Linux Kommandolinjen

Terje Berg-Hansen

ITFakultetet.no

Linux Kommandolinjen

Av Terje Berg-Hansen.

Publisert av ITFakultetet AS, Kåsabakken 28, 3804 Bø i Telemark, Norge

Denne E-boken brukes som dokumentasjon til disse kursene på ITFakultetet.no:

- Linux Workshop: Kommandolinjen
- Linux Sysadmin – trinn 1

Sjekk gjerne www.itfakultetet.no for kursbeskrivelser og aktuelle kursdatoer.

Forord

Denne E-boken er skrevet for den komplette nybegynner, men bør også fungere bra for de som har jobbet en del med kommandolinjen fra før, kanskje uten noen formell opplæring, men som har «Googlet» løsninger, klippet og limt litt, og gjerne vil ha litt større forståelse for hva som egentlig foregår når kommandoene gir forventede eller overraskende resultater – eller ingen resultater i det hele tatt.

Alle tilbakemeldinger mottas med takk, spesielt slike som kan forbedre boken og gjøre den mest mulig tilgjengelig og nyttig for leseren.

Oslo, 2021

Terje Berg-Hansen

Kursleder
ITFakultetet.no

Epost: terje@itfakultetet.no

INNHOLD

Innledning

Kapittel 1
Introduksjon og installering

Introduksjon til GNU/Linux

GNU/Linux blir til

- Richard Stallman startet i 1983 prosjektet GNU for å lage et fritt operativsystem
- GNU = Gnu is Not Unix
- Linus Thorvalds lagde i 1991 et studentprosjekt han kalte Linux, som skulle være en Unix-lignende kjerne som kunne kjøres på vanlige PCer.
- GNU manglet en kjerne, og adopterte Linux-kjernen. Slik ble GNU/Linux et komplett operativsystem med en kjerne og et omkringliggende system av rutiner, verktøy og programmer.
- Idag kan Linux kjøres på PCer, Power PCer (Apple), Alpha-baserte maskiner, MIPS-baserte maskiner, IBMs S/390, ARM-maskiner og en rekke andre plattformer

Linux er et sikret flerbrukersystem

- Maskinvare var dyrt da Linux ble laget, og Linux er bygget for at mange brukere skal kunne bruke samme maskin. Brukerne er medlem av en eller flere grupper, og rettigheter tildeles på bruker- eller gruppenivå.
- Brukere kan være innlogget samtidig, kommunisere med hverandre og dele systemressurser på en intelligent måte.
- Linux er et operativsystem som håndterer protected mulittasking noe som innebærer at hver bruker kan kjøre mer enn en prosess samtidig. Prosessene kan kommunisere med hverandre, men er fullt beskyttet fra hverandre. Jobber kan kjøres i bakgrunnen, mens man fokuserer på den jobben som vises på skjermen.
- Filstrukturen er hierarkisk bygget opp gjennom en rot-mappe med undermapper. Lese- og skrivetillatelser gis til brukere eller grupper av brukere for hver mappe og hver fil. En vanlig bruker har ikke tilgang til f.eks. å slette viktige systemfiler, så hvis noen (en fremmed eller et virus) får tilgang til brukerens passord, kan ikke hele systemet ødelegges - kun denne brukerens egne mapper og filer.

Noen grunner til å bruke Linux:

- **Frihet / Leverandøruavhengighet**

Linux og "Open Source" (åpen kildekode) programvare er gratis. Dette innebærer at lisensen er en "fri lisens", og den vanligste av disse er GPL (General Public License). Av denne lisensen framgår det at alle og enhver har rett til å bruke programvaren, distribuere programvaren, endre den, og distribuere endringene under forutsetning at den forblir lisensiert med GPL.

- **Sikkerhet: tilnærmet virusfritt**

Linux har så og si ingen virus. Det er nok ikke umulig å få det, men det er uhyre sjeldent at det opptrer fordi Linux er bygd på en måte som gjør det svært vanskelig for virus å trenge igjennom.

- **Mengder av gratis programvare - enkelt installert og oppdatert gjennom internett.**

Siden programvaren for det aller meste er fri og gratis, ligger den samlet i programvarekartoteker (brønner eller kilder) på internett. Det gjør at du kan installere og oppdatere ikke bare operativsystemet, men all programvare gjennom et par enkle klikk eller kommandoer.

Linux på Serveren

En Linux-server er en system-administrators drøm. Linux tilbyr det beste og mest brukte innen web-servere, epost-servere, fil-servere, database-servere, media-streaming-servere, Hadoop-klynger mm. Linux-servere er stabile og sikre og blir stadig enklere å administrere.

Linux er mye brukt som web-server gjennom det såkalte LAMP-oppsettet. LAMP står for Linux, Apache, MySQL og Php. I praksis vil dette si at man setter opp en Linux-server med Apache web-server, MySQL database-server og Php skript-språk.

Det finnes en rekke verktøy for installasjon, administrasjon og overvåking av Linux-servere, og det er en stor community av Linux-entusiaster på diverse kanaler på internett som er behjelpelig hvis du står fast eller trenger løsning på et problem raskt. Et Google-søk er ofte nok til å vise deg en eller flere måter å løse problemet på.

Web-baserte grensesnitt, som f.eks. cockpit, gjør det enkelt å administrere de vanligste oppgavene, mens rot-tilgang via SSH (Secure SHell) gir en fantastisk detaljert kontroll over alle aspekter av serveren.

Noen populære Server-distribusjoner

Debian

Debian er en populær server-distro, som flere andre distroer bygger på, bl.a. Ubuntu. Debian er community-drevet, som innebærer at det ikke er ett selskap som har ansvar for utvikling, brukerstøtte osv.

Ubuntu

Ubuntu Server har økt eksponensielt i popularitet de siste årene, i takt med Ubuntus generelle fremgang. Ubuntu støttes profesjonell av firmaet bak Ubuntu- Canoncical - og er en stabil og enkel installasjon. Den er også gratis (dersom du ikke ønsker support-avtale), og har en stor Community-støtte.

Red Hat - RHEL / Centos / Fedora

RHEL (Red Hat Enterprise Linux) er en profesjonell, stabil og gjennomtestet server-installasjon, som støttes profesjonelt av Red Hat. For å ha et tilbud til de som ikke trenger Red Hats supportavtale, har man laget en Community-versjon av RHEL, som heter **Centos,** og som er identisk med den originale Red Hat serveren, minus support-avtale. Centos støttes nå også offisielt av Red Hat. **Fedora** er utviklerutgaven av RedHat, som inneholder nyere komponenter enn RHEL/Centos.

SUSE - SLES / OpenSUSE Leap

Suse har en betydelig del av spesielt det europeiske servermarkedet med sin SLES (Suse Linux Enterprise Server). Etter at Suse ble kjøpt opp av Novell, har det blitt lagt inn mye av Novells Know-how i serveren, som bl.a støtter XEN-virtualisering og andre teknologier.

Installasjon av Ubuntu Desktop og Server

Desktop-utgaven av Ubuntu kan lastes ned fra ubuntus nettsider:

https://ubuntu.com/download/desktop

Server-utgaven av Ubuntu kan lastes ned fra ubuntus nettsider:

https://ubuntu.com/download/server

Testinstallasjon
Dersom du vil teste ut Ubuntuserver, kan du installere den som en virtuell server i f.eks. VirtualBox.

Her er en lenke til en detaljert gjennomgang av installering til VirtualBox:

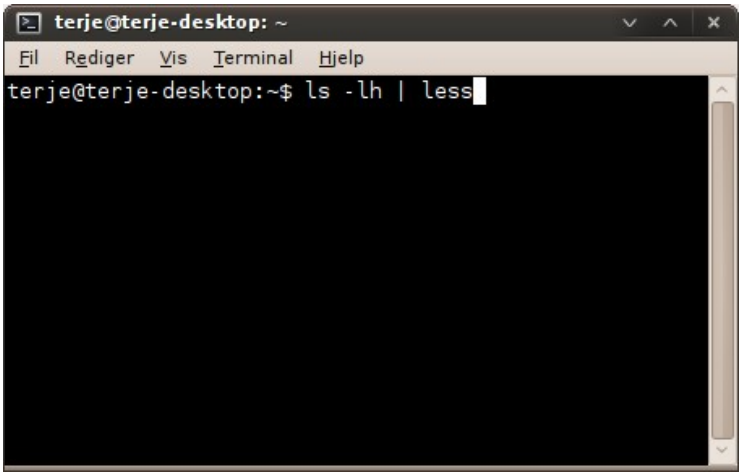
https://www.wikihow.com/Install-Ubuntu-on-VirtualBox

Hva er kommandolinjen?

Kommandolinjen er et fleksibelt, allsidig verktøy som kan gjøre en rekke jobber raskt og effektivt. Den kan bli brukt interaktiv gjennom et *skall* eller *terminal-vindu* eller ved å skrive og kjøre såkalte *skallskript*, ofte kalt *Bash-skript* etter det populære Bash-skallet (**B**ourne **A**gain **SH**ell). Resultatet en kommando produserer kan f.eks. sendes gjennom et *"rør"* (*pipe*), dvs. brukes som input til en annen kommando, det kan vises i et terminalvindu, printes eller lagres i en tekstfil. Resultatet av en kommando kan også lagres til en fil ved å omdirigere det fra standard output, som er skjerm, til et filnavn med tegnet > eller >> (det siste "appender" til fil) etterfulgt av banen/navnet til filen. For eksempel:

```
$ ls -lh > mappeinnhold.txt
```

Kommandoen i terminalvinduet nedenfor lister opp filene i en mappe (ls) med detaljert visning og i lettlest format (-lh), og sender resultatet gjennom et rør (|) til programmet *less* som bl.a. lar deg bruke piltastene til å "scrolle" opp og ned i resultatet.



Kommandolinjen i et terminalvindu, -emulator eller *skall*

Grunnleggende bruk

- Standard input = tastatur og standard output = skjerm
- Kommandoer skrives i et terminal-skall med standard input og resultatet sendes til standard input - hvis ikke input og/eller ouput er omdirigert med < eller >
- Omdirigering av output med > eller >> - omdirigerer fra skjerm til fil eller til ingenting (/dev/null - "the bit bucket")
    - find -type f 2>/dev/null

- Kjeding av kommandoer med | - output fra en kommando blir input til neste kommando
  - sudo grep failed /var/log/secure | wc -l (søker opp linjer som inneholder "failed" fra secure-loggen og sender resultatet til word count for å telle linjer, dvs. telle mislykkede innlogginger, passord-sjekker o.l.
- Kjør to kommandoer etter hverandre med && eller ||
  - mkdir mappe1 && cd mappe1
  - rm fil.txt || true
- Initialiser variabler med = og referer til dem med $
  - PATH = $PATH:/home/terje/bin
- Bruk (( )) rundt matematiske beregninger og referer til dem med $(( ))
  - $ echo $((4+2*3)
  - 10
- Bruk { } til å erstatte noe med noe annet, og referer til det med ${ }
  - $ tekst="Dette er riktig"
  - $ echo ${tekst/er/var}
  - Dette var riktig
- Kjør programmer i bakgrunnen med &
  - gimp &
- Bruk tab til å fylle ut det som mangler ("tab completion")
  - cat fore + tab fyller ut resten av filnavnet til : cat forekomster_av_ord_i_war-and-peace.txt
  - Gjelder også for programmer, kommandoer og noen steder også parametre
- Få hjelp via **man** og **info**
  - man cut (gir manualen til kommandoen cut)

Kommandoer versus museklikk

Fordeler

- Raskere å bruke enn grafiske brukergrensesnitt (fingrene dine forlater aldri tastaturet)
- Konfigurerbare snarveier og taste-bindinger.
- Virker også når du ikke har tilgang til grafiske grensesnitt, f.eks. når du logger deg inn på en tjener gjennom et terminalvindu

Ulemper

- Du må huske kommandoer og snarveier (selv om det finnes måter å forenkle dette på)
- Vanskelig å viser bilder og video (men ikke umulig)

Tekstbaserte programmer

Kommandolinjen kan også brukes til å kjøre tekstbaserte programmer i terminalvinduet - også kalt ***TUI-applikasjoner*** (Text-based User Interface). TUI-applikasjoner kan være raskere og mer fleksible og

konfigurerbare enn sine grafiske motparter: GUI-applikasjoner (Graphical User Interface).

Eksempler på tekstbaserte programmer

- Nettlesere
    - Lynx, Links, w3m
- Epost-programmer
    - Mutt, Pine
- Kalendere
    - Calcurse, cal
- Mediaspillere
    - Mocp, Mp3blaster, play
- IRC
    - irssi
- Tekstbehandlere
    - Vim, Emacs, Nano, Joe etc

history - gjenbruk av tidligere kommandoer

history er et program som lagrer et angitt antall kommandoer (som regel er default 1000) i en fil, og som inneholder kommandoer for å hente dem fram igjen.

Eksempler på bruk

1) Finn foregående kommandoer:

a) Tast <Piltast opp> for forrige kommando

a) Tast <Piltast ned> for neste kommando

2) Søk etter en tidligere kommando:

a) Tast <ctrl>+r og tast inn begynnelsen på søkeorde(ne)

b) Repeter <ctrl>+r til du finner riktig kommando

3) Vis alle lagrede kommandoer:

```
$ history
```

4) Vis n siste lagrede kommandoer:

```
$ history <n>
```

5) Send alle lagrede kommandoer til egen fil:

```
$ history > history
```

.bashrc

Den skjulte filen **.bashrc** ligger i brukerens hjemmemappe og inneholder default-innstillinger, path, systemvariabler, aliaser osv for den aktuelle brukeren.

I **.bashrc** kan du legge innstillinger som bare skal gjelde for deg. Dersom innstillingene skal gjelde for alle brukere, oppretter du i stedet en fil med et passende navn, og legger denne i mappen **/etc/profile.d/** (krever rot-tilgang).

Her er et par eksempler på hva du kan legge i din egen **.bashrc**

alias upgrade='sudo apt update && sudo apt full-upgrade' (for debian-baserte distroer)

PATH=$PATH:/home/terje/programmer (legger til mappen programmer i søke-stien for kjørbare programmer)

HISTSIZE=2000 (antall kommandoer som skal lagres i hist - filen - endret fra default 1000)

Tmux og Screen

tmux

Programmet tmux er glimrende hvis du trenger å ha flere terminalvinduer samtidig på en maskin uten grafisk grensesnitt. Det er også genialt hvis du f.eks. logger deg inn på en server med ssh, må avbryte og logge deg ut, men vil fortsette senere - uten å miste det du holdt på med.

Start tmux med denne kommandoen:

```
$ tmux
```

Etter en velkomstbeskjed (klikk enter for å bli kvitt den) er du klar til å lage flere vinduer. Screen bruker **<ctrl>+b** som kommandotast og her er de viktigste kommandoene:

```
$ <ctrl>+b+c = Lag nytt vindu (c = create)
$ <ctrl>+b+n = gå til neste vindu (n = next)
$ <ctrl>+b+x = slett vinduet du er i, når det siste er slettet, avsluttes
tmux
```

Men her er den beste:

```
$ <ctrl>+b+d = frigjør vinduene (d = detatch)
```

Nå kan du logge ut fra serveren og komme tilbake dagen etter og logge deg inn, og så starter du tmux med denne kommandoen:

```
$ tmux a
```

(a for attach)

Og så kan du fortsette å jobbe der du slapp dagen før. Hvis du har flere gamle sesjoner kjørende, kan du taste:

```
$tmux a -t <nummer>
```

for å gjenopprette den sesjonen du vil gå inn i. Sesjonene nummereres med et løpenummer fra 0 og oppover.

Dersom du ikke kan installere tmux, kan du antagelig installere screen, som er forløperen til, og fungerer som en enklere utgave av tmux

screen

Programmet screen er alternativet til tmux når du trenger flere terminalvinduer samtidig på en maskin uten grafisk grensesnitt.

Start screen med denne kommandoen:

```
$ screen
```

Etter en velkomstbeskjed (klikk enter for å bli kvitt den) er du klar til å lage flere vinduer. Screen bruker **<ctrl>+a** som kommandotast og her er de viktigste kommandoene:

$ <ctrl>+a+c = Lag nytt vindu (c = create)
$ <ctrl>+a+n = gå til neste vindu (n = next)
$ <ctrl>+a+a = gå frem og tilbake mellom to vinduer (a = alternate)
$ <ctrl>+a+k = slett vinduet du er i (k = kill)
$ <ctrl>+a+d = frigjør vinduene (d = detatch)

Nå kan du logge ut fra serveren og komme tilbake dagen etter og logge deg inn, og så starter du screen med denne kommandoen:

```
$ screen -r
```

(r = resume)

Og så kan du fortsette å jobbe der du slapp dagen før. Har du åpnet flere screen-sesjoner, får du en liste over dem, og må angi en id for å komme til den sesjonen du vil jobbe i.

Hvis du har flere gamle sesjoner kjørende, vil du få beskjed om det og blir bedt og å taste inn PID-nummeret (Prosess ID) til den sesjonen du ønsker å fortsette i (PID-nummerne til de aktuelle sesjonene vises på skjermen).

Hvis du starter screen uten -r, startes en ny sesjon som legges til evt. eksisterende sesjoner.

Introduksjon til tekstbehandling med Vim

Vim står for VI Improved, og er som navnet antyder en forbedret utgave av den klassiske tekstbehandleren VI (uttales vi-ai). Det finnes flere tekstbehandlere som fungerer uten grafisk brukergrensesnitt - f.eks. Emacs, Nano og Joe - men Vim er mye brukt, ofte installert og ikke helt intuitiv i bruk, derfor kan det være på plass med en kort brukerveiledning til den.

Behovet for en tekstbasert tekstbehandler

Det heter seg at "alt i Linux er tekstfiler", og f.eks. er de aller fleste konfigurasjonsfiler tekstbaserte. Når du skal redigere en tekstfil på en server, f.eks. via SSH, eller direkte på en server med en tekstbasert terminal, er Vim (eller en annen tekstbasert tekstbehandler) ofte svaret. Har man tilgang til et grafisk brukergrensesnitt vil nok mange velge f.eks. gedit, kate, geany, bluefish eller en annen grafisk editor, men det finnes også en gui-basert versjon av Vim -gVim, hvis man skulle ønske det.

Vims særegenheter

Noe av det som forvirrer mest ved første møte med Vim er at programmet har to ulike modus - kommandomodus og redigeringsmodus. Når du starter Vim, starter programmet i kommandomodus. Slik åpner du en tekstfil for redigering med Vim:

```
$ vim tekstfil.txt
```

Dersom filen ikke finnes fra før vil Vim opprette en tom fil med filnavnet du tastet inn.

Gå til redigeringsmodus

For å gå over i redigeringsmodus, slik at du kan begynne å skrive eller redigere, taster du bokstaven **i** (for *insert*) eller bokstaven **a** (for *append*). Flytt gjerne markøren med piltastene til linjen du vil redigere før du taster **i** eller **a**.

Du kan slette tegn på vanlig måte med tastene **<Del>** eller **<BackSpace>**.

Gå til kommandomodus

Når du har redigert ferdig, må du gå over i kommandomodus igjen for å lagre og avslutte Vim. Dette gjøre du ved å taste **<escape>**.

Lagre og avslutt

Når du er i kommandomodus, kan du gi Vim diverse kommandoer. Alle kommandoer begynner med tegnet kolon **:** etterfulgt av kommandoen, f.eks. slik:

:w (write) - lagrer filen, uten å lukke filen eller avslutte Vim

:x (exit) - lagrer filen, lukker den og avslutter Vim.

:q (quit) - avslutter Vim uten å lagre filen, dersom du ikke har gjort endringer)

:q! (quit anyway) - avslutter Vim uten å lagre filen, selv om du har gjort endringer.

Andre nyttige kommandoer og funksjoner

Linjenummerering

Du kan slå av og på linjenummerering med disse kommandoene:

:set number

:set nonumber

Gå til en linje i et åpent dokument med kommandoen **:n** - hvor n er linjenummeret, f.eks vil :234 flytte markøren til linje 234

Du kan åpne et dokument og gå direkte til en linje i dokumentet ved å skrive + og linjenummeret etter filnavnet når du åpner filen, f.eks slik:

```
vim main.cfg +367
```

Angre

u - angrer siste endring (kan repeteres)

Søke etter tekst

/ (søk) - skriv søkeord rett etter /, f.eks. slik:

/test - søker etter første forekomst av ordet test.

n (next) - søker etter neste forekomst av ordet test.

Vil du gjøre "case insensitive" søk, dvs. ikke skille mellom stor og små bokstaver, gjør du dette ved å sette vim til å være case insensitive før du søker, slik:

:set ic (ic står for: ignore case)

Etter søket kan du sette Vim tilbake til case sensitive modus slik:

:set noic

Søk og erstatt

:%s/ord1/ord2 - erstatter første forekomst av ord1 med ord2 (søker i hele teksten)

:%s/ord1/ord2/g - erstatter alle forekomster av ord1 med ord2 (i hele teksten)

:%s/ord1/ord2/gc - erstatter alle forekomster av ord1 med ord2 og ber om bekreftelse for hver erstatning (i hele teksten)

:%s/ord1/ord2/i - erstatter første forekomst av ord1 med ord2 uten å skille mellom store og små bokstaver (i hele teksten)

:%s/ord1/ord2/I - erstatter første forekomst av ord1 med ord2 og skiller mellom store og små bokstaver (i hele teksten) - dette er også default innstilling, men kan brukes etter å ha gjort om default til *case insensitive* med kommandoen **:set ignorecase**

Slette tekst

dw - sletter ett ord

dd - sletter en hel linje

Kopiere og lime inn tekst

yy - kopierer en linje til minnet

yn - kopierer n+1 linjer tekst til minnet

p - limer inn kopiert tekst

Åpne flere filer i hvert sitt vindu (splittet skjerm)

Du kan åpne flere filer samtidig, og la Vim plasserer dem i hvert sitt vindu, enten horisontalt eller vertikalt, med disse kommandoene:

```
vim -o fil1.txt fil2.txt fil3.txt (splitter skjermen i tre horisontale
vinduer)
vim -O fil1.txt fil2.txt fil3.txt (splitter skjermen i tre vertikale
vinduer)
```

Du kan så redigere teksten i hvert vindu, f.eks. lagre og avslutte vinduet med **:x,** og bla til neste vindu med kommandoene: **<ctrl+w>+pil ned** eller **<ctrl+w>+pil opp** - eller ved vertikal deling, med **<ctrl+w>+pil venstre** eller **<ctrl+w>+pil høyre. <ctrl>+w+w** skifter frem og tilbake mellom to vinduer**.**

Splitte et vindu i to deler

Du kan splitte et vindu horisontalt med kommandoen **:split**, og vertikalt med kommandoen **:vsplit**

Tast **<ctrl>+w+w** for å flytte markøren fra det ene vinduet til det andre.

Det nye vinduet vil inneholde det samme dokumentet som det gamle. Du kan redigere et nytt dokument i det nye (eller gamle) vinduet med kommandoen **:edit <filnavn>**

Vise forskjellene mellom to filer med vimdiff

Programmet **vimdiff** lar deg sammenligne innholdet i to eller flere filer. Filene åpnes i vertikalt splitede vinduer, og ulikhetene markeres med fargekoder. Skriv filnavnene til filene du vil sammenligne som parametre tli vimdiff, slik:

```
vimdiff <fil1> <fil2> <fil3>
```

Filene kan redigeres på vanlig måte

Åpne flere filer i hver sin fane

Du kan åpne filer i faner i stedet for i egne vinduer. Dersom du redigerer en tekst og vil åpne en ny fil i en egen fane, kan du bruke denne kommandoen: **:tabe myfile.txt.** Du kan bla mellom fanene med kommandoene: **<ctrl +pgup>** og **<ctrl+pgdn>**

Du kan åpne flere filer direkte i faner ved å laste dem inn med flagget: -p, slik

```
vim -p first.txt second.txt
```

Her er noen flere kommandoer relatert til faner:

```
:tabedit {filnavn}   åpner en fil e en ny fane
:tabfind {filnavn}   søker etter filnavn (bruk tab) og åpner filen i ny
fane
:tabclose            lukker gjeldende fane
```

```
:tabclose {i}        lukker fane nummer i
:tabonly             lukker alle andre faner enn den gjeldende
```

Vims config-fil: .vimrc

Du kan opprette en config-fil for vim i hjemmemappen din. Kall filen **.vimrc** (punktumet angir at det er en skjult fil). I denne filen kan du angi default-verdier for oppstart av vim, feks:

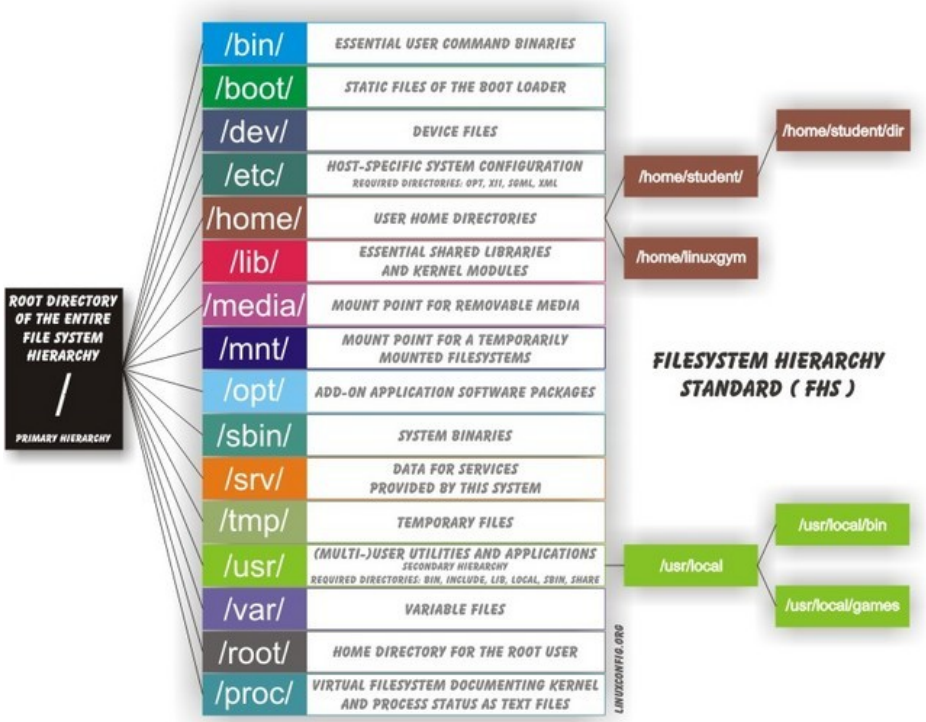
set number
colorscheme darkblue

Eksterne ressurser:

- **http://www.vim.org - Vims offisielle hjemmeside**, med dokumentasjon, nedlasting etc,
- **http://vim.wikia.com/ - Vim Tips og triks**
- **http://vimdoc.sourceforge.net/ - Vim dokumentasjon**

Kapittel 2
Filsystemer, mapper og filer

Linux Filsystem og Systemfiler



Bildet over viser standard-mapper i en vanlig Linux-installasjon. Øverste nivå kalles gjerne **rot-nivå** og angis med:

/

Brukernes hjemmemapper lagres i mappen

/home

Feks: /home/petter (Denne mappen tilsvarer mer eller mindre "Mine Dokumenter" i Windows)

Systemets konfigureringsfiler ligger lagret i mappen:

/etc

Systemets programmer ligger lagret i mappen:

/bin

Midlertidige filer ligger lagret i mappen:

/tmp

Filer som kan brukes av flere brukere ligger i mappen:

/usr

Feks. /usr/share/wallpapers (mappe med bakgrunnsbilder til skrivebordet)

Flyttbare media, som CD-rom, DVD, USB, Ipod osv finner man i mappen:

/media

Når Linux-kjernen lastes ved oppstart, lages det et filsystem i mappen:

/proc

I denne mappen lagres innstillinger og parametre som brukes av kjernen. Man kan lese og endre disse parametrene ved å lese eller skrive til filer i undermappen:

/proc/sys

For eksempel ligger det en fil i mappen **/proc/sys/wm** som heter **swappiness**. Denne filen inneholder et parameter for hvor ofte kjernen skal skrive til swap-filen - mellom 0 (sjeldnest) og 100 (oftest). Du kan lese gjeldende innstilling ved å lese filen:

$ cat /proc/sys/vm/swappiness
$ 60

og du kan endre innstillingen ved å skrive til filen:

$ sudo echo "30" > /proc/sys/vm/swappiness

Dette setter parameteret til 30, som gjør at kjernen skriver sjeldnere til swap-området. Merk at endringen ikke beholdes ved restart av kjernen

ls - list innholdet i en mappe

ls er antagelig den kommandoen du vil bruke mest. ls er en forkortelse for *list* og kommandoen lister opp filer og undermapper i den mappen du befinner deg i - dvs i din *working directory* (kommandoen ***pwd*** viser deg hvilken mappe dette er).

ls kan brukes med ett eller flere av følgende parametre. (Merk at man alltid skriver en bindestrek før parametrene):

$ ls -l	Lister filer og undermapper i detaljert (langt) format
$ ls -lh	Lister filer og undermapper i detaljert og "human readable format", som f.eks. å viser filstørrelser i Megabytes istedenfor bytes
$ ls -a	Lister alle filer, inkludert skjulte filer
$ ls -t	Lister filer sortert etter når de sist ble endret
$ ls -S	Lister filer sortert etter størrelse
$ ls -r	Lister filer sortert i omvendt (reversert) rekkefølge

stat - list status for filer og filsystem

stat lister opp informasjon om en eller flere filer eller filsystemer.

OPTIONS

-L, --dereference
follow links

-f, --file-system
display file system status instead of file status

-c --format=FORMAT
use the specified FORMAT instead of the default; output a newline after each use of FORMAT

--printf=FORMAT

like --format, but interpret backslash escapes, and do not output a mandatory trailing newline; if you want a newline,
include \n in FORMAT

-t, --terse
print the information in terse form

Eksempler:

```
$ stat log.txt
  File: log.txt
  Size: 114             Blocks: 8          IO Block: 4096   regular file
  Device: 902h/2306d      Inode: 105124502   Links: 1
  Access: (0664/-rw-rw-r--)  Uid: ( 1000/   terje)   Gid: ( 1000/   terje)
  Context: system_u:object_r:user_home_t:s0
  Access: 2020-11-25 02:44:21.568729138 +0100
  Modify: 2020-04-27 16:11:58.000000000 +0200
  Change: 2020-10-03 13:16:17.657501333 +0200
  Birth: -
$ stat -t log.txt
log.txt 114 8 81b4 1000 1000 902 105124502 1 0 0 1606268661 1587996718
1601723777 0 4096 system_u:object_r:user_home_t:s0
$ stat -f log.txt
  File: "log.txt"
  ID: 9f5d5b62d777948e Namelen: 255     Type: ext2/ext3
  Block size: 4096       Fundamental block size: 4096
  Blocks: Total: 476160368  Free: 318903772  Available: 294698690
  Inodes: Total: 121012224  Free: 119242500
```

wc - tell antall linjer, ord og tegn i en tekst eller fil

wc (word count) gir oss antall linjer, ord og tegn i en pipe eller en tekstfil.

Parametere

-l kun antall linjer
-w kun antall ord
-c kun antall tegn

Eksempler på bruk:

```
$ wc war-and-peace.txt
  63846  562489 3266164 war-and-peace.txt
```

```
$ wc -l war-and-peace.txt
  63846  war-and-peace.txt
$ wc -w war-and-peace.txt
  562489  war-and-peace.txt
$ wc -c war-and-peace.txt
  3266164 war-and-peace.txt
$ echo "Dette er en tekst" | wc -c
18
$ echo -n "Dette er en tekst" | wc -c
17
```

MERK: echo legger til et linjeskift (\n), som kan fjernes med flagget **-n**

cd - change directory

cd (change directory) - Bytter til en annen mappe

Bruk:
cd <sti til ny mappe>. Stien kan være absolutt eller relativ.

Eksempler:

```
$ cd .. (bytter til mappen som ligger to nivåer opp).

$ cd (bytter til hjemmemappen din)

$cd ~/Musikk (bytter til mappen Musikk i hjemmemappen din)

$ cd - (bytter til forrige mappe du var i)

$ cd / (bytter til rot-mappen i filsystemet)
```

mkdir - Oppretter en ny mappe(make directory)

Bruk:

$ mkdir Bilder (Oppretter mappen *Bilder*)

$ mkdir -p mappe1/mappe2/mappe3 (Oppretter mappe3 og også mappe2 og mappe2 hvis de ikke finnes fra før

rmdir - Sletter en tom mappe

Bruk:

$ rmdir Dokumenter (sletter mappen Dokumenter, men bare hvis mappen er tom)

$ rmdir -p mappe1/mappe2/mappe3 (Sletter mappe1 og mappe2 og mappe3 hvis de er tomme

rm (remove) - Sletter en eller flere mapper eller filer

rm er kommandoen for å slette filer, men kan også slette hele mapper og undermapper.

Eksempler:

```
$ rm fil.txt (sletter filen fil.txt)

$ rm -i fil.txt (spør om du virkelig vil slette filen fil.txt, sletter
hvis du bekrefter. -i står for interactiv)

$ rm -rf Dokumenter (sletter hele mappen Dokumenter, inkludert mappens
filer og undermapper, uten flere spørsmål)

$ rm -I *.txt (sletter alle dokumenter som har navn som slutter med .txt,
men ber om bekreftelse etter å ha slettet tre dokumenter. -I er en svakere
beskyttelse enn -i, som ber om bekreftelse for hver fil som skal slettes.)
```

cp (copy) - Kopierer filer og mapper

Bruk:

```
cp <kilde> [<sti>/]<kopi>
```

Eksempler:

```
$ cp fil.txt ..
```

- kopierer fil.txt til mappen ett nivå opp

```
$ cp -r Mp3/ Musikk/
```

- kopierer mappen Mp3 til mappen Musikk, inkludert alle undermapper og filer i mappen Mp3. -r står for *recursive.*

Merk at -r endrer fil-eierskap til den som utfører kopieringen. For å beholde originale filegenskaper, bruk:

```
$ cp -a       (som også kopierer rekursivt)
```

```
$ cp -u fil.doc /home/petter/dokumenter
```

- kopierer filen fil.doc til mappen /home/petter/dokumenter, men bare hvis filen ikke finnes der fra før, eller hvis filen er nyere enn den som finnes der fra før. -u står for *update*

mv (move) - Flytter eller gir nytt navn til filer eller mapper

Merk: Linux har ingen egen kommando for å endre navn, men bruker mv til å gjøre denne operasjonen.

Bruk:

```
$ mv filnavn ..
```

- flytter *filnavn* en mappe opp i filstrukturen

```
$ mv gammeltnavn nyttnavn
```

- filen *gammeltnavn* heter nå *nyttnavn*

```
$ mv /home/petter/gammeltnavn /home/petter/video/nyttnavn
```

- flytter og gir nytt navn til filen *gammeltnavn*

ln - lag hard eller symbolsk lenke

Forskjellen mellom en hard og en symbolsk lenke er i korte trekk at en symbolsk lenke peker til en fil, mens en hard lenke peker til filens inode. Når man oppretter en fil, lages det 1 hard lenke til filen. Oppretter man en ny hard lenke er det 2 likeverdige harde lenker til filen. Når den siste harde lenken er slettet, slettes også filens inode (den blir *unlinked*). Sletter man en fil, vil den symbolske lenken ikke lenger virke, men en hard lenke vil fortsatt virke, siden den peker til filens inode.

En hard lenke er en mindre fleksibel løsning enn en symbolsk lenke (se nedenfor), og kan ikke brukes på tvers av filsystemer.

ln - Lag en hard lenke (snarvei) til en fil eller mappe.

Bruk:

```
$ ln /home/petter/fil.txt /home/petter/Skrivebord/fil.txt
```

(lager en lenke til filen fil.txt. Lenken ligger i mappen Skrivebord)

ln -s - Lag en symbolsk eller soft lenke (snarvei) til en fil eller mappe.

Bruk:

```
$ ln -s /home/petter/fil.txt /home/petter/Skrivebord/fil.txt
```

(lager en symbolsk lenke til filen fil.txt. Lenken ligger i mappen Skrivebord)

df og du - vis størrelsen til partisjoner og mapper

df

df - Viser partisjoner og hvor mye lagringsplass de har, samt hvor mye som er brukt og ledig.

Bruk:

```
$ df -h
```

(viser ledig plass "human readable format", dvs. Gigabytes, Megabytes etc.)

```
$ df -l
```

(viser kun ledig plass på lokale filsystemer)

du

du - Viser hvor mye lagringsplass som blir brukt av den aktuelle mappen og dens undermapper

Bruk:

```
$ du -h
```

(lister i *"human readable format"*)

```
$ du -s
```

(summerer opp for hver mappe)

```
$ du -c
```

(viser en totalsum)

find - søk etter filer og mapper

find er et kraftig verktøy for å søke etter mapper og filer. I motsetning til locate (mlocate eller slocate på noen systemer) bruker ikke find en indeksert database, men søker gjenom filsystemet i sanntid. Dette kan ta litt tid, men til gjengjeld har find flere muligheter til bl.a. å endre filene det søkes etter.

```
Syntaks:  find [-H] [-L] [-P] [-D] [-O]
               [bane]
               [ -type [-f] [-d]  ]
               [ -name ]
               [ -amin, -cmin, -mmin [-] [+] <antall minutter>]
               [ -size [-] [+] <antall> [b] [k] [M] [G] ]
```

De tre første parametrene: -H, -L eller -P angir om søket skal følge symbolske lenker eller ikke, det vil si, hvis det dukker opp en symbolsk lenke i søkeresultatet, f.eks.:

Eksempler på bruk:

Søke etter navn, type, filstørrelse og tid

```
$ find -name "*.txt" (søker i den mappen du er i og dens undermapper -
etter filer og mapper som ender med .txt)
$ find / -type f -name "core" (søker i alle mapper i filsystemet etter
filer som heter core)
$find -mmin -1   (søker etter filer og mapper i den mappen du står i, som
er endret for mindre enn ett minutt siden).
$ man find (lister opp alle valgene find har)
```

Søke etter mapper/filer og endre de vi finner

med **find** kan vi også utføre operasjoner på søkeresultatet, som for eksempel endre eieskap eller tilgangsrettigheter, slette filer og mapper etc.

Endre søkeresultatet med -exec

```
$ find /home/bruker/public_html -type d -exec chmod 755 {} \;   (finner
alle mapper i brukerens apache-hjemmemappe m/undermapper og gir dem
tilgang 755)
$ find /home/bruker/public_html -type f -exec chmod 644 {} \;  (finner
alle filer i brukerens apache-hjemmemappe m/undermapper og gir dem tilgang
644)
$ find /home/bruker/public_html -name .htaccess -exec rm {} \;   (fjerner
alle .htaccess-filer i brukerens apache-hjemmemappe og dens undermapper)
```

Endre søkeresultatet med xargs

```
MERK: Istedenfor -exec kan vi også lage et rør (med | ) som sender
resultatet til xargs etterfulgt av en kommando, f.eks. slik:
$ find /home/bruker/public_html -type d | xargs chmod 755  (finner alle
mapper i brukerens apache-hjemmemappe m/undermapper og gir dem tilgang
755)
$ find /home/bruker/public_html -type f | xargs chmod 644  (finner alle
filer i brukerens apache-hjemmemappe m/undermapper og gir dem tilgang 644)
```

chown - endre eierskap til mapper og filer

chown er kommandoen for å endre eierskap til mapper og filer.

Syntaksen for **chown** er denne:

```
$ sudo chown <brukernavn> [:<gruppenavn>]  mappe[r] og/eller fil[er]
```

Eksempler:

$ sudo chown ole:apache test.html (setter eier til ole og gruppe til apache for filen test.html)

$ sudo chown petter pettersmappe (setter eier til petter for mappen pettersmappe, men ikke innholdet i mappen (endrer ikke gruppe)

$ sudo chown petter pettersmappe/* (setter eier til petter for innholdet i mappen pettersmappe, men ikke selve mappen, eller innholdet i undermapper (endrer ikke gruppe)

$ sudo chown -R :felles fellesmappe (setter gruppe til felles for mappen fellesmappe og alt den inneholder)

chmod - endre tilgangsrettigheter til mapper og filer

Mapper og filer blir opprettet med default tilgangsrettigheter. Disse kan man se (og evt. endre) med kommandoen **umask**, og de kan konfigurereres ved å sette en umask-verdi i filen /etc/pofile, eller enda bedre i en egen fil i mappen /etc/profile.d/ - eller eventuelt lokalt i brukerens ~/.bashrc - fil.

chmod er kommandoen for å endre tilgangsrettighetene til mapper og filer ***etter*** at de er opprettet.

Først litt om rettighetene. Linux deler brukerne i tre hoveddeler:

1. Eier - den brukeren som står som eier av mappen eller filen (angitt med bokstaven **u** for user)
2. Gruppe - den gruppen som mappen eller filen tilhører (angitt med bokstaven **g** for group)
3. Alle andre - alle brukere som ikke hører inn under 1. eller 2. (angitt med bokstaven **o** for other)

Tilgangsrettighetene er også delt i tre deler:

1. Lese-tilgang - som angis ved tallet **4** eller bokstaven **r**
2. Skrive-tilgang - som angis ved tallet **2** eller bokstaven **w**
3. Kjøre-tilgang - som angis ved tallet **1** eller bokstaven **x**

MERK: For mapper er "kjøretilgangen" definert som tilgang til å åpne mappen. For filer er kjøretilgang definert som tilgang til å kjøre filen som et program

Tallene som definerer tilgangsnivåene er laget slik at de gir unike kombinasjoner:

- Kun kjøretilgang = 1,
- Kun skrivetilgang = 2
- Kjøre- og skrivetilgang = 3,
- Kun Lesetilgang = 4
- Lese- og kjøretilgang gir 4+1=5,

- Lese- og skrivetilgang gir 4+2 = 6.
- Alle tilganger gir 4+2+1= 7

Det vil si at vi med ett siffer kan angi riktig kombinasjon av rettigheter. Vi kan sette tilgangsrettigheter for de tre brukergruppene, og dermed får vi en kombinasjon av tre siffer, en for eier, en for gruppe og en for alle andre. Rettighetene kan angis med tall eller bokstaver.

Her er noen eksempler på hvordan endring av tilgangsrettigheter ser ut i praksis:

chmod med tall

$ chmod 755 mappe1 (Eier har alle rettigheter, gruppen og alle andre har tilgang til å åpne mappen og lese innholdet)

$ chmod 700 mappe2 (Eier har alle rettigheter, gruppen og alle andre har ingen tilgang)

$ chmod 775 mappe3 (Eier og gruppen har alle rettigheter, alle andre har tilgang til å åpne mappen og lese innholdet)

$ chmod 644 fil1.txt (Eier har lese- og skrivetilgang, gruppen og alle andre har kun lesetilgang)

$ chmod 500 program1 (Eier har lese- og kjøretilgang, alle andre har ingen tilganger)

$ **find fellesmappe/ -type f | xargs chmod 664** (Finner alle filer i mappen fellesmappe, og dens undermapper, og setter ilgangen til lese+skrive for eier og gruppe, og kun lese for andre)

chmod med bokstaver

$ chmod +x program2 (legger til kjøretilgang for alle brukere)

$ chmod u+x program1 (legger til kjøretilgang for eieren av program1)

$ chmod o-w *.txt (fjerner skrivetilgang for andre enn eier og gruppe til alle filer i mappen med navn som slutter på ***.txt***)

OBS! Sett aldri alle mapper og filer til 777 på en Linux-maskin (feks. med sudo chmod -R 777 /) - da vil den slutte å fungere. Det er viktige systemfiler som ikke vil kjøre med en så usikker tilgang.

Komprimering og dekomprimering av filer og mapper

Linux har flere ypperlige kommandolinjeverktøy for pakking og komprimering / dekomprimering av filer og mapper med filer.

Her er en kort oversikt:

Komprimering med zip

zip er en nyttig kommando siden den lager arkiver med komprimerte filer som er kompatible med bl.a. Microsoft Windows.

Eksempler på bruk av zip/unzip:

$ zip *filnavn.zip filnavn*
- oppretter arkivet *filnavn.zip* og kopierer en komprimert versjon av filen *filnavn inn i arkivet*.

$ zip arkivnavn.zip *
- oppretter arkivet "arkivnavn.zip", komprimerer aller filer i gjeldene mappe og kopierer dem inn i arkivet.

$ unzip *filnavn.zip*
- oppretter filen *filnavn* som en dekomprimert versjon av filen *filnavn.zip*.

Komprimering med gzip

gzip er det mest brukte zip-formatet på Linux, og brukes gjerne sammen med arkiv-programmet *tar* (se nedenfor)

gunzip er kommandoen for å dekomprimere filer som er komprimert med gzip

Eksempler på bruk av gzip/gunzip:

Filer
$ gzip *filnavn*
- gzipper filen *filnavn* og gir den det nye navnet: *filnavn.gz*
$ gzip -k *filnavn*
- gzipper filen *filnavn* og gir den det nye navnet: *filnavn.gz*, men beholder originalfilen (k = keep)
$ gunzip *filnavn.gz*
- dekomprimerer filen *filnavn.gz* og gir den navnet *filnavn*

Mapper
$ gzip *-rv mappenavn*
- gzipper alle filene i mappen *mappenavn* og gir dem nye navn med *.gz* endelse
$ gunzip *-rv mappenavn*
- dekomprimerer alle filene i mappen *mappenavn* og fjerner endelsen *.gz*

Komprimering med bzip2

bzip2 gir en meget god komprimering, og blir mer og mer brukt. Dekomprimering gjøres med kommandoen **bunzip2. bzip2/bunzip2** brukes på tilsvarende måte som gzip/gunzip:

Eksempler på bruk av bzip2/bunzip2:

Filer
$ bzip2 *filnavn*
- bzipper filen *filnavn* og gir den det nye navnet: *filnavn.bz2*
$ bunzip2 *filnavn.bz2*
- dekomprimerer filen *filnavn.bz2* og gir den navnet *filnavn*

Mapper
$ bzip2 *mappenavn/* (merk forskjellen fra gzip)*
- bzipper alle filene i mappen *mappenavn* og gir dem nye navn med *.bz2* endelse
$ bunzip2 *mappenavn/* (merk forskjellen fra gzip)*
- dekomprimerer alle filene i mappen *mappenavn* og fjerner endelsen *.bz2*

Pakking og komprimering med tar

tar (Tape ARchive) er et kraftig verktøy som er mye brukt i Linux-verden, både til å samle filer i et arkiv, og til å komprimere/dekomprimere filer og mapper. Dette kan gjøres med samme kommando.

Eksempler på bruk av tar:

$ tar -cf arkivnavn.tar filnavn1 filnavn2 filnavn3
- samler filene *filnavn1 filnavn2 filnavn3* i arkivet *arkivnavn.tar*

$ tar -xf arkivnavn.tar
- trekker ut filene *filnavn1 filnavn2 filnavn3* fra arkivet *arkivnavn.tar*

$ tar -czf arkivnavn.tar.gz filnavn1 filnavn2 filnavn3
- samler filene *filnavn1 filnavn2 filnavn3* i det komprimerte arkivet *arkivnavn.tar.gz*

$ tar -xzf arkivnavn.tar.gz
- trekker ut og dekomprimerer filene *filnavn1 filnavn2 filnavn3* fra det komprimerte arkivet *arkivnavn.tar.gz*

$ tar -tf arkiv.tar
- lister opp innholdet i arkivet *arkiv.tar*

$ tar -tzf arkiv.tar.gz
- lister opp innholdet i det komprimerte arkivet *arkiv.tar*.gz

$ tar -rf arkiv.tar filnavn4
- legger til filen *filnavn4* i slutten av arkivet *arkiv.tar*

Sjekk gjerne tar-manualen for flere valgmuligheter:

$ man tar

rsync - synkronisering av filer mellom maskiner

rsync ble lansert i 1996 som et terminal-program for å synkronisere mapper eller hele trestrukturer mellom ulike steder på en maskin, eller mellom to maskiner. **rsync** blir ofte brukt til f.eks. sikkerhetskopiering.

En av de viktigste egenskapene ved **rsync** er at den bruker *checksums* til å sjekke filene - om alle, ingen eller noen blokker er endret siden sist. Kun de blokkene som er endret blir overført. Det sparer tid og båndbredde. Hvis første gangs synkronisering tar timer, vil de påfølgende kunne ta minutter, avhengig av hvor mye som er endret.

Eksempler:

```
$ rsync -a /home/terje/mappe1/ terje@itfakultetet.no:mappe1/
```

-a står for **"archive mode",** som bevarer symbolske lenker, eierskap og tilgangsrettigheter mm.

```
$ rsync -a /home/terje/utvikling2/synctest/ terje4:synctest/
```

Det siste eksemplet forutsetter en .ssh/config - fil hvor terje4 er definert med HostName, User og Port

ØVELSE
Endre filnavn på mange filer samtidig med find og xargs

I denne lille øvelsen vil vi søke opp alle konfig-filene i en mappe, definert som at de har filendelsen **.cfg** og gi dem nytt navn ved å legge **.gammel** til filnavnet.

Her er syntaksen til find med xargs til å endre navnene

```
$ find </sti/til/mappe/> -name "*.cfg" --print0 | xargs --null -I{} mv {}
{}.gammel
```

Gjennomføring:

```
1) La oss først lage en mappe og gå inn i den:
$ mkdir config
$ cd config
2) Så lager vi noen tomme config-filer og sjekker at de er laget:
$ touch 1.cfg 2.cfg 3.cfg 4.cfg
$ ls
1.cfg  2.cfg  3.cfg  4.cfg
3) så bruker vi find til å gi dem nytt navn, og sjekker at det gikk bra
$ find -name "*.cfg" -print0 | xargs --null -I{} mv {} {}.gammel
$ ls
1.cfg.gammel  2.cfg.gammel  3.cfg.gammel  4.cfg.gammel
```

Forklaring:

1. **find** søker opp alle filer med filendelse **.cfg**
2. -print0 gir beskjed til find om å ikke printe linjeskift for hver fil den finner, men et null-tegn. Default for find er -print, som lager linjeskift etter hver funnet fil
3. Send filnavnene til xargs med |
4. Parameteret --null (evt -0) forteller xargs at hvert element er avsluttet med null-tegn og ikke whitespace
5. Parameteret -I{} sier at vi skal utføre en erstatning av tegn og mv {} sier flytt alle elementene til {}.gammel. ({} sybmoliserer alle elementene som xargs mottar)

Kapittel 3
Pakke- og brukerhåndtering

Pakkehåndtering fra kommandolinjen

Ulike Linux distroer bruker ulike applikasjonspakker (installerbare programmer). En *pakke* i dette tilfelle er en måte å håndtere hvordan applikasjoner innstalleres i et system, hvordan håndtere dets avhengigheter av andre pakker osv. Pakken inneholder med andre ord programmet som skal installeres samt informasjon om hvilke andre programmer som må være installert for at programmet skal kunne kjøres og instruksjoner om å installere disse hvis de ikke er installert allerede.

Installere og oppgradere DEB-pakker

DEB

DEB er Debians pakkesystem (Debian Packaging system). Deb fil-formatet blir brukt av Debian, Ubuntu, Mint og mange andre distroer.

APT

APT står for *Advanced Package Tool* og inneholder programmet **apt** (tidligere apt-get), en enkel måte og laste ned og installere pakker fra flere ulike kilder via kommandolinjen. I motsetning til dpkg, forstår ikke apt .deb filer, men installerer pakkene etter navn, og aptkan bare installere .deb-pakker fra kilder spesifisert på forhånd i filen **/etc/apt/sources.list**. apt bruker dpkg direkte etter at .deb-pakkene er lastet ned fra kildene.

Noen vanlige måter å bruke ***apt / apt-get*** på:

- Først lønner det seg å oppdatere pakkelistene - listene over tilgjengelig programvare - slik at systemet ditt vet hvilke nye versjoner som lagt til siden sist. Denne kommandoen gjør dette:

```
$ sudo apt update
```

(Dette bør gjøres før hver gang du oppdaterer)

- For å oppgradere alle installerte programmer på PCen din til siste tilgjengelige versjon, uten å slette noe eller legge til noe nytt kjør denne kommanden:

```
$ sudo apt upgrade
```

- For å oppgradere alle programmene på PCen og, hvis det er nødvendig for oppgraderingen, installere ekstra pakker eller fjerne eksisterende pakker, kjør denne kommandoen:

```
$ sudo apt full-upgrade  (eller for eldre versjoner: dist-upgrade)
```

- (Med kommandoen `upgrade` beholdes den eksisterende versjonen av et program hvis oppgradering innebærer at en tilleggspakke må installeres for å tilfredsstille nye avhengigheter. Med kommandoen `full-upgrade` eller dist-upgrade vil pakken bli oppgradert og tilleggspakken(e) installert og evt gamle pakker fjernet)
- For å installere programmet *foo* og alle tilleggsprogrammer den er avhengig av for å fungere, kjør denne kommandoen:

```
$ sudo apt install foo
```

- For å avinstallere programmet *foo* og fjerne det fra systemet, men la programmets konfigusasjonsfiler være igjen, kjør denne kommandoen:

```
$ sudo apt remove foo
```

- For å avinstallere programmet *foo* og slette programmets konfigurasjonsfiler, kjør kommandoen:

```
$ sudo apt --purge remove foo
```

merk at du må være innlogget som ***root*** - evt. tilføye ***sudo*** før kommandoene for å kunne installere, oppgradere eller avinstallere programpakker.

Apt inkluderer også verktøyet **apt-cache** som du kan bruke til å søke etter programpakker i pakkelistene. Du kan bruke det til å finne programmer som inneholder en viss funksjonalitet gjennom enkle tekstsøk eller mer avanserte søk med regulære uttrykk. I nyere versjoner av Ubuntu (14.04 og senere) kan du også bruke **apt search.**

Her er noen vanlige bruksområder for **apt / apt-cache:**

- For å finne pakker som inneholder ordet *word*:

```
$ sudo apt-cache search word
```

- For å vise detaljert informasjon om en pakke:

```
$ sudo apt-cache show package
```

- For å vise hvilke andre pakker en pakke avhenger av:

```
$ sudo apt-cache depends package
```

- For å vise detaljert informasjon om hvilke versjoner av en pakke som er tilgjengelig og informasjon om pakker som er avhengige av denne pakken:

```
$ sudo apt-cache showpkg package
```

For mer informasjon, installer apt og les apt-get(8), sources.list(5), og installer pakken apt-doc og les /usr/share/doc/apt-doc/guide.html/index.html.

dpkg

Dette er den opprinnelige pakkehåndtereren for debian-pakker. dpkg kan kjøres med mange ulike opsjoner. Noen vanlige valg er:

- Finn ut hvilke mulige opsjoner programmet har:
  `dpkg --help`.
- Vis informasjonsfilen (og annen info) for en pakke :
  `dpkg --info foo_VVV-RRR.deb`
- Installer en pakke (inkludert å pakke opp og konfigurere) på hardiskens filsystem:
  `dpkg --install foo_VVV-RRR.deb`.
- Pakk opp (men ikke konfigurer) en Debian pakke til harddiskens filsystem:
  `dpkg --unpack foo_VVV-RRR.deb`.
  Merk at dette ofte ikke er nok til å kunne kjøre programmet. Denne kommandoen fjerner tidligere installerte versjoner av programmet og kjører programmets pre-installasjons-skript.
- Konfigurere en pakke som allerede er pakket ut:
  `dpkg --configure foo`.
  Denne kommandoen kjører postinstallasjons-skriptet til pakken, og den oppdaterer også pakkens

fillister. Legg merke til at 'configure'-kommandoen etterfølges av et pakkenavn, (f.eks., foo), *ikke* navnet til Debian arkivfilen (f.eks., foo_VVV-RRR.deb).

- Trekke ut en enkelt fil kalt "blurf" (eller en gruppe filer kalt "blurf*") fra et Debian arkiv:
  `dpkg --fsys-tarfile foo_VVV-RRR.deb | tar -xf - blurf*`
- Fjerne/avinstallere en pakke (men ikke pakkens konfigurasjonsfiler):
  `dpkg --remove foo`.
- Fjerne/avinstallere en pakke (inkludert pakkens konfigurasjonsfiler):
  `dpkg --purge foo`.
- Liste opp installasjons-statusen til pakker som inneholder strengen (eller regulære uttrykket) "foo*":
  `dpkg --list 'foo*'`.

Installere og oppgradere RPM-pakker

Ulike Linux distroer bruker ulike applikasjonspakker (installerbare programmer). En *pakke* i dette tilfelle er en måte å håndtere hvordan applikasjoner innstalleres i et system, hvordan håndtere dets avhengigheter av andre pakker osv. Pakken inneholder med andre ord programmet som skal installeres samt informasjon om hvilke andre programmer som må være installert for at programmet skal kunne kjøres og instruksjoner om å installere disse hvis de ikke er installert allerede.

RPM - Pakkehåndtering med Red Hat Package Manager

RPM

RPM er RedHats pakkesystem (Redhat Package Manager). Fil-formatet **rpm** blir brukt av RHEL, Centos, Fedora, Suse og flere andre distroer.

YUM / DNF

YUM står for *Yellowdog Updater, Modified* og inneholder programmet **yum,** en enkel måte og laste ned og installere pakker fra flere ulike kilder via kommandolinjen.
DNF står for "Dandified Yum", og ble introdusert i Fedora i 2013, som en arvtager etter yum. Fra og med RHEL/Centos 8 er **dnf** default pakkehåndterer for RedHat distroer.

I motsetning til rpm, forstår ikke yum eller dnf .rpm-filer, men installerer pakkene etter navn, og kan bare installere .rpm-pakker fra kilder spesifisert på forhånd i mappen **/etc/yum.repos.d** Yum og dnf bruker rpm direkte etter at .rpm-pakkene er lastet ned fra kildene.

Noen vanlige måter å bruke ***yum / dnf*** på:

- For å oppgradere alle installerte programmer på PCen din til siste tilgjengelige versjon, uten å slette noe eller legge til noe nytt kjør denne kommanden:

```
$ sudo yum/dnf upgrade
```

- For å installere programmet *foo* og alle tilleggsprogrammer den er avhengig av for å fungere, kjør denne kommandoen:

```
$ sudo yum/dnf install foo
```

- For å avinstallere programmet *foo* og fjerne det fra systemet, kjør denne kommandoen:

```
$ sudo yum/dnf remove foo
```

merk at du må være innlogget som ***root*** - evt. tilføye ***sudo*** før kommandoene for å kunne installere, oppgradere eller avinstallere programpakker.

- For å søke etter pakker som inneholder ordet *word*:

```
$ sudo yum/dnf search word
```

- For å liste opp alle pakke-brønner som er installert:

```
$ sudo yum/dnf repolist
```

Tilgjengelige kommandoer for dnf:

· alias
· autoremove
· check
· check-update
· clean
· deplist
· distro-sync
· downgrade
· group
· help
· history

- info
- install
- list
- makecache
- mark
- module
- provides
- reinstall
- remove
- repoinfo
- repolist
- repoquery
- repository-packages
- search
- shell
- swap
- updateinfo
- upgrade
- upgrade-minimal
- upgrade-to

rpm

Dette er den opprinnelige pakkehåndtereren for rpm-pakker. programmet **rpm** kan kjøres med mange ulike parametere. Noen vanlige valg er:

- Finn ut hvilke mulige valg programmet har:
  `rpm --help`.

Her er et lite utdrag fra manualen til rpm som viser de vanligste kommandoene og parameterene:

QUERYING AND VERIFYING PACKAGES:
rpm {-q|--query} [select-options] [query-options]
rpm --querytags
rpm {-V|--verify} [select-options] [verify-options]

INSTALLING, UPGRADING, AND REMOVING PACKAGES:
rpm {-i|--install} [install-options] PACKAGE_FILE ...
rpm {-U|--upgrade} [install-options] PACKAGE_FILE ...
rpm {-F|--freshen} [install-options] PACKAGE_FILE ...
rpm {--reinstall} [install-options] PACKAGE_FILE ...
rpm {-e|--erase} [--allmatches] [--justdb] [--nodeps] [--noscripts]
[--notriggers] [--test] PACKAGE_NAME ...