

Ukesoppgaver 12 – Linux kildekode, nice, fork, effektivitet, Docker

RØD - Obligoppgaver

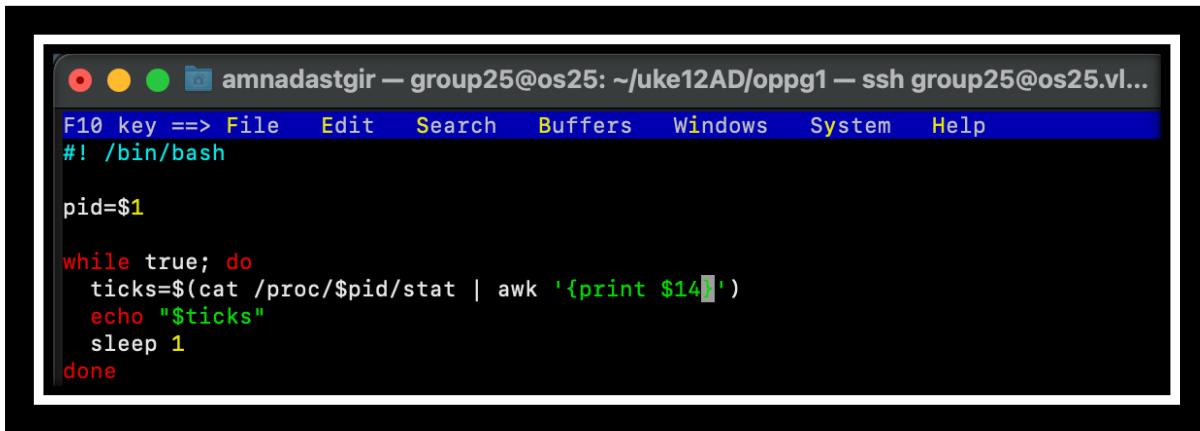
GUL – Ikke obligoppgaver

TURKIS – Ukens nøtt og utfordringer

1. (Ikke oblig)

Kommandoen ps -ef gir liste over alle kjørende prosesser med deres prosess ID-er.

Under vil skriptet skrive ut antall ticks som prosessen med pid bruker i user-mode, og så sove i ett sekund før den gjentar seg. I skriptet vil -filen hente informasjon om prosessen med den gitte PID.-en også plukke ut antall ticks prosessen bruker i user-mode fra den 14-kolonnen ved hjelp av awk.



The screenshot shows a terminal window titled "amnadastgir — group25@os25: ~/uke12AD/oppg1 — ssh group25@os25.vl...". The window has a dark background and a blue header bar with icons. The terminal content is a bash script:

```
#! /bin/bash

pid=$1

while true; do
    ticks=$(cat /proc/$pid/stat | awk '{print $14}')
    echo "$ticks"
    sleep 1
done
```

2. (Oblig)

3. (Ikke oblig)

4. (Oblig)

Under er det en regnejobb som kjører to regneoppgaver som tar litt lengre tid. Har lagt til en ekstra divisjonsoppgave i tillegg til multiplikasjonen sånn at jeg tydelig ser forskjell i CPU når jeg bruker nice.



```
amnadarstgir — group25@os25: ~/uke12AD/oppg4 — ssh group25@os25.vl...
F10 key ==> File Edit Search Buffers Windows System Help
#!/bin/bash

for i in {1..1000000000}; do
    resultat=$(( $i * $i ))
done

for j in {1..1000000000}; do
    resultat=$(( $j / 2 ))
done

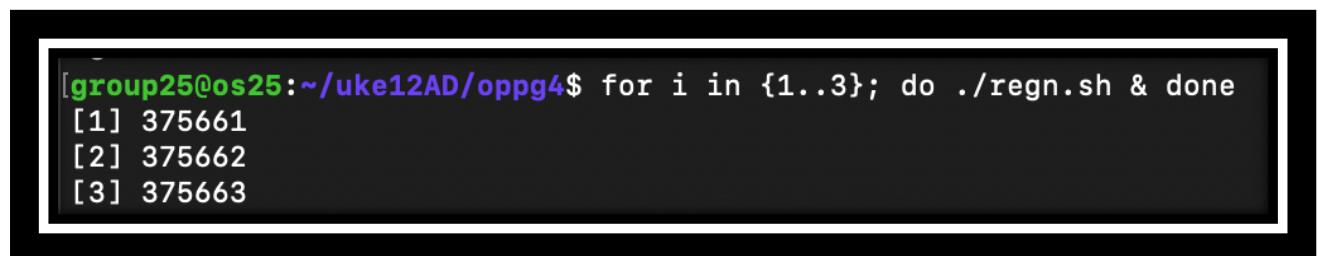
echo "Resultat: $resultat"
```

Under endres rettigheter til regn fila



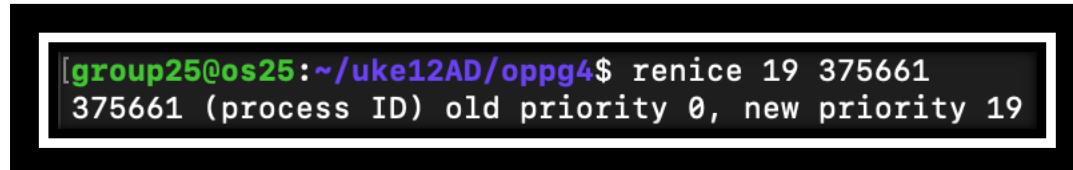
```
[group25@os25:~/uke12AD/oppg4$ chmod 700 regn.sh
```

Lager en løkke som starter tre regne-oppgaver



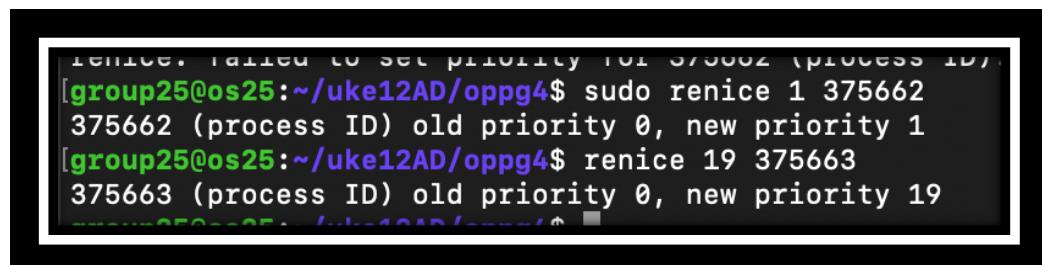
```
[group25@os25:~/uke12AD/oppg4$ for i in {1..3}; do ./regn.sh & done
[1] 375661
[2] 375662
[3] 375663
```

Endrer nice verdien til den første (ender med 61)



```
[group25@os25:~/uke12AD/oppg4$ renice 19 375661
375661 (process ID) old priority 0, new priority 19
```

Endrer nice verdien til andre (ender med 62) og tredje (ender med 63)



```
[renice: failed to set priority for 375662 (process ID)
[group25@os25:~/uke12AD/oppg4$ sudo renice 1 375662
375662 (process ID) old priority 0, new priority 1
[group25@os25:~/uke12AD/oppg4$ renice 19 375663
375663 (process ID) old priority 0, new priority 19
```

Det kan være flere faktorer som påvirker hvor mye CPU en prosess bruker, inkludert mengden begrensningsintensivt arbeid som må utføres, og hvor mange ressurser andre prosesser på systemet tar opp. Når du kjører flere prosesser samtidig vil hver prosess konkurrere om tilgjengelige CPU-ressurser, og hvis mange kjører samtidig, vil hver prosess få en del av de ressursene.

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND	P
375661	group25	39	19	65.1g	65.1g	2704	R	67.1	8.6	1:38.64	regn.sh	94
375662	group25	21	1	65.3g	65.3g	2816	R	67.1	8.6	1:38.58	regn.sh	20
375663	group25	39	19	65.1g	65.1g	2780	R	66.4	8.6	1:38.59	regn.sh	12

5. (Oblig)

Starter to prosesser på CPU 0 med taskset

[group25@os25:~/uke12AD/oppg4\$ taskset -c 0 ./regn.sh&
[1] 377472
[group25@os25:~/uke12AD/oppg4\$ taskset -c 0 ./regn.sh&
[2] 377473

Under kan man se CPU-bruk når begge kjører på samme CPU:

top - 18:52:07 up 38 days, 9 min, 1 user, load average: 2.93, 10.21, 479.54
Tasks: 308 total, 3 running, 305 sleeping, 0 stopped, 0 zombie
%Cpu(s): 2.0 us, 1.8 sy, 0.0 ni, 96.3 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 773178.5 total, 712666.3 free, 44855.2 used, 15657.1 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 723662.5 avail Mem
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND P
377472 group25 20 0 11.6g 5.1g 2696 R 40.1 0.7 0:10.73 regn.sh 0
377473 group25 20 0 11.0g 4.4g 2672 R 39.7 0.6 0:09.34 regn.sh 0

6. UKENS UTFORDRING 1!

Kjører to kommandoer med

sudo taskset -c 0 ./regn & sudo taskset -c 0 ./regn2

her startet jeg to jobber med samme regn oppgave som fra oppgaven over og brukte taskset til å begrense dem til å kjøre på CPU-kjerne 0. for å gi den ene prosessen så stor nive verdi som mulig brukte jeg renice med prosessID også nice-verdien. Jo høyere nice-verdi, jo lavere prioritet får prosessen, og jo mer CPU-tid vil den gi fra seg til andre prosesser.

sudo renice -n 10 -p regn1

dette øker nice-verdien for jobb1 til 10, noe som gjør at den vil gi fra seg mer CPU-tid til andre prosesser

for å gi den andre prosessen en negativ nice-verdi, kan vi kjøre følgende kommando:

sudo renice -n 10 -p regn2

dette reduserte nice-verdien for regn2 til -10, som gjør at den tar mer CPU-tid fra andre prosesser.

Hvis jeg får en permission denied melding når jeg prøver å endre nice-meldingen til en prosess, kan jeg løse dette ved å kjøre kommandoen med «sudo» foran. Dette gir tilstrekkelige rettigheter til å endre nice-verdien.

Hvor stor andel av CPU-tiden som går til hver prosess, avhenger av faktorer som jobbenes størrelse og kompleksitet, og hvor mye CPU-tid de andre prosessene på systemet krever. Men siden vi har begrenset begge jobbene til å kjøre på samme CPU-kjerne, vil de i utgangspunktet dele på CPU-tid en tilgjengelig på den kjernen. Hvis vi øker nice-verdien for den ene prosessen, vil den gi fra seg meg CPU-tid til den andre prosessen og andre prosesser på systemet. Omvendt vil en negativ nice-verdi gjøre at prosessen tar mer CPU-tid fra andre prosesser.

7. UKENS UTFORDRING 2!

FRA NETT:

Linux-kjernen bruker en prioriteringsalgoritme kalt Completely Fair Scheduler (CFS) til å fordele CPU-tid til prosesser. CFS-algoritmen er designet for å fordele CPU-tid basert på en prosess vekting, hvor prosesser med høyere vekting får tildelt mer CPU-tid enn prosesser med lavere vekting.

Nice-verdien påvirker vektingen til en prosess i CFS-algoritmen. Jo høyere nice-verdi, jo lavere vekting får prosessen, og jo mindre CPU-tid vil den få tildelt av kjernen. Omvendt vil en negativ nice-verdi øke vektingen til en prosess, og gi den mer CPU-tid.

8. (Oblig)

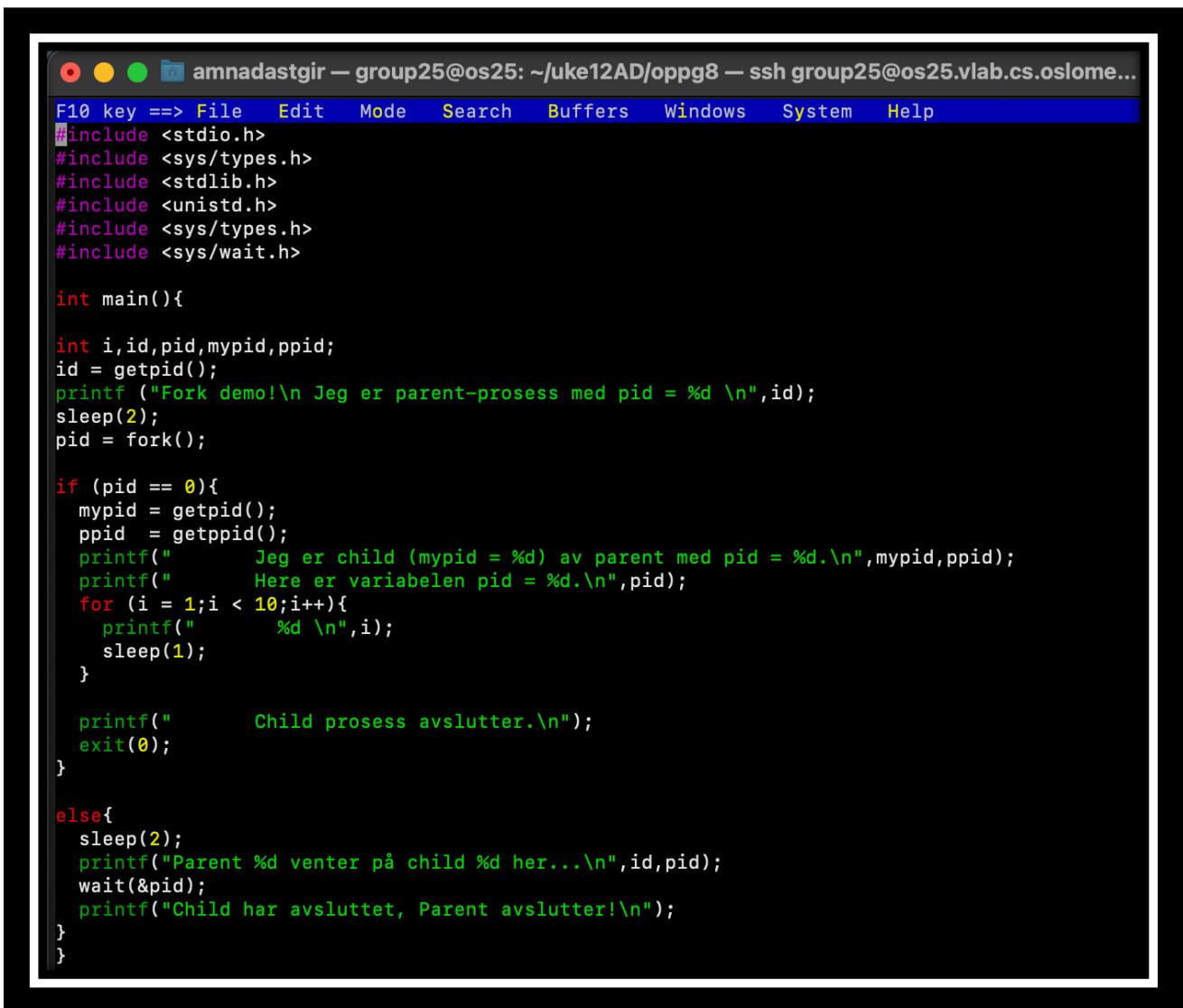
Under er det et C-program som demonstrerer hvordan man kan bruke fork() funksjonen i Linux til å lage en ny prosess fra en eksisterende prosess. Programmet starter med å skrive ut en beskjed til skjermen, og sover deretter i to sekunder med sleep-funksjonen.

Også bruker programmet fork() til å lage en ny prosess. Hvis fork() returnerer 0, betyr det at programmet nå kjører i barneprosessen. Hvis fork() funksjonen returnerer et positivt tall, betyr

det at programmet fortsatt kjører i foreldreprosessen, og tallet som returneres er barneprosessens PID.

Hvis programmet kjører i barneprosessen, skriver det ut noen meldinger til skjermen og teller fra 1 til 9 ved hjelp av en for-løkke. Også avsluttes barneprosessen med exit.

Hvis programmet fortsatt kjører i foreldreprosessen, sover det igjen i to sekunder og skriver ut en melding til skjermen som indikerer at den venter på at barneprosessen skal avslutte. Også bruker programmet wait() funksjonen til å vente på at barneprosessen skal avslutte. Når barneprosessen er ferdig, skriver programmet ut en siste melding til skjermen, og avslutter.



The screenshot shows a terminal window titled "amnadastgir — group25@os25: ~/uke12AD/oppg8 — ssh group25@os25.vlab.cs.oslome...". The window contains a C program source code:

```
#include <stdio.h>
#include <sys/types.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main(){

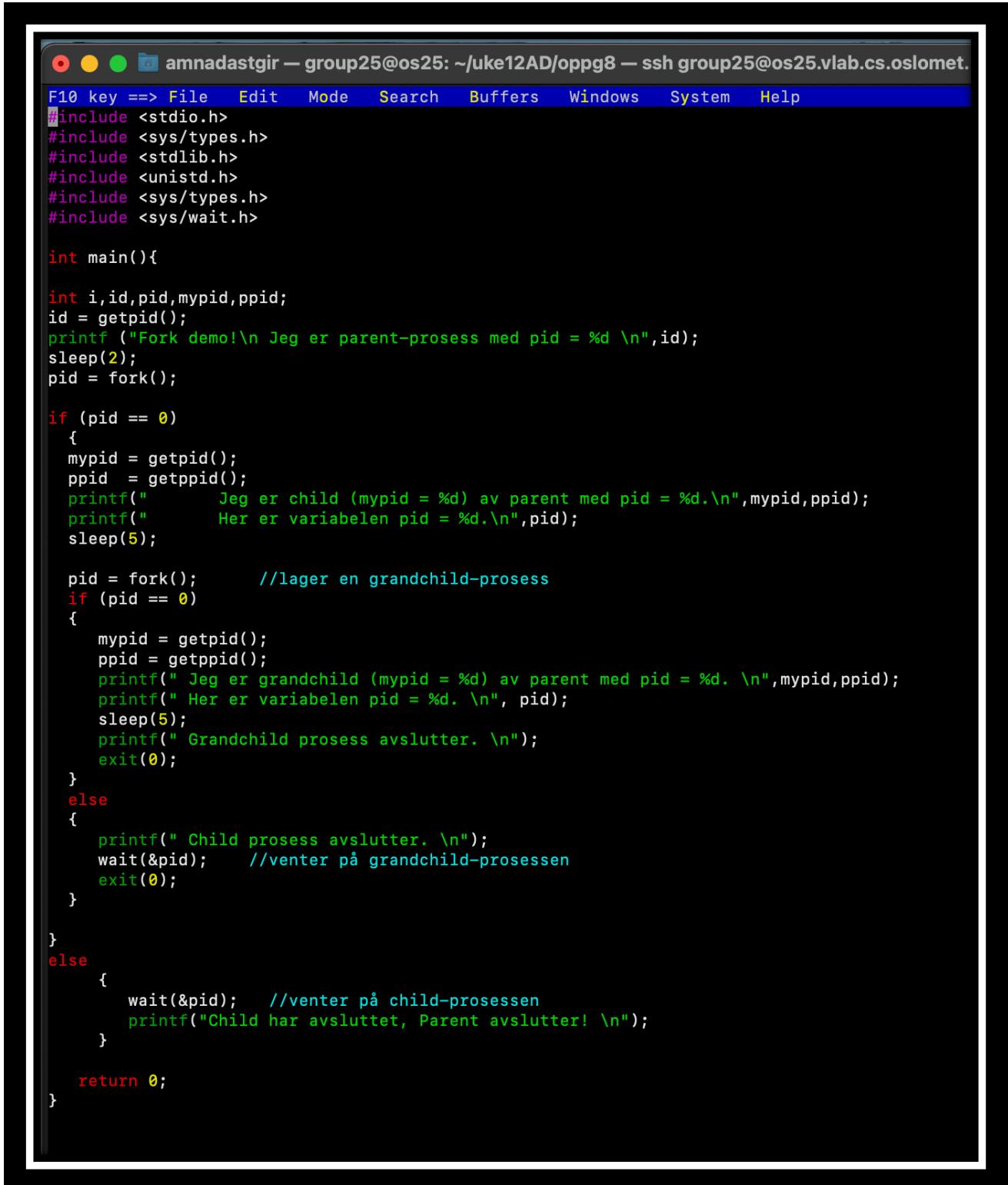
    int i,id,pid,mypid,ppid;
    id = getpid();
    printf ("Fork demo!\n Jeg er parent-prosess med pid = %d \n",id);
    sleep(2);
    pid = fork();

    if (pid == 0){
        mypid = getpid();
        ppid = getppid();
        printf("      Jeg er child (mypid = %d) av parent med pid = %d.\n",mypid,ppid);
        printf("      Here er variablen pid = %d.\n",pid);
        for (i = 1;i < 10;i++){
            printf("      %d \n",i);
            sleep(1);
        }

        printf("      Child prosess avslutter.\n");
        exit(0);
    }

    else{
        sleep(2);
        printf("Parent %d venter på child %d her...\n",id,pid);
        wait(&pid);
        printf("Child har avsluttet, Parent avslutter!\n");
    }
}
```

Her er min modifisering:



The screenshot shows a terminal window titled "amnadastgir — group25@os25: ~/uke12AD/oppg8 — ssh group25@os25.vlab.cs.oslomet.no". The window contains the following C code:

```
#include <stdio.h>
#include <sys/types.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main(){

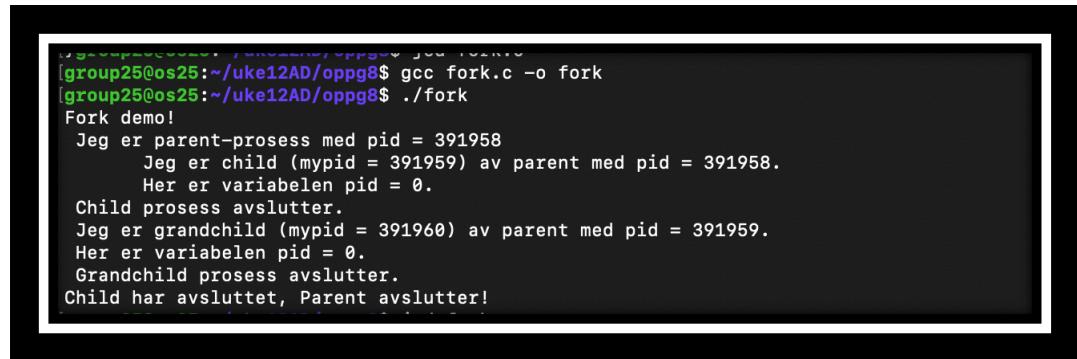
    int i,id,pid,mypid,ppid;
    id = getpid();
    printf ("Fork demo!\n Jeg er parent-prosess med pid = %d \n",id);
    sleep(2);
    pid = fork();

    if (pid == 0)
    {
        mypid = getpid();
        ppid = getppid();
        printf("      Jeg er child (mypid = %d) av parent med pid = %d.\n",mypid,ppid);
        printf("      Her er variabelen pid = %d.\n",pid);
        sleep(5);

        pid = fork();      //lager en grandchild-prosess
        if (pid == 0)
        {
            mypid = getpid();
            ppid = getppid();
            printf(" Jeg er grandchild (mypid = %d) av parent med pid = %d. \n",mypid,ppid);
            printf(" Her er variabelen pid = %d. \n", pid);
            sleep(5);
            printf(" Grandchild prosess avslutter. \n");
            exit(0);
        }
        else
        {
            printf(" Child prosess avslutter. \n");
            wait(&pid);    //venter på grandchild-prosessen
            exit(0);
        }
    }
    else
    {
        wait(&pid);    //venter på child-prosessen
        printf("Child har avsluttet, Parent avslutter! \n");
    }

    return 0;
}
```

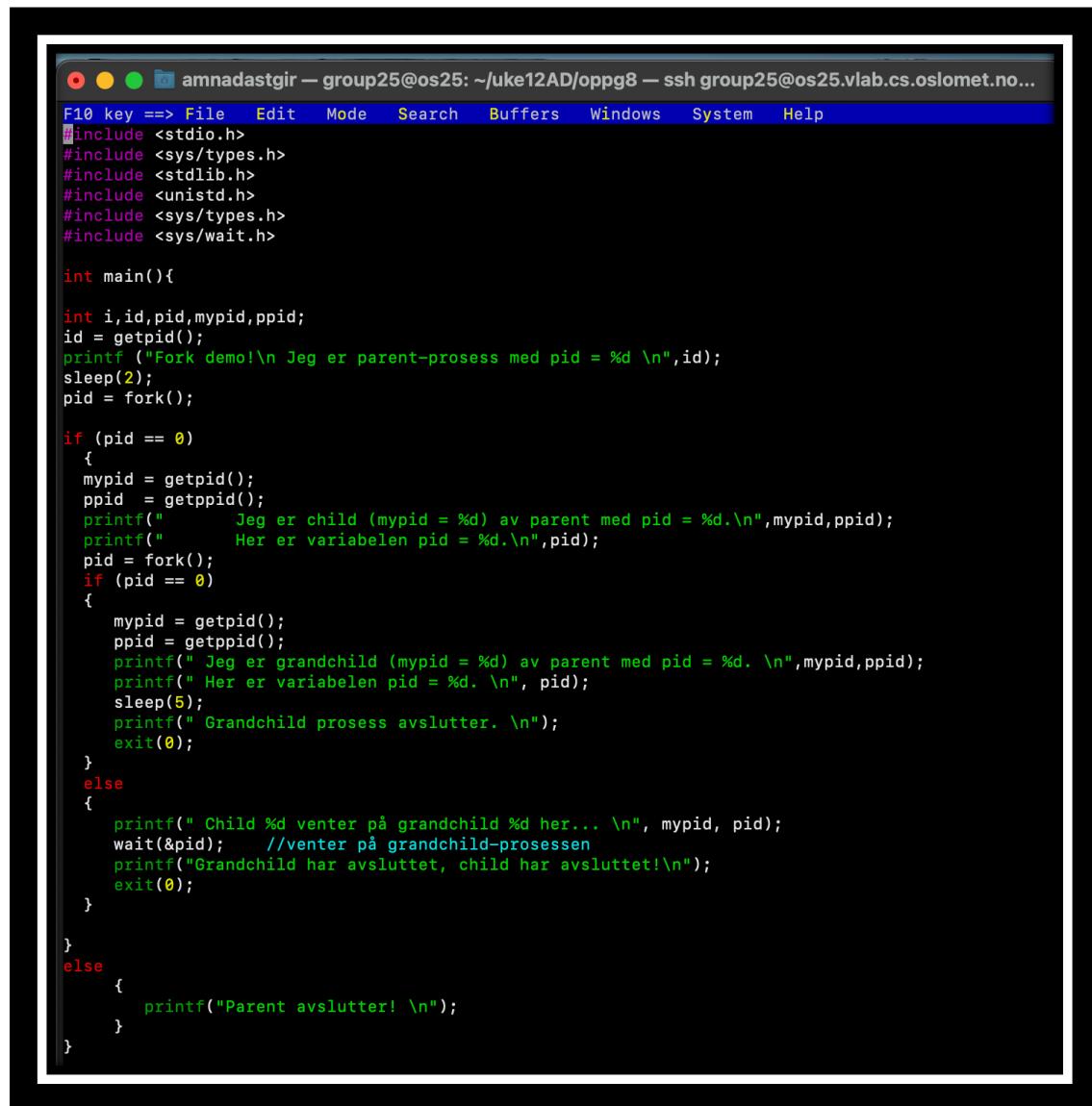
Under kompileres koden og kjøres:



```
[group25@os25:~/uke12AD/oppg8$ gcc fork.c -o fork
[group25@os25:~/uke12AD/oppg8$ ./fork
Fork demo!
Jeg er parent-prosess med pid = 391958
    Jeg er child (mypid = 391959) av parent med pid = 391958.
    Her er variablene pid = 0.
    Grandchild prosess avslutter.
Child har avsluttet, Parent avslutter!
```

9. (Ikke oblig)

Tror dette er riktig



```
amnadarstgir — group25@os25: ~/uke12AD/oppg8 — ssh group25@os25.vlab.cs.oslomet.no...
F10 key ==> File Edit Mode Search Buffers Windows System Help
#include <stdio.h>
#include <sys/types.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main(){

int i,id,pid,mypid,ppid;
id = getpid();
printf ("Fork demo!\n Jeg er parent-prosess med pid = %d \n",id);
sleep(2);
pid = fork();

if (pid == 0)
{
    mypid = getpid();
    ppid = getppid();
    printf("    Jeg er child (mypid = %d) av parent med pid = %d.\n",mypid,ppid);
    printf("    Her er variablene pid = %d.\n",pid);
    pid = fork();
    if (pid == 0)
    {
        mypid = getpid();
        ppid = getppid();
        printf("    Jeg er grandchild (mypid = %d) av parent med pid = %d. \n",mypid,ppid);
        printf("    Her er variablene pid = %d. \n", pid);
        sleep(5);
        printf("    Grandchild prosess avslutter. \n");
        exit(0);
    }
    else
    {
        printf("    Child %d venter på grandchild %d her... \n", mypid, pid);
        wait(&pid); //venter på grandchild-prosessen
        printf("Grandchild har avsluttet, child har avsluttet!\n");
        exit(0);
    }
}
else
{
    printf("    Child %d venter på grandchild %d her... \n", mypid, pid);
    wait(&pid); //venter på grandchild-prosessen
    printf("Grandchild har avsluttet, child har avsluttet!\n");
    exit(0);
}
```

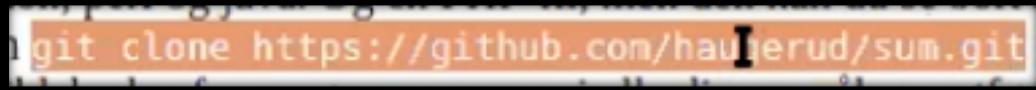
10. (Oblig)

Man trenger først verktøyene:

```
[root@os25:/home/group25/uke12AD/programmer# apt-get install build-essential
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.8ubuntu1.1).
0 upgraded, 0 newly installed, 0 to remove and 39 not upgraded.
[root@os25:/home/group25/uke12AD/programmer# apt-get install default-jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done
default-jdk is already the newest version (2:1.11-72).
0 upgraded, 0 newly installed, 0 to remove and 39 not upgraded.
[root@os25:/home/group25/uke12AD/programmer# apt-get install python3
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3 is already the newest version (3.8.2-0ubuntu2).
0 upgraded, 0 newly installed, 0 to remove and 39 not upgraded.
```

- Build-essential er for gcc, for å kompilere C
- Jdk er for Java
- Python for Python

Programmet som skal brukes ligger på GitHub hvor et repository heter sum.git:



Kloner fra Git

```
[root@os25:/home/group25/uke12AD/programmer# git clone https://github.com/haugerud/sum.git
Cloning into 'sum'...
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 15 (delta 4), reused 12 (delta 4), pack-reused 0
Unpacking objects: 100% (15/15), 2.36 KiB | 807.00 KiB/s, done.
```

Sum.bash skriptet regner ut en sum 50k ganger og de andre programmene regner ut denne arbeidsoppgaven x ganger, og vi skal finne ut hvem som klarer flest sånne oppgaver på like lang tid. Dette må vi få til å gjøre INNE i dockercontaineren.

Jeg har filene under:

```
[root@os25:/home/group25/uke12AD/oppg10/programmer/sum# ls -l
total 44
-rw-r--r-- 1 root root 357 Mar 28 01:15 Dockerfile
-rw-r--r-- 1 root root 317 Mar 28 01:10 Dockerfile~
-rw-r--r-- 1 root root 35 Mar 28 00:49 README.md
-rwxr-xr-x 1 root root 416 Mar 28 00:49 Sum.java
-rw-r--r-- 1 root root 194 Mar 28 01:13 komp.sh
-rwxr-xr-x 1 root root 210 Mar 28 00:49 sum.bash
-rwxr-xr-x 1 root root 355 Mar 28 00:49 sum.c
-rwxr-xr-x 1 root root 276 Mar 28 00:49 sum.perl
-rwxr-xr-x 1 root root 279 Mar 28 00:49 sum.php
-rwxr-xr-x 1 root root 206 Mar 28 00:49 sum.py
-rwxr-xr-x 1 root root 420 Mar 28 00:49 sum0.c
```

Her er dockerfila mi:

```
F10 key ==> File Edit Search Buffers Windows System Help
FROM ubuntu
RUN apt-get -y update
RUN apt-get -y install apache2
RUN apt-get -y install build-essential
RUN apt-get -y install default-jdk
RUN apt-get -y install python3
RUN apt-get -y install git
RUN apt-get -y install time
RUN git clone https://github.com/haugerud/sum.git
COPY komp.sh .
RUN chmod 755 komp.sh
CMD ["/usr/sbin/apachectl","-D","FOREGROUND"]
```

Bygger dockeren min:

```
[root@os25:/home/group25/uke12AD/oppg10/programmer/sum# docker build --no-cache -t sum .
[+] Building 62.9s (16/16) FINISHED
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 396B 0.0s
=> [internal] load metadata for docker.io/library/ubuntu:latest 1.2s
=> [internal] load build context 0.0s
=> => transferring context: 230B 0.0s
=> CACHED [ 1/11] FROM docker.io/library/ubuntu@sha256:67211c14fa74f070d27cc59d69a7fa9 0.0s
=> [ 2/11] RUN apt-get -y update 4.2s
=> [ 3/11] RUN apt-get -y install apache2 7.8s
=> [ 4/11] RUN apt-get -y install build-essential 9.7s
=> [ 5/11] RUN apt-get -y install default-jdk 21.7s
=> [ 6/11] RUN apt-get -y install python3 1.6s
=> [ 7/11] RUN apt-get -y install git 2.5s
=> [ 8/11] RUN apt-get -y install time 2.1s
=> [ 9/11] RUN git clone https://github.com/haugerud/sum.git 1.3s
=> [10/11] COPY komp.sh . 0.0s
=> [11/11] RUN chmod 755 komp.sh 0.6s
=> => exporting to image 10.0s
=> => exporting layers 10.0s
=> => writing image sha256:d3461b9c8c4563085632cad7d94c2277f0fa3db9ad7ccf06ffd570b9dc5 0.0s
=> => naming to docker.io/library/sum 0.0s
```

Viser under at jeg kan kjøre kommandoene ./sum.perl og python3 sum.py:

```
[root@os25:/home/group25/uke12AD/oppg10/programmer/sum# ./sum.perl
Ferdig.
[root@os25:/home/group25/uke12AD/oppg10/programmer/sum# python3 sum.py
Ferdig! Sum: 250000000000
```

Javac fungerer også

```
[root@os25:/home/group25/uke12AD/oppg10/programmer/sum# javac Sum.java
[root@os25:/home/group25/uke12AD/oppg10/programmer/sum# java Sum
Ferdig 250000000000
```

11. (Oblig)

Ta tiden på programmene fra containeren.

Skriptet sum.bash regner en sum 500k ganger:

```
[group25@os25:~/uke12AD/oppg10/programmer/sum$ time ./sum.bash
Ferdig, sum: 250000000000
Real:4.087 User:4.087 System:0.000 99.99%
```

Alle andre programmer regner ut denne summen TIMES ganger. Resultatet jeg skal se på er user + sys som vanligvis er det samme som real.

Hvis real tiden er større enn user + sys betyr det at andre programmer bruker mye CPU.

Jeg skal nå bruke kommandoen time på de andre programmene og lage en tabell under som vil demonstrere hvor mye raskere de andre programmene er enn sum.bash. Hvis de andre programmene ikke bruker tiden jeg fikk for sum.bash (4 sekunder), må jeg endre størrelsen på TIMES.

PHP:

```
[group25@os25:~/uke12AD/oppg10/programmer/sum$ cat sum.php
#!/usr/bin/php

<?php
[
    $TIMES = 220;
    $maxC = 500000;
    $sum = 0;
    for ($j = 0;$j < $TIMES;$j++)
    {
        $sum = 0;
        for ($i = 0;$i < $maxC;$i++)
        {
            $tall = ($i + 1)*($i + 1) - $i*$i;
            $sum = $sum + $tall;
        }
    }

    print "Ferdig. Sum $sum\n";
?
group25@os25:~/uke12AD/oppg10/programmer/sum$ time ./sum.php
Ferdig. Sum 25000000000
Real:4.028 User:4.020 System:0.008 99.99%
```

PERL:

```
[group25@os25:~/uke12AD/oppg10/programmer/sum$ cat sum.perl
#!/usr/bin/perl

# $ time sum.perl
$TIMES = 41.6;

$maxC = 500000;
$sum = 0;
for ($j = 0;$j < $TIMES;$j++)
{
    $sum = 0;
    for ($i = 0;$i < $maxC;$i++)
    {
        $tall = ($i + 1)*($i + 1) - $i*$i;
        $sum = $sum + $tall;
    }
}
print "Ferdig.\n";
group25@os25:~/uke12AD/oppg10/programmer/sum$ time ./sum.perl
Ferdig.
Real:4.021 User:4.021 System:0.000 99.99%
```

JAVA:

```
[group25@os25:~/uke12AD/oppg10/programmer/sum$ cat Sum.java
class Sum {
    // Unix: $ javac Sum.java
    //           $ time java Sum
    public static void main(String args[])
    {
        int TIMES = 16500;

        int maxC = 500000;
        int i,j;
        long sum = 0;
        int tall;
        for(j=0;j < TIMES;j++)
        {
            sum = 0;
            for(i=0;i < maxC;i++)
            {
                tall = (i + 1)*(i + 1) - i*i;
                sum = sum + tall;
            }
        }
        System.out.println("Ferdig " + sum);
    }
}

[group25@os25:~/uke12AD/oppg10/programmer/sum$ time java Sum
Ferdig 250000000000
Real:4.096 User:4.072 System:0.068 101.08%
```

C-KODE:

```
[group25@os25:~/uke12AD/oppg10/programmer/sum$ cat sum.c
#include <stdio.h>

// $ cc sum.c
// $ time a.out
int main()
{
    int TIMES = 2750;

    int maxC = 500000;
    int i,j;
    long long int sum = 0;
    int tall;

    for(j=0;j < TIMES;j++)
    {
        sum = 0;
        for(i=0;i < maxC;i++)
        {
            tall = (i + 1)*(i + 1) - i*i;
            sum = sum + tall;
        }
    }
    printf("Ferdig %llu\n",sum);
}

[group25@os25:~/uke12AD/oppg10/programmer/sum$ sudo gcc sum.c -o sum
[group25@os25:~/uke12AD/oppg10/programmer/sum$ time ./sum
Ferdig 250000000000
Real:4.046 User:4.046 System:0.000 99.99%
```

OPTIMALISERT C-KODE:

```
[group25@os25:~/uke12AD/oppg10/programmer/sum$ cat sum0.c
#include <stdio.h>

// $ cc -O sum0.c # Nesten 7 ganger raskere enn om -O (optimalisering) ikke er med
// $ time a.out
int main()
{
    int TIMES = 2770;

    int maxC = 500000;
    int i,j;
    long long sum = 0;
    int tall;

    for(j=0;j < TIMES;j++)
    {
        sum = 0;
        for(i=0;i < maxC;i++)
        {
            tall = (i + 1)*(i + 1) - i*i;
            sum = sum + tall;
        }
    }
    printf("Ferdig %llu\n",sum);
}
[group25@os25:~/uke12AD/oppg10/programmer/sum$ sudo gcc sum0.c -o sum
[group25@os25:~/uke12AD/oppg10/programmer/sum$ time ./sum
Ferdig 250000000000
Real:4.078 User:4.078 System:0.000 99.99%
```

PYTHON:

```
[group25@os25:~/uke12AD/oppg10/programmer/sum$ cat sum.py
#!/usr/bin/python

TIMES = 30
maxC = 500000

for j in range (0,TIMES):
    sum = 0
    for i in range (0,maxC):
        tall = (i + 1)*(i + 1) - i*i
        sum = sum + tall
    print("Ferdig! Sum: %d" % sum)
[group25@os25:~/uke12AD/oppg10/programmer/sum$ time python3 sum.py
Ferdig! Sum: 250000000000
Real:4.158 User:4.150 System:0.008 99.99%
```

12. UKENS STØRSTE UTFORDRING!

13. (Ikke oblig)

14. EFFICODE

```
[s264@os5264:~]$ docker ps -a
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock"
/v1.24/containers/json?all=1": dial unix /var/run/docker.sock: connect: permission denied
[s264@os5264:~]$ sudo su
[root@os5264:/home/s264# docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
bdd110db902b public.ecr.aws/lts/ubuntu:latest "tail -f /dev/null" 8 days ago Exited (137) 8 days ago uke12
3da1f0c1b6b6 public.ecr.aws/lts/ubuntu:latest "tail -f /dev/null" 13 days ago Up 13 days suspicious_elgamal
31ba6dc3a26f public.ecr.aws/lts/ubuntu:latest "tail -f /dev/null" 2 weeks ago Exited (255) 13 days ago tender_hugle
[root@os5264:/home/s264# docker exec -it 3da1f0c1b6b6 /b
OCI runtime exec failed: exec failed: unable to start container process: exec: "/b": stat /b: no such file or directory: unknown
[root@os5264:/home/s264# docker start bdd110db902b
bdd110db902b
[root@os5264:/home/s264# docker exec -it bdd110db902b /bin/bash
root@bdd110db902b:# cat /etc/shadow
root::*19424:0:99999:7:::
daemon::*19424:0:99999:7:::
bin::*19424:0:99999:7:::
sys::*19424:0:99999:7:::
sync::*19424:0:99999:7:::
games::*19424:0:99999:7:::
man::*19424:0:99999:7:::
lp::*19424:0:99999:7:::
mail::*19424:0:99999:7:::
news::*19424:0:99999:7:::
uucp::*19424:0:99999:7:::
proxy::*19424:0:99999:7:::
www-data::*19424:0:99999:7:::
backup::*19424:0:99999:7:::
list::*19424:0:99999:7:::
irc::*19424:0:99999:7:::
gnats::*19424:0:99999:7:::
nobody::*19424:0:99999:7:::
_apt::*19424:0:99999:7:::
os:9L27y7BA63:19435:0:99999:7:::
root@bdd110db902b:# ]
```