

Ukesoppgaver 13 – Plattformavhengighet, Java-threads, Dockercompose

RØD - Obligoppgaver

GUL – Ikke obligoppgaver

TURKIS – Ukens nøtt og utfordringer

1. (Oblig)

Under er det bilde av koden som kompileres og kjøres. Når man kompilerer et C-program med gcc så lager kompilatoren en kjørbar maskinkode fil som er optimalisert for den gjeldende plattformen som betyr at den kjørbare filen er kompilert for operativsystemet som kjører på datamaskinen man kompilerer programmet på. Og det er i dette tilfellet Linux eller annet Unix-liknende operativsystem.

```
[group25@os25:~/uke13AD/oppg1$ cat hello.c
#include <stdio.h>
int main()
{
    printf("c: Hello World!\n");
    return 0;
}
[group25@os25:~/uke13AD/oppg1$ gcc hello.c | chmod 700 a.out
[group25@os25:~/uke13AD/oppg1$ ./a.out
c: Hello World!]
```

Når jeg kjører a.out på en Windows maskin vil jeg få en feilmelding fordi a.out er en Linux-eksekverbare fil og er ikke kompatibel med Windows.

2. (Oblig)

Under er det Java-program som kompiles med javac og kjøres. Det er i prinsippet mulig å kjøre en .class-fil som er kompilert fra et Java-program på en Windows-PC, så lenge det finnes en installasjon av Java runtime environment (JRE) på PC-en. Det er viktig å huske at .class-filen må være kompilert med en versjon av Java som er kompatibel med JRE-installasjonen på Windows-PC-en. Hvis versjonene ikke er kompatible, kan det oppstå problemer ved kjøring av filen.

```
[group25@os25:~/uke13AD/oppg2$ cat hello.java
class Hello
{
    public static void main(String args[])
    {
        System.out.println("Java: Hello World!");
    }
}[group25@os25:~/uke13AD/oppg2$ sudo 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).
^[[A0 upgraded, 0 newly installed, 0 to remove and 39 not upgraded.
[group25@os25:~/uke13AD/oppg2$ javac hello.java
[group25@os25:~/uke13AD/oppg2$ java Hello
Java: Hello World!
-]
```

3. (Ikke oblig)

Java og Python kan regnes som plattformuavhengige, mens C og C++ er plattformavhengige. C og C++ kompileres direkte til maskinkode som er avhengig av datamaskinens arkitektur, operativsystem og annen maskinvare. Da må vi kompilere koden på en maskin som kjører samme operativsystem og arkitektur som den maskinen hvor koden skal kjøres.

Java og Python derimot kompileres til en mellomliggende bytecode, som kan kjøres på en såkalt VM. Denne VM-en er en programvare som kan kjøre på forskjellige plattformer, og oversetter bytecode til maskinkode som passer til den spesifikke plattformen. Dermed kan Java- og Python-kode skrives en gang og kjøres på forskjellige OS og arkitekturen uten å måtte skrives om.

4. (Oblig)

```
group25@os25:~/uke13AD/oppg4$ cat Calc.java
import java.lang.Thread;

class CalcThread extends Thread
{
    static int count = 0;
    int id;

    CalcThread()
    {
        count++;
        id = count;
    }

    public void run()
    {
        System.out.println("Thread nr." + id + " is starting");
        System.out.println("Thread nr." + id + " calculated " + work());
    }

    private float work()
    {
        int i, j;
        float res = 0;
        System.out.println("Thread nr." + id + " calculating");
        for(j = 1;j < 100;j++)
        {
            for(i = 1;i < 300000000;i++)
            {
                res += 1.0/(1.0*i*i*i*i*i*i*i);
            }
            System.out.println("Thread nr." + id + " calculating" + j);
        }
        return(res);
    }
}

class Calc
{
    public static void main(String args[])
    {
        System.out.println("Starts two threads !\n");
        CalcThread s = new CalcThread();
        System.out.println("Thread s has id " + s.id + "\n");
        s.start(); // Allokerer minne og kaller s.run()

        CalcThread s2 = new CalcThread();
        System.out.println("Thread s2 has id " + s2.id + "\n");
        s2.start();
        System.out.println("s2 started !\n");
    }
}
```

Over er koden til Calc.java.

```
[^[[A^[[A^Cgroup25@os25:~/uke13AD/oppg4$ taskset -c 0 java Calc | ps
 PID TTY      TIME CMD
 462878 pts/0    00:00:00 bash
 463482 pts/0    00:00:00 java
 463483 pts/0    00:00:00 ps
```

For å tvinge Java til å kjøre på bare en CPU med taskset skriver jeg

```
Taskset -c 0 java Calc
```

Da fikk hver tråd i programmet tilgang til bare den ene CPU-en. I dette tilfellet, siden det bare er to tråder (s og s2) og bare en CPU, vil hver tråd få tilgang til halvparten av CPU-tiden.

Hver tråd får ca. 50% av CPU-tiden. Dette kan variere avhengig av hvor mye annen aktivitet som foregår på systemet samtidig.

```
[group25@os25:~/uke13AD/oppg4$ java Calc | taskset -c 0 java Calc
Starts three threads !

Thread s has id 1

Thread s2 has id 2

Thread nr.1 is starting
Thread nr.1 calculating
s2 started !

Thread s3 has id 3

Thread nr.2 is starting
Thread nr.2 calculating
s3 started !

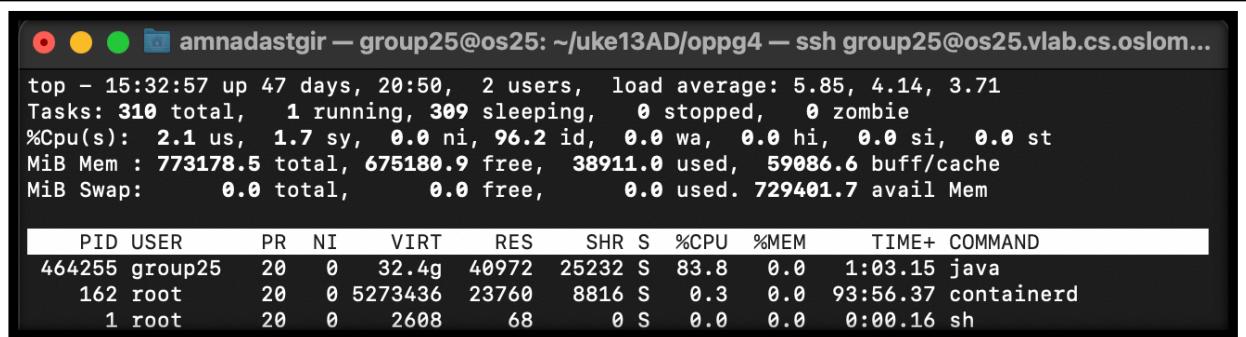
Thread nr.3 is starting
Thread nr.3 calculating
Thread nr.3 calculating1
```

Under starter jeg koden med 3 threads på en CPU.

```
[group25@os25:~/uke13AD/oppg4$ taskset -c 0 java Calc
Starts three threads !

Thread s has id 1
```

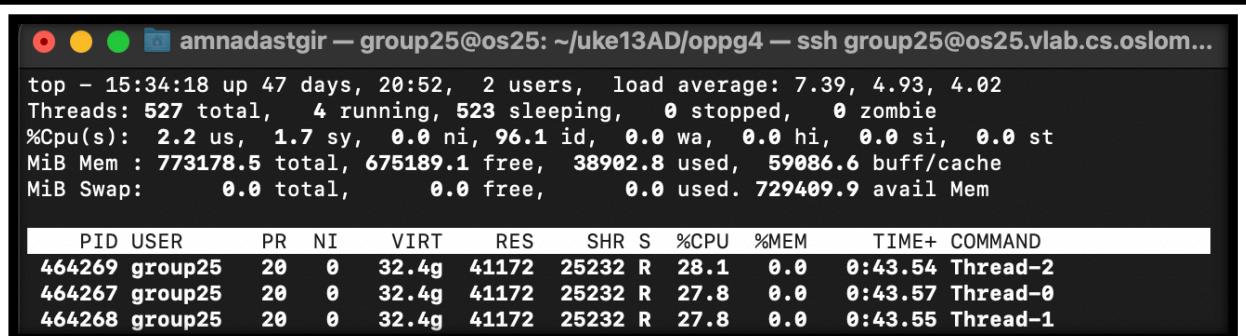
Under er det et bilde av hvordan %CPU bruken ser ut.



The screenshot shows the output of the top command. It displays system load average (5.85, 4.14, 3.71), task counts (310 total, 1 running, 309 sleeping, 0 stopped, 0 zombie), and CPU usage (%Cpu(s)). Below this, it shows memory usage (MiB Mem: 773178.5 total, 675180.9 free, 38911.0 used, 59086.6 buff/cache) and swap usage (MiB Swap: 0.0 total, 0.0 free, 0.0 used). The bottom part of the screenshot shows a table of process details:

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
464255	group25	20	0	32.4g	40972	25232	S	83.8	0.0	1:03.15	java
162	root	20	0	5273436	23760	8816	S	0.3	0.0	93:56.37	containerd
1	root	20	0	2608	68	0	S	0.0	0.0	0:00.16	sh

Her kan vi se threads ved å taste h i top. Og siden jeg har brukt taskset -c 0 på programmet så får hver thread 1/3 CPU.



The screenshot shows the output of the top command. It displays system load average (7.39, 4.93, 4.02), task counts (527 total, 4 running, 523 sleeping, 0 stopped, 0 zombie), and CPU usage (%Cpu(s)). Below this, it shows memory usage (MiB Mem: 773178.5 total, 675189.1 free, 38902.8 used, 59086.6 buff/cache) and swap usage (MiB Swap: 0.0 total, 0.0 free, 0.0 used). The bottom part of the screenshot shows a table of process details:

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
464269	group25	20	0	32.4g	41172	25232	R	28.1	0.0	0:43.54	Thread-2
464267	group25	20	0	32.4g	41172	25232	R	27.8	0.0	0:43.57	Thread-0
464268	group25	20	0	32.4g	41172	25232	R	27.8	0.0	0:43.55	Thread-1

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
464267	group25	20	0	32.4g	41192	25232	R	28.1	0.0	0:58.82	Thread-0
464269	group25	20	0	32.4g	41192	25232	R	28.1	0.0	0:58.79	Thread-2
464268	group25	20	0	32.4g	41192	25232	R	27.8	0.0	0:58.81	Thread-1
25088	root	20	0	52724.24	22724	2814	S	0.3	0.0	1:11.54	containerd

5. (Oblig)

Under er det programmet

```
[group25@os25:~/uke13AD/oppg5$ vim Calc.java
[group25@os25:~/uke13AD/oppg5$ cat Calc.java
import java.lang.Thread;
import java.io.PrintStream;

class CalcThread extends Thread
{
    static int count = 0;
    int id;
    long res=0;

    CalcThread()
    {
        count++;
        id = count;
    }

    public void run()
    {
        System.out.println("Thread nr." + id + " is starting");
        res = work();
    }

    private long work()
    {
        long i,j;
        if(id == 1)
        {
            System.out.println("Thread nr." + id + " calculating");
            for(i = 1;i <= 4000000000L;i++)
            {
                res += i;
            }
        }
        return(res);
    }
}
```

```
class Calc
{
    public static void main(String args[])
    {
        System.out.println("Starts two threads!");
        CalcThread s1 = new CalcThread();
        System.out.println("Thread s1 has id " + s1.id);
        s1.start(); // Allocates memory and calls s.run()

        CalcThread s2 = new CalcThread();
        System.out.println("Thread s2 has id " + s2.id);
        s2.start();
        System.out.println("s2 started !\n");
        try
        {
            s1.join();
        }
        catch (InterruptedException e)
        {
        }

        try
        {
            s2.join();
        }
        catch (InterruptedException e)
        {
        }
        System.out.printf("Thread nr. 1 calculated %d\n", s1.res);
        System.out.printf("Thread nr. 2 calculated %d\n", s2.res);
    }
}
```

Under kompileres koden og kjøres med time hvor realtime er 8,5 sekund.

```
[group25@os25:~/uke13AD/oppg5$ javac Calc.java
[group25@os25:~/uke13AD/oppg5$ java Calc
Starts two threads!
Thread s1 has id 1
Thread s2 has id 2
s2 started !

Thread nr.1 is starting
Thread nr.2 is starting
Thread nr.1 calculating
Thread nr. 1 calculated 8000000002000000000
Thread nr. 2 calculated 0
[group25@os25:~/uke13AD/oppg5$ time java Calc
Starts two threads!
Thread s1 has id 1
Thread s2 has id 2
s2 started !

Thread nr.2 is starting
Thread nr.1 is starting
Thread nr.1 calculating
Thread nr. 1 calculated 8000000002000000000
Thread nr. 2 calculated 0
Real:8.628 User:8.639 System:0.076 101.01%
```

Vi har en tråd som regner ut summen mellom 1 til 4 milliarder og en annen tråd som ikke gjør noe. Hvis jeg deler arbeidet mellom disse slik at den ene tråden regner ut summen mellom 1 til 2 milliarder og den andre tråden regner mellom 2 til 4, vil det gå dobbelt så fort, teoretisk.

Dette kan jeg gjøre ved å endre arbeidsfordelingen i work() metoden ved å for eksempel legge til en parameter som indikerer hvilken del av arbeidet tråden skal gjøre.

Under gjør koden dette:



The screenshot shows a terminal window with the following Java code:

```
amnadarstgir ~ amnadarstgir:~/uke13AD/oppg5 ssh group25@os25.vlab.cs.oslomet.no F10 key ==> File Edit Search Buffers Windows System Help
```

```
class CalcThread extends Thread
{
    static int count = 0;
    int id;
    long res = 0;

    CalcThread()
    {
        count++;
        id = count;
    }

    public void run()
    {
        System.out.println("Thread nr." + id + " is starting");
        res = work();
    }

    private long work()
    {
        long start, end;
        if (id == 1)
        {
            start = 1;
            end = 2000000000L;
        }
        else
        {
            start = 2000000001L;
            end = 4000000000L;
        }

        System.out.println("Thread nr." + id + " calculating");
        for (long i = start; i <= end; i++)
        {
            res += i;
        }
    }

    return res;
}
```

Under kan vi se at koden bruker halvert tid, 4 sekunder.

```
[group25@os25:~/uke13AD/oppg5$ time java Calc
Starts two threads!
Thread s1 has id 1
Thread s2 has id 2
s2 started !

Thread nr.1 is starting
Thread nr.2 is starting
Thread nr.1 calculating
Thread nr.2 calculating
Thread nr. 1 calculated 2000000001000000000
Thread nr. 2 calculated 6000000001000000000
Real:4.392 User:8.719 System:0.016 198.90%
```

6. (Ikke oblig)

GJORT!

7. (Oblig)

SVAR FRA NETT:

Selv om en datamaskin bare har ett sett med registre, vil hver tråd i en prosess ha sin egen stakk og programteller som vil være forskjellige for hver tråd. Registrene i en datamaskin er en begrenset ressurs som brukes av CPU-en for å lagre midlertidige data og instruksjoner som skal utføres. Hver tråd som kjører på CPU-en har sin egen stakk og programteller, som brukes til å lagre og spore midlertidige data og instruksjoner spesifikke for den tråden. Derfor blir registrene vanligvis betraktet som «Per thread item», siden hver tråd har sitt eget sett med registerverdier.

På den andre siden vil globale variabler, åpne filer og child prosesser være delte ressurser mellom alle trådene i en prosess, og disse vil derfor bli betraktet som «Per process item».

Det er viktig å skille mellom disse to typer ressurser, slik at man kan forstå hvordan de brukes og påvirker hverandre i en flertrådet applikasjon.

8. (Ikke Oblig)

En tråd kan kalle `yield()` sånn at andre tråder eller prosesser får sjansen til å kjøre, og det kan forbedre effektiviteten og ressursbruken til systemet. Selv om det ikke finnes periodiske

klokke-interrupt innenfor en prosess, vil OS vanligvis gi hver tråd en rettferdig andel av CPU-tiden til å utføre sine oppgaver, så det er usannsynlig at tråden aldri får CPU-tiden tilbake.

9. (Ikke Oblig)

En tråd kan tvinges til å stoppe å kjøre (preempted) av et klokke-interrupt når OS bestemmer seg for å bytte til en annen tråd som skal kjøre på samme CPU-kjerne. Dette skjer vanligvis når OS planlegger å dele CPU-tiden rettferdig mellom de forskjellige trådene som kjører på systemet.

10. (Oblig)

I et system med user-level threads, er det en stack per thread ikke en stack per prosess. Dette er fordi hver thread i brukermodus har sin egen kontekst, som inkluderer en egen stack, som er uavhengig av andre tråder i samme prosess. Når det gjelder kernel-level threads, så lages en egen kernel stack for hver thread, uavhengig av om trådene tilhører samme prosess eller ikke. Kernel-level threads bruker også vanligvis et eget sett med register, som betyr at hver thread krever sin egen kontekst og dermed en egen stack.

Så i begge tilfeller uansett om det er user-level eller kernel-level threads, er det en stack per thread som er nødvendig for å håndtere trådens egen kontekst og for å sikre at tråden kan kjøre på en pålitelig måte.

11. (Ikke Oblig)

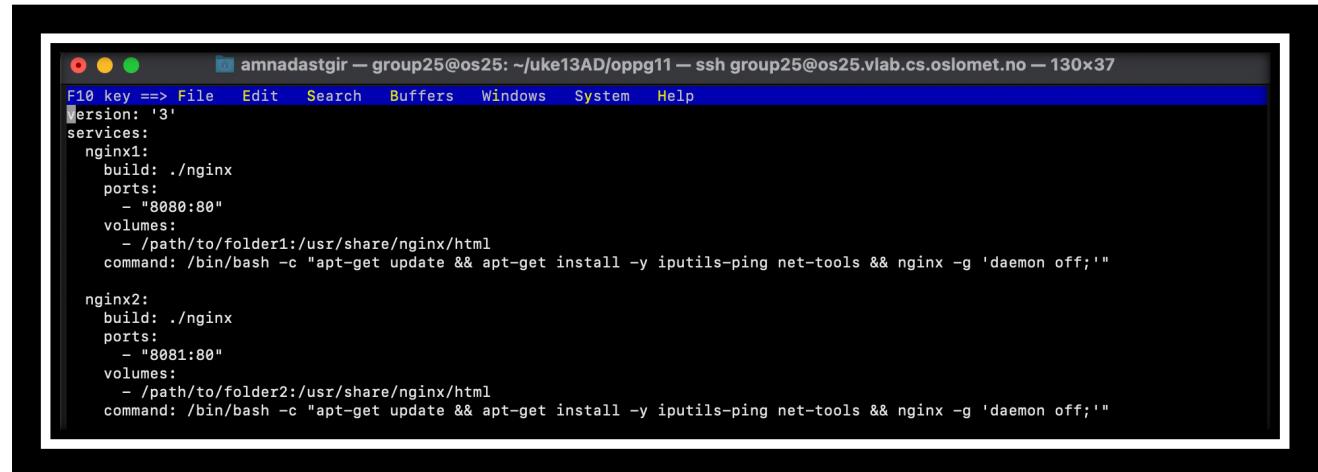
Under er strukturen til oppg11 mappa:

```
[group25@os25:~/uke13AD/oppg11$ ls -l
total 12
-rwx----- 1 group25 group25 489 Apr  7 02:45 docker-compose.yaml
-rwx----- 1 group25 group25 477 Apr  7 02:43 docker-compose.yaml~
drwxrwxr-x 2 group25 group25 4096 Apr  7 02:51 nginx
```

```
[group25@os25:~/uke13AD/oppg11$ tree
.
├── docker-compose.yaml
├── docker-compose.yaml~
└── nginx
    └── dockerfile

1 directory, 3 files
```

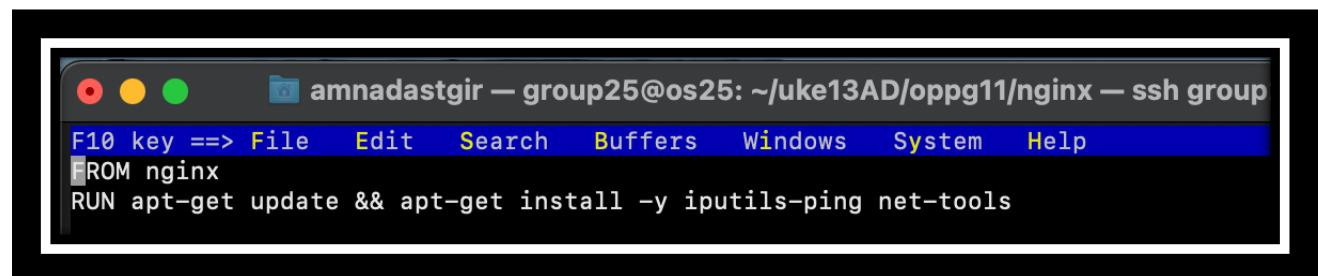
Følgende fil vil opprette to Nginx-containere med ulike porter og monterte volumer. Den vil bygge imagene fra ‘./nginx’, portene 8080 og 8081, montere ulike kataloger på vertsmaskinene som volum.



```
amnadarstgir — group25@os25: ~/uke13AD/oppg11 — ssh group25@os25.vlab.cs.oslomet.no — 130x37
F10 key ==> File Edit Search Buffers Windows System Help
version: '3'
services:
  nginx1:
    build: ./nginx
    ports:
      - "8080:80"
    volumes:
      - /path/to/folder1:/usr/share/nginx/html
    command: /bin/bash -c "apt-get update && apt-get install -y iputils-ping net-tools && nginx -g 'daemon off;'"

  nginx2:
    build: ./nginx
    ports:
      - "8081:80"
    volumes:
      - /path/to/folder2:/usr/share/nginx/html
    command: /bin/bash -c "apt-get update && apt-get install -y iputils-ping net-tools && nginx -g 'daemon off;'"
```

Under er det dockerfile



```
amnadarstgir — group25@os25: ~/uke13AD/oppg11/nginx — ssh group
F10 key ==> File Edit Search Buffers Windows System Help
FROM nginx
RUN apt-get update && apt-get install -y iputils-ping net-tools
```

12. (Oblig)

Feil, men kan

13. EFFICODE!

```
[group25@os25:~$ sudo su
[root@os25:/home/group25# docker pull haugerud/os:s256
s256: Pulling from haugerud/os
f1f26f570256: Already exists
84181e80d10e: Already exists
1ff0f94a8007: Already exists
d776269cad10: Already exists
e9427fcfa864: Already exists
d4ceccbf269: Already exists
3056bf902c7f: Pull complete
Digest: sha256:6924e0b789282fc7983dce64e7a26ebf6b3c9a4b5451ae61eb437072002a8db7
Status: Downloaded newer image for haugerud/os:s256
docker.io/haugerud/os:s256
[root@os25:/home/group25# /home/s264# docker images
bash: /home/s264#: No such file or directory
[root@os25:/home/group25# docker pull haugerud/os:s256
s256: Pulling from haugerud/os
Digest: sha256:6924e0b789282fc7983dce64e7a26ebf6b3c9a4b5451ae61eb437072002a8db7
Status: Image is up to date for haugerud/os:s256
docker.io/haugerud/os:s256
[root@os25:/home/group25# docker images
REPOSITORY          TAG      IMAGE ID      CREATED     SIZE
oppg11-nginx1      latest   d8dac49d8127  7 minutes ago  163MB
oppg11-nginx2      latest   ea23ff8ec0a0  7 minutes ago  163MB
sum                latest   acc3646a09ff  10 days ago   1.13GB
oppg12_loadbalancer latest   2dab0c9d7a17  10 days ago  142MB
oppg11_nginx2      latest   915960bc7a13  10 days ago  163MB
oppg11_nginx       latest   915960bc7a13  10 days ago  163MB
haugerud/os        s256    bcb1875cfabc  11 days ago  142MB
nginx              latest   ac232364af84  2 weeks ago  142MB
webapache_amna     latest   64f41950eda9  2 weeks ago  228MB
[root@os25:/home/group25# docker run -it acc36 bin/bash
[root@dba248c41074:/# find -name "index.html"
./var/www/html/index.html
./usr/share/apache2/default-site/index.html
find: './proc/tty/driver': Permission denied
[root@dba248c41074:/# cat ./usr/share/nginx/html/index.html
cat: ./usr/share/nginx/html/index.html: No such file or directory
[root@dba248c41074:/# cd /usr
[root@dba248c41074:/usr# cd /share
bash: cd: /share: No such file or directory
[root@dba248c41074:/usr# exit
exit
[root@os25:/home/group25# docker run -it 2dab0 bin/bash
[root@4b295a541865:/# cd /usr
[root@4b295a541865:/usr# cd /share
bash: cd: /share: No such file or directory
[root@4b295a541865:/usr# exit
exit
[root@os25:/home/group25# docker run -it bcb18 bin/bash
[root@b2665270b8a3:/# cat ./usr/share/nginx/html/index.html
Y0VRMChmX4
root@b2665270b8a3:/# ]
```

