OS OBLIG 1

- Jørgen Berntsen - s354410
- Atif Aziz - s354542
- Adrian August Grøtter - s354378
- Tony Strømsnæs - s354366

Uke 2

2.1. Hva er de to viktigste oppgavene til et operativsystem?

1. Det gir applikasjoner og brukere tilgang til datamaskinens ressurser gjennom en enhetlig, enkelt og mer abstrakt lag.
2. Den administrere ressursene til maskinen, og sørger for at prosesser og brukere ikke ødelegger for hverandre når de skal aksessere samme ressurser.

2.2. I figuren i forelesningsnotatene er det to feil i output på høyre side, hva er galt?

1. Or skal være 1, siden en av inputene er det.
2. Not skal være 1, siden input er 0;

2.3.Forklar utifra sannhetsverditabellene til AND og OR-portene hvorfor resultatet i den nederste kretsen blir 0, når øverste input er 1(rødt) og nederste input er 0(hvitt).

Det øverste signalet går inn i en AND-gate, hvor kun 1 av inputene er 1, så derfor blir outputen 0. Begge input-signalene må være 1 for at output skal bli 1.

Forklar utifra sannhetsverditabellene til AND og OR-portene hvorfor resultatet i den nederste kretsen blir 0, når øverste input er 1(rødt) og nederste input er 0(hvitt).

A	B	A AND B	AND OR B
1	1	1	1

A	B	A AND B	AND OR B
0	1	0	1
1	0	0	0
0	0	0	0

2.13. Bruk mkdir, cp og touch til å opprette en katalogstruktur som den på figuren, der passwd er en kopi av systemets passordfil mens fil1 og fil2 er tomme filer. Katalogen ~ er din hjemmekatalog.

- cp /etc/passwd .
- touch fil1
- mkdir etc
- cd etc
- mkdir bin
- touch fil2

2.14.Gi en kommando som flytter deg to kataloger oppover i filtreet.

```
cd ../../
```

2.15.Lag en mappe i din brukers hjemmekatalog. Gå inn i den mappen og lag noen tomme filer med kommandoen touch filnavn. Kopier alle filer i katalogen du står i som slutter på .java til katalogen over deg. Sørg for at du har laget noen slike filer først.

```
mkdir temp
cd temp
touch fil1.java fil2.java fil3.java
find . -name \*.java -exec cp {} .. \\;
```

2.16.List alle filer og kataloger under /usr/bin som har filnavn som begynner på "b".

```
$ ls /usr/bin/b*
```

2.18.I denne oppgaven skal du lage et lite shellscript. Start en editor med

```
$ jed info.sh
```

og skriv inn

```
#! /bin/bash<br>
whoami<br>
hostname <br>
uname -a
```

og lagre filen. Dette er et lite shellscript med navn info.sh som utfører de tre kommandoene når du kjører det. For at det skal bli kjørbart, må du sette kjørerettigheter på scriptet med

```
$ chmod 700 info.sh
```

og du skal kunne kjøre det ved å taste inn kommandoen

```
$ chmod 700 info.sh
```

Hvis det ikke går, prøv

```
$ chmod 700 info.sh
```

Hva er forskjellen på de to måtene å kjøre scriptet på?

Den første fungerer ikke fordi den ser etter scriptet i $PATH, og den andre fungerer fordi vi spesifiserer at scriptet ligger i mappen vi er i.

2.19.Start top i kommandolinjen og forklar hva du ser. Beskriv hvordan top er delt opp i to visuelle deler. Beskriv hva de to forskjellige delene viser og nevn de to datafeltene du mener er mest interessant i den øverste delen. For en forklaring av alle feltene, se "man

top". Prøv å taste "1" i top. Hva skjer og hvilken ekstra info får du nå? (tast "1" på nytt for å gå tilbake til slik det var)

Øverste del viser en totaloversikt over CPU, minnebruk, antall prosesser og deres tilstand. Andre del viser en oversikt over alle prosesser som er i live, og diverse info om de.

Når en trykker på "1" så vises oversikt over ressursbruk per CPU-kjerne, i stedenfor hele samlet.

2.20.Top har flere "hotkeys" av typen "1" som man kan bruke til å forandre hva som vises. Prøv å taste "U" i top og så ditt eget brukernavn og se hva som skjer. Ser du noe gevinst med å bruke top på denne måten? Gi eksempel på situasjoner hvor dette kan være nyttig.

Kan være greit å kunne se hvor mange ressurser en user bruker. Hvor lenge en prosess har kjørt f eks hos en bruker. Hvor mye cpu og memory eventuelt de bruker.

2.21. I de neste oppgavene skal du lage ditt eget alternativ til hva du fikk til i forrige oppgave. Du skal lage et shell-script som lister opp alle prosessene til en bruker.Prøv kommandoen "ps aux". Forklar kort utskriften til den kommandoen.

Kommandoen «ps aux» lister alle prosesser som kjører uavhengig av hvilken bruker de hører til og hvordan de ble startet.

2.22.grep er en kommando som kan plukke ut linjer som matcher en spesiell tekst. For at grep skal kunne plukke ut enkelte linjer fra ps aux, må vi lime dem sammen på et vis. Dette gjør vi med en pipe (engelsk for rør, pipe-tegnet er til venstre for 1-tasten):

```
ps aux | grep  tekst
```

Prøv ut den kommandoen selv og bytt ut "tekst" med noe mer fornuftig, f.eks et brukernavn. Forklar utskriften du får nå. Lag et shellscript som heter "psuser" som utfører denne kommandoen.

```
#! /bin/bash
ps aux | grep <user>
```

2.23.Det kan være praktisk å ikke måtte endre selve koden når man vil endre litt på hvordan et script kjører. Utvid shellscriptet "psuser", slik at det kan kjøres på denne måten: ./psuser root og det da skriver ut alle linjer som inneholder teksten root. Hint:

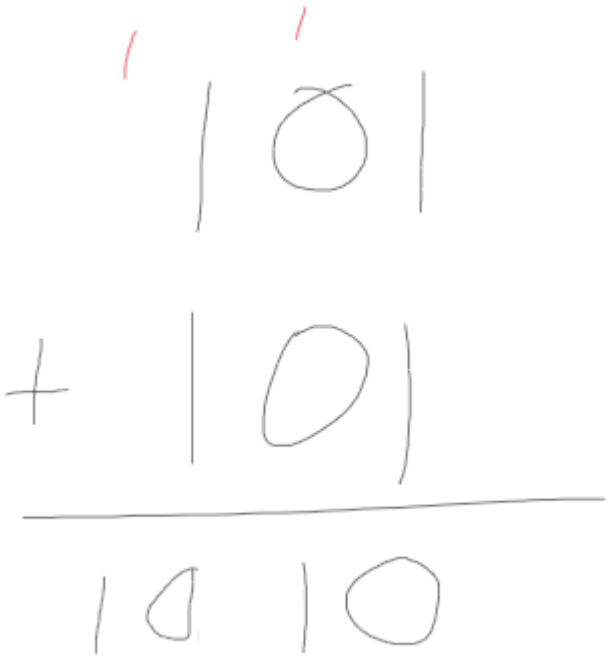
inne i scriptet vil argumentet root legges i variabelen $1. Dermed kan du erstatte teksten du vil lete etter med $1.

```
#! /bin/bash
ps aux | grep $1
```

```
$ ./psuser.sh <username>
```

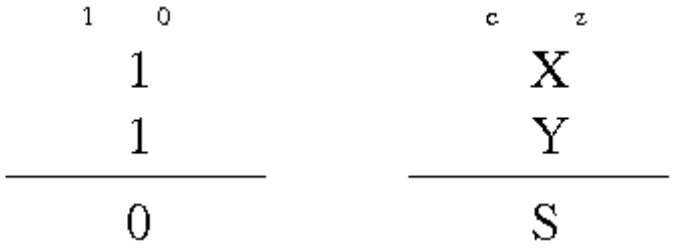
Uke 3

3.1. Utfør addisjonen 5 + 5 = 10 med binære tall for hånd ved å bruke den samme addisjonsmetoden som man bruker når man legger sammen desimaltall med penn og papir.



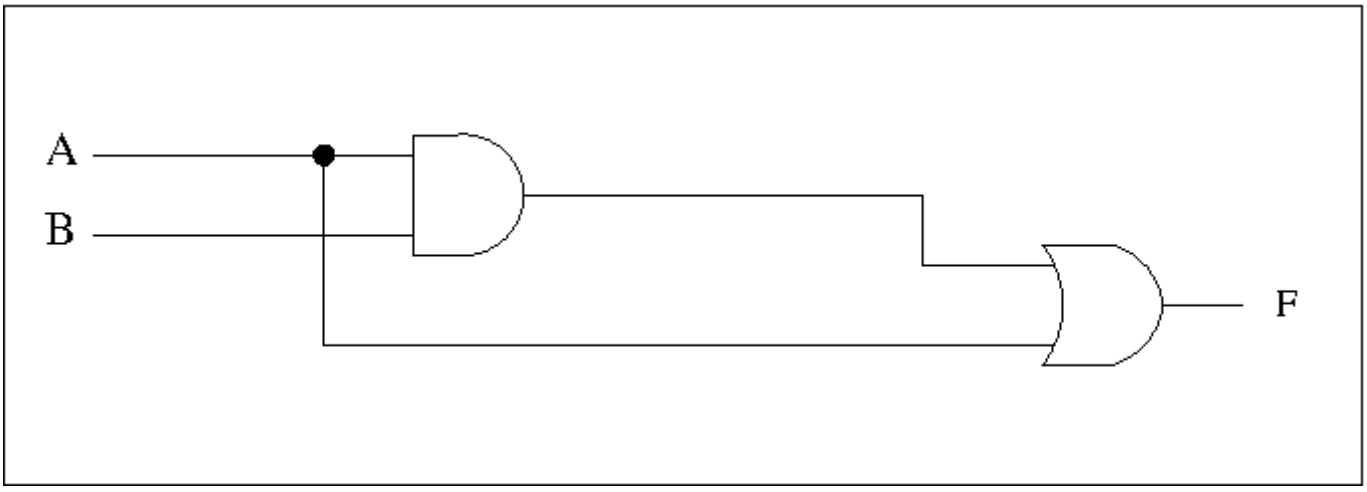
3.2. Se på hva som skjer når man legger sammen to siffer X og Y som står under hverandre i denne algoritmen. Anta at z er mente fra forrige operasjon til høyre for

sifferene vi behandler, sifferet under streken blir S og c blir mente til neste operasjon, slik det er vist i figuren under:



X	Y	Z	S	C
1	1	1	1	1
1	1	0	0	1
0	1	1	0	1
0	1	0	1	0
1	0	1	0	1
1	0	0	1	0
0	0	1	1	0
0	0	0	0	0

3.3. Hva blir den boolske funksjonen F som funksjon av A og B i figuren under? Skriv ned det boolske uttrykket.

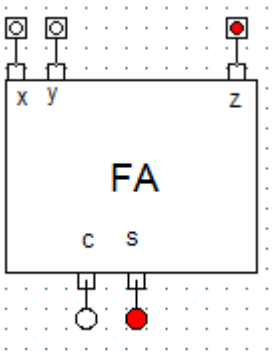


$F(A,B) = A \cdot B + A$

3.4. Forenkle det boolske uttrykket for F i forrige oppgave så mye som mulig.

F(A,B) = A

3.5.



3.8. Let deg frem til den eneste filen som ligger i en mappe på ditt hjemmeområde. Innholdet er et ord med 10 tegn. Finn dette ordet og skriv det inn i rapporten, ett ord for hvert medlem i gruppen. Skriv inn ordene sammen med s-nummeret det tilhører.

- s194 - 5cHVO5ZMUV
- s108 - klZ1phWjUI
- s253 - p071Zou8tV
- s15 - ddZB8HXqkx

3.10. Gå til katalogen /usr/bin ved å bruke absolutt path.

cd /usr/bin

3.11. Gå til hjemmekatlogen din. Gå deretter til katalogen /usr/bin ved å bruke relativ path.

- cd ../../usr/bin

3.12. Lag to filer med følgende innhold og kjør kommandoen diff

```
haugerud@data2500:~$ cat fa.txt
1a
2a
3a
4a
5a
haugerud@data2500:~$ cat fb.txt
1a
2a
3a
4b
5a
```

```
s194@os694:~/tmp$ diff fa.txt fb.txt
4c4
< 4a
---
> 4b
```

4c4 betyr linje 4 fa, c for change, og linje 4 i fb, det er der forskjellen er. < 4a betyr at venstre fil(fa.txt) innholder dette, og omvendt.

3.14. Utfør følgende Linux-kommandoer i en tom mappe:

```
mkdir tex
mkdir oblig
mv oblig tex
mkdir oblig
mv tex oblig
```

```
s194@os694:~/tmp$ tree
.
└── oblig
    └── tex
        └── oblig
```

3.15. Gå til roten (/) av systemet. Finn minst tre forskjellige måter å gå tilbake til hjemmektalogen din på.

1. cd ~
2. cd home/s194/
3. cd
4. cd $HOME

3.16. Skriv kommandoen echo bla bla bla > newfile, og bruk more for å se på filens innhold. Kan du forklare hva som har skjedd? Dette kalles redirection.

echo skriver ut meldingen i terminalen igjen, med med redirection sendes de til den spesifiserte filen i steden for terminalen.

3.25. Lag først en fil hemmelig.txt. Utfør en Linux-kommando som setter filrettighetene for filen hemmelig.txt slik at eieren av filen (du) kun kan lese den, mens alle andre ikke har noen rettigheter.

- chmod 400 hemmelig.txt

3.26. Lag først en fil fil.txt. Utfør en Linux-kommando som setter filrettighetene for filen fil.txt slik at eieren av filen (du) har alle rettigheter, medlemmer av filens gruppe har alle rettigheter untatt å skrive til filen, og alle andre kun kan lese den.

chmod 754 fil.txt

3.28. Hva blir rettighetene til en ny fil du lager med touch? Blir rettighetene de samme om du bruker en editor til å lage en ny fil? Bruk kommandoen umask til å sørge for at nye filer som lages kun kan leses og skrives til av eieren, mens alle andre ikke får noen rettigheter.

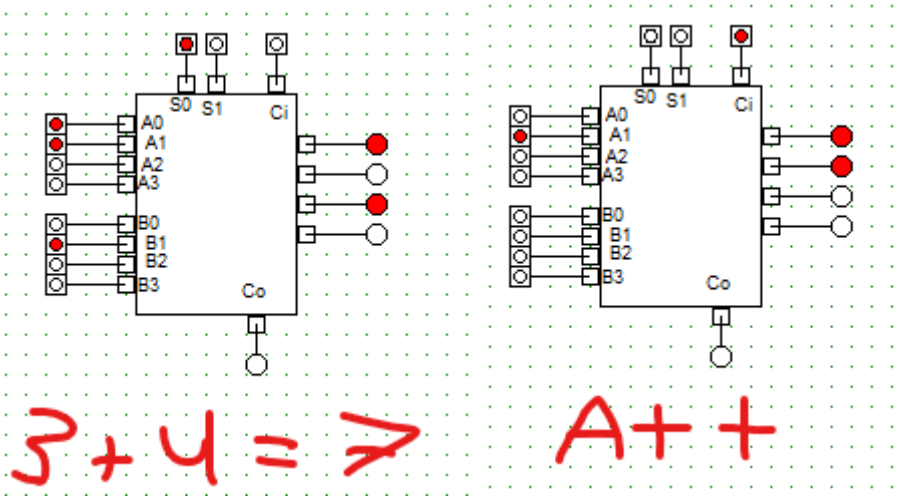
Med touch har filen har 664 rettigheter. Det samme med VIM og jed. umask 177 gir skrive-/leserettigheter til eier, og ingen til resten.

Uke 4

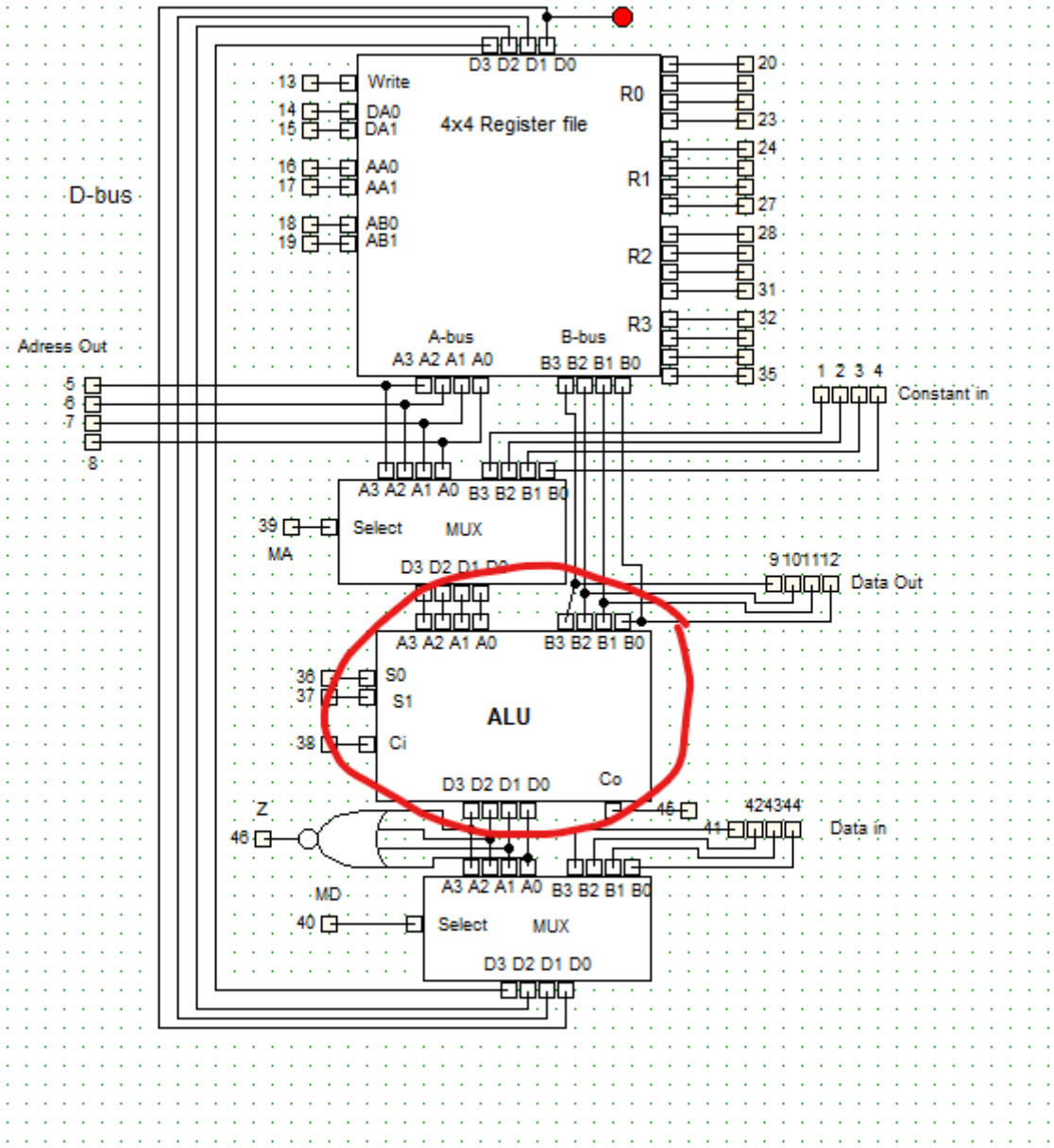
4.1. Last ned filen latch2.dwm Hva slags logisk krets er dette(dvs. hva kan denne kretsen brukes til)? Hvilken funksjon har de to input'ene A og B? Forklar og test at forklaringen er riktig ved å velge de fire mulige kombinasjonene av input. (Noen ganger får man en "race condition" i simulatoren. Det er tilfeller hvor det har betydning hviklet signal som kommer først frem og dette bør generelt unngås. Men se bort fra dette i denne oppgaven)

Det er en D-lås (data) krets. Verdien som kommer inn gjennom A kan lagres. Den lagres ved å skru på og av B.

4.2. Last ned filen 4bitALUtest.dwm åpne den og bruk den til å teste ALU'en som er den samme som brukes i maksinen i senere oppgaver. Gå til Uke 10 i menyen til venstre på siden datamaskinarkitektur og se på den nederste tabellen. Den viser hva input må være for å kunne få ALU'en til å utføre bestemte operasjoner. Bruk denne informasjonen til å få ALU'en til å gjøre regnestykket 3 + 4 = 7 og til å øke verdien av A med en.



4.4. Last ned filen machine.dwm som er en simulering av en 4-bits datamaskin og åpne den i Digital Works. Høyreklikk på boksene som utgjør denne maskinen, velg "Edit Macro" og prøv å lokalisere ALU. Hva er ALU og hva er dens funksjon?



ALU ligger inne i "Data path"-boksen. ALU står "Arithmetic Logic Unit", en enhet som kan gjøre følgende:

- Adder
- Subtraher
- Inkrement (++)
- Dekrement ($- -$)
- Multipliser
- Divider
- Shift (flytt alle bit i en retning)
- Sammenligne
- AND, OR, NOT, XOR

Den har forskjellige kontroll-bits som kan forandre hvilken operasjon som utføres.

4.5. Maskinen som simuleres er en liten CPU som kan kjøre programmer som ligger i boksen merket ROM. Høyreklikk på boksen og velg "Edit Memory Contents". Dette er maskininstruksjonene i programmet fra forelesningen som er en liten for-løkke. Kjør programmet ved å klikke deg igjennom med Step-knappen eller ved å trykke på play. Følg med i ROM og se hvordan instruksjonene blir utført en for en. Hva blir resultatet av beregningen og hvor ligger dette resultatet når beregningen er ferdig?

Resultatet lagres i R2, og er 3.

4.6. I forelesningsnotatene står det at i den simulerte datamaskinen er det "gjort visse endringer i forhold til von Neumann arkitekturen". Les forelesningsnotater eller let på nettet og finn ut hva som er den viktigste forskjellen på von Neuman arkitekturen og Harvard arkitekturen. Hvilken arkitektur ligner den simulerte datamaskinen mest på?

4.7. Ta utgangspunkt i DigitalWorks simuleringen av en datamaskin fra forelsningen og endre maskinkoden i ROM slik at løkken gjennomløpes bare to ganger og hvor S økes med to og ikke i, det vil si utfører maskinkode som er den kompilerte versjonen av følgende høynivåkode:

```
S = 0;
for(i=1;i < 3;i++)
{
   S = S + 2;
}
```

4.12. Logg inn som du tidligere har gjort til s-serveren hvor du har en s-bruker og hvor passordet ligger i s-gruppene i Canvas. Men bruk intel2 istedet for intel3 som du har brukt tidligere:

```
$ ssh -p 635 s135@intel2.vlab.cs.oslomet.no
```

Som før er portnummeret s-nummeret + 500, 635 er kun for s135. Da kommer du inn på en helt nylaget server, du har fortsatt tilgang til s-serveren på intel3 med alt du har laget der. Her er det en mengde mapper med forskjellige navn, 10 mapper som hver har 10 undermapper som igjen hver har 10 undermapper, altså totalt tusen mapper. Mappesystemet er forskjellig for hver eneste student, så

det hjelper ikke å se på naboen sine mapper. Nede i en av disse mappene ligger det en fil som heter file.txt og ukens praktiske oppgave er å finne denne filen og det ordet på 10 tegn som filen inneholder. Finn dette ordet og skriv det inn i rapporten, ett ord for hvert medlem i gruppen. Skriv inn ordene sammen med s-nummeret det tilhører.

```
find /home -name '*.txt'
```

Supercommand 😄 (hopper inn i mappen) :

```
cd "$(find -name '*.txt' -type f -printf '%h\n' -quit)"
```

- s194 - h9jOTAgq7A
- s253 - Q7lGly8a70
- s108 - Y31Jxa2vtC
- s15 - TPtzCCZVX4

4.13. Gi en Linux-kommando som gir deg brukernavnet ditt.

```
whoami
```

4.18. Hva skjer med eierskapet om man kopierer en annens fil? Prøv ved å kopiere en fil fra en annen bruker, for eksempel /home/haugerud/haugerud.txt på serveren data2500.

Den kopierte filen kommer under eierskap av brukeren som kopierte. Dette vil jeg tro skjer fordi vi har leserettighet, og derfor kan lese filen og skriver det i en egen ny fil.

4.21. På din private s-server (med intel3-innlogging), utfør en Linux-kommando som bruker grep til å skrive ut den linjen i /etc/passwd som inneholder ditt brukernavn, uten å bruke ditt brukernavn eksplisitt.

```
cat /usr/passwd | grep $USER
```

4.22. Tast inn inn ved shell-promptet de to linjene:

```
minvar=hei
export DINVAR=HALLO
start et nytt bash-shell med
bash
og taster deretter inn i dette shellet
$ echo $minvar
$ echo $DINVAR
```

minvar=hei lagrer hei i en variabel i shellet man er i, men når man åpner opp ett nytt, er det kun variabelen som ble eksportert som er tilgjengelig. Output blir :

```
s194@os694:~$ echo $minvar

s194@os694:~$ echo $DINVAR
HALLO
```

4.23. Lag et script vari.sh:

```
#! /bin/bash
min=hei
export DIN=HALLO
som du kjører med
$ ./vari.sh
og taster deretter
$ echo $min
$ echo $DIN
```

Output blir :

```
s194@os694:~$ echo $min
```

```
s194@os694:~$ echo $DIN
```

Dette var ikke som forventet, vi forventet samme output som forrige oppgave. Dette fungerer ikke på samme måte, fordi et bash-script kjøres i et eget subshell. Om man skriver

```
s194@os694:~$ source <script>.sh
eller
s194@os694:~$ . <script>.sh
```

vil det fungere. Da kjøres scriptet i samme shell.

4.24. Start en editor på data2500. Anta at editoren henger av en eller annen grunn. Logg inn på data2500 i et annet vindu og drep editor-prosessen med kommandoen kill. Hint: Du må finne editorens PID med ps.

Jeg fant prosessen med ps aux fra en annen terminal, og drepte den med kill.
Da avsluttet editoren i det andre vinduet med en gang med følgende melding:

```
***Fatal Error: Killed by signal 15.

jed version: 0.99.19/Unix
 Compiled with GNU C 9.2
S-Lang version: 2.3.2

jed compile-time options:
 +LINE_ATTRIBUTES +BUFFER_LOCAL_VARS +SAVE_NARROW +TTY_MENUS
 +EMACS_LOCKING +MULTICLICK +SUBPROCESSES +DFA_SYNTAX +ABBREVS
 +COLOR_COLUMNS +LINE_MARKS +GPM_MOUSE +IMPORT
CBuf: 0x557f76e645b0, CLine: 0x557f76e162e0, Point 11
CLine: data: 0x557f76e64580, len = 11, next: (nil), prev (nil)
Max_LineNum: 1, LineNum: 1
JWindow: 0x557f76e5dbc0, top: 1, rows: 27, buffer: 0x557f76e645b0
```

4.25.Bruk uname slik at navnet på OS'et som kjøres legges i variabelen $OS (etterpå skal echo $OS gi Linux).

```
s194@os694:~$ export OS="$(uname)"
s194@os694:~$ echo $OS
Linux
```

4.29. Hvis du leter etter filer med navn passwd på data2500 med følgende kommando

```
data2500:~$ find /  -name "passwd"
```

får du en masse linjer med "Permission denied". Lag en kommando som gjør at bare linjer hvor "passwd" blir funnet vises.

```
find / -name "passwd" 2>/dev/null
```

4.34. Lag et script usrbin.bash:

```
#! /bin/bash
cd /usr/bin
echo "er i $(pwd)"
som du kjører med
$ ./usrbin.bash
```

Hvor i filtreet er du etter at scriptet har kjørt? Forklar. Prøv å kjøre scriptet på en slik måte at du befinner deg i /usr/bin etter at det har kjørt.

Vi er på samme sted. For å kunne havne /usr/bin må vi kjøre scriptet i samme shell, enten ved bruk av 'source' eller '.' slik:

```
. usrbin.bash
```

Uke 5

5.1. Kompiler C-programmet Hello world fra forelesningen på data2500 med gcc hello.c og kjør det med

```
$ ./a.out
```

Kompiler det så med gcc hello.c -o hello. Hvordan kan du nå også kjøre programmet? Hvis du ikke har programmert i C før, kan denne C-tutorial være nyttig.

En kan kjøre den med å skrive

```
$ ./hello
```

5.2. Kompiler sum.c fra forelesningen sjekk at du får riktig svar. Kompiler så sammen sumMain.c og as.s fra samme forelesning med

```
$ gcc sumMain.c as.s
```

Endre så assembly-koden as.s slik at løkken gjennomløpes en gang til og sjekk at du får svaret 10. Hva er likheten mellom koden i as.s og maskinkoden i oppgave 31 i uke 3?

```
.globl sum
# C-signatur:int sum ()

# 64 bit assembly

# b = byte (8 bit)
# w = word (16 bit, 2 bytes)
# l = long (32 bit, 4 bytes)
# q = quad (64 bit, 8 bytes)


# Opprinnelige 16bits registre: ax, bx, cx, dx
# ah, al 8 bit
# ax 16 bit
# eax 32 bit
# rax 64 bit

sum:                # Standard
```

```
mov   $4, %rcx       # 4 -> rcx, maks i løkke
mov   $1, %rdx       # 1 -> rdx, tallet i økes med for hver runde
mov   $0, %rbx       # 0 -> rbx, variabelen i lagres i rbx
mov   $0, %rax       # 0 -> rax, summen = S

# løkke
start: # label
add  %rdx, %rbx # rbx = rbx + rdx (i++)
add  %rbx, %rax # rax = rax + rbx (S = S + i)
cmp  %rcx, %rbx # compare, er i = 3?
jne  start      # Jump Not Equal til start:

ret  # Verdien i rax returneres
```

```
jorber@JBMain:~/uke5$ gcc sumMain.c as.s
jorber@JBMain:~/uke5$ ./a.out
Sum = 10
```

5.3. Kopier as.s til en ny fil as2.s og endre assembly-koden slik at den utfører

```
S = 0;
for(i=1;i < 3;i++)
{
   S = S + 2;
}
```

istedet. Lag gjerne en ny versjon av sum.c også slik at du er sikker på at as2.s virker som den skal.

```
sum:                 # Standard

mov   $3, %rcx       # 3 -> rcx, maks i løkke
mov   $1, %rdx       # 1 -> rdx, tallet i økes med for hver runde
mov   $1, %rbx       # 0 -> rbx, variabelen i lagres i rbx
mov   $0, %rax       # 0 -> rax, summen = S

# løkke
start: # label
add  %rdx, %rbx # rbx = rbx + rdx (i++)
add  $2, %rax # rax = rax + rbx (S = S + 2)
cmp  %rcx, %rbx # compare, er i = 3?
```

```
jne  start      # Jump Not Equal til start:

ret  # Verdien i rax returneres
```

```
$ gcc sumMain.c as2.s -o as2
$ ./as2
$ Sum = 4
```

5.6. Skriv et shell-script som skriver ut verdien av den globale variabelen SHELL hvis den er satt og gir melding om at den er udefinert hvis den ikke er satt.

```
#! /bin/bash

if [ $SHELL ]
then
        echo $SHELL
else
        echo Variabelen er udefinert
fi
```

```
s194@os694:~/uke5$ . sr.sh
/bin/bash
s194@os694:~/uke5$ unset SHELL
s194@os694:~/uke5$ . sr.sh
Variabelen er udefinert
```

5.7. Lag et bash-script med navn publiser som setter rettighetene til alle filer i ~/www slik at eier kun kan lese og skrive, mens alle andre kun kan lese dem. I tillegg skal scriptet gi tilsvarende rettigheter for alle filer i ~/www/bilder. Hvis mappene ~/www og ~/www/bilder ikke eksisterer, skal de opprettes og gis alle rettigheter for eier, men kun lese- og kjørerettigheter for alle andre. Om det finnes andre kataloger i ~/www skal disse ikke endre rettigheter, heller ikke underkataloger og filer i disse.

```
#! /bin/bash

perm() {
```

```
if [ ! -e $1 ]
then
        mkdir $1
fi

chmod 744 $1

for fil in $1/*
do
        if [ -f "$fil" ]
        then
        chmod =r $fil
        fi
done
}

perm ~/www
perm ~/www/bilder
```

5.8. Lag et bash-script count.bash som skriver ut en oversikt over hvor mange linker, filer og kataloger det finnes i katalogen scriptet kjøres fra og i alle dens underkataloger. Bruker du doble parenteser rundt aritmetiske uttrykk ((x++)) kan du bruke samme syntaks som i Java (og sløyfe $ foran variabelnavn). Hint: Test kommandoene ls -R, tree -if og find . og se om noen av dem kan brukes i scriptet.

```
#! /bin/bash

filer=0
mapper=0
linker=0

for fil in $(tree $1 -i -f --noreport)
do
        if [ -d $fil ]
        then
                ((mapper++))
        elif [ -f $fil ]
        then
                ((filer++))
        elif [ -l $fil]
        then
                ((linker++))
        fi
        echo Leser fil $fil
done

echo Antall filer: $filer
```

```
echo Antall mapper: $mapper
echo Antall linker: $linker
```

5.11. Skriv et shell-script som tar en streng som argument og skriver ut en melding som avgjør om dette er en fil og om i såfall den er angitt med absolutt eller relativ path (om en relativ path er angitt til filen, skal scriptet sjekke om den finnes relativt til der scriptet kjøres fra. Merk: hvis man gjør en test på en fil i et script, utføres testen fra den mappen som scriptet starter i.). (hint: Testen if [-f $fil] ; then slår til om $fil er en fil. Bruk konstruksjonen ${variabel:offset:length} for å trekke ut ett tegn fra en streng. Eventuelt cut -c 1.)

```
#! /bin/bash

if [ -f $1 ]
then
        echo Er en fil
else
        echo Er ikke en fil
fi
```

5.12. Utvid scriptet i forrige oppgave slik at brukeren kan angi flere enn en fil. Gjør det ved å gå igjennom argumentene ett for ett og utføre det samme for hvert argument.

```
#! /bin/bash

for fil in "$@"
do
        if [ -f $fil ]
        then
                echo $fil er en fil
        else
                echo $fil er ikke en fil
        fi
done
```

```
s194@os694:~/uke5$ ./isFile2.sh sr.sh count.bash publiser.sh finnesikke!
sr.sh er en fil
count.bash er en fil
```

```
publiser.sh er en fil
finnesikke! er ikke en fil
```

5.15. Lag et bash-script "finnOrd" som tar ett ord som argument og går igjennom alle filer i katalogen scriptet blir utført fra og i alle underkataloger, og finner linjer i disse filene som inneholder dette ordet. For hver fil som inneholder ordet, skal scriptet gi meldingen:

```
########### Fant "ord" i fil "filnavn" i følgende linje(r):
```

Og så skrive ut alle linjene (Hint: bruk find . og grep).

```
#! /bin/bash

for fil in $(find .)
do
    if [ -f $fil ]
    then
        linje=$(grep $1 $fil)
        if [ $linje ]
        then
                echo "########## Fant $1 i fil $fil i følgende linjer:"
                echo $linje
        fi
    fi
done
```

```
s194@os694:~/uke5$ ./finnOrd.sh "Hei"
########## Fant Hei i fil ./fil2 i følgende linjer:
Hei
########## Fant Hei i fil ./fil5 i følgende linjer:
Hei
```

5.17. Logg inn på din private intel2 s-server som du har brukt tidligere med noe slikt som

```
cd ~/...
chmod u+x ...
./...
```

- s194 - oKQeWOa2lD
- s253 - Cq3ZKHKU6h
- s108 - 6Yev7Yqtbx
- s15 - 97gtNk1GFr