystem från Namesys
  - **JFS**, journalförande från IBM
  - **XFS**, journalförande filsystem från SGI

Nätverksfilsystem

- Linux har även stöd för nätverksburna filsystem.
- De vanligaste är
  - NFS, nätverksbaserat UNIX-filsystem
    - Dela kataloger mellan Linux och andra Unixliknande operativsystem
  - SMB, Windows NT nätverksfilsystem
    - Installeras med SAMBA
    - Delar kataloger och skrivare mellan Linux/Unix och Windowsmaskiner
  - AFS, Andrew File System
    - Avancerat, distribuerat och krypterat

Användare

- En användare är en rad i filen /etc/password och en hemmakatalog
- Finns oftast kommandon som lägger till användare i systemet (useradd, adduser)
- Lösenordet ändras med passwd
- Oftast finns standardfiler för nya hemmakataloger i /etc/skel

/etc/passwd

- Information om alla användare
- Sju fält, separerade med : (kolon)
  - Användarnamn
  - Krypterat lösenord (eller x) en ogiltig sträng medför att man inte kan logga in.
  - UID
  - GID
  - Gecos (Extra informationsfält)
  - Hemmakatalog
  - Standardskal

Skuggade lösenord

- /etc/passwd måste vara läsbar för alla
  - Medför att alla kan se de krypterade lösenorden.
  - Shadow password löser detta och lägger till andra “finesser”
- Ett x i lösenordsfältet betyder att lösenordet hämtas i /etc/shadow
- Shadowfilen är bara läsbar av root
- `username:passwd:last:may:must:warn:expire:disable:reserved`

Grupper

- Specificeras i /etc/group
- Varje användare har en primär grupp som specas i /etc/passwd.
- En användare kan tillhöra flera olika grupper
- Grupplösenord används inte
- Exempel:

```
cvs:x:102:rejas,johan,jonas,fredrik
rejas:x:1000:
```

Accessrättigheter

- Kallas accessrättigheter eller filrättigheter men skulle kanske egentligen heta filskydd eller accessmöjligheter ...
- Bestämmer vem som kan (inte får) göra vad med en fil
- Rättigheterna visas med till exempel **ls -l** eller en filhanterare
- Det finns tre sorters rättigheter
  - Läsa (r)
  - Skriva (w)
  - Exekvera, köra (x)

Accessrättigheter

- Accessrättigheterna representeras av 3 x 3 tecken: rwxrwxrwx
  - De första tre (rwx) är ägaren
  - De andra tre (rwx) är gruppen
  - De sista tre (rwx) är alla andra
- Varje grupp om tre består av tecknen r, w och x (alltid på samma plats) visar vad den kategorin får göra. Ett – (minus) visar att den behörigheten inte finns.
- Rättigheterna kollas från vänster till höger

Accessrättigheter, exempel

`rwxrwxrwx`

- Alla kan läsa, skriva och exekvera

`rwx---r-x`

- Ägaren kan läsa, skriva och exekvera
- Gruppen kan inte göra något
- De andra kan läsa och exekvera.

`------rwx`

- Bara de som inte tillhör ägare eller grupp kan läsa, skriva och exekvera

Filrättigheter med chmod

- chmod (CHange MODe) används för att ändra accessrättigheterna.
- chmod (vem)+-=(vad),... fil(er)
- `chmod u+rwx,g-rwx,o=rx` -> `rwx---r-x`
- `chmod ugo+r` -> Alla får “`r`” (+ vad de var innan)
- `chmod a+r` -> samma som ovan
- Visa exempel ...

Filrättigheter med siffror

Ofta är det enklare att med siffror ange accessrättigheterna till en fil med siffor.

```
r = 4
w = 2
x = 1
- = 0

r + w + x
4 + 2 + 1 = 7
```

Exempel:

```
rxwrwxrwx = 777
rwxr-xr-x = 755
rw------- = 600
rw-rw---- = 660
r-------- = 400
```

Accessrättigheter, SUID

- SUID
  - Set User ID
  - Om en fil med SUID satt exekveras kommer den att köras som den användare filen tillhör istället för den som den som kör filen tillhör.
  - Exempel `rwsr-x---`
- `chmod u+s` eller `chmod 4nnn`
- Naturligtvis farligt ur säkerherhetssynpunkt
- Visa exempel ...

Accessrättigheter, SGID

- SGID
  - Set Group ID
  - Om en fil med SGID satt exekveras kommer den att köras som den grupp filen tillhör istället för den som den som kör filen tillhör.
  - Exempel `rwxr-s---`
- `chmod g+s` eller `chmod 2nnn`
- Suid och sgid kan naturligtvis kombineras
  - `chmod ug+s` eller `chmod 6nnn`
- Visa exempel ...

Accessrättigheter, Sticky

- En fil kan markeras “sticky”. Eller som det brukar kallas ha “sticky-biten satt”.
- Detta innebär att filen inte, så långt det går, skall swappas ut från minnet.
- Detta används inte längre då minnen blivit både stora och billiga.
- `chmod +t` eller `chmod 1nnn`
- **Stickybiten på filer ignoreras i Linux**
- Dock används den på kataloger som vi skall titta på mer snart.

Accessrättigheter, Kataloger

- R och w fungerar ungefär som på filer
- X innebär att man har rätt att göra katalogen till aktiv katalog
- SUID innebär att alla filer man skapar i en katalog blir ägda av katalogens ägare. Skall inte användas och ignoreras av de flesta system (av säkerhetsskäl).
- SGID innebär att alla filer man skapar i en katalog kommer att tillhöra gruppens katalog. Mycket användbart.

Kataloger, sticky

- Sticky hade ju oftast ingen funktion på filer men på kataloger har den det.
- Sticky på en katalog innebär att en användare bara får byta namn på eller radera sina egna filer
- Används på kataloger där alla får skriva, till exempel /tmp
- chmod 1777 /tmp

Filtyper

- Det första tecknet i utdatat från `ls -l` är vilken filtyp en fil är. Följande sorter finns

  `-` Vanlig fil
  `d` Katalog
  `l` Symbolisk länk
  `p` Named Pipe, FIFO
  `b` Blockorienterad enhetsfil
  `c` Teckenorienterad enhetsfil

Länkar

- I Linux kan man länka filer till varandra för att öka flexibiliteten. Man kan till exempel ge en fil två namn eller få den att finnas på flera ställen.
- Det finns två typer av länkar
  - Hårda (eller vanliga)
  - Symboliska eller mjuka (symlänkar)

Länkar

- Hårda länkar
  - Fungerar som ett extra namn till filen på disken.
  - Ingen skillnad mellan länk och källa.
  - Raderas den ena finns den andra kvar
  - Fungerar inte på kataloger och mellan olika filsystem
- Symboliska länkar
  - En egen filtyp
  - Skillnad mellan länk och källa
  - Tas filen bort blir länken kvar, trasig
  - Fungerar mellan filsystem och på kataloger

Fönsterhanteraren

- Till Linux (eller till X) finns det en mängd olika fönsterhanterare.
- Olika fönsterhanterare är bra på olika saker
- Visa några exempel
  - Gnome och KDE (Kraftfulla mogna desktopmiljöer)
  - Wmaker (Snabb, lätt att konfigurera)
  - Ion (För de som helst jobbar i textläge)
  - BlackBox (Snabb)

Kommandoskalet

- Kommandoskalet är i många fall överlägset mycket snabbare än klick-klick-klick.
- Kommandon är lättare att dokumentera
- Kommandon är lättare att skripta och automatisera
- Kommandoskalet fungerar precis lika bra över nätet.
- Bli inte skrämd, du kan klicka i Linux med!

Exempel:

- Byt ändelse på alla filer i en katalog
- Ändra filnamnet på alla filer i en katalog
- Skriv ut alla filer i en eller flera kataloger
- Var tar allt diskutrymme vägen?
- Skapa små rapporter

Bash

- Standardskal i de flesta Linux-distributioner
- Namnet är en förkortning av “Bourne-Again SHell”
- Är inspirerad av Bourne-shell (sh) men har influenser även av andra skal
- Mycket bra även som skriptspråk!

Anpassning av skalet (bash)

- Efter uppstart läses flera filer om de finns
  - Först /etc/profile
  - Sedan den första som finns av:
    - ~/.bash_profile
    - ~/.bash_login
    - ~/.profile
- Om skalet inte är ett “login-skal” körs ~/.bashrc
- Alla dessa filer är skript som kan innehålla kommandon och variabeldeklarationer.

Processer (1/x)

- Processer är program eller delar av program som exekveras
- Visas till exempel med programmet ”ps”
- En process kan ”dödas” med ”kill <PID>”
- Flera kan dödas med ”killall <namn>”
- Init har alltid PID = 1

Processer (2/2)

- Varje process har en egen ”omgivning”
- Alla processer (utom init) har en förälder
- Processer vars föräldrar dör, adopteras av init
- Innan en process avslutas måste den meddela sin förälder det. Att detta misslyckas är den vanligaste anledningen till så kallade Zombies
- Processträdet kan enkelt studeras med ”pstree”

Processhantering

- Skalen kan enkelt hantera processer
- $ kommando & <- Startar i bakgrunden
- $ fg <- Tar till förgrunden
- Ctrl-Z <- Stoppar en process
- $ bg <- Skickar till bakgrunden
- Ctrl-C <- Avslutar en process

Screen

- Mycket bra för dig som jobbar i terminalfönster
- Ha dina program körandes och redo på din maskin
- Ta upp din session var du än befinner dig
- Fönsterhanterare för textbaserade program
- Titta på hur screen fungerar

Pipes/Rör

- En grundläggande filosofi i Unix är att allt består av filer och att alla program skall vara små, göra en sak och göra den väl
- Många små program kan kopplas ihop till ett större
- Små, kanske vid första ögonblicket meningslösa program blir helt plötsligt väldigt användbara.

Exempel

- **$ ps aux | wc -l**
  (Hur många processer)
- **$ ps aux | awk '{print $4" "$11}' | sort -nr | head -5**
  (De 5 mest minneskrävande programmen)
- **$ ls -l *.doc | wc -l**
  (Hur många dokument i aktuell katalog)
- **$ find . -name '*.doc' | wc -l**
  (I aktuell katalog och alla underkataloger)
- **$ du -s * | sort -nr | head -5**
  (De 5 största katalogerna eller filerna)
- **$ if ! ping -c 2 10.0.0.1 >/dev/null; then echo Fixa; fi**
  (echo Fixa är ett skript eller kommando som lagar internet-uppkopplingen)

Skalskript

- I princip allt du kan göra i textläge kan du skripta och autmatisera.
- Gör inget krångligt eller tråkigt flera gånger, skripta det!
- Skript är den ultimata dokumentationen
- “Exekverbara dokument” (Pragmatic Programmer)

Skalskript

- De flesta skal är fullfjädrade programmeringsspråk
- Bash och sh är vanligast
- Finns variabler, vilkorssatser, loopar och det mesta annat du hittar i “riktiga” programspråk.

Webgrep

- Enkelt skript som greppar efter något på en hemsida.

```
#!/bin/sh
if [ $# -ne 2 ]; then
        echo “Usage: $0 url <Text string>”
        exit
fi
wget -O - $1 2>/dev/null | grep $2
```

Tips & tricks

- Svårt att veta vilken terminal som är vilken?
  - echo -e "\033]0;Någon Text\007"
- Sätter titelraden i de flesta terminalemulatorer
  - Används med fördel med PROMPT_COMMAND i Bash.

```
if [ "$DISPLAY" ]; then
    PROMPT_COMMAND='echo -ne "\033]0;${USER}@${HOSTNAME}: ${PWD}
\007"'
    export PROMPT_COMMAND
fi
```

Schemalägga med cron

- Cron är ett system för att schemalägga saker
- Kan användas både för underhåll och av användare för att spara tid
- Utdata av kommandon skickas med e-post till användaren
- Du hanterar din shemaläggning med kommandot **crontab**
  - crontab -l (lista dina schemalagda job)
  - crontab -e (editera dina job)

Crontab, syntax

- Crontabfilen har en kraftfull syntax.
- 6 kolumner:
  - Minut [0-59]
  - Timme [0-23]
  - Dag [1-31]
  - Månad [1-12]
  - Veckodag [0-7] (Både 0 och 7 är söndag)
  - Kommando

Crontab, syntax

- Alla värden kan ha använda följande regler
  - * (matchar alla)
  - 1-9 (matchar 1,2,3,4,5,6,7,8,9)
  - 1,2,4-7 (matchar 1,2,4,5,6,7)
  - 0-2,4-6 (matchar 0,1,2,4,5,6)
  - 1-6/2 (matchar 2,4,6)
  - */2 (matchar alla möjliga värden som är jämt delbara med 2)

Crontab exempel

```
* * * * * touch /tmp/cron_running
0 3 * * * /usr/local/sbin/backup -inc
0 16 * * 5 echo “Dags att göra helg”
0 4 1 * * /usr/local/sbin/backup -full
0 * * * * ~/bin/hourly.job
0 0 * * * ~/bin/daily.job
0 0 1 * * ~/bin/monthly.job
0 0 * * 5 ~/bin/weekly.job
```

Installera program

- Varför skall det vara så krångligt?
  - Det är inte så krångligt
  - Om man håller sig till paket för sin distribution så är det väldigt enkelt
- RedHat, m.fl. använder RPM (man rpm)
- Debian, använder deb och apt-get (man apt-get)
- Om man letar efter programmet paketerat för sin distribution så är det enkelt att installera, det kan också enkelt uppdateras

Installera från källkod (1/7)

- Linux är posix-kompatibelt
  - I princip alla program för Linux och Unix kan installeras om de kompileras för din miljö
  - Många fria program distribueras som källkod
- För att kunna kompilera källkod måste man ha en kompilator och alla de komponenter som programmet kräver
- Linuxdistributioner kommer vanligtvis med flera kompilatorer

Installera från källkod (2/7)

- Packa upp filerna
  - De flesta distribueras om ett gzippat tar-arkiv (**.tgz** eller **.tar.gz**)
  - `tar -tzf filnamn.tgz` <- Listar filerna
  - `tar -xzf filnamn.tgz` <- Packar upp filerna
  - Är det ett **.zip** arkiv kan man använda **unzip**
  - Är det ett **bzip2** arkiv kan tar användas, byt ut **x** mot **j**. Eller använd **bunzip2**.

Installera från källkod (3/7)

- Läs README och INSTALL
- I nästan alla paket finns en fil README och/eller en INSTALL. I dessa står det hur man bygger programmet och installerar det (säkert!)
- I dessa filer står det också om några speciella komponenter måste installeras. Till exempel grafiska program brukar behöva en mängd komponenter

Installera från källkod (4/7)

- Anpassa för din miljö
  - De flesta program använder någon som heter autoconf. Det gör det enkelt att anpassa för ditt system
    - Kör **`./configure`**
  - Andra program kan ha andra system eller inte behöva ändras alls
  - Detta står i filerna vi nämnde tidigare ...

Installera från källkod (5/7)

- Kompilera
  - Kompileringen görs oftast med **make**
  - Detta kan ta lång tid. Vissa program genererar varningar under kompileringen dessa kan du, i de flesta fall, ignorera (men upphovsmannen borde bry sig om dem)
  - Om allt går som det skall så kommer prompten tillbaka utan att det står **error** på raderna ovanför

Installera från källkod (6/7)

- Kopiera till rätt kataloger
  - Program vars utvecklare gjort rätt skall installera sig i /usr/local/*
  - Vanligtvis **make install**, man måste som regel vara **root** för att kunna göra detta.
- Sedan skall det bara vara att provköra
- Nu kan du radera källkoden om du vill

Installera från källkod (7/7)

- Sammanfattning
  - Läs README och/eller INSTALL
  - Packa upp (tar -xzf filnamn.tar.gz)
  - Gör källkodskatalogen till aktuell (cd katalog)
  - Konfigurera (./configure)
  - Kompilera (make)
  - Installera (make install)
  - Klart :-)

Slutsatser

- Förmodligen är ni helt slut efter detta virriga, krångliga föredrag
- Förhoppningsvis är ni även sugna på att utforska Linux nu när ni sett mer vad det kan göra
- Linux är väldigt spännande och kan göra det mesta du kan tänka dig ...
- Lycka till med ditt Linuxande!!