Ukesoppgaver 7 – Assembly og Shell-scripting

1. **(Oblig)**

main.c under:

```
[s364520@data2500:~$ cat main.c
#include <stdio.h>

extern int enlinje();

int main (void)
{
   int svar;
   printf("Kaller enlinje()...\n");
   svar = enlinje();
   printf("Svar = %d\n", svar);
}
```

en.s under:

```
[s364520@data2500:~$ cat en.s
.globl enlinje
# C-signatur:int enlinje ()

enlinje:                #Standard start av funksjon

mov memvar, %rbx        #Man trenger to linjer kode for åutfore en hoynivakode
add %rbx, svar
mov svar, %rax          #Returnerer svar

ret                     #verdien i rax returneres

#folgende avsnitt av koden viser hvordan man definerer variabler som lagres i minnet
#andre linje tilsvarer linjen int svar=32; i et c-program
#dette avsnittet kunne også statt overst i fila

.data
svar:   .quad 32        #deklarerer variabelen svar i RAM
memvar: .quad 10        #8 byte = 64 bit variabler
```

I bildet under er det først brukt gcc (kompilerer, lager maskinkode) til main.c og en.s. Så endrer jeg rettighetene til fila a.out som blir lagd for å kunne ha rettighetene til å kjøre fila. Svaret blir 42. Nede ved data settes svar og memsvar til to variabler til 32 og 10. I linje 6 settes memsvar sin verdi (10) inn i %rbx, i linje 7 legges verdien i %rbx (som nå er 10) inn i variabelen svar (som fra før har 32, 10 + 32 = 42), også er det en return nede. Det pleier å være verdien i %rax som returneres.

```
[s364520@data2500:~$ gcc -no-pie main.c en.s
[s364520@data2500:~$ chmod 700 ./a.out
[s364520@data2500:~$ ./a.out
 Kaller enlinje()...
 Svar = 42
 s364520@data2500:~$
```

2. (Oblig)

enlinje.c under:

```
[s364520@data2500:~$ cat enlinje.c
int enlinje()
{
   int svar = 32;
   int memsvar = 10;

   svar = svar + memsvar;

   return(svar);
}
s364520@data2500:~$
```

Kompilering av begge kodene og som vist er det samme output med svar 42:

```
[s364520@data2500:~$ gcc -no-pie main.c enlinje.c -o test
[s364520@data2500:~$ ./test
 Kaller enlinje()...
 Svar = 42
```

Her ber jeg gcc lage assemblykode for meg for å analysere koden.

```
[s364520@data2500:~$ gcc -S enlinje.c
[s364520@data2500:~$ cat enlinje.s
        .file   "enlinje.c"
        .text
        .globl  enlinje
        .type   enlinje, @function
enlinje:
.LFB0:
        .cfi_startproc
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        movl    $32, -4(%rbp)
        movl    $10, -8(%rbp)
        movl    -8(%rbp), %eax
        addl    %eax, -4(%rbp)
        movl    -4(%rbp), %eax
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   enlinje, .-enlinje
        .ident  "GCC: (Debian 10.2.1-6) 10.2.1 20210110"
        .section        .note.GNU-stack,"",@progbits
s364520@data2500:~$
```

Det markerte over med oransje tar for seg den enkle operasjonen: svar = svar + memsvar.

```
        movl    $10, -8(%rbp)
        movl    -8(%rbp), %eax
        addl    %eax, -4(%rbp)
        movl    -4(%rbp), %eax
```

Når en deklarasjonstype endres for en variabel i en C-funksjon fra «int» til «long long», vil det påvirke størrelsen og representasjonen av denne variabelen i minne og dermed også endre den resulterende assembly-koden. Ettersom «long long» typen er en 64-bits verdi, vil den kreve dobbelt så mye minne som en «int», som typisk er en 32-bits verdi. Det vil medføre endringer i instruksjonene som brukes for å allokere minne til variabelen, samtidig instruksjonene som brukes til å utføre operasjoner på denne variabelen.

Så det å endre type-deklarasjon fra «int» til «long long» påvirker ytelsen til programmet, da det kan kreve mer tid for CPU-en til å utføre instruksjoner på større tall. Det kan også føre til at programmet bruker mer minne, siden større tall krever minne til å lagres.

3. UKENS UTFORDRING:

Assemblyprogram som:

Returnerer 1 hvis svar > 0 Returnerer -1 hvis svar < 0 Returnerer ellers 0

Assemblykoden min:

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

iftest:         # Standard start av funksjon

#Returnerer 1 hvis svaret > 0
#Returnerer -1 hvis svaret er < 0
#Returnerer ellers 0

#if (svar > 0) {
#       return(1):
#}
#elif (svar < 0) {
#       return (-1):
#}
#else {
#       return (0):
#}

mov $0, %rbx

cmp %rbx, svar  #compare
jg greater      #hvis svar er > 0
jl less         #hvis svar er < 0

mov $0, %rax    #ellers sett 0 i %rax, og da vil det returneres
jmp return

greater:
mov $1, %rax
ret

less:
mov $-1, %rax
ret

return:
ret     #verdien i rax returneres

.data
svar:   .quad 40        #deklarerer variabelen svar i RAM
```

Høynivåkoden ifMain.c i C-kode:

```
[s364520@data2500:~$ cat ifMain.c
#include <stdio.h>

extern int iftest();

        int main (void) {
                int svar;
                printf("Kaller iftest()...\n");
                svar = iftest();
                printf("Svar = %d\n", svar);
        }
```

Kompilerer dem sammen også tester:

```
[s364520@data2500:~$ gcc -no-pie iftest.s ifMain.c -o if
[s364520@data2500:~$ ./if
Kaller iftest()...
Svar = 1
s364520@data2500:~$
```

4. UKENS UTFORDRING 2

C-koden til fibo.c:

```
[s364520@data2500:~$ cat fibo.c
int fibo(int last)
{
        int i;
        int a=1,b=1,c;
        /*Har allerede de forste to*/
        for(i=3;i <= last;i++){
                c = a;
                a = a + b;
                b = c;
        }
        return(a);
}
```

C-koden til fiboMain.c:

```
[s364520@data2500:~$ cat fiboMain.c
#include <stdio.h>

extern int fibo();

int main(void)
{
        int last=47;
        int res = fibo(last);
        printf("Res: %d \n",res);
}
```

Når jeg regner ut Fibonacci-tall nr 46 med disse kodene:

```
[s364520@data2500:~$ gcc -no-pie fibo.c fiboMain.c -o fibonacci
[s364520@data2500:~$ ./fibonacci
Res: -1323752223
```

Hva skjer om du regner ut tall nr 47 i rekken? Hvorfor skjer dette? Prøv å endre type fra `int` til `long long` og om du da kan klare å regne ut tall nr 92 i rekken. Hva skjer om du regner ut tall nr 93? Hvorfor skjer dette?

```
        .section        .note.GNU-stack,"",@progbits
[s364520@data2500:~$ vim fiboAssembly.s
 s364520@data2500:~$
```

5. UKENS UTFORDRING

```
[s364520@data2500:~$ cat regnHashVerdi.sh
#! /bin/bash

hashpass='$6$AB.f/K06$IsV3oABaBO4UEBertVwViFgqFcuRvPfBDBVojDJkwg43AlPlgfD.y8nCpjnb01EgwwrVaxpYRzYjgT5G1g4lw.'
salt=$(echo "$hashpass" | cut -d$ -f 3)

#3 tegn langt passord med små bokstaver
for p in {a..z}{a..z}{a..z}
do
    sjekk=$(echo -n "$p" | mkpasswd -m sha-512 -S "$salt" -s)
    if [ "$sjekk" = "$hashpass" ]
    then
        echo "Passordet er: $p"
        exit 0
    fi
done

#echo "Kunne ikke finne passordet."
#exit 1
```

Under kjøres koden og gir svaret: tre

```
[s364520@data2500:~$ ./regnHashVerdi.sh
Passordet er: tre
s364520@data2500:~$
```

6. Forklaring av resultatene til kommandoer:

Kommandoen skriver ut «dette er en fisk» til standardutgangen. På den andre siden tar kommandoen «sed s/test/fisk/» denne utdatastrømmen som input og erstatter ordet «test» med «fisk».

```
[s364520@data2500:~$ echo dette er en test | sed s/test/fisk/
dette er en fisk
s364520@data2500:~$
```

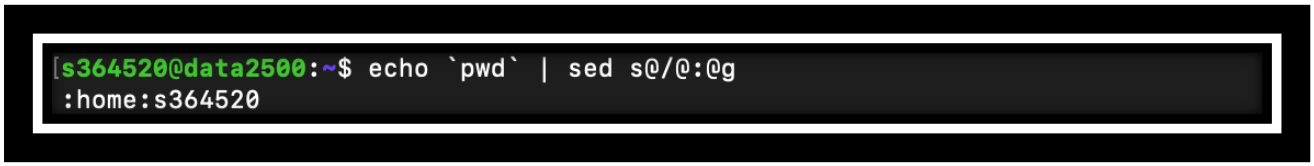
Echo skriver ut «test og test» til stdout men sed kommandoen erstatte den forekomsten av ordet «test» med ordet «fisk».

Echo skriver ut «test og test» men sed bytter ut alle forekomster av test med fisk.

Akkurat samme som den over:

Kommandoen "echo pwd" skriver ut den nåværende arbeidsmappen (på formen /path/to/current/directory/) til standardutgangen ved hjelp av pwd-kommandoen. Kommandoen "sed s/:/:/g" tar denne utdatastrømmen som input og erstatter alle forekomster av kolon (:) med dobbeltpunkt (:). Siden det ikke er noen kolon i utdataen fra pwd-kommandoen, vil sed-kommandoen ikke gjøre noen endringer på teksten.

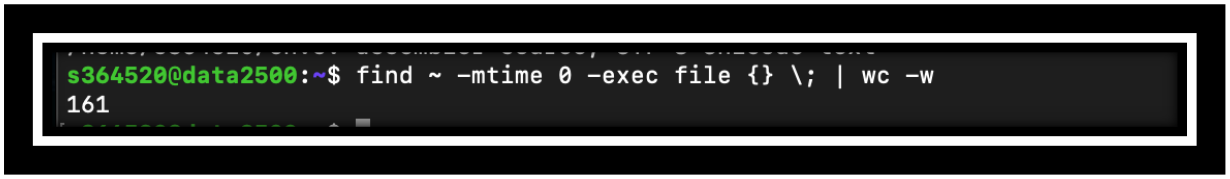
Kommandoen "echo pwd" skriver ut den nåværende arbeidsmappen (på formen /path/to/current/directory/) til standardutgangen ved hjelp av pwd-kommandoen. Kommandoen "sed s@/@:@g" tar denne utdatastrømmen som input og erstatter alle forekomster av skråstrek (/) med kolon (:). Siden det ikke finnes, skjer det ikke.



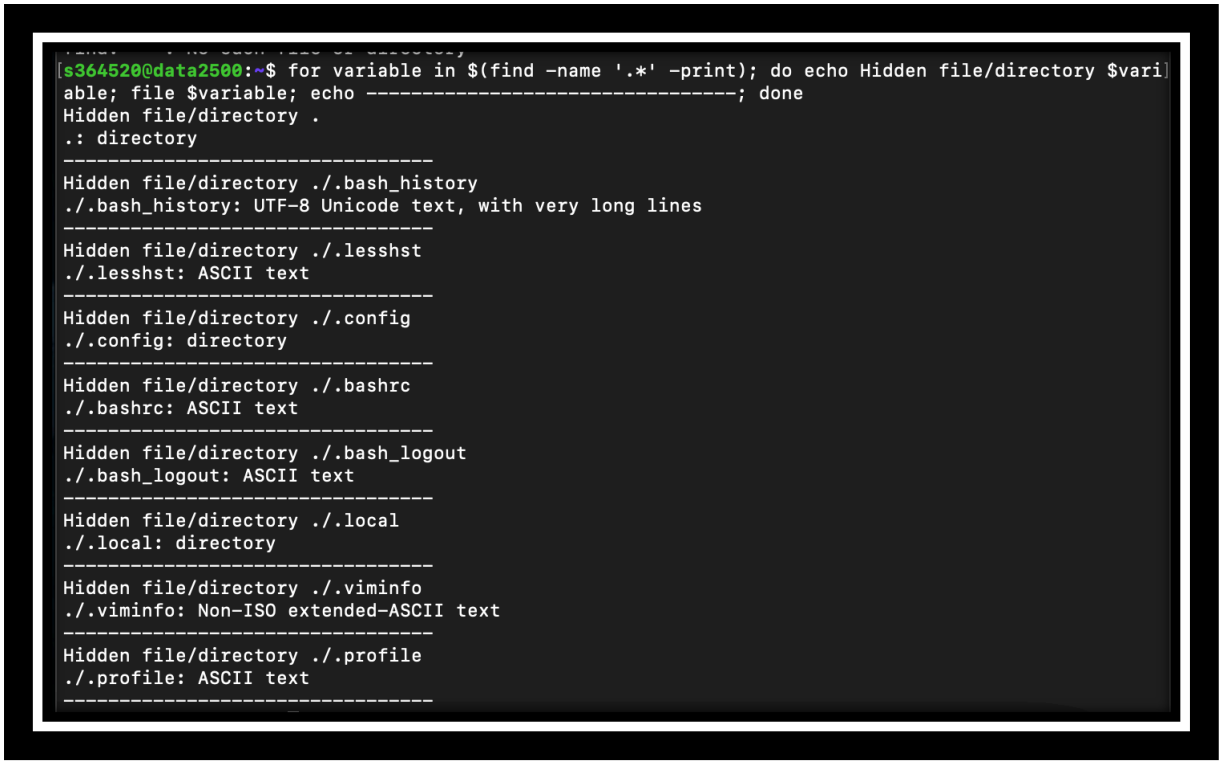
Akkurat samme som forrige.

Søker for filer i hjemme directory av den nåværende brukeren som sist ble oppdater for mer en dag siden og printer dens navn.

Søker for filer i hjemme directory i den nåværende brukeren for sist oppdatering i de siste 24 timene, og for hver fil som befinner seg da, skal den 'execute' «file» kommandoen for å bestemme dens fil type.



Søker for skjulte filer/directories, for hvert funn skal fil-type bestemmes.



7. (Oblig) Filer som ble redigert ila en dag.

```
[s364520@data2500:~$ find . -newermt "30 Jan 2023" ! -newermt "31 Jan 2023"
[s364520@data2500:~$ find . -newermt "10 Feb 2023" ! -newermt "11 Feb 2023"
./SHELL.sh
./count.bash~
./www/alle
./www/hei
./www/bilder
./www/bilder/hallo
./www/bilder/hei
./www/bilder/du
./www/sammen
./esum.c
./count.bash
./as2.s
./sum2.c
./SHELL.sh~
./publiser.sh
./publiser.sh~
./esum.c~
```

8. (Oblig) Kommando som bytter ut inneholder stud.hioa med Oslomet til en ny fil

```
s364520@data2500:~$ sed 's/stud.hioa.no/oslomet.no/g' oppgavefil.txt > nyoppgavefil.txt
[s364520@data2500:~$ cat ny
nyoppgavefil.txt  nyttScript.sh
[s364520@data2500:~$ cat ny
nyoppgavefil.txt  nyttScript.sh
[s364520@data2500:~$ cat nyoppgavefil.txt
s802399@oslomet.no
[s886878@oslomet.no
s886876@oslomet.no
s886885@oslomet.no
s886884@oslomet.no
s850806@oslomet.no
s886873@oslomet.no
s886888@oslomet.no
s808855@oslomet.no
s856627@oslomet.no
s886878@oslomet.no
s299507@oslomet.no
s850798@oslomet.no
s803434@oslomet.no
s886879@oslomet.no
s886877@oslomet.no
s959938@oslomet.no
s886880@oslomet.no
s826650@oslomet.no
s886882@oslomet.no
s886874@oslomet.no
s859987@oslomet.no
s803833@oslomet.no
s886885@oslomet.no
s364520@data2500:~$
```

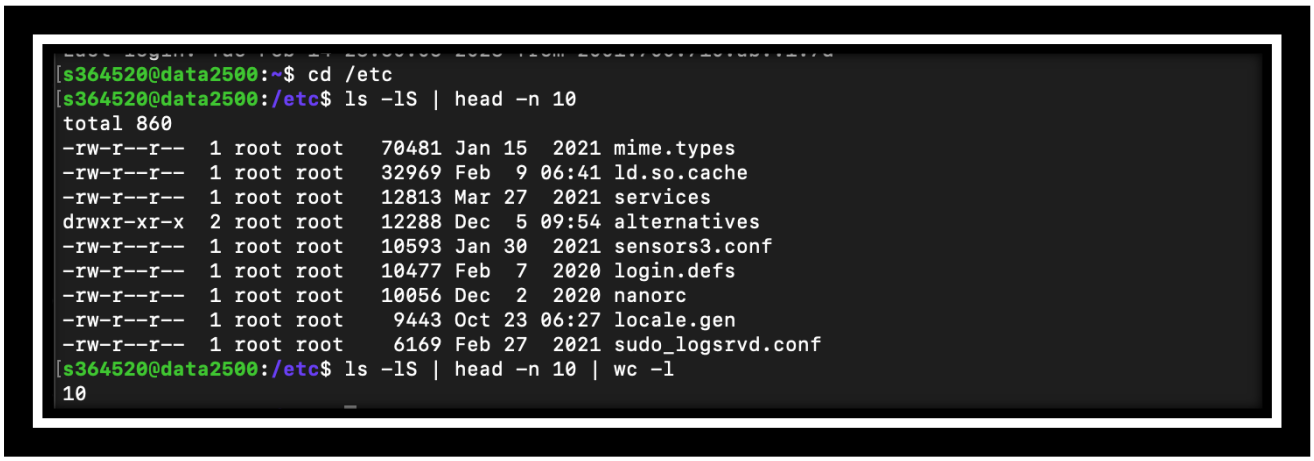
9. Mener koden under skal virke, men den gjorde ikke det, er usikker på hvorfor ikke.

```
[s364520@data2500:~$ cat navnfil.txt | awk '{print $4}' | sed 's/@stud.hioa.no/@oslomet.no/' > nynavnfil.txt
```

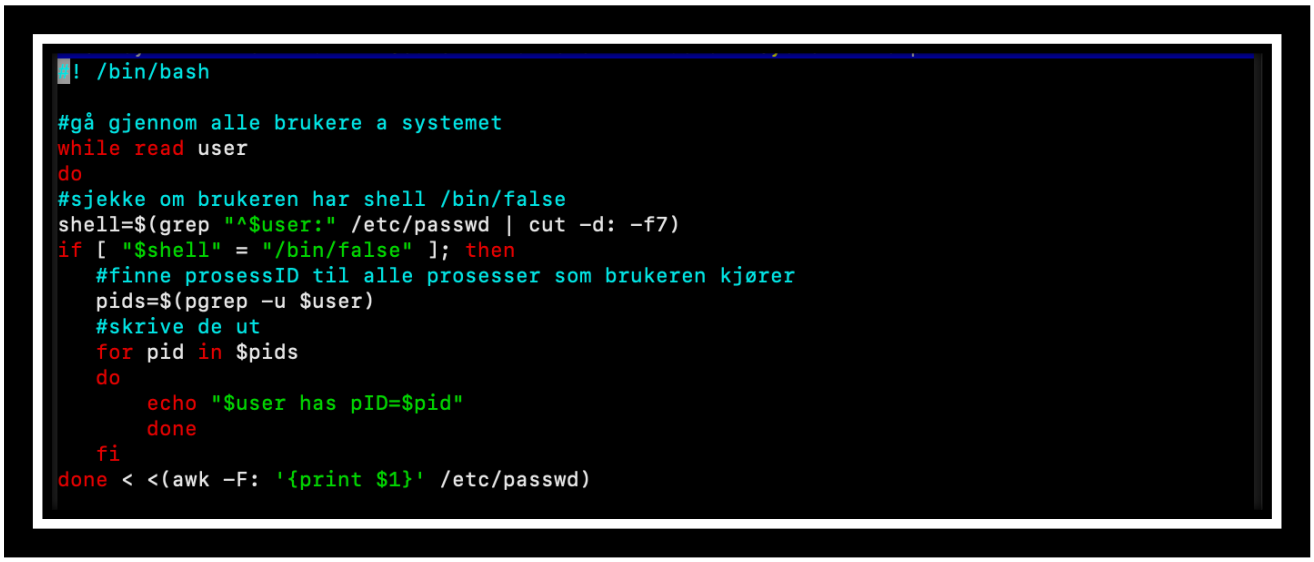
I bildet under klarte jeg å filtrere ut kun epostene med å bruke awk. Awk sin tilleggsbetingelse '$4 ~ /@/' sjekker om det fjerde elementet i hver linje inneholder "@"-tegnet.

Navnfil.txt og nynavnfil.txt

10. de 10 største filene listet opp. Hvis man vil ha med mapper også er kommandoen -laS



11. jed oppgave11.sh koden er under:



Etter å ha endret rettigheter «chmod 700 oppgave11.sh» og kjørt scriptet:

```
[s364520@data2500:~$ ./oppgave11.sh
Debian-snmp has pID=582
```

12.

$0: representerer navnet på skriptet som kjører. Hvis skriptet blir kalt med et absolutt eller relativt filnavn, vil $0 inneholde dette navnet. Hvis skriptet blir kalt med et kommandonavn, vil $0 inneholde navnet på kommandoen.

$$: representerer PID (prosess-ID) for skriptet som kjører. Dette er et unikt nummer som identifiserer skriptprosessen.

```
#! /bin/bash

# du har et brukernavn $whoami og kjører skriptet $0 med PID = $ på maskin

# $hostname som kjører med operativsystemt $uname som har versjonsnummer
# $uname -r

# oversikt over din hjemmekatalog $HOME:

# Filer:$(find . -type d | wc -l)
# Linker:$(find . -type f | wc -l)
# Kataloger:$(find . -type l | wc -l)

# Ditt default shell er $SHELL og du befinner deg i katalogen $pwd
# Totalt $(cut -d: -f1 /etc/passwd | sort | uniq | wc -l) brukerer er oppfort i passordfilen og det er definert $(cut -d: -f4 /etc/passwd | sort | uniq | wc -l) grupper

# Oversikt over dine grupper:
# whoami : $GROUPS :



echo Du har brukernavn $(whoami) og kjører skripet $0 med PID = $$ på maskinen
echo $(hostname) som kjører
:
echo Operativsystemet: $(uname)
echo Versjonsnummer: $(uname -r)

echo Oversikt over din hjemmekatalog "$HOME"
echo Filer:$(find . -type d| wc -l)
echo Linker:$(find . -type f | wc -l)
echo Kataloger:$(find . -type l | wc -l)

echo Ditt default shell er $SHELL, og du befunner deg i katalogen $(pwd)
echo Totalt $(cut -d: -f1 /etc/passwd | sort | uniq | wc -l) brukere er oppf?rt i passordfilen og det er definert $(cut -d: -f4 /etc/passwd | sort | uniq | wc -l) grupper
```

Under er det endret rettigheter til 700 på skriptet også kjørt:

```
[s364520@data2500:~$ ./info.sh
Du har brukernavn s364520 og kjører skripet ./info.sh med PID = 870843 på maskinen
data2500 som kjører
Operativsystemet: Linux
Versjonsnummer: 5.10.0-21-amd64
Oversikt over din hjemmekatalog /home/s364520:
Filer:14
Linker:94
Kataloger:0
Ditt default shell er /bin/bash, og du befunner deg i katalogen /home/s364520
Totalt 32 brukere er oppf?rt i passordfilen og det er definert 29 grupper
```

13. UKENS NØTT 1

```
#! /bin/bash

#lese 5 første linjene av /etc/host, lage to array: ip og navn
#etter skriptethar kjørt:5 første ip og 5 første navn
#og?så skrive ut lengden på arrayene og hva som er i 3 elem i begge

#tomme array
ip=()
navn=()

i=1

while read -r ipAdresse hostname && [ $i -le 5 ]; do
      ip+=("$ipAdresse") #legger til ip adresse i ip array
      navn+=("$hostname") #legger til hostname i navn array
      ((i++)) #øker telleren med en
done < /etc/hosts #leser fra /etc/hosts

#skriver ut lengden av ip- og navn-arrayene, og hva som er på indeks 2 i hver av dem
echo "Totalt: ${#ip[@]}"
echo "Array 3: ${ip[2]} ${navn[2]}"
```

Med koden gitt over endres det rettigheter og kjøres under.

```
[s364520@data2500:~$ jed arr.sh
[s364520@data2500:~$ chmod 700 arr.sh
[s364520@data2500:~$ ./arr.sh
Totalt: 5
Array 3: 128.39.89.61 csstud.cs.hioa.no csstud
```

14. UKENS NØTT 2

Koden under som tar og leser (-r, og ikke blir avbrutt av noe før telleren er ferdig) og skriver ut for hver linje. Fullføres når i er lik 5. Setningen etter done gjør at det er fra /etc/hosts verdier leses fra:

```
#! /bin/bash

#leser fem første linjer /etc/hosts, lager assosiativt array
#etter kjøring inneholder 5 første ip-adressenemed navn som indeks

#tomt array ip
declare -A ip

i=0
while read -r ipAdresse hostname && [ $i -le 5 ]; do
#echo "${hostname} sin IP er $ipAdresse"
    ip["$hostname"]=$ipAdresse
    ((i++))
done < /etc/hosts

echo "Totalt: ${#ip[@]} elementer i lista."

#for ((j=1; j<=i; j++)); do
#    read -r ipAdresse hostname < <(sed -n "$j{p;q}" /etc/hosts)
#    ip["$ipAdresse"]="$hostname"
#done

for nokkel in "${!ip[@]}"; do
    echo "Hostnavn:  ${ip[$nokkel]}, IP: $nokkel"
done
```

Endrer rettigheter og kjører

```
[s364520@data2500:~$ chmod 700 arr2.sh
[s364520@data2500:~$ ./arr2.sh
Totalt: 6 elementer i lista.
Hostnavn:  128.39.89.65, IP: studssh.cs.hioa.no studssh
Hostnavn:  127.0.0.1, IP: localhost
Hostnavn:  128.39.89.23, IP: nexus.cs.hioa.no nexus
Hostnavn:  128.39.89.61, IP: csstud.cs.hioa.no csstud
Hostnavn:  128.39.89.11, IP: ssh.cs.hioa.no ssh
Hostnavn:  127.0.1.1, IP: data2500.cs.oslomet.no        data2500
s364520@data2500:~$
```

EFFICODE! MITT BRUKERNAVN ER: DASTGIR S264

```
[s264@os5264:~/...$ cd ~
[s264@os5264:~$
[s264@os5264:~$ find -type f
./.bash_logout
./.bashrc
./.profile
./.bash_history
./dir.tgz
./dir/__pycache__/PatSyn/nvmem/file.txt
./.cache/motd.legal-displayed
./.../...
./skjult/.skjult/.skjult.txt
./.ssh/authorized_keys
[s264@os5264:~$ cd ./.../
[s264@os5264:~/...$ ls -l
total 12
drwxrwxr-x 2 s264 s264 4096 Feb 12 22:19 dir1
drwxrwxr-x 2 s264 s264 4096 Feb 12 22:19 dir2
drwxrwxr-x 2 s264 s264 4096 Feb 12 22:19 dir3
[s264@os5264:~/...$ ls -la
total 40
drwxrwxr-x 5 s264 s264  4096 Feb 12 22:19 .
drwxr-x--- 1 s264 s264  4096 Feb 12 22:19 ..
-rw-rw-r-- 1 s264 s264 16800 Feb 12 22:19 ...
drwxrwxr-x 2 s264 s264  4096 Feb 12 22:19 dir1
drwxrwxr-x 2 s264 s264  4096 Feb 12 22:19 dir2
drwxrwxr-x 2 s264 s264  4096 Feb 12 22:19 dir3
[s264@os5264:~/...$ chmod 700 ...
[s264@os5264:~/...$ ./...
ZGc9R8ma5q
s264@os5264:~/...$
```