Innføring i operativsystemer:

Linux (eller Unix, som er det opprinnelige navnet på dette operativsystemet)

Det et operativt system gjør er nettopp gjør en maskin operativ, siker

Samling med programvare som ligger som et lag over hardware

Programvaren tilbyr grensesnitt mellom bruker og applikasjonene som nettleser osv…

Os har grensesnitt med hardware, effektiv – siker

Av eksperter for eksperter, programvareutviklere fleksibelt

C programmering brukes for operativsystemer, adresse minne og hardware

Ikke veldig teoretisk

2014 utgave av modern operating systems

Krav: java for eksempel, bruk av windows, tilgang til linux-system: linux-server, dual-boot windows/linux

10 av 12 godkjent arbeidskrav

Flervalg eksamen

**$ ssh aleksawo@itstud**

Kommandolinjen i Linux er et tegnbasert grensesnitt som gjør at du kan kjøre både OS-kommandoer og andre programmer på maskinen. Når du bruker kommandolinjen kommuniserer du med et program som kalles et **shell**:

* Shellet tilbyr et "Command Line Interface" (CLI). CLI lar deg kontrollere datamaskinen ved å bruke kommandoer som legges inn bare med tastaturet, i stedet for å navigere i et grafisk brukergrensesnitt (GUI) med en mus eller annen pekeredskap.
* Shellet lar deg kjøre andre programmer eller script, og er også svært nyttig for å administrere datamaskinen når du vet hvilke kommandoer du skal bruke.
* De fleste Linux-distribusjoner bruker programmet **bash** (Bourne-Again Shell) som standard shell.

Merk at shellet ikke må forveksles med et annet program som kalles en **terminalemulator**. Terminalemulatoren etterligner/emulerer gammeldagse og tegnbaserte dataterminaler, som tidligere ble brukt som "arbeidsstasjoner" for brukere som delte på én stor, sentral datamaskin (mainframe). Emulatoren tilbyr bare et GUI-terminalvindu som "rammer inn" shellets kommandolinjegrensesnitt.

Et bilde som inneholder tekst, Elektronisk anordning, elektronikk, Kontorutstyr

Automatisk generert beskrivelseBildet viser en typisk terminal fra slutten av 70-årene, med skjerm og tastatur. Dette var **ikke** en datamaskin, men en tegnbasert (ikke-grafisk) I/O-enhet der man kunne kommunisere med den sentrale datamaskinen som den var koblet opp mot:

Et bilde som inneholder tekst, skjermbilde, programvare, Multimedieprogramvare

Automatisk generert beskrivelseSkjermbildet nedenfor er tatt fra desktopen til et moderne Linux-system med GUI, med en aktiv terminalemulator (det svarte vinduet). Det tegnbaserte shellet kjører linje-for-linje interaktiv kommunikasjon med brukeren inne i emulatoren, på samme måte som i de gamle terminalene:

Når du starter bash inne i et terminalvindu, er det første du vil se en **ledetekst** (aka "command prompt"), som betyr at shellet venter på at du skal skrive inn en kommando. Ledeteksten kan tilpasses og settes slik du vil ha den, men ser typisk slik ut: **username@hostname:currentdirectory$**

Når jeg logger på serveren it.stud.hiof.no, får jeg denne ledeteksten: **aleksawo@itstud:~$**

Symbolet '~' (tilde) i ledeteksten betyr at stående katalog er brukerens hjemmeområde (mer om kataloger og slikt kommer nedenfor i dette dokumentet). Kommandoene du gir til maskinen skrives etter "prompt-tegnet" '$' og avsluttes med en return/linjeskift. Shellet tolker da kommandoen du har gitt og setter i gang et program som utfører kommandoen, hvis den er korrekt. Hvis ikke, gis det en feilmelding.

**$ who**

Du vil da få se en liste med informasjon om påloggede brukere, bl.a. brukernavn og tidspunkt de logget på systemet.

Merk at alle Linux-kommandoer har såkalte **opsjoner** og **argumenter** som kan endre måten de fungerer på. Disse angis på kommandolinjen etter selve kommandonavnet, oftest som en eller to bindestreker etterfulgt av et tegn eller et ord.

Kommandoen *who* har mange opsjoner, blant annet:

* *-q* : viser alle påloggende brukernavn og antall brukere pålogget
* *-b* : viser tidspunktet for siste oppstart av systemet
* *--version:* skriver ut versjonsinformasjon for programmet who

**$ who -q**

**$ who -b**

**$ who --version**

For se alle mulige opsjoner som kommandoen *who* kan ha (og også mer informasjon om hvordan den virker og kan brukes), kan du gi følgende kommando som viser deg **manualsiden** for *who*:

**$ man who**

Kommandoen *man* brukes for å få hjelp og informasjon om Linux-kommandoer. Du kan "bla gjennom" hjelpesider fra *man* ved å trykke *SPACE*. *man* (og alle andre Linux-kommandoer) kan avbrytes ved å trykke *Ctrl-C* (som sender et såkalt interrupt som stopper det kjørende programmet). Mer om manualsider finner du nedenfor.

**$ cal**

Som standard skriver denne kommandoen ut kalenderen for nåværende måned. Hvis du ønsker å skrive ut en kalender for et bestemt år og måned, bruk denne syntaksen: *cal month year*.

Bruk ***uptime*** til å finne ut hvor lenge det er siden maskinen du er logget på var ute av drift.

Bruk ***finger*** til å finne ut mere om brukeren

Send en melding med ***write*** til en annen bruker på systemet.

Lær deg å bruke den enkle, selvforklarende teksteditoren ***nano***.

Finn ut hva kommandoen ***apropos*** kan brukes til:

I Linux er apropos et kommandoverktøy som brukes til å søke etter manualsidetitler og beskrivelser som inneholder et bestemt nøkkelord. Det er nyttig når du vet hva en kommando gjør eller hva du leter etter, men ikke husker den nøyaktige kommandoen.

**Hvordan bruke apropos:**

* Hvis du for eksempel leter etter kommandoer relatert til "network", kan du skrive:

Bash

Kommandoen ***which*** forteller hvor på disken de eksekverbare programmene for hver Linux-kommando er lagret. Finn ut hvor Java Virtual Machine ("**java**") ligger.

Programmet ***cat*** kan brukes til å se på innholdet av tekstfiler. Kjør kommandoen ***cat /etc/passwd*** for å se den "rå" informasjonen om alle brukerne på systemet.

Les brukerinformasjonen fra systemet en "side" om gangen ved å gi kommandoen ***cat /etc/passwd | more***. bruk også ***man*** for se hva som utføres av kommandoen *more*

Tegnet |, kjent som "pipe" i en kommandolinje i Linux (og andre Unix-lignende systemer), brukes til å sende utdata (output) fra én kommando som input til en annen kommando. Dette gjør det mulig å kjede flere kommandoer sammen for å utføre komplekse oppgaver i en enkelt kommandolinje.

**Hvordan | fungerer:**

* **Eksempel 1:** La oss si at du vil se innholdet i en katalog og deretter telle hvor mange filer som er i den katalogen. Dette kan du gjøre ved å kombinere ls-kommandoen (som lister opp filer) med wc -l (som teller antall linjer).

bash

I Linux brukes kommandoen *echo* til å skrive ut en linje med tekst på skjermen

**echo Cashmere Cat**

Linux-shell har en rekke innebygde **systemvariabler** som lagrer informasjon om bl.a. oppsett av systemet, opsjoner som styrer hvordan shellet virker, informasjon om bruker etc. Variablene lagrer tekststrenger og identifiseres med variabelnavn. Du kan se på innholdet i systemvariable i *bash* ved å sette '$' (et dollartegn) foran variabelnavnet.

**$ echo Hei $USER**

**$ echo $HOSTNAME**

**$ echo $TERM**

**$ echo $PATH**

Linux tilbyr også kommandoer for å sjekke maskinvare- og systeminformasjon. For eksempel vil kommandoen *lscpu* gi informasjon om prosessorene, men *lspci* lister opp alle PCI-bussene (PCI: Peripheral Component Interconnect) og alle enhetene som er koblet til en buss.

**$ lscpu**

**$ lspci**

Her er noen flere kommandoer som gir systeminformasjon. Prøv å forstå hva disse kommandoene gjør, ved å lese manualsider og studere utskriften:

**$ cat /proc/version**

Viser informasjon om kjernen (kernel) som kjører på systemet. Denne kommandoen leser innholdet av filen /proc/version, som inneholder versjonsinformasjon om Linux-kjernen, kompilatorbruk, og når kjernen ble kompilert.

**$ cat /proc/partitions**

Viser en liste over alle partisjonene som er registrert på systemet. Denne kommandoen leser innholdet av filen /proc/partitions, som gir informasjon om diskpartisjoner på systemet, inkludert størrelsen på hver partisjon og deres enhetsnavn.

**$ cat /proc/cpuinfo**

Viser detaljer om systemets CPU(er). Denne kommandoen leser innholdet av filen /proc/cpuinfo, som inneholder informasjon om CPU-arkitekturen, som prosessormodell, antall kjerner, klokkefrekvens, cache-størrelse, og andre CPU-relaterte detaljer. Utdataene kan være omfattende og typisk inneholde detaljer for hver prosessor (kjerne) i systemet.

**$ df**

Viser diskbruk for alle monterte filsystemer. df står for "disk free", og kommandoen viser hvor mye plass som er brukt og tilgjengelig på hvert montert filsystem, inkludert prosentandelen av brukte ressurser.

**$ lsblk**

Viser en liste over alle blokkenheter (som disker og deres partisjoner) på systemet. lsblk står for "list block devices". Den gir en hierarkisk visning av disker og tilhørende partisjoner, sammen med deres størrelse og monteringspunkter. Dette kan hjelpe deg å forstå diskstrukturen på systemet.

**$ free -m**

Viser informasjon om systemets minnebruk, inkludert RAM og bytteområde (swap). Kommandoen free viser hvor mye av det fysiske minnet (RAM) og swap-plassen som er brukt, tilgjengelig, og cachet. Flagget -m viser informasjonen i megabyte.

Etter at du har logget på et Linux-system, vil din arbeidskatalog (eller "stående katalog"/"current directory") oftest være hjemmeområdet ditt på disken, der du kan lagre dine egne filer.

Lag nå en katalog med *mkdir*

**$ mkdir katlognavn**

Bytt deretter stående katalog til å være denne nye katalogen ved å bruke kommandoen *cd* (change directory):

**$ cd katlognavn**

Deretter oppretter du en ny fil og redigerer den med den enkle, selvdokumenterende editoren *nano*:

**$ nano nyfil.txt**

Du vil da gå inn i *nano*-redigeringsgrensesnittet. Skriv inn litt tekst og trykk ***Ctrl-X*** for å lagre filen og avslutte *nano*. Tilbake på kommandolinjen, bruk *ls* til å liste ut navnene på alle filer i stående katalog:

**$ ls**

Filen ***nyfil.txt*** vil da skrives ut som eneste fil i katalogen ***katalognavn***.

Opprett noen flere tekstfiler i katalogen ***katalognavn***, f.eks. ved å bruke editoren ***nano*** eller kommandoen ***touch***, eller ved å laste ned noen filer fra internett med bruk av kommandoen ***wget***. Gjøre deretter følgende for å teste litt håndtering av filer og kataloger:

***ls***: Skriver en liste med informasjon om filene i en katalog

***cp***: Kopierer filer og kataloger

***mv***: Flytter eller bytter navn på filer og kataloger

***rm***: Sletter filer og kataloger (vær forsiktig med denne...)

***chmod***: Setter tilgangskontroll for filer (mer om dette senere i kurset)

***man mv*** for å se hvordan ***mv*** virker

Se på innholdet av tekstfilene dine med ***cat*, *more* og *less***.

// ikke ferdig med 1.6

Opprett en ny katalog som heter ***katalog2*** i katalogen ***katalognavn***. Lag flere nivåer av forskjellige underkataloger inne i***katalog2***. Bruke kommandoen ***cd*** til å flytte deg rundt i denne katalogstrukturen du har laget. Finn deretter ut hvordan du kan fjerne hele "katalogtreet" som starter i ***katalog2*** (og alle filene som evt. ligger der), med bare én enkel Linux-kommando.

**$ ls -a**

Kommandoen ls -a viser alt innholdet i en katalog. For f.eks. å vise informasjon om bl.a. alle filenes størrelser (i et lettlest "human-readable" format), bruk følgende kommando:

**$ ls -a -l -h**

eller "klump" de tre opsjonene sammen på denne måten:

**$ ls -alh**

I oversikten over filene som ***ls -a*** skriver ut, vil du først se en linje for en fil som heter bare "." (punktum) og deretter en som heter ".." (to punktum). Dette er to spesialfiler som finnes i alle Linux-kataloger. "." refererer til stående katalog (working directory), mens ".." refererer til katalogen på nivået over; forelderkatalogen (parent directory).

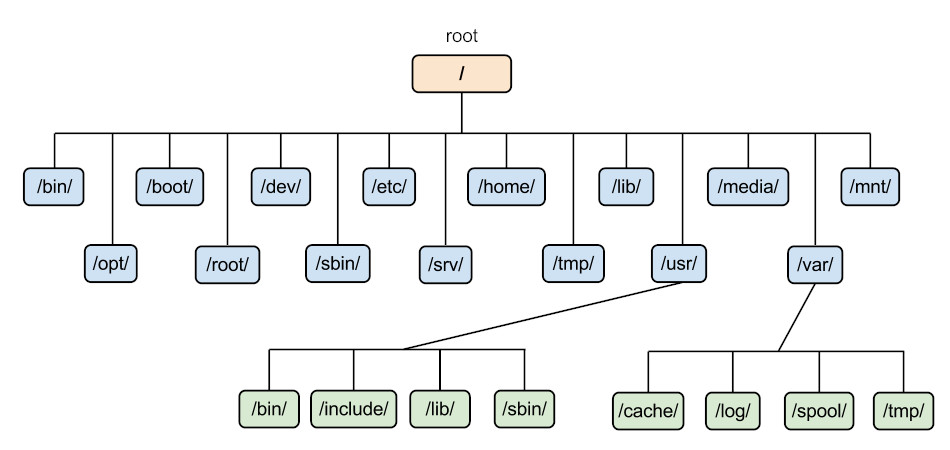
**$ cd .**

forandres ikke stående katalog. Kommandoen:

**$ cd ..**

vil sette stående katalog til å være forelderkatalogen. Prøv disse to kommandoene på katalogene som du har laget i ditt eget hjemmeområde.

Alle katalogene i et Linux-system er ordnet i et hierarkisk **tre** med mange nivåer. Hver katalog ligger inne i en annen (forelder)katalog som er på nivået ovenfor. Helt øverst i treet ligger en katalog som ikke har noen forelderkatalog, dette er **rotkatalogen** (root) som betegnes bare med en '/'. Figuren nedenfor viser et (relativt) standard oppsett av systemkataloger på de øverste nivåene i katalogtreet i Linux:



**Hjemmekatalogen** (home directory) til en bruker er den katalogen som settes til å være stående katalog etter pålogging. Som oftest er dette en katalog som heter det samme som brukernavnet og ligger inne i systemkatalogen */home*. I shellet det et eget, reservert tegn for å angi hjemmekatalogen i kommandoer som har kataloger som parametre: '~' (tilde).

**$ cd ~**

vil alltid sette stående katalog til å være hjemmeområdet ditt. Kommandoen ***cd*** uten et katalognavn som parameter gjør det samme som ***cd ~***.

En søkesti i Linux er en angivelse av hvor i filsystemet/katalogtreet du finner en bestemt katalog. Søkestien er en liste av katalognavn adskilt med spesialtegnet '/' (skråstrek/slash). Kommandoen *pwd* (print working directory) vil skrive ut søkestien til stående katalog.

Prøv å gi kommandoen *pwd* etter pålogging. Du vil da få skrevet ut søkestien til hjemmekatalogen din.

$ pwd

/home/aleksawo

Det finnes to ulike måter å angi søkestier på:

Absolutt søkesti:

Starter enten på toppen av filsystemet i rotkatalogen som angis med tegnet '**/**', eller i hjemmekatalogen som angis med tegnet '**~**'. Deretter kommer listen av katalognavn adskilt med '**/**' som går helt frem til katalogen der filen vi skal bruke befinner seg.

Relativ søkesti:

Stien starter med et katalognavn som ligger inne i stående katalog (working directory). Deretter kommer listen av katalognavn adskilt med '**/**'.

Prøv å bruke en absolutt søkesti for å bytte stående katalog -- du skal allerede ha laget katalogen ***katalognavn*** i din hjemmekatalog -- ved å gi kommandoen

**$ cd /home/aleksawo/Intro2OS**

Du skal nå være i ***Intro2OS***-katalogen din. For å bekrefte dette, skriv inn:

**$ pwd**

For å gå tilbake til hjemmekatalogen, skriv inn:

**$ cd**

Bytt nå til underkatalogen igjen ved å bruke en relativ søkesti:

**$ cd Intro2OS**

Bruk ***pwd*** for å bekrefte at du nå er i underkatalogen.

I eksemplet ovenfor er det ingen forskjell mellom disse to måtene å komme til en katalog på. Anta imidlertid at du var i en annen brukers hjemmekatalog. I det tilfellet ville kommandoen ***cd Intro2OS*** ta deg til hans eller hennes ***Intro2OS***-katalog. Men hvis du brukte den absolutte søkestien med brukernavnet ditt, ville du gå til din egen katalog.

Du kan bruke relative og absolutte søkestier for å beskrive filer og kataloger i mange kommandoer. For eksempel, hvis du er i hjemmekatalogen din, bør følgende to kommandoer gi samme resultat (en liste av filene i rotkatalogen på Linux-systemet):

**$ ls /**

**$ ls ../..**

Den første kommandoen bruker en absolutt søkesti (den starter med **/**), så den vil alltid returnere samme resultat uansett hvor du er. Hvis du er i katalogen ***/home/brukernavn*, vil ../..** referere til katalogen to nivåer over hjemmekatalogen (forelder til forelderkatalogen). Hvis du var i ***/home/brukernavn/Intro2OS*** og skrev ***ls ../..***, ville du se filene i katalogen ***/home*** i stedet.

Linux har to standard editorer, ***vi* og *emacs***. Det finnes også enklere editorer som f.eks. ***nano* og *joe***. Vi anbefaler **sterkt** at dere lærer å bruke ***emacs*,** som er et svært bra verktøy for det aller meste og spesielt for å skrive programkode i Linux.

stopp den igjen med ***:q***

Start (den fantastiske!) editoren ***emacs*** i terminal-modus med kommandoen ***emacs -nw*.**

Gå gjennom hele **nybegynneropplæringen** i ***emacs***, som du får frem ved å trykke ***Ctrl-h t*.**

Prøv psykoanalyse i **emacs** med kommandoen ***Esc-x doctor***

Det interaktive shellet ***bash*** har en snarvei-funksjon som lar deg skrive inn bare de første tegnene i en kommando. Trykk deretter på tabulatortasten og shellet vil skrive ferdig resten av kommandoen for deg (tab completion). Ved å trykke to ganger på tabulator vises alle kommandoer som matcher det du hittil har skrevet.

Du kan bruke opp- og ned-piltastene til å bla gjennom en liste med nylig innskrevne kommandoer. ***Ctrl-R*** vil søke gjennom kommandohistorikken din

Manualsidene er dine beste venner i Linux. De gir informasjon om bl.a. alle kommandoer, applikasjoner, systemkall, programmeringsspråk osv. Det krever litt trening/tilvenning for å få godt utbytte av manualsidene -- de er presise og korrekte, men også detaljerte, tekniske og omfattende.

Et eksempel: For å vise manualsiden til kommandoen ***ping***, som brukes for å sjekke hastighet og kvalitet på forbindelsen til andre maskiner på nettet, skriv inn *man ping* i shellet. Du vil da se noe slikt:

PING(8) iputils PING(8)

**NAME**

ping - send ICMP ECHO\_REQUEST to network hosts

**SYNOPSIS**

**ping** [**-aAbBdDfhLnOqrRUvV46**] [**-c** count] [**-F** flowlabel] [**-i** interval]

[**-I** interface] [**-l** preload] [**-m** mark] [**-M** pmtudisc\_option]

[**-N** nodeinfo\_option] [**-w** deadline] [**-W** timeout] [**-p** pattern]

[**-Q** tos] [**-s** packetsize] [**-S** sndbuf] [**-t** ttl]

[**-T** timestamp option] [hop...] {destination}

**DESCRIPTION**

**ping** uses the ICMP protocol mandatory ECHO\_REQUEST datagram to elicit

an ICMP ECHO\_RESPONSE from a host or gateway. ECHO\_REQUEST datagrams

("pings") have an IP and ICMP header, followed by a struct timeval and

then an arbitrary number of “pad” bytes used to fill out the packet [...]

**Manualsidene er delt inn i flere seksjoner**. F.eks. betyr **PING(8)** i eksemplet ovenfor at ***ping* sine manualsider finnes i seksjon 8**. For en beskrivelse av hva som ligger i de forskjellige manualseksjonene, gi kommandoen:

**$ man man**

**Noen viktige seksjoner: Shellkommandoer og en del systemprogrammer er i seksjon 1, systemkall i programmeringsspråket C er i seksjon 2, biblioteksfunksjoner er i seksjon 3, administrative systemverktøy er i seksjon 8.**

Du kan angi hvilken seksjon du vil se manualsider fra. For eksempel vil ***man write* vise manualsidene i seksjon 1 for shellkommandoen *write* som sender meldinger til andre brukere. For å se dokumentasjon for C-systemkallet *write()*, må du gi kommandoen *man 2 write*.**

I tillegg til manualsider tilbyr Linux også såkalte "**infosider**" for en del kommandoer og programvare. Infosider leses med kommandoen ***info***. De er ofte bedre oppdatert enn manualsider og kan være mer lettleste. Et eksempel er infosidene for kommandoen ***sort***, som inneholder mer informasjon om hva dette programmet gjør enn bare en beskrivelse av syntaks og funksjonalitet. Prøv kommandoen:

**$ info sort**

Andre nyttige hjelpeverktøy i Linux er ***apropos* og *whatis***. Prøv:

**$ man apropos**

**$ man whatis**

De fleste Linux-kommandoene som du bruker er egne, ferdig kompilerte C-programmer som ligger lagret et sted på disken. Når du gir en kommando til *bash*, vil shellet lete etter et program med samme navn som kommandoen. Katalogene som det letes i ligger lagret som en liste i shellvaraiabelen ***PATH***. For å se denne listen, gi kommandoen:

**$ echo $PATH**

Hvis ***bash*** finner kommandoprogrammet, lastes dette inn i RAM og startes. Hvis ikke, får du en feilmelding. For å finne ut hvor på disken en Linux-kommando ligger lagret, kan du bruke kommandoen ***which*.**

Shellet ***bash*** har i tillegg en del kommandoer som er **"innebygget",** dvs. at de **ikke** ligger på disk men er en del av shellprogrammet selv. Slike kommandoer kalles **"bash-builtins".** Du kan få en oversikt over dem ved å gi denne kommandoen:

**$ man builtins**

Builtins er **ikke** dokumentert i manualsystemet. I stedet tilbyr ***bash*** kommandoen ***help*** som gir kortfattet dokumentasjon om **builtins**. Et eksempel er builtin-kommandoen ***alias***, som gjør at du kan definere egne navn/shortcuts for f.eks. kommandoer du bruker ofte. Hvis du gir kommandoen *man alias* vil du kanskje (avhengig av versjonen av Linux so brukes) se at det ikke finnes noen manualside for ***alias***. Prøv i stedet:

**$ help alias**

Sjekk også om det finnes **manualside for kommandoen *cd*. Eller må du bruke *help cd*?**

Merk at en del kommandoer finnes **både** som en **builtin i *bash*** og som en vanlig Linux-kommando på disk, f.eks. kommandoen ***echo***. Det finnes også C-systemkall som har samme navn som **builtins, f.eks. *read***. Det betyr at det kan finnes manualsider for en kommando eller et systemkall med samme navn som en **builtin.**

Mange kommandoer i Linux er tekstbaserte "filterprogrammer", som:

* Leser en strøm av tekst/tegn, linje for linje.
* Gjør noe med ("filtrerer") tekstlinjene.
* Skriver deretter ut filtrert tekst linje for linje.

Et av filterprogrammene i Linux er kommandoen ***tr*: "translate or delete characters". *tr*** leser tekstlinjer og bytter ut eller fjerner tegn før linjene skrives ut igjen. Her er noen eksempler på bruk av ***tr*** til å gjøre om tekst, test gjerne disse med input fra tastaturet (se også *man* ***tr***):

* Små til store (engelske) bokstaver:

**$ tr a-z A-Z**

* Slett alle siffer i input:

**$ tr -d 0-9**

* Erstatt alle siffer med tegnet X:

**$ tr 0-9 X**

* Erstatt alle ikke-siffer(!) med X:

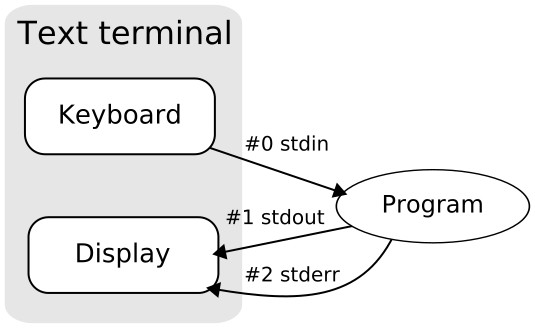
**$ tr -c 0-9 X**

* Erstatt alle påfølgende siffer med én X:

**$ tr -s 0-9 X**

Alle filterprogrammer bruker i utgangspunktet tre input/output enheter:

* Input leses fra standard innenhet: ***stdin*. *stdin*** er i utgangspunktet (**default)** satt til å være tastaturet
* Output skrives til standard utenhet: ***stdout*. *stdout*** er i utgangspunktet **terminalvinduet (skjermen).**
* Feilmeldinger skrives til standard feilenhet: ***stderr*. *stderr*** er i utgangspunktet også **terminalvinduet.**

Figuren nedenfor viser sammenhengen mellom filterprogram, terminalvindu/tastatur og de tre standard I/O-enhetene:

Shellet i Linux tilbyr **redirigering** (redirection) av input og output for filterprogrammer. Vi kan sette ***stdin* og *stdout*** til å være vanlige filer i stedet for tastatur og skjerm. Dette betyr at det er lettvint (og elegant) å få Linux-program og -kommandoer til å lese fra og skrive til filer.

Spesialtegnene **'<' og '>'** brukes på kommandolinjen, sammen med et filnavn, for å gjøre **redirigering av I/O:**

* Les input fra fil i stedet for fra tastatur:

**< filnavn**

* Skriv output til fil i stedet for til skjerm, overskriv hvis filen finnes fra før:

**> filnavn**

* Skriv output til fil, men legg til på slutten av filen **("append")** hvis filen finnes fra før:

**>> filnavn**

For å forstå hvordan **redirigering** virker, kan vi se på filterprogrammet ***cat***. Dette er en kommando som bare kopierer **input (fra *stdin*) til output (*stdout*). *cat*** kan også ta filnavn som **parametre**, den vil da skrive innholdet av filen ut til skjermen.

Kjør kommandoen ***cat*** og skriv inn noen linjer med tekst. Du vil da se at linjene bare **kopieres tilbake (som et ekko). *cat* kan avsluttes ved å trykke *Ctrl-D* (som er end-of-file tegnet i Linux) eller *Ctrl-C* (interrupt).**

Hvis du kombinerer ***cat* med redirigering av I/O**, kan du få denne enkle kommandoen til å gjøre andre nyttige ting:

* Viser innholdet av filen *fil\_1* på skjermen:

**$ cat < fil\_1**

* Lager en kopi av *fil\_1* og lagrer denne i *fil\_2*:

**$ cat < fil\_1 > fil\_2**

* Legger til innholdet av *fil\_1* på slutten av *fil\_2*:

**$ cat < fil\_1 >> fil\_2**

* Det som tastes inn blir lagret i *fil\_1*:

**$ cat > fil\_1**

Lag noen filer og prøv eksemplene ovenfor selv, for å se at det virker som det skal. Merk at i det siste eksemplet har ***cat*** faktisk blitt til en enkel teksteditor!

Noen flere eksempler på redirigering av I/O (les manualsider for å forstå hvordan kommandoene her virker):

* Lagrer en liste med navn på alle filene i stående katalog på en egen fil:

**$ ls > liste-fil.txt**

* Legger til nåværende tidspunkt og en liste med alle påloggede brukere på den samme filen:

**$ date >> liste-fil.txt  
$ who >> liste-fil.txt**

* Legger til de 20 siste linjene på en fil på slutten av en annen fil:

**$ tail -20 < fil\_1.txt >> fil\_2.txt**

Vi kan også redirigere I/O slik at output fra en kommando sendes som input til neste kommando, dette kalles for **piping** ("rørlegging" på norsk -- vi "kobler" to kommandoer sammen). Til dette brukes spesialtegnet '**|**' (vertikal strek eller "pipe") som skrives mellom de to kommandoene. Noen eksempler du kan teste (du vil antagelig trenge å lese manualsider for å forstå hvordan de virker):

* Telle antall filer (og kataloger) i stående katalog:

**$ ls | wc -l**

* Sortere listen over innloggede brukere på brukernavn:

**$ who | sort**

* Liste med alle brukere som heter "Magnus":

**$ cat /etc/passwd | grep -w Magnus**

* Hvor mange brukere heter "Magnus"?:

**$ cat /etc/passwd | grep -w Magnus | wc -l**

* Lage tilfeldig passord med 32 tall/bokstaver (denne er litt tricky):

**$ tr -cd '[:alnum:]' < /dev/urandom | fold -w32 | head -n1**