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:

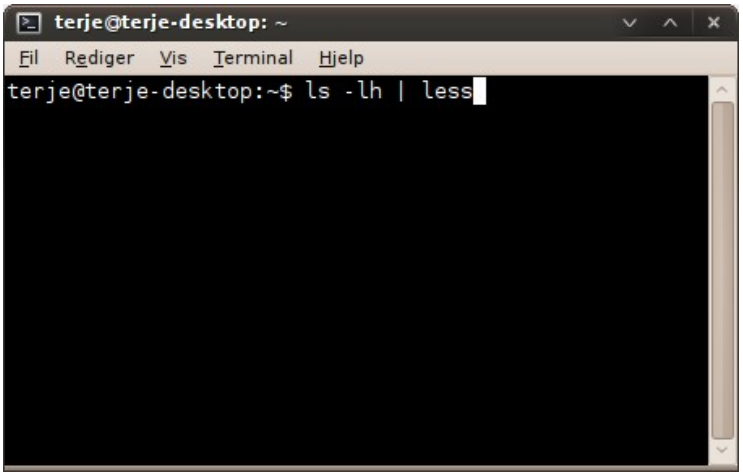
https://www.wikihow.com/Install-Ubuntu-on-VirtualBox

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:

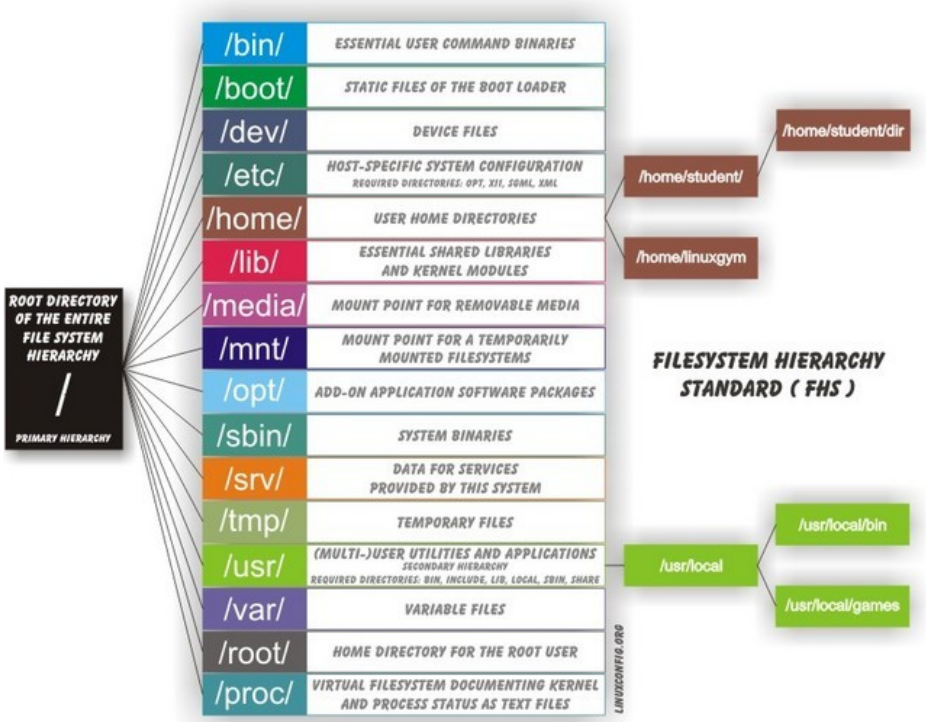
set number
colorscheme darkblue

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 ...

Brukerhåndtering fra kommandolinjen

Det finnes gode grafiske verktøy for brukerbehanding, men det er ikke alltid man har tilgang til dem, f.eks. hvis man skal jobbe på en server med kun ssh-tilgang (ssh = secure shell). Heldigvis er det enkelt å administrere brukere, brukergrupper og tillatelser via kommandolinjen, som er det vi skal se på i denne teksten.

Merk: Du må være rot-bruker eller bruke sudo for å håndtere brukere og brukergrupper.

Informasjon om en bruker

Du kan få grunnleggende informasjon om en bruker på flere måter. her er noen av dem:

Før du endrer gruppetilhøriheter o.l. kan du se hvilken id en bruker har og hvilke grupper en bruker er medlem av gjennom kommanoen **id**, slik:

```
$ id <username>
```

For eksempel:

```
$ id terje
uid=1000(terje) gid=1000(terje) groups=1000(terje),10(wheel),54335(felles)
```

Hvis du bare vil se gruppene en bruker er medlem av, kan du også bruke kommandoen groups, slik:

```
$ groups terje
terje : terje wheel felles
```

Legge til en ny bruker

For å legge til en ny bruker fra kommandolinjen, kan man bruke verktøyet ***adduser*** slik:

```
$ sudo  adduser petter
Oppretter bruker «petter» ...
Oppretter ny gruppe «petter» (1002) ...
Oppretter ny bruker «petter» (1002) med gruppe «petter» ...
Oppretter hjemmemappe «/home/petter» ...
Kopierer filer fra «/etc/skel» ...
Angi nytt UNIX-passord:
Bekreft nytt UNIX-passord:
passwd: passordet ble oppdatert
```

```
Changing the user information for petter
Enter the new value, or press ENTER for the default
 Full Name []: Petter Testbruker
 Room Number []:
 Work Phone []:
 Home Phone []:
 Other []:
Is the information correct? [Y/n]
#
```

Som det fremgår av tilbakemeldingene fra adduser-programmet ovenfor, opprettes en hjemmemappe med brukerens navn i mappen «/home». Hit kopieres et sett med standardfiler fra mappen «/etc/skel» (skel står for "skeleton", og alle filer vi legger i denne mappen blir kopiert til hver ny bruker).

Endre en eksisterende bruker

Endre en eksisterende bruker kan gjøres via kommandoen **usermod**:

Her er parametrene for å endre en bruker:

-c, --comment NEW_NAME setter ny verdi for brukerens fulle navn
-d, --home HOME_DIR setter ny hjemmemappe for brukeren
-e, --expiredate EXPIRE_DATE setter kontoens utløpsdato til EXPIRE_DATE
-f, --inactive INACTIVE setter passordet etter kontoens utløp til INACTIVE
-g, --gid GROUP setter GROUP som ny primærgruppe
-G, --groups GROUPS setter ny liste av tilleggsgrupper brukeren er med i.
-a, --append legger brukeren til tilleggsgruppene GROUPS, spesifisert ved -G i tillegg til eksisterende gruppemedlemskap
-h, --help Viser hjelpeteksten
-l, --login NEW_LOGIN setter nytt login navn
-L, --lock låser brukerkontoen
-m, --move-home flytter hjemmemappen til ny mappe, brukes bare sammen med -d
-p, --password PASSWORD setter nytt passord til en kryptert versjon av PASSWORD
-s, --shell SHELL setter nytt login shell for brukeren
-u, --uid UID setter ny bruker-ID for brukeren
-U, --unlock låser opp brukerkontoen

Eksempler på bruk av usermod:

```
$ sudo usermod -c "Nytt fullt navn" <username>
```

```
$ sudo usermod -l "Nytt brukernavn" <username>
```

Merk: Har du tilgang til et grafisk brukergrensesnitt, kan du også bruke det grafiske programmet **user-admin** til å endre bruker-data**.** Du kan starte det fra applikasjonsmenyen eller fra terminalvinduet ved å kjøre kommandoen:

```
$ user-admin
```

Slette en bruker fra systemet

En bruker kan slettes fra systemet via kommandoen **userdel,** slik:

```
$ sudo userdel <username>
```

Merk: Dette sletter ikke brukerens hjemmemappe

For å slett brukerens hjemmemappe og lokal epost, bruk denne kommandoen:

```
$ sudo userdel -r <username>
```

Grupper

En gruppe er en samling brukerkontoer som opptrer som en enkelt enhet. Hvis en gruppe får tilgang til å utføre en handling, har alle gruppens medlemmer samme tilgang.

Her er noen nyttige kommandoer for å jobbe med Linuxgrupper:

- **groups** (lister opp hvilke grupper en bruker er medlem av)
- **groupadd** (oppretter en ny gruppe)
- **groupdel** (sletter en gruppe)
- **groupmod** (endrer en gruppe)
- **gpasswd -a <bruker> <gruppe>** (legg en bruker til en gruppe)
- **members <gruppenavn>** (list medlemmer i en gruppe - debian/ubuntu)
- **getent group <gruppenavn>** (list medlemmer i en gruppe - alle distroer)

Slik kan de brukes:

$ whoami (vis brukernavnet til den som skriver kommandoen)

petter (viser at brukernavnet er *petter*)

$ groups (vis hvilke grupper petter er medlem av)
petter users (viser at petter med i gruppene *petter* og *users*

$ groups per petter root (vis hvilke grupper *per*, *petter* og *root* er medlem av)
per: per users (viser gruppene *per* er medlem av)
petter: petter users (viser gruppene *petter* er medlem av)
root: root bin daemon sys adm disk wheel src (viser gruppene *root* er medlem av)

$ sudo groupadd felles Oppretter gruppen "felles"

$ sudo gpasswd -a petter felles
Legger brukeren petter til gruppen felles

$ members felles
petter

$ getent group felles
felles:x:1034:petter

/etc/passwd, /etc/shadow og /etc/group

/etc/passwd og **/etc/shadow** er to filer som inneholder informasjon om alle brukerne på en Linux-maskin. /etc/shadow krever rot-tilgang siden den inneholder brukernes krypterte passord.

/etc/group er en fil som inneholder informasjon om maskinens definerte grupper, inkludert hvilke brukere som er medlemmer i dem.

Eksempler på format:

1) /etc/passwd

 # cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
nobody:x:65534:65534:Kernel Overflow User:/:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
mssql:x:992:990::/var/opt/mssql:/bin/bash
terje:x:1000:1000::/home/terje:/bin/bash
backup:x:1001:1001::/home/backup:/bin/bash

postgres:x:26:26:PostgreSQL Server:/var/lib/pgsql:/bin/bash
mysql:x:991:989:MySQL server:/var/lib/mysql:/sbin/nologin
itfakultetet:x:1006:1006::/home/itfakultetet:/bin/bash
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
nginx:x:990:988:Nginx web server:/var/lib/nginx:/sbin/nologin
mongodb:x:1010:1010::/home/mongodb:/bin/bash
jenkins:x:989:987:Jenkins Automation Server:/var/lib/jenkins:/bin/false
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
dovecot:x:97:97:Dovecot IMAP server:/usr/libexec/dovecot:/sbin/nologin

2) /etc/shadow

$ cat /etc/shadow

sshd:!!:18191::::::
chrony:!!:18191::::::
rngd:!!:18443::::::
saslauth:!!:18535::::::
mssql:!!:18535::::::
terje:$6$7hNOgw1zdWfDMsLJ$iz5.illxtzSsJdKJQuPoh1c2Joj6Q7DN.Z71MRZR6S5XQtJ/cbc6fsc6rMzPAiHlwzQD2qnbaCkZuB8Z9Nj1j0:18535:0:99999:7:::
backup:6g0hhw1X/tL0oeHu9$cdGt9Zg5V1gZjVZO/.SnsPWI.vim2eSWPhgFjcKtQhtUksSJXC8xQ7ovrrvWCSKyhEXnlHOr94HixBmEaKqxr0:18535:0:99999:7:::

Strukturen til /etc/shadow består av disse feltene, med kolon som skilletegn:

```
root:$6$.n:12236:0:66669:7:::
[--] [----] [--] - [---] ----
|      |      |   |   |   |||+-----------> 9. Unused
|      |      |   |   |   ||+------------> 8. Expiration date
|      |      |   |   |   |+-------------> 7. Inactivity period
|      |      |   |   |   +--------------> 6. Warning period
|      |      |   |   +------------------> 5. Maximum password age
|      |      |   +----------------------> 4. Minimum password age
|      |      +--------------------------> 3. Last password change
|      +---------------------------------> 2. Encrypted Password
+----------------------------------------> 1. Username
```

1. **Username** : It is your login name.
2. **Password** : It is your encrypted password. The password should be minimum 8-12 characters long including special characters, digits, lower case alphabetic and more. Usually password format is set to `$id$salt$hashed`, The `$id` is the algorithm used On GNU/Linux as follows:
    1. **`$1$`** is MD5
    2. **`$2a$`** is Blowfish

    3. **$2y$** is Blowfish
    4. **5** is SHA-256
    5. **6** is SHA-512
3. **Last password change (lastchanged)** : Days since Jan 1, 1970 that password was last changed
4. **Minimum** : The minimum number of days required between password changes i.e. the number of days left before the user is allowed to change his/her password
5. **Maximum** : The maximum number of days the password is valid (after that user is forced to change his/her password)
6. **Warn** : The number of days before password is to expire that user is warned that his/her password must be changed
7. **Inactive** : The number of days after password expires that account is disabled
8. **Expire** : days since Jan 1, 1970 that account is disabled i.e. an absolute date specifying when the login may no longer be used.

3) /etc/group

mongod:x:981:
kurs1:x:54325:
kurs2:x:54326:
kurs3:x:54327:
kurs4:x:54328:
kurs5:x:54329:
kurs6:x:54330:
kurs7:x:54331:
kurs8:x:54332:
kurs9:x:54333:
kurs10:x:54334:
felles:x:54335:terje,kurs,kurs1,kurs2,kurs3,kurs4,kurs5,kurs6,kurs7,kurs8,kurs9,kurs10

Kapittel 4
4: SSH / SCP / SFTP

Installasjon og konfigurering av OpenSSH - secure shell server

Det viktigste administrasjonsgrensesnittet vårt mot serveren er gjennom SSH - Secure SHell. Så lenge vi kan logge oss inn på en SSH-konto med rot-tilgang til serveren har vi full kontroll over den. Noe av det første vi gjør er derfor å sørge for at denne tilgangen er tilstede, satt opp riktig og at den ikke kommer i hendene på uønskede inntrengere.

Installasjon

Dersom ***Open SSH server*** ikke ble valgt ved installasjon av Ubuntu-serveren, kan du installere OpenSSH - klient og server med følgende kommando:

```
$ sudo apt-get install openssh-client openssh-server
```

Konfigurering

SSH-serveren kan konfigureres ved å endre parametrene i konfigurasjonsfilen:

/etc/ssh/sshd_config

Sjekk gjerne manualen for konfigurering gjennom kommandoen:

```
$ man sshd_config
```

Merk: Lag en kopi av den originale konfig-filen før du endrer noe. Dersom du gjør en feil som hindrer SSH-serveren fra å fungere, kan det bli vanskelig å nå serveren fra nettverket.

Her er noen nyttige endringer du kan gjøre i konfig-filen:

- Endre portnummeret SSH-serveren lytter på. Dette er en effektiv måte å hindre "script kiddies" i å forsøke å hacke seg inn på serveren via SSH.
  port 22 er standard port for SSH, så du kan endre til noe over 1024, f.eks. slik:

  port 2222

- For å tillate innlogging med offentlig nøkkel:

  PubkeyAuthentication yes

- For å kunne kjøre programmer på serveren med grafisk brukergrensesnitt (GUI) fra en klient, kan du sette:

  X11Forwarding yes

  Dette fungerer dersom grafisk grensesnitt (X-server) er installert på serveren. Klienten logger seg på med flagget -X eller -Y og kan starte GUI-programmer som vises på klientens PC som om man satt ved serveren.

Merk: Hvis du endrer portnummeret til noe annet enn 22 må klienten logge seg inn med flagget -p etterfulgt av det oppsatte portnummeret. F.eks. slik:

$ ssh -p 2222 bruker@server.com

eller

$ ssh -p -X 2222 bruker@server.com

hvis man ønsker å kjøre GUI-baserte programmer fra serveren

Bruk av aliaser via SSHs konfigurasjonsfil

Først rediger eller lag en fil som heter **config** i din egen **.ssh**-mappe, f.eks slik:

$ vim ~/.ssh/config (eller bruk din favoritt-editor)

Filen skal inneholde dette oppsettet for hver alias du lager til en ssh-konto:

host aliasnavn
HostName server.domain.com

Port 5555
User username

F.eks. slik:

host hjemme
HostName server1.hjemme.no
Port 2222
User Petter

Når dette er lagret, kan du skrive:

```
$ ssh hjemme
```

istedenfor:

```
$ ssh -p 2222 petter@server1.hjemme.no
```

Passordløs innlogging

Innlogging med nøkler, uten passord

Først lager vi nøkler som vi kan bruke til innloggingen, slik:

```
$ ssh-keygen
```

La passordet stå blankt, slik at nøkkelen lages uten passord.

Så kopierer du den offentlige nøkkelen over til serveren, slik:

```
$ ssh-copy-id  -p <port> bruker@server.com
```

eller slik (old school):

```
$ cat .ssh/id_rsa.pub | ssh bruker@server.com 'cat >>
.ssh/authorized_keys'
```

Dette kopierer nøkkelen fra id_rsa.pub og legger den til filen **authorized_keys** på serveren (bytt ut med ditt brukernavn og servernavn)

Og det er alt som trengs for en passordløs ssh-bruk. Og dette gjelder også for scp og sftp.

Merk: Det er nå viktig å ha et godt passord for innlogging på din egen pc, slik at du ikke åpner et sikkerhetshull av dimensjoner her.

Sikker kopiering av filer og mapper med scp

Kommandoen **scp** er en del av SSH-serveren og brukes til kryptert kopiering til og fra servere.

Eksempler:

```
$ scp filnavn.txt brukernavn@server.com:/mappe
```

- Kopierer *filnavn.txt* til mappen *mappe* på serveren *server.com* innlogget som *brukernavn*

```
$ scp -P 20300 filnavn.txt brukernavn@server.com:/mappe
```

- Samme som ovenfor, men bruker port 20300 istedenfor port 22 (default).

```
$ scp brukernavn@server.com:/mappe/filnavn.txt .
```

- Kopierer *filnavn.txt* fra mappen *mappe* på serveren *server.com* innlogget som *brukernavn, til den mappen du er i lokal - angitt ved "."*

```
$ scp -P 20300 -r mappe1 brukernavn@server.com:mappe1
```

- Samme som ovenfor, men kopierer rekursivt mappen *mappe1* til mappen *mappe1* i brukerens hjemmemappe på serveren

SFTP - Sikker FTP

SFTP er en del av OpenSSH, og fungerer som en vanlig ftp-server eller klient med fordelen at all trafikk er kryptert.

Her er noen enkle eksempler på bruk av SFTP

```
$ sftp -P 2022 brukernavn@servernavn.com:/home/bruker/www/
```

- logger deg på *servernavn.com* som bruker *brukernavn* via port 2022, og går til mappen */home/bruker/www*

Merk: Dersom ssh-serveren kjøres på default-porten (port 22) trenger du ikke ha med port-parametret

```
$ pwd
```

- viser hvilken mappe du er i

```
$ lpwd
```

- viser hvilke mappe du er i lokalt (på din egen pc)

```
$ cd dokumenter
```

- går til mappen *dokumenter* på serveren

```
$ lcd dokumenter
```

- går til mappen *dokumenter* lokalt

```
$ put *.php
```

- laster opp alle php-filer fra den mappen du er i lokalt til den mappen du er i på serveren

```
$ get filnavn1 filnavn2
```

- laster ned dokumentene *filnavn1* og *filnavn2* fra serveren til mappen du er i lokalt

Merk: De fleste kommandoer du kan kjøre på serveren har en tilsvarende kommando for å kjøres lokalt - ved rett og slett å skrive "l" foran kommandoen, som i eksemplene ovenfor.

SSH - tunneler

Et vanlig bruksområde for SSH er å lage såkalte tunneler fra en port på en maskin til en port på en annen. Slik kan man f.eks. kryptere trafikk som ellers ville gått i klartekst.

Eksempel 1:

Du er på en server uten internet-tilgang (server1), men som er koblet i et lokalt nettverk til en annen maskin med internetttilgang (server2). Serveren har problemer, og du trenger å Google frem en løsning. Slik gjør du:

```
$ ssh -L 12345:google.com:80 bruker@server2
```

Nå kan du åpne en forbindelse til port 12345 på server1 og få tilgang til Google fra server2 - f.eks. ved å peke Firefox til http://localhost:12345

For å unngå å utføre kommandoer på server2, er det vanlig å ta med paramteret -N (no command) slik:

```
$ ssh -L 12345:google.com:80 -N bruker@server2
```

Kjører server2 ssh-serveren på en annen port en port 22, må du også ta med portnummeret, f.eks slik (hvis den kjører på port 2222):

```
$ ssh -p 2222 -L 12345:google.com:80 bruker@server2
```

Eksempel 2:

serverA er en server som kjører CUPS printerserver på standardport (631), men webgrensenittet til CUPS er bare tilgjengelig lokalt. Slik kan du f å tilgang til det fra en annen maskin gjennom en ssh-tunnel:

```
$ ssh -L 12345:localhost:631 -N bruker@serverA
```

Port 12345 vil nå peke til port 631 på serverA, og CUPS er tilgjengelig fra:

http://localhost:12345

Eksempel 3:

SSH kan gi deg full tilgang til internett fra en annen maskin gjennom den innebygde støtten for SOCKS 5. F.eks. slik:

```
$ ssh -D 1234 bruker@serverA
```

Når dette er gjort kan du sette opp f.eks. Firefox til å bruke localhost og port 1234 som Socks 5 Proxy, og så har du full tilgang til internett.

Kapittel 5
Data Wrangling

cat - skjøt sammen filer og mye mer

cat har navnet sitt fra det engelske ordet *con**cat**enate* som betyr å skjøte noe sammen, og det er ett av bruksområdene for **cat**.

cat tar disse parameterene:

```
-A, --show-all
      equivalent to -vET
-b, --number-nonblank
      number nonempty output lines, overrides -n
-e    equivalent to -vE
-E, --show-ends
      display $ at end of each line
-n, --number
      number all output lines
-s, --squeeze-blank
      suppress repeated empty output lines
-t    equivalent to -vT
-T, --show-tabs
      display TAB characters as ^I
-u    (ignored)
-v, --show-nonprinting
      use ^ and M- notation, except for LFD and TAB
--help display this help and exit
--version
      output version information and exit
```

Eksempler:

```
$ cat navn1
Ole Moen
Petter Jensen
$ cat navn2
Kari Diesen
Rolf Juster
$ cat navn1 navn2
Ole Moen
Petter Jensen
Kari Diesen
Rolf Juster
$ cat -n navn1 navn2
     1  Ole Moen
     2  Petter Jensen
     3  Kari Diesen
     4  Rolf Juster
```

Skriv til fil fra tastaturet:

```
$ cat frukt
eple
appelsin
$ cat >>frukt
papaya
ananas
$ cat frukt
eple
appelsin
papaya
ananas
```

Skriv til en fil på en server:

```
$ cat ~/.ssh/id_rsa.pub | ssh server4 'cat >> .ssh/authorized_keys'
```

Kommandoen over kopierer brukerens offentlige nøkkel over il server4 og legger den til i server-brukerens authorized_keys

head og tail

head

head er et program som viser de første linjene i en tekstfil eller en tekststrøm. Default er 10 linjer, men du kan spesifisere antallet med en bindestrek etterfulgt av et tall, slik:

```
$ head -20  tekstfil.txt   (viser de 20 første linjene i tekstfil.txt)
$ history | head -5   (viser de 5 første linjene i kommando-historikken)
$ head -n -5000 enhetsregisteret (viser alle bortsett fra de siste 5000
linjene i filen enhetsregisteret)
```

tail

tail viser de siste linjene i en tekstfil eller tekststrøm. Hvor mange kan du spesifisere med en bindestrek etterfulgt av et tall, slik:

```
$ tail -20  tekstfil.txt   (viser de 20 siste linjene i tekstfil.txt
```

tail -f

tail tar et parameter -f (follow) som lar deg se i sanntid hvordan en fil endres, dvs. hva som legges til (eller slettes) fra slutten av filen. **tail -f** brukes ofte til å følge med på loggfiler og andre filer som endres ofte. Eksempel:

```
# tail -f /var/log/maillog
```

Avslutt **tail -f** med **<ctrl>+c**

more og less

Et potensielt problem med **cat** er at begynnelsen av teksten forsvinner hvis teksten rommer mer enn ett skjembilde. **more** ble laget for å bøte på dette. Ved å bruke **more** istedenfor **cat** vil skjermen fylles av den første delen av teksten, og ved å taste mellomrom-tasten kommer vises neste skjermbilde, helt til hele teksten er vist. Du kan når som helst avslutte visningen ved å taste bokstaven "q" (for quit). Hvis teksten ikke fyller mer enn ett skjermbilde, fungerer **more** akkurat som **cat.**

more har en åpenbar begrensning i at man ikke kan bla seg bakover til forrige skjermbilde. Det ga inspirasjon til utviklingen av **less**, som fikk navnet sitt ut fra idéen om at ***more is less.*** Med **less** kan du "scrolle" frem og tilbake i teksten med piltastene, og avslutte med å taste bokstaven "q".

Moderne terminal-emulatorer, som ***gnome terminal*** eller kde sin ***konsole*** har innebygde scrollbars, som gjør more og less mer eller mindre overflødige, men de blir viktige når man er logget direkte inn på en server uten grafisk brukergrensesnitt, og det ikke er mulighet til scrolling.

MERK: Hvis vi sender resultatet av more eller less videre med en | , vil de fungere på samme måte som cat:

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

Kommandoer for å bevege seg rundt i tekst med **less;**

Key	Description
Arrow	Move by one line
Space	Move down one page
b	Move up one page
g	Go to the first line
G	Go to the last line
100g	Go to the 100th line
/string	Search for the string from current position
n/N	Go to the next or previous search match
q	Exit less

date

date er en kommando som uten parametere gir oss dagens dato.

Her er noen eksempler:

```
$ date
ti. 08. des. 20:58:01 +0100 2020
$ date -I      (ISO-format)
2020-12-08
$ date --date='+2 weeks'
ti. 22. des. 20:59:53 +0100 2020
$ echo "filnavn_`date -I`.txt"
filnavn_2020-12-08.txt
$ echo "filnavn_`date --date='3 days ago' -I`.txt"
filnavn_2020-12-05.txt
```

grep - fgrep - egrep - søk i tekstfiler

grep er et kraftig søkeverktøy for søk i tekstfiler. **grep** kan brukes direkte mot filer, eller på input fra en strøm, f.eks. via en | fra et annet program. **grep** kan søke etter fast tekst, men også etter regulære uttrykk (regex), men bare ***basic*** og ikke ***extended*** regex. (forskjellen mellom basic og extended).

grep har parameteret **-E** som lar deg bruke extended regex, og parameteret **-F** som lar deg søke i fast tekst uten regulære uttrykk (som er noe raskere).

egrep - er grep som også kan søke i **extended regex. (e** står for extended)

fgrep - er grep som ikke bruker regulære uttrykk i det hele tatt, og er demed litt raskere enn grep og egrep. (**f** står for fixed, som i "fixed string")

MERK: både fgrep og egrep er ***deprecated***, som betyr at de vil bli fjernet på et senere tidspunkt - og det anbefales å erstatte dem med **grep -E** eller **grep -F**

Eksempler:

1) Tell hvor mange linjer som starter med ordet Prince i Tolstoys Krig og Fred:

```
$ grep -c '^Prince ' war-and-peace.txt
274
```

MERK: Tegnet ^ er en del av basic regex, og angir at det som følger må stå begynnelsen av teksten, som her blir i begynnelsen av en linje. Flagget **-c** ber grep om å telle antall forekomster.

2) List alle linjer i /var/log/secure som inneholder ordene failed og mysql

```
$ sudo grep -i failed /var/log/secure | grep -i mysql
Aug  1 19:05:07 wp520 useradd[4429]: failed adding user 'mysql', exit
code: 9
Aug 19 15:41:13 wp520 useradd[5263]: failed adding user 'mysql', exit
code: 9
Nov  2 21:43:55 wp520 useradd[22115]: failed adding user 'mysql', exit
code: 9
Nov 11 01:55:40 wp520 useradd[44976]: failed adding user 'mysql', exit
code: 9
Dec 10 14:52:55 wp520 useradd[4596]: failed adding user 'mysql', exit
code: 9
Jan 19 15:04:17 wp520 useradd[14589]: failed adding user 'mysql', exit
code: 9
```

3) Tell alle feilede loginforsøk på mailserveren

```
$ sudo grep -ic 'authentication failed' /var/log/maillog
74374
```

4) søk etter navn som slutter på sen eller son

```
$ cat navn
Ole Olsen
Jan Janson
Per Petterson
Kari Svendsen
Olga Konkova
$ grep 'sen' navn
Ole Olsen
Kari Svendsen
$ grep 'son' navn
Jan Janson
Per Petterson
$ grep 's\(o\|e\)n' navn
Ole Olsen
Jan Janson
Per Petterson
Kari Svendse
$ grep -E 's(o|e)n' navn
Ole Olsen
Jan Janson
Per Petterson
Kari Svendsen
```

sort - sorter tekstfiler

sort er en nyttig kommando som sorterer linjer i en tekstfil eller i en strøm av tekst i en kommandokjede.

sort har disse parameterene;

-b, --ignore-leading-blanks
ignore leading blanks
-d, --dictionary-order
consider only blanks and alphanumeric characters
-f, --ignore-case
fold lower case to upper case characters
-g, --general-numeric-sort
compare according to general numerical value

-i, --ignore-nonprinting
consider only printable characters
-M, --month-sort
compare (unknown) < 'JAN' < ... < 'DEC'
-h, --human-numeric-sort
compare human readable numbers (e.g., 2K 1G)
-n, --numeric-sort
compare according to string numerical value
-R, --random-sort
shuffle, but group identical keys. See shuf(1)
--random-source=FILE
get random bytes from FILE
-r, --reverse
reverse the result of comparisons
--sort=WORD
sort according to WORD: general-numeric -g, human-numeric -h, month -M, numeric -n, random -R, version -V
-V, --version-sort
natural sort of (version) numbers within text

Eksempler:

```
$ cat >>tall
34
12
67
123
7
56
^C
$ sort tall
12
123
34
56
67
7
$ sort -nr tall
123
67
56
34
12
```

```
7
$ curl -s https://web.itfakultetet.no/navn
Ola
Petter
Kari
Anders
$ curl -s https://web.itfakultetet.no/navn | sort
Anders
Kari
Ola
Petter
```

uniq - Fjern duplikater

Kommandoen **uniq** lar oss luke bort duplikate linjer i en tekstfil eller -strøm.

MERK: uniq sjekker en linje i forhold til foregående linje, så vi må sortere filen eller strømmen for å luke bort alle duplikater.

uniq tar disse parameterene:

-c, --count
prefix lines by the number of occurrences
-d, --repeated
only print duplicate lines, one for each group
-D print all duplicate lines
--all-repeated[=METHOD]
like -D, but allow separating groups with an empty line;
METHOD={none(default),prepend,separate}
-f, --skip-fields=N
avoid comparing the first N fields
--group[=METHOD]
show all items, separating groups with an empty line;
METHOD={separate(default),prepend,append,both}
-i, --ignore-case
ignore differences in case when comparing
-s, --skip-chars=N
avoid comparing the first N characters
-u, --unique
only print unique lines
-z, --zero-terminated
line delimiter is NUL, not newline

-w, --check-chars=N
compare no more than N characters in lines
--help display this help and exit
--version
output version information and exit

Eksempler:

```
$ cat frukt
eple
appelsin
eple
appelsin
eple
pære
pære
mango
$ uniq frukt
eple
appelsin
eple
appelsin
eple
pære
mango
$ sort frukt | uniq
appelsin
eple
mango
pære
$ sort frukt | uniq -c
      2 appelsin
      3 eple
      1 mango
      2 pære
```

tr (translate) - endre tegn i en tekst

tr er en effektiv kommando for å endre (oversette) ett eller flere tegn i en tekst. **tr** kan brukes med eller uten regulære uttrykk.

Eksempler på bruk:

```
$ cat navn

Ole Olson
Jan Janson
Per Petterson
Kari Svendson

$ cat navn | tr '[:lower:]' '[:upper:]'

OLE OLSON
JAN JANSON
PER PETTERSON
KARI SVENDSON

$ cat navn | tr 'OJP' 'ojp'

ole olson
jan janson
per petterson
Kari Svendson

$cat domener.txt

www. tecmint. com
www. fossmint. com
www. linuxsay. com

$ cat domener.txt | tr -d ' '

www.tecmint.com
www.fossmint.com
www.linuxsay.com

$ tr -d ' ' < domener.txt > domener_fixed.txt

$ cat domener_fixed.txt
www.tecmint.com
www.fossmint.com
www.linuxsay.com
```

sed - søk og erstatt tekst

sed

sed (stream editor) er en videreutvikling av den opprinnelige editoren **ed,** og brukes til å modifisere strømmer av tekst fra pipes eller filer.

sed kan ikke brukes interaktivt, i stedet spesifiserer man instruksjoner som **sed** utfører på devn valgte teksten.mmand-line shells.

Dette kan **sed** brukes til:

- Søke i tekst
- Erstatte tekst
- Legge linjer til tekst
- Slette linjer fra tekst
- Endre (eller bevare) en original fil

sed kan brukes med **regulære uttrykk,** som gjør det effektivt og fleksibelt å søke opp tekst.

Eksempler

1) Erstatte tekst fra en pipe

$ echo "Dette er en tekst som er uendret" | **sed s/" er "/" var "/g**
Dette var en tekst som var uendret

2) Erstatte tekst fra en fil, uten å endre filen

```
$ cat navn
Ole Olsen
Jan Jansen
Per Pettersen
Kari Svendsen
$ sed 's/sen/son/g' navn
Ole Olson
Jan Janson
Per Petterson
Kari Svendson
```

Resultatet skrives til skjerm, men filen endres ikke

3) Erstatte tekst i en fil, slik at filen endres, med flagget -i

```
$ sed -i 's/sen/son/g' navn
$ cat navn
Ole Olson
Jan Janson
Per Petterson
Kari Svendson
```

MERK: g står for "globally", dvs alle forekomster.

4) Velg linjer fra en tekst

```
$ sed -n '3,4p' navn
Per Petterson
Kari Svendson
```

Velger linje 3 til og med linje 4. p står for print og n angir at resten av filen ikke skal printes samtidig.

5) Erstatt mellomrum med linjeskift, slik at hvert ord kommer på en egen linje

```
$ sed 's/\s/\n/g' navn
Ole
Olson
Jan
Janson
Per
Petterson
Kari
Svendson
```

MERK: \s står for "space" og \n står for "new line".

awk - et programmeringsspråk for behandling av tekst

awk er et eget programmeringsspråk utviklet ved AT&T Bell i 1977 og har navnet sitt fra utviklerne: Aho, Weinberger og Kernighan.

awk har bukerdefinerte funksjoner, kan håndtere multiple input-strømmer, TCP/IP networking og et rikt sett med regulære uttrykk. Det brukes ofte til å prosessere rene tekstfiler, hvor awk kan tolke data som rader og felt som brukeren kan behandle.

awk søker i filer etter tekst-enheter (som regel linjer avsluttet med en *end-of-line character*) som inneholder bruker-definerte mønstre.

Eksempler:

$ awk '/foo/ { print toupper($0); }'
Dette er en tekst
Dette er en tekst som inneholder ordet foo
DETTE ER EN TEKST SOM INNEHOLDER ORDET FOO

$ cat navn
Ole Olsen
Jan Janson
Per Petterson
Kari Svendsen
Olga Konkova

$ awk '/Kari/ { print $2 }' navn
Svendsen

GNU-versojnen av awk kalles gawk, men programmet er linket til awk, og kan startes med kommandoen: awk.

gawks online manual:

https://www.gnu.org/software/gawk/manual/

cut

cut lar deg klippe ut kolonner fra en kolonne-inndelt tekst og enten lagre dem til en ny fil eller sende dem videre med en | til en ny kommando.

cut tar disse parametrene:

```
       -b, --bytes=LIST
              select only these bytes
       -c, --characters=LIST
              select only these characters
       -d, --delimiter=DELIM
              use DELIM instead of TAB for field delimiter
       -f, --fields=LIST
              select only these fields;  also print any line that contains
no delimiter character, unless the -s option is specified
       -n     with -b: don't split multibyte characters
       --complement
              complement the set of selected bytes, characters or fields
```

```
       -s, --only-delimited
              do not print lines not containing delimiters
       --output-delimiter=STRING
              use STRING as the output delimiter the default is to use the
input delimiter
       -z, --zero-terminated
              line delimiter is NUL, not newline
       --help display this help and exit
       --version
              output version information and exit
```

Eksempler:

1) Hent ut navn, orgnummer og antall ansatte fra Brønnøysunds enhetsregister, og erstatt semikolon med komma som skilletegn:

Hent filen fra difi:

```
$ wget https://hotell.difi.no/download/brreg/enhetsregisteret
```

(vi kan bruke **head** for å se strukturen på filen)**:**

```
$ head enhetsregisteret
orgnr;navn;organisasjonsform;forretningsadr;forradrpostnr;forradrpoststed;
forradrkommnr;forradrkommnavn;forradrland;postadresse;ppostnr;ppoststed;pp
ostland;regifr;regimva;nkode1;nkode2;nkode3;sektorkode;konkurs;avvikling;t
vangsavvikling;regiaa;regifriv;regdato;stiftelsesdato;tlf;tlf_mobil;url;re
gnskap;hovedenhet;ansatte_antall;ansatte_dato
"974766507";"VOSS HERAD KOMMUNALAVDELING
OPPVEKST";"ORGL";"";"5700";"VOSS";"4621";"VOSS";"Norge";"";"5701";"VOSS";"
Norge";"N";"N";"84.120";"";"";"6500";"N";"N";"N";"J";"N";"01.09.1996";"01.
02.1969";"";"";"";"";"960510542";"1072";"15.04.2020"
"974774356";"VOSS HERAD KOMMUNALAVDELING
TEKNISK";"ORGL";"";"5700";"VOSS";"4621";"VOSS";"Norge";"";"5701";"VOSS";"N
orge";"N";"N";"84.130";"";"";"6500";"N";"N";"N";"J";"N";"01.09.1996";"01.0
4.1988";"";"";"";"";"960510542";"184";"12.03.2020"
"974781468";"VOSS HERAD RÅDMANNEN SIN
STAB";"ORGL";"";"5700";"VOSS";"4621";"VOSS";"Norge";"";"5701";"VOSS";"Norg
e";"N";"N";"84.110";"";"";"6500";"N";"N";"N";"J";"N";"01.09.1996";"01.01.1
995";"56 51 94 00";"";"";"";"960510542";"72";"15.04.2020"
"974770962";"VOSS
KINO";"ORGL";"";"5704";"VOSS";"4621";"VOSS";"Norge";"";"5701";"VOSS";"Norg
e";"N";"N";"59.140";"";"";"6500";"N";"N";"N";"J";"N";"01.09.1996";"01.12.1
973";"";"";"";"";"960510542";"4";"11.03.2020"
```

```
"922201722";"& DALE";"ENK";"H0101 Bjerkelundgata
6A";"0553";"OSLO";"0301";"OSLO";"Norge";"";"";"";"";"N";"N";"70.210";"";""
;"8200";"N";"N";"N";"N";"N";"11.02.2019";"";"";"";"";"";"";""
"917887721";"&ACTION";"FLI";"c/o Håvard Sørli Helgesmark
29";"7716";"STEINKJER";"5006";"STEINKJER";"Norge";"";"";"";"";"N";"N";"94.
991";"";"";"7000";"N";"N";"N";"N";"J";"17.10.2016";"12.10.2016";"";"";"";"
";"";"";""
"999015611";"&M HOLDING AS";"AS";"Bygg D12 Sandakerveien
24C";"0473";"OSLO";"0301";"OSLO";"Norge";"";"";"";"";"J";"N";"74.102";"";"
";"2100";"N";"N";"N";"N";"N";"16.10.2012";"04.10.2012";"";"900 54
237";"";"2019";"";"";""
"812467182";"&MORE AS";"AS";"Sofies gate
72";"0454";"OSLO";"0301";"OSLO";"Norge";"";"";"";"";"J";"J";"71.129";"";""
;"2100";"N";"N";"N";"J";"N";"19.09.2013";"15.09.2013";"";"926 33
798";"";"2018";"";"1";"11.07.2017"
"920352928";"&TRADITION NORWAY AS";"AS";"Parkveien
41B";"0258";"OSLO";"0301";"OSLO";"Norge";"";"";"";"";"J";"J";"46.150";"";"
";"2100";"N";"N";"N";"J";"N";"31.01.2018";"05.01.2018";"";"";"";"2018";"";
"2";"12.02.2020"

$ cut -d ';' -f 1,2,32 --output-delimiter=',' enhetsregisteret
orgnr,navn,ansatte_antall
"974766507","VOSS HERAD KOMMUNALAVDELING OPPVEKST","1072"
"974774356","VOSS HERAD KOMMUNALAVDELING TEKNISK","184"
"974781468","VOSS HERAD RÅDMANNEN SIN STAB","72"
"974770962","VOSS KINO","4"
"922201722","& DALE",""
"917887721","&ACTION",""
"999015611","&M HOLDING AS",""
"812467182","&MORE AS","1"
"920352928","&TRADITION NORWAY AS","2"
... etc
```

2) Hent brukernavn og bruker-ID fra /etc/passwd og sorter etter navn

```
$ cut -d: -f1,3 /etc/passwd | sort
abrt:173
adm:3
akmods:974
apache:48
avahi:70
bin:1
chrony:994
colord:981
daemon:2
```

```
dbus:81
dnsmasq:986
flatpak:978
ftp:14
```

paste

paste er motsvarigheten til **cut**, og lar deg "lime inn" kolonner fra en fil i en annen fil, evt. til skjerm eller en | til et annet program.

paste tar disse parametrene:

-d, --delimiters=LIST
reuse characters from LIST instead of TABs

-s, --serial
paste one file at a time instead of in parallel

-z, --zero-terminated
line delimiter is NUL, not newline

--help display this help and exit

--version
output version information and exit

Eksempler:

```
$ cat fornavn
Ole
Jan
Per
Kari

$ cat etternavn
Olsen
Jansen
Pettersen
Svendsen

$ paste fornavn etternavn
```

```
Ole    Olsen
Jan    Jansen
Per    Pettersen
Kari   Svendsen

$ paste -d ',' etternavn fornavn
Olsen,Ole
Jansen,Jan
Pettersen,Per
Svendsen,Kari
```

comm og diff

comm og **diff** er to kommandoer som alr deg sammenlikne to tekstfiler og se hva som er likt og hva som er forskjellig mellom de to filene.

comm

Her er et enkelt eksempel på bruk av **comm:**

```
$ cat frukt
appelsin
eple
mango
papaya
$ cat frukt2
ananas
banan
eple
mango
pære
$ comm frukt frukt2
    ananas
appelsin
    banan
        eple
        mango
papaya
    pære
```

Forklaring: comm produserer tre kolonner:

1. Det som finnes i fil a men ikke i fil b

2. Det som finnes i fil b men ikke i fil a
3. Det som er likt i de to filene

Vi kan velge å fjerne en eller flere av kolonnene, f.eks. kolonne 1 og 2, slik at vi bare står igjen med det som er felles:

```
$ comm frukt frukt2 -1 -2
eple
mango
```

diff

diff gir oss en oversikt over forskjellen mellom to filer.

Her er et eksempel basert på lignende, men mer like frukt-filer enn ovenfor:

```
$ cat frukt1
ananas
banan
papaya
mango
pære
$ cat frukt2
ananas
banan
eple
mango
pære
$ diff frukt1 frukt2
3c3
< papaya
---
> eple
```

Forklaring:

Utgangspunktet for diff er at den første filen skal redigeres med den andre filen som referanse-fil. Konkret:

3c3 betyr at linje 3 i fil a skal byttes ut (c = change) med linje 3 i fil b, så ut med papaya og inn med eple, så er filene like.

split - del opp en fil i flere mindre filer

Med kommandoen **split** kan vi dele opp en stor fil i mindre filer. Default-innstillingen er 1000 linjer per ny fil, men vi kan også dele filer opp etter størrelse.

Eksempler:

1) Hent teksten til boken Krig og Fred:

```
$  curl https://web.itfakultetet.no/war-and-peace.txt > kof.txt
$ ls -lh kof.txt
-rw-r--r-- 1 Lenovo 197609 3,2M des  7 00:46 kof.tx
```

2) Del filen i filer på 1Mb størrelse hver, med prefiks krig_og fred_

```
$ split -b 1M kof.txt krig_og_fred_
$ ls -lh krig*
-rw-r--r-- 1 Lenovo 197609 1,0M des  7 00:59 krig_og_fred_ab
-rw-r--r-- 1 Lenovo 197609 1,0M des  7 00:59 krig_og_fred_ac
-rw-r--r-- 1 Lenovo 197609 118K des  7 00:59 krig_og_fred_ad
-rw-r--r-- 1 Lenovo 197609 1,0M des  7 00:59 krig_og_fred_aa
```

3) Slå sammen delene til 1 fil igjen

```
$ cat krig_og_fred_aa krig_og_fred_ab krig_og_fred_ac krig_og_fred_ad >
krig_og_fred.txt
```

Øvelse: Skriv direkte til en fil med echo og/eller cat

echo <tekst eller variabel> skriver til standard output, hvis ikke annet angis
cat <filnavn> - leser inn en fil og skriver til skjerm eller ny fil

Du kan skrive rett til en fil med **echo** og/eller **cat** på flere måter. Her er et par eksempler

1) med echo "tekst" > (eller >>) filnavn (> overskriver filen, >> legger til en linje på slutten av filen)

```
$echo "Dette er noe som skal logges" >> loggfil.txt

$echo "Dette er også noe som skal logges" >> loggfil.txt
```

```
$cat loggfil.txt
Dette er noe som skal logges
Dette er også noe som skal logges
```

2) med cat << avslutningstegn > filnavn

Skriv tekst over flere linjer og avslutt med avslutningstegnet (f.eks. EOF) på en egen linje tilslutt.

```
$ cat <<EOF>test.txt
> DETTE ER TITTELEN
> og her kommer resten av teksten
> som slutter på neste linje
> EOF

$ cat test.txt
DETTE ER TITTELEN
og her kommer resten av teksten
som slutter på neste linje

$cat<<slutt>>loggfil.txt
> og her er en linje til
> og enda en linje
> slutt

$ cat loggfil.txt
Dette er noe som skal logges
Dette er også noe som skal logges
og her er en linje til
og enda en linje
```

Øvelse: Lese og endre en fil med while og read

I denne enkle øvelsen skal vi lese inn en fil med navn, hvor hver linje består av et fornbavn og et etternavn, og endre den slik at etternavn kommer før fornavn, med et komma mellom de to.

La oss først lage filen, som vi kan kalle *navn*:

```
$ cat >>navn
Ole Moen
Petter Jensen
```

```
Kari Diesen
Rolf Juster
(avslutt med <ctrl>  + c )
```

Slik kan vi skrive filen ut til skjermen med etternavn først:

```
$ while read fornavn etternavn
> do
> echo $etternavn, $fornavn
> done < navn
Moen, Ole
Jensen, Petter
Diesen, Kari
Juster, Rolf
```

Vil vi lagre resultatet i en ny fil, f.eks. *navn2*, kan vi gjøre det slik

```
while read fornavn etternavn; do echo $etternavn, $fornavn; done < navn >
navn2
```

(her står alt på 1 linje, og da bruker vi semikolon der vi ellers ville brukt linjeskift)

Vi ser at innholdet i navn2 er riktig:

```
$ cat navn2
Moen, Ole
Jensen, Petter
Diesen, Kari
Juster, Rolf
```

Vi kan også endre teksten underveis, for eksempel ved å bruke bash sin innebygde endringsfunksjon:

```
${<tekst>/<gammel verdi>/<ny verdi>}
```

Vil vi endre alle forekomster av <gammel verdi>, setter vi to // etter teksten vi vil endre, slik:

```
${<tekst>//<gammel verdi>/<ny verdi>}
```

Her endre vi alle forekomster av jensen til Jansson og lagrer resultatet i filen: navn3:

```
$  while read fornavn etternavn; do echo ${etternavn//Jensen/Jansson},
$fornavn; done < navn > navn3
```

Vi kan sjekke resultatet ved å se på navn3:

```
$ cat navn3
Moen, Ole
Jansson, Petter
Diesen, Kari
Juster, Rolf
```

Øvelse: Finn de 20 mest brukte kommandoene dine

I denne øvelsen skal vi se hvordan vi enkelt (relativt sett) kan hente ut en oversikt over de mest brukte kommandoene fra skallets **history.**

Vi kan dele operasjonen inn i trinn:

1. Hent alle lagrede kommandoer med kommandoen **history**
2. Set felt 1 til et mellomrom med kommandoen **awk '$1=" "'**
3. Klipp ut felt 3 med kommandoen **cut -d ' ' -f3**
4. Sorter alfabetisk med kommadoen **sort**
5. Tell unike forekomster med kommandoen **uniq -c**
6. Sorter etter antall i synkende rekkefølge med kommandoen **sort -n -r**
7. Velg de 20 øverste på listen med kommandoen **head 20**

Slik vil hele kjeden av kommandoer se ut:

```
$  history | awk '$1=" "'| cut -d ' ' -f3 | sort | uniq -c | sort -n -r |
head -20
     135 sudo
     77 ls
     59 grep
     49 cat
     48 man
     42 echo
     34 sed
     33 egrep
     23 getent
     23 find
```

```
     21 nmcli
     18 chmod
     17 resolvectl
     17 history
     16 hashcat
     15 cut
     14 ssh
     14 head
     13 iwlist
     11 systemctl
```

Dersom du vil ha med parameterene også, kan du sette et minus-tegn etter 4 tallet i cut-kommandoen, som betyr hent fra felt 4 og ut linjen

```
$ history | awk '$1=" "'| cut -d ' ' -f3- | sort | uniq -c | sort -n -r |
head -20
     51  ls
     46  systemctl restart httpd
     25  ls -lh
     22  vim pgadmin4.conf
     20  htop
     19  su terje
     16  dnf upgrade
     14  ls -lrt
     14  cd ..
     13  vim pgadmin4-v1-le-ssl.conf
      9  cd /home/backup/server4/
      8  vim /etc/httpd/conf.d/pgadmin4-le-ssl.conf
      7  ls -lrth
      7  journalctl -xe
      7  du -sh
      6  ls -l
      5  systemctl restart jenkins
      4  vim config_local.py
      4  vim /etc/sysconfig/jenkins
      4  vim /etc/httpd/conf.d/pgadmin4.conf
```

Kapittel 6
Sikkerhet