Wiskunde en Computers: UNIX/Linux

Ben Polman, Bernd Souvignier, Wieb Bosma

Najaar 2008

Hoofdstuk 1

Inleiding

Een besturingssysteem is het interface tussen de hardware en de gebruiker. Een computer bestaat typisch uit een processor, geheugen, harde schijf, toetsenbord, muis en een scherm. Het besturingssysteem regelt de interactie tussen deze onderdelen en het vertaalt de abstracte commando's die een gebruiker geeft (bv. *kopieer een bestand*) naar de concrete hardware operaties. Het besturingssysteem dat we zullen gaan gebruiken heet **UNIX**. Het is begin jaren zeventig ontwikkeld bij Bell Labs en heeft sindsdien een enorme ontwikkeling doorgemaakt. De belangrijkste kenmerken zijn:

multitasking: op ieder moment kunnen er meerdere programma's actief zijn, het besturingssysteem verdeelt automatisch de beschikbare processortijd over de actieve programma's zodat ieder programma met regelmaat aan de beurt komt.

multiuser: er kunnen meerdere gebruikers (ook één gebruiker meermalen) tegelijkertijd ingelogd zijn op één computer.

hiërarchisch filesysteem: (zie §3)

portabiliteit: Unix is beschikbaar voor een groot aantal verschillende hardware platforms.

Unix bestaat uit drie stukken, de *kernel*, de *shell* en een grote verzameling *utilities* (hulpprogramma's). De kernel regelt de verdeling van *resources* (processortijd, geheugen, diskgebruik) over de verschillende taken, beheert de toegang tot data en bestanden en handelt alles af wat met de hardware te maken heeft. De **shell** is het belangrijkste Unix programma voor de gebruiker, je geeft commando's aan de shell die deze interpreteert en vervolgens uitvoert. Er zijn hulpprogramma's voor allerlei veel voorkomende taken, bestandsbeheer, bewerken van teksten, printen, etc.

Unix is ontwikkeld in een tijd dat terminals beschikten over hooguit een scherm geschikt voor 25 regels van 80 karakters, vergelijkbaar met een DOS-scherm op PC's. Vandaar de nadruk op de shell, het is een commando-regel geöriënteerd interface. In de jaren '80 kwam er een belangrijke verandering, het **X-window** systeem ontwikkeld aan het MIT[1] werd een standaard binnen de Unix wereld. Dit betekende een verschuiving van op karakters-gebaseerde terminals naar grafische terminals met een *windowsysteem* bovenop Unix. X of X11 zoals het meestal aangeduid wordt is bijzonder geschikt voor gebruik in een netwerkomgeving. Het stelt

[1] Massachusetts Institute of Technology

een gebruiker in staat om allerlei programma's op verschillende computers te laten draaien en de uitvoer in verschillende windows op zijn eigen lokale computerscherm te laten verschijnen.

Dit laatste gevoegd bij de grote stabiliteit van Unix heeft ervoor gezorgd dat de combinatie Unix/X11 een van de meest gebruikte computeromgevingen is geworden binnen universiteiten en grote onderzoeksinstituten. Door de ontwikkeling van **Linux**, een public-domain versie van Unix voor PC's, is Unix in de laatste jaren ook voor veel PC-gebruikers het besturingssysteem van keuze geworden. Ook grote bedrijven of organisaties overwegen nu om van MS-Windows naar Linux over te stappen (of zijn al overgestapt).

Meer informatie

Voor meer informatie, zie

`http://curser.science.ru.nl/content-e/pub_NWI/`
`Inleiding_Computergebruik_1180512141473/index.htm`

Hoofdstuk 2

Inloggen

2.1 PC's

We zullen werken op PC's, meestal in terminalkamer HG00.023, of in HG03.761 in de gang bij wiskunde. In deze terminalkamers vind je PC's met een versie van Fedora-Linux. In HG00.023 heb je de keuze om onder Linux of onder MS Windows op te starten. Voor dit vak moet je altijd Linux kiezen.

Als je inlogt zie je een dialoogbox met `Welcome to` *computernaam*. De eerste keer dat je inlogt moet je door als login *session* **KDE** kiezen. De K-Desktop-Environment is een soort Linux versie van het X11 windowing systeem.

2.2 Wachtwoorden

Ik wil er hier op wijzen dat je login strikt persoonlijk is, wees dus voorzichtig met je wachtwoord en wijzig dat ook met enige regelmaat. Een **wachtwoord** moet uit minstens 6 tekens bestaan, en alleen de eerste acht zijn van belang. Daarbij gelden de volgende regels:

- Het wachtwoord moet minstens twee alfabetische karakters bevatten en minstens één ander karakter (cijfer, leestekens, etc.).
- Het mag niet gelijk zijn aan je loginnaam of door rondschuiven of omdraaien van je loginnaam zijn afgeleid. Hoofd- en kleine letters tellen hierbij als gelijk, terwijl het wachtwoord zelf gevoelig is voor hoofd- en kleine letters.
- Een nieuw wachtwoord moet op minstens drie posities verschillend zijn van het oude.

Enkele tips voor het kiezen van een wachtwoord:

- Kies een gemakkelijk te onthouden woord en maak daar expres een of twee typefouten in, bij voorkeur met gebruik van hoofdletters en kleine letters en speciale tekens als spaties, sterretjes, dollars, etc. Typische (en daarom ook niet zo veilige) operaties zijn bijvoorbeeld het vervangen van een `i` door `!` of van een cijferwoord door het cijfer. Je zult dus bijvoorbeeld vanuit het woord *machtiging* het wachtwoord `m8ig!nG` kiezen.
- Kies een bestaand woord, maar voeg daar*in* enkele extra tekens toe. Letters of (vooral) cijfers toevoegen aan het begin of het einde van het woord is niet voldoende! Zo maak je uit het woord *poes* misschien het wachtwoord `p3o1e4s1`.

- Neem een willekeurige combinatie van letters, cijfers en symbolen en bedenk daar een ezelsbruggetje voor, bijvoorbeeld een kort zinnetje of rijmpje. Als het resultaat ook nog min of meer uit te spreken is, dan vereenvoudigt dat het onthouden en intypen ervan. Andersom kun je ook met een langer woord beginnen en daar letters weglaten (bijvoorbeeld de klinkers) of vervangen. Uit *ezelsbrug* kan zo het wachtwoord `1zls?brg` worden.

Het wijzigen van je wachtwoord gebeurt op de faculteit NWI op een speciale webpagina van de afdeling C&CZ (Computer- en Communicatiezaken), namelijk de **Doe-Het-Zelf** pagina `http://dhz.science.ru.nl`. Op deze pagina log je in met je facultaire login-naam met bijbehorend wachtwoord en vindt dan aan de linkerkant in het `Verander:` menu de optie `wachtwoord`. Hier wordt je gevraagd je nieuwe wachtwoord twee keer in te tikken, door op de `Opslaan` knop te klikken wordt het opgeslagen.

Deze manier om het wachtwoord te wijzigen heeft het voordeel dat het automatisch voor alle computers in alle *domeinen* van de faculteit NWI wordt gewijzigd. (Een domein is een deel van het netwerk van computers; de computers van de subfaculteit wiskunde zoals `netelblom` en `bommel` zitten bijvoorbeeld in het `math`-domein, maar je hebt ook toegang tot andere computers van de faculteit NWI.

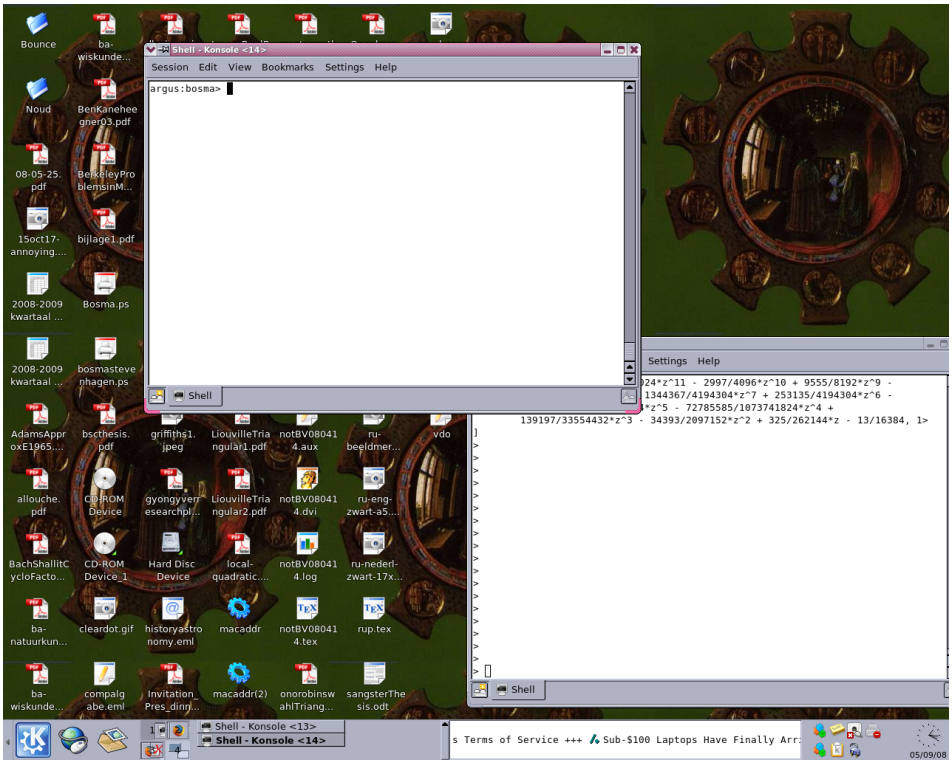
Meer informatie

Zie ook

`http://curser.science.ru.nl/content-e/pub_NWI/`
`Inleiding_Computergebruik_1180512141473/index.htm`

2.3 De desktop

Na inloggen zag mijn scherm er ongeveer zo uit: Hierop zie je de volgende onderdelen



Figuur 2.1: mogelijke situatie na inloggen

- **shell:** Een (hier grotendeels wit) schermpje met daarin het belangrijkste Unix commando, de *shell*, vergelijkbaar met een dosbox onder MS-Windows. De shell is het interface tussen de gebruiker en het besturingssysteem. Je geeft opdrachten aan de shell die deze interpreteert en vervolgens uitvoert. Unix commando's worden altijd afgesloten met een ⟨`Return`⟩, pas dan wordt de invoer door de shell geïnterpreteerd.
- **achtergrond:** Hier een afbeelding van een schilderij van Vermeer.
- **iconen:** Een aantal iconen op het *bureaublad*, de **Desktop**, die verwijzen naar documenten die daar staan.
- **K-menu:** Linksonder; met je muis kun je hier makkelijke toegang krijgen tot veelgebruikte programma's.
- **meer iconen:** op de balk onderaan zie je meer iconen van programma's; ondere andere een lijst van actieve programma's op je huidige Desktop, naast (in dit geval) een overzicht van de 4 verschillende Desktops die hier actief zijn. In het Control Center (zie onder) kun je dat aantal zelf kiezen, en je verandert van Desktop door op dit overzicht te klikken.

Het windowing systeem waar we mee werken zorgt ervoor dat meerdere programma's tegelijkertijd gebruik kunnen maken van één beeldscherm. Daarbij kunnen die programma's op verschillende computers draaien.

Het belangrijkste programma uit het K-menu waar we in het begin mee te maken hebben is het *Control Center*. Hierin kun je allerlei belangrijke instellingen veranderen. We zullen hier later uitgebreider op ingaan

De muis drie knoppen, die allemaal verschillende functies hebben afhankelijk van de plaats (window of deel van window) waar je klikt, dubbelklikt of de knop ingedrukt houdt. Voorlopig volstaat de linkermuisknop, de knop waarmee vrijwel alle selecties gemaakt worden.

Alhoewel we voorlopig nog niet uitloggen hier toch maar vast de procedure, klik met de linkermuisknop in het *K-menu*, er verschijnt een *popup menu* met als laatste entry `Log Out`; klik hier met je linkermuisknop en je krijgt opnieuw een keuzemenu. Selecteer hier (bijna altijd) **End Current Session**.

Hoofdstuk 3

Unix Filesysteem

3.1 De structuur

Het Unix **filesysteem** is een (omgekeerde) *boomstructuur*. De top van de boom geven we aan met '/', de *root*-directory. Iedere directory kan vervolgens subdirectories en files bevatten. Een directory wordt ook wel aangegeven met folder of map en in plaats van file wordt ook bestand of document gebruikt. Hieronder zie je een stukje van het filesysteem op een lokaal Unix systeem.
In een shell is er altijd een werkdirectory (huidige directory), als je inlogt is dat altijd je *home directory*, `/home/group/username`. De werkdirectory is van belang omdat alle commando's die je aan de shell geeft betrekking hebben op die directory. Wat op een gegeven moment de werkdirectory in een shell is, losjes gezegd 'waar sta ik in de boom', kun je altijd eenvoudig achterhalen middels het commando `pwd`, **p**rint **w**orking **d**irectory.

```
netelblom> pwd
/home/wiskstud/henk
netelblom>
```

Met `ls` (**l**i**s**t) kun je zien wat er in een directory staat.

```
netelblom> ls
Mail/            vragen10.mws     vragen14.txt     vragen17.mws
P1/              vragen11.mws     vragen15.mws     vragen20.txt
agenda           vragen12.mws     vragen15.txt     windows/
mapleonderwijs@  vragen13.mws     vragen16.mws
p1_opg_3         vragen14.mws     vragen16.txt
netelblom>
```

Met `ls -l` krijg je veel meer informatie, zie §3.3.

Bewegen door het filesysteem doe je met `cd` (**c**hange **d**irectory). Daarbij zijn er twee mogelijkheden, een absoluut pad (begint altijd met '/') of een relatief pad, relatief aan de werkdirectory. Enkele voorbeelden van gebruik (zoals altijd: pas na ⟨()Return) wordt dit uitgevoerd):

- `cd /usr/local/bin`
  verplaatst je naar /usr/local/bin ongeacht waar je stond.

- `cd usr/local/bin`
  zal behalve als je werkdir de rootdir was een foutmelding geven in de trant van: `No such file or directory`

- `cd ..`
  ga één directory omhoog, '..' is het Unix equivalent voor de parent directory (ouder directory). '.' is een afkorting voor de huidige directory.

- `cd ../../wiskstud/peter`
  ga twee dirs omhoog (vanuit je homedir sta je dan in `/home`) en ga vervolgens omlaag in `wiskstud` en vervolgens omlaag naar de homedir van user peter.

- `cd ~peter`
  ga naar de homedir van user `peter`, `~username` is een afkorting voor het absolute pad van de homedir van username. Usernames zijn uniek binnen hetzelfde domein.

- `cd ~/Mail`
  brengt je terug naar je eigen homedir en daarin naar de directory `Mail`. `~` zonder username is een afkorting voor je eigen homedir.

- `cd`
  brengt je terug naar je eigen homedir.

In Unix zijn filenamen willekeurige strings van maximaal 255 tekens, alleen de '/' is uitgesloten, het scheidingsteken voor subdirectories. Desalniettemin is het verstandig geen spaties, `\ * # $ @ ! ~ ‘ ’ "` en haakjes in namen te gebruiken en hoogstens één '.' met daarachter een zinvolle extensie. Het is gebruikelijk om middels de extensie aan te geven om wat voor soort bestand het gaat, bijvoorbeeld `index.html` is een HTML-bestand (zie hfdst. 8).

Merk op dat hoofd- en kleine letters verschillend zijn.

Dan volgen er nu korte beschrijvingen van de belangrijkste Unix commando's om je eigen home directory te beheren.

- `mkdir naam`
  maakt in de werkdir een nieuwe subdirectory `naam`. Zorg voor orde in je homedir door bestanden die bij elkaar horen in een aparte directory te plaatsen.

- `cp file1 file2`
  maakt een kopie van `file1` in de huidige directory onder de naam `file2`. cp komt natuurlijk van **co**p**y**.

- `cp file1 [file2 ... filen] dir`
  maakt kopiën van `file1` (... `filen`) in directory `dir` onder dezelfde naam (namen).

- `mv file1 file2`
  verplaatst (geeft een andere naam) `file1` naar `file2`, beide namen mogen een (deel van een) pad bevatten. (mv is een afkorting van **m**o**v**e)

- `mv dir1 dir2`
  hernoemt `dir1` in `dir2`

- `rm file1 [file2 ...]`
  verwijdert bestand `file1` (`file2` ...). (rm staat voor re**m**ove)

- `rm -r dir`
  verwijdert `dir` en al diens subdirectories en files. Zonder de `-r` optie (recursief) zou dit een foutmelding produceren. Let op met het gebruik van `rm -r`, want als je dit in je homedir intikt, verwijder je je complete bestanden. Het is dus een goed idee vóór elk gebruik van `rm -r` even met `pwd` te checken of je inderdaad in de directory bent waar je denkt te zijn.

- `rmdir dir`
  verwijdert alleen lege directories.

Voor al deze commando's geldt dat ze meerdere mogelijkheden hebben die worden gekozen middels opties, een '-' gevolgd door één of meer letters. Kijk bijvoorbeeld naar de manual pagina van `cp`: `man cp`

`man` is het Unix commando om de manual pagina's te lezen, een zeer uitgebreide bron van informatie. Zie `man man` voor de mogelijkheden. Standaard wordt door `man` het commando `more` gebruikt om de pagina's in beeld te brengen, een bijzonder handige mogelijkheid van `more` is dat je in de pagina's kunt zoeken, dat gaat door een '/' gevolgd door de te zoeken tekst gevolgd door een ⟨`Return`⟩, vervolgens is een 'n' gelijk aan ga naar het volgende voorkomen van de te zoeken tekst. Een vraagteken in `more` levert je een beknopt overzicht van de mogelijkheden.

Meer informatie

Zie ook

`http://curser.science.ru.nl/content-e/pub_NWI/`
`Inleiding_Computergebruik_1180512141473/index.htm`

3.2 Quota: hoeveel diskruimte gebruik ik

Je home directory staat samen met de home directories van een groot aantal andere gebruikers op een grote harde schijf die beheerd wordt door een *fileserver*, een computer die als hoofdtaak heeft het beheren van de bestanden die op de disks staan die aan deze computer gekoppeld zijn. Om te voorkomen dat één gebruiker de totaal beschikbare vrije ruimte op een disk in beslag kan nemen heeft iedere gebruiker een diskquotum, een limiet op de totale diskruimte die iemand mag hebben. Daarbij is er sprake van een zogenaamde zachte limiet en een harde limiet. De zachte limiet mag je tijdelijk overschrijden, om precies te zijn zeven dagen, de harde limiet nooit. Als je met een of andere schrijfactie de harde limiet overschrijdt dan zal die schrijfactie mislukken, dat kan tot het verlies van een bestand leiden! Het systeem zal je bij inloggen waarschuwen als je over de zachte limiet heen bent, het is verstandig daar dan wat aan te doen! Als je de tijdslimiet laat overschrijden dan kun je vanaf dat moment niets meer in je home directory schrijven totdat je weer onder de zachte limiet bent gekomen door bestanden weg te gooien. Het gebeurt nogal eens dat iemand niet meer kan inloggen omdat hij over zijn quota is gegaan, dat komt omdat als je inlogt via een windowing systeem één van de eerste acties bestaat uit het schrijven van een bestand in je homedir, als dat mislukt wordt je direct weer uitgelogd.

Met het `quota` commando kun je zien hoe je ervoor staat

```
netelblom> quota -v
Disk quotas for wisktest (uid 30049):
Filesystem     usage  quota  limit   timeleft  files  quota  limit   timeleft
/home/wiskstud  2703  25000  30000               386   8000  10000
netelblom>
```

Achtereenvolgens wordt het echte gebruik (in kilobytes), de zachte limiet en de harde limiet gegeven eventueel gevolgd door hoeveel tijd je nog hebt voor dat je weer onder de zachte limiet moet zijn gekomen. Daarna volgen nog het totaal aantal files dat je hebt, de zachte limiet en de harde limiet voor het totaal aantal files dat je mag hebben eventueel weer gevolgd door de resterende tijd als je over de zachte limiet heen bent.

Hoe kom je erachter waar al die diskruimte aan op is gegaan? Natuurlijk kun je met `ls` in al je directories gaan kijken maar een bijzonder handig commando in dit verband is `du` (disk usage).

```
du -k dir
```

geeft recursief per subdirectory van `dir` de hoeveelheid diskruimte in kilobytes weer (daar zorgt de `-k` vlag voor). Dus `du -k .` vanuit je homedir geeft een goed idee waar de ruimte gebleven is. Een verdere mogelijkheid is het `find` commando. Met

```
find .  -s +100000c
```

vind je bijvoorbeeld alle bestanden die groter dan 100 kilobytes zijn (het `find` commando is nuttig om zekere bestanden op te sporen, de verschillende opties vind je met `man find`).

Meestal zal het geen probleem zijn een hoop geschikte bestanden en directories te vinden die je gerust weg kan gooien om zo weer ruimte te scheppen. Maar voor bestanden die groot zijn maar die je niet kwijt wilt raken is er nog een andere oplossing. Met `gzip file` kun je een bestand comprimeren, dit verandert de naam van het bestand van *filename* naar *filename.gz*. Om het bestand weer uit te pakken, gebruik je `gunzip file.gz`, daardoor verdwijnt ook de `.gz` suffix. Hoe veel ruimte je door comprimeren wint hangt van de bestanden af, voor tekst bestanden zou dit redelijk veel zijn, voor binaire data (zo als plaatjes) minder.

3.3 Protecties, wie mag wat met mijn files

In een open omgeving als Unix waar op ieder moment veel mensen tegelijkertijd zijn ingelogd en toegang hebben tot hetzelfde filesysteem is het natuurlijk belangrijk om ervoor te zorgen dat bestanden alleen gelezen, veranderd of weggegooid mogen worden door gebruikers die je zelf toestemming daartoe hebt gegeven. In Unix zijn er drie niveaus van permissies, 'user', 'group' en 'others'. De 'user' ben je natuurlijk zelf als we het over je home directory hebben. Iedere user is lid van één of meerdere 'groups'. In eerste instantie zijn studenten alleen lid van de groep `wiskstud`. Met `groups` kun je zien in welke groepen je zit. De 'others' zijn alle gebruikers binnen het `math.ru.nl` domein. Per niveau kun je '**r**ead', '**w**rite' en 'e**x**ecute' permissies geven. Daarbij betekent het 'x'-bit voor een directory dat het toegestaan is om een `cd` naar de directory te doen en voor een bestand dat het uitgevoerd mag worden (heeft dus alleen zin voor programma's). Met `ls -l` krijg je veel meer informatie over de inhoud van een directory.

```
netelblom> ls -l
total 534
drwx------   4 henk     wiskstud       512 Jun 25 08:32 Mail/
drwxr-xr-x   3 henk     wiskstud       512 Dec  8  1998 P1/
-rw-r--r--   1 henk     wiskstud       259 Jun 22 13:18 agenda
lrwxrwxrwx   1 henk     wiskstud        36 Sep  4  1998 mapleonderwijs ->
/vol/wiskalg/mapleonderwijs/wiskunde/
-rw-r--r--   1 henk     wiskstud      1296 Mar  8 16:54 p1_opg_3
-rw-r--r--   1 henk     wiskstud     10327 Mar 15 10:38 vragen10.mws
-rw-r--r--   1 henk     wiskstud     11845 Mar 22 10:30 vragen11.mws
-rw-r--r--   1 henk     wiskstud     26119 Apr  1 10:27 vragen12.mws
-rw-r--r--   1 henk     wiskstud     14837 Apr 14 09:44 vragen13.mws
-rw-r--r--   1 henk     wiskstud     11589 Apr 22 10:27 vragen14.mws
-rw-r--r--   1 henk     wiskstud      9454 Apr 23 09:26 vragen14.txt
-rw-r--r--   1 henk     wiskstud     19720 May 10 10:32 vragen15.mws
-rw-r--r--   1 henk     wiskstud     15180 May 10 10:32 vragen15.txt
-rw-r--r--   1 henk     wiskstud     22196 May 19 12:29 vragen16.mws
-rw-r--r--   1 henk     wiskstud     19880 May 19 12:30 vragen16.txt
-rw-r--r--   1 henk     wiskstud     17170 May 31 10:30 vragen17.mws
-rw-r--r--   1 henk     wiskstud     83857 Oct 16  1998 vragen20.txt
netelblom>
```

We zien achtereenvolgens per regel een blok met de permissies ("`-rw-r--r--`"), het aantal verwijzingen naar dit bestand, de eigenaar, de groep, de afmeting in bytes, de datum van laatste wijziging en tenslotte de naam van het bestand of de directory. Het permissie (of modes) blok bestaat uit 10 entries. De eerste geeft het type aan, 'd' voor directory, 'l' voor een symbolische link (zie 3.4) en '-' voor een normaal bestand. De volgende drie geven de permissies aan voor de 'user' in volgorde rwx, dan drie voor de 'group' en de laatste drie voor 'others'. Daarbij betekent een '-' dat de permissie niet gegeven is. In bovenstaand voorbeeld zien we dat de Mail directory alleen voor gebruiker `henk` zelf op welke manier dan ook gebruikt kan worden. Dit is voor de Mail directory standaard, post is prive. Al zijn andere bestanden zijn leesbaar voor iedereen, en dat is misschien wat overdreven. Het wijzigen van permissies gaat met `chmod` (change modes).

`chmod [ugoa][+-=][rwx] file [more files]`

Uit het eerste blokje kies je voor wie je de permissie wilt veranderen, **u**ser, **g**roup, **o**ther, **a**ll waarbij je er meerdere tegelijk mag gebruiken en waar 'all' natuurlijk een afkorting is voor ugo. Uit het tweede blokje kies je wat je wilt, '+' voor geven van permissie, '–' voor ontnemen en '=' voor gelijk zetten aan. Uit het laatste blokje kies je tenslotte welke permissie(s) je wilt veranderen. Dus met

`chmod go-rx P1`

ontneem je iedereen behalve jezelf read en execute permissie voor de directory `P1`. En met

`chmod o-r v*`

maak je alle bestanden die met een 'v' beginnen onleesbaar voor mensen in een andere groep. Na deze twee operaties zien de permissies er zo uit:

```
netelblom> ls -l
total 534
drwx------   4 henk     wiskstud       512 Jun 25 08:32 Mail/
drwx------   3 henk     wiskstud       512 Dec  8  1998 P1/
-rw-r--r--   1 henk     wiskstud       259 Jun 22 13:18 agenda
```

```
lrwxrwxrwx   1 henk     wiskstud        36 Sep  4  1998 mapleonderwijs ->
/vol/wiskalg/mapleonderwijs/wiskunde/
-rw-r--r--   1 henk     wiskstud      1296 Mar  8 16:54 p1_opg_3
-rw-r-----   1 henk     wiskstud     10327 Mar 15 10:38 vragen10.mws
-rw-r-----   1 henk     wiskstud     11845 Mar 22 10:30 vragen11.mws
...
netelblom>
```

De protectie van een directory is belangrijk omdat als iemand mag schrijven in een directory hij bestanden kan weggooien ook al heeft hij geen schrijfpermissie voor die bestanden. Het verwijderen van een bestand is een directory operatie!

Merk op dat je natuurlijk alleen permissies kunt wijzigen van bestanden waar je de eigenaar van bent.

3.4 Bijzondere files

We zagen hierboven in de directory van `henk` de wat merkwaardig ogende regel met `mapleonderwijs`. Zoals uit de permissies blijkt gaat het hier om een symbolische link, dat is een verwijzing naar een bestand of directory. Alles wat je doet met een symbolische link doe je in werkelijkheid met het bestand of de directory waar de link naar verwijst. In dit geval is het een verwijzing naar een directory, dus als `henk` in zijn home directory `cd mapleonderwijs` intikt dan is dat equivalent met `cd /vol/wiskalg/mapleonderwijs/wiskunde/` maar het eerste is natuurlijk veel makkelijker. Daarmee zie je ook direct het gemak van symbolische links. Het handige van symbolische links is, dat je op meerdere plekken een *virtuele* kopie van een bestand of een directory kunt hebben, zonder meer diskruimte te gebruiken. Je maakt een symbolische link met het commando `ln`

```
 netelblom> ln -s source dest
```

waarbij `source` en `dest` alletwee een bestandsnaam of alletwee een directorynaam mogen zijn. De `mapleonderwijs` entry is gemaakt door

```
netelblom> ln -s /vol/wiskalg/mapleonderwijs/wiskunde/ mapleonderwijs
```

Het verwijderen van een symbolische link gaat gewoon met **rm**.

Hoofdstuk 4

De Shell

4.1 Intro

De `shell` is het interface tussen gebruiker en besturingssysteem. We zullen hier alleen een korte beschrijving geven van de shell `tcsh`. Er zijn er meer, `sh`, `csh`, `ksh`, `bash` om de belangrijkste te noemen, ieder met zijn eigen voor- en nadelen.

Bij het starten van een shell wordt een initialisatiebestand ingelezen, in het geval van `tcsh` is dit `.cshrc` in je homedir. Hiermee kun je het gedrag van de shell aan je eigen behoeften aanpassen. Wees hiermee voorzichtig, als je een fout maakt in dit bestand kan dat tot gevolg hebben dat je niet meer kunt inloggen! De belangrijkste wijzigingen die je zou kunnen willen maken bespreken we iets later. Laat hoe dan ook altijd de regel
`source /system.cshrc`
in dit bestand staan want daardoor worden allerlei standaard instellingen geregeld.

4.2 Commando's en proces controle

Communicatie met het besturingssysteem vindt plaats door het geven van commando's aan de shell. Deze zijn van de vorm
`cmd [opties] [argumenten] [&]`
en worden pas door de shell geïnterpreteerd en uitgevoerd nadat een ⟨`Return`⟩ is gegeven. Zolang nog geen ⟨`Return`⟩ is gegeven kun je in de regel wijzigingen aanbrengen met behulp van de ⟨`Backspace`⟩, ⟨`Delete`⟩ en de pijltjes-toetsen om de cursor binnen de regel te verplaatsen. Opties beginnen meestal met een minteken, zie de *man*-pagina's van het betreffende commando voor een overzicht van de mogelijkheden. Vaak krijg je ook met `cmd -h` een korte overzicht van de opties. Normaal wacht de shell tot het gestarte commando afgelopen is en geeft pas dan een nieuwe prompt te zien waarna je een volgend commando kunt geven. Dat is in het geval van de commando's die we in de eerdere hoofdstukken zijn tegen gekomen ook wat je wilt maar er zijn ook veel commando's die lang lopen en vaak een eigen window openen, in zo'n geval wil je de shell kunnen blijven gebruiken om andere commando's te kunnen geven. Unix is multitasking, dus naast de shell kunnen er nog vele andere *processen* (programma's in Unix worden aangeduid met processen) tegelijkertijd actief zijn (denk bv. ook aan de windowmanager). Om direct controle over de shell terug te krijgen voordat het programma afgelopen is, kun je het in de achtergrond laten lopen. Dat doe je door de commandoregel af te sluiten met een `&`. Mocht je dat vergeten zijn dan kun je het lopende programma onderbreken met

⟨Control⟩-z, het proces wordt dan gestopt maar blijft in het systeem aanwezig. Je kunt het vervolgens met het commando bg (background) in de achtergrond laten doorlopen. Middels fg (foreground) kun je een proces weer naar de voorgrond halen.

Om te zien welke processen actief zijn, zijn er de commando's ps en top. De laatste is erg handig om te zien met welke processen de computer op dat moment bezig is. top sorteert de processen standaard op processorgebruik, maar dat is middels de opties van top aan te passen. Typische uitvoer ziet er zo uit

```
load averages:  1.20,  1.25,  1.26                                      15:34:57
259 processes: 251 sleeping, 5 running, 1 zombie, 1 stopped, 1 on cpu
CPU states:  0.0% idle, 94.5% user,  5.5% kernel,  0.0% iowait,  0.0% swap
Memory: 640M real, 51M free, 463M swap in use, 293M swap free

  PID USERNAME THR PRI NICE  SIZE   RES STATE   TIME    CPU COMMAND
14546 henk      1   0   19   23M   19M run    93.4H 89.79% netscape-3.01
26574 michielb   1  58    0   16M   13M sleep   0:17  3.26% netscape-3.01
26729 polman     1  43    0 1664K 1160K cpu     0:00  1.38% top-sun4u-SunOS
14516 ahml       1  28   10 3312K 2400K run     2:39  0.74% xlock
  190 root       5  58    0 6888K 3928K sleep 359:24  0.19% automountd
 5240 polman     1  58    0 3704K 2584K sleep   0:00  0.15% xterm
20656 axelsson   1  52    0 3432K 1984K sleep   2:21  0.12% xbiff
24956 guta       1  58    0 3472K 2136K sleep   0:00  0.09% pine
 4658 hhendr     1  49    0 3152K 2024K sleep   0:29  0.09% xbiff
26520 michielb   1  58    0 3416K 2096K sleep   0:00  0.09% xbiff
18593 ahml       1  58    0 3416K 1776K sleep  16:22  0.03% xbiff
26283 ebeling    1  48    0 2056K 1360K sleep   0:00  0.03% tcsh
26186 dondergo   1  58    0 3424K 2072K sleep   0:02  0.03% xbiff
24730 reijnier   1  58    0 4352K 3200K sleep   0:00  0.03% xterm
26728 ebeling    1  58    0 1168K  808K sleep   0:00  0.03% jove
```

ps is vooral nuttig in de vorm

```
fanth:~> ps -u henk
   PID TTY      TIME CMD
 26181 ?        0:00 xterm
 26186 ?        0:02 xbiff
 26155 ?        0:01 fvwm
 26191 pts/30   0:01 csh
 26207 ?        0:00 FvwmPage
 26190 ?        0:00 GoodStuf
 26184 ?        0:00 xclock
 26185 ?        0:00 xmh
 26261 pts/30   0:00 ttsessio
 26761 pts/30   0:06 xemacs-2
fanth>
```

Je krijgt dan alle processen van henk te zien. Het wil nog wel eens gebeuren dat een proces zonder dat je dat in de gaten hebt in de achtergrond blijft lopen. Met ps kun je zien of dat het geval is. Vervolgens kun je met het commando kill zo'n proces beëindigen. De syntax is

```
kill [-9] pid
```

waarbij pid (**p**roces **id**entity) het nummer is van het proces dat je wilt afsluiten. Ieder proces op een Unix machine krijgt bij het starten een uniek nummer. Het is verstandig om altijd

eerst `kill pid` te proberen, dit is de nette manier om aan een proces mee te delen dat het moet afsluiten. Als het proces hierop reageert zal het eventuele bestanden die door het proces gebruikt worden netjes afsluiten. Mocht dit echter niet werken dan is `kill -9 pid` een laatste redmiddel. Het ligt voor de hand dat je met `kill` alleen maar je eigen processen mag afsluiten.

4.3 Zoekpad en andere shell variabelen

Als je een commando geeft zal de shell aan het besturingssysteem de opdracht geven om het betreffende programma te starten. Daarbij doet zich een probleem voor, welk programma moet gestart worden? Het filesysteem is vreselijk groot zodat het volledig doorzoeken ervan op zoek naar het gevraagde programma erg duur is. Bovendien wat als het programma op meerdere locaties voorkomt? Om beide problemen te omzeilen hanteert de shell een standaard zoekpad (*path*) voor commando's. Dit is een lijst van directories die als een shell start wordt gebruikt om een lijst van beschikbare programma's op te bouwen waar vervolgens snel in gezocht kan worden. De volgorde waarin de directories gegeven worden in de shellvariabele *path* is bepalend in geval een programma meerdere keren voorkomt. Normaal gesproken zul je hier voorlopig niets aan hoeven te veranderen, alle programma's die je nodig hebt staan in het standaard zoekpad dat gedefinieerd wordt in `/system.cshrc`. Met het commando `set` kun je zien welke shell variabelen gedefinieerd zijn en welke waarde ze hebben (uitvoer hieronder is slechts een selectie)

```
fanth>set
cdpath  (. /home/wiskstud/henk ..)
cwd     /home/wiskstud/henk
dirstack        /home/wiskstud/henk
fignore (.aux .dvi .log .o)
filec
history 50
home    /home/wiskstud/henk
ignoreeof
mail    (10 /var/spool/mail/henk)
noclobber
notify
path    (/home/wiskstud/henk/bin /usr/local/bin /usr/local/X11/bin
/usr/local/X11R5/bin /usr/openwin/bin /usr/dt/bin /usr/local/gnu/bin
/usr/local/mh/bin /usr/local/pbmplus/bin /usr/local/ispell/bin
/opt/SUNWspro/bin /usr/ccs/bin /usr/bin /usr/ucb /vol/texlive/bin/sparc-solaris2.5.1
/usr/games . /opt/SUNWdtpcv/bin /opt/SUNWrtvc/bin)
prompt  fanth>
shell   /bin/tcsh
user    henk
fanth>
```

Naast deze shell variabelen bestaan er omgevingsvariabelen (*environment variables*). Deze worden gewoonlijk alleen met hoofdletters geschreven. Het nut van deze variabelen zit in het feit dat elk programma wat vanuit de shell gestart wordt deze omgevingsvariabelen mee krijgt. Daarmee kun je dus bepaalde standaard instellingen doorgeven aan alle programma's. Met het commando `printenv` (print environment) krijg je te zien welke omgevingsvariabelen gedefinieerd zijn en wat hun waarde is. De belangrijkste voor nu zijn `PRINTER` en `EDITOR`. De eerste wordt door veel programma's gebruikt om te achterhalen welke printer je wilt

gebruiken. Voor studenten staat deze standaard op `lpstud` maar als je standaard op een laserprinter wilt afdrukken dan is het verstandig om in het bestand `~/.cshrc` de regel
`setenv PRINTER lpstud`
te wijzigen in `setenv PRINTER math` (zie ook §6)
Merk op dat veranderingen in `~/.cshrc` alleen effect hebben op shells die na de wijziging gestart worden dus niet op de lopende shell. Om ook in de lopende shell veranderingen in werking te laten treden kun je het commando `source ~/.cshrc` gebruiken.

De waarde van `EDITOR` is van belang omdat een aantal programma's deze variabele gebruiken om te achterhalen welke editor gestart moet worden als er een bestand gewijzigd moet worden.

4.4 Alias gebruik

De shell ondersteund een alias mechanisme, je mag zelf afkortingen bedenken voor veelgebruikte commando's of voor een commando met een aantal opties gezet. Deze definities worden normaalgesproken in het bestand `~/.cshrc` geplaatst. De syntax is eenvoudig
`alias [cmd [command]]`
waarbij alleen `alias` de lijst van gedefinieerde aliassen oplevert, `alias cmd` alleen de definitie voor `cmd` laat zien en tenslotte `alias cmd command` het alias `cmd` definieert als afkorting voor `command`. Als er voor de shell speciale karakters in `command` voorkomen dan dien je die te 'escapen' door ze door een '\' vooraf te laten gaan. Als er spatie(s) in `command` voorkomen dan dien je de hele tekst van `command` te omsluiten met enkele of dubbele aanhalingstekens. De shell is zelf zo 'slim' om loops te voorkomen zodat een alias in de vorm
`alias ls 'ls -F'`
geen problemen geeft.

Pas op dat je bij het kiezen van een naam voor een alias niet per ongeluk een al bestaand Unix commando vervangt. Zoals je ziet als je een keer `alias` intikt zijn er standaard aliassen gedefinieerd voor `rm`, `cp` en `mv`. Wil je een keer de normale definitie van bv. `rm` dan kun je het alias mechanisme uitzetten door het commando vooraf te laten gaan door een '\', dus
`fanth>\rm kan.weg`
zal `kan.weg` opruimen zonder om een bevestiging te vragen.

4.5 History

Het komt nogal eens voor dat je een commando wilt herhalen of met een (kleine) wijziging in de argumenten nog een keer wilt uitvoeren. Daarvoor is in `tcsh` het gebruik van de cursortoetsen ⟨↑⟩ en ⟨↓⟩ bijzonder handig. Met deze toetsen kun je heen en terug door de lijst van commando's die je eerder in de lopende shell hebt gegeven. Met ⟨↑⟩ krijg je telkens de vorige commando regel te zien die je vervolgens middels een ⟨`Return`⟩ opnieuw kunt laten uitvoeren. Maar je mag de regel ook eerst wijzigen alvorens een ⟨`Return`⟩ te geven.

Met het commando `history` krijg je de lijst van gegeven commando's te zien voorzien van een nummer en de tijd van uitvoering. Naast de cursortoetsen om door de *history*-lijst te bladeren zijn er ook nog de volgende mogelijkheden om snel eerder gegeven commando's te herhalen.

`!!`	Het vorige commando.

`!n`	Commando met nummer `n`.
`!-n`	Het `n`-vorige commando.
`!string`	Het meest recente commando dat begon met de letters `string`.
`!?string`	Het meest recente commando dat `string` bevat.

Er zijn nog veel meer mogelijkheden, zie `man tcsh`. Merk op dat door deze speciale betekenis van `!` je iets bijzonders moet doen als je toevallig een uitroepteken als argument wilt geven. De oplossing daarvoor is de speciale betekenis te 'escapen', dat doe je door het karakter met een speciale betekenis vooraf te laten gaan door een '\' (een backslash).

4.6 File completion

Naast de *history* om snel commando's te kunnen geven is er nog een handigheid in de shell ingebouwd, *file completion*. Als je een deel van een bestandsnaam hebt ingetikt (zij het de programmanaam of een bestandsnaam die als argument wordt meegegeven) kun je middels ⟨`Tab`⟩ de shell de naam laten aanvullen. Deze zal voor zover dat uniek kan de naam aanvullen. Daarbij wordt het volgende onderscheid gemaakt: als de shell denkt dat het om een commando naam gaat zal hij dit proberen aan te vullen tot een bestandsnaam die voorkomt in het standaard zoekpad, anders zal hij proberen aan te vullen tot een bestands- of directorynaam in de huidige directory. Als het uniek gelukt is wordt er achter een commando of bestand een spatie gezet en achter een directory een slash ('/'). Als dit niet gebeurt zijn er dus verschillende manieren om aan te vullen, met ⟨`Control`⟩`-d` kun je alle mogelijkheden te zien krijgen.

Ook hier geldt dat er veel meer mogelijkheden zijn, zie *man*-pagina's van `tcsh` onder `Completion and listing`.

4.7 Wildcards

Om tikwerk tot een minimum te beperken ondersteunt de shell een aantal bijzondere afkortingen om filenamen te selecteren. Dit mechanisme wordt *filenaamsubstitutie* of *globbing* genoemd. Als de shellvariabele `noglob` is gezet werkt dit niet (standaard is deze niet gezet en werkt het dus wel). De volgende afkortingen kunnen gebruikt worden:

`*`	Nul of meerdere willekeurige tekens met uitzondering van '/'
`?`	Precies één willekeurig teken, behalve een '/'
`[b-p]`	Precies één teken uit de reeks 'b t/m p'
`[bp]`	Precies één van de tekens 'b' of 'p'

Daarbij dient nog te worden opgemerkt dat `*` en `?` niet matchen met een '.' aan het begin van een filenaam om eventuele problemen met de huidige dir en de parentdir te voorkomen. Als er meerdere bestanden als resultaat van één filenaamsubstitutie optreden dan worden deze door de shell alfabetisch gesorteerd. Een paar voorbeelden van gebruik

- `ls -ld .??*`
  Laat informatie over alle files in de huidige dir zien die met een '.' beginnen maar vermijdt '.' en '..'. In Unix is het gebruikelijk om bestanden die instellingen van programma's vastleggen met een '.' te laten beginnen. De 'd' optie zorgt ervoor dat `ls` info over eventuele directories die met een '.' beginnen laat zien in plaats van de inhoud van zo'n directory.

- `lpr *.txt`
  Print in één printjob alle bestanden met extensie '`txt`' in de huidige dir.

- `rm *.oud`
  gooi alle files met extensie '`.oud`' weg. Pas op dat je geen spatie tussen de '`*`' en de '.' zet want dan worden alle bestanden in de huidige directory weggegooid. Gelukkig is voor nieuwe logins, `rm` beveiligd door middel van een *alias* definitie zodat `rm` vertaald wordt in `rm -i`, d.w.z. per bestand moet je aangeven middels een 'y' of 'n' of je het echt wilt weggooien.

4.8 IO-redirection en *pipes*

In Unix zijn er altijd drie *input/output* kanalen gekoppeld aan een proces. Te weten, *standard input* (`stdin`), *standard output* (`stdout`) en *standard error* (`stderr`). Normaal zijn die gekoppeld aan je terminal, d.w.z. invoer komt van het toetsenbord, normale uitvoer en foutmeldingen gaan naar het window waarin je het commando hebt gegeven. Het is in Unix mogelijk die kanalen te koppelen aan een bestand. Dus bv. in plaats van invoer van het toetsenbord kan de invoer uit een file gehaald worden. De volgende *redirections* zijn mogelijk:

`cmd < file` Het programma `cmd` leest nu uit `file` i.p.v. het toetsenbord.

`cmd > file` Alle normale output van `cmd` komt nu in het bestand `file`, dat nog niet mag bestaan. Anders foutmelding en `cmd` wordt niet uitgevoerd,

`cmd >> file` Als vorige maar nu wordt de uitvoer achteraan het al bestaande bestand `file` geplakt. Het is een fout als `file` nog niet bestaat.

`cmd >& file` Alle normale output en alle foutboodschappen gaan naar `file` en `file` mag nog niet bestaan.

`cmd >>& file` Als vorige maar nu wordt uitvoer achteraan al bestaande `file` geplakt, wederom fout als `file` nog niet bestaat.

Van al deze constructies bestaat ook een versie met een uitroepteken achter de *redirection* met als effect dat het nu geen fout is als `file` al dan niet al wel bestaat. Als voorbeeld
`cmd >\&! file`
Alle normale output en alle foutboodschappen gaan naar `file` en als `file` al bestaat wordt het zondermeer overschreven. Pas dus op met het gebruik van het uitroepteken, je kunt daarmee per ongeluk al bestaande bestanden overschrijven.

Natuurlijk kun je deze redirections ook combineren tot bv. `cmd <infile >outfile &`

Daarnaast bestaat er de mogelijkheid om de uitvoer van één commando direct door te sluizen naar de invoer van een ander, dit heet in Unix een *pipe* aangegeven met een '`|`'. Als

voorbeeld
`man man | lpr -Z2`
dit stuurt de uitvoer van `man` direct naar de standaard printer. Dit is equivalent met

```
fanth> man man >man.out
fanth> lpr -Z2 man.out
fanth> rm man.out
```

maar natuurlijk veel handiger.

Hoofdstuk 5

Editors

Je hebt nu al een hoop informatie gekregen over hoe je je home directory kunt organiseren maar hoe maak je nu zelf nieuwe bestanden en hoe verander je al bestaande. Dat doe je met behulp van een *editor*, daarvan zijn er veel voorhanden die in twee catagoriën uiteen vallen, grafische editors en terminal editors. De eerste werken alleen als je in de omgeving met windows werkt en openen een eigen window en hebben meestal een *menubalk* waarmee je alle belangrijke commando's van de editor kunt kiezen. De tweede werken binnen het shell window waarin je het edit commando geeft. Deze zijn ook te gebruiken op eenvoudige terminals waar geen X-windows op draait en in een terminal window als je middels een modem zou inbellen naar een computer van de faculteit. Het is verstandig om van beide soorten er minstens één te leren kennen. Het is moeilijk een bepaalde editor aan te raden omdat smaken verschillen en elke editor zijn sterke en zwakke punten heeft. Het is daarom verstandig om na verloop van tijd zo nu en dan eens met een andere editor te experimenteren om te zien of die je bevalt.

In de categorie X-editors is er o.a. de keuze uit: `nedit`, `gvim`, `axe`, `emacs` en `xedit` waarbij `nedit` een redelijke keuze is. In `nedit` is een beknopte instructie aanwezig voor alle belangrijke functies, daarnaast is het elementaire gebruik ervan zeer voor de hand liggend. `gvim` is een *gui*-versie van `vim` en beschikt ook over een ingebouwde Help (gui is een afkorting voor *grafical user interface*).

In de categorie terminal-editors is er o.a. de keuze uit `jove`, `vi`, `vim` (een verbeterde versie van `vi`), `pico`, `joe` en `emacs`. Voor een aantal van deze editors zijn mini tutorials beschikbaar om interactief de betreffende editor te leren kennen. Voor `jove` is er het commando `teachjove`, voor `vi(m)` is er het commando `vitutor`, voor `emacs` is er een ingebouwde tutorial die je oproept door in `emacs` achtereenvolgens een ⟨`Control`⟩`-h` gevolgd door een '?', gevolgd door een 't' in te tikken. In alle gevallen geldt dan vervolgens lees de instructies en volg de aanwijzingen op.

Voor alle bovengenoemde editors geldt dat *editorname filename* de betreffende editor start die vervolgens de file met de gegeven naam inleest om te bewerken. Als de file nog niet bestaat zal deze automatisch aangemaakt worden. Merk op dat voor terminal-editors het in de achtergrond starten (door een `&` achter het commando te tikken) weinig zinvol is omdat zo'n editor niet zonder in/output-kanalen kan werken. Het effect zal zijn dat het proces direct gestopt wordt. Met `fg` kun je het vervolgens weer naar de voorgrond halen alsof je de `&` niet gegeven had. Voor X-editors daarentegen is het direct starten in de achtergrond wel zinvol, je krijgt de shell prompt direct terug om andere commando's te kunnen geven.

We herhalen hier nog maar een keer, zorg voor orde in je home directory, maak per

onderwerp subdirectories, geef directories en bestanden zinnige namen en zorg ervoor dat in je home directory niet teveel losse bestanden komen te staan.

Hoofdstuk 6

Printers

Alle printers kun je vanaf een willekeurige werkplek gebruiken. Het print commando is `lpr`
`lpr [-Pprinter] [-m] [-Zopties] file [file2 ...]`

- `-Pprinter`
  uitvoer op de printer met naam printer. Indien je dit weglaat wordt er geprint op jouw standaard printer, d.w.z. de waarde van de environment variabele PRINTER.

- `-m`
  zorgt ervoor dat je email krijgt na afloop van de printopdracht. Daarin melding van de kosten en eventuele foutmeldingen als er iets mis is gegaan met de printopdracht.

- `-Zopties`
  uit de man pagina van lpr

```
Printer and file specific options can be given after the  -Z
option.   These  extra  options must be separated by commas.
If an extra options does not apply to the printer or file,
it is ignored.  Below is a list of some extra options appli-
cable to most PostScript printers.

     Option   Description
     ------   -----------
     m        manual feed
     u        upper tray (obsolited by i= option)
     s        simplex (single sided)
     t        tumble (turn 180 degrees)
     c=num    number of copies
     n        staple the output (only on HP-Mopier)
     i=num    select input tray
     o=num    select output bin (only on HP-Mopier)
     1        1 page per print (only text files; default)
     2        2 pages per print (only text files)
     4        4 pages per print (only text files)
     h        give every page a header (only text files)

Consult the file /vol/src/local/doc/printerh for more infor-
mation.
```

Voor vrijwel alle printers is een printbudget nodig. De meeste printers zijn *postscript* printers. Postscript is een printertaal, het is de standaard printertaal op Unix-systemen. Het print-commando zorgt zelf voor de conversie naar postscript als het om ascii bestanden gaat.

De belangrijkste printers zijn te vinden op

`http://www.cncz.science.ru.nl/ned/hardware/randapparatuur/printers`

Voor de meeste printers geldt dat je een printbudget moet kopen bij de bibliotheek. Alleen voor de `math` printer, de laserprinter van de subfaculteit wiskunde, moet je printbudget kopen bij het secretariaat Wiskunde. Met

```
printbudget printer
```

kun je zien hoe het met je budget gesteld is voor `printer`. Met

```
lpq [-Pprinter] [-l]
```

kun je zien wat de printer aan het doen is, i.h.b. of jouw job al aan de beurt is. De `-l` optie geeft meer uitgebreide informatie.

```
fanth> lpq -Pdante
Final queue dante (wnhome.sci.kun.nl) PS-prt; HP-Mopier HP-5SI-MX (A5016a)
  29 Jun 11:31:31  Initializing printer
  Rank Owner    Pr Job Host         Files           Size     Date
active cjwinkel N  888 ferrari      (stdin)         430762   29 Jun 11:31
```

Hierin zie je ook het Job nummer, dat is belangrijk als je een printjob wilt verwijderen uit de queue of wilt afbreken. Dat gaat met

```
lprm [-Pprinter] jobid
```

Hoofdstuk 7

Email

7.1 Intro

Email (elektronische post) is een belangrijk communicatiemiddel, het is goedkoop, snel, efficiënt en het is makkelijk in gebruik. Mailadressen zijn van de vorm `user@domein`. Het algemene domein voor de β-faculteit is `science.ru.nl`. Voor studenten komt hier nog `student` voor, je hebt dus het mailadres `user@student.science.ru.nl`. Let op dat `user` hierbij niet je UNIX-loginnaam is, maar typisch zoiets als `V.Oorbeeld`, dus voorletter-punt-achternaam. Als je hier heel ongelukkig over bent, kan je bij C&CZ een *alias* voor `user` aanvragen.

Je kunt binnen de Faculteit NWI mailadressen van iedereen behalve studenten achterhalen met het command `telgids naam`.

7.2 Thunderbird

Een goed mail-programma dat wordt aanbevolen is het mail programma `thunderbird` van de browser `firefox`. In hoofdstuk 8 wordt deze browser nader besproken. Als je dit mail programma voor de eerste keer opstart, moet je een aantal dingen instellen.

Kijk voor de laatste versie op
`http://www.cncz.science.ru.nl/ned/faq/email/thunderbird/`

7.3 Webmail

Als je bijvoorbeeld van thuis of onderweg naar je e-mail wilt kijken, bestaat er hiervoor een eigen webpagina van de faculteit NWI. Deze heet `http://webmail.science.ru.nl` en hier log je weer met je gewone UNIX loginnaam en wachtwoord in. Je kunt hiervoor ook gebruik maken van `squirrel`

`https://squirrel.science.ru.nl/src/login.php`
Kijk in ieder geval ook op
`https://www.cncz.science.ru.nl/ned/`

7.4 MIME en attachments

MIME staat voor Multimedia Mail Extensions; lange tijd werd mail alleen gebruikt om teksten te versturen. Langzaam maar zeker ontstond de behoefte om ook plaatjes, geluid en allerlei andere bijzondere bestandsformaten met mail te kunnen versturen. Het antwoord daarop is MIME. Het komt er in het kort op neer dat een willekeurig bestand gecodeerd als 'attachment' (bijvoegsel) aan een mail bericht kan worden toegevoegd en dat het mail programma van de ontvanger dit weer kan uitpakken zodat de ontvanger een exacte kopie krijgt.

De kans is groot dat je zelf al vrij snel zo nu en dan berichten voorzien van attachments zult ontvangen. De meeste mail programma's herkennen zo'n bericht inmiddels. Bij mozilla heb je bijvoorbeeld rechts een raampje met de naam `Attachments`. Hier verschijnen de namen van de verschillende attachments. Door een dubbelklik met de linkermuisknop open je een attachment meteen, maar dit lukt alleen maar als mozilla de type herkent (bijvoorbeeld bij `.pdf` of `.ps` bestanden) en weet met welk programma het attachment bekeken moet worden. Je kunt een attachment ook als gewoon bestand op de harde schijf opslaan (bijvoorbeeld een Word-document dat je met `sdtpcv` of `soffice` wilt bekijken). Klik hiervoor met de rechtermuisknop op de naam, dan verschijnt er een popup menu met mogelijke acties met als belangrijkste `Save As`. Als je deze kiest verschijnt er een fileselectorbox waarmee je de directory en de naam, daar wordt meestal een voorstel voor gedaan, van het te bewaren bestand kunt kiezen. Dit kun je per attachment apart doen.

Je kunt ook zelf een attachment meesturen, nadat je een `Compose` of `Reply` hebt gedaan verschijnt er een window met een `Attach` knop. Als je hier op klikt, kan je met een fileselectorbox het bestand kiezen dat je mee wilt sturen.

Hoofdstuk 8

WWW en browsers

8.1 Het Web

WWW is een afkorting voor *World Wide Web*, een enorme verzameling documenten die naar elkaar verwijzen via zogenaamde *hyperlinks*, enigszins vergelijkbaar met verwijzingen in boeken. Deze documenten bevinden zich op een groot aantal computersystemen verspreid over de hele wereld die met elkaar gekoppeld zijn via het internet, het de wereld omspannende computernetwerk. Alle computers in de faculteit zijn altijd verbonden met internet.

Het grote verschil tussen *hyperlinks* en verwijzingen in boeken is dat bij de laatste je zelf op zoek moet naar een boek waar naar verwezen wordt terwijl bij een *hyperlink* het volstaat om op de *hyperlink* te klikken met de linkermuisknop en het document waar naar verwezen komt automatisch op je scherm, ongeacht waar ter wereld de computer staat die dat document beheert. Om makkelijk toegang te krijgen tot het *Web* heb je een *browser* (bladerprogramma) nodig, de twee bekendste zijn *mozilla-firefox*) en *Internet Explorer* van Microsoft. Als je een terminal zonder X-windows gebruikt of over een langzaam modem met het internet verbonden bent, is soms ook `lynx` een optie, een tekst-gebaseerde browser die de plaatjes over slaat en daarom veel sneller is. Wij zullen *firefox* gebruiken, deze start je d.m.v. het commando `firefox`. De documenten op het web die door de *browser* worden vertoond zijn zogenaamde *hypertext* documenten, teksten voorzien van *hyperlinks*. Daarvoor is een apart bestandsformaat bedacht, *HTML*: HyperText Markup Language, vandaar dat de meeste web documenten file-extensie `.html` of `.htm` hebben. De browser zorgt ervoor dat je slechts op een *hyperlink* hoeft te klikken om de tekst waar naar verwezen wordt op je scherm te krijgen. Meestal gaat dat erg snel, zeker als het om teksten gaat die lokaal (binnen de universiteit) beheerd worden, soms duurt het erg lang. Dat is afhankelijk van de snelheid van de netwerkverbinding tussen de computer waarop de browser loopt en de computer die de gevraagde tekst moet leveren.

Als je firefox start zonder argument dan opent deze met de *home page*, een door de gebruiker bepaald begin document. Dit kan via de `Preferences` in het `Edit` menu veranderd worden. Hoe geef je een document aan of meer algemeen hoe werken de verwijzingen? Je hebt iets nodig om een document te lokaliseren, dat gebeurt met behulp van *URL*'s. *URL* staat voor 'Uniform Resource Locator' (een 'resource' is een hulpmiddel of bron). Een *URL* is van de vorm

```
methode://computernaam/[pad_naar_file]
```

waarbij `methode` meestal `http` (**H**yper**T**ext **T**ransfer **P**rotocol, de standaard methode om

hypertext documenten te benaderen) is en soms `file` (equivalent met `ftp`). Deze tweede methode wordt vooral gebruikt om lokale hypertext bestanden (bv. in je eigen homedir) buiten het web te bekijken.

Wat voorbeelden van *URL*'s:

- `http://www.math.ru.nl`: De home page van de subfaculteit Wiskunde.
- `http://www.math.ru.nl/~bosma/onderwijs/najaar08/wiscom.html`: Informatie over het vak Wiskunde en Computers.
- `http://www.math.ru.nl/desda`: De home page van Desda.
- `http://www.science.ru.nl/cncz`: De home page van C&CZ.
- `http://www.ru.nl/fnwi`: De home page van de β-faculteit.

Er zijn twee manieren om in firefox een *URL* te openen: de eerste is via de menubalk `File --> Open Web Location`, de tweede manier is om de *URL* in het raampje in te tikken en af te sluiten met een ⟨`Return`⟩.

Het web wordt in snel tempo één van de belangrijkste bronnen van informatie. Daarbij is het vooral van belang hoe dingen te vinden in het enorme aanbod. Voor informatie over de studie kijk je natuurlijk in bovenstaande pagina's en volgt de *hyperlinks*. In het algemeen zul je vaak een zoekmachine gebruiken, bijvoorbeeld *Google* op `http://www.google.nl`.

Bij het heen en weer springen over het web zul je regelmatig pagina's tegen komen waarvan je denkt, die zijn interessant daar wil ik nog een keer naar terug. Om niet te hoeven onthouden hoe je daar gekomen bent zijn er *bookmarks*, het equivalent van een boekenlegger. Je klikt met de rechtermuisknop ergens op de achtergrond van de betreffende pagina, zolang je de knop ingedrukt houdt verschijnt er een *popup menu* in beeld met o.a. de optie `Bookmark This Page`. Selecteren doe je door de muisknop los te laten terwijl de cursor op deze regel staat. Je kunt dan voortaan terug keren naar deze pagina door in het Menu *Bookmarks* de entry die correspondeert met deze pagina te selecteren.

8.2 Disk Cache gebruik in firefox

Firefox gebruikt een zogeheten *Disk Cache*, een lokale directory, om pagina's die je recentelijk bezocht hebt op te slaan zodat als je terugkeert op zo'n pagina deze niet opnieuw over het netwerk hoeft te worden opgehaald maar uit de lokale directory gehaald kan worden. Dit is meestal veel sneller. Om problemen met diskquotum te vermijden is het verstandig deze *cache* in zijn grootte te beperken. Kies daartoe in `firefox` uit de menubalk `Edit --> Preferences`. Er verschijnt dan een menu van iconen: kies daaruit `Privacy` en daaronder de laatste optie `Cache`. In de dialoogbox die je nu krijgt kan je de grootte van de cache aangeven, dit moet zeker niet meer dan 5 MB zijn (je hebt waarschijnlijk een totaal diskquotum van 25 MB ter beschikking). Er is ook een knopje `Clear Cache` in dezelfde dialoogbox, hiermee verwijder je alle bestanden uit het Cache directory.