
NOTATER UKE 9

Docker containere er en teknologi for å kjøre programvare i isolerte og selvstendige miljøer. En container kan sees på som en virtuell maskin, men i stedet for å virtualisere en hel maskin, virtualiserer Docker kun selve applikasjonen og dens avhengigheter.

En Docker container inneholder alt som trengs for å kjøre en applikasjon, inkludert koden, kjøretidsmiljøet, biblioteker og avhengigheter. En container er isolert fra vertsmaskinen og andre containere, noe som gjør det enkelt å distribuere og kjøre applikasjoner uavhengig av systemmiljøet. Dette gjør det enklere å utvikle, teste og deployere applikasjoner.

Docker-containere kan kjøres på ulike operativsystemer og plattformer, og er blitt svært populære innen skybasert databehandling og distribuerte systemarkitektur.

Logget inn i Docker (s25) og kan se Ubuntu som kjøres:

```
group25@os25:~$ cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=20.04
DISTRIB_CODENAME=focal
DISTRIB_DESCRIPTION="Ubuntu 20.04.5 LTS"
group25@os25:~$
```

Kommandoen ifconfig viser info om nettet ved siden av inet:

Hit kom jeg etter at jeg skrev kommandoen «sudo su» og oppga passordet dorg1soap.

```
[root@os25:~# ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
      inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
        ether 02:42:6b:f0:21:27 txqueuelen 0 (Ethernet)
          RX packets 0 bytes 0 (0.0 B)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 0 bytes 0 (0.0 B)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 10.196.22.25 netmask 255.255.255.0 broadcast 10.196.22.255
        ether 02:42:0a:c4:16:19 txqueuelen 0 (Ethernet)
          RX packets 291487 bytes 24645417 (24.6 MB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 3511 bytes 245164 (245.1 KB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 netmask 255.0.0.0
        loop txqueuelen 1000 (Local Loopback)
          RX packets 12 bytes 1211 (1.2 KB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 12 bytes 1211 (1.2 KB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Fra uke 7:

- En linje C-kode kan gi mange linjer assembly/maskinkode
- Kompilering av høynivåkode til maskinkode er komplisert og ikke entydig
- Kompilering av assembly til en maskinkode er direkte og entydig
- En enkelt assembly instruksjon er delt inn i mikro-operasjoner og utføres i parallele pipelines

BRANCH PREDICITON

Branch prediction er en teknikk som brukes i datamaskiner for å øke hastigheten til programvare som inneholder såkalte «if-statements» eller andre grener i programflyten. Når en datamaskin kommer til en gren i programflyten, må den bestemme hvilken vei den skal fortsette å utføre koden. For eksempel, hvis en «if-statement» returnerer «true», vil datamaskinen fortsette å utføre koden i if-blokken, ellers vil den fortsette til koden etter if-blokkene.

I stedet for å vente til datamaskinen når grenen i programflyten, bruker branch prediction en teknikk der datamaskinen gjør en antakelse om hvilken vei programflyten vil gå basert på tidligere utføringer. Hvis datamaskinen gjør riktig antakelse, kan den fortsette å utføre koden uten å vente på at grenen skal nås, og dermed øke hastigheten på utførelsen av programmet.

Branch prediction brukes i mange moderne datamaskiner, og det er en viktig teknikk for å øke hastigheten på utførelsen av programvare.

HVORDAN HENGER BRANCH PREDICTION MED PIPELINING?

Pipelining er en teknikk som brukes for å øke hastigheten til en datamaskin. Den deler instruksjonene som skal utføres av datamaskinen inn i flere trinn, og lar forskjellige instruksjoner utføres samtidig i forskjellige trinn. På denne måten kan datamaskinen utføre flere instruksjoner på samme tid, og dermed øke hastigheten.

Midlertidig kan det være en utfordring å bruke pipelining sammen med grenhopp i instruksjonene (branching). Dette skyldes at datamaskinen ikke kan vite på forhånd hvilken vei en greninstruksjon vil føre, og dermed kan det hende at pipeliningen må stoppes midlertidig mens datamaskinen venter på resultatet av greninstruksjon.

Her kommer branch prediction inn i bildet. Dette er en teknikk som brukes for å gjette hvilken greninstruksjon som vil bli utført, og dermed kan pipeliningen fortsette uten avbrudd. Greninstruksjonen vil deretter bli utført hvis gjetningen var riktig, eller så vil datamaskinen gå tilbake og prøve igjen med den riktige grenen hvis gjetningen var feil.

Så branch prediction brukes i pipelining for å unngå midlertidige stopp i pipeliningen som skyldes greninstruksjoner, og dermed øke hastigheten på datamaskinen.

- Ved en branch i programmet (if-test), vet man ikke hva neste instruksjon er
- Stort problem for pipelining, må vente på resultatet fra forrige instruksjon
- Gjetter, basert på erfaring, hvilken branch (gren) som følges i programmet og utfører den
- Speculative execution, må gjøres om hvis feil
- I et superskalær arkitektur kan begge grener delvis utføres på forhånd

MELTDOWN

Meltdown er en sikkerhetsrisiko som påvirker moderne mikroprosessorer. Det oppstår når en angriper utnytter en sårbarhet i mikroprosessorens arkitektur for å få tilgang til sensitiv informasjon som normalt ikke skulle være tilgjengelig. Meltdown utnytter en egenskap ved moderne mikroprosessorer kalt «out-of-order execution», som gjør at prosessoren kan utføre flere instruksjoner samtidig i en annen rekkefølge enn de står i programmet.

Angriperen kan utnytte denne egenskapen til å få prosessoren til å laste inn sensitiv informasjon, som for eksempel passord eller private nøkler, i cachen til prosessoren. Deretter kan angriperen lese denne informasjonen fra cachen ved hjelp av et annet program som kjører på samme datamaskin. Dette kan gi angriperen tilgang til sensitiv informasjon som normalt ikke skulle være tilgjengelig.

Meltdown-angrepet er en alvorlig sikkerhetsrisiko, og det er viktig å installere de nødvendige sikkerhetsoppdateringene fra leverandørene av mikroprosessorene for å beskytte seg mot det.

Hvordan kompile et C++-program i terminalen, og kjøres med a.out som vanlig:

```
haugerud@studssh:~$ jed b.cpp  
haugerud@studssh:~$ g++ b.cpp  
haugerud@studssh:~$ jed
```

C og C++ er to forskjellige programmeringsspråk med ulike egenskaper og bruksområder. C++-koding er mer objektorientert.

VIKTIG FRA DATAMASKINARKITEKTUR

Alt CPU-en gjør er å slavisk utføre maskininstruksjoner en for en, i en evigvarende løkke. Ingen en til en forbindelse mellom instruksjoner i høynivåkode og maskinkode. En linje kode i høynivåspråk fører ofte til mange maskininstruksjoner i det kompilerte programmet.

Pseudo-kode for den evige hardware-løkken som CPU'en kjører:

```
while(not HALT)
{
    IR = mem[PC];      # IR = Instruction Register
    PC++;              # PC = Program counter
    execute(IR);
    if(InterruptRequest)
    {
        savePC();
        loadPC(IRQ); # IRQ = Interrupt Request
                        # Hopper til Interrupt-rutine
    }
}
```

Hvis den ikke har noen oppgaver å utføre vil den runne med no operation, som vil si den står og slår i lufta. Det har i mer moderne teknologier kommet CPU'er som kan legges i dvale modus. Med engang et interrupt kommer (brukeren taster inn noe eller hva som helst) så vil CPU-en gå og jobbe med den.

HVA ER UBUNTU OG DEBIAN?

Ubuntu og Debian er to ulike operativsystemer basert på Linux-kjernen.

Debian er en av de eldste og mest anerkjente Linux-distribusjonene, og er kjent for sin stabilitet og sikkerhet. Debian er utviklet av et stort, internasjonalt samfunn av utviklere og er gratis og åpen kildekode.

Ubuntu ble opprinnelig utviklet som en "nybegynner-vennlig" variant av Debian, og er nå en av de mest populære Linux-distribusjonene. Ubuntu har en mer brukervennlig tilnærming til å installere og konfigurere programvare enn Debian, og har også et større fokus på å være enkel å bruke og installere for nybegynnere.

Begge operativsystemene har en rekke likheter, inkludert at de begge er gratis og åpen kildekode, og de bruker mange av de samme programvarepakkene. Forskjellene ligger hovedsakelig i installasjon, konfigurasjon og brukeropplevelse.

HVA ER FORSKJELLEN PÅ WINDOWS OG LINUX?

Windows og Linux er to forskjellige operativsystemer med ulike egenskaper og funksjonaliteter.

Windows er et operativsystem utviklet av Microsoft. Det er kjent for sin brukervennlighet og det brede utvalget av programvare og spill som er tilgjengelige på plattformen. Windows er også det mest utbredte operativsystemet for stasjonære datamaskiner og bærbare PC-er.

Linux er et operativsystem som er utviklet av et stort fellesskap av utviklere og er basert på åpen kildekode. Linux er kjent for sin stabilitet, sikkerhet og fleksibilitet. Det er vanligvis gratis å bruke og tilgjengelig på en rekke forskjellige plattformer, fra små innebygde systemer til store servere.

En av hovedforskjellene mellom Windows og Linux er at Windows er proprietært, det vil si at kildekoden er hemmelig og eies av Microsoft, mens Linux er åpen kildekode, som betyr at kildekoden er åpen og tilgjengelig for alle. Dette gir Linux-brukere muligheten til å tilpasse og endre operativsystemet etter deres behov.

En annen forskjell er at Windows vanligvis kommer forhåndsinstallert på de fleste datamaskiner og er mer brukervennlig for nybegynnere, mens Linux ofte krever mer teknisk kunnskap og tilpasning for å sette opp og bruke effektivt.

Det er også forskjeller i programvaretilbudet på de to plattformene, med Windows som har et større utvalg av kommersielle programmer og spill, mens Linux er kjent for sin brede støtte for åpen kildekodeprogramvare og verktøy for utvikling og systemadministrasjon.

KORT OM HISTORIEN TIL MICROSOFT OS

- 1981: Microsoft lanserer MS-DOS, som stod for "Microsoft Disk Operating System". Dette var et tekstbasert operativsystem som ble levert med IBM PC-er og kompatibelt utstyr, og det var den dominerende standarden for PC-er i 1980-årene.
- 1985: Microsoft lanserer Windows 1.0, som var det første grafiske brukergrensesnittet fra Microsoft. Det var imidlertid ikke særlig populært på dette tidspunktet.
- 1987: Windows 2.0 lanseres, med bedre grafikk og mulighet for å overlappe vinduer.

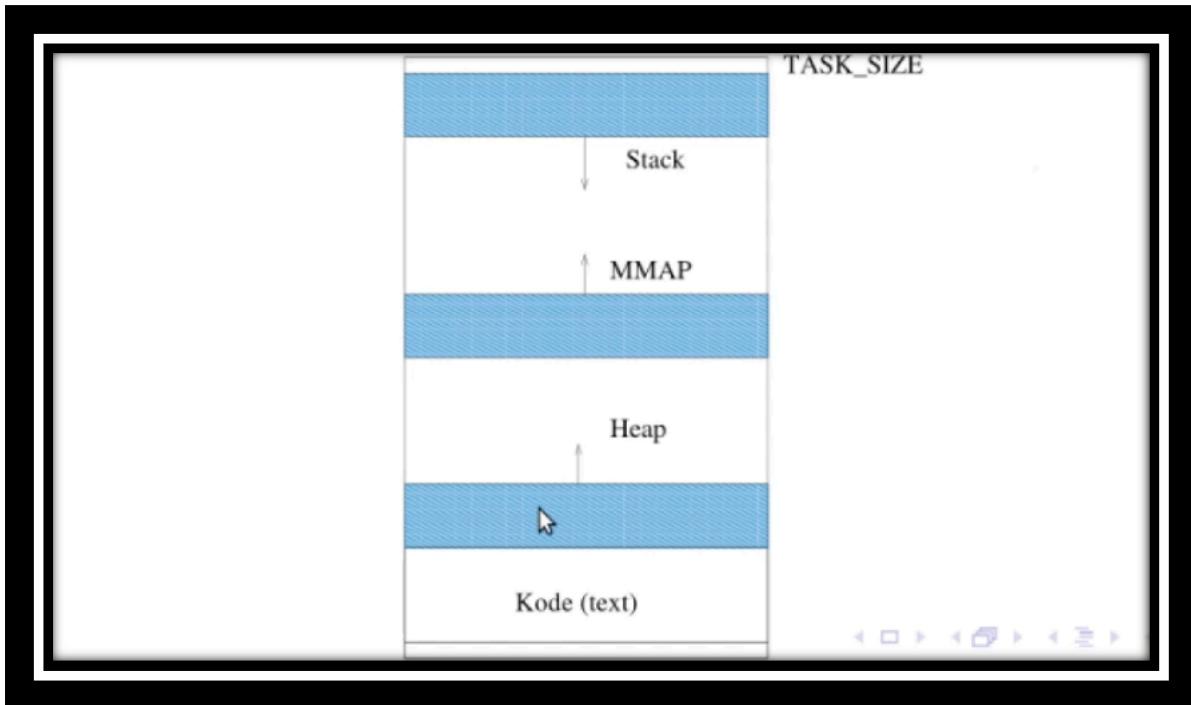
- 1990: Windows 3.0 lanseres, og blir en stor suksess for Microsoft. Det hadde bedre ytelse og støttet flere applikasjoner enn tidligere versjoner.
- 1995: Windows 95 lanseres, med et helt nytt brukergrensesnitt og bedre nettverksstøtte. Det ble en enorm suksess og solgte over 7 millioner kopier i løpet av de første fem ukene.
- 2000: Windows 2000 lanseres, som var beregnet for bedriftsmarkedet og hadde bedre sikkerhet og stabilitet enn tidligere versjoner.
- 2001: Windows XP lanseres, og blir en av de mest populære versjonene av Windows noensinne. Det var en stabil og brukervennlig versjon som ble brukt av millioner av mennesker over hele verden.
- 2006: Windows Vista lanseres, men det var ikke særlig populært blant brukerne på grunn av en rekke problemer og høye systemkrav.
- 2009: Windows 7 lanseres, som ble en av de mest populære versjonene av Windows noensinne på grunn av sin stabilitet og brukervennlighet.
- 2012: Windows 8 lanseres, med et helt nytt brukergrensesnitt som var beregnet for berørings-skjerm-enheter. Det ble ikke særlig populært blant brukere av vanlige datamaskiner.
- 2015: Windows 10 lanseres, og er den nyeste versjonen av Windows. Den har fått gode anmeldelser, og har en rekke nye funksjoner, inkludert en personlig assistent og muligheten til å kjøre universelle apper på tvers av forskjellige enheter.

Dette er selvfølgelig bare en kort oversikt, og Microsoft har gjort mye mer enn dette innen operativsystemer, inkludert utvikling av server-operativsystemer, mobil- og spillplattformer og mer.

CPU-en står og kjører (som sett i hardware-nivå koden), og av å til kommer det et fysisk interrupt request (f eks ved å taste på tastaturet). Maskinen må da reagere på den inntastingen veldig hurtig. Ofte bruker prosessoren et par mikrosekunder på å fullføre det den holdt på med også hopper den inn i if-løkka med interrupt requesten. Så CPU-en avbrytes for å håndtere signalet også lagrer den adressen til neste instruksjon på stack og hopper til interrupt-rutinen. Hver interrupt-nr (IRQ) har sin rutine.

For å kunne forstå multitasking er det viktig å forstå hvordan minne/RAM er koblet med prosessorer. Det finnes flere deler i RAM som er tildelt en prosess. Operativsystemet er også

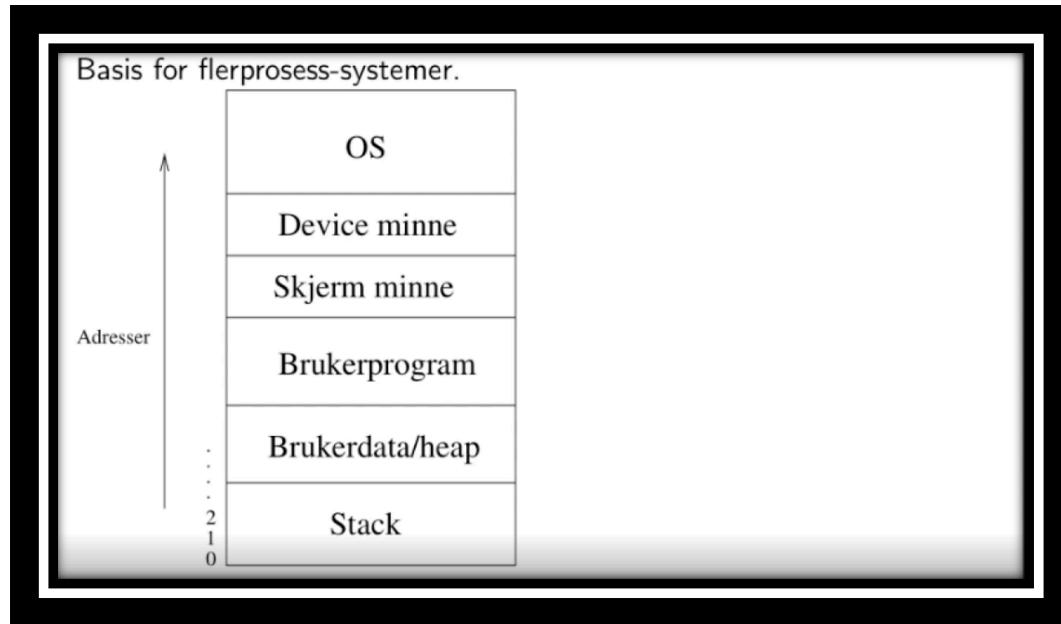
en egen oppgave og dermed har tildelt en egen del i RAM. Bildet under består av en bit av RAM som er tilordnet en prosess. Den viktigste delen er koden nede som er maskininstruksjoner som prosessoren skal kjøre, som ligger etter hverandre, instruksjon for instruksjon. Det typiske som skjer når en prosess kjører er at man starter på første operasjon og det løper nedover, også er det kanskje en loop som hopper opp og ned.



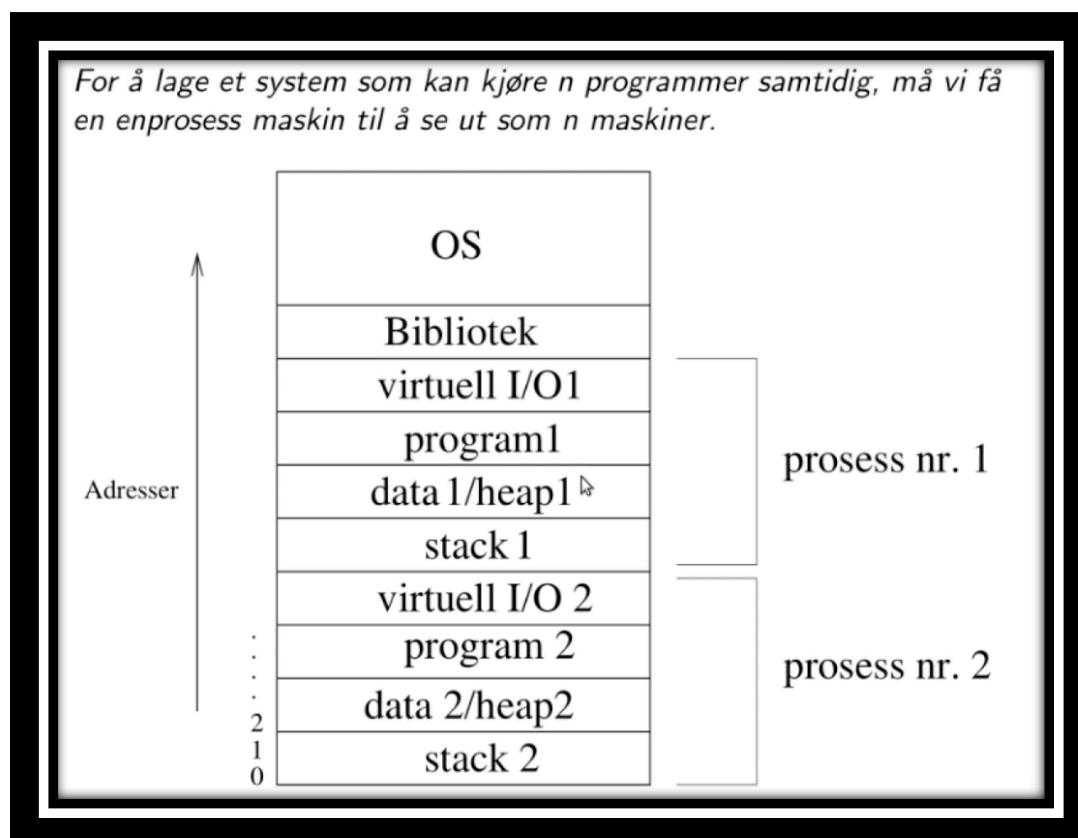
I heap'en kan globale variabler og data som dynamisk genereres lagres. Stacken er et område hvor lokale variabler lagres for eks: hvis man gjør et funksjonskall så legger man returverdien på stacken så man vet hvor man skal tilbake. MMAP, minneavbildninger av filer (og devicer) på disk direkte i virtuelle minnet, og det er egentlig for at ting skulle gå forttere. Men som sagt ligger koden og alle variabler i TEXT.

SINGLETASKING

Tidligere var singletasking måten datamaskiner kjørte på. Som vil si det var en prosess av gangen, enten brukerprogrammet eller OS.



Nå har alle moderne systemer multitasking, som har et antall programmer som kan kjøre samtidig. Måten de gjør det er ved å dele opp tiden og late som de kjører samtidig, men egentlig så går det litt her og der. Hvert område har sitt eget område med heap og stack osv. Og I områder, for å snakke med disk. Hver prosess har et område på RAM.



CPU-en gjør en og en maskininstruksjon av gangen. En multitasking operativsystem får det til å se ut som om mange programmer kan kjøre samtidig ved å dele opp tiden i små biter

(timeslices). Hver prosess som kjører får en bit CPU-tid, 1/100 sek av gangem, i et køsystem (Round Robin kø). Preemptive multitasking: en hardware timer (klokke) sender hvert hundredels sekund et interrupt-signal til CPU. Første OS-instruksjon legges inn i instruksjonsregisteret i CPU'en. OS lar hver prosess etter tur bruke CPU'en i et 1/100 sekund. Unngår at en brukerprosess henger eller tar over kontrollen. Alle prosesser ser da ut til å kjøre samtidig.

Når OS switcher fra prosess P1 til prosess P2 utføres en såkalt Context Switch (kontekst svitsj).

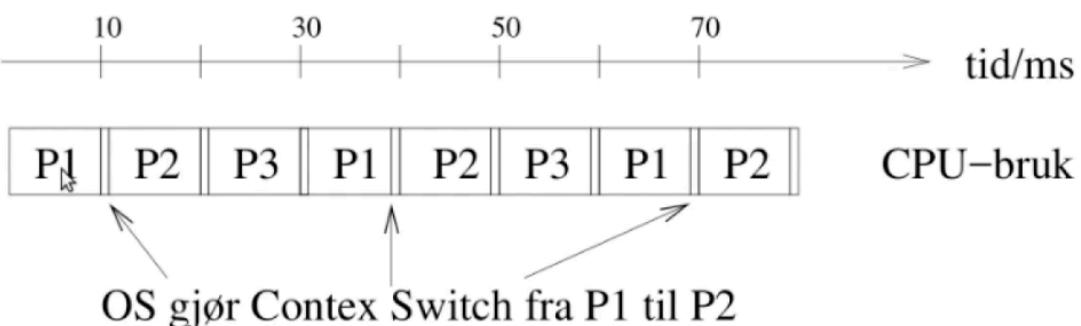


Figure: Prosessene P1, P2 og P3 kjører samtidig under et multitasking OS. En Context Switch utføres hver gang en prosess gis CPU-tid.

Typisk tid som brukes på en context-switch: 0.001 ms (ms = millisekund = tusendels sekund). Timeslice = 10 ms for Linux på Intel.

OM BILDET OVER: Når en context switch oppstår (fra P1 til P2) må OS lagre alt om prosessen P1 (verdier i registret, fordi alt som lagret i RAM er lagret). Og når det drar til P2 må den laste inn all infoen som var der.

PCB – PROCESS CONTROL BLOCK

En blokk I RAM som inneholder all infoen som trengs for en prosess for eks:

- CPU-registere
- Pekere til stack
- Prosesstilstand (sleep, run, ready, wait, new, stopped)
- Navn (PID)

- Eier (bruker)
- Prioritet (styrer hvor mye CPU-tid den får)
- Parent prosess
- Ressurser (åpne filer, etc)

SCHEDULING

(er en del av operativsystemet)

- CPU-scheduling = å fordele CPU-tid mellom prosessorene = Time Sharing
- Ved hvert timer-interrupt til CPU, vurderer OS om scheduler-rutinen skal kalles
- Scheduleren avgjør hvilken prosess som skal velges til å bruke CPU-en
- Når OS switcher fra prosess P1 til prosess P2 utføres en Context Switch
- All PCB-info må lagres i en Context Switch → tar tid → systemoverhead

I det siste bildet over kan det hende at ved siste P2 lengst til høyre så sjekkes det om P1 eller P3 trenger noe annet som skal gjøres, og hvis det så bare fortsettes det å jobbe på P2.

Oppgaver som å laste inn og laste ut verdier er det hardware som gjør i et smell.

MULTITASKING I PRAKSIS, CPU-INTENSIVE PROGRAMMER

Et program som oversetter kildekode til maskinkode (kompilator) eller et program som hele tiden regner med tall, vil bruke så mye CPU-tid som det klarer å få tak i. Kalles CPU-intensive.

De fleste vanlige programmer som browsere, tekstbehandlingsprogrammer, tegneprogram etc. bruker lite CPU.

Dette gjør at multitasking av hundretalls samtidige prosesser går helt greit uten at brukeren oppfatter datamaskinen som treg.

- Kjører et lite shell-script som står i en løkke og regner og regner
- Vil hele tiden bruke så mye CPU det kan få
- Har aldri behov for å vente på data fra disk, tastatur eller andre prosesser, kan regne uten stans

```
#!/bin/bash
# regn (bruker CPU hele tiden)
(( max = 100000 ))
(( i = 0 ))
(( sum = 0 ))

echo $0 : regner...
while (( $i < $max ))
do
    (( i += 1 ))
    (( sum += i ))
done
echo $0, resultat: $sum
```

CPU AVHENGIG PROSESS ^. Hvis det er flere slike programmer, må de hele tiden bytte om på å bruke CPU.

I/O PROSESS

Vanlige programmer som webbrowsers, teksteditorer, regneark etc bruker stort sett lite CPU.
Venter på I/O (Input/Output), input fra brukere, nettverkskort eller disk.

Bare rene regneprosesser bruker CPU hele tiden. Vanlige prosesser venter mye på I/O (Input/Output fra disk, nettverk etc.) og multitasking gir da mer effektiv utnyttelse av CPU.

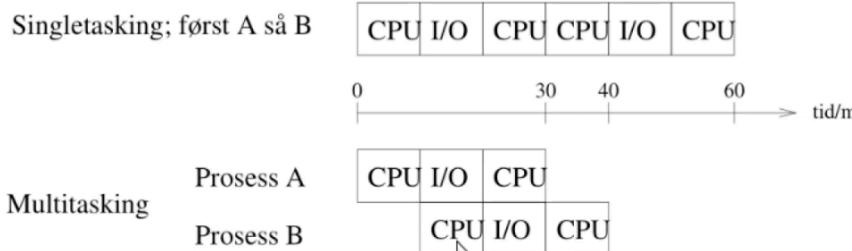


Figure: Prosessene A og B kjørt med single og multitasking

LINUX

Logge inn i VM-docker → ssh groupX@osX.vlab.cs.oslomet.no

sudo su er en kommando som brukes i Unix-liknende operativsystemer som Linux og macOS.

sudo er en forkortelse for "superuser do" og brukes til å gi midlertidig administrative rettigheter til en vanlig brukerkonto.

su står for "substitute user" og lar deg bytte bruker, som regel til en superbruker eller root-bruker.

Så sudo su lar deg midlertidig bytte til superbruker-kontoen på systemet, som gir deg fullstendig tilgang til alle filer og systeminnstillinger. Det er en kraftig kommando som bør brukes med forsiktighet, da feil bruk kan føre til utilsiktede endringer eller skade på systemet.

Hjem til root:

```
group100@os100:~$ sudo su  
[sudo] password for group100:  
root@os100:/home/group100# cd  
root@os100:~# pwd  
/root
```

GRUPPER OG LAGE NY GRUPPE PÅ LINUX-VM

Alle Linux-brukere er med i en eller annen gruppe. For å finne ut hvilken gruppe:

```
group100@os100:~$ groups group100  
group100 : group100 sudo
```

Man kan endre gruppe som en fil er med i ved å skrive kommandoen change group, hvilken gruppe man vil endre på og på hvilken fil man vil påføre endringene på.

```
group100@os100:~$ ls -l
total 32
-rwxrwxr-x 1 group100 group100 16704 Feb  8 22:03 a.out
-rw-rw-r-- 1 group100 group100    131 Feb 11 16:11 aptHistFeb10.2021.txt
-rw-rw-r-- 1 group100 group100     0 Feb  9 09:45 fil.txt
-rw-rw-r-- 1 group100 group100    190 Feb  8 22:03 run.c
-rw-rw-r-- 1 group100 group100    190 Mar  4 2020 run.c~
group100@os100:~$ chgrp sudo fil.txt
group100@os100:~$ ls -l
total 32
-rwxrwxr-x 1 group100 group100 16704 Feb  8 22:03 a.out
-rw-rw-r-- 1 group100 group100    131 Feb 11 16:11 aptHistFeb10.2021.txt
-rw-rw-r-- 1 group100 sudo      0 Feb  9 09:45 fil.txt
-rw-rw-r-- 1 group100 group100    190 Feb  8 22:03 run.c
-rw-rw-r-- 1 group100 group100    190 Mar  4 2020 run.c~
```

Man må være root for å lage nye grupper. Under lages det en nygruppe ved addgroup som heter studgruppe, med gruppeID 1001.

```
group100@os100:~$ sudo addgroup studgruppe
[sudo] password for group100:
Adding group `studgruppe' (GID 1001) ...
Done.
```

Her sjekker vi om det finnes en gruppe studg-noe i /etc/group.

```
group100@os100:~$ grep studg /etc/group
studgruppe:x:1001:
```

Under kan vi se at en bruker har blitt lagt til i studgruppe som heter group100.

```
group100@os100:~$ sudo adduser group100 studgruppe
Adding user `group100' to group `studgruppe' ...
Adding user group100 to group studgruppe
Done.
group100@os100:~$ grep studg /etc/group
studgruppe:x:1001:group100
```

Under kan vi se at han bytter gruppe til studgruppe i fil.txt.

```
group100@os100:~$ chgrp studgruppe fil.txt
group100@os100:~$ ls -l
total 32
-rwxrwxr-x 1 group100 group100 16704 Feb  8 22:03 a.out
-rw-rw-r-- 1 group100 group100 131 Feb 11 16:11 aptHistFeb10.2021.txt
-rwxrwxr-x 1 group100 studgruppe 0 Feb  9 09:45 fil.txt
-rw-rw-r-- 1 group100 group100 190 Feb  8 22:03 run.c
-rw-rw-r-- 1 group100 group100 190 Mar  4 2020 run.c~
```

Kommandoen «head /etc/group» vil vise de første linjene i filen «group» som ligger i mappen «/etc» på Linux-systemet. Filen «group» inneholder informasjon om brukergrupper på systemet. Hver linje representerer en brukergruppe, og feltene i hver linje er separert av kolon(:). De forskjellige feltene inneholder følgende informasjon:

- Gruppenavn
- GruppeID
- Gruppemedlemmer

En vanlig bruker har stort sett bare rettigheter til å gjøre endringer på sitt eget hjemmeområde:

```
group100@os100:~$ cd
group100@os100:~$ pwd
/home/group100
```

Kommandoen «sudo su» brukes til å skifte brukeridentitet på Linux-filsystemet. Når du skriver «sudo su» i terminalen og trykker enter så forventes det at du skal skrive passordet for å bekrefte at du har tillatelse til å utføre denne handlingen. Dersom innloggingen er vellykket vil man da være logget inn som superbruker (root-bruker) i terminalen. Som superbruker har du ubegrenset tilgang til systemet, og kan utføre alle handlinger uten restriksjoner, installere og fjerne programvare, og utføre systemadministrasjon.

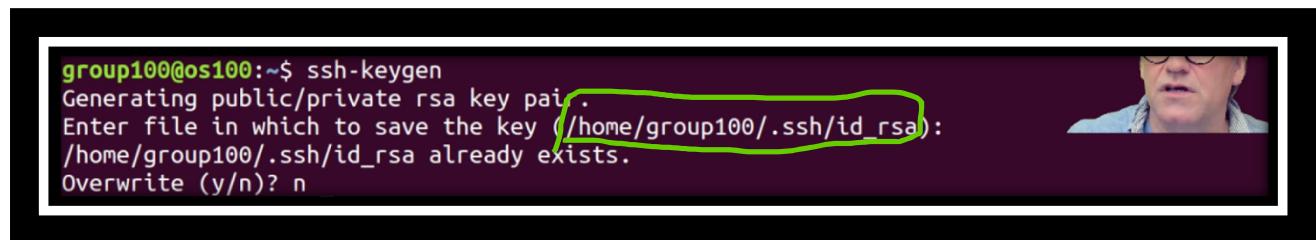
SSH

SSH (secure shell) er en protokoll for å koble til og kommunisere med en fjernmaskin over et sikret nettverk. Dette gir en kryptert og autentisert tilkobling mellom to systemer, slik at brukeren kan logge inn på en annen maskin på en sikker måte og utføre kommandoer på den eksterne maskinen som om han eller hun var til stede på den.

SSH-KEYGEN

«ssh-keygen» er et kommandolinjeverktøy som brukes til å generere og administrere SSH-nøkler på Linux og Unix systemer. Disse nøklene brukes til å autentisere brukeren som prøver å koble til en annen maskin ved hjelp av SSH-protokollen. En SSH-nøkkel består av en offentlig nøkkel og en privat nøkkel. Den offentlige nøkkelen blir vanligvis lagt til på den eksterne maskinen, mens den private nøkkelen beholdes på brukerens lokale maskin.

Kommandoen genererer et nytt nøkkelpar, også spørres det hvor nøklene skal ligges. Hårek tastet bare enter-tasten som er default. Det markert med grønn under er den private nøkkelen.



```
group100@os100:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/group100/.ssh/id_rsa):
/home/group100/.ssh/id_rsa already exists.
Overwrite (y/n)? n
```

Kommandoen «ls -l .ssh/» viser en detaljert liste over filene i mappen .ssh/, som egentlig er en skjult mappe som vanligvis brukes til å lagre SSH-nøkler og konfigurasjonsfiler for SSH. Hver linje representerer en fil eller mappe i ssh/-mappen og hvert felt gir følgende informasjon:

- Filrettigheter
- Antall hardlinker
- Eier
- Gruppe
- Filstørrelse
- Dato klokkeslett
- Filnavn

Etter å ha generert en ny nøkkel og trykket enter listes disse opp med den såkalte kommandoen. Da kan man i den skjulte mappen se at det opprettes to nøkler: id_rsa som er den private nøkkelen og id_rsa.pub som er den offentlige nøkkelen. Det som er innholdet i den offentlige nøkkelen, er det du skal legge inn et annet sted da du vil logge deg inn.

```
group100@os100:~$ ls -l .ssh/
total 16
-rw-r--r-- 1 group100 users      394 Feb  8 15:25 authorized_keys
-rw----- 1 group100 group100 2602 Feb  9 00:10 id_rsa
-rw-r--r-- 1 group100 group100   568 Feb  9 00:10 id_rsa.pub
-rw-r--r-- 1 group100 group100 1332 Feb 11 20:31 known_hosts
```

For å få ut innholdet i denne nøkkelen kan man skrive «cat .ssh/id_rsa.pub. D

```
group100@os100:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAABgQDXvydkQxWIYBuHLkrAoQkQZCFWys1F7WlnypIizOa0k5vhtm1atgTncxhHyD8dqp+bwddKIOare9lpUzHniK5ethN7lAqQtXw7/x00VQTwJ5nQQSd9XpZ63cMAQQQlbKthlA1pRYCqmRXWH3fODfrRVQ9Jnwox2Apde1pAvddWkIso8TwxnYXeY/VvZmmnzk0u16I+5IvkxE17r1DGnw3ZX/iMkf1YqjjDW3SP1frgZQ2kWGAJGBYdur6+RRv7X4+IDA1FEX3kBz10gS39P6uhxviYvWMnzEjMJLtvJ7/QzmowLiPgXAJAe80eSD6wCytzsZ1RTSUjhKQYwxqDnGobnzBvyRHfPEN2trp1KvtrvX+PgoxX5uFIYJYKiWNdu+2HaDWXnRzwAUW1nyNvHq74ip9enWp8W+5leH9aNZ6Fai1ciCPRkw6Rt/GxMncJVlpr5YpkjB/jpr5FS9j/QIGDK6dNIlcZdBujaffFx/HYGj4p0oMRM/o5OJ3/1a8dy+c= group100@os100
```

Hele nøkkelen skal kopieres til student ssh.

Kommandoen "ssh-copy-id" brukes til å kopiere en SSH-nøkkel til en fjernmaskin, slik at brukeren kan koble til fjernmaskinen ved hjelp av SSH-protokollen uten å måtte oppgi et passord. Når du kjører "ssh-copy-id", vil kommandoen be deg om å oppgi passordet ditt for brukeren på den eksterne maskinen. Deretter vil den kopiere den offentlige nøkkelen din til filen ".ssh/authorized_keys" på den eksterne maskinen. Når nøkkelen er lagt til i denne filen, vil du kunne koble til den eksterne maskinen ved hjelp av SSH-protokollen uten å måtte oppgi et passord.

Merk at både SSH-serveren og klienten må være installert og kjøres på begge maskinene for at denne kommandoen skal fungere. Du må også ha SSH-tilgang til den eksterne maskinen og ha tillatelse til å legge til din offentlige nøkkel på den eksterne maskinen.

```
group100@os100:~$ ssh s318329@studssh.cs.hioa.no hostname
studssh
group100@os100:~$ ssh s318329@studssh.cs.hioa.no 'hostname;uname -a'
studssh
Linux studssh 4.4.0-141-generic #167-Ubuntu SMP Wed Dec 5 10:40:15 UTC 2018
4 x86_64 GNU/Linux
```

Kommandoen under hvor authorized keys brukes er en fil som brukes av SSH for å autentisere en bruker på en annen datamaskin eller server ved hjelp av SSH-nøkler. Når SSH-nøkkelen til en bruker er lagt til i filen «authorized keys» på en ekstern server, kan brukeren logge seg på den eksterne serveren uten å måtte oppgi et passord.

Når en bruker prøver å logge seg på en annen datamaskin eller server ved hjelp av SSH-nøkler, sender SSH-klienten nøkkelen til serveren for å be om autentisering. Serveren sjekker om den offentlige nøkkelen som er sendt, tilsvarer en av nøklene som er lagt til i filen «authorized keys» for brukeren som prøver å logge på. Hvis nøkkelen matcher en av nøklene i filen, blir brukeren autentisert og logget på.

```
group100@os100:~$ cat .ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQJC6B5FiwOLHJpMQWv8AFR60wxNwz+RCQHe+3phqGeeCoIMplp9
naY61Ma300ts0iDuxQMA6UhN/aNgDSyN4HbjYD99tfar+6z87soblk8ZrCOWoUyoVWw16SkAs/nDQOk7D4GPyzK
clBEDSJPyeXMRLL0f3XdvhkS0dxpxfvobEUXNh0wKACwEzzd8VlOF7AFo+mOXhpeYU0UYwsh/Aj1JgpSz/5WMu3n
dZ38tX1re5SeKz8f6UIJ/ufI5AqxAGerx6Djgcxvkfo6P0iQ3UQNbbmy+87PuuCHVks/S4we5fmDOvaNX1g+AUNm
Cx4vD8FrSJXHPKuBJHicsT9jWm+V haugerud@lap
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQDFWstyMXFk8paQD5mpvpzHodxIXLi9SaT4RciteMa00T+NgJjr
X4q5a/rWoehaF3pLyhem0tK5NG090xmJf2WhXFppiVbillBEGdHZERbbhi+S4PP0KpJslVHgzkzlGq0eZWa/wC+
2DHpMeZLRLucKw5mBBbHXSE03C9eENGYc2awCM5sKP/8DHp5LJvDQDoQSkN9Mr8ChWz0uesaQZ+kztcRsu/wXx0T
Ky4iQFCQus6v8HYFE1D/0hVbgFuR8i6o7S0zE00eWetpAIS813wl7J86nMGa+bCwtAEuUG74EnB7yDBgb1wwcy/E
0o6JkyqdTTDOBfVkimUwv73IXnDB haugerud@studssh
```

SCREEN

«Screen» er en terminalmultiplexer for Unix/Linux-operativsystemer som lar deg opprette flere virtuelle terminalvinduer på samme tid. Dette kan være nyttig hvis du jobber med flere prosjekter eller oppgaver samtidig, eller hvis du trenger å fortsette en aktivitet i bakgrunnen mens du arbeider på andre oppgaver.

Når man åpner screen programmet i terminalen, vil man bli tatt med til et nytt terminalvindu. Fra dette vinduet kan du starte nye prosesser, navigere mellom forskjellige vinduer, og dele skjermen i flere seksjoner for å vise forskjellige prosesser samtidig. Du kan også la prosesser kjøre i bakgrunnen, selv om du logger terminalen eller mister tilkoblingen til serveren.

Her er noen grunnleggende kommandoene som kan brukes ved screen:

- «screen» - åpner et nytt screen vindu
- «ctrl+a c» - oppretter en ny terminal inne i skjermen
- «ctrl+a n» - går til neste skermvindu

- «ctrl+a p» - går til forrige skjermvindu
- «ctrl+a d» - detacherer skjermen og lar prosessene fortsette i bakgrunnen
- «screen -ls» - viser en liste over aktive screen-sesjoner
- «screen -r» - gjenopptar en tidligere screen-sesjon

Andre liknende programmer som screen er «tmux» og «byobu» som også tilbyr liknende funksjonaliteter. På våre VM er ikke screen installert så vi må først få det til ved «sudo apt install screen» og sette inn passord.

KJØRER EGENTLIG SCREEN?

Med kommandoen ps vil man ikke se det, men man må skrive «screen -ls».

Med kommandoen under kan man navngi et screen da man lager det.

```
group100@os100:~$ screen -S loop
```

Under lister han opp screen og hvis man skal starte screen igjen må man skrive «screen -r <navn>».

```
group100@os100:~$ screen -ls
There is a screen on:
    29378.pts-1.os100          (03/09/21 19:29:49)      (Detached)
1 Socket in /run/screen/S-group100.
group100@os100:~$ screen -r os100
There is no screen to be resumed matching os100.
group100@os100:~$ screen -r 29378
[detached from 29378.pts-1.os100]
```



Dersom man logger helt ut, altså kommer tilbake til stud ssh, og logger inn igjen kan man sjekke at screen fortsatt runner da man kobler til dem. Dersom man er inni et screen og vil scrolle opp og ned er det «ctr+a esc», som aktiverer en spesiell modus.

Dersom man ikke husker om man er i et screen eller ikke kan man bruke kommandoen \$STY:

```
group100@os100:~$ echo $STY
30337.loop
```

Og resultatet betyr at vi er i en screen.

SAMARBEID PÅ SCREEN

Under lages det et felles screen med auksjonen -dm og den kalles «felles».

```
group100@os100:~$ screen -dm -S felles
```

For å koble seg til en slik felles screen må man bruke en auksjon -x slik som følgende:

```
group100@os100:~$ screen -x felles
```

SECURE COPY

Kommando som brukes i Unix/Linux-terminalen for å kopiere filer eller mapper mellom to forskjellige datamaskiner over et sikret nettverk.

```
group100@os100:~/tmp$ scp enfil.txt haugerud@nexus.cs.hioa.no:~/mappe  
haugerud@nexus.cs.hioa.no's password:  
enfil.txt
```

100% 0 0.0KB/s

Med denne kommandoen kan man hente filer også:

```
group100@os100:~/tmp$ scp haugerud@nexus.cs.hioa.no:~/mappe/dinfile.txt .  
haugerud@nexus.cs.hioa.no's password:
```

100% 0 0.0KB/s

Kommandoen under vil kopiere over en mappe og da legger man til auksjonen -r for rekursiv.

```
nexus:~/mappe$ scp -r dir1 group100@os100.vlab.cs.hioa.no:~/tmp  
fil.txt  
nexus:~/mappe$
```

100% 0 0.0KB/s 00:00

RSYNC

Kommandoen rsync er intelligent og vil sjekke om hva som finnes, og ikke finnes, før den kopierer over ting og lager duplikater. "-a" -flagget står for "archive" og indikerer at "rsync" skal kopiere filene og mappene med alle tilhørende metadata og innstillinger, inkludert filtilgangsrettigheter, eierinformasjon og tidsstemplar. Hvis feilmeldingen under dukker opp betyr det at rsync ikke er lastet ned på begge sider.

```
haugerud@studssh:~$ rsync -a group100@os100.vlab.cs.hioa.no:~/tmp .
bash: rsync: command not found
rsync: connection unexpectedly closed (0 bytes received so far) [Receiver]
rsync error: remote command not found (code 127) at io.c(226) [Receiver=3.1.1]
```

Kommandoen «rsync –delete -a» er en kommando som samtidig sørger for at eventuelle overflødige filer i målmappen blir slettet.

CRONTAB

«Crontab» er en kommando som brukes for å opprette, endre eller vise planlagte oppgaver eller jobber som skal utføres automatisk på en bestemt tid eller med jevne mellomrom. En «crontab»-fil er en konfigurasjonsfil som inneholder en liste over tidsbaserte regler som forteller systemet når en bestemt oppgave skal utføres. Hver linje i «crontab»-filen representerer en separat oppgave, og angir tidspunktet eller tidsintervallene for når oppgaven skal kjøres.

«crontab -e» hvor -e auksjonen er for edit. Da åpnes det en fil som kan redigeres og nederst her har hårek lagt inn fila «sync.sh». her i crontab kan man ikke bruke ~ så man må oppgi fullpath til skript man kjører. Stjernene i starten oppgir hvor ofte koden skal kjøres. Ifølge det som står der nå vil skriptet kjøres hvert minutt.

```
haugerud@studssh:~$ crontab -e
# mail to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
* * * * * /iu/nexus/ua/haugerud/sync.sh
```

```
haugerud@studssh:~$ crontab -e
crontab: installing new crontab
haugerud@studssh:~$ date
sø. 21. feb. 01:33:13 +0100 2021
```

```
haugerud@studssh:~$ tail -f sync.log
Backup tatt Sun Feb 21 01:23:31 CET 2021
Backup tatt Sun Feb 21 01:24:31 CET 2021
Backup tatt Sun Feb 21 01:25:31 CET 2021
```

Cron er en tjeneste som lar deg automatisk utføre oppgaver på bestemte tidspunkter eller på et fast intervall. Du kan bruke den til å planlegge oppgaver som backup eller programvareoppdateringer på et spesifikt tidspunkt eller gjentas på faste tidspunkter. Du angir tiden og datoene for når oppgaven skal kjøre, og cron vil starte oppgaven automatisk for deg. På denne måten kan du frigjøre tid og slippe å huske å gjøre oppgaver manuelt.

WGET

Kommandoen wget er et program for nedlasting av filer fra internett. Det er en gratis og åpen kildekode-kommando for både Windows og Unix-liknende operativsystemer. Wget er designet for å være robust og pålitelig, og støtter nedlastinger av filer via HTTP, HTTPS og FTP protokoller, samtidig mange andre protokoller. Det er et populært verktøy for å laste ned store filer eller hele nettsteder, og kan brukes både fra kommandolinjen og via skripting.

Kan brukes til å hente websider:



```
haugerud@lap:~$ wget http://www.cs.hioa.no/~haugerud/regn
--2021-02-21 01:56:48-- http://www.cs.hioa.no/~haugerud/regn
Resolving www.cs.hioa.no (www.cs.hioa.no)... 128.39.89.23, 2001:700:700:3::23
Connecting to www.cs.hioa.no (www.cs.hioa.no)|128.39.89.23|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://www.cs.hioa.no/~haugerud/regn [following]
--2021-02-21 01:56:49-- https://www.cs.hioa.no/~haugerud/regn
Connecting to www.cs.hioa.no (www.cs.hioa.no)|128.39.89.23|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 194
Saving to: 'regn'

regn                                100%[=====] 194  --.-KB/s   in 0s

2021-02-21 01:56:49 (13,4 MB/s) - 'regn' saved [194/194]

haugerud@lap:~$
```

Hvis man skriver «cat regn»:

```
#!/bin/bash
(( max = 5300000 ))
(( i = 0 ))
(( sum = 0 ))
echo "regneprogram"

echo $0 : regner...
while (( $i < $max ))
do
    (( i += 1 ))
    (( sum += i ))
done
echo $0, resultat: $sum
haugerud@lap:~$
```

Med operatoren `-o` kan man sende utdata til en fil i stedet for standard utdata. Så kommandoen «`wget -o`» lar deg laste ned en fil fra internett og sende utdataen til en fil i stedet for å skrive den ut til terminalen.



```
haugerud@Lap:~$ wget -o log.txt http://www.cs.hioa.no/~haugerud/regn
haugerud@Lap:~$ cat log.txt
--2021-02-21 01:57:15-- http://www.cs.hioa.no/~haugerud/regn
Resolving www.cs.hioa.no (www.cs.hioa.no)... 128.39.89.23, 2001:700:700:3::23
Connecting to www.cs.hioa.no (www.cs.hioa.no)|128.39.89.23|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://www.cs.hioa.no/~haugerud/regn [following]
--2021-02-21 01:57:15-- https://www.cs.hioa.no/~haugerud/regn
Connecting to www.cs.hioa.no (www.cs.hioa.no)|128.39.89.23|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 194
Saving to: 'regn.1'

OK                                         100% 18,8M=0s

2021-02-21 01:57:15 (18,8 MB/s) - 'regn.1' saved [194/194]

haugerud@Lap:~$
```

Man kan se den lagrer til «regn.1» fordi det allerede ligger en fil der som heter det samme, så setter den på versjonsnummer.

Man kan også eksplisitt si hvilken fil man vil lagre det til ved operatoren `-O REGN`:



```
haugerud@Lap:~$ cat REGN
#!/bin/bash

(( max = 5300000 ))
(( i = 0 ))
(( sum = 0 ))
echo "regneprogram"

echo $0 : regner....
while (( $i < $max ))
do
    (( i += 1 ))
    (( sum += i ))
done
echo $0, resultat: $sum
haugerud@Lap:~$
```

Under er det en kommando som bruker «`wget`» verktøyet til å laste ned en fil fra en URL, og lagre loggen av nedlastingsprosessen i en fil som heter «log.txt». I tillegg, bruker kommandoen «`-O`» flagget for å skrive nedlastet innhold til standard utgang, i stedet for å skrive det til en fil. Dette kan være nyttig for å rute utdataen fra «`wget`» til andre verktøy i en kjede av kommandoer. Under ser man at output skrives direkte ut til standard output:



```

haugerud@lap:~$ wget -o log.txt -O - http://www.cs.hioa.no/~haugerud/regn
#!/bin/bash

(( max = 5300000 ))
(( i = 0 ))
(( sum = 0 ))
echo "regneprogram"

echo $0 : regner....
while (( $i < $max))
do
    (( i += 1 ))
    (( sum += i ))
done
echo $0, resultat: $sum
haugerud@lap:~$ wget -o log.txt -O - http://www.cs.hioa.no/~haugerud/regn

```

Slutten av kommandoen legger alt til i RE fila:



```

haugerud@lap:~$ wget -o log.txt -O - http://www.cs.hioa.no/~haugerud/regn > RE
haugerud@lap:~$ cat RE
#!/bin/bash

(( max = 5300000 ))
(( i = 0 ))
(( sum = 0 ))
echo "regneprogram"

echo $0 : regner....
while (( $i < $max))
do
    (( i += 1 ))
    (( sum += i ))
done
echo $0, resultat: $sum
haugerud@lap:~$ 

```

KOMPRIMERING FRA KOMMANDOLINJA

Kommandoen «ls -lh» vil liste som vanlig med filstørrelse og fila under på bildet har størrelse 293kilobytes:

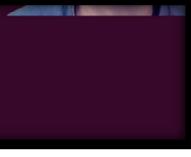


```

haugerud@studssh:~/zip$ cp /etc/passwd passwd
haugerud@studssh:~/zip$ ls -lh
total 296K
-rw-r--r-- 1 haugerud drift 293K feb. 21 02:15 passwd

```

«gzip for å komprimere fila passwd og lister igjen med ls -lh (merk forskjell på kilobytes):



```

haugerud@studssh:~/zip$ gzip passwd
haugerud@studssh:~/zip$ ls -lh I
total 88K
-rw-r--r-- 1 haugerud drift 86K feb. 21 02:15 passwd.gz

```

For å pakke ut bruker man gunzip passwd.gz:

```
haugerud@studssh:~/zip$ gunzip passwd.gz
haugerud@studssh:~/zip$ ls -l
total 296
-rw-r--r-- 1 haugerud drift 299905 feb. 21 02:15 passwd
```

Bzip2 er litt bedre til å komprimere, merk 64 kilobytes under:

```
haugerud@studssh:~/zip$ bzip2 passwd
haugerud@studssh:~/zip$ ls -l
total 64
-rw-r--r-- 1 haugerud drift 64913 feb. 21 02:15 passwd.bz2
```

Bzip2 og gzip er begge verktøy som brukes til å komprimere filer, men bzip2 har vanligvis en høyere komprimeringsgrad enn gzip. Dette betyr at bzip2 kan gi mindre filstørrelser enn gzip for de samme filene. Imidlertid er bzip2 også saktere enn gzip når det gjelder både komprimering og dekompresjon, så det kan være en trade-off mellom filstørrelse og ytelse avhengig av situasjonen. Generelt sett vil bzip2 være mer egnet for å komprimere store filer eller når plass er en begrenset ressurs, mens gzip vil være mer egnet for å komprimere mindre filer eller når rask komprimering/dekomprimering er viktigere enn filstørrelse.

Under vil ‘tar cf tmp.tar tmp’ opprette en ny tar-fil som heter ‘tmp.tar’ og inkluderer alle filene og underkatalogene i tmp-mappen. Kort forklaring for hver del av kommandoen:

- **tar**: Kommandoen for å lage og behandle tar-arkiver
- **c**: Forteller **tar** at du vil opprette et nytt arkiv
- **f**: Forteller **tar** at du vil bruke en spesifikk fil som arkivfil
- **tmp.tar**: Navnet på arkivfilen som skal opprettes
- **tmp**: Navnet på mappen som skal arkiveres

```
haugerud@studssh:~$ ls -l tmp
total 4
-rw----- 1 haugerud drift 0 feb. 21 02:10 dinfil.txt
drwxrwxr-x 2 haugerud drift 4096 feb. 21 02:10 dir1
-rw-rw-r-- 1 haugerud drift 0 feb. 21 02:10 NYFIL
-rw-rw-r-- 1 haugerud drift 0 feb. 21 02:10 nyfil.txt
haugerud@studssh:~$ tar cf tmp.tar tmp
```



```
haugerud@studssh:~$ ls -l tmp.tar  
-rw-r--r-- 1 haugerud drift 10240 feb. 21 02:17 tmp.tar
```

Tmp er ikke komprimert men den er pakket sammen. Pakke ut:

```
haugerud@studssh:~/zip$ tar xf tmp.tar  
tar: tmp: time stamp 2021-02-21 02:32:22 is 865.048722913 s in the future  
haugerud@studssh:~/zip$ ls -l  
total 80  
-rw-r--r-- 1 haugerud drift 64913 feb. 21 02:15 passwd.bzz  
drwxr-xr-x 3 haugerud drift 4096 feb. 21 2021 tmp  
-rw-r--r-- 1 haugerud drift 10240 feb. 21 02:17 tmp.tar
```

Husk at kommandoen «rm -rf» står for remove recursive og kan slette filer og mapper.

Hvis du finner en fil .tgz så er den både tar-et og zip-et.

Kommandoen ‘tar cfz tmp.tgz tmp’ oppretter en komprimert tar-fil med navn tmp.tgz fra mappen tmp og bruker gzip-komprimering for å redusere størrelsen på filen.

```
haugerud@studssh:~$ tar cfz tmp.tgz tmp  
haugerud@studssh:~$ ls -l tmp.tgz  
-rw-r--r-- 1 haugerud drift 234 feb. 21 02:18 tmp.tgz
```



Under kan det sees at i kommandoen tar xfz tmp.tgz står x for extract, f for file, og z for zip:

```
haugerud@studssh:~/zip$ ls -l  
total 80  
-rw-r--r-- 1 haugerud drift 64913 feb. 21 02:15 passwd.bzz  
-rw-r--r-- 1 haugerud drift 10240 feb. 21 02:17 tmp.tar  
-rw-r--r-- 1 haugerud drift 234 feb. 21 02:18 tmp.tgz  
haugerud@studssh:~/zip$ tar xfz tmp.tgz  
tar: tmp: time stamp 2021-02-21 02:32:22 is 776.991649872 s in the future  
haugerud@studssh:~/zip$ ls -l  
total 84  
-rw-r--r-- 1 haugerud drift 64913 feb. 21 02:15 passwd.bzz  
drwxr-xr-x 3 haugerud drift 4096 feb. 21 2021 tmp  
-rw-r--r-- 1 haugerud drift 10240 feb. 21 02:17 tmp.tar  
-rw-r--r-- 1 haugerud drift 234 feb. 21 02:18 tmp.tgz
```

VIKTIGE FILER

/etc/passwd er en systemfil i Unix- og Unix-lignende operativsystemer som inneholder informasjon om brukerkontoene på systemet. Hver linje i filen representerer en brukerkonto, og inkluderer informasjon som brukernavn, bruker-ID (UID), gruppe-ID (GID), hjemmekatalog og standard shell for brukeren. Filen brukes av systemet for å validere brukerkontoer og bestemme tillatelser for tilgang til ressurser. Det er vanligvis kun tilgjengelig for administratorbrukere på systemet.

Innholdet i /etc/shadow inneholder passordhasher for alle brukere på systemet, og dette skal ikke vises for andre enn administratorer med tillatelse til å se dem. Passordhasher er krypterte versjoner av passordene som brukes for å autentisere brukerne, og av sikkerhetsgrunner skal disse holdes hemmelige.

Hjemmekatalogen er katalogen som tilhører en spesifikk bruker på en Linux-basert datamaskin. Hjemmekatalogen kan inneholde alle brukerens personlige filer, inkludert dokumenter, bilder, konfigurasjonsfiler og programvare. Hjemmekatalogen er vanligvis plassert i /home/<brukernavn>.

Filen **/etc/group** inneholder informasjon om alle brukergrupper på systemet. Hver linje i filen beskriver en enkelt gruppe og inneholder felt som gruppenavn, gruppe-ID (GID) og en kommaseparert liste over medlemmer i gruppen. Gruppenavn og GID brukes til å identifisere og tildele rettigheter til gruppene, mens medlemslisten gir en oversikt over hvilke brukere som er medlem av gruppen.

LAGE NY BRUKER OG LEGGE TIL SUDO GRUPPEN

Man kan lage nye bruker med kommandoene adduser eller useradd. Sistnevnte er kun anbefalt dersom man skal legge til en bruker via et script eller lignende. Kommandoen adduser fungerer mer interaktivt og passer best for å legge til en og en bruker. Den kan brukes slik

```
group60@os60:~$ sudo adduser s123456
```

for å legge til brukeren s123456.

For å legge denne brukeren til sudo-gruppen slik at den blir en sudo-bruker, kan gjøres med

```
group60@os60:~$ sudo addgroup s123456 sudo
```

ROOT OG ANDRE BRUKERE



På et UNIX/Linux system er det vanlig å ha flere brukere i tillegg til de vanlige brukerne som er mennesker som bruker systemet. Disse ekstra brukerne er ofte tjenester eller programmer som kjører på systemet og de bør ikke kjøre som superbrukeren root av sikkerhetsmessige årsaker. Root-brukeren har UID 0 og har ubegrenset tilgang til systemet, noe som kan være farlig hvis den brukes feil. Derfor er det vanlig praksis å opprette en separat bruker for hver tjeneste eller programvare som krever tilgang til systemressurser. Eksempler på slike brukere er nobody, sshd og sys.

GRUPPER

6.5 Grupper

- gruppe av brukere
- Definert i /etc/group
- må defineres av root

```
$ groups haugerud          # lister gruppene haugerud er med i
$ chgrp studgruppe fil.txt    # fil.txt tilhører nå studgruppe
$ chmod 770 fil.txt        # alle rettigheter til alle i studgruppe
$ sudo adduser s123456 studgruppe # Gjør s123456 til medlem av studgruppe
```

Man kan lage en ny gruppe med addgroup:

```
mroot@os50:~$ sudo addgroup newgroup
[sudo] password for mroot:
Adding group `newgroup' (GID 1003) ...
Done.
mroot@os50:~$ grep newgroup /etc/group
newgroup:x:1003:
```

SECURE SHELL DAEMON (SSHD)



sshd står for Secure Shell daemon, som er en programvare for å opprette og administrere sikre, krypterte tilkoblinger mellom to datamaskiner over et usikkert nettverk. Sshd er en serverprosess som kjører på en vertsmaskin og lytter etter innkommende SSH-forbindelser fra klientmaskiner. Sshd gir brukere muligheten til å koble seg til vertsmaskinen og utføre oppgaver på den, som om de var til stede på samme fysiske sted. Sshd er en viktig komponent i sikkerhetsinfrastrukturen på mange datamaskiner og servere, og brukes ofte av systemadministratorer for å administrere og vedlikeholde systemene sine.

REGLER FOR BRUKERE

- Brukere kan kun sende signaler til sine egne prosesser (f.eks med kommandoen kill). Unntak er root.
- Brukere kan kun forandre på rettighetene til filer som de eier selv. Unntak er root.
- Brukere kan ikke "gi" filer til andre brukere ved å forandre hvem som er eier av filen med kommandoen chown. Root kan.
- Brukere KAN forandre hvilken gruppe som eier en fil med kommandoen chgrp, men KUN til grupper som den selv er medlem av. Root kan uansett.
- Brukere kan ikke lage nye brukere. Bare root kan det.

BYTTE EIERE (CHANGE OWNER)



`chown` er en kommando i Unix og Linux-systemer som brukes til å endre eierskapet til en fil eller mappe. "chown" står for "change owner". Når du bruker "chown" kommandoen, kan du endre både eieren av en fil/mappe og gruppen som filen/mappen tilhører. Kommandoen brukes vanligvis med superbruker-rettigheter (root-tilgang) for å endre eierskapet til filer og mapper som eies av systembrukere eller systemprosesser. For eksempel kan du bruke "chown" til å endre eierskapet til en fil fra "root" til en annen bruker, eller endre gruppetilhørigheten til en mappe fra "users" til en annen gruppe.

SUDO SU

Det finnes to måter å bli root på. Den ene er å bruke kommandoen su, som ber deg om root sitt EGET passord. Denne metoden er mest kjent. I de nyeste versjonene av Linux er det blitt mer vanlig å bruke sudo, som gir rettigheter til vanlige brukere dersom de kan autentisere seg. Man må vanligvis være medlem av en spesiell gruppe for å få lov til å kjøre kommandoen sudo su, som ber brukeren om sitt eget passord istedefor root sitt passord.

Det er også mulig å logge seg rett på systemet som root dersom man har root passordet. Det er derimot ikke anbefalt av følgende årsaker:

- Alt den brukeren gjør vil bli gjort med root-rettigheter.
- Ved å bruke su eller sudo su kan vi se i logfilene HVEM som ble root. Denne typen informasjon kan være veldig viktig når man vil finne ut hva som har skjedd (og hvem som skal få skylden).
- Root skal kun brukes når man har behov for det, og ikke behandles som en normal bruker/person.

SSH-COPY-ID

Logge inn på andre SSH-servere uten å måtte skrive passord hver gang. Dermed kan man velge å tillate kun innlogging med ssh-nøkler og unngå alle Brute-force-angrep. Først må man lage ssh-keys:

```
group1@os1:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/group1/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/group1/.ssh/id_rsa.
Your public key has been saved in /home/group1/.ssh/id_rsa.pub.
group1@os1:~$
```

Også kan man med ssh-copy-id (som kopierer over id_rsa.pub) sørge for at man senere kan logge inn uten passord.

```
group1@os1:~$ ssh-copy-id group49@os49.vlab.cs.oslomet.no
group49@os49.vlab.cs.oslomet.no's password:

group1@os1:~$ ssh group49@os49.vlab.cs.oslomet.no
Linux os49 2.6.32-5-xen-amd64 #1 SMP Fri Feb 5 17:48:36 UTC 2016 x86_64
group49@os49:~$
```

Man kan deretter også utføre kommandoer direkte på andre servere:

```
group1@os1:~$ ssh group49@os49.vlab.cs.oslomet.no whoami;hostname  
group49  
os1  
group1@os1:~$ ssh group49@os49.vlab.cs.oslomet.no "whoami;hostname"  
group49  
os49
```

BACKUP MED RSYNC OG CRON-TAB

Anta man ønsker å ta backup av /home/group1 på en os-VM. Det kan man nå gjøre med:

```
scp -r /home/group1/ haugerud@studssh.cs.oslomet.no:/home/haugerud/kopiaos1
```

og hele mappen og alle undermapper blir kopiert over. Men når man gjør det på nytt en gang til, vil alle filer kopieres over enda en gang. Linux-kommandoen `rsync` gjør det samme, men den kopierer bare over filer som har endret seg fra gang til gang:(Gjør først `$ sudo apt install rsync`):

```
rsync -a /home/group1/ haugerud@studssh.cs.oslomet.no:/home/haugerud/rsynckopiaos1
```

Når man lager en ny fil eller gjør en endring, er det bare dette som kopieres over neste gang.

Den enkleste måten å få dette til å bli en daglig rutine (eventuelt hver time) er å bruke cron.

På Linux-VM må cron først installeres med:

```
group100@os100:~$ sudo apt install cron
```

Om man kjører

```
crontab -e
```

Kommandoen crontab -e vil åpne og en fil og man kan redigere den nederst:

```
# Edit this file to introduce tasks to be run by cron.  
#  
# m h dom mon dow command  
  
30 1 * * * /home/group1/bin/rsyncStudssh.sh
```

Det fører til at skriptet kjører hver natt 01:30. Hvis man bytter ut tallet 1 med *, vil skriptet rsyncStudssh.sh kjøres 30 minutter etter hver hele time. Forkortelsene i linjen over forklarer syntaksen i crontab:

m	minute
h	hour
dom	day of month
mon	month
dow	day of week