20HT – 1DV512 – Operating Systems Group Assignment 2

i

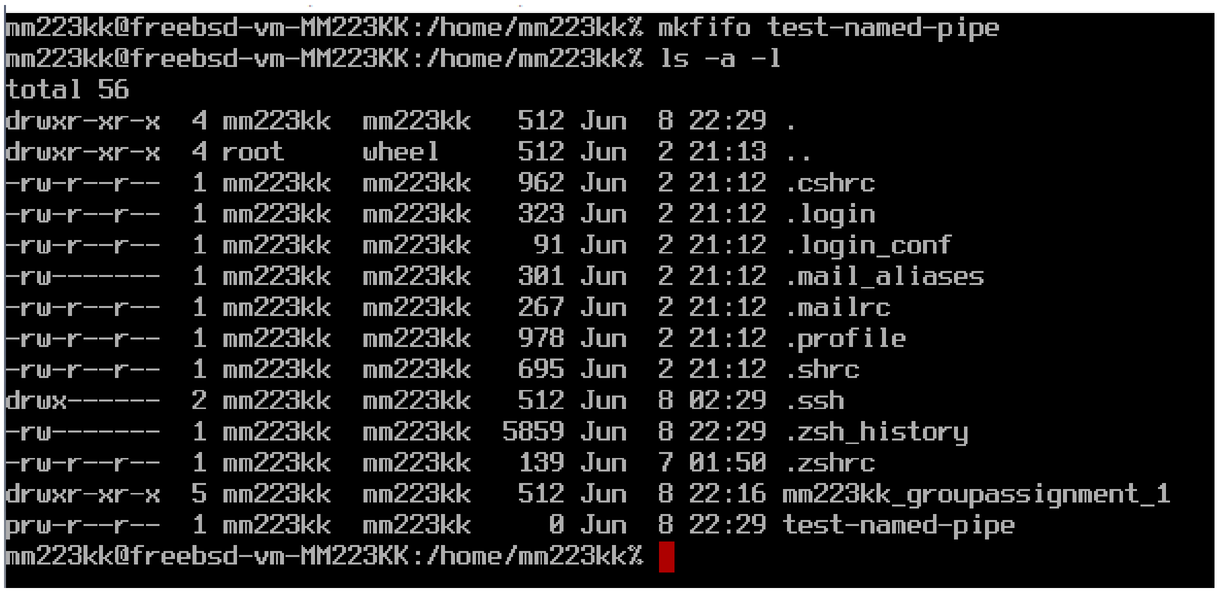
Student: Sami Mwanje

ID/mail: mm223kk@student.lnu.se

Assignment date: 2020-12-06

Hand in date: 2021-06-18

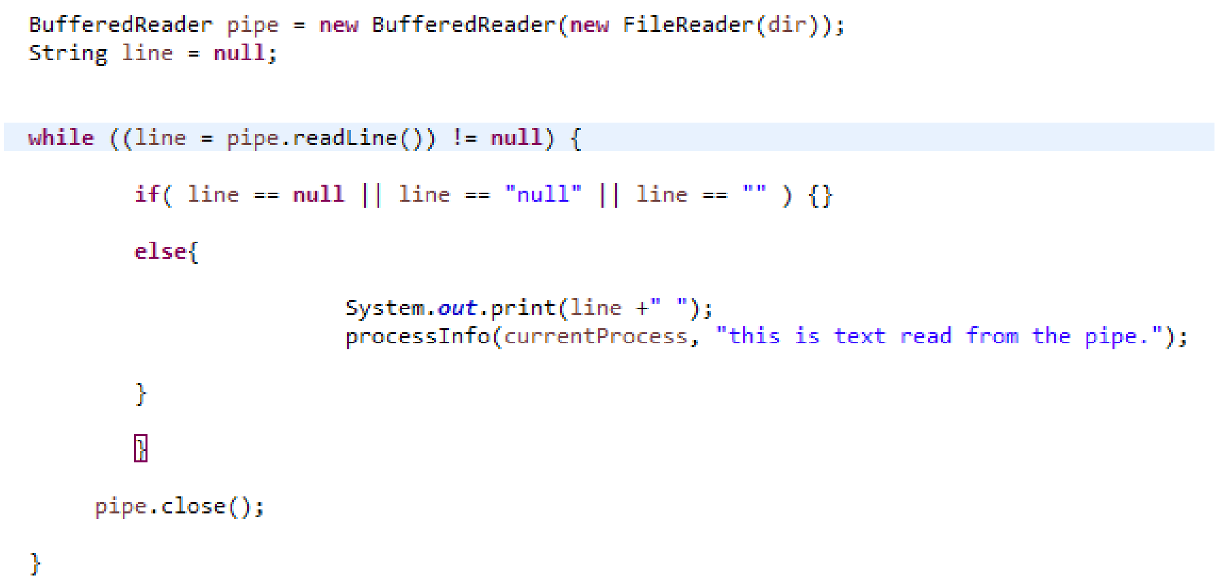
Task 1:



```
mm223kk@freebsd-vm-MM223KK:/home/mm223kk% mkfifo test-named-pipe
mm223kk@freebsd-vm-MM223KK:/home/mm223kk% ls -a -l
total 56
drwxr-xr-x  4 mm223kk  mm223kk   512 Jun  8 22:29 .
drwxr-xr-x  4 root     wheel     512 Jun  2 21:13 ..
-rw-r--r--  1 mm223kk  mm223kk   962 Jun  2 21:12 .cshrc
-rw-r--r--  1 mm223kk  mm223kk   323 Jun  2 21:12 .login
-rw-r--r--  1 mm223kk  mm223kk    91 Jun  2 21:12 .login_conf
-rw-------  1 mm223kk  mm223kk   301 Jun  2 21:12 .mail_aliases
-rw-r--r--  1 mm223kk  mm223kk   267 Jun  2 21:12 .mailrc
-rw-r--r--  1 mm223kk  mm223kk   978 Jun  2 21:12 .profile
-rw-r--r--  1 mm223kk  mm223kk   695 Jun  2 21:12 .shrc
drwx------  2 mm223kk  mm223kk   512 Jun  8 02:29 .ssh
-rw-------  1 mm223kk  mm223kk  5859 Jun  8 22:29 .zsh_history
-rw-r--r--  1 mm223kk  mm223kk   139 Jun  7 01:50 .zshrc
drwxr-xr-x  5 mm223kk  mm223kk   512 Jun  8 22:16 mm223kk_groupassignment_1
prw-r--r--  1 mm223kk  mm223kk     0 Jun  8 22:29 test-named-pipe
mm223kk@freebsd-vm-MM223KK:/home/mm223kk%
```

First, I create I pipe using the line "**mkfifo test-named-pipe**". To check if the new pipe has been created and to see the permissions I in invoke the line "**ls -a -l**" A pipe is a special file that allows first in first out. We can see that the permission code is "**prw-r—r—".** The first later on the permission code "**p**" stand for pipe a regular file has "**-**" in the beginning.

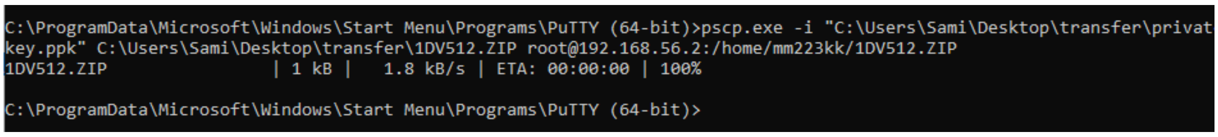
1.2:



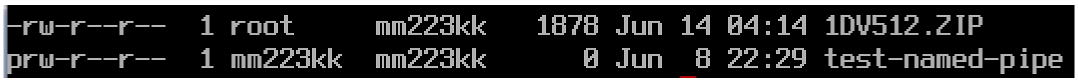
```
BufferedReader pipe = new BufferedReader(new FileReader(dir));
String line = null;


while ((line = pipe.readLine()) != null) {

        if( line == null || line == "null" || line == "" ) {}

        else{

                        System.out.print(line +" ");
                        processInfo(currentProcess, "this is text read from the pipe.");

        }

        }

      pipe.close();

}
```

Implementing a pipe reader method. Had a problem with the program block itself after the pipe has been read. The program returns to the loop when "**CTRL+C**" is used on another shell calling "**cat > name.pipe**".

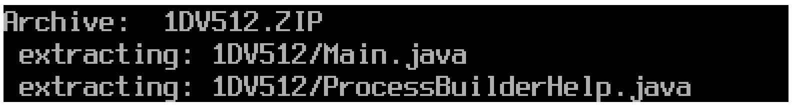
1.3:



```
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\PuTTY (64-bit)>pscp.exe -i "C:\Users\Sami\Desktop\transfer\privat
key.ppk" C:\Users\Sami\Desktop\transfer\1DV512.ZIP root@192.168.56.2:/home/mm223kk/1DV512.ZIP
1DV512.ZIP                | 1 kB |   1.8 kB/s | ETA: 00:00:00 | 100%

C:\ProgramData\Microsoft\Windows\Start Menu\Programs\PuTTY (64-bit)>
```

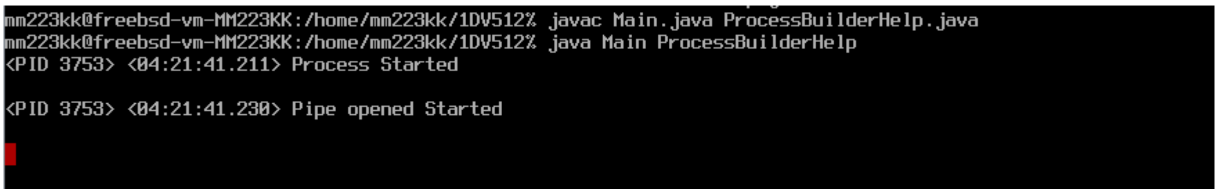


```
-rw-r--r--  1 root     mm223kk   1878 Jun 14 04:14 1DV512.ZIP
prw-r--r--  1 mm223kk  mm223kk     0 Jun  8 22:29 test-named-pipe
```



```
Archive:  1DV512.ZIP
 extracting: 1DV512/Main.java
 extracting: 1DV512/ProcessBuilderHelp.java
```

After I had created the java program from the instructions, I transferred it to the VM using **PuTTy** and **pscp.exe**. I unzipped the files using "**unzip 1DV512.ZIP**". The program will have a Main class and a **ProcessBuilderHelp** class.

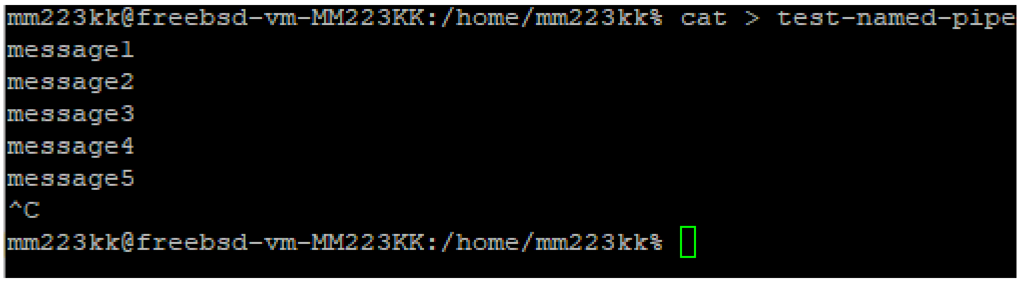


```
mm223kk@freebsd-vm-MM223KK:/home/mm223kk/1DV512% javac Main.java ProcessBuilderHelp.java
mm223kk@freebsd-vm-MM223KK:/home/mm223kk/1DV512% java Main ProcessBuilderHelp
<PID 3753> <04:21:41.211> Process Started

<PID 3753> <04:21:41.230> Pipe opened Started
```

I then compiled the files using "**javac**" and went for a run suing "**java**".

1.4 - 1.5:

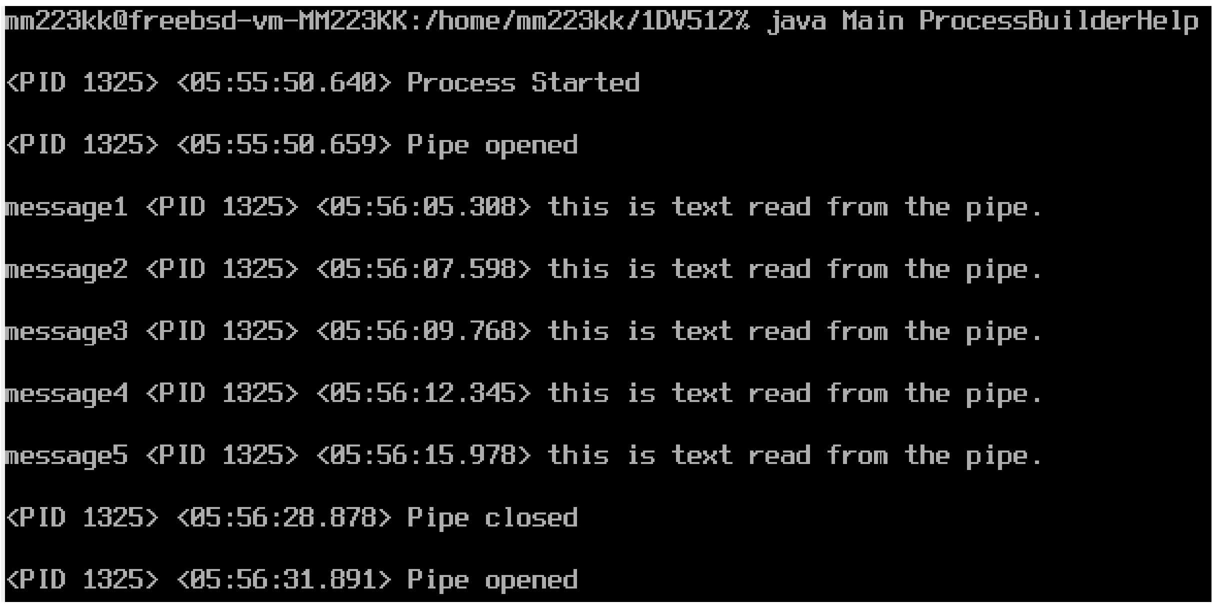
PuTTy terminal:



```
mm223kk@freebsd-vm-MM223KK:/home/mm223kk% cat > test-named-pipe
message1
message2
message3
message4
message5
^C
mm223kk@freebsd-vm-MM223KK:/home/mm223kk%
```

On the **PuTTy** terminal I typed message1, message2… Iin order to see if the other shell would echo this. I closed the pipe by using "CTRL+C" which makes the java program to read null and return to the outer loop.

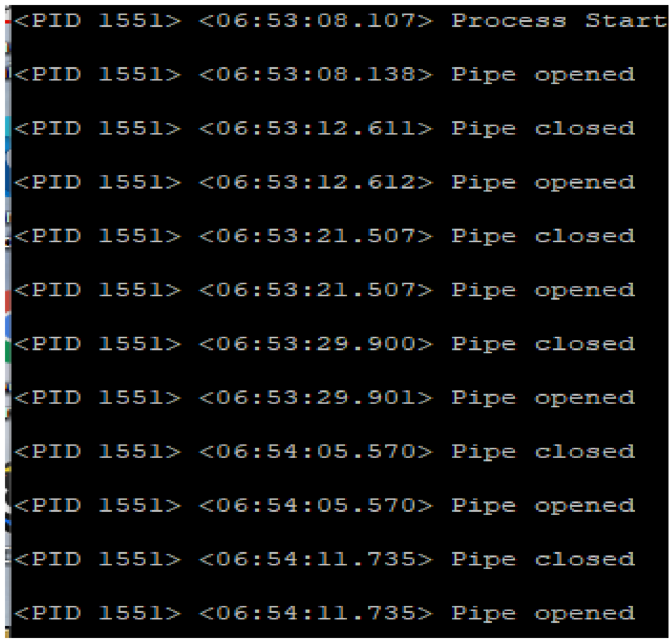
Vm terminal:



```
mm223kk@freebsd-vm-MM223KK:/home/mm223kk/1DV512% java Main ProcessBuilderHelp
<PID 1325> <05:55:50.640> Process Started
<PID 1325> <05:55:50.659> Pipe opened
message1 <PID 1325> <05:56:05.308> this is text read from the pipe.
message2 <PID 1325> <05:56:07.598> this is text read from the pipe.
message3 <PID 1325> <05:56:09.768> this is text read from the pipe.
message4 <PID 1325> <05:56:12.345> this is text read from the pipe.
message5 <PID 1325> <05:56:15.978> this is text read from the pipe.
<PID 1325> <05:56:28.878> Pipe closed
<PID 1325> <05:56:31.891> Pipe opened
```

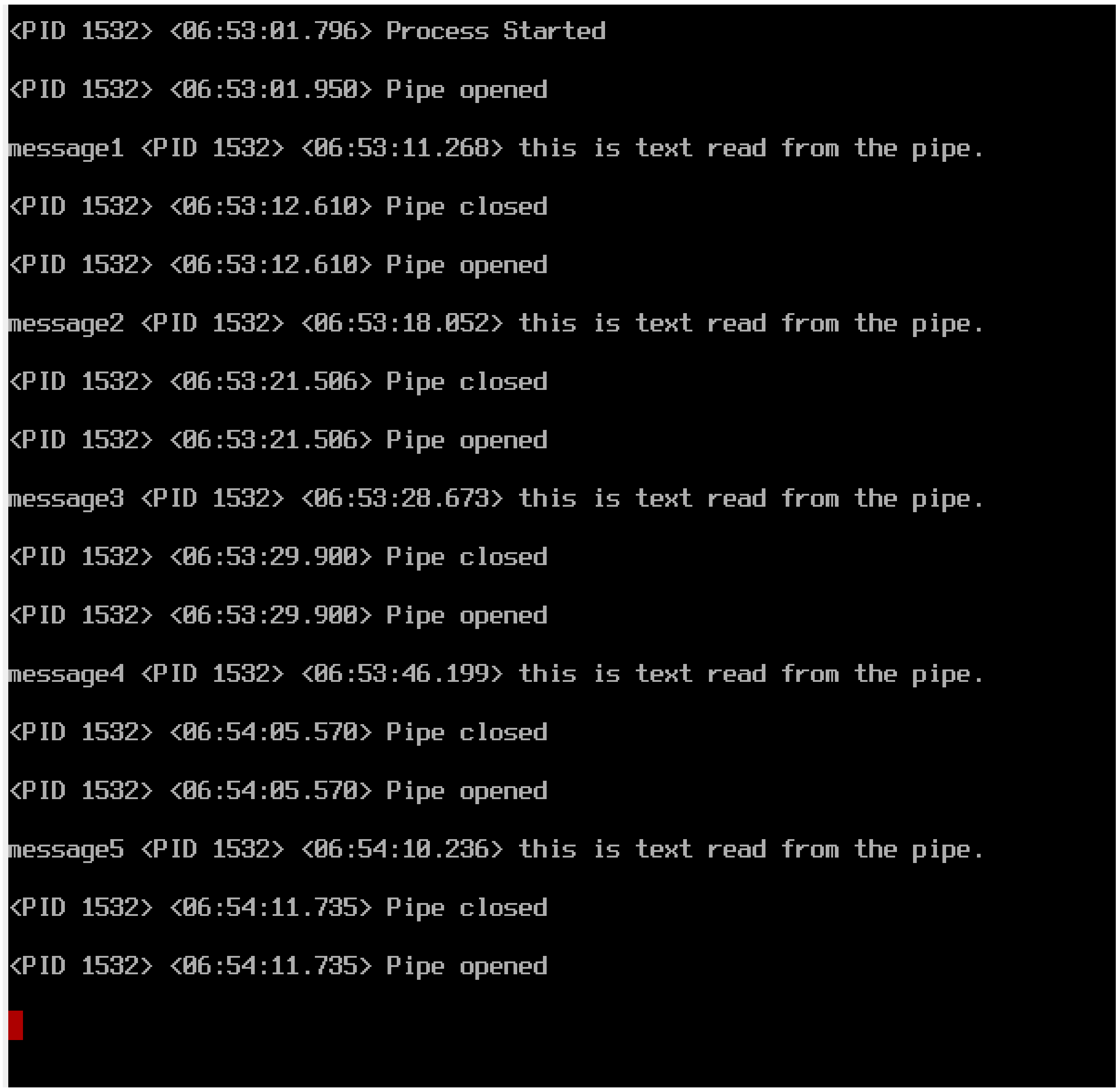
By looking on the VM terminal I could see that everything that was typed on the **PuTTy** shell was echoed on the VM terminal, thanks to the Java Program. The "**cat > test-named-pipe"** is used to write data into a pipe. The pipe is then read by the java program once it receives. To be more precise the data is waiting for someone to open it on the other end, since the java program is opening it, the typed command is directly echoed. I had some problem after the java program start reading the pipe it blocks the code from going further, even though there is no command attending. The block could only be avoided by using "**Ctrl+c**" on the "**cat > pipe.name**" shell.

1.6: Terminal 1:



```
<PID 1551> <06:53:08.107> Process Start
<PID 1551> <06:53:08.138> Pipe opened
<PID 1551> <06:53:12.611> Pipe closed
<PID 1551> <06:53:12.612> Pipe opened
<PID 1551> <06:53:21.507> Pipe closed
<PID 1551> <06:53:21.507> Pipe opened
<PID 1551> <06:53:29.900> Pipe closed
<PID 1551> <06:53:29.901> Pipe opened
<PID 1551> <06:54:05.570> Pipe closed
<PID 1551> <06:54:05.570> Pipe opened
<PID 1551> <06:54:11.735> Pipe closed
<PID 1551> <06:54:11.735> Pipe opened
```

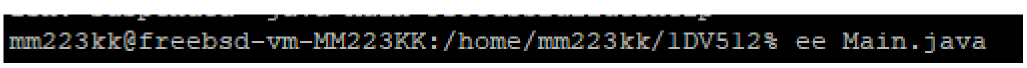
Terminal 2:



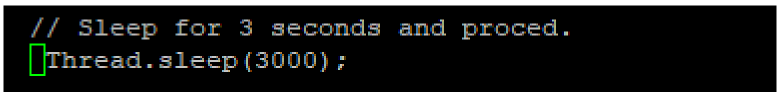
```
<PID 1532> <06:53:01.796> Process Started

<PID 1532> <06:53:01.950> Pipe opened

message1 <PID 1532> <06:53:11.268> this is text read from the pipe.

<PID 1532> <06:53:12.610> Pipe closed

<PID 1532> <06:53:12.610> Pipe opened

message2 <PID 1532> <06:53:18.052> this is text read from the pipe.

<PID 1532> <06:53:21.506> Pipe closed

<PID 1532> <06:53:21.506> Pipe opened

message3 <PID 1532> <06:53:28.673> this is text read from the pipe.

<PID 1532> <06:53:29.900> Pipe closed

<PID 1532> <06:53:29.900> Pipe opened

message4 <PID 1532> <06:53:46.199> this is text read from the pipe.

<PID 1532> <06:54:05.570> Pipe closed

<PID 1532> <06:54:05.570> Pipe opened

message5 <PID 1532> <06:54:10.236> this is text read from the pipe.

<PID 1532> <06:54:11.735> Pipe closed

<PID 1532> <06:54:11.735> Pipe opened
```

As it can be viewed only on of the terminals receive the messages from the "**cat > name.pipe**" While the PuTTy terminal only opens and closes the read of the pipe the other one is printing the received messages. This means that once a program/shell starts reading from a pipe it will be the "main" reader until it goes to sleep. The other shell can not view the other end of the pipe until the other one is done viewing. First to get access gets access.

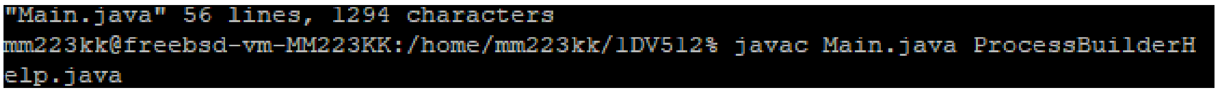
1.7:



```
mm223kk@freebsd-vm-MM223KK:/home/mm223kk/1DV512% ee Main.java
```

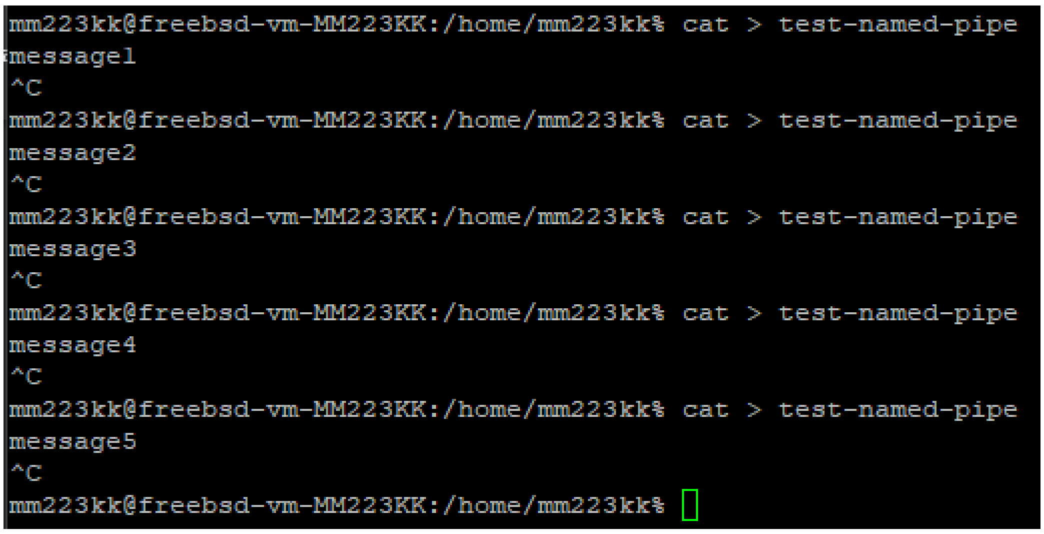


```
 // Sleep for 3 seconds and proced.
 Thread.sleep(3000);
```



```
"Main.java" 56 lines, 1294 characters
mm223kk@freebsd-vm-MM223KK:/home/mm223kk/1DV512% javac Main.java ProcessBuilderH
elp.java
```

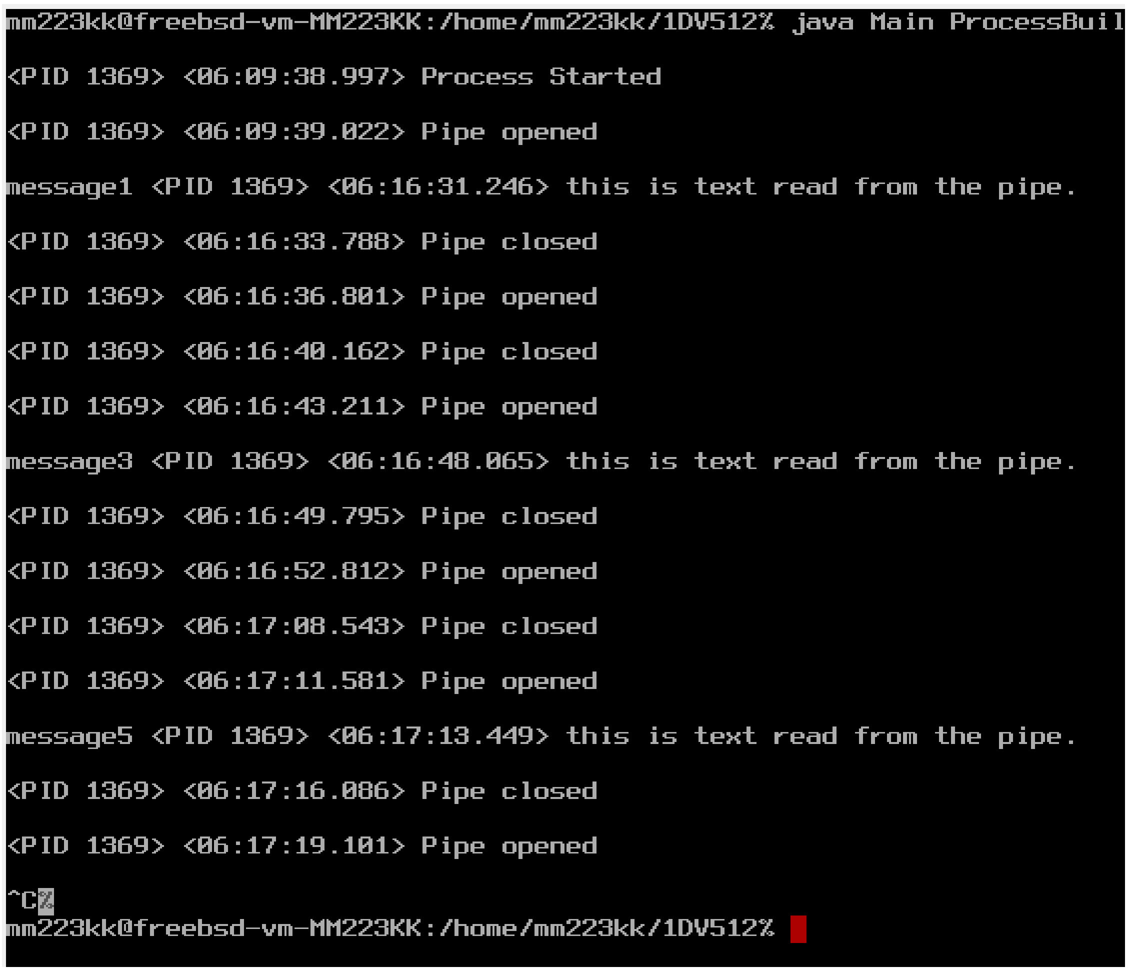
Since this was a quite easy task to do, I thought it would be unnecessary to redo the whole process by transferring a new edited java file into the VM again. I used "**ee Main.java**" to edit the current java file. I uncommented the line where **thread.sleep(3000)** is after that I recompiled the file and it was now ready to run but with a newly added 3s sleep.



```
mm223kk@freebsd-vm-MM223KK:/home/mm223kk% cat > test-named-pipe
message1
^C
mm223kk@freebsd-vm-MM223KK:/home/mm223kk% cat > test-named-pipe
message2
^C
mm223kk@freebsd-vm-MM223KK:/home/mm223kk% cat > test-named-pipe
message3
^C
mm223kk@freebsd-vm-MM223KK:/home/mm223kk% cat > test-named-pipe
message4
^C
mm223kk@freebsd-vm-MM223KK:/home/mm223kk% cat > test-named-pipe
message5
^C
mm223kk@freebsd-vm-MM223KK:/home/mm223kk%
```

The first terminal was used to "**cat > test-named-pipe**" type a message and then exit the pipe with "**CTRL + C**" in order to make the other shells pause 3s and exit from the block.

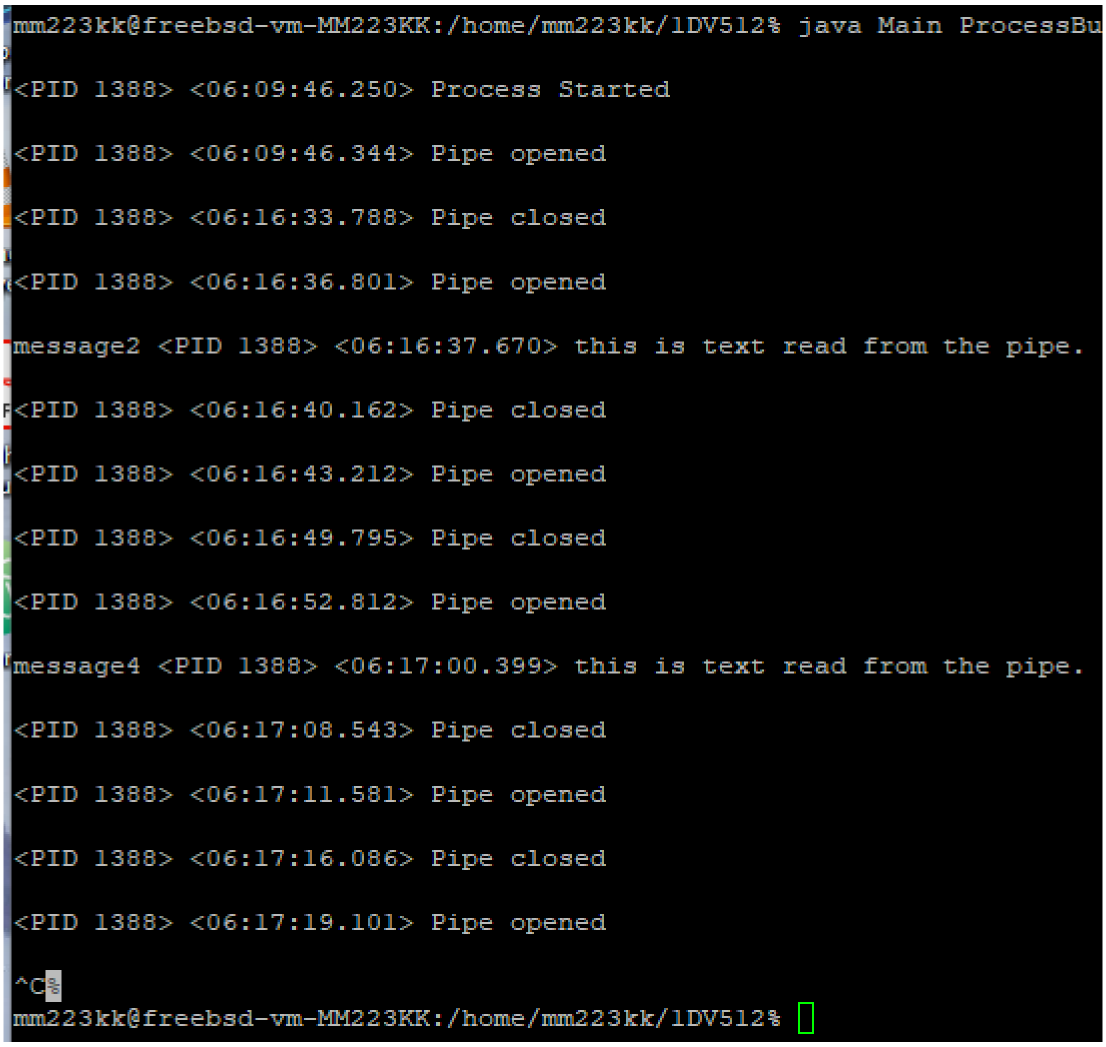
The result on the VM shell can be viewed here:



```
mm223kk@freebsd-vm-MM223KK:/home/mm223kk/1DV512% java Main ProcessBuil
<PID 1369> <06:09:38.997> Process Started
<PID 1369> <06:09:39.022> Pipe opened
message1 <PID 1369> <06:16:31.246> this is text read from the pipe.
<PID 1369> <06:16:33.788> Pipe closed
<PID 1369> <06:16:36.801> Pipe opened
<PID 1369> <06:16:40.162> Pipe closed
<PID 1369> <06:16:43.211> Pipe opened
message3 <PID 1369> <06:16:48.065> this is text read from the pipe.
<PID 1369> <06:16:49.795> Pipe closed
<PID 1369> <06:16:52.812> Pipe opened
<PID 1369> <06:17:08.543> Pipe closed
<PID 1369> <06:17:11.581> Pipe opened
message5 <PID 1369> <06:17:13.449> this is text read from the pipe.
<PID 1369> <06:17:16.086> Pipe closed
<PID 1369> <06:17:19.101> Pipe opened
^C%
mm223kk@freebsd-vm-MM223KK:/home/mm223kk/1DV512%
```

As it can be viewed on the result the VM shell only received message 1,3 and five. This tell us that data on exist in the pipe once until it has been read from another end. When data is finale read it no longer exist inside the pipe. So, it is all about which terminal reads the “**catted**” data first. As seen on the terminal the VM shell read **message 1**,**3** and **5**. This means that message 2 and 4 whet to the other shell in PuTTy. Each time “CTRL + C” was invoked the program went to sleep for 3s on the current terminal. Which gives the other terminal the possibility of receiving coming data in 3s.

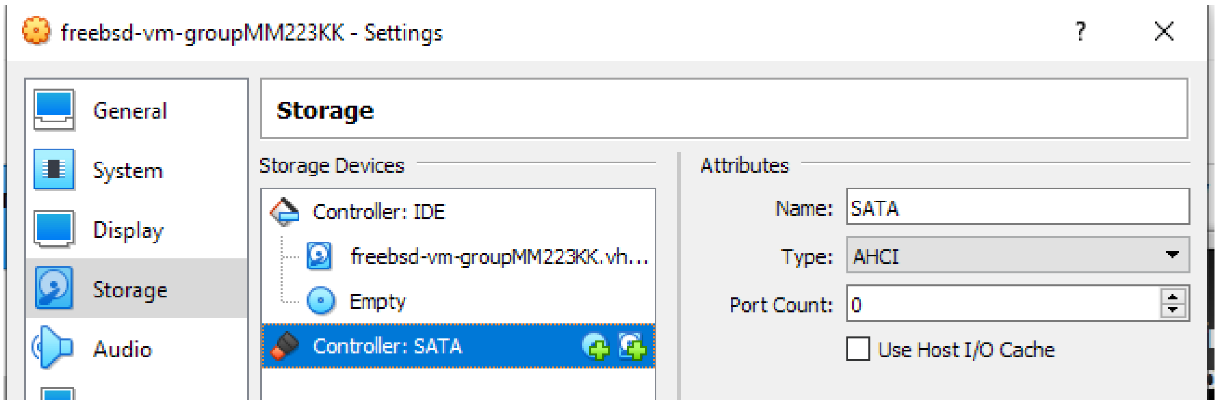
The Putty terminal received message 2 and 4. As just stated the Putty terminal was the winner of receiving these two messages first. Therefore, it was printed in that terminal. If I would have not used “**CTRL + S**” only one terminal would have printed all the data, while the other never gets the chance to view it.



```
mm223kk@freebsd-vm-MM223KK:/home/mm223kk/1DV512% java Main ProcessBu
<PID 1388> <06:09:46.250> Process Started
<PID 1388> <06:09:46.344> Pipe opened
<PID 1388> <06:16:33.788> Pipe closed
<PID 1388> <06:16:36.801> Pipe opened
message2 <PID 1388> <06:16:37.670> this is text read from the pipe.
<PID 1388> <06:16:40.162> Pipe closed
<PID 1388> <06:16:43.212> Pipe opened
<PID 1388> <06:16:49.795> Pipe closed
<PID 1388> <06:16:52.812> Pipe opened
message4 <PID 1388> <06:17:00.399> this is text read from the pipe.
<PID 1388> <06:17:08.543> Pipe closed
<PID 1388> <06:17:11.581> Pipe opened
<PID 1388> <06:17:16.086> Pipe closed
<PID 1388> <06:17:19.101> Pipe opened
^C%
mm223kk@freebsd-vm-MM223KK:/home/mm223kk/1DV512%
```

Sami Mwanje
mm223kk@student.lnu.se

Task 2:

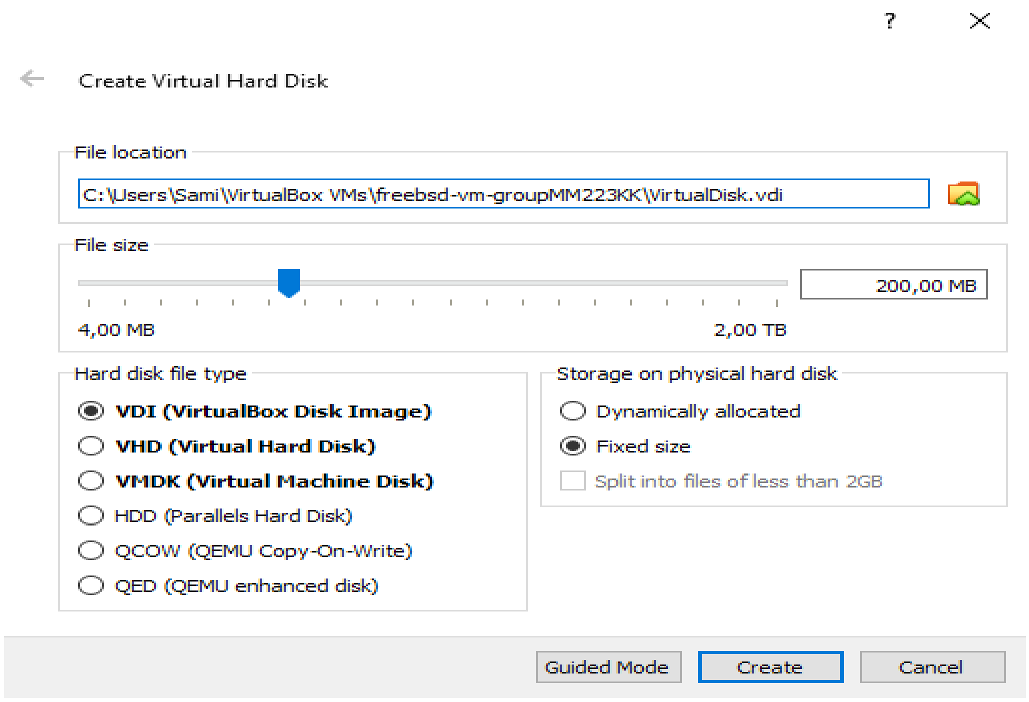

freebsd-vm-groupMM223KK - Settings

Storage

Storage Devices
- Controller: IDE
  - freebsd-vm-groupMM223KK.vh...
  - Empty
- Controller: SATA

Attributes

Name: SATA
Type: AHCI
Port Count: 0
☐ Use Host I/O Cache

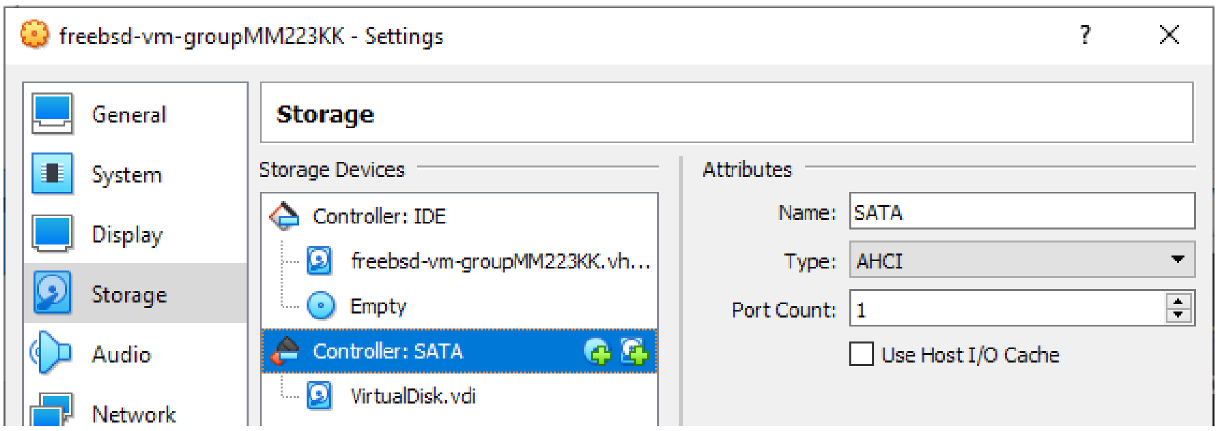

Create Virtual Hard Disk

File location
C:\Users\Sami\VirtualBox VMs\freebsd-vm-groupMM223KK\VirtualDisk.vdi

File size
200,00 MB
4,00 MB — 2,00 TB

Hard disk file type
- [✓ VDI (VirtualBox Disk Image)]
- VHD (Virtual Hard Disk)
- VMDK (Virtual Machine Disk)
- HDD (Parallels Hard Disk)
- QCOW (QEMU Copy-On-Write)
- QED (QEMU enhanced disk)

Storage on physical hard disk
- Dynamically allocated
- [✓ Fixed size]
- ☐ Split into files of less than 2GB


freebsd-vm-groupMM223KK - Settings

Storage

Storage Devices
- Controller: IDE
  - freebsd-vm-groupMM223KK.vh...
  - Empty
- Controller: SATA
  - VirtualDisk.vdi

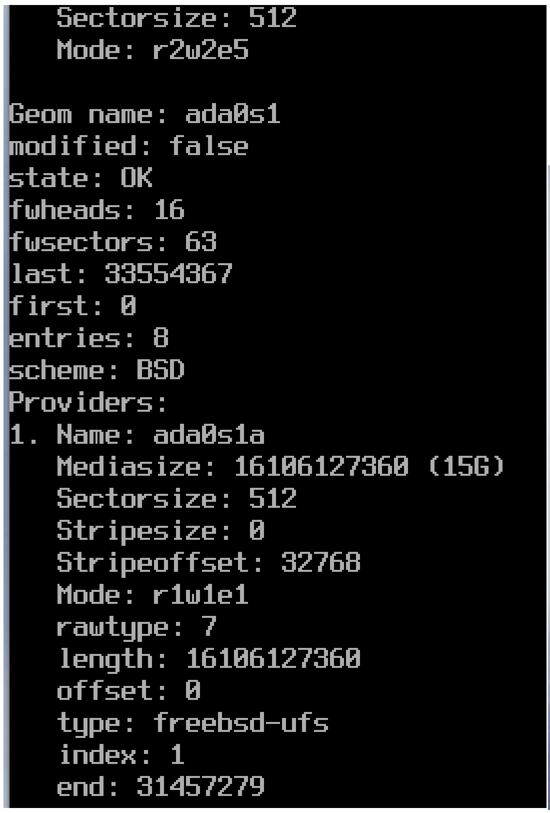
Attributes

Name: SATA
Type: AHCI
Port Count: 1
☐ Use Host I/O Cache

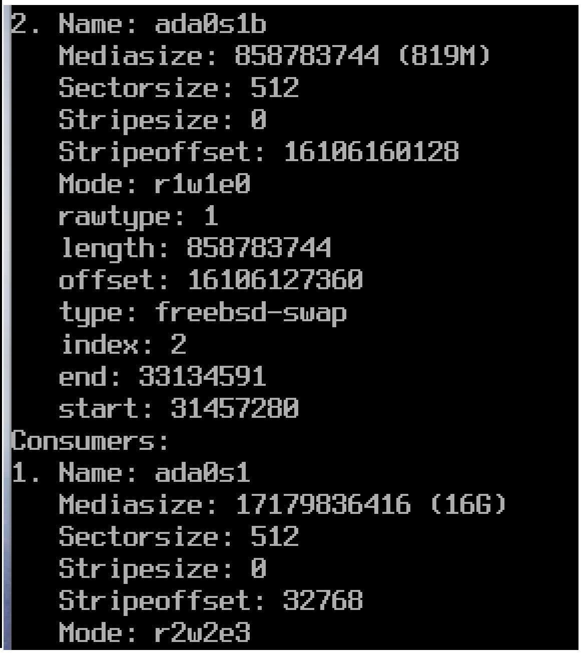
I added a SATA controller which I named SATA. Then I created a new Virtual Hard Disk named “**VirtualDisk**” with the size of 200 MB. And added it to the SATA-controller.

Sami Mwanje
mm223kk@student.lnu.se

1.2:



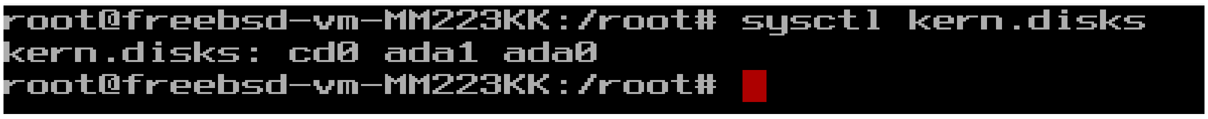
```
   Sectorsize: 512
   Mode: r2w2e5

Geom name: ada0s1
modified: false
state: OK
fwheads: 16
fwsectors: 63
last: 33554367
first: 0
entries: 8
scheme: BSD
Providers:
1. Name: ada0s1a
   Mediasize: 16106127360 (15G)
   Sectorsize: 512
   Stripesize: 0
   Stripeoffset: 32768
   Mode: r1w1e1
   rawtype: 7
   length: 16106127360
   offset: 0
   type: freebsd-ufs
   index: 1
   end: 31457279
```



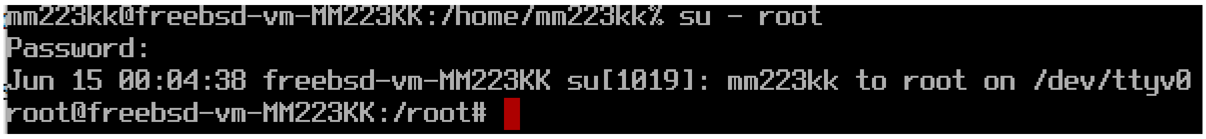
```
2. Name: ada0s1b
   Mediasize: 858783744 (819M)
   Sectorsize: 512
   Stripesize: 0
   Stripeoffset: 16106160128
   Mode: r1w1e0
   rawtype: 1
   length: 858783744
   offset: 16106127360
   type: freebsd-swap
   index: 2
   end: 33134591
   start: 31457280
Consumers:
1. Name: ada0s1
   Mediasize: 17179836416 (16G)
   Sectorsize: 512
   Stripesize: 0
   Stripeoffset: 32768
   Mode: r2w2e3
```

Typing "**gpart list**" shows a list of the partitions of the main device. The BSD name for the first device is "**ada0s1X**" ada0s1 is the given device name by the system the number/letter after that split the device into partitions. So the first partitions has a size of 15 GB and is named "**ada0s1a"** the second with the size of 819 MB is named "**ada0s1b"** . The first screenshot shows the name of the device and some other information. On line 1. We can see the first partition which is named "**ada0s1a".** The seconds screenshot shows the name of the second partition "**ada0s1b".** After the line "Consumer: " we see the name and the whole size of the full disk image.

1.3:



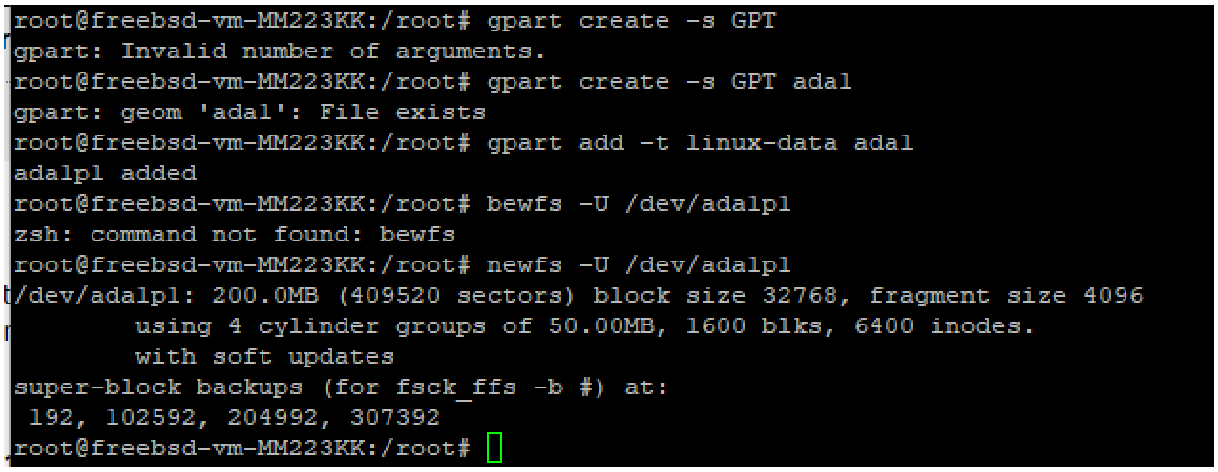
```
root@freebsd-vm-MM223KK:/root# sysctl kern.disks
kern.disks: cd0 ada1 ada0
root@freebsd-vm-MM223KK:/root# 
```



```
mm223kk@freebsd-vm-MM223KK:/home/mm223kk% su - root
Password:
Jun 15 00:04:38 freebsd-vm-MM223KK su[1019]: mm223kk to root on /dev/ttyv0
root@freebsd-vm-MM223KK:/root# 
```

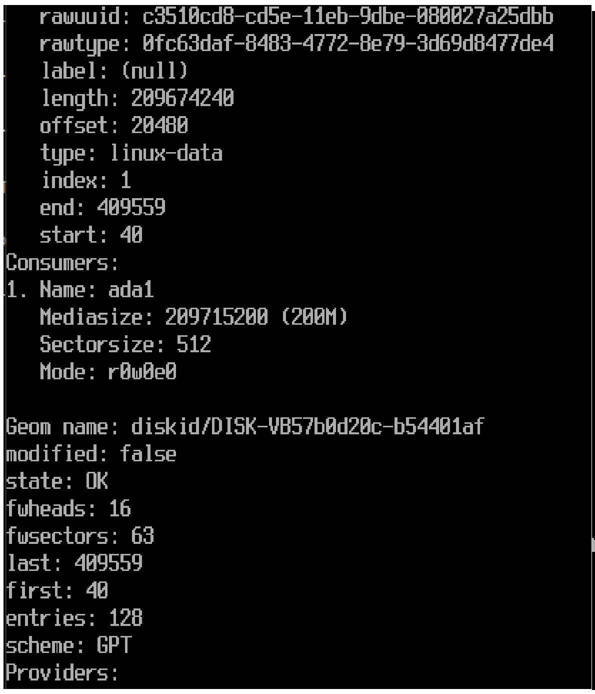
Using "**sysctl kern.disks**" we display the names of all available devices. We can see **cd0**, **ada1** and **ada0**. Since the **ada0** was the main hard drive and **cd0** is a cd hardware it can only then make sense that "**ada1"** is the newly installed virtual disk device. I made sure that I was inside the root account in order to avoid errors on coming tasks.

2.4:

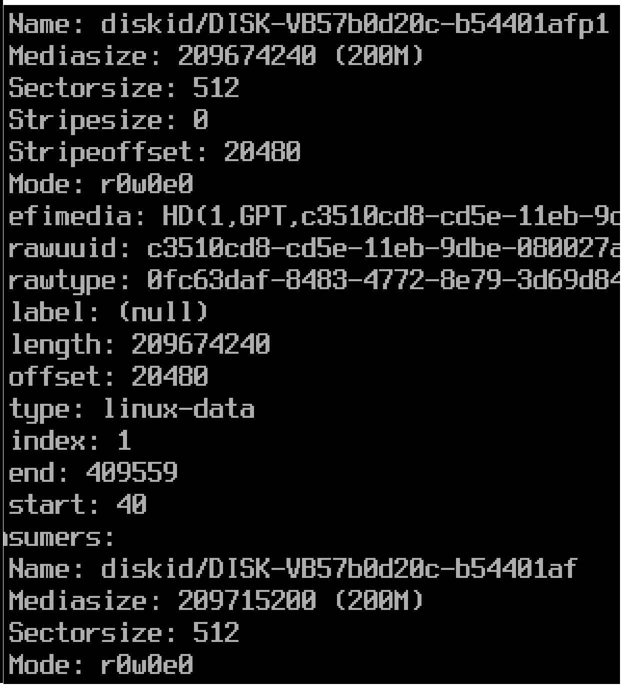


```
root@freebsd-vm-MM223KK:/root# gpart create -s GPT
gpart: Invalid number of arguments.
root@freebsd-vm-MM223KK:/root# gpart create -s GPT ada1
gpart: geom 'ada1': File exists
root@freebsd-vm-MM223KK:/root# gpart add -t linux-data ada1
ada1p1 added
root@freebsd-vm-MM223KK:/root# bewfs -U /dev/ada1p1
zsh: command not found: bewfs
root@freebsd-vm-MM223KK:/root# newfs -U /dev/ada1p1
/dev/ada1p1: 200.0MB (409520 sectors) block size 32768, fragment size 4096
        using 4 cylinder groups of 50.00MB, 1600 blks, 6400 inodes.
        with soft updates
super-block backups (for fsck_ffs -b #) at:
 192, 102592, 204992, 307392
root@freebsd-vm-MM223KK:/root#
```

Gpart create -s GPT ada1 and **gpart -t linux-data ada1** were used to create a partition scheme.



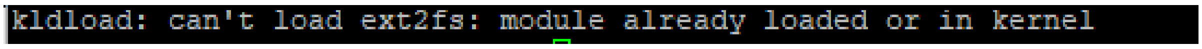
```
   rawuuid: c3510cd8-cd5e-11eb-9dbe-080027a25dbb
   rawtype: 0fc63daf-8483-4772-8e79-3d69d8477de4
   label: (null)
   length: 209674240
   offset: 20480
   type: linux-data
   index: 1
   end: 409559
   start: 40
Consumers:
1. Name: ada1
   Mediasize: 209715200 (200M)
   Sectorsize: 512
   Mode: r0w0e0

Geom name: diskid/DISK-VB57b0d20c-b54401af
modified: false
state: OK
fwheads: 16
fwsectors: 63
last: 409559
first: 40
entries: 128
scheme: GPT
Providers:
```



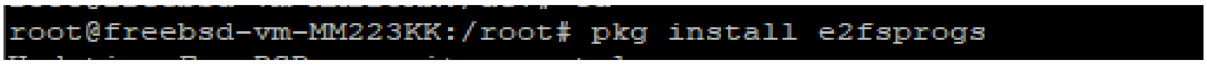
```
Name: diskid/DISK-VB57b0d20c-b54401afp1
Mediasize: 209674240 (200M)
Sectorsize: 512
Stripesize: 0
Stripeoffset: 20480
Mode: r0w0e0
efimedia: HD(1,GPT,c3510cd8-cd5e-11eb-9d
rawuuid: c3510cd8-cd5e-11eb-9dbe-080027a
rawtype: 0fc63daf-8483-4772-8e79-3d69d84
label: (null)
length: 209674240
offset: 20480
type: linux-data
index: 1
end: 409559
start: 40
sumers:
Name: diskid/DISK-VB57b0d20c-b54401af
Mediasize: 209715200 (200M)
Sectorsize: 512
Mode: r0w0e0
```

2.5:

Typing gpart list now shows the new 200 MB disk.

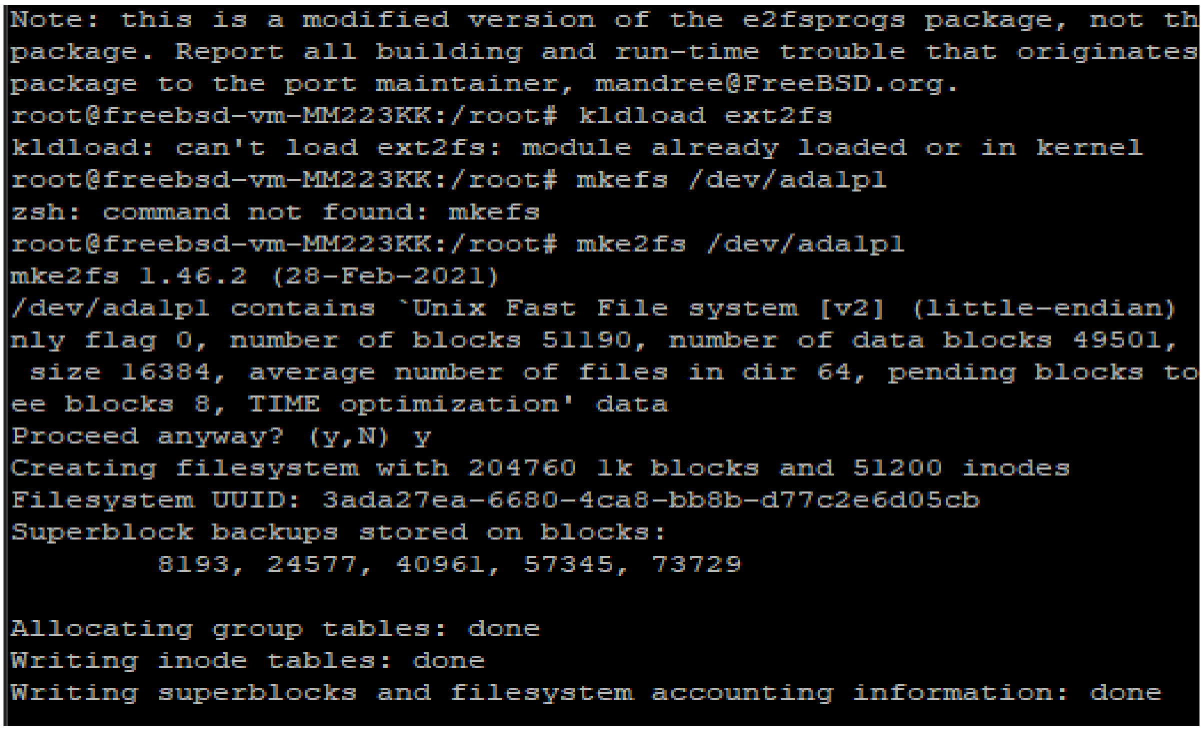


```
kldload: can't load ext2fs: module already loaded or in kernel
```



```
root@freebsd-vm-MM223KK:/root# pkg install e2fsprogs
```

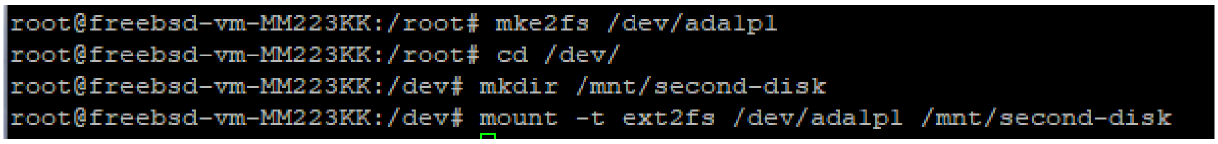
Installed the e2fsprogs package using “**pkg install e2fsprogs**”. The kernel model was already loaded.



```
Note: this is a modified version of the e2fsprogs package, not th
package. Report all building and run-time trouble that originates
package to the port maintainer, mandree@FreeBSD.org.
root@freebsd-vm-MM223KK:/root# kldload ext2fs
kldload: can't load ext2fs: module already loaded or in kernel
root@freebsd-vm-MM223KK:/root# mkefs /dev/ada1p1
zsh: command not found: mkefs
root@freebsd-vm-MM223KK:/root# mke2fs /dev/ada1p1
mke2fs 1.46.2 (28-Feb-2021)
/dev/ada1p1 contains `Unix Fast File system [v2] (little-endian)
nly flag 0, number of blocks 51190, number of data blocks 49501,
 size 16384, average number of files in dir 64, pending blocks to
ee blocks 8, TIME optimization' data
Proceed anyway? (y,N) y
Creating filesystem with 204760 1k blocks and 51200 inodes
Filesystem UUID: 3ada27ea-6680-4ca8-bb8b-d77c2e6d05cb
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

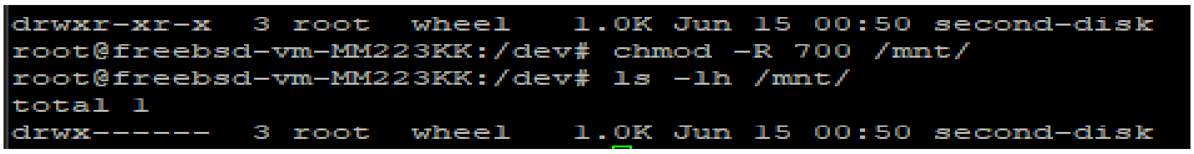
The new EXT2 filesystem is installed using "**mke2fs /dev/ada1p1**" the seconds line is the location. I had in someway already installed a filesystem there by I proceeded, and it seemed like it got wiped away.

2.6:

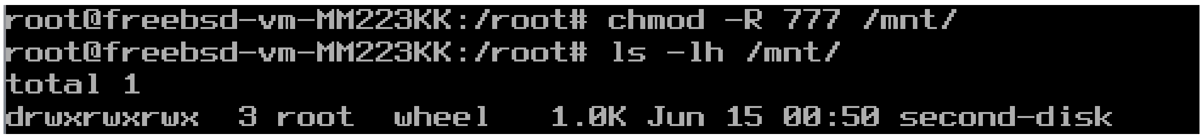


```
root@freebsd-vm-MM223KK:/root# mke2fs /dev/ada1p1
root@freebsd-vm-MM223KK:/root# cd /dev/
root@freebsd-vm-MM223KK:/dev# mkdir /mnt/second-disk
root@freebsd-vm-MM223KK:/dev# mount -t ext2fs /dev/ada1p1 /mnt/second-disk
```

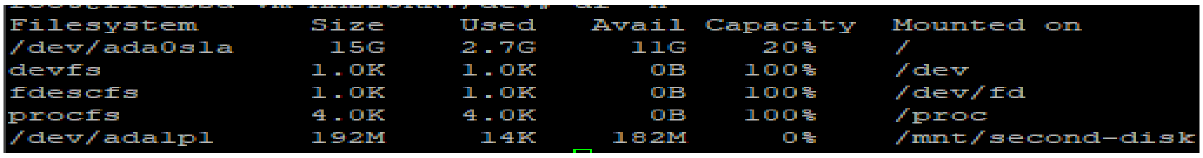
Here I created a directory "**/mnt/second-disk**" using "**mkdir**". After that I mounted the new partition to this device using "**mount -t ext2fs /dev/ad1p1 /mnt/second-disk".** This seemed to have worked without any error.



```
drwxr-xr-x  3 root  wheel   1.0K Jun 15 00:50 second-disk
root@freebsd-vm-MM223KK:/dev# chmod -R 700 /mnt/
root@freebsd-vm-MM223KK:/dev# ls -lh /mnt/
total 1
drwx------  3 root  wheel   1.0K Jun 15 00:50 second-disk
```



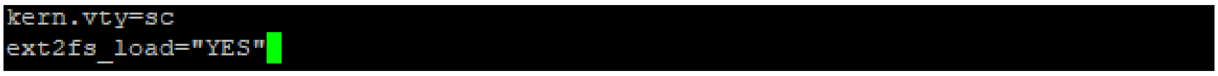
```
root@freebsd-vm-MM223KK:/root# chmod -R 777 /mnt/
root@freebsd-vm-MM223KK:/root# ls -lh /mnt/
total 1
drwxrwxrwx  3 root  wheel   1.0K Jun 15 00:50 second-disk
```



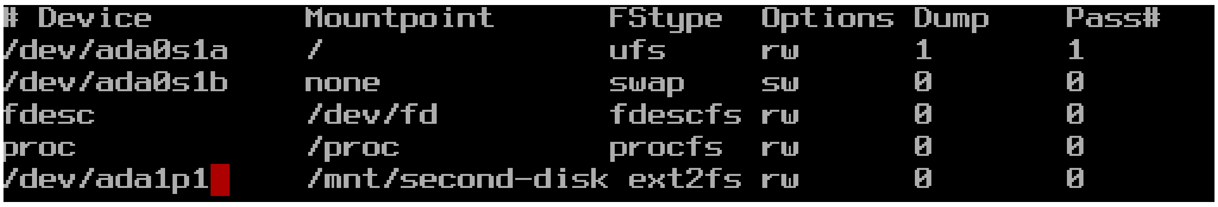
```
Filesystem        Size     Used    Avail  Capacity  Mounted on
/dev/ada0s1a       15G     2.7G      11G     20%    /
devfs             1.0K     1.0K       0B    100%    /dev
fdescfs           1.0K     1.0K       0B    100%    /dev/fd
procfs            4.0K     4.0K       0B    100%    /proc
/dev/ada1p1       192M      14K     182M      0%    /mnt/second-disk
```

Using “**ls -lh /mnt/”** showed the current permissions of the /**mnt**/ folder. By using “**chmod -R 777”** the permissions were edited so everyone could read write and edit the **/mnt/** folder. “**df -h**” is later used to confirm that everything is working, which can be seen.

2.7:



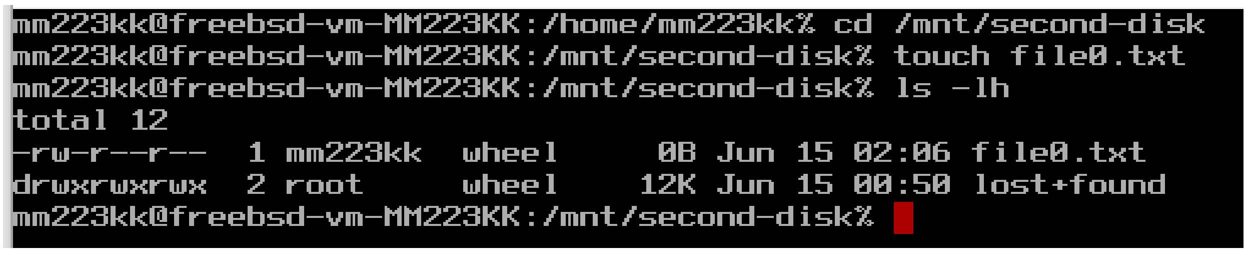
```
kern.vty=sc
ext2fs_load="YES"
```



```
# Device          Mountpoint       FStype   Options Dump     Pass#
/dev/ada0s1a      /                ufs      rw      1        1
/dev/ada0s1b      none             swap     sw      0        0
fdesc             /dev/fd          fdescfs  rw      0        0
proc              /proc            procfs   rw      0        0
/dev/ada1p1       /mnt/second-disk ext2fs   rw      0        0
```

In **“ee /boot/loader.conf** “ the line “**ext2fs_load="YES"**” was added, and in “**/etc/fstab**” the line **“/dev/ada1p1   /mnt/second-disk ext2fs   rw   0   0**” was added. This is for the system to automatically mount the file system in /mnt/second disk. After a reboot the system started with an error got fixed by a little edit in the **fstab**.

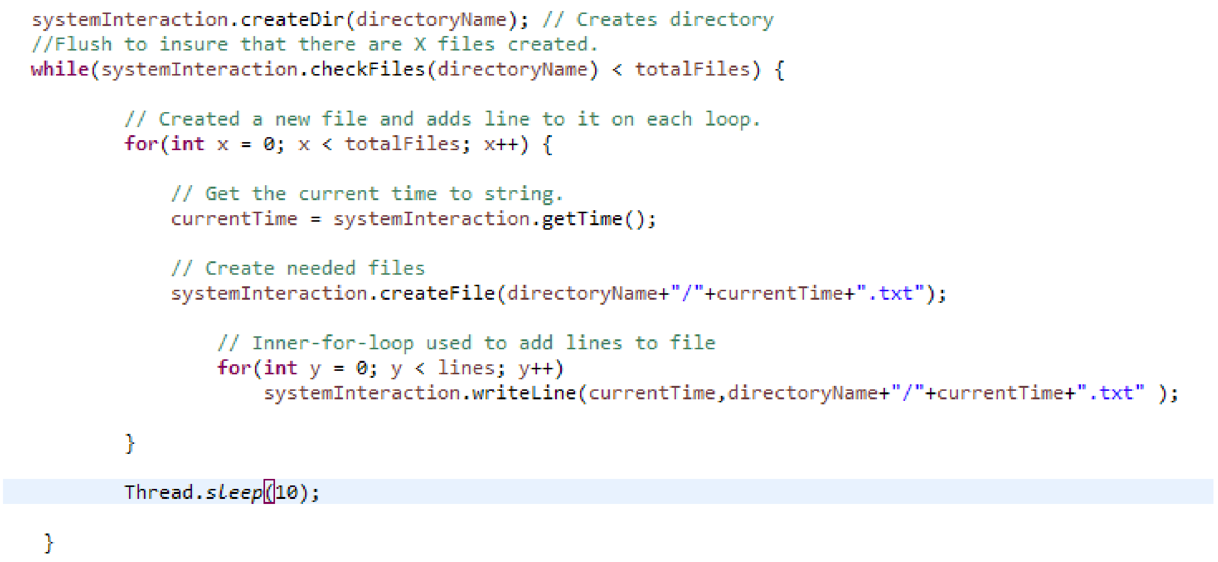
2.8:



```
mm223kk@freebsd-vm-MM223KK:/home/mm223kk% cd /mnt/second-disk
mm223kk@freebsd-vm-MM223KK:/mnt/second-disk% touch file0.txt
mm223kk@freebsd-vm-MM223KK:/mnt/second-disk% ls -lh
total 12
-rw-r--r--   1 mm223kk   wheel      0B Jun 15 02:06 file0.txt
drwxrwxrwx   2 root      wheel     12K Jun 15 00:50 lost+found
mm223kk@freebsd-vm-MM223KK:/mnt/second-disk%
```

Finale I changed to the user “mm223kk” and navigated to the folder “**mnt/second-disk**” with an error which I fixed by changing the permission to “**chmod -R 777”.** Now I had access to the folder without any error. In the folder I created a file using “**touch**” and checked the permissions using “**ls -lh”.** Everything then seemed okey.

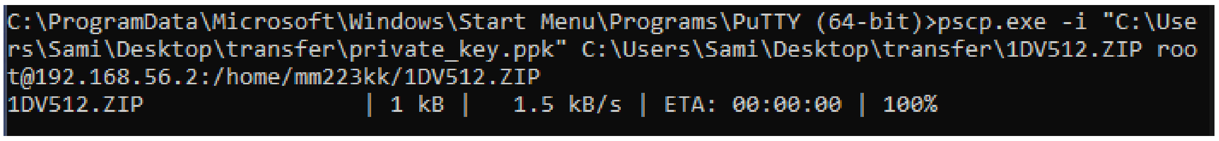
TASK 3:



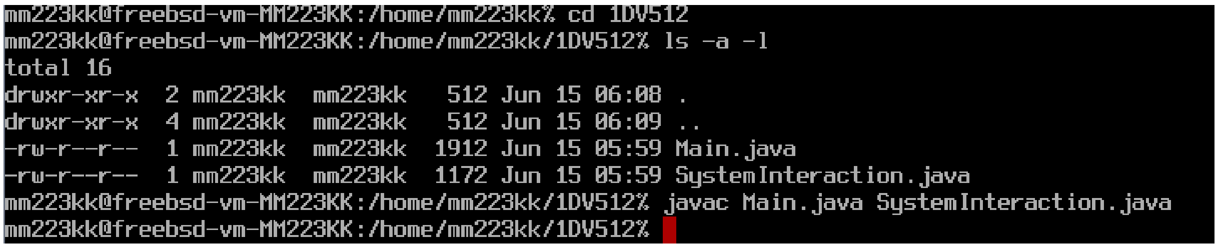
```
systemInteraction.createDir(directoryName); // Creates directory
//Flush to insure that there are X files created.
while(systemInteraction.checkFiles(directoryName) < totalFiles) {

        // Created a new file and adds line to it on each loop.
        for(int x = 0; x < totalFiles; x++) {

            // Get the current time to string.
            currentTime = systemInteraction.getTime();

            // Create needed files
            systemInteraction.createFile(directoryName+"/"+currentTime+".txt");

                // Inner-for-loop used to add lines to file
                for(int y = 0; y < lines; y++)
                    systemInteraction.writeLine(currentTime,directoryName+"/"+currentTime+".txt" );

        }

        Thread.sleep(10);

 }
```

Creating the code for the program. When it comes to the flush, I used a code that checked if there are X files created. My program always made perfect 500 files with 10000 lines. So, this may solve missing line or files, but once again it may depend on the running system itself. This time I used a **Main** and a “**SystemInteraction**” class.

3.2:



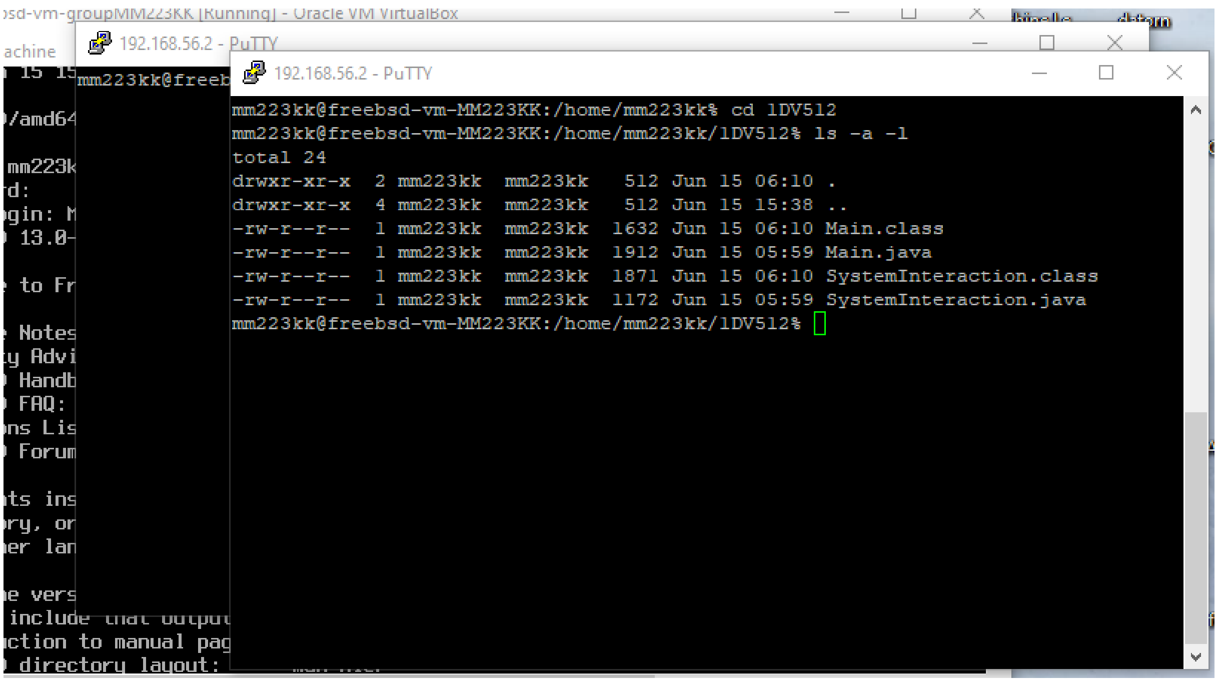
```
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\PuTTY (64-bit)>pscp.exe -i "C:\Use
rs\Sami\Desktop\transfer\private_key.ppk" C:\Users\Sami\Desktop\transfer\1DV512.ZIP roo
t@192.168.56.2:/home/mm223kk/1DV512.ZIP
1DV512.ZIP                | 1 kB |   1.5 kB/s | ETA: 00:00:00 | 100%
```



```
mm223kk@freebsd-vm-MM223KK:/home/mm223kk% cd 1DV512
mm223kk@freebsd-vm-MM223KK:/home/mm223kk/1DV512% ls -a -l
total 16
drwxr-xr-x  2 mm223kk  mm223kk   512 Jun 15 06:08 .
drwxr-xr-x  4 mm223kk  mm223kk   512 Jun 15 06:09 ..
-rw-r--r--  1 mm223kk  mm223kk  1912 Jun 15 05:59 Main.java
-rw-r--r--  1 mm223kk  mm223kk  1172 Jun 15 05:59 SystemInteraction.java
mm223kk@freebsd-vm-MM223KK:/home/mm223kk/1DV512% javac Main.java SystemInteraction.java
mm223kk@freebsd-vm-MM223KK:/home/mm223kk/1DV512%
```

Transferred the files to the VM once again using **PuTTy** and **pscp.exe.** The transformation went smoothly. Compiling the program with “**javac”** also went well. I am not sure where the file should be located yet, but I suppose that the main location for a java program is where the program is called from using “**java**”. So, the current path may not really matter.

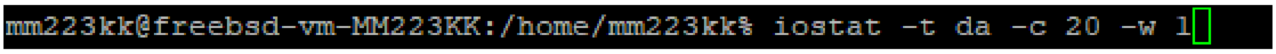
3.3:



```
mm223kk@freebsd-vm-MM223KK:/home/mm223kk% cd 1DV512
mm223kk@freebsd-vm-MM223KK:/home/mm223kk/1DV512% ls -a -l
total 24
drwxr-xr-x  2 mm223kk  mm223kk   512 Jun 15 06:10 .
drwxr-xr-x  4 mm223kk  mm223kk   512 Jun 15 15:38 ..
-rw-r--r--  1 mm223kk  mm223kk  1632 Jun 15 06:10 Main.class
-rw-r--r--  1 mm223kk  mm223kk  1912 Jun 15 05:59 Main.java
-rw-r--r--  1 mm223kk  mm223kk  1871 Jun 15 06:10 SystemInteraction.class
-rw-r--r--  1 mm223kk  mm223kk  1172 Jun 15 05:59 SystemInteraction.java
mm223kk@freebsd-vm-MM223KK:/home/mm223kk/1DV512%
```

Preparing to run program from several shell window using PuTTy and VirtualBox.

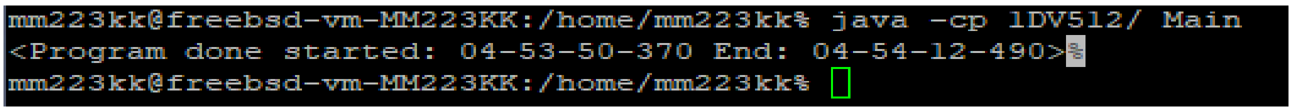
3.4



```
mm223kk@freebsd-vm-MM223KK:/home/mm223kk% iostat -t da -c 20 -w 1
```

Launching the command "**iostat -t da -c 20 -w 1**" in one of the shells.

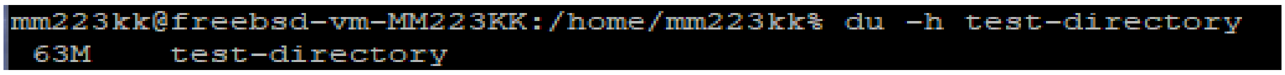
3.5



```
mm223kk@freebsd-vm-MM223KK:/home/mm223kk% java -cp 1DV512/ Main
<Program done started: 04-53-50-370 End: 04-54-12-490>%
mm223kk@freebsd-vm-MM223KK:/home/mm223kk%
```

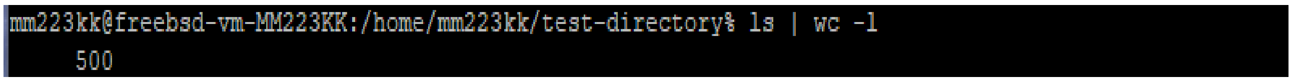
Launching the java program from another shell/window in the **"/home/mm223kk/"** directory. The program is launched using the command "**java -cp 1DV512/ Main"**

The lunch went well, and I also added an out print of the run time after the lunch. The out print was "<**Program done started: 04-53-50-370 End: 04-54-12-490**>" This is in some way much faster than running the program on the host which took minutes. Probably because my java files are located on an old **Mechanical Hard disk** environment.



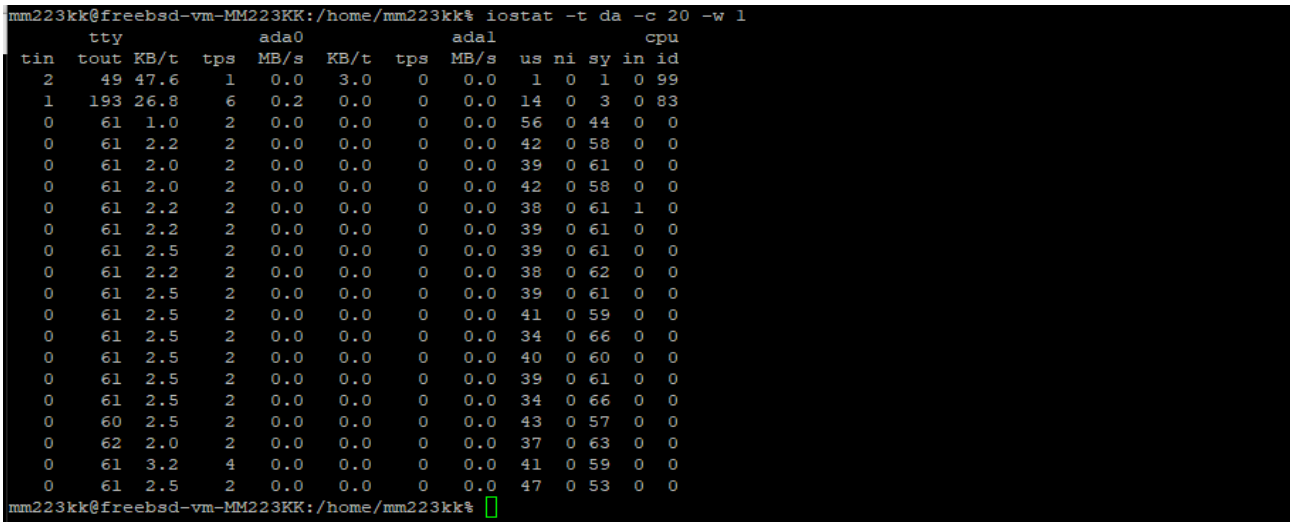
```
mm223kk@freebsd-vm-MM223KK:/home/mm223kk% du -h test-directory
 63M    test-directory
```

Using "**du -h test-directory**" I got the size of the directory "**63 MB**", which I can confirm with the size that I got on the host system when I launched the program.



```
mm223kk@freebsd-vm-MM223KK:/home/mm223kk/test-directory% ls | wc -l
     500
```

I can also confirm that all the expected 500 txt-files were created successfully in the test-directory folder using the command “**ls | wc -l**”.



```
mm223kk@freebsd-vm-MM223KK:/home/mm223kk% iostat -t da -c 20 -w 1
       tty           ada0             ada1             cpu
 tin  tout KB/t  tps  MB/s  KB/t  tps  MB/s  us ni sy in id
   2    49 47.6    1   0.0   3.0    0   0.0   1  0  1  0 99
   1   193 26.8    6   0.2   0.0    0   0.0  14  0  3  0 83
   0    61  1.0    2   0.0   0.0    0   0.0  56  0 44  0  0
   0    61  2.2    2   0.0   0.0    0   0.0  42  0 58  0  0
   0    61  2.0    2   0.0   0.0    0   0.0  39  0 61  0  0
   0    61  2.0    2   0.0   0.0    0   0.0  42  0 58  0  0
   0    61  2.2    2   0.0   0.0    0   0.0  38  0 61  1  0
   0    61  2.2    2   0.0   0.0    0   0.0  39  0 61  0  0
   0    61  2.5    2   0.0   0.0    0   0.0  39  0 61  0  0
   0    61  2.2    2   0.0   0.0    0   0.0  38  0 62  0  0
   0    61  2.5    2   0.0   0.0    0   0.0  39  0 61  0  0
   0    61  2.5    2   0.0   0.0    0   0.0  41  0 59  0  0
   0    61  2.5    2   0.0   0.0    0   0.0  34  0 66  0  0
   0    61  2.5    2   0.0   0.0    0   0.0  40  0 60  0  0
   0    61  2.5    2   0.0   0.0    0   0.0  39  0 61  0  0
   0    61  2.5    2   0.0   0.0    0   0.0  34  0 66  0  0
   0    60  2.5    2   0.0   0.0    0   0.0  43  0 57  0  0
   0    62  2.0    2   0.0   0.0    0   0.0  37  0 63  0  0
   0    61  3.2    4   0.0   0.0    0   0.0  41  0 59  0  0
   0    61  2.5    2   0.0   0.0    0   0.0  47  0 53  0  0
mm223kk@freebsd-vm-MM223KK:/home/mm223kk%
```

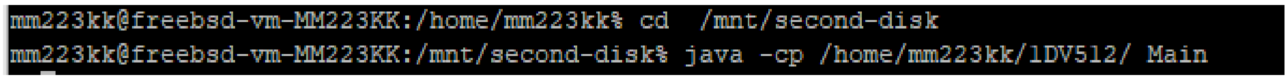
Launching the command “**iostat -t da -c 20 -w 1**” on the other shell/terminal displayed an output of some statics while the program was running as displayed above. “**-t**” specifies which type of device to display, so the “**da**” means “**direct access devices**”. “**-c**” indicates how many times the display should be repeated, here the value is “**20**”. The “**w**” tells the “**iostat”** to wait “**Xs**” after each display, here the time is “**1**” s. The **KB/t** means kilobytes per transfer, **tps** transfers per second and **MB/s** megabytes per second. Tout and tin are characters written and read from the terminal.

Since the program was running on ada0 we can take a closer look on ada0. The **KB** transferred per second starts with **47** and **26.8 KB/t** after that it stays on an average of **2.5 KB/t** until the end. There is a little ripple of **3.2 KB/t** that may indicate that the java program is closing.

When it comes to the **tps** transfers per second, the **6** at the beginning probably indicates the transfers made when the java program is launched. The **4** at the end may indicate the transfers happening when the java program finishes its executions. The average **tps** is **2** for the majority of the run. Other measures had no really crucial meaning on this output. **MB/s** and **KB/s** stayed on 0 for most of the run.

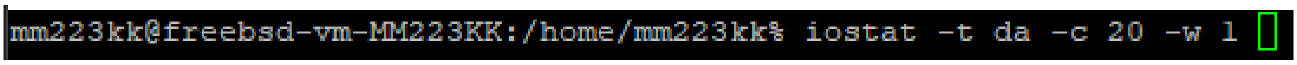
Looking at the CPU performance **us** and **sy** have values that differ from the others. **Us** stand for % of CPU time in user mode and **sy** means % of CPU time in system mode. User mode is on about **40 %** and system mode is on **60 %** during the run. Another notation is **id** which means the % of time the CPU spends sleeping and not working at all. We can see that on the two lines in the beginning. Judging by the results and that my “**Thread.sleep(10)”** is called at the end of the execution I can confirm that this had no affect on making the **CPU** sleep. The behavior may indicate that some executions are scheduled to execute though the java program itself is done executing.

3.6



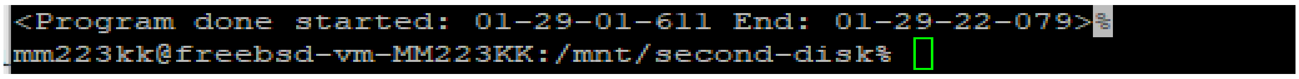
```
mm223kk@freebsd-vm-MM223KK:/home/mm223kk% cd  /mnt/second-disk
mm223kk@freebsd-vm-MM223KK:/mnt/second-disk% java -cp /home/mm223kk/1DV512/ Main
```

Switching to the “**mnt/second-disk**”-directory using “**cd /mnt/second-disk**”, also preparing for the launch of the java program using **”java -cp /home/mm223kk/1DV512   Main” .** I have learned that only main needs to be launched with the command as long as it is somehow connected to the other classes.

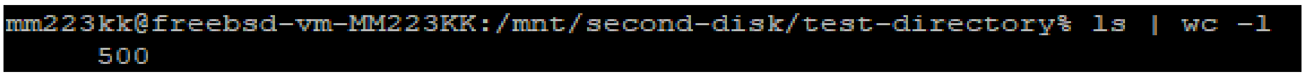


```
mm223kk@freebsd-vm-MM223KK:/home/mm223kk% iostat -t da -c 20 -w 1
```

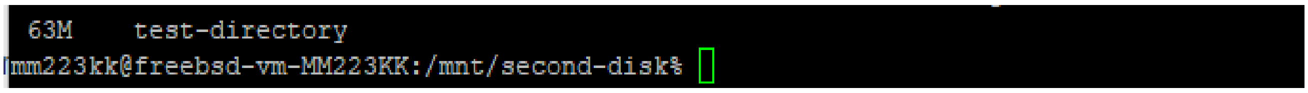
Here I will once again launch the static display using “**iostat -t da -c 20 -w 1**”.



```
<Program done started: 01-29-01-611 End: 01-29-22-079>%
mm223kk@freebsd-vm-MM223KK:/mnt/second-disk%
```

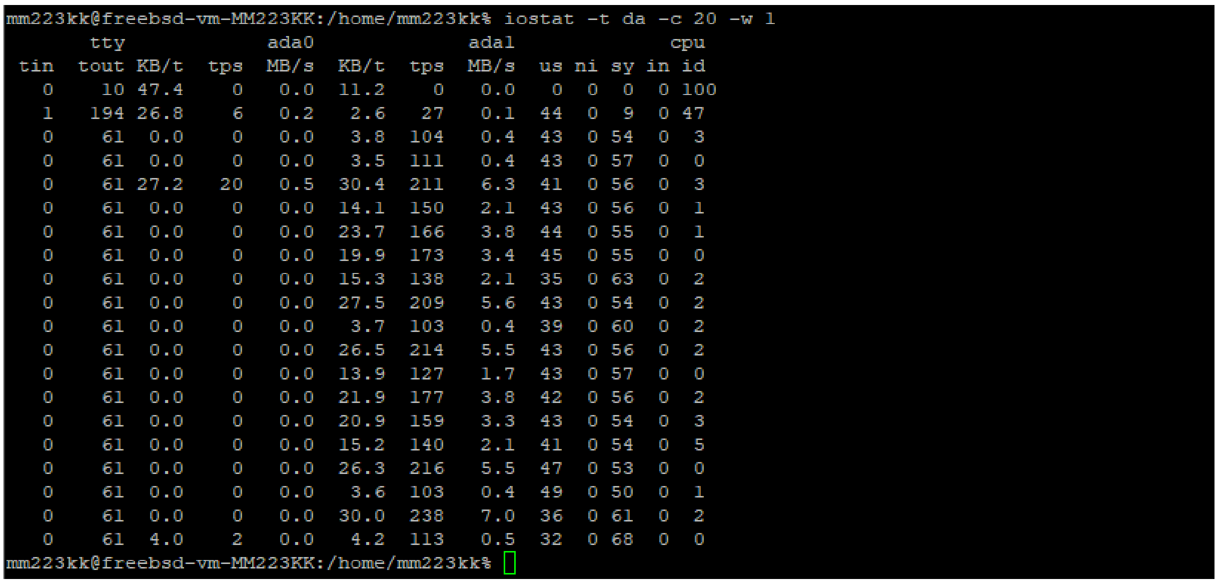


```
mm223kk@freebsd-vm-MM223KK:/mnt/second-disk/test-directory% ls | wc -l
     500
```



```
 63M    test-directory
mm223kk@freebsd-vm-MM223KK:/mnt/second-disk%
```

Once again, the launch went smoothly. Launching “**ls |wc -1**” once again indicates that all the 500 files were created as planned. “**du -h test-directory**” shoed 63 MB which is the same size as the previously monitored **UFS** file system.



```
mm223kk@freebsd-vm-MM223KK:/home/mm223kk% iostat -t da -c 20 -w 1
       tty            ada0              ada1             cpu
 tin  tout KB/t  tps  MB/s  KB/t  tps  MB/s  us ni sy in id
   0    10 47.4    0   0.0  11.2    0   0.0   0  0  0  0 100
   1   194 26.8    6   0.2   2.6   27   0.1  44  0  9  0 47
   0    61  0.0    0   0.0   3.8  104   0.4  43  0 54  0  3
   0    61  0.0    0   0.0   3.5  111   0.4  43  0 57  0  0
   0    61 27.2   20   0.5  30.4  211   6.3  41  0 56  0  3
   0    61  0.0    0   0.0  14.1  150   2.1  43  0 56  0  1
   0    61  0.0    0   0.0  23.7  166   3.8  44  0 55  0  1
   0    61  0.0    0   0.0  19.9  173   3.4  45  0 55  0  0
   0    61  0.0    0   0.0  15.3  138   2.1  35  0 63  0  2
   0    61  0.0    0   0.0  27.5  209   5.6  43  0 54  0  2
   0    61  0.0    0   0.0   3.7  103   0.4  39  0 60  0  2
   0    61  0.0    0   0.0  26.5  214   5.5  43  0 56  0  2
   0    61  0.0    0   0.0  13.9  127   1.7  43  0 57  0  0
   0    61  0.0    0   0.0  21.9  177   3.8  42  0 56  0  2
   0    61  0.0    0   0.0  20.9  159   3.3  43  0 54  0  3
   0    61  0.0    0   0.0  15.2  140   2.1  41  0 54  0  5
   0    61  0.0    0   0.0  26.3  216   5.5  47  0 53  0  0
   0    61  0.0    0   0.0   3.6  103   0.4  49  0 50  0  1
   0    61  0.0    0   0.0  30.0  238   7.0  36  0 61  0  2
   0    61  4.0    2   0.0   4.2  113   0.5  32  0 68  0  0
mm223kk@freebsd-vm-MM223KK:/home/mm223kk%
```

Looking at the output when the program was writing on the **EXT2** file system we got some interesting results. We expected ada0 to be idle, but this is not the case. The statics showed values on **KB/t kilobytes per transfer,** which may mean that the files are created on the UFS file system and then transferred to the EXT2 filesystem.

Taking a look under **ada1** which is where the program was called from, we can first find **tps** (**transfers per second**) values that are much larger than those we earlier got on the UFS file system. While these values had an average of 2 on the UFS, they now have an average of **100-200**. This may indicate the transfers that were made from the programs original directory to the EXT2 file system directory. Even MB/s displayed some values compared to the UFS file system.

Observing the CPU performance while the program was launching the EXT2 file system had almost the same stress on the CPU as the UFS. There may be a lower percentage when it came to system mode percentage. I cannot really tell. We can also see under **id** that the CPU went idle several times while running the program from the EXT2 file system. I cannot really determine if the "**Thread.sleep(10)**" call really made any affect on the **id** values. But this may indicate the number **5** that can be seen in the end.

This group assignment was a bit more time consuming then the first one, especially the first task which I first had to understand what a pipe really is and what I was meant to do with it in the java program. After understanding this part, the assignment went on pretty smoothly. I am though not very happy with that I could not get the pipe to unblock itself while it was waiting for another command from the other end. I tried several codes but most of them led to more advanced algorithm which may affect the task itself. Unblocking the java program manually with "**CTRL + C**" was the solution that I used.

---

[i] https://blog.desdelinux.net/sv/sl%C3%A4ppt-freebsd-9-0/

Sami Mwanje
mm223kk@student.lnu.se