RADBOUD UNIVERSITEIT NIJMEGEN
AFDELING STERRENKUNDE
JÖRG R. HÖRANDEL
ANNA NELLES

Programmeren 1 – 2012/13
Computer Practicum 1 – 12.11.2012

- Iedere student moet op **blackboard** geregistreerd zijn. Indien nog niet het geval voor volgend college registreren.
- Website voor informatie over het college: **http://particle.astro.ru.nl/goto.html?prog1213**
- Voor algemene vragen over de organisatie en coördinatie graag contact met Anna Nelles (a.nelles@astro.ru.nl) opnemen.
- Inleveren via mail bij jouw werkcollege assistent. De indeling wordt nog bekend gemaakt. **De opgaven van deze week hoeven niet ingeleverd te worden.**

1 Introductie

In de wetenschappelijke wereld worden de besturingssystemen Linux of UNIX heel veel gebruikt, tegenwoordig voornamelijk in de vorm van de afgeleide besturingssystemen Linux en OSX. In dit vak zullen jullie dus te eerst leren met dit soort besturingssysteem te werken. Omdat dit waarschijnlijk jullie eerste kennismaking met Linux is, geven we hieronder een korte inleiding, en een paar basiscommando's die je veel zult gebruiken. Voor deze hele tekst geldt: alleen de basiszaken worden uitgelegd, voor uitgebreidere voorbeelden en functionaliteit kun je terecht bij de assistenten voor dit vak. Vergeet ook niet dat er online enorm veel op te zoeken is over het gebruiken van Linux!

De doel van dit practicum is dat jullie gewoon alle commandos uitproberen en onthouden die in deze handleiding staan en een beetje met LINUX vertrouwt worden. **Wij gaan voor de volgende practica aannemen dat jullie een Linux computer kunnen gebruiken en met de commando's vertrouwd zijn.**

2 De Linux-interface

In het Huygensgebouw kun je op de computers in de terminalkamers bij het opstarten kiezen tussen 2 besturingssystemen: Windows en Linux. Hoewel je verbinding kunt maken met andere Linux-machines vanuit Windows (door bijvoorbeeld gebruik te maken van Xwin32), is het veel praktischer om direct onder Linux in te loggen wanneer je gebruik wilt maken van Linux-software die op andere systemen draait. Zorg er daarom voor, dat je voor deze oefening onder Linux ingelogd bent.

Wanneer je inlogt onder Linux, kom je in een grafische omgeving terecht analoog aan die van Windows. Hierbinnen kun je applicaties opstarten vanuit een drop-down menu, je kunt je systeem doorzoeken, systeeminstellingen wijzigen (wanneer je er de nodige permissies

voor hebt), en de folders en bestanden op het systeem navigeren. Dit alles is analoog aan hoe het bij Windows werkt. Voor deze oefening (en alle volgende practica) echter zullen we de commandoprompt gebruiken: dit houdt in dat je de instructies voor het systeem gaat invoeren via een terminalscherm, door deze commando's in te typen op het toetsenbord.

Je gebruikt de commandoprompt in eem omgeving van een terminal. Deze kan je in de computer kamer via de grafische menu opstarten. In andere LINUX systemen is de terminal gewoon het enige wat opent als je het systeem start.

Als je niet gewend bent aan een commandoprompt, kan het zijn dat je je in eerste instantie nogal beperkt voelt in je mogelijkheden met zo'n interface. Je hebt niet hetzelfde overzicht over je systeem zoals je dat hebt bij een grafische gebruikersinterface, waar je direct kunt zien welke bestanden en folders er op je schijf staan. Je ziet geen help-functie, en je hebt überhaupt geen idee welke taken je het systeem allemaal kunt laten uitvoeren. Dit alles is geen probleem, zodra je de eenvoudigste commando's eenmaal kent. Werken met een commandoprompt kan betekenen, afhankelijk van welke dingen je de computer precies wilt laten doen, dat je vele malen sneller en efficiënter kan werken dan vanuit een grafische omgeving. In deze oefening komen de volgende dingen aan bod qua gebruik van een commandoprompt:

- Het navigeren van de folders en bestanden op het systeem (`ls, cd, pwd, ~, /`)
- Het kopiëren, verplaatsen en verwijderen van folders en bestanden (`cp, mv, rm`)
- Het aanmaken, weergeven, en wijzigen van folders en bestanden (`cat, mkdir, touch, nano/vi/emacs`)
- Het maken van een verbinding met een andere computer (`ssh`)
- Handige tools en commando's (`grep, less/more, top, tail, fg, awk`)
- Het maken en runnen van scripts

Hieronder worden deze items stuk voor stuk behandeld.

2.1 Navigatie met de commandoprompt

Wanneer je een terminalvenster hebt geopend, zie je een nagenoeg leeg scherm met linksbovenin de prompt. De prompt is de plaats waarachter je je gewenste commando's intypt, en doorgaans laat de prompt zien waar in het bestandssysteem je op dat moment zit. Als ik bijvoorbeeld ingelogd ben op de computer genaamd 'Lilo' op de faculteit (Linux-systemen hebben over het algemeen namen die gekozen zijn door de systeembeheerder), zie ik in het terminalvenster de volgende prompt:

```
studentX@lilo2:~ $
```

Deze prompt geeft aan dat ik als gebruiker 'studentX' ingelogd bent op de machine 'lilo2', en dat ik me op dit moment in mijn homefolder bevind (de ~ is een afkorting voor je homefolder, voor mij is die voluit geschreven: '/home/studentX'). De prompt is afgesloten met een $.

Het bestandssysteem op een Linux-computer heeft een (ongeveer) gelijksoortige structuur als een Windows- bestandssysteem: het is een boomstructuur met folders die op zich weer andere folders en bestanden kunnen bevatten. Als je wilt weten waar je je bevindt in het

bestandssysteem (omdat het, bijvoorbeeld, niet in de prompt past) typ je in:

```
pwd
```

Dit is een afkorting van 'present working directory'. Leuk om te weten waar je zit, maar wat is hier te doen? Met het commando 'ls' krijg je een lijst te zien van de folders en bestanden die binnen deze folder staan. Maar let op: op deze manier zie je niet of iets een folder of een bestand is! Om dit onderscheid duidelijk te maken, kun je '`ls -l`' gebruiken. Een voorbeeld:

```
studentX@lilo2:~/NMDA/ex1$ ls -l
total 20
-rw-r--r-- 1 studentX nkstud 356 2007-09-19 15:21 alpha.m
-rw-r--r-- 1 studentX nkstud 1020 2007-09-19 15:20 primes.m
-rw-r----- 1 studentX nkstud 597 2007-09-19 15:18 question2.asv
-rw-r----- 1 studentX nkstud 749 2007-09-19 15:22 question2.m
drwxr-xr-x 2 studentX nkstud 4096 2011-12-09 11:22 testdir
studentX@lilo2:~/NMDA/ex1$
```

Bovenaan zie je de prompt ('studentX@lilo2:~/NMDA/ex1$'). studentX zit dus in de folder 'ex1' binnen de folder 'NMDA' die op zijn beurt weer in de homefolder staat. De regel 'total 20' heeft betrekking op de schijfruimte (aantal blocks) die gebruikt wordt door de inhoud van deze folder. Elke regel daaronder beschrijft een folder of bestand. Bijvoorbeeld de eerste regel bestaat uit de volgende informatie:

- permissies ('-rw-r–r–')
- aantal links naar item ('1')
- eigenaar ('studentX')
- groep van eigenaar ('nkstud')
- grootte in bytes ('356')
- datum en tijd van laatste wijziging ('2007-09-19 15:21')
- naam van het bestand of de folder ('alpha.m')

Als we naar het eerste gedeelte van zo'n regel kijken, zien we de permissies voor dat bestand of die folder. Het allereerste teken op de regel vertelt ons of het een bestand is (dan staat er een '-') of een folder (dan is het een 'd'). De tekens direct daarachter moet je opdelen in groepjes van 3. Bij het bestand 'alpha.m' bijvoorbeeld zijn die 3 groepjes respectievelijk 'rw-', 'r–' en 'r–'. Die tekens geven aan wie welke permissies heeft voor dat bestand. Het eerste groepje tekens ('rw-') heeft betrekking op de eigenaar van dat bestand (in dit geval ben ik dat zelf, 'studentX'). Er staat dat ik het bestand mag lezen ('r' van 'read'), ik mag het wijzigen ('w' van 'write'), maar ik mag het niet uitvoeren (dan zou er ook een 'x' moeten staan, van 'execute'). Het tweede groepje tekens slaat op alle gebruikers binnen dezelfde groep als de eigenaar (in dit geval alle natuurkunde-studenten, 'nkstud'). De leden van deze groep mogen het bestand alleen lezen, maar niet wijzigen of uitvoeren. Het derde groepje van 3 tekens geeft de permissies voor de rest van de wereld aan (dus mensen die buiten de groep van de eigenaar vallen): deze mogen het bestand ook alleen maar lezen. Het tweede item op de regel, het aantal links naar het bestand of de folder, is hier niet relevant en slaan we even over. Het derde item is

de eigenaar, en het vierde item is de groep waar de eigenaar in zit. De overige items behoeven geen uitleg. Het commando 'ls' heeft nog veel meer mogelijkheden: zo kun je bijvoorbeeld ook kijken wat er in een andere folder zit dan degene waar je nu in zit. Dit doe je simpelweg door de folder-aanduiding erachter te zetten: 'ls -l /home/freddie' laat bijvoorbeeld zien wat er allemaal in /home/freddie staat. Tenminste: als die folder wel door jou geopend mag worden!

Nu weten we dus waar we zijn ('pwd') en wat er in de huidige folder staat ('ls' of 'ls -l'). Om van folder te veranderen, gebruik je het commando 'cd' gevolgd door de folder waar je naartoe wilt. Stel dat ik in de folder /home/studentX sta. Enkele voorbeelden:

Commando	Actie	Resultaat
cd ..	ga een folder terug	/home
cd spul	ga een folder dieper	/home/studentX/spul
cd spul/enzovoort	ga meerdere folders dieper	/home/studentX/spul/enzovoort
cd ../freddie	ga een folder terug, en van daaruit een folder dieper	/home/freddie
cd /bladiebla/iets	ga direct naar de genoemde folder	/bladiebla/iets
cd /	ga naar de root-folder (naar het 'totaaloverzicht')	/
cd ∼	ga naar je eigen homefolder (scheelt typwerk)	/home/studentX

In het bovenstaande tabelletje zie je dus dat met de aanduiding '..' de folder bedoeld wordt die op zijn beurt de huidige folder bevat. Zo heeft ook de folder waar je op dat moment in staat zijn eigen afkorting: deze wordt ook aangeduid met een punt ('.'). Als je dus in een commando './test.txt' typt, bedoel je het bestandje 'test.txt' dat in de folder staat waar je je op dat moment bevindt.

2.2 Kopiëren, verplaatsen, hernoemen, verwijderen

Het kopiëren, verplaatsen, hernoemen en verwijderen van bestanden en folders onder linux kan met enkele eenvoudige commando's. **PAS OP: deze commando's vragen niet standaard of je de opdracht echt wilt uitvoeren, maar doen meteen hun werk! Een fout is snel gemaakt, kijk dus goed uit wanneer je deze opdrachten uit wilt voeren en lees je opdracht goed na voordat je 'm uitvoert.**

Het kopiëren van bestanden doe je met het commando 'cp'. Je gebruikt het commando standaard met 2 dingen erachter: het bestand of de folder dat je wilt kopiëren, en het doelbestand of folder waar het naartoe gekopieerd moet worden. Voorbeelden:

Commando	Resultaat
cp test.txt anderetest.txt	Kopieer de inhoud van 'test.txt' naar 'anderetest.txt' in de huidige folder
cp test.txt stuff/test.txt	Kopieer de inhoud van 'test.txt' naar een bestand met dezelfde naam, een folder dieper
cp test.txt ../fred.txt	Kopieer de inhoud van 'test.txt' naar 'fred.txt', een folder hoger
cp *.txt backups/	Kopieer alle txt-bestanden naar de subfolder 'backups' (met dezelfde namen als hun originelen)
cp -r folder1/ folder2/	Kopieer folder1 met inhoud naar folder2 ('-r' staat voor 'recursive': alle subfolders worden meegenomen)

Let op: als het doelbestand al bestaat, wordt dit zonder pardon overschreven en is het oorspronkelijke bestand met die naam dus weg. In de bovenstaande voorbeelden komt ook een asterisk voor: dit is een 'wildcard', een tekentje dat staat voor 'alles'. '*.txt' betekent dus alle bestanden met extensie '.txt', terwijl bijvoorbeeld 'freddie*' alle bestanden en folders beginnend met 'freddie' betekent. Een uitdrukking zoals 'a*z' betekent dan alle bestanden waarvan de naam met 'a' begint, en eindigt met 'z'. Qua wildcards is het vraagteken '?' ook de moeite waard: dit betekent namelijk 'een enkel willekeurig karakter'. 'freddie?' betekent dus alle bestanden met de naam freddie gevolgd door een enkel willekeurig teken, zoals 'freddie1' of 'freddieq'. 'freddie10' hoort daar dus niet bij!

Het hernoemen en verplaatsen van bestanden werkt op dezelfde manier als het kopiëren. Voor hernoemen gebruik je het commando 'mv' (voor 'move'), en voor verplaatsen hetzelfde commando (het is namelijk eigenlijk dezelfde bewerking). 'mv test.txt test1.txt' hernoemt je bestand naar 'test1.txt', terwijl 'mv test.txt stuff/test.txt' je bestand verplaatst naar de subfolder 'stuff'.

Het verwijderen van bestanden en/of folders doe je met het commando 'rm' (van 'remove'). 'rm' gevolgd door het doelbestand verwijdert het direct. Als je een folder met zijn inhoud wilt verwijderen, gebruik je 'rm -r' gevolgd door de naam van de folder. Hiermee wordt recursievelijk de hele folder en alles erin verwijderd. **LET OP: er wordt dus niet aan je gevraagd of het het zeker weet! Kijk dus uit.**

2.3 Het aanmaken, weergeven, en wijzigen van folders en bestanden

Om snel te zien wat er in een bestand staat, kun je het commando 'cat' gebruiken. 'cat' gevolgd door de bestandsnaam geeft direct de inhoud van het bestand weer in je terminalvenster. Let op: dit kan dus een hele hoop zijn voor grotere bestanden! Verderop in deze instructie laten we zien hoe je zo'n output kunt gebruiken of er doorheen kunt bladeren.

Het commando 'mkdir' gevolgd door de gewenste naam maakt een nieuwe folder aan.

'touch' gevolgd door de gewenste naam maakt een nieuw bestand aan. Als het bestand al bestaat, wordt alleen het tijdstip van de laatste wijziging van het bestand naar de huidige tijd gezet, de inhoud van het bestand blijft bestaan.

Om bestanden aan te passen, zijn er diverse teksteditors beschikbaar. De meest eenvoudige is wellicht 'nano'. Wij gaan nano in het volgende hoofdstuk verder uitleggen en de functionaliteit beschrijven. Andere editors waar je uit kunt kiezen zijn vi en emacs. Welke je gebruikt hangt alleen af van je persoonlijke voorkeur. De keuze van een teksteditor is vaak een grote discussie om dat iedereen zijn eigen voorkeur heeft. Belangrijk voor deze cursus is dat jullie leren een editor te gebruiken die binnen een terminal werkt.

2.4 Editor

Nano is een teksteditor die binnen het terminalscherm werkt en waarmee je alle basisfunctionaliteit ter beschikking hebt. Om nano te gebruiken type je **nano** gevolgd door de naam van het bestand dat je wilt bewerken (dit kan ook een nieuw bestand zijn). In de editor heb je de beschikking over commando's die onder in beeld genoemd staan. Let op de notatie: '^ X' betekent 'Control-X'.

Met **Control-G** kan je de hulp oproepen waarin je meer commando's kan vinden. De eerste resultaten van dit commando staan ook in de volgende tabel:

Commando	Beschrijving
Ctrl G (F1)	Display this help text
Ctrl X (F2)	Close the current file buffer, exit from nano
Ctrl O (F3)	Write the current file to disk
Ctrl J (F4)	Justify the current paragraph
Ctrl R (F5)	Insert another file into the current one
Ctrl W (F6)	Search for a string or a regular expression
Ctrl Y (F7)	Move to the previous screen
Ctrl V (F8)	Move to the next screen
Ctrl K (F9)	Cut the current line and store it in the cutbuffer
Ctrl U (F10)	Uncut from the cutbuffer into the current line
Ctrl C (F11)	Display the position of the cursor
Ctrl T (F12)	Invoke the spell checker, if available
Ctrl _ (F13) (M-G)	Go to line and column number
Ctrl \(F14) (M-R)	Replace a string or a regular expression

Het verschil tussen een editor zo als nano en andere grafische tekstprogramma's (Word, Open Office, Notepad, etc.) is dat tekstprogramma's vaak ook een formattering in de bestand opslagen, die je niet kan zien. Als je dus een eigen programma schrijft en een grafische programma gebruikt heb je heel veel dingen in je file staan die ja da eigenlijk niet wil hebben en dus kan het programma niet uitgevoerd worden. Dus gebruik voor alle opgaven in programmeren een editor zo als nano, vi or emacs.

2.5 Het maken van een verbinding met een andere computer

Via de commandoprompt kun je direct inloggen op een andere computer dan de machine waar je achter zit. Hiervoor gebruiken we de tool 'ssh' (voor 'Secure SHell'). Om in te loggen op een ander systeem typ je 'ssh -X gebruikersnaam@systeemnaam'. De '-X' optie (let op de hoofdletter) zorgt ervoor dat je ook grafische vensters kunt openen vanaf de andere machine (X11-forwarding). Dit is bijvoorbeeld nodig om nedit te kunnen gebruiken als editor, of om ds9 op te starten (dit is een programma dat vaak samen met IRAF gebruikt wordt, om bijvoorbeeld foto's weer te geven).

Als ik bijvoorbeeld in wil loggen op de machine 'stitch' typ ik in: 'ssh -X studentX@stitch.science.ru.nl', waarna ik gevraagd word om mijn wachtwoord. Je gebruikersnaam en wachtwoord zijn gewoon hetzelfde als wanneer je direct achter een computer in zou loggen in het Huygensgebouw. In dit voorbeeld gebruik ik de 'volledige' systeemnaam van stitch (met het subdomein 'science.ru.nl' erbij), omdat het op deze manier niet uitmaakt vanaf welk netwerk ik inlog. Als je al in het science-netwerk ingelogd bent kun je ook alleen de computernaam gebruiken, zonder het subdomein (dus met 'studentX@stitch'). Nu je ingelogd bent op een ander systeem kun je alles doen wat je normaal ook op dat systeem zou kunnen: van folder wisselen, programma's draaien, in je homefolder rondsnuffelen, enzovoort. Als je de verbinding naar de andere computer wilt stoppen, typ je 'exit'.

2.6 Handige tools en commando's

Linux heeft een enorm scala aan andere commando's en tools die je vanaf de prompt kunt gebruiken. Enkele hiervan worden hier even kort aangestipt.

Met het commando 'grep' kun je zoeken naar specifieke uitdrukkingen in een lijst met tekst. Het is erg handig om te gebruiken in combinatie met 'cat', omdat je dan kunt zoeken naar uitdrukkingen in een bestand. Het combineren van commando's is te doen met een zogenoemde 'pipe': dat is het doorsturen van de output van een programma naar de input van een ander programma. Dit doe je in een terminal met een verticale streep ('|'). Enkele voorbeelden:

Commando	Resultaat
cat fred.txt | grep ding	Stuur de inhoud van fred.txt door naar 'grep', zoek naar regels met de uitdrukking "ding" erin, en geef die weer.
cat fred.txt | grep -v ding	Zelfde als hierboven, maar geef nu regels weer ZONDER "ding" erin.
cat fred.txt | grep ^ding	Geef regels uit fred.txt weer die "ding" aan het BEGIN van de regel hebben staan.
cat fred | grep ding$	Geef regels uit fred.txt weer die "ding" aan het EIND van de regel hebben staan.
ls -al | grep ^d	Geef alle items weer in de resultaten van 'ls -al' die beginnen met een 'd' (dus: alle folders!)

Een andere handige tool is het doorsturen van de output van een programma naar een bestand: dit doe je met een '>'-teken. Wanneer je bijvoorbeeld intypt: 'ls > filesandfolders.txt' wordt er een bestand aangemaakt met daarin de output van het 'ls' -commando. Door 'ls >> filesandfolders.txt' te typen, dus met een dubbele pijl, wordt de output van het 'ls'-commando toegevoegd aan het bestand, dus de oude inhoud ervan blijft dan ook bewaard. Een enigszins omslachtige manier om een bestand te kopiëren is dan bijvoorbeeld 'cat fred.txt > fred2.txt'.

Wanneer je de inhoud van een bestand even scherm per scherm wilt bekijken, kun je de tools 'less' en 'more' gebruiken. Door bijvoorbeeld 'less logfile.txt' te typen krijg je de inhoud van 'logfile.txt' te zien, en kun je (onder andere) met de cursortoetsen naar beneden en weer omhoog scrollen door het bestand (door 'q' te typen krijg je de prompt weer terug). 'more logfile.txt' werkt bijna net zo, maar met andere toetsen om te navigeren. Deze tools kun je ook gebruiken in combinatie met 'cat': bijvoorbeeld door te typen 'cat logfile.txt | less'. Het commando 'tail <bestandsnaam>' laat je (standaard) de laatste 10 regels van een bestand zien, wat handig kan zijn bij een logbestand dat je wilt controleren op recente activiteit.

Met het commando 'top' bekijk je de actieve processen op het systeem, hoeveel geheugen/processoraandeel ze gebruiken, en wie ze gestart heeft. Handig als je vermoedt dat er nog een programma actief is, maar je weet niet welke. Het commando 'ps' kun je ook gebruiken om eenmalig een lijst met alle actieve processen te krijgen, zodat je bijvoorbeeld een procesnummer kunt opzoeken om een programma dat nog in de achtergrond draait stop te zetten (met 'kill').

2.7 &, fg, Ctrl-Z en Ctrl-C

Als je een programma uit wilt voeren dat tijd nodig heeft om zijn werk te doen (bijvoorbeeld het verwerken van een hoop data) neemt dat de terminal in beslag als je het gewoon start -

je moet dan wachten totdat het programma klaar is voordat je weer nieuwe commando's kan invoeren. Dit kun je oplossen met de ampersand ('&'): door je commando af te sluiten met dit tekentje wordt het gestart, maar draait het in de achtergrond terwijl jij de prompt kunt blijven gebruiken. Als je bijvoorbeeld 'firefox' intypt aan de prompt, start je de browser op - maar terwijl die actief is, zit de prompt vast. Wanneer je echter 'firefox &' invoert, start de browser normaal op en krijg je ook weer de beschikking over de prompt.

Wanneer je een programma hebt gestart zonder '&' maar je wilt de prompt weer kunnen gebruiken terwijl het programma nog actief is, kun je in het terminalvenster Control-Z indrukken om zo het programma op pauze te zetten. Je hebt dan de prompt weer terug, maar het programma is gepauzeerd en doet dus niets. In het terminalvenster komt dan zoiets te staan als '[2]+ Stopped firefox'. Wanneer je het programma weer wilt hervatten typ je in 'fg <nummer>', waarbij <nummer> het getal is dat tevoorschijn kwam toen je het programma op pauze zette.

Het onderbreken - en dus stoppen - van een lopend programma dat je prompt in beslag neemt kun je doen door op Control-C te drukken in het terminalvenster. Je krijgt de prompt dan terug, en het actieve programma wordt afgebroken. Dit is handig om een script stop te zetten dat je niet meer nodig hebt, maar ook om een lang commando dat je in hebt getypt weg te gooien (nog voordat je op 'Enter' hebt gedrukt) als het toch niet hetgene is wat je wilde doen.

Een tool die nog een korte aanbeveling verdient is 'awk'. Dit is een zeer krachtige tool om data- en tekstbestanden te manipuleren. Hier bieden we geen handleiding voor het gebruiken van 'awk', maar online zul je vele voorbeelden kunnen vinden van handige dingen die je ermee kunt doen. Een klein voorbeeld: stel dat je een bestand hebt ('freddie.txt') met daarin 2 kolommen van getallen. Je wilt een bestand aanmaken waarin slechts een kolom getallen staat, namelijk het product van de 2 kolommen in het eerste bestand. Met 'awk' doe je dit als volgt: "awk ' print $1*2$ ' < freddie.txt> output.txt". Met de pijl naar links ('<') geef je de inhoud van 'freddie.txt' als input aan awk, en vervolgens geef je het resultaat weer door naar het nieuwe bestand 'output.txt'. 'awk' kan nog enorm veel meer dingen - maar dit voorbeeld laat al zien dat het aanmerkelijk sneller is dan het opstarten van excel, het importeren van je tekstbestand, het berekenen van het product en het exporteren van het resultaat!

2.8 Scripts

Een van de krachtigste dingen die je kunt doen onder Linux is het automatiseren van je commando's. Dit kun je doen met zogenaamde shell-scripts. Een shell-script in zijn meest eenvoudige vorm is niets anders dan een tekstbestandje met daarin een lijstje met commando's die je graag achter elkaar wilt laten uitvoeren door het systeem. Zo'n script kan er bijvoorbeeld zo uit zien:

```
#!/bin/bash

cp /home/studentX/data.txt /home/studentX/results/data.txt
/home/studentX/results/data-analysis data.txt
pnmtojpeg output.ppm > output.jpg
rm output.ppm
```

De eerste regel van dit script ('#!/bin/bash') vertelt Linux met welke shell we het script willen laten uitvoeren. De regels die daarna komen zijn gewoon commando's die achter elkaar

uitgevoerd worden: eerst wordt er een databestand gekopieerd naar een subfolder, daarna wordt een programma opgestart dat met dat databestand aan de slag gaat, vervolgens wordt de output daarvan omgezet naar een jpg-plaatje en tot slot wordt het tussenresultaat verwijderd.

Zo'n script als hierboven kun je gewoon met een tekst-editor maken. Om je script ook uitvoerbaar te maken, moet je de permissies ervan aangepast hebben. Stel dat je een script hebt gemaakt dat 'mijnscript.sh' heet. Om het mogelijk te maken om het script uit te voeren typ je dan in: 'chmod 700 mijnscript.sh'. 'chmod' is een commando waarmee je de permissies van bestanden kunt wijzigen (daarvoor moet je wel de eigenaar zijn van het bestand). De code '700' betekent dat je jezelf alle permissies geeft (7 betekent dan lezen, schrijven, en uitvoeren) en alle anderen alle permissies ontneemt (ze mogen niets met het bestand doen,en ze kunnen er niet in kijken). Nu is je script klaar om uitgevoerd te worden, en kun je het runnen door in te typen: `./mijnscript.sh`.

Scripts worden pas echt handig als ze acties kunnen ondernemen die afhangen van wat er eerder gebeurt: als het script een bestand bijvoorbeeld ergens niet aantreft kun je het een waarschuwing laten geven, of het script kan alsnog het bestand ergens anders vandaan proberen te halen. We zullen hier geen voorbeelden van extreem ingewikkelde scripts gaan geven, maar wederom zijn er online talrijke voorbeelden te vinden van scripts die nuttige taken kunnen automatiseren.

Dit is de eerste stap in programmeren. Eigenlijk is zo een shell script een eerste programma. Maar dan niet in de 'taal C', die wij gaan gebruiken maar in de taal 'bash'. (Zo noem je de taal die je in een terminal gebruikt.)
Wij gaan in de loop van deze cursus meer leren over programma's en hoe je ingewikkeldere dingen kan programmeren.

3 Oefening

Hier volgt even een hele korte oefening in het werken met Unix. De commando staan hier nog even bij. Probeer de oefening ook een keer te doen zonder naar de commando's te kijken.

- Open vanuit een xterminal een nieuwe xterminal, met een zijbalk en een geheugenbuffer van 500 lijnen voor de vorige tekst: type `xterm -sb -sl 500 &`. Commando opties geef je in Unix aan met een `-` er voor (zoals '`-sb`' wat aangeeft dat je een scrollbar er bij krijgt. De `'&'` achter het commando geeft aan dat je het commando in de achtergrond wilt runnen, dwz. in het scherm waar je het hebt ingetypt krijg je de prompt weer terug en kun je door werken. Als je deze `'&'` niet mee geeft, krijg je geen prompt terug en is je originele xterminal 'bevroren'. Wil je dan alsnog verder werken dan typ je in de originele window 'CRTL-Z' in, waarmee het nieuwe commando 'suspend'. Hiermee bevries je je *nieuwe* window, maar kun je in het oude wel doorwerken. Wil je nu in beide kunnen werken typ dan in het oude window nog eens '`bg`' (voor 'background') en je kunt weer in beide werken. Mocht je een scherm of proces uit de achtergrond terughalen dan kan dat met het commando '`fg`' in het oude window.
- Waarschijnlijk ben je in het nieuwe window al in je home-directory, maar mocht dat niet het geval zijn ga daar dan naartoe met: `cd`
- Maak in je home-directory een subdirectory 'test' aan: `mkdir test`

- Ga naar deze sub-directory: `cd test`
- Open vanuit deze directory een file die je wilt bewerken met de editor 'nano': `nano probeer.txt &`. Unix/Linux werkt net als windows met file-extensies, maar in Unix/Linux is er geen limiet op het aantal karakters voor *of achter* de punt, en ook niet aan het aantal punten. Een filenaam zoals 'probeer.observationele_sterrenkunde.textfile.eerste.praktikum' is in Unix/Linux prima mogelijk. Probeer het maar eens!
- In de nano editor typ wat tekst in de file die je net hebt gemaakt. Probeer ook de andere commando's van nano om met nano beter kennen te leren. Wij verwachten dat je in de volgende practica makkelijk met nano kan werken.
- Check in je xterminal of de file nu inderdaad bestaat: `ls`. Als je de filelijst wilt sorteren van oud naar nieuw is dat: `ls -lrt`. Hier zie je dat je meerdere opties ook achter elkaar achter één '-' kunt plaatsen. De eerste 'l' geeft een gedetailleerde lijst, de 't' geeft de tijdsordening, en de 'r' geeft een omgekeerde volgorde aan (van 'reverse').
- Sluit je nano window af.
- Gooi in je directory de gemaakt file weg: `rm probeer.txt`.
- Check of je file inderdaad weg is: `ls .` In Unix kun je ook *wildcards* gebruiken. Als je bijvoorbeeld alle files met een naam wilt zien die begint met een 's', geef je het commando: `ls s*`. De *wildcard* mag ook vooraan staan!, of zelfs twee. Als je alles wilt weten over de files met ergens 'txt' in de naam: `ls -lart *txt*` **(Let op: het meest gevaarlijke commando ooit in Unix/Linux is: `rm *.*`. Gebruik dit alleen als je werkelijk weet dat je alles in de huidige directory en alle subdirectories wilt weggooien. Unix/Linux maakt geen reserve bestanden en vraagt niet om bevestiging. Weg is weg!**
- Je nieuwe directory is nu leeg.
- Ga terug naar je homedirectory: `cd` , of `cd ..`. Met die twee punten ga je naar een directory die één niveau hoger ligt dan de huidige. Je kunt ook verder omhoog door bv. `cd ../..` te gebruiken.
- Je kunt nu de directory 'test' weggooien met `rmdir test`.

Een speciale opmerking voor het sluiten van vensters! In windows gebeurt dit door op het $\times$-je rechtsboven in het scherm te klikken. Dit werkt over het algemeen in UNIX ook, maar het kan gebeuren dat het programma nog blijft draaien. Dus als er een andere optie is om een programma af te sluiten, kies hiervoor.