

2017

Arnoud Bevers & Bram Hoven – Groep 1

Fontys HBO-ICT & Software Engineering

1-1-2017

JSF31 Practica Documentatie

Inhoudsopgave

[Week 1 3](#_Toc475297844)

# Week 1

## Opdrachten

1. Voer de opdrachten van paragrafen

- 8.8 practice: working with directories

- 9.9 practice: working with files

- 10.7 practice: file contents

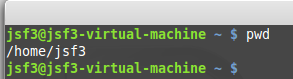
- 11.10 practice: file system tree

uit LinuxFun uit en schrijf op welke commando's je precies gebruikt.

### 8.8 practice: working with directories

1. Display your current directory.

=> pwd



2. Change to the /etc directory.

=> cd /etc



3. Now change to your home directory using only three key presses.

=> cd



4. Change to the /boot/grub directory using only eleven key presses.

=> cd /boot/[TAB]



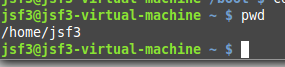
5. Go to the parent directory of the current directory.

=> cd ..



6. Go to the root directory.

=> cd ~



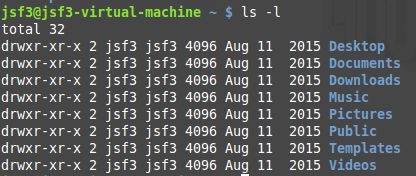
7. List the contents of the root directory.

=> ls



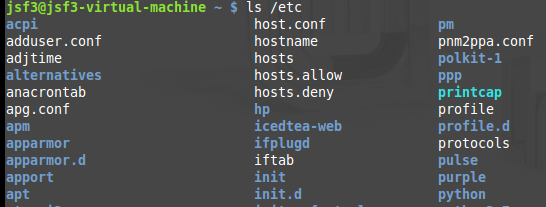
8. List a long listing of the root directory.

=> ls -l



9. Stay where you are, and list the contents of /etc.

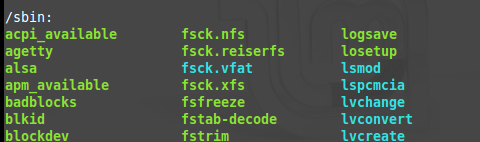
=> ls /etc



10. Stay where you are, and list the contents of /bin and /sbin.

=> ls /bin /sbin





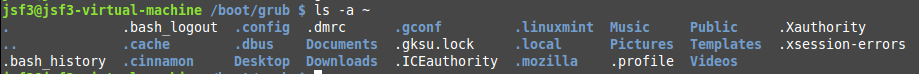
11. Stay where you are, and list the contents of ~.

=> ls ~



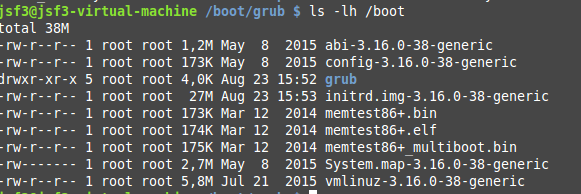
12. List all the files (including hidden files) in your home directory.

=> ls -a ~



13. List the files in /boot in a human readable format.

=> ls -lh /boot



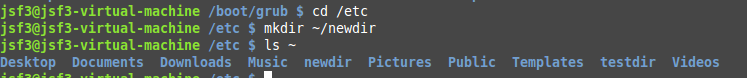
14. Create a directory testdir in your home directory.

=> mkdir ~/testdir

15. Change to the /etc directory, stay here and create a directory newdir in your home

directory.

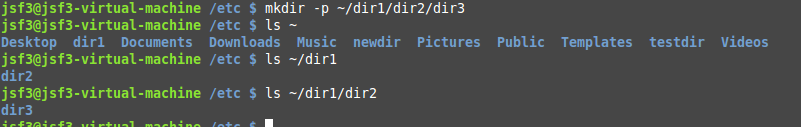
=> mkdir ~/newdir



16. Create in one command the directories ~/dir1/dir2/dir3 (dir3 is a subdirectory from dir2,

and dir2 is a subdirectory from dir1 ).

=> mkdir -p ~/dir1/dir2/dir3



17. Remove the directory testdir.

=> rmdir ~/testdir



18. If time permits (or if you are waiting for other students to finish this practice), use and

understand **pushd** and **popd**. Use the man page of **bash** to find information about these

commands.

Zorg dat je alle stappen begrijpt en herlees hoofdstukken indien nodig.

### 9.9 practice: working with files

1. List the files in the /bin directory

=> ls /bin

2. Display the type of file of /bin/cat, /etc/passwd and /usr/bin/passwd.

=> file /bin/cat /etc/passwd /usr/bin/psswd

3a. Download wolf.jpg and LinuxFun.pdf from

http://linux-training.be (wget http://

linux-training.be/files/studentfiles/wolf.jpg and wget http://linux-training.be/files/books/

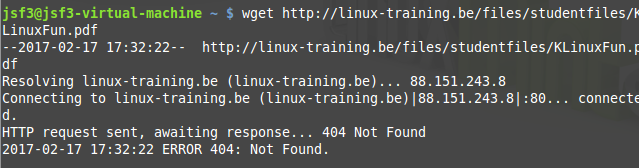
LinuxFun.pdf)

wget http://linux-training.be/files/studentfiles/wolf.jpg

wget http://linux-training.be/files/studentfiles/wolf.png

wget <http://linux-training.be/files/books/LinuxFun.pdf>

=> Werkt niet? 404: not found



3b. Display the type of file of wolf.jpg and LinuxFun.pdf -

3c. Rename wolf.jpg to wolf.pdf (use mv). -

3d. Display the type of file of wolf.pdf and LinuxFun.pdf. -

4. Create a directory ~/touched and enter it.

=> mkdir ~/touched

=> cd ~/touched

5. Create the files today.txt and yesterday.txt in touched.

=> touch today.txt yesterday.txts

6. Change the date on yesterday.txt to match yesterday's date.

=> touch -t 201702171736 yesterday.txt

7. Copy yesterday.txt to copy.yesterday.txt

=> cp yesterday.txt copy.yesterday.txt

8. Rename copy.yesterday.txt to kim

=> mv copy.yesterday.txt kim

9. Create a directory called ~/testbackup and copy all files from ~/touched into it.

=> mkdir ~/testbackup

=> cp -r ~/touched ~/testbackup

10. Use one command to remove the directory ~/testbackup and all files into it.

=> rm -rf ~/testbackup

11. Create a directory ~/etcbackup and copy all \*.conf files from /etc into it. Did you include

all subdirectories of /etc ?

=> cp -r /etc/\*.conf ~/etcbackup (-r zorgt ervoor dat alle subdirectories ook weggaan)

12. Use rename to rename all \*.conf files to \*.backup . (if you have more than one distro

available, try it on all!)

=> touch 1.conf 2.conf

=> rename 's/conf/backup/' \*.conf

### 10.7 practice: file contents

1. Display the first 12 lines of /etc/services.

=> head -12 /etc/services

2. Display the last line of /etc/passwd.

=> tail -1 /etc/passwd

3. Use cat to create a file named count.txt that looks like this:

One

Two

Three

Four

Five

=> cat > count.txt [ENTER]

One [ENTER]

Two [ENTER]

Three [ENTER]

Four [ENTER]

Five [ENTER] [CTRL+D]

4. Use cp to make a backup of this file to cnt.txt.

=> cp count.txt cnt.txt

5. Use cat to make a backup of this file to catcnt.txt.

=> cat count.txt > catcnt.txt

6. Display catcnt.txt, but with all lines in reverse order (the last line first).

=> tac catcnt.txt

7. Use more to display /etc/services.

=> more /etc/services

8. Display the readable character strings from the /usr/bin/passwd command.

=> strings /usr/bin/passwd

9. Use ls to find the biggest file in /etc.

=> ls -lrS /etc

10. Open two terminal windows (or tabs) and make sure you are in the same directory in

both. Type echo this is the first line > tailing.txt in the first terminal, then issue tail -f

tailing.txt in the second terminal. Now go back to the first terminal and type echo This is

another line >> tailing.txt (note the double >>), verify that the tail -f in the second terminal

shows both lines. Stop the tail -f with Ctrl-C.

=> WOW it works!

11. Use cat to create a file named tailing.txt that contains the contents of tailing.txt followed

by the contents of /etc/passwd.

=> cat /etc/passwd >> tailing.txt

12. Use cat to create a file named tailing.txt that contains the contents of tailing.txt preceded

by the contents of /etc/passwd.

=> mv tailing.txt tmp.txt

=> cat /etc/passwd tmp.txt > tailing.txt

### 11.10 practice: file system tree

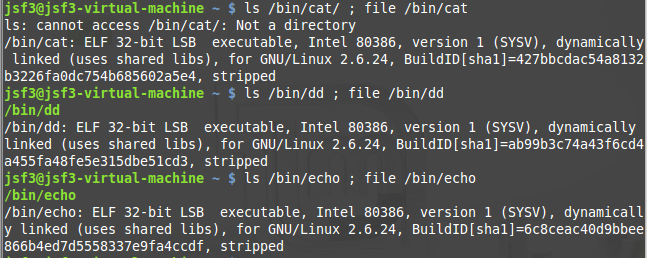
1. Does the file /bin/cat exist ? What about /bin/dd and /bin/echo. What is the type of these

files ?

=> ls /bin/cat ; file /bin/cat

=> ls /bin/dd ; file /bin/dd

=> ls /bin/echo ; file /bin/echo



2. What is the size of the Linux kernel file(s) (vmlinu\*) in /boot ?

=> ls -lh /boot/vm\* => size: 5.8M

3. Create a directory ~/test. Then issue the following commands:

- cd ~/test

- dd if=/dev/zero of=zeroes.txt count=1 bs=100

- od zeroes.txt

dd will copy one times (count=1) a block of size 100 bytes (bs=100) from the file /dev/zero

to ~/test/zeroes.txt. Can you describe the functionality of /dev/zero ?

=> mkdir ~/test

Geen idee…

4. Now issue the following command:

dd if=/dev/random of=random.txt count=1 bs=100 ; od random.txt

dd will copy one times (count=1) a block of size 100 bytes (bs=100) from the file /dev/

random to ~/test/random.txt. Can you describe the functionality of /dev/random ?

Random number generator?

5. Issue the following two commands, and look at the first character of each output line.

ls -l /dev/sd\* /dev/hd\*

ls -l /dev/tty\* /dev/input/mou\*

The first ls will show block(b) devices, the second ls shows character(c) devices. Can you

tell the difference between block and character devices?

Geen idee..?

6. Use cat to display /etc/hosts and /etc/resolv.conf. What is your idea about the purpose

of these files?

Hosts: Hostnames met de bijbehorende IP-adressen

Resolv.conf: IP-adressen van DNS-nameservers

7. Are there any files in /etc/skel/ ? Check also for hidden files.

=> ls -al /etc/skel/ => Yep, er zijn files, ook hidden files.

8. Display /proc/cpuinfo. On what architecture is your Linux running ?

=> head /proc/cpuinfo => Intel Core i5-4460; 3.20 GHz

9. Display /proc/interrupts. What is the size of this file ? Where is this file stored?

=> ls -lh /proc/interrupts => 0, deze file staat op de RAM

10. Can you enter the /root directory ? Are there (hidden) files ?

=> cd /root => Nee, is niet bereikbaar voor ‘standaard’ users

11. Are ifconfig, fdisk, parted, shutdown and grub-install present in /sbin ? Why are these

binaries in /sbin and not in /bin ?

=> ls -lh /sbin => sbin is voor administrators

12. Is /var/log a file or a directory ? What about /var/spool?

Beiden directories.

13. Open two command prompts (Ctrl-Shift-T in gnome-terminal) or terminals (Ctrl-Alt-F1,

Ctrl-Alt-F2, ...) and issue the who am i in both. Then try to echo a word from one terminal

to the other.

=> echo Hello > /dev/pts/0

14. Read the man page of random and explain the difference between /dev/random and /

dev/urandom.

2. Voeg jezelf toe als nieuwe gebruiker. Beschrijf hoe je dat gedaan hebt.  
Er is nu een speciale folder aangemaakt voor deze nieuwe gebruiker, waar is deze folder geplaatst?

=> sudo useradd Arnoud => zonder sudo mag je de command niet uitvoeren, hiervoor zijn de sudo bevoegdheden nodig. De map staat in /etc/… hierin staat alles over het account

Voer de volgende opdrachten uit en beschrijf hoe je dat gedaan hebt:

3. Verwijder een applicatie uit de Favorites, en voeg een andere toe.

Start > select app > rechtermuisknop > Add to favorites/remove from favorites

4. Verhoog de timeout van je screensaver tot 30 minuten.

=> gsettings set org.gnome.desktop.session idle-delay 600

## Reflectie

### Arnoud

Voor mij was het uitvoeren van de commands nieuw en zeker een uitdaging. Het kostte veel tijd om de manuals en het .pdf bestand door te lezen, voordat ik er iets van snapte, maar daarna kostte het maken van de opdrachten niet veel moeite.

Vooral het echo command vond ik erg leuk om te gebruiken, cool om te zien dat iets erg vanzelfsprekends toch zo kan werken. Ook was het leerzaam om meer te weten over wat bepaalde computer files/directories zijn en inhouden, hier denk je normaal gesproken niet over na.

Het enige waar ik nog naar moet kijken is hoofdstuk 11, omdat ik het vermoeden heb dat dit nog niet helemaal is doorgedrongen.

### Bram

Ik heb al redelijk wat ervaring met Linux, ik heb gekeken naar de opdrachten en vond dat ik ze al wel goed genoeg beheerste om deze voor deze week niet te hoven doen. Ik zal wel elke week naar de opdrachten kijken, want ik weet natuurlijk ook niet alles en vind het ook leuk om nieuwe commando’s te leren want die kan ik dan misschien ook wel toepassen in mijn eigen omgevingen.

## Feedback

Goed begin, nu ervoor zorgen dat Bram ook betrokken wordt bij de opgaven; eventueel extra werkzaamheden voor hem?

# Week 2

## Opdrachten

### 12.9 practice: commands and arguments

1. How many arguments are in this line (not counting the command itself).

touch '/etc/cron/cron.allow' 'file 42.txt' "file 33.txt"

=> 3 argumenten

2. Is tac a shell builtin command ?

=> type tac -> tac is tac is /usr/bin/tac => een external command

3. Is there an existing alias for rm ?

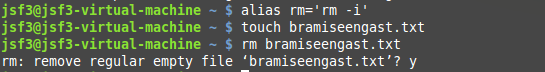
=> alias rm -> bash: alias: rm: not found => geen alias voor rm

4. Read the man page of rm, make sure you understand the -i option of rm.

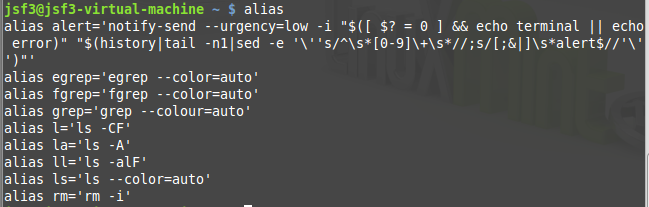
Create and remove a file to test the -i option.

=> Manual gelezen, -i geeft een waarschuwing voordat je files verwijderd

5. Execute: alias rm='rm -i' . Test your alias with a test file. Does this work as expected ?

=> 

6. List all current aliases.

=> 

7a. Create an alias called 'city' that echoes your hometown.

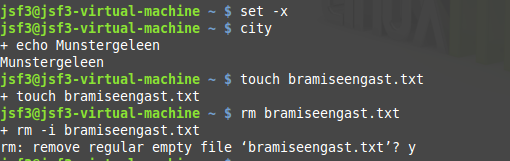
7b. Use your alias to test that it works.

=> 

8. Execute set -x to display shell expansion for every command.

=> set -x

9. Test the functionality of set -x by executing your city and rm aliases.

=> 

10 Execute set +x to stop displaying shell expansion.

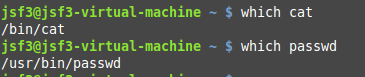
=> set +x

11. Remove your city alias.

=> unalias city

12. What is the location of the cat and the passwd commands ?

=> which cat ; which passwd



13. Explain the difference between the following commands:

echo

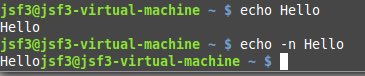
/bin/echo

=> Echo is een shell built-in command, /bin/echo is external -> echo is een alias van /bin/echo

14. Explain the difference between the following commands:

echo Hello

echo -n Hello

=>  => echo -n voert geen new line

15. Display A B C with two spaces between B and C.

=> echo ‘A B C’

(optional)16. Complete the following command (do not use spaces) to display exactly the

following output:

4+4 =8

10+14 =24

=> echo ‘4+4 /t=8’ ; echo ‘10+14/t=24’

17. Use echo to display the following exactly:

??\\

Find two solutions with single quotes, two with double quotes and one without quotes (and

say thank you to René and Darioush from Google for this extra).

=> 1. echo ‘??\\’

2. echo -e ‘??\\\\’

3. echo “??\\\\”

4. echo “??\\\\\\”

5. echo ??\\\\

18. Use one echo command to display three words on three lines.

=> echo -e “one \ntwo \nthree”

### 13.9 practice: control operators

1. When you type **passwd**, which file is executed?

=> which passwd -> /usr/bin/passwd

2. What kind of file is that?

=> file /usr/bin/passwd -> LSB executable

3. Execute the **pwd** command twice. (remember 0.)

=> pwd ; pwd

4. Execute **ls** after **cd /etc**, but only if **cd /etc** did not error.

=> cd /etc && ls

5. Execute **cd /etc** after **cd etc**, but only if **cd etc** fails.

=> cd etc || cd /etc

6. Echo **it worked** when **touch test42** works, and echo **it failed** when the **touch** failed. All

on one command line as a normal user (not root). Test this line in your home directory and

in **/bin/**.

=> touch test42 && echo it worked || echo it failed

7. Execute **sleep 6**, what is this command doing?

=> Het pauzeert de terminal voor 6 seconden.

8. Execute **sleep 200** in background (do not wait for it to finish).

=> sleep 200 &

9. Write a command line that executes **rm file55**. Your command line should print 'success'

if file55 is removed, and print 'failed' if there was a problem.

=> rm file55 && echo success || echo failed

(optional)10. Use echo to display "Hello World with strange' characters \ \* [ } ~ \

\ ." (including all quotes)

=> echo \"Hello World with strange\' characters \\ \\* \[ \} \~ \\ \. \"

### 14.13 practice: shell variables

1. Use echo to display Hello followed by your username. (use a bash variable!)

=> echo Hello $USER

2. Create a variable answer with a value of 42.

=> answer = 42

3. Copy the value of $LANG to $MyLANG.

=> MyLANG = $LANG

4. List all current shell variables.

=> set

5. List all exported shell variables.

=> env

6. Do the env and set commands display your variable?

=> set | more -> displays answer

=> env | more -> doesn’t display answer

6. Destroy your answer variable.

=> unset answer

7. Create two variables, and export one of them.

=> var1=one ; export var2=two

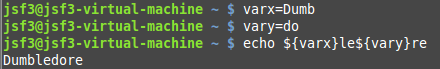
8. Display the exported variable in an interactive child shell.

=> bash

=> echo $var2

9. Create a variable, give it the value 'Dumb', create another variable with value 'do'. Use

echo and the two variables to echo Dumbledore.

=> 

10. Find the list of backslash escaped characters in the manual of bash. Add the time to your

PS1 prompt.

=> PS1='\t \u@\h \W$ '

## Reflectie

### Arnoud

De opdrachten van de pdf waren voor mij niet echt moeilijk, het leuke was om berichten te echo-en. Ik vond het maken van de Java-applicaties wel uitdagend, hier heeft Bram me erg mee geholpen. Uiteindelijk snapte ik het wel doordat Bram me dingen uitlegde.

### Bram

De opdrachten vond ik voor mijzelf goed te doen, ik kende de commando’s al wel en heb met Arnoud samen gekeken naar de Java applicatie. We hebben alle 3 de applicaties samen gemaakt en getest. Het was voor mij wel leuk om te zien hoe dat precies werkte met zo’n properties bestand want daar had ik zelf nog nooit mee gewerkt, maar de principes daaromheen al wel.

# Week 3

Als Software-engineer wil je zo goed mogelijke software afleveren, die niet alleen voldoet aan functionele eisen, maar ook aan kwaliteitseisen zoals performance. Inzicht in de performance van je software is hiervoor noodzakelijk. Hieraan gaan we deze week werken.

## Processen en procesbeheer in Linux

1. Maak een tekening van de proces “stamboom” van het Linux systeem vlak na het opstarten.   
   Dit kan met pen en papier, gebruikmakend van de output van het commando ps –eF (als root). Dit commando geeft van ieder proces het eigen process-id (PID) en het process-id van zijn parent (PPID) en veel andere detailinformatie per proces.   
   Je mag ook gebruik maken van een standaard aanwezig shell-commando om dit voor je uit te ‘tekenen’.

=> init─┬─ModemManager───2\*[{ModemManager}]

├─NetworkManager─┬─dhclient

│ ├─dnsmasq

│ └─3\*[{NetworkManager}]

├─accounts-daemon───2\*[{accounts-daemon}]

├─acpid

├─at-spi-bus-laun─┬─dbus-daemon

│ └─3\*[{at-spi-bus-laun}]

├─at-spi2-registr───{at-spi2-registr}

├─avahi-daemon───avahi-daemon

├─bash───java───26\*[{java}]

├─bluetoothd

├─colord───2\*[{colord}]

├─console-kit-dae───64\*[{console-kit-dae}]

├─cron

├─csd-printer───{csd-printer}

├─cups-browsed

├─cupsd

├─4\*[dbus-daemon]

├─3\*[dbus-launch]

├─dconf-service───2\*[{dconf-service}]

├─gconfd-2

├─6\*[getty]

├─gnome-keyring-d───6\*[{gnome-keyring-d}]

├─gvfs-afc-volume───2\*[{gvfs-afc-volume}]

├─gvfs-gphoto2-vo───{gvfs-gphoto2-vo}

├─gvfs-mtp-volume───{gvfs-mtp-volume}

├─gvfs-udisks2-vo───2\*[{gvfs-udisks2-vo}]

├─gvfsd───{gvfsd}

├─gvfsd-fuse───4\*[{gvfsd-fuse}]

├─gvfsd-metadata───{gvfsd-metadata}

├─gvfsd-trash───6\*[{gvfsd-trash}]

├─kerneloops

├─mdm───mdm─┬─Xorg

│ └─cinnamon-sessio─┬─applet.py───{applet.py}

│ ├─cinnamon-killer───2\*[{cinnamon-killer}]

│ ├─cinnamon-launch─┬─cinnamon─┬─gnome-terminal─┬─bash───pstree

│ │ │ │ ├─gnome-pty-helpe

│ │ │ │ └─3\*[{gnome-terminal}]

│ │ │ └─5\*[{cinnamon}]

│ │ └─{cinnamon-launch}

│ ├─cinnamon-screen───3\*[{cinnamon-screen}]

│ ├─cinnamon-settin───4\*[{cinnamon-settin}]

│ ├─mintupdate-laun───sh───mintUpdate───2\*[{mintUpdate}]

│ ├─nemo───3\*[{nemo}]

│ ├─nm-applet───2\*[{nm-applet}]

│ ├─polkit-gnome-au───{polkit-gnome-au}

│ ├─ssh-agent

│ └─3\*[{cinnamon-sessio}]

├─nmbd

├─polkitd───2\*[{polkitd}]

├─pulseaudio───2\*[{pulseaudio}]

├─rsyslogd───3\*[{rsyslogd}]

├─rtkit-daemon───2\*[{rtkit-daemon}]

├─smbd───2\*[smbd]

├─sshd

├─systemd-logind

├─systemd-udevd

├─udisksd───4\*[{udisksd}]

├─upowerd───2\*[{upowerd}]

├─upstart-file-br

├─upstart-socket-

├─upstart-udev-br

├─vmtoolsd───{vmtoolsd}

└─vmtoolsd

1. Start NetBeans op. Zoek het PID van het NetBeans proces (via ps en eventueel grep). Zoek het parent proces van NetBeans, welk proces is dit?

=> Command: ps -efj | grep NetBeans

=> UID PID PPID PGID SID C STIME TTY TIME CMD

jsf3 13478 13367 13477 13367 0 14:01 pts/2 00:00:00 grep --colour=auto NetBeans

## Processen en Java

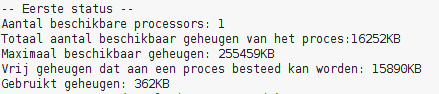
Vanuit Java is het mogelijk om processen op te starten, informatie over processen op te vragen en processen te managen. Dit gaan we onderzoeken in de volgende opdracht.

Gebruik JavaDoc of Google om informatie over klassen op te zoeken.

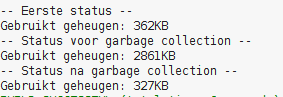
Maak hiertoe in NetBeans in je Linux VM een nieuwe Java Application aan met de naam 'JSF31\_w03\_OS\_process'. In de Wizard kun je bij “Create Main Class” aangeven in welke klasse de main methode moet komen; dit moet klasse 'JSF31\_w03\_OS\_Runtime' in package 'runtime' zijn.

Maak de hieronder gevraagde programma's door code toe te voegen aan de methode main, demonstreer ze aan de docent en zorg dat je hun werking kunt uitleggen.

1. Elke Java applicatie heeft een Runtime object dat de koppeling tussen de Java applicatie en het bijbehorende proces op het OS bijhoudt.   
   Vraag dit Runtime object op, en print de volgende gegevens:
   1. het aantal beschikbare processoren
   2. de totale hoeveelheid geheugen die het proces momenteel beschikbaar heeft
   3. de hoeveelheid geheugen die maximaal beschikbaar gesteld kan worden aan het proces
   4. de hoeveelheid vrij geheugen die het proces momenteel heeft
   5. de hoeveelheid geheugen die het proces momenteel gebruikt (dat is het verschil tussen het momenteel beschikbare geheugen en het vrije geheugen)

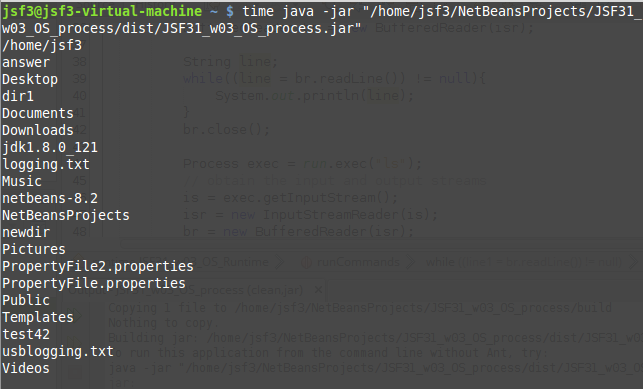


1. Het is ook mogelijk om aan de garbage collector van Java een verzoek te doen om objecten vrij te geven. Om dit goed te kunnen zien, moeten we eerst veel objecten maken. Dit doen we door in een loop veel (100000) verschillende String objecten aan te maken:  
   String s;  
   for(int i=0; i<100000; i++) {  
   s = “Hello”+i;  
   }  
   Print vóór en na de loop de hoeveelheid gebruikt geheugen (zie ook vraag 3e). Roep vervolgens de garbage collector aan via gc().   
   Print hierna weer het geheugen.   
   Verklaar wat je ziet.



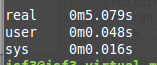
De ongebruikte variabelen (in het geheugen) worden opgeruimd.

1. Start een proces (bijvoorbeeld de “gnome-calculator”) op vanuit je applicatie. Wacht een aantal seconden in je main methode (met Thread.sleep) en sluit het proces weer.   
   Doe dit een maal door een ProcessBuilder object te gebruiken (de methode uit de sheets), en een maal door de methode exec uit de klasse Runtime te gebruiken. Bewaar en toon de code.
2. Het is vaak nodig dat je van een opgestart proces de uitvoer opvangt en die weergeeft. Je realiseert dit, door het proces aan een inputstream te hangen, en die uit te lezen. Als proc een Process is, kun je dat als volgt doen:   
   // obtain the input and output streams  
   InputStream is = proc.getInputStream();  
   InputStreamReader isr = new InputStreamReader(is);  
   BufferedReader br = new BufferedReader(isr);  
     
   String line;  
   while ( (line = br.readLine()) != null ) {  
   System.out.println(line);  
   }  
   br.close();  
     
   Test dit door een Linux commando te nemen waarvan je weet wat de output zou moeten zijn (bv “ls” of “pwd”).



## Performance meten (in tijd)

1. Run van bovenstaande project de bijbehorende jar file in een shell (zie week 2, stap 3).   
   Gebruik het Linux commando 'time' (H20.6 uit LinuxFun), om te meten hoelang dit programma runt.



1. Je programma gaan we nu meten vanuit de code zelf.   
   Maak hiertoe gebruik van het gegeven package 'timeutil' (zie SharePoint ). Voeg dit toe aan het NetBeans project (eenvoudigweg door met drag-and-drop de uitgepakte folder timeutil in NetBeans te slepen).  
   Bekijk en run de MetingDemo om te zien hoe je timeutil kunt gebruiken.  
   Voeg nu in de code uit onderdeel 5 methode-aanroepen toe die meten hoelang het creëren van een proces met ProcessBuilder duurt, en hoe lang het duurt met Runtime.exec.  
   Meet ook de tijdsduur voor het hele programma.   
   Vergelijk deze meting met de metingen die je in vraag 7 d.m.v. het Linux commando 'time' hebt gedaan.



## Threads en Java

We gaan nu de bovenstaande code opdelen, zodat het deel dat een proces opstart in een eigen thread kan gaan draaien. Dit doen we in een aantal stappen.

1. Breid je main methode zo uit, dat deze meerdere commando's kan uitvoeren. De uit te voeren commando's moeten meegegeven worden als command line arguments.  
   Gebruik 2 command line arguments per uit te voeren commando om het mogelijk te maken om commando's met 1 optie te kunnen uitvoeren.   
   Gebruik een lege string “” als een commando geen opties heeft.  
     
   Let op: ProcessBuilder en exec gaan op verschillende manieren met de command line arguments om: bij ProcessBuilder worden de commandonaam en de arguments ieder in aparte strings gezet, en bij exec worden commandonaam en arguments allemaal in één string gezet, gescheiden door spaties.  
     
   De commando's dienen na elkaar uitgevoerd te worden, dus elk commando wordt pas gestart als het voorafgaande klaar (terminated) is (gebruik hiervoor een van de methodes van klasse Process).   
   De output van elk commando moet getoond worden in het console, zoals in vraag 6.
   * Voorbeeld:  
     Met de volgende command line:  
     java -jar MyProg.jar ls -al gnome-calculator “” cat /etc/passwd  
     worden achtereenvolgens de commando's   
     ls -al  
     gnome-calculator  
     cat /etc/passwd  
     opgestart.   
     Let op: java moet eigenlijk aangeroepen worden met de volledige padnaam ervoor (zoals in week 2, stap 3); dat is hierboven weggelaten om het commando op 1 regel te laten passen.
2. Voeg code toe om de totale tijd hiervan te meten.  
   Bewaar en toon de code van vraag 9 en 10.



1. Maak nu voor elk met main meegegeven commando een eigen thread aan.   
   Maak hiervoor een nieuwe Runnable klasse, die parameters accepteert in een constructor:
   * het uit te voeren commando, en
   * het met dit commando meegegeven argument.

Als voorbeeld zou je dus 'ls' en '-al' mee moeten kunnen geven om   
'ls -al' te kunnen starten.  
De thread moet het commando starten, wachten tot het commando klaar is, en dan een tekst printen als “command ls -al is terminated” (maar dan natuurlijk niet altijd met 'ls -al', maar met de echte naam van het commando).  
Voeg aan het eind van je main methode een sleep(10000) toe, die er voor zorgt dat de main methode niet te snel klaar is.   
Meet ook nu weer de totale benodigde tijd.  
Demonstreer dit programma aan de docent en leg uit hoe het werkt.



## Reflectie

### Arnoud

Deze week was erg leerzaam. Door mijn gelimiteerde ervaring wat betreft Java/Linux was hier Bram ook weer grote hulp. Uiteindelijk was het principe van Threads wel duidelijk, als ook de rest van de stof (uitvoeren commando’s via de terminal). Het was ook leuk om te zien dat de code die we hadden geschreven, werkte.

### Bram

Voor mij was deze week erg interessant, ik ken het principe wel qua Multi-threading en processen maar had nog nooit gewerkt met processen. Eigenlijk is dit heel makkelijk te doen en zeer nuttig. Ook vond ik het wel leuk om te kijken naar hoe het zat met hoeveel tijd het kost om een java applicatie te runnnen.

# Week 4

1. Maak een console applicatie in NetBeans en kopieer het package calculate vanuit het KochFractal – startup project hierin. We gaan nu de coördinaten van alle edges van de fractal van level 1 uitrekenen en in het console tonen. Doe dit door in de main methode:
   * het KochFractal object te maken,
   * het level op 1 in te stellen en tenslotte
   * de 3 generate...Edge methodes één voor één aan te roepen.

Als we het hierbij zouden laten, zouden we geen resultaat zien omdat er geen Observer is voor de KochFractal Observable.   
Voeg een Observer toe die de edge-coördinaten print. Dit kan in principe op 2 manieren:

* + Maak een nieuwe klasse die de interface Observer implementeert, implementeer daarin de methode update, zodanig dat deze het begin- en eindpunt van een edge afdrukt (zorg voor een duidelijke begeleidende tekst in de output). De betreffende edge is te vinden in de tweede parameter (arg) van update. Deze kan bijvoorbeeld als volgt gebruikt worden:  
     Edge e = (Edge)arg;  
    Maak in de main methode een object van deze klasse en voeg dat als Observer toe aan het KochFractal object.
  + Of maak van de al bestaande klasse (waar ook de main methode in zit) een Observer door deze de interface Observer te laten implementeren, en door de methode update aan deze klasse toe te voegen.   
    Maak in de main methode een object van deze klasse en voeg dat als Observer toe aan het KochFractal object.

1. Als het goed is worden nu de coördinaten van 3 edges afgedrukt omdat we het level hadden ingesteld op 1, hetgeen een driehoek voorstelt. De eind-coördinaat van elke edge is gelijk aan de begin-coördinaat van de volgende edge.  
   Stel het level vervolgens in op 2. Er moeten dan 12 edges geprint worden, waarbij ook weer de eind-coördinaat van elke edge gelijk is aan de begin-coördinaat van de volgende edge. Merk op dat alle geprinte coördinaten in de range [0..1] liggen.
2. We gaan nu verder met de Java FX applicatie in het project JSF31KochFractalFX. Maak een nieuwe klasse KochManager in het package calculate. Creëer een KochFractal object en zorg ervoor dat klasse KochManager een Observer wordt van de fractal. De constructor van deze klasse wordt aangeroepen in JSF31KochFractalFX met een referentie naar de Java FX applicatie zelf.
3. Test nu je applicatie. Als het goed is kun je het level van de Koch fractal verhogen en verlagen. Probeer ook of je de Koch fractal kunt verplaatsen (slepen met linkermuisknop), de fractal kunt vergroten (linker muisknop) en de fractal kunt verkleinen (rechter muisknop).
4. Definieer een ArrayList als private attribuut van de klasse KochManager.   
   In methode update moet nu telkens de edge toegevoegd worden aan deze ArrayList. Als je dit goed doet zijn, nadat de 3 generate...Edge methodes klaar zijn, alle edges berekend en opgeslagen in de ArrayList. Gebruik een for-statement om alle edges uit de ArrayList te tekenen in methode drawEdges().
5. Gebruik de TimeStamp klasse van week 2 om steeds de tijd te bepalen die nodig is om de fractal te tekenen (bepaal dus hoe lang drawEdges() duurt). Geef deze tijd weer door een aanroep van application.setTextCalc().
6. Geef ook het aantal edges van de fractal weer door een aanroep van application.setTextNrEdges(). Klasse KochFractal heeft een methode om dit aantal op te vragen. Bewaar deze versie van het project.
7. Run het programma voor steeds hogere levels en maak een tabel waarin voor oplopende levels de benodigde tijd en het aantal edges weergegeven wordt.   
   Bij een bepaald level zul je een heap-overflow melding krijgen omdat de ArrayList dan te groot wordt voor het beschikbare geheugen. Het kan ook gebeuren dat de berekeningen extreem lang gaan duren, dus stop als dat meer dan 30 seconden is.
8. Zoals we het nu geïmplementeerd hebben wordt een fractal iedere keer opnieuw berekend als drawEdges() wordt aangeroepen, dus als er gezoomd wordt, als de fractal verplaatst wordt, en als het level veranderd wordt. In principe doen we dan te veel, want als een fractal gezoomd of verplaatst wordt, veranderen de edges niet. We hoeven eigenlijk alleen de edges opnieuw te berekenen bij het veranderen van level.   
   Verplaats daarom de aanroepen van de 3 generate...Edge methodes naar changeLevel(). Zorg dat de fractal opnieuw getekend wordt als de edges berekend zijn. Test weer het zoomen, verplaatsen en veranderen van level.
9. Bepaal nu apart de tijd die nodig is om de fractal te *berekenen* (de aanroep van de 3 generate...Edge methodes) en om de fractal te *tekenen* (het for-statement).   
   Geef deze 2 tijden weer door een aanroep van application.setTextCalc() en een aanroep van application.setTextDraw(). Ga na dat, bij zoomen en verplaatsen, het tekenen nu sneller verloopt dan voorheen.

Door welke thread worden de berekeningen gedaan?  
Run je applicatie