

Seksjon 1

To kjerner

Forklar kort forskjellen på en operativsystem-kjerne og en CPU-kjerne. Hvilken av dem er hardware og hvilken er software? Forklar kort.

B <i>I</i> <u>U</u> ☰ ☷ á
0 / 10000 Word Limit

[Her er et Cheat-Sheet vedlagt som gjelder for hele oppgavesettet . Hvis du trenger å se på det for senere oppgaver, husk at du må tilbake til første oppgave for å få tak i det.]

[cheat.pdf](#)

Seksjon 2

Docker

Du har akkurat startet docker-tjenesten på en Linux-VM og vil teste at alt fungerer ved å kjøre den klassiske **hello-world** -containeren. Hvilken kommando er riktig?

- ☐ docker start hello-world
- ☐ docker container run hello-world
- ☐ docker pull hello-world --run
- ☐ service hello-world start

Seksjon 3

Docker

Du startet tidligere en ubuntu-container med apache2 installert, og forlot den med `CTRL-p CTRL-q` slik at den fortsatte å kjøre i bakgrunnen. Senere har du stoppet containeren. Du kjører `docker container ps -a` og ser at containeren har ID `3376df05f434` og status `Exited`. Du vil starte den samme containeren igjen, med apache fortsatt installert, uten å lage en ny container. Hvilken kommando bruker du?

- ☐ `docker container run 3376df05f434`
- ☐ `docker container restart ubuntu`
- ☐ `docker container start 3376df05f434`
- ☐ `docker container commit 3376df05f434`

Seksjon 4

Docker

Hva er hovedforskjellen mellom en `Dockerfile` og en `docker-compose.yaml` ?

- ☐ En `Dockerfile` beskriver hvordan et image bygges, mens en `docker-compose.yaml` beskriver hvordan en eller flere containere skal kjøres.
- ☐ En `Dockerfile` definerer hvilke porter og volumer en container skal bruke når den starter, mens en `docker-compose.yaml` definerer hvilken programvare som skal installeres i imaget.
- ☐ De gjør nøyaktig det samme, men `docker-compose.yaml` er den nyere versjonen som har erstattet `Dockerfile`.
- ☐ En `Dockerfile` kan bare definere én container, mens `docker-compose.yaml` må alltid definere minst to.

Seksjon 5

Innlogging på Linux-VM

Øverst i browser-vinduet ved siden av WISEflow-fanen er det en fane der det står "Apache Guacamole". Når du trykker på denne får du opp et innloggingsvindu der du skal logge inn med

Username: kan26

Password: kanelbollevaffel26

og du vil etter 5-10 sekunder få opp et terminal-vindu til en VM som er av samme type som VM-ene som har blitt brukt i kurset, bortsett fra at denne ikke har nett-tilgang. Og det kan så ta noen sekunder før du får opp et bash-prompt.

Hvis denne fanen av en eller annen grunn forsvinner, skriv inn data2500.s2.cs.oslomet.no i en ny fane.

Du kommer inn i et bash-shell som brukeren "kan26" i dennes hjemmemappe /home/kan26 på din egen Linux-VM som man som en del av eksamen skal bruke for å løse oppgaver. Denne VMen er nesten helt lik Linux-VMene som dere brukte under kurset, med navn som os13, de er også sysbox Docker-containere, men de har ikke nett-tilgang.

Om man ønsker det, kan Linux-VM under eksamen også brukes til å skrive shell-script og teste ut kommandoer. Men vær oppmerksom på at om man lagrer en fil i Linux-VM og så refresher eller kobler seg til Linux-VM på nytt, vil en ny container-VM startes og filen man har lagret vil være borte; filsystemet vil se slik ut som første gang du koblet deg til.

Det er viktig at du bare har en Linux-VM oppe av gangen, ellers blir ressursene på serverene oppbrukt! Ikke start flere samtidige VMer!

Du kan kopiere tekst i Linux-VM ved å markere den med musen og så taste Shift-Ctrl-Alt. Da kommer det opp et clipboard med den avmerkede teksten og så kan du kopiere med Ctrl-C og paste ut her i Inspira-vinduet med Ctrl-V. Internt inne i Linux-VM kan man kopiere tekst ved å markere den med musen og så paste ved å høyreklikke (som i PowerShell).

Filen **file1** ligger øverst i hjemme-mappen til brukeren du er logget inn som og som har brukernavn "kan26". Fyll inn innholdet av filen (5 tegn) i boksen under ved å skrive eller kopiere og paste.

Fyll inn:

Seksjon 6

Finne fil

Øverst i hjemme-mappen til brukeren du er logget inn som på Linux-VM og som har brukernavn **kan26** ligger det en mappe som heter **mapper26**. Et sted ned i denne mappen eller nedover i dens undermapper ligger det en fil som heter **deep.txt**. Finn denne filen og skriv eller paste ut tekststrengen den inneholder her (5 tegn):

Seksjon 7

Finne fil som alle har alle rettigheter til

Øverst i hjemme-mappen til brukeren du er logget inn som på Linux-VM og som har brukernavn **kan26** ligger det en mappe som heter **mapper26**. Et sted ned i denne mappen eller nedover i dens undermapper ligger det en fil som har rettigheter 777, det vil si at alle brukere har alle rettigheter. Finn denne filen og skriv eller paste ut tekststrengen den inneholder her (5 tegn):

Seksjon 8

Konfigurasjonsfil

På Linux-VM som du kan logge inn på, finnes det en konfigurasjonsfil for bash-shell hvor brukeren root kan sette aliaser, miljøvariabler og andre shell-innstillinger. Nederst i denne filen ligger den 5 tegns koden du skal finne. Finn denne koden og skriv eller paste ut tekststrengen den inneholder her (5 tegn):

Nedenfor er 10 Linux-kommandoer listet opp. Koble dem til riktig betydning.

cp		
ls		
mv		
ln		
chmod		
rm		
cat		
mkdir		
pwd		
ps		

⚙ Kopier

⚙ Lag mappe

⚙ Lag link

⚙ Flytt

⚙ Endre rettigheter

⚙ List innhold i mappe

⚙ Slett/fjern

⚙ Skriv til stdout

⚙ Vis mappen du står i

⚙ List prosesser

Seksjon 10

Vurdere ulike svar fra LLMer

LLMer har de siste årene blitt bedre og bedre til å løse alle slags problemer. Men det er viktig å kunne vurdere hvor riktige svarene er. Vi spurte både ChatGPT 5.5 Thinking og Claude Sonnet 4.6 det samme spørsmålet og fikk to litt forskjellige svar.

Følgende spørsmål ble gitt til både ChatGPT og Claude:

Utfør følgende kommandoer på en Ubuntu Linux server som en vanlig bruker:

```
$ cd
$ mkdir fengsel
$ cd fengsel
$ chmod 055 ..
$ chmod 055 .
$ ls
$ cd ..
$ cd
```

Hjelp! Hvordan kommer du deg ut av dette?

```
$ chmod 755 .
$ chmod 755 ..
```

løser vel problemet, eller? Skriv et kort og presist svar.

Svar fra ChatGPT 5.5 Thinking

Ja, **nesten**.

Kjør først:

```
chmod 755 .
```

Det gir deg tilbake tilgang til **fengsel**, slik at **cd ..** fungerer.

Derefter:

```
cd ..
chmod 755 .
```

eller, siden **..** fra inni **fengsel** peker på hjemmekatalogen:

```
chmod 755 ..
```

Men rekkefølgen er viktig: **chmod 755 .** **må komme først**, ellers får du ikke traversert ut av katalogen.

Svar fra Cluade Sonnet 4.6

Nei, det løser ikke problemet — du sitter fortsatt fast.

Når du står i **~/fengsel** og kjører **chmod 755 .**, så endrer du rettighetene på **fengsel** (som du står i), ikke på hjemmekatalogen din. Og **chmod 755 ..** feiler, fordi **..** er hjemmekatalogen, som nå har modus **055** — du mangler skriverettighet på den, og uten skriverettighet kan du ikke endre dens egne rettigheter (selv om du eier

den).

Faktisk: med `055` på hjemmekatalogen mangler du også *eksekveringsrettighet* for eieren (siden `0` gjelder for owner, og du er owner). Det betyr at du ikke engang kan `cd` inn i `fengsel` på nytt etter at du har gått ut, og mange operasjoner på filer under hjemmekatalogen vil feile.

Løsningen er å fikse hjemmekatalogen direkte med absolutt sti, fra hvor som helst:

```
$ chmod 755 ~
```

eller tilsvarende `chmod 755 /home/brukernavn .` `chmod` bryr seg om rettighetene på *foreldrekatalogen* til målet (for å finne det), ikke på målet selv — og siden eieren av hjemmekatalogen er deg, har du lov til å endre dens modus uavhengig av hva modusen tilfeldigvis er akkurat nå.

Vurder begge de to svarene fra ChatGPT og Claude. Forklar kort om noen av dem er korrekte og gi et riktig svar om begge tar feil.

B *I* U | ☰ ☷ á

0 / 10000 Word Limit

Seksjon 11

Hvilken av filene er angitt med absolutt path?

Velg ett alternativ

- ☐ /usr/bin/diff
- ☐ ../bin/fil.txt
- ☐ etc/passwd
- ☐ home/hh/os/mc/bash.txt
- ☐ ./fil.txt
- ☐ mydir

Seksjon 12

Du utfører følgende i et bash-shell:

```
kan26@os:~$ filbane="/etc/passwd"
kan26@os:~$ echo ${filbane:0:1}
```

Hva blir output fra echo-kommandoen?

- ☐ etc
- ☐ passwd
- ☐ /etc/passwd
- ☐ /
- ☐ filbane

Seksjon 13

Shell-scripting

Skriv et shell-script som tar en streng som argument og skriver ut:

1. Om strengen viser til en eksisterende fil på systemet.
2. Hvis filen finnes: om den er angitt med absolutt eller relativ path.

Du trenger ikke å sjekke antall argumenter. Hint: det kan være nyttig å teste ut forrige oppgave.

B *I* U | ≡ ≡ á

0 / 10000 Word Limit

Seksjon 14

Hva er **IKKE** en av fordelene ved virtualisering?

- ☐ Effektivitet
- ☐ Isolasjon
- ☐ Skytjenester
- ☐ Fleksibilitet
- ☐ Ressurssparing

Seksjon 15

Hva slags kode er listet opp nedenfor? Velg alternativene som passer sammen.

ls -l script.sh	—	
#include <stdio.h>	—	
Get-ChildItem	—	
add %rdx, %rbx	—	
getstatic #18	—	

⚡ Java bytecode

⚡ C

⚡ bash

⚡ PowerShell

⚡ Assembly

Assembly

På Linux-VM som du kan logge inn på har du i mappen /home/kan26/as følgende to program:

```
kan26@os:~/as$ cat sumMain.c
#include

extern int sum();

int main(void)
{
    int summ;
    summ = sum();
    printf("Sum = %d \n", summ);
}

kan26@os:~/as$ cat as.s
.globl sum
# C-signatur:int sum ()
# 64 bit assembly

sum:
mov    $3, %rax
mov    $1, %rbx
add    %rax, %rbx
ret

.section .note.GNU-stack,"",@progbits
```

Vis hvordan du kan compilere disse to programmene med gcc og kjøre den resulterende filen. Forklar kort hvorfor du får det resultatet du får.

Hvis du ikke får til å compilere og kjøre programmet, forklar kort hva du tror blir resultatet av kjøringen.

B *I* U | ≡ ≡

0 / 10000 Word Limit

Seksjon 17

Datamaskinarkitektur

På Linux-VM som du kan logge inn på har du i mappen /home/kan26/sum følgende C++ program med navn **sum.cpp**. Programmet genererer et array som inneholder 32768 tilfeldige og usorterte heltall mellom 0 og 255 og summerer ved hjelp av en if-test opp alle tallene som er større enn 127:

```
#include <algorithm>
#include <iostream>

using namespace std;

int main()
{
    // Lager et data-array
    int i,c;
    int arraySize = 32768;
    int data[arraySize];

    for (c = 0; c < arraySize; ++c)
    {
        data[c] = rand() % 256;
    }

    // Gir tilfeldig tall mellom 0 og 255
    // Gir samme array med tall for hver kjøring

    // sort(data, data + arraySize);
    // sorterer data-arrayet

    // Skriver ut de 10 første verdiene
    for (c = 0; c < 10; c++)
        cout << data[c] << "\n";

    // Legger sammen alle tall større enn 127
    long sum = 0;

    // Ytre løkke for at det skal ta litt tid...
    for (i = 0; i < 20000; ++i)
    {
        // Indre løkke
        for (c = 0; c < arraySize; ++c)
        {
            if (data[c] > 127)
                sum += data[c];
        }
    }

    cout << "sum = " << sum << "\n";
}
```

Kompilerer programmet og ta tiden på hvor lang tid det bruker på å kjøre med

```
$ g++ sum.cpp
$ time ./a.out
```

Sørg så for å fjerne de to tegnene // foran følgende linje

```
// sort(data, data + arraySize);
```

slik at arrayet blir sortert.

Kompilerer programmet på nytt og ta tiden på hvor lang tid det nå bruker på å kjøre.

Ta utgangspunkt i det du har lært om datamaskinarkitektur og forklar kort hvorfor det er forskjell i tiden programmene tar, til tross for at de i begge tilfeller utfører nøyaktig like mange instruksjoner.

Skriv ditt svar her

B *I* U |  

0 / 10000 Word Limit

Seksjon 18

Synkronisering

Denne innledende oppgaven og de neste to, handler om forsøk på å synkronisere prosesser.

På Linux-VM har du i mappen /home/kan26/teller1 et bash-script:

```
kan26@os:~/teller1$ cat teller1.sh
#!/bin/bash
teller="/home/kan26/telleFil.txt"
read nr < $teller
((nr++))
sleep 3
echo $nr > $teller
```

Forklar kort hva shell-scriptet gjør og hva som skjer når du kjører scriptet to ganger på denne måten:

```
kan26@os:~/teller1$ echo 0 > /home/kan26/telleFil.txt
kan26@os:~/teller1$ ./teller1.sh
kan26@os:~/teller1$ ./teller1.sh
kan26@os:~/teller1$ cat /home/kan26/telleFil.txt
2
kan26@os:~/teller1$
```

B *I* U |   á

0 / 10000 Word Limit

Seksjon 19

Det samme shell-scriptet som i forrige oppgave kjøres nå på følgende måte, hvor du starter det to ganger i bakgrunnen med ca ett sekunds mellomrom, slik at scriptene delvis kjører samtidig.

```
kan26@os:~/teller1$ echo 0 > /home/kan26/telleFil.txt
kan26@os:~/teller1$ ./teller1.sh&
[1] 144
kan26@os:~/teller1$ ./teller1.sh&
[2] 146
kan26@os:~/teller1$ cat /home/kan26/telleFil.txt
1
[1]-  Done                ./teller1.sh
[2]+  Done                ./teller1.sh
kan26@os:~/teller1$
```

Forklar kort hva som nå skjer og hvorfor scriptet nå ikke teller opp til 2 som i forrige oppgave.

B *I* U | ≡ ≡ á

0 / 10000 Word Limit

Seksjon 20

På Linux-VM har du i mappen /home/kan26/teller2 en annen versjon av teller-scriptet, som er forbedret med tanke på å unngå telle-problemet i forrige oppgave:

```
kan26@os:~/teller2$ cat teller2.sh
#!/bin/bash
teller="/home/kan26/telleFil.txt"
lock="/home/kan26/.lock"

while [ -f $lock ]
do
    : # Kommando som ikke gjør noe
done
touch $lock
read nr < $teller
((nr++))
sleep 3
echo $nr > $teller

/bin/rm $lock
```

Og når du nå kjører dette på samme måte med omtrent ett sekunds mellomrom skjer følgende:

```
kan26@os:~/teller2$ echo 0 > /home/kan26/telleFil.txt
kan26@os:~/teller2$ ./teller2.sh &
[1] 181
kan26@os:~/teller2$ ./teller2.sh &
[2] 184
kan26@os:~/teller2$ cat /home/kan26/telleFil.txt
2
[1]-  Done                  ./teller2.sh
[2]+  Done                  ./teller2.sh
kan26@os:~/teller2$
```

Forklar kort hvordan scriptet er endret og hvorfor det nå teller korrekt opp til 2 når du igjen starter to kjøring i bakgrunnen med ett sekunds mellomrom. (kommandoen : som brukes i scriptet kalles en null command og gjør ingenting). Forklar også kort om denne løsningen alltid vil fungere og i alle tilfeller løse dette synkroniseringsproblemet; hvis ikke, hva er i så fall problemet?

NB! Du behøver ikke å teste ut dette selv i Linux-VMen for å svare på oppgaven. Men hvis du tester dette ut selv, pass på å kjøre scriptet /home/kan26/teller2/kill.sh etterpå for å være sikker på at det ikke blir kjørende noen CPU-slukende prosesser etter at du er ferdig med kjøringene.

B *I* U | ≡ ≡ á

0 / 10000 Word Limit

Seksjon 21

Ranger fra hurtigst til senest:

≡ SSD

≡ L1 cache

≡ L2 cache

≡ L3 cache

≡ RAM

≡ Registere

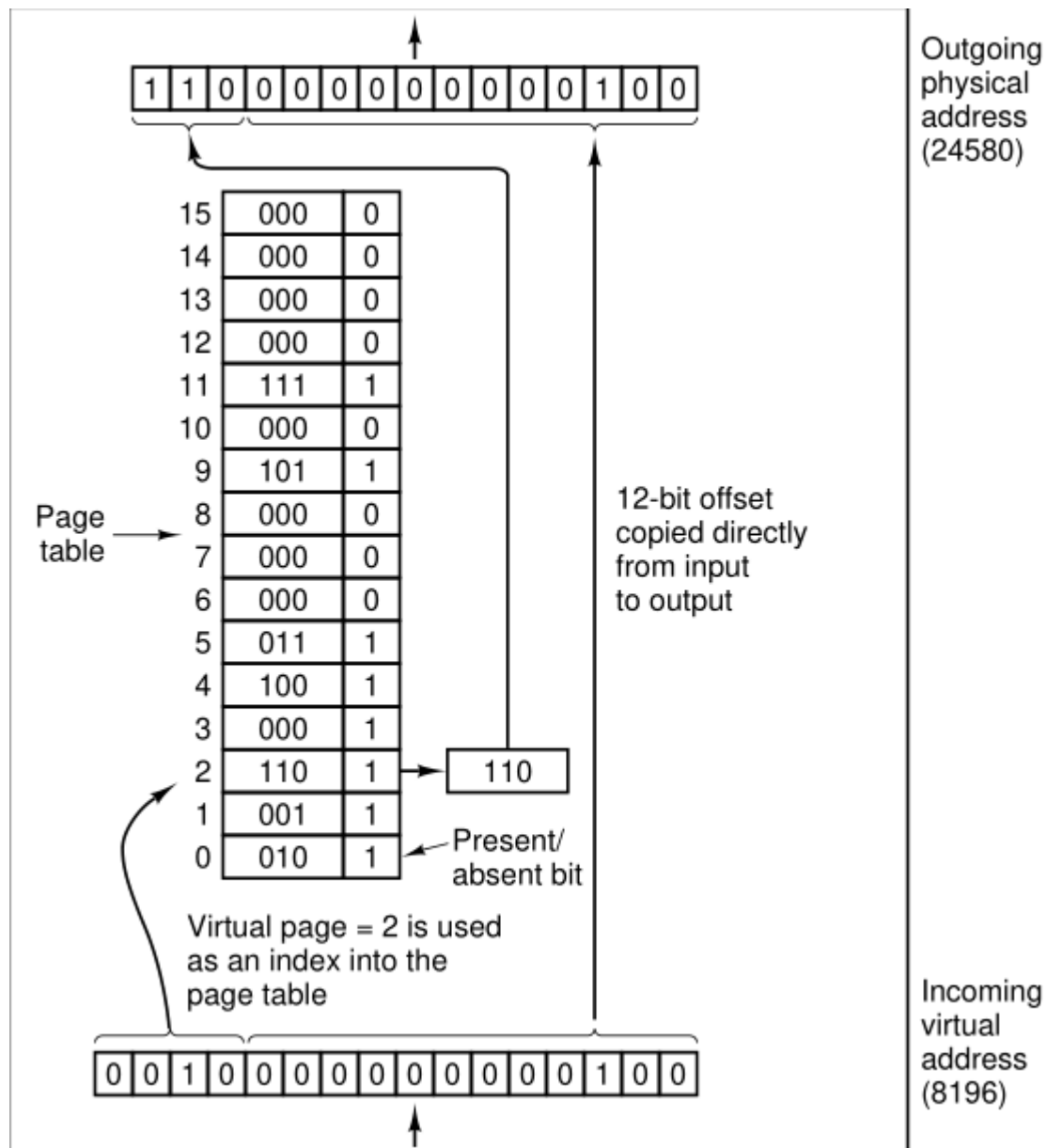
≡ HDD

MMU

Anta at en innkommende virtuell adresse til MMU er 12. Hva blir da den utgående fysiske adressen?

Legg merket til at du kan bruke shellen i Linux-VM til å for eksempel regne ut den desimale verdien av "Outgoing physical address" i eksempelet under:

```
kan@os:~$ echo 'ibase=2; 110000000000100' | bc
24580
```



Velg ett alternativ:

- ☐ 8200
- ☐ 8196
- ☐ 8204
- ☐ 24588
- ☐ 12
- ☐ 24592
- ☐ 24580
- ☐ Gir page fault
- ☐ 4096
- ☐ 4108

Seksjon 23

Hvorfor kan ikke MMU implementeres i software og styres av OS?

Velg ett alternativ:

- ☐ Fordi minne-aksess da ville ha tatt alt for lang tid
- ☐ Fordi RAM må kontrolleres av hardware, OS kan kun styre CPU
- ☐ Fordi man da ikke kunne finne mappingen for en virtuell adresse siden den ligger i fysisk RAM
- ☐ Fordi også vanlige brukerprosesser må kunne aksessere RAM
- ☐ Fordi det ikke ville være plass i RAM til hele page-tabellen
- ☐ Fordi det da ikke ville være mulig å bruke cache

Seksjon 24

Hva gjør følgende PowerShell-kommando?

((Get-date) - (Get-Process dockerd).StartTime).TotalHours

Velg ett alternativ

- ☐ Lister alle docker-prosesser som kjører nå og som har blitt startet de siste 60 minuttene.
- ☐ Gir antall timer dockerd-prosessen har kjørt
- ☐ Starter opp dockerd-prosessen
- ☐ Gir antall CPU-timer dockerd-prosessen har brukt
- ☐ Viser hvilket tidspunkt dockerd-prosessen startet opp
- ☐ Gir antall timer alle docker containere på systemet har kjørt

Seksjon 25

Skriv en PowerShell oneliner som gir deg hvor mange timer det er siden prosessen **pwsh** startet å kjøre.

*Du må ikke, men du kan prøve deg frem ved å kjøre kommandoen pwsh i Linux-VMen. Da vil du få et PowerShell der prosessen **pwsh** kjører.*

B

I

U

☰

☰
☰

á

0 / 10000 Word Limit