Linux

Commando’s

lspci => info opvragen over de hardware van je toestel

ctr + alt + l => Lock screen (werkt enkel in de Linux desktop dus niet in de server)

sudo shutdown -h now => zet server/pc direct uit

sudo apt-get install “software” => zo kan je software installeren op je pc

ls => laat zien welke zichtbare mappen er zijn in de map waar je op dat moment bevind

-a => laat ook alle hidden files zien

-h => laat de grote zien van de bestanden in een normaal cijfer dus altijd zo een klein mogelijk cijfer

-l => geeft meer details over het bestaande die je zoekt

-LR => geeft alle files van een bepaald plaats

-lh => laat de grote van een file zien

cd => gaat naar de plaats die je na de cd plaats bv cd /home/student

pwd => laat het hele pad zien waar je opdat moment bevind bv /home/student

sudo addgroup 'groepsnaam' => voegt een groep toe met de naam die je daar zelf voor kiest

sudo adduser 'gebruikersnaam' 'groepsnaam' => voegt een persoon in een bepaalde groep

sudo deluser 'gebruikersnaam' 'groepsnaam' => verwijderd de persoon uit de groep maar niet de gebruiker zelf

sudo delgroup => verwijderd een groep

man -k commando => geeft info over het commando dat je wilt en ook commando’s die er op lijken

man –f commando => geeft een korte informatie over wat dit doet(zelfde als whatis commando)

whatis “commando” => doet juist het zelfde maar gaat enkel kijken naar het commando dat je hebt ingegeven

whereis –m “commando” => geeft de locatie waar de manpage van dat commando staat op je pc

mandb => maakt of update de manpages hun index

cd - => gaat naar de vorige map

cd .. gaat naar de bovenliggende map

cd ../.. twee mappen boven

cd ~ => gaat naar home directory van de huidige gebruiker

mkdir => maakt een map aan

-p maakt nodige sub mappen aan die nodig zijn om te gebruiken als je een pad geeft.

rm => verwijderd een bestand rm -rf verwijder alle onderstaande bestanden en forceer het(veel gebruikt) geeft ook geen error als het commando niet mogelijk is.

rm - i vraagt eerste of je het bestand echt wilt verwijderen(word niet vaak gebruikt)

rmdir => verwijderd enkel een lege map kan ook rm-rf gebruikt worden maar rmdir kan enkel voor een lege map.

-p doet hetzelfde als de –p bij mkdir het maar hier verwijderd het ok de lege submappen die er zouden zijn

ipconfig => geeft info ip adress macadress en netwerk....

cp => copy een bestand en hernoem het

cp -i => je kan geen bestanden overschrijven

cp -p => het kopieert ook de rechten van het bestand en ook de datum en tijd waar het origineel bestand is gemaakt.

Cp –r => zo kan je een hele map kopiëren naar een subfolder

touch => maakt een bestand aan(je kan er nog wel niet direct iets in schrijven)(om een ander commando is cat > “filename”)

touch -t => veranderd de datum van een bestand(jaar, maand, dag, tijd) 200510251405(eerst de tijd en dan de file)

mv => verplaats bestand kan ook een naam van een bestand veranderen.

rename => veranderd de naam van meerdere bestanden word vooral gebruikt om extensie van bestanden in 1 keer te veranderen of om van hoofdletters naar kleine letters te gaan bv rename 'y/A-Z/a-z/' \*

file => geef de soort van het bestand dus

head => geef de eerste 10 regels van een bestand(de 10 regels is een automatische waarde)

tail => geef de laatste 10 regels van een bestand (de 10 regels is een automatische waarde)

head/tail -5 => geef de eerste/laatste 5 regels van het bestand (natuurlijk kan de 5 aangepast worden door een random getal)

cat => geef het volledige bestand weer

cat > "filename.\*" => maakt een bestand aan en je kan typen tot dat je ctrl - d doet dan sluit je het bestand af maar ook lukt ctrl –c.

cat > "filename.\*" <<"stopwoord" => je kan zolang typen tot dat je het woord typt dat je na de twee kleiner dan teken hebt getypt.

(ook mogelijk cat <<"stopwoord"> "filename")

cat "filename.\*" > "filename2.\*" => kopieert een bestand en veranderd de naam in wat je ingegeven hebt ook de inhoud van het bestand word gekopieerd.

Cat “filename” >> “filename2” => maakt een bestand aan en zet de inhoud van filename2 achter de inhoud van filename

Cat “filename” << “filename => maakt een bestand aan en zet de inhoud van filename2 voor de inhoud van filenme

tac => geeft het bestand omgekeerd weer

more/less => beide commando's laten grote bestanden zien en begint van de eerste lijn die hij laat zien en met de pijltjes toetsen kan je navigeren(je kan wel enkel een bestand laten zien met more of less dus je kan niet een groot commando laten zien met more of less.)

(best less omdat je bij less ook terug naar boven kan gaat als je naar beneden bent gegaan bij more lukt dit niet)

strings => geef een bestand dat normaal niet gemaakt is om te lezen(ascii code) weer in leesbare tekst

ls -lrS => geeft het grootse bestand in een map als laatste

echo => print een tekst af. Als je tekst print zonder quotes en je dan spaties doet worden overige spaties weggelaten.

Als je typt met single of double quotes dan worden spaties gerespecteerd dus ook spaties die te veel zijn.

echo -e => als je een \n doet dan kom krijg je na de backslash een enter als je een \t doet dan doet hij een tab. In beide gevallen moet er wel de tekst tussen single of dubbele quotes gebruikt worden

type => geeft weer waar het commando zich bevind en zegt of het exernal of build in is

type -a => kan twee commando’s geven “build in” en “exteranal”

which => geeft het pad waar het commando zich bevind

alias =>je kan een woord gelijk stellen aan een bestaand commando bv alias = dog =tac als je gewoon alias typt krijg je alle aangemaakte aliassen. met dit commando kan je ook een bestaand commando veranderen naar een bestaand commando bv alias rm = 'rm -rf' elke keer dat je nu rm typt word er eigenlijk rm –rf gedaan door de pc.

Of alias city =’echo Antwerp’

vi ~ /.bashrc => zo kan je aliassen in geven door ze te typen na bestaande maar dan worden ze opgeslagen als je vm herstart(deze aliassen kan je pas gebruiken nadat je je vm herstart hebt)

unalias => hier kan je een alias verwijderen

cc test.c –o test.out =>compileer het bestand met een c compiler (zippen)

set => laat alle variabele zien die zijn opgeslagen op je pc

je kan laten zien wat de pc doet door set -x te typen als je het terug wilt afzetten doe je set +x

set -u/set -o nounset => als je een van de twee commando’s intypt dan krijg je in elke keer als je een variabele intypt die niet bestaat dan krijg je een melding dat de variabele niet bestaat maar dan word er ook de rest niet getoond.

set +u/set +o nounset => als je een van de twee intypt na een van de bovenste dan krijg je terug zoals eerst gewoon geen fout melding en toch de rest van uitvoer te zien dus word de foute variabele niet door de pc naar gekeken.

unset => als je een variabele een bepaalde waarde hebt gegeven dan kan je altijd die variabele leeg maken door middel van unset "variabele" daarna is je variabele terug leeg

; => als je een ; plaats nadat je een commando typt hebt dan kan je achter die ; een volgend commando typen zo kan je meerdere commando’s op een lijn typen.

& => als je een & typt na een commando typt dan laat hij je zien wat er op de achtergrond gebeurd word

$? => als je $? na een commando typt dan zijn er maar twee mogelijke dingen die hij je kan terug geven een 0 als het commando gelukt is en er geen foutmelding gegeven is en een 1 als er fout is gebeurd.

&& => als je je eerste commando typt en dan && en daarna je tweede commando typt gaat hij enkel iets op het scherm tonen als de beide commando’s juist zijn in getypt en kunnen uitgevoerd worden. Dus als er een fout zit in een van de twee commando’s word er niets uitgevoerd dus ook niet een commando die wel juist is.

|| => als je je eerste commando typt en dan || en daarna je tweede commando typt dan gaat hij kijken of hij het eerste commando succesvol kan uitvoeren als dat gaat dan voert hij dat uit en het en doet hij niets met het tweede commando.

Als het eerste commando een fout geeft dan krijg je een foutmelding zo als anders maar omdat hij het eerste commando niet heeft kunnen uitvoeren voert hij nu wel het tweede commando uit tenminste als dit wel correct is.

(natuurlijk kan je de bovenste twee commando’s ook combineren in een groot commando bv rm file1 && echo it worked || echo it failed

hier ga je dus iets verwijderen en iets zeggen hier moeten ze allebei juist zijn om te tonen want als er een niet werkt word er de tweede optie getoond it failed

# => met dit teken kan je tekst toevoegen achter een tekst zonder dat er door de computer/server naar gekeken word het is enkel als extra informatie voor de gebruiker.

\ => met een backslash kan je twee dingen doen het eerste is een speciaal teken typen wat anders niet word herkent door de computer dus echo \; voor een punt komma te typen want anders laat hij niet zien.

De tweede mogelijkheid van een backslash is dat je een groot commando typt in kleinere stukken zodat het overzichtelijker. Dit doet wel niets aan wat je krijgt het is gewoon makkelijker voor de gebruiken dus bv

echo This command line \

is split is three \

parts

geeft het zelfde als

echo this command line is not split in three part but still the same

$ => als je een $ teken voor een naam van iets plaats gaat hij niet gewoon de naam afdrukken maar de verwijzing er naar dus als je doet echo SHELL dan krijgt je gewoon SHELL maar als je typt echo $SHELL dan krijg je /bin/bash.

Er moet in dat geval wel gelet worden op hoofdletters want als je het typt zonder hoofdletters terwijl het in het geheugen staat met hoofletters dan krijg je niets omdat je verwijst naar niets.

Ook kan je variabele zo afdrukken als je zegt MyVar = 555 echo MyVar dan krijg je gewoon MyVar maar als je typt dan krijg je echo $MyVar dan krijg je 555.

Hier moet je ook wel oppassen voor quotes want als je typt echo '$MyVar' krijg je gewone tekst $MyVar bij dubbele quotes heb je dit probleem niet daar word ook de verwijzing gemaakt maar bij enkele word het gewoon gezien als tekst.

$PS1 => je kan de terminal aan passen door de terminal van verschillende kleuren aan tekst of achtergrond

* **\a** : an ASCII bell character (07)
* **\d** : the date in "Weekday Month Date" format (e.g., "Tue May 26")
* **\D{format}** : the format is passed to strftime(3) and the result is inserted into the prompt string; an empty format results in a locale-specific time representation. The braces are required
* **\e** : an ASCII escape character (033)
* **\h** : the hostname up to the first '.'
* **\H** : the hostname
* **\j** : the number of jobs currently managed by the shell
* **\l** : the basename of the shell’s terminal device name
* **\n** : newline
* **\r** : carriage return
* **\s** : the name of the shell, the basename of $0 (the portion following the final slash)
* **\t** : the current time in 24-hour HH:MM:SS format
* **\T** : the current time in 12-hour HH:MM:SS format
* **\@** : the current time in 12-hour am/pm format
* **\A** : the current time in 24-hour HH:MM format
* **\u** : the username of the current user
* **\v** : the version of bash (e.g., 2.00)
* **\V** : the release of bash, version + patch level (e.g., 2.00.0)
* **\w** : the current working directory, with $HOME abbreviated with a tilde
* **\W** : the basename of the current working directory, with $HOME abbreviated with a tilde
* **\!** : the history number of this command
* **\#** : the command number of this command
* **\$** : if the effective UID is 0, a #, otherwise a $
* **\nnn** : the character corresponding to the octal number nnn
* **\\** : a backslash
* **\[** : begin a sequence of non-printing characters, which could be used to embed a terminal control sequence into the prompt
* **\]** : end a sequence of non-printing characters

$PATH => het geef de locatie waar de bestande staan waar de commando’s in staan die de pc nakijkt elke keer als je een commando intypt.

env => dit commando geef een lijst van alle commando’s die je geëxporteerd hebt.

export => met dit commando kan je een variabele gaan exporteren zodat de variabele ook geldig is in een andere bash want als je dit niet doet kan je de variabele enkel gebruiken in de huidige bash.

${} => als je een variabele aanmaakt en je wilt er direct iets achter plaatsen kan je bv doen zoals prefix=Super echo Hallo ${prefix}man and ${prefix}girl want als je het zonder de haakjes doet dan gaat de pc opzoek naar de volledige variabele naam die dan niet bestaat en dus gewoon een leeg veld geeft.

$() => je kan door een $() rond bepaalde commando’s een andere bash venster nabootsen dit kan handig zijn als je één keer een variabele wilt aanpassen. bv. echo $(var1=5 ; echo $var1) 5 maar als je nu nog is $var1 zou afdrukken zou je niets krijgen omdat het enkel een waarde heeft binnen de haakjes als je dat toch zou willen moet je een export tussen de haakjes doen.

`` => een ander manier om te je tekst te doen verwijzen naar je variabelen is gewoon door het gebruik van `` met je commando tussen dus bv `var1=5;echo $var1` dan krijg je als uitkomst 5 en dus niet het commando DIT ZIJN GEEN single quotes maar backticks (alt gr onder de µ)

echo $- => geef de huidige bash opties die actief zijn dus als je het normaal invoert dan krijg je himBH maar als je set -C doet en doet dan echo $- dan krijg je himBCH.

dev/null => kan een bestand sturen naar een zwart gat dus verwijdert het.

!! => voert het vorige commando uit.(en laat het vorige commando ook zien)

!"letters" => voert het laatste commando uit dat met die voorwaarde voldoet dus bv !c zal gaan zoeken naar een commando dat begin met een c en dan dat commando uitvoert.(en het toont ook het commando dat word gebruikt)

history => geeft een lijst van de commando’s die je getypt hebt je kan er ook een getal achter zetten om de lijst kleiner of groter te maken(als je geen getal mee geeft zal hij gewoon de laatste 10 geven)

!"nummer" => als je historie getypt hebt staat ook voor elk commando een cijfer je kan ook !en dan dat cijfer van het commando dat je wilt dus bv !45 doet bv cd

$HISTSIZE => een variabele waar het aantal commando’s inzit. dit kan je aanpassen door HISTSIZE="getal"

$HISTFILE => een variabele die de historie bijhoud van de commando’s die je hebt in getypt dit kan je op de zelfde manier wijzigen.

$HISTFILESIZE => een variabele die word onthouden in je geheugen ook dit kan aangepast worden

!c:s/1/2 => dit is een voorbeeld van een commando dit commando gaat terug naar een commando met een c en dan voert hij dat commando terug uit maar vervangt alle 1 en door 2

ls File\* => dit is een voorbeeld van een commando dit teken geeft alles met de naam File wat er ook achter staat zolang de eerst 4 char maar File zijn

ls File? => dit is juist het zelfde maar dan ipv meerdere dingen achter kan hier maar juist een char komen te staan.

ls File[a-z] dit is juist het zelfde als een vraagteken maar hier maak je de selectie nog kleiner naar enkel een letter tussen a en z natuurlijk kan je dit ook een paar keer herhalen en of een selectie van letters pakken of met nummers werken.

Ls \*.{jpg,exe,mp4} => geeft alle files met de volgende exentise ingegeven

Ls f(iR)\*[0-9] => geeft een lijst van bestanden die beginnen met een f dan een i of een R en eindigen met een cijfer

Ls [!F]\* => geef een lijst van bestanden die niet beginnen met een F

$LANG => als je LANG = C zet dan kan je het bovenstaande commando invoeren en dan werkt dat ook met hoofdletter gevoelig want van begin is dat geen verschil dus eerst LANG = C => en daarna gaat ls [a-z]ile en [A-Z]ile een ander resultaat geven wat eerst het zelfde zou zijn

| => maakt gebruik van een pipe zo kan je de uitvoer van een commando gebruiken als invoer voor een volgend commando dus door gebruik bv van tac count.txt | cat dat geeft dus de uitvoer van het achterste voren count.txt

tee => zet de output van het vorig commando in een bestand die je een naam geeft zo kan je dit bestand later nog gebruiken dus bv tac count.txt | tee test.txt| cat test.txt dus hier zal tac count.txt en cat test.txt dezelfde output geven omdat het eigenlijk een kopie is.

grep => word gebruikt om te kunnen filteren hier kan je gebruik maken van verschillende zoals [aze] of door verschillende parameters hier dus word er gezocht naar een bestand met een a z of een e

grep-i => zo zoek je maar word er niet gekeken naar hoofdletter gevoelig dus grep -i a test.txt zal alle lijnen met een a weergeven ook al zijn dat hoofdletters

grep-v => zo zoek je naar alles behalve naar het gene wat je in geeft dus als je ingeeft gerp -v a zal je alles krijgen waar geen a in zit dit kan je natuurlijk ook combineren met het bovenstaande commando.

grep-A1 => zo zoek je naar iets maar de je zoekopdracht maar elke keer dat je er een lijn voldoet aan de voorwaarde zal er ook de volgende rij worden afgedrukt ook al bevat die niet de waardes die in de zoekopdracht ingegeven zijn.

grep-B1 => zo zoek je naar iets naar de zoekopdracht maar elke keer dat je er een lijn voldoet aan de voorwaarde zal hij ook de vorige rij worden afgedrukt.

grep-C1 => dit is eigenlijk een combinatie van A1 en B1 je krijgt een lijn voor en een lijn na

grep => best altijd gebruik maken van grep -E

grep -E 'i| a' test => zoekt naar alle lijnen met een i of een a in

grep -E 'o\*'test => zoekt naar 0,1,2,... van het teken in een bestand

grep -E 'o+' test => zoekt naar 1,2,3,..... van het teken in een bestand

grep -E a$ test => zoekt naar een lijn die eindigt met een a in een bestand

grep -E ^Val test => zoekt naar een lijn die start met Val

grep '\bover\b'test => zoekt naar alle lijnen waar het woord in voor komt

grep -w => zoekt enkel naar een woord ipv letters in een woord

grep -E 'ooo?' test => zoekt naar een lijn die 2 o's of 3 bevat want het ? zegt dat de laatste o niet moet maar het wel kan

grep -E 'o{3}' test => zoekt een lijn die juist 3 o's bevat.

grep -E 'o{2,3}' test => zoekt naar een lijn met twee of drie o's

grep –E …e => geeft een lijn met op de vierde plaats een e

grep –n ‘$’$ laat de lijn zien waar een $ op het einde is

cut =>dit commando word gebruikt om uit een grote tekst een bepaalt aantal record te halen dit doe je door cut dan -d daarna komt hoe je scheiding steken wilt hebben dat kan een spatie zijn " " maar dit kan ook een ":" zijn daarna volgt in welke tab die wilt gaan kijken altijd te beginnen met een f en dan een cijfer dus bv f1,2,3 dan ga je tab 1 2 en 3 kijken en dan daarna de bestandnaam dus bv cut -d ":" f1,3 test5.txt

dit commando kan je natuurlijk ook combineren met de pipes

cut -c zo kan je gaan zoeken ok karakters ipv op tabellen door cut -c2-7 test zo krijg je van alle lijnen alle tweede karakters tot de zevende karakters

Tree => laat de files zien van in een 3d boomstructuur tree –L “aantal niveaus” geef de maximum diepte van je boomstructuur

tr => zo kan je een karakters veranderen dus bv een a naar een o naar een hoofdletters of een dubbel spatie naar een spatie. dit werkt ook met de selectie dus bv tr ‘a-z’ ‘A-Z’ of enters door spaties dus tr '\n' '' < “filename” hier

tr -s => zo kan je simpel meerder karakters verwijderen in een tekst bv meerder spaties bv tr -s ' ' zo gaat hij kijken als er meerdere spaties zijn gaat hij die vervangen door één spatie

dit commando kan je dus ook gebruiken om een tekst te versleutelen door bv tr

tr -d => met dit commando kan je ook karakters verwijderen dus kan je een stuk van de naam of een getal verwijderen dus bv tr -e verwijderd alle kleine letters e.

wc -l => geeft de aantal lijnen in een bestand

wc -w => geef het aantal woorden in een bestand

wc -c => geef de grootte van het bestand

ls –ah |cat > lijn | mijnvar=$(wc –l < lijn) | echo “in mijn homefoolder zitten $mijnvar files en folders” => laat geeft doormiddel van een variabel de lijnen met een pipelijn

sort => sorteert een bestand alfabetisch (dus als er enkel getallen instaan gaan ze niet van klein naar groot of van groot naar klein staan)

sort -k1 => nu sorteer je op een kolom dit kan ook k2 of k3 zijn dus bv sort -k2 test.txt

cat /etc/passwd |head | sort –t: -k4 –n => ordent via de gebruikers id dus numeriek

Join =>laat bestanden toevoegen achter elkaar maar moeten wel blijde bestanden geordend zijn en meestal word er vergelen op het eerste veld dit kan met cat –n maar kan je ook op een andere manier bv sort produkten –t: -k2 –n | cat > sort1 |sort prijzen –t: -k4 –n | cat > sort2 |join –t: -1 2 -2 1 sort1 sort2

Dus de –t: is voor te kijken op welk kolom hij moet ordenen-t en de –n zijn voor orden met numeriek dan join –t zegt dat het gescheiden is met een : de- 1 zegt bestand 1 tabel 2 en -2 1 bestand 2 tab 1

uniq => (als je dit commando wilt gebruiken moet er eerst een sort gebeurt zijn want anders werkt dit niet) dit commando zorgt er voor dat als er meerdere keren de zelfde regel hebt verwijdert hij alle regels totdat je er maar 1 meer over hebt.

uniq -c => dit doet het zelfde als bovenstaand commando maar hier zet hij er cijfers voor de lijn een 1 wilt zeggen dat er maar een in stond en dus niets heeft verwijdert een 2 voor ....

comm => laat de lijnen die uniek zijn in een bestand zijn daarna laat hij de lijnen zien die uniek zijn in dat tweede bestand en daarna geeft hij de lijnen die in beide bestanden voorkomen. dus bv comm test1 test2

comm -1 => laat zien wat er in file 1 te zien is maar wel met uniq

comm -2 => laat zien wat er in file 2 te zien is maar wel met uniq

comm -3 => laat zien wat er in de beide files uniq is

(natuurlijk kan je ook de verschillende commando’s combineren bv zo is comm -12 gelijk aan comm -3)

od -t xl => geeft de waarde van alle letters in de file als ascci weer (dus hexadecimaal)

od -b => geeft de waarde octaal waar

od -c => laat een bestand zien elke letter apart en met spaties geschreven als een \n

ps –aux => Geeft de namen van alle users die minstens één proces hebben lopen,geordend op wie het meeste processen heeft lopen

sed => dit vervangt een karakters door een anders karakter dus bv sed 's/42/5/' hier word de 42 vervangen door een 5 maar als je meerder dingen wilt vervangen dan moet je sed 's/5/42/g' nu worden alle bestande die je voor de pipe hebt ingegeven waar er een 42 in zit vervangen door een 5

ook hier kan je een lijnen verwijderen bv sed '/BE/d' verwijder je alle lijnen waar er BE in staat.

(ook kunnen er andere tekens gebruikt worden ipv een / bv een : \_ |)

sed 's/sun/&&/' => vervangt het woord dat er gezocht word dus in dit geval sun en zet er dan dat gedeelte nog is voor dus sunsun (hoe vaak dit herhaalt word hangt van het aantal & en af)

echo Sunday sed 's\_\(Sun\)\_\1ny\_' => hier word Sun in het register 1 gestoken en weggehaald waar het vandaan kwam daarna plak je Sun aan ny en daar achter zet je dan day

echo Sunday sed 's\_\(Sun\)\_\1ny\_ \1' => hier word het zelfde gedaan maar hier word sun aan ny geplakt en aan day van het oorspronkelijk woord

echo 2014-04-01 | sed 's/\(.)-\(..\)-\(..\)/\1+\2+\3/' =>(jaarzetten) hier word de plaats van 2014 in het register 1 gestoken 04 in 2 en 01 in 3 daarna worden die registers terug opgeroepen en worden er plussen toegevoegd dit kan natuurlijk ook streepjes of iets ander zijn.

ln -s /etc/init.d/myscript /etc/rc3.d/S99myscript => begin met het uitvoeren van het eerste bestand myscript tot het laatste s99myscript

who => laat zien wie er allemaal op de pc is aan gelogd en wanneer

who | wc -l => zegt hoeveel mensen er op dat moment zijn aan gelogd

who | cut -d' ' -f1 | sort => geeft het zelfde als gewoon who maar hier krijg je enkel de naam te zien van wie er is ingelogd

who | cut -d' ' -f1 | sort | uniq => dit doet hetzelfde als bovenstaand commando maar dit commando laat een gebruiker maar 1 keer zien als die meerder keren zou aan gelogd zijn

grep bash /etc/passwd | cut -d: -f1 => hier laat hij alle account zonder extra info zien die op de pc bevinden.

echo it is cold today > winter.txt => door middel van gebruik te maken van de > zet je de uitvoer van een commando dat je doet in een bestand hiervoor moeten er wel geen fouten zitten in je commando

set -o noclobber => met dit commando kan je er voor zorgen dat je geen bestanden overschrijft want elke keer als je een tekst in een bestand plaats met > overschrijf je het bestand als je iets wil toevoegen moet je echo test >| test.txt hier voeg je test toe en verwijder je niet wat er al instond ook kan je iets toevoegen door gebruik te maken van >> ipv >

set -C => dit doet juist hetzelfde als het commando hier boven

set +o noclobber => als je dit uitvoert na -o dan kan je terug bestanden gaan overschrijven

set +C => dit commando doet juist hetzelfde als het commando hierboven

2> => met 2> steek je alleen de error die je krijg in een bestand dit kan je natuurlijk ook zeggen dat alle error naar de /dev/null moeten gestuurd worden (verwijderd worden dus)

Rm “file” 2>/dev/null && echo “succesvol verwijder” || echo “fout tijdens het verwijderen” => geeft een voorbeeld van het verwijderen van een file met tekst output

/dev/zero => je kan er niet naar toe stuurne maar het bevad enkel nullen

/dev/random => random nummer generator van linux

Hard diskt zijn blokken van 64 K bytes niet gewoon

2>&1 => hier plaats je zowel de info die word weergegeven dus de output en ook de error die je krijgt samen in een bestand

je kan dit ook door gebruiken met het pipe systeem dus als je rm test1 test2 test3 2>&1 | grep test2 dan krijg je enkel de foutboodschap van test2(in dit voorbeeld moeten dan natuurlijk test1,test2,test3 niet bestaan)

&> => steekt zowel de info die goed is als de error in een bestand dat je wilt

echo $- => laat de shell opties zien

<<< => word gebruikt om een string in een commando te gieten dus bv base64 <<< test.test

(base64 <<< test.test is net het zelfde als echo test.test | base64)(het terug omzetten van een base64 word gedaan via base64 -d en de gegeven encryptie)

soorten notatie van een commando met >

cat test.txt > test2.txt 2> error.txt

2> errors.txt cat test.txt > test2.txt

< test.txt >test2.txt 2> errors.txt cat

dit zijn alle drie commando’s die werken alleen het ene is beter leesbaar en makkelijker te lezen

crtl r => als je crl r typt en dat letters krijg je je laatste commando dat overeenkomt met je hebt in getypt natuurlijk kan er ook gezocht worden met de pijltjestoetsen

clear a file => echo > test.txt

of een makkelijke manier om een file leeg te maken is ook gewoon > “filename”

hostname => zo kan je de hostname van je pc krijgen en die afprinten bv. Echo dit is de pc $(hostname)

IP-adres alleen => ifconfig eth0 | grep ‘inet addr :’ |cut –d –f2 | awk ‘{print $s1}’(dit is het ipadres)

Mac adres alleen => ifconfig eth | grep HWaddr |cut –d ‘ ‘ f-11

Om het mac adress te krijgen met hoofdletters en geen : echo $mac | tr ‘a-z’ ‘A-Z’ | tr-d “:”

Alle drie te samen zet je eerst de dire in een var met een pipeline en als laaste in de pipe echo je de zin die je wilt met de var

clear a file met noclobber => echo >| test.txt

find /etc > etcfiles.txt => zoekt alle bestanden in een bepaalde folder en zet de namen van deze files in een file

find . -name "\*.conf" => gaat zoeken in een bepaalde map zoeken naar een stuk van de naam en toont die

find . -type f => zoekt naar een bepaalt soort van een bestand

c = character

d = directory

f = file

find . -neweir file42.txt => zoekt naar files die nieuwer zijn dan file42.txt

find /data -name "test\*" -exec cp {} test/ \; => maakt een ba

find /data -name "test\*" -ok rm {} \; => verwijderd alle files met test

locate => geeft het pad van alle files die je zoekt

ps ax |grep $$ |awk ‘{print $2}’ => geef een lijst van de gebruikers die tty’s gebruiken

date => geeft de tijd weer in dag / maand / dag(cijfer) uur-minuten-seconde tijdzone - jaar

je kan zelf ook kiezen wat je ingeeft door bv date +'%A %d-%m-%Y' => dag dag(nummer) - maand-jaar

date +%s => de tijd in seconde vanaf 1 jan 1970

date -d 'jaar-maand-dag + aantalsseconde seconds' => geeft de dag weer die je bekomt als je een aantal seconde bij een jaartal doet.vi

cal 'maand' 'jaar' => geeft een kalender weer van de gekozen maand in een gekozen jaar zonder info is het de huidige maand en jaar.

sleep => het programma wacht de totale seconde dat je ingeeft

sleep “second” & => wacht niet tot de tijd voorbij is

time => zegt hoe lang een commando duurt eerst hoe lang het echt duurt dan voor de gebruiker meestal 0 en daarna hoe lang de cpu nodig heeft voor het commando uit te voeren.

gzip => zipt een file naar een .gz file

gunzip => unzipt een gzip zipfile

zcat/zmore => maakt het mogelijk om een gezipte gzip file te lezen die gezipt zijn met gzip

bzip2 => kan een file een zipfile van maken dit duurt iets langer dan gzip maar de grote van het bestand word wel kleiner slower but smaller zip and vice versa

bunzip2 => kan een bzip2 file unzippen

bzcat / bzmore => kan een bzip2 gezipte file lezen

**tar -czvf name-of-archive.tar.gz /path/to/directory-or-file**-> hele dir zippen

This will tell tar to:

-compress it using the z (gzip) algorithm

-c (create) an archive(tar is recursive by default)

-v (verbosely) list (on /dev/stderr so it doesn't affect piped commands) all the files it adds to the archive. list wat de terminal doet (v is niet echt nodig)

-and store the output as a f (file) named archive.tar.gz

The tar command offers gzip support (via the -z flag) purely for your convenience. The gzip command/lib is completely separate. The command above is effectively the same as

tar -cv directory | gzip > archive.tar.gz

To decompress and unpack the archive into the current directory you would use

tar -zxvf archive.tar.gz

MEERDERE DIRS IN 1 ZIP is eerste command met alle paden erachter met spatie tusse  
 alles maar iets niet is --exclude=\*.jpeg alles wat eindigt op .jpeg niet zippe

vi => vi kan gebruikt worden om een bestand aan te passen of een nieuw aan te maken dit is een belangrijk commando omdat er een heleboel opties zijn die dit vergemakkelijken

esc => gaat naar de command mode

a => start typen na de huidige karakter

A => start typen na het einde van de huidige lijn

i => start typen voor de huidige karakter

I => start typen voor de huidige lijn

o => start typen op een nieuwe lijn na de huidige lijn

O => start typen op een nieuwe lijn voor de huidige lijn

x => verwijderd een karakter op de plek van de cursor

X => verwijderd een karakter voor de cursor

r => vervangt een karakter door een karakter dat je daarna ingeeft

p => plakt na de cursor de laatste karakter die je verwijdert hebt

xp => verwisselt twee karakters

u => crtl -z in vi

. => doet nog eens wat je laatste bewerking was bv nog een lijn plakken of een charter verwijderen

dd => verwijdert de huidige lijn

yy => kopieert de huidige lijn(maar plakt die nog nergens gewoon in het geheugen gestoken

p => plakt iets (bv de lijn van yy) na de huidige lijn

P => plakt iets voor de huidige lijn

3dd => verwijdert 3 lijnen na de cursor (de 3 kan ook een ander getal zijn dan is het enkel anders voor de lijnen)

4yy => coöpteert 4 lijnen na de cursor(zelfde als hier boven)

0 => springt naar het begin van de huidige lijn

^ => springt naar het begin van de huidige lijn

$ => springt naar het einde van de huidige lijn

d0 => verwijdert tot het begin van de huidige lijn

d$ => verwijdert tot het einde van de huidige lijn

J => voegt de volgende lijn toe aan de huidige lijn

yyp => kopieert de huidige lijn(word direct na de huidige lijn geplaats)

ddp => verwisselt twee lijnen(snap dit nog niet goed)

w => gaat naar het begin van het volgend woord in dezelfde lijn

b => gaat naar het begin van het vorig woord in dezelfde lijn

3w => gaat nar het begin van het woord 3 woorden verder(dit getal kan aangepast worden)

dw => verwijderd één woord

yw => kopieert één woord

5yb => kopieert 5 lijnen(als je het ook wilt plakken moet je wel nog p doen)

7dw => verwijdert 7 woorden

:w => bewaard het bestand

:w fname => bewaard het bestand en hernoem het naar de naam die je geeft (de naam mag niet al in gebruik zijn)

:q => ga uit de vi

:wq => bewaar het bestand en ga er uit

ZZ => bewaar het bestand en ga er uit

:q! => ga uit het bestand zonder te bewaren (geforceerd)

:w! => bewaar het bestand en schrijft het na ar een niet leesbaar bestand

/string => zoekt van u plaats naar beneden voor een bepaalt woord in uw tekstbestand

?string => zoekt van u plaats naar boven voor een bepaalt woord in uw tekstbestand

n => ga naar de volgende plaats waar het woord gevonden is

:4,8 s/foo/bar/g => vervangt foor door bar op lijn 4 door lijn 8

:1,$ s/foor/bar/g => vervangt foor door bar op alle lijnen

:r fname => plakt de tekst van een bestand in het huidige bestand

:r !cmd => plakt de output van een commando in je vi

"add => verwijdert een lijn en zet die lijn in het register a

"g7yy => kopieert 7 lijnen en zet die in het register g

"ap => print het a register

vi file1 file2 file3 => maakt of opent meerdere bestanden in één keer

:args => laat zien welke vi er op dat moment open staan

:n => zo kan je naar het volgend bestand gaan dat open staat

:e => ga naar het laatste aangepaste document

:rew => ga terug naar het eerste bestand

:set number => geeft de regelnummer weer

:set nonumber => geeft geen regel nummers weer

Voorbeeld om de aantal woorden te tellen cat regtest.txt | sed ‘s/vijf/vijf \n/g’ |cat > spatie |cat spatie |grep –w vijf |cat vijf |cat vijf |wc –l

“REDIRECTION”. Is hoofdletter gevoelig want er bestaat ook redirection in kleine letters dus het heeft een ander betekenis

whoami => geeft de gebruikersnaam waarmee je op dat moment bent aan gelogd

Who am i => laat het zelfde zien als het commando who maar dan enkel van de gebruiker die je op dat moment bent.

w => laat de mensen zien die opdat moment aan gelogd zijn en dat ze aan het doen zijn

id => geeft je de gebruikersid(usersid) de groepid(groupid) en een lijst van groepen waar ze inzitten

useradd => voegt een gebruiker toe aan het systeem bv useradd –m –d /home/yanina –c “yanina wickmayer” yinina => hier is de –m voor direct ook een home directory –d maakt de naam voor de home directory en de –c als extra informatie in dit geval is dit de volledige naam en als laatste word de naam van de gebruiker meegegeven.

-g groupid

-u userid

-s andere shell dan default

useradd –D => toont een paar default variabele opties voor als er een account gemaakt word

userdel => verwijderd een gebruiker met userdel –r kan verwijder je de gebruiker maar ok de home directory van die persoon.

usermod => met dit commando kan je gegevens van een account aanpassen bijvoorbeeld de extra info wel natuurlijk met de letter –m –d of andere

usermod –d => kan je de naam van de homefolder van een persoon aanpassen.

usermod –L => maakt een wachtwoord niet actief zodat het account niet kan gebruikt worden dit kan je zien als een account gedisabelt is als er in de /etc/shadow file een !voor het versleutelt paswoord staat dat wilt zeggen dat het gedisablet is

usermod –s => hier kan je de shell van de persoon aapassen dus bv usermod –s /bin/bash laura dat veranderd /bin/ksh naar /bin/bash in dit voorbeeld

usermod –a – G => zo kan je personen aan een groep toevoegen. Dus bv usermod –a –G testgroep testgebruiker

passwd => hier kan je het commando van de huidige gebruiker mee aanpassen eerst één keer het oude en dan twee keer het nieuwe

chage => laat info over het account weten dat je intypte de datum dat het is aangemaakt de vervaldag wanneer het inactief is wanneer het account verloopt minimum dagen tussen voor nieuw passwoord de maximum dagen en de aantal dagen wanneer het paswoord verloopt. Bv chage –l harry

su => verandert van gebruiker

su –“gebruiker” => dit doet het zelfde als su maar nu gaat hij ook naar de homefolder van de gebruiker waar je naar toe wilt

Su root => gaat naar de gebruiker root

Su - => als je gewoon su – doet zonder een persoon mee te geven dan word er door de computer naar de root gebruiker geswitcht .

Mkdir /home/”gebruiekrsnaam” => normaal gezien maak je een homefolder aan door als je useradd doet de optie –m mee te geven dan word die aangemaakt maar je kan dat natuurlijk ook zelf doen maar het is wat meer werk bv mkdir /home/”gebruikersnaam/

chown “gebruikersnaam”: “gebruikersnaam” /home/”gebruikersnaam”

chmod 700 /home/“gebruikersnaam”

(chown om eigenaar van de map te worden en chmod om de juiste rechten te hebben op de map)

Chsh => zo kan je de login shell van een gebruiker aanpassen naar iets dat beter zou zijn.

Sudo su => hier switch je user naar de root gebruiker zolang je als deze gebruiker bent moet je nooit sudo voor een commando typen want je kan alles doen als root natuurlijk kan dit ook negatief zijn.

Groupadd => maakt een groep aan met een bepaalde naam zonder gebruikers in.

Groupmod -n => veranderd de naam van een groep dus groupmod –n nieuwtest test

Groupdel => verwijderd een groep

Groups => laat alle groepen zien waar de gebruiker toebehoord

Gpasswd => dit commodo heeft te maken met het aanpassen van een groep bv gpasswd –A “gebruikersnaam” “groep” dit maakt een persoon administrator van een groep. Bv gpasswd –a “gebruikersnaam” “groep” dit voegt een persoon toe aan een bepaalde groep

Vigr / vipw => laat de file /etc/group zien hier kan je die files ook aanpassen want hier staan alle groepen in de id’s en de personen die tot de groepen behoren.

Cut –d: -f1 /etc/passwd | column => laat alle gebruikers zien op je pc.

Chgrp => Hier mee verander je de groep owner van een file bv chgrp “groepsnaam” “filenaam”

Chown => Hier mee verander je de gebruiker van een bestand dus chown “gebruikersnaam” “filenaam” maar ook kan je met dit commando aan de file meerder groepen aan toe kennen bv chown “groepsnaam”:”eventuele groep” “file name”

Letters voor rechten van een bestand

- => een gewone file

d => een map

l => een link

p => een pipe

b => een block device

c => een karakter device

s => een socket

Use chmod => chmod kan je op twee manieren gebruiken de eerste manier

Gebruik van letters => hier type je de letters die je wilt toeven er zijn drie plaatsen waar je de rechten van het bestand kunt aanpassen

u = eigenaar van het bestand

g = de groep eigenaar van het bestand

o = alle andere die iets willen doen met het bestand

dan daarna de mogelijk heden die je kan verwijderen van de bovenstaande

r = je kan het bestand lezen

w = je kan in het bestand schrijven en dus aanpassen

x = je kan het bestand uitvoeren

een voorbeeld van een commando chmod u-rw, g-rw, o-r “filename” hier zeg je dat de eigenaar r en w rechten krijgt de groep ook r en w rechten krijgt en alle ander mogen enkel het bestand lezen

ook kan je een recht toevoegen aan bestand chmod o+w hier zeg je dat iedereen zijn rechten behoud en dat de others de rechten w bij krijgen boven hun rechten die ze al hadden.

De ander manier is met octale cijfers

Hier zeg je dat elke groep een getal voorstelt dus chmod 751 test hier zeg je dat de eigenaar alle rechten krijgt de groep r en x rechten krijgt en de others krijgen x rechten dit is eenvoudig uitte rekenen rwx staat eigenlijk voor 421 dus als j een rechte wilt voor een bepaalt iemand tel je het getal er bij op.

Umask => dit laat de default waarde zien die je pc helpt om als je een bestand aanmaakt de juiste default rechten er aan te geven op server is dit 0002 en dan krijgt je in een bestand –rw-rw-r—

Dus 664 bij een file is de hoogste rechten altijd maar 666 omdat een bestand nooit direct uitvoerrechten mag hebben dus als je 666 – 002 dan krijg je 664 dus bv enkel lees rechten voor iedereen is dan default bestand 666 en de uitkomst moet dan 444 zijn dus moet je een umask maken van 0222 dit doe je door gewoon umask 0222 te doen(umask kan enkel zo gebruikt worden voor files)

Mkdir –m => met de –m optie kan je direct als je een map aanmaakt er ook rechten aan geven bv mkdir –m 777 “testmap” dus je geeft iedereen alle rechten.

Aanmaken van een script file

vi script.sh

Dan er een bestand aangemaakt als je het ook nog wilt gaan uitvoeren waar een script voor dient moet je wel de file nog execute rechten geven dus chmod u+x script.sh

Dan kan je gaan typen in die file als je commentaar in het bestand wilt zeten die genegeerd word door de pc doe je dit door een #

In het begin van elk script type je altijd #!/bin/bash – zodat het bestand kan worden uitgevoerd. Daarna type je alle commando’s die je wilt in dat script en ga je uit het bestand :wq en voer je het uit door ./script.sh

Een voorbeeld voor als je iets kunt ingeven en die waarde dan terug word gebruikt

#!/bin/bash –

Echo –n “geef een bestaande map in”

Read variablenaam

Echo files/dirs uit $ variablenaam startedn met een d :

Ls $variablenaam | grep –i ‘d’

Theorie over linux belangrijke plaatsen

/ => dit is de root, de hoofdmap van het systeem

Root => dit is de user root, de super user die alles kan

/root => dit is de home-directory van de user root(deze staat niet zoals alle ander home-directory’s van andere gebruikers in de map /home)

Absoluut pad => dit is altijd een volledig pad dit start altijd met / bv /home/student/testfile

Relatief pad => dit is een verkort pad zodat je niet altijd onnodig een helleboel mappen moet afgaan en gewoon van uit je huidige map naar een ander plaats kunt gaan bv van /home/student moet je kan je direct aan testfile van /home is het cd /student en vanuit /home/student/desktop cd ..

/home

• Elke gebruiker krijgt zijn eigen map in de map /home

• Opslag voor persoonlijke mappen en bestanden per gebruiker

• Gebruiker Jan heeft als homefolder /home/jan

• Komt overeen met binnen Windows c:\Gebruikers

/root

• Homefolder van de gebruiker “root”

• NIET in /home

/bin &/sbin

• Locatie van Binairies (uitvoerbare bestanden)

• /bin: belangrijke (systeem) binaries

• Bijvoorbeeld: pwd, cat, cp,...

•/sbin: systeem binairies, staan in voor de werking van het OS,

meestal enkel uitvoerbaardoor root

•Bijvoorbeeld: reboot, shutdown, fsck

/usr

• Unix system resources

• Read only

• Bijkomende software binaries

• /usr/bin vult /bin en /sbin aan

• /usr/lib: libraries

• /usr/share/doc: documentatie

/etc

Etcetera – Editable Text Configuration

• Configuratie files

• Per toepassing

•/etc/skel: default files & mappen om een nieuwe homefolder te maken

/var

Variable data

• Belangrijkste is /var/log

• Systeemlogs

•Bv. /var/log/syslog

/opt

• Optional software

• Meestal leeg

• Wordt gebruikt om optional 3rd party software te installeren

• bv. LAMPP, eclipse, ...

/dev

• “Bestanden” in /dev staan niet fysiek op de schijf

• Alle hardware in je toestel

• bv. /dev/sda = harde schijf

• /dev/tty & /dev/pts

• Instanties van een terminal/console

• /dev/null: zwart gat

/media

• Voor het mounten van removable media

•bv. USB stick, camera, floppy, cdrom, ...

/mnt

• Standaard leeg

• Wordt gebruikt om (tijdelijke) lokale- en netwerkschijven aan te maken

• bv. Linken van een fileserver

/tmp

• Voor opslag van tijdelijke bestanden

• Bestanden die je wil bewaren hier NIET in opslaan

• Wordt leeggemaakt bij afsluiten

/etc/passwd

Bevat de gebruikersnaam, een x , een gebruikers id, een primaire groep, algemene info, de naam van de home directory en als laatse de login shell

ann:x:519:525:flute player: /home/ann:/bin/bash

root

Hoofdgebruiker kan alles doen.(heeft userid0)

/etc/default/useradd

In deze file zitten een paar default opties voor als er een account word bij gemaakt deze file kan ook getoond worden door useradd –D

/etc/shadow

In deze file staat alle gebruikers en hun wachtwoorden maar dan versleuteld.

/etc/login.defs

Hier staan een paar dingen in waar een wachtwoord zich moet aan houden bv max aantal dagen, minimum dagen, minimum lengte en, minimum tekens en nog veel.

/etc/group

Dit is een bestand waar alle groepen inzitten en de groepsiden de leden.

Algemene info over linux uitleg afkortingen

Linux is ontstaan als afgeleide van unix

De broncode van unix is vrij

Unix is de kern (=kernel) van een besturingssysteem maar ook alle daarmee samenhangede programma’s die het werken met dit besturingssysteem eenvoudiger maken.

De shell = command prompt de gegeven die het meest gebruikt word is de bash maar er zijn er ook andere zoals sh,ksh

GNU is ook een besturingssysteem zoals als unix maar is niet het zelfde het heeft wel dezelfde tools als unix en is ook gratis

GPL een samenvatting van allemaal software die gratis is ook de code van de programma’s is open van iedereen maar als je aanpassingen doet aan de originele code dan moet je wel jouw programma ook onder de samenhang van GPL zetten en mag er dus geen geld voor gevraagd worden

LTS editon => deze versie van linux word lang ondersteund door de makers van linux dit staat voor long term support en word 5 jaar ondersteund als je een versie van linux installeert zonder lts dan word die maar voor 9 maanden ondersteund. ( om de twee jaar nieuwe lts)

GUI => Graphical user interface dus alles wat visueel is

CLI => command line interface dus de bash , commandprompt

Sudo => super user do

APT => advanced packaging tool (de package management system van linux)

Repository => de databank van linux waar de software word gezet zodat er kan naar worden gezocht.

PPA => Personal Pacage Archive dit word gebruikt als er geen externe repositories gevonden worden door linux

Update => er word software die op de pc staat aangepast

Upgrade => het besturingssysteem word aangepast

Linux desktop info

Menubar => balk helemaal vanboven in beeld

Launcher => de balk links in beeld met de pictogrammen

Dash => het pictogram in de launcher helemaal vanboven (cirkel met 3 bolletjes)(vergelijkbaar met het startscherm van windows)

Nautilus bestandsbeheer => dit is het tweede icoontje van boven met een bestandkast die open gaat dit kan je vergelijken met de verkenner van windows

Workspaces => De extra 3 achtergronden van linux

Softwarecenter => hier kan je als je allemaal software voor je desktop downloaden zonder gebruik te maken van de terminal venster. (vergelijkbaar met app store)