

Wiskunde en Computers: UNIX

Ben Polman, Bernd Souvignier, Wieb Bosma

Najaar 2006

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 wat 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 besturings-systeem 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 MIT¹ 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

¹Massachusetts 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://www.hef.kun.nl/ic-hand/comp.shtml`

Hoofdstuk 2

Inloggen

2.1 PC's

We zullen werken op PC's; meestal log je dan eerst in onder MS-Windows en gebruik je vervolgens XWin32 om daarin een UNIX omgeving te maken; dat doe je door **solost** te selecteren.

2.1.1 Linux PC's

In de wiskunde terminalkamer HG 03.761 staan ook enkele PC's met Fedora-Linux; dan heb je niets meer met MS-Windows te maken. Als je daarop inlogt dan zie je al direct een dialoogbox met **Welcome to computernaam**. De eerste keer dat je inlogt moet je door met de linkermuisknop op *options* te klikken en vervolgens in het menu wat dan verschijnt op *session*, controleren of daar als login sessie *KDE* is gekozen. Indien dat niet het geval is selecteer deze dan met de linkermuisknop. De inlogprocedure is dan verder duidelijk: tik je loginnaam in gevolgd door een **<return>** en je wachtwoord achter **passwd** wederom gevolgd door een **<return>**.

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 de cijfer. Je zult dus bijvoorbeeld vanuit het woord *machtiging* het wachtwoord *m8ig!nG* kiezen.
- Kies een bestaand woord, maar voeg daarin 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 *1z1s?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 met je wachtwoord in 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 soort afdeling in het netwerk van computers, de computers van de subfaculteit wiskunde zoals *fanth* zitten bijvoorbeeld in het *math*-domein, maar je hebt ook toegang tot de andere computers van de faculteit NWI, bijvoorbeeld *solost* en de computers waar je in de algemene terminalkamers op inlogt.)

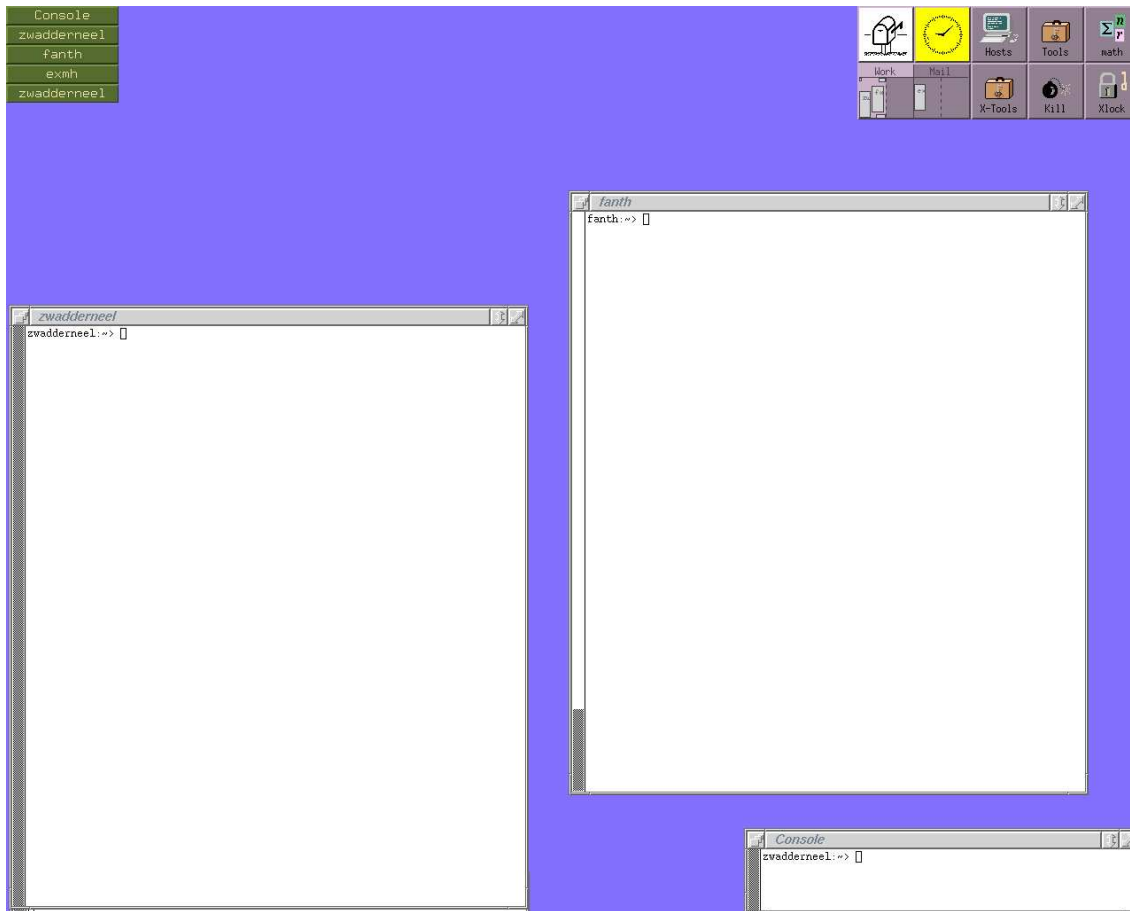
Meer informatie

Zie ook

<http://www.hef.kun.nl/ic-hand/bas.shtml>

2.3 De desktop

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



Figuur 2.1: Standaard situatie na inloggen

- xterm: Een terminal window 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.
- Windowlist: Lijst met windows (=‘programma’s’) die op dat moment door de window-manager beheerd worden.
- Buttonbox: Makkelijke toegang tot veelgebruikte programma's, hierover later meer
- Console: Een Console window, kun je verder het beste negeren. Enige functie is het opvangen van foutboodschappen die anders dwars over je scherm geschreven zouden worden.

Het window systeem waar we mee werken heet *X-windows* vaak ook aangeduid met *X11*. *X11* zorgt ervoor dat meerdere programma's tegelijkertijd gebruik kunnen maken van één beeldscherm. Daarbij kunnen die programma's op verschillende computers draaien.

Het belangrijkste X programma waar we mee te maken hebben is de *window manager*, in ons geval *fvwm2*. Dit programma is verantwoordelijk voor alles wat er op het scherm gebeurt en de interactie tussen gebruiker en programma's. We zullen hier later uitgebreider op ingaan maar op dit moment is alleen van belang te weten dat altijd één window de *keyboardfocus* heeft, dat wil zeggen alles wat je intikt wordt aan het programma doorgegeven waarvan het window de focus heeft. Standaard werkt *fvwm2* in *focus-follows mouse* modus, d.w.z. het window waarin de cursor zich bevindt heeft de focus. Een *xterm* window houdt de focus als de muiscursor uit dat window naar het achtergrondscherf of werkblad (in X is dit ook een window, het *root window*) wordt verplaatst, dat is erg praktisch omdat het achtergrondscherf toch niets met invoer van het toetsenbord kan doen en de muis nog wel eens per ongeluk uit een window wordt geschoven.

Zoals je al wel gezien zult hebben, heeft de muis drie knoppen. Die hebben allemaal verschillende functies 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 *root window*, er verschijnt een *popup menu* met als laatste entry **Exit Fvwm**, selecteer deze regel met de linkermuisknop en je wordt uitgelogd. Mocht je nadat dit menu in beeld is gekomen toch niet willen uitloggen dan kun je het menu weer kwijtraken door een keer met de linkermuisknop ergens in het *root window* te klikken.

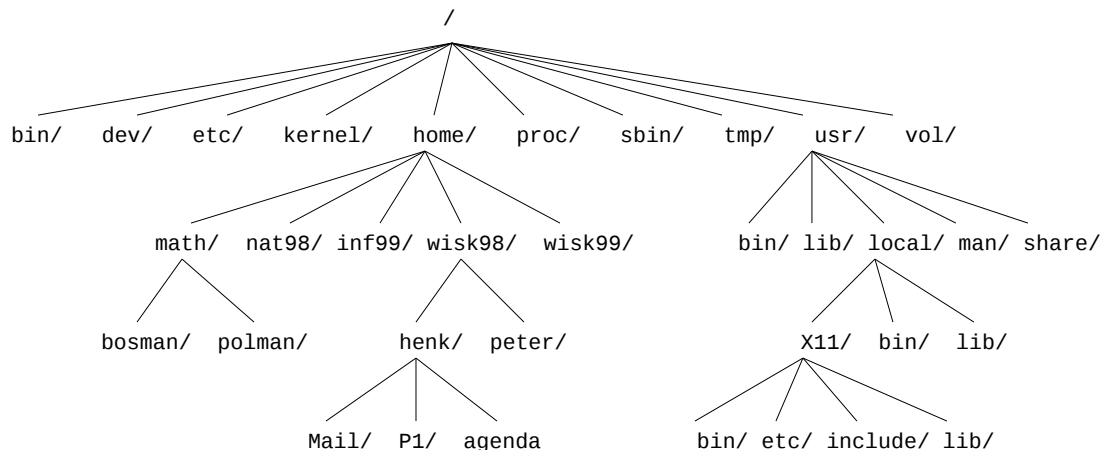
Hoofdstuk 3

Unix Filesysteem

3.1 De structuur

Het Unix filesysteem is een 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*



Figuur 3.1: Een deel van filesysteem op fanth

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**, **print working directory**.

```
fanth> pwd
/home/wisk99/henk
fanth>
```

Met **ls** (**list**) kun je zien wat er in een directory staat.

```

fanth> 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
fanth>

```

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

Bewegen door het filesysteem doe je met `cd` (change directory). Daarbij zijn er twee mogelijkheden, een absoluut pad (begint altijd met '/') of een relatief pad, relatief aan de werkdirectory. Enkele voorbeelden van gebruik

- `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 **copy**.
- `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. (**m**ove)
- `mv dir1 dir2`
hernoemt `dir1` in `dir2`
- `rm file1 [file2 ...]`
verwijdert bestand `file1` (`file2` ...). (**r**emove)
- `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://www.hef.kun.nl/ic-hand/bas-unix-com.shtml>

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 *files*erver, 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 op een systeem waar X-windows draait één van de eerste acties bestaat uit het schrijven van een bestand (`~/.Xauthority`) in je homedir, als dat mislukt wordt je direct weer uitgelogd.

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

```
fanth> 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
fanth>
```

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 ‘read’, ‘write’ en ‘execute’ 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.

```
fanth> 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
fanth>
```

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 `rw`x, 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, **user**, **group**, **other**, **all** 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:

```
fanth> 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
...
fanth>
```

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.

Meer informatie

Zie ook

<http://www.hef.kun.nl/ic-hand/prot.shtml>

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**

```
fanth> ln -s source dest
```

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

```
fanth> 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**, **csch**, **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 window-manager). 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	michieltb	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	michieltb	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` (proces **i**dentify) 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 filesystem 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
ignore (.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/pbplus/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 ondersteunt 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 aliases 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 aliases 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.

<code>!n</code>	Commando met nummer <code>n</code> .
<code>!-n</code>	Het <code>n</code> -vorige commando.
<code>!string</code>	Het meest recente commando dat begon met de letters <code>string</code> .
<code>!?string</code>	Het meest recente commando dat <code>string</code> 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 programmaam 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 *filenaamssubstitutie* 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:

<code>*</code>	Nul of meerdere willekeurige tekens met uitzondering van <code>/</code>
<code>?</code>	Precies één willekeurig teken, behalve een <code>/</code>
<code>[b-p]</code>	Precies één teken uit de reeks ‘b t/m p’
<code>[bp]</code>	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 filenaamssubstitutie 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:

<code>cmd < file</code>	Het programma <code>cmd</code> leest nu uit <code>file</code> i.p.v. het toetsenbord.
<code>cmd > file</code>	Alle normale output van <code>cmd</code> komt nu in het bestand <code>file</code> , dat nog niet mag bestaan. Anders foutmelding en <code>cmd</code> wordt niet uitgevoerd,
<code>cmd >> file</code>	Als vorige maar nu wordt de uitvoer achteraan het al bestaande bestand <code>file</code> geplakt. Het is een fout als <code>file</code> nog niet bestaat.
<code>cmd >& file</code>	Alle normale output en alle foutboodschappen gaan naar <code>file</code> en <code>file</code> mag nog niet bestaan.
<code>cmd >>& file</code>	Als vorige maar nu wordt uitvoer achteraan al bestaande <code>file</code> geplakt, wederom fout als <code>file</code> 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 categorieën uiteen vallen, X-editors en terminal editors. De eerste werken alleen als je in de X-windows omgeving 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** en **gvim** goede keuzes zijn. 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

```
fanth> editor file
```

de betreffende **editor** start die vervolgens **file** inleest om te bewerken. Als **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.

Meer informatie

Zie ook

`http://www.hef.kun.nl/ic-hand/editors.shtml`

Hoofdstuk 6

Printers

Alle printers kun je vanaf een willekeurige werkplek gebruiken. Het print commando is `lpr`

```
lpr [-Pprinter] [-m] [-Zoptions] file [file2 ...]
```

- **-Pprinter**
uitvoer op de printer met naam printer. Indien je dit weglaat wordt er geprint op jou 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.
- **-Zoptions**
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 applicable to most PostScript printers.

Option	Description
-----	-----
m	manual feed
u	upper tray (obsoleted 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 information.

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, zie

<http://www.cncz.science.ru.nl/ned/hardware/randapparatuur/budgetten/>

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 jou 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 van onderweg naar je nieuwe mails 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://squirrel.science.ru.nl/src/login.php>

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/najaar06/compgeb.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.

Meer Informatie

Zie ook

<http://www.hef.kun.nl/ic-hand/web.shtml>

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 dialogbox 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 dialogbox, hiermee verwijder je alle bestanden uit het Cache directory.

Hoofdstuk 9

De X11 omgeving

9.1 Intro

Na verloop van tijd zal waarschijnlijk de behoefte ontstaan om je werkomgeving aan je eigen voorkeuren aan te passen. Je wilt misschien veel gebruikte programma's via een menu of icoon direct kunnen starten, andere afmeting voor de gebruikte fonts, andere kleuren, een aantal programma's automatisch starten als je inlogt, etc. De instelling van je standaardomgeving wordt in hoofdzaak bepaald door een klein aantal bestanden, `.xsession`, `.Xresources`, `.fvwm2rc`, die we apart zullen bespreken.

9.2 `.xsession`

Dit is een voorbeeld van een *shell-script*, een bestand met daarin een verzameling shell commando's die uitgevoerd worden alsof je de commando's hebt ingetikt als je het script als commando geeft. Het `.xsession` bestand wordt automatisch gestart door de X11 omgeving zodra je inlogt. Zodra alle commando's zijn uitgevoerd wordt de X11 omgeving weer verlaten. Hier komen we dadelijk nog op terug.

Het is het van groot belang dat er geen fouten instaan, die leiden ertoe dat de executie van het script wordt afgebroken en je direct weer wordt uitgelogd. Wees dus voorzichtig met het maken van veranderingen!

Het standaard `.xsession` bestand wat je hebt gekregen is rijkelijk voorzien van commentaar en het is dan ook verstandig om middels `more .xsession` het een keer door te lezen. Alle regels die met een `#` beginnen zijn commentaar. Je ziet dat er bijna alleen maar commentaarregels instaan, dat komt omdat het opstarten van x11 programma's tegenwoordig merendeels geregeld wordt door de *windowmanager*. De windowmanager is een programma dat je desktop beheert, je kunt daarmee windows verplaatsen, kleiner/groter maken, tijdelijk sluiten, etc. Zonder windowmanager is dat allemaal niet mogelijk.

De belangrijkste functie van het `.xsession` bestand is dan ook de keuze van windowmanager. Wij gebruiken `fvwm2`, deze is makkelijk te configureren en biedt veel mogelijkheden, zie de volgende paragraaf. Het is gebruikelijk om de windowmanager als laatste te starten in `.xsession` via `exec fvwm2`

De `exec` is een speciale constructie, het zorgt ervoor dat het lopende proces (de shell die `.xsession` uitvoert) wordt vervangen door het proces wat door `exec` gestart wordt, in dit geval `fvwm2`. Dat heeft tot gevolg dat de X11 omgeving stopt zodra de windowmanager wordt

afgesloten, en dat is precies wat je wilt.

Het heeft ook tot gevolg dat de eventuele regels na een `exec` niet meer worden uitgevoerd!

9.3 .fvwm2rc

Een windowmanager beheert je scherm, zorgt voor (ver)plaatsing van windows, de omlijsting van windows, welk window de keyboard input ontvangt, welke kleuren worden gebruikt, etc. Fvwm2 is een zogeheten virtuele windowmanager, d.w.z. de desktop die door `fvwm2` beheerd wordt, kan (veel) groter zijn dan het fysieke scherm. De windowmanager laat een stuk zien van deze grotere desktop, op die manier kun je net doen alsof je op een groter scherm werkt. In `fvwm2` is een desktop altijd een geheel aantal keren het fysieke scherm hoog en breed. Daarnaast kun je meerdere desktops hebben, alsof je meerdere fysieke schermen hebt, `fvwm2` geeft je dan een viewport op één van je desktops.

`Fvwm2` bestaat uit een hoofdprogramma en een aantal modules, hier volgt eerst een korte beschrijving van de belangrijkste modules. Bekijk de afzonderlijke man-pagina's voor meer informatie.

`FvwmPager` toont een miniatuur versie van je desktops. Je kunt de pager gebruiken om je viewport (i.e. dat wat je op je scherm ziet) te verplaatsen tussen de desktops, je viewport te verschuiven binnen een desktop en om windows te verplaatsen. Met de linkermuis klikken in de pager maakt dat stuk van de desktop tot viewport, met de middelste muisknop kun je windows verslepen in de pager, de rechtermuis verandert ook je viewport maar nu kun je vrij verschuiven, niet per se per pagina.

`FvwmWinList` produceert een boxje met knoppen, iedere knop correspondeert met een window dat door `fvwm2` wordt beheerd. Door op een knop te klikken kun je het betreffende window respectievelijk naar voren halen, openen dan wel sluiten, en informatie over het window krijgen via `FvwmIdent`

`FvwmIdent` geeft informatie over het betreffende window

`FvwmBacker` biedt de mogelijkheid om als je van desktop wisselt automatisch een commando te laten uitvoeren, bijvoorbeeld om de achtergrond(kleur) te veranderen.

`FvwmButtons` produceert een window met buttons waar je zelf allerlei commando's aan kunt verbinden

`FvwmAuto` laat een window wat de keyboard input krijgt automatisch naar boven komen.

De configuratie van `fvwm2` en de modules gebeurt door het bestand `.fvwm2rc` in je home-dir. Dit is redelijk van commentaar voorzien, voor alle mogelijkheden en precieze details zie de man-pagina van `fvwm2` en de afzonderlijke modules. Hieronder zullen we kort de belangrijkste elementen van het bestand `.fvwm2rc` bespreken.

9.3.1 Algemene stijlopties

- `DeskTopSize` `2x1`
bepaalt de afmeting van een desktop in *pages*, d.w.z. eenheden van het fysieke scherm (hoogte × breedte)

- **EdgeScroll 100 100**
bepaalt hoeveel procent van een page (scherm) je viewport horizontaal of verticaal verschuift als je met de muis over een rand van het scherm beweegt. In dit geval dus met hele schermen.
- **EdgeResistance msec pixels**
hoeveel miliseconden moet de muis aan de rand blijven staan voordat de viewport verplaatst wordt, hoeveel pixels moet een window over een rand zijn voordat hij echt gedeeltelijk naar een volgend scherm gaat.
- **Style "windowname" Optie(s)**
Hiermee kun je voor alle windows die overeenkomen met windowname een bepaalde optie zetten. Ook hier geldt weer, er zijn veel mogelijkheden en we bespreken er slechts een paar.
- **Style "*" SmartPlacement**
de windowmanager probeert ieder nieuw window ('*' correspondeert met alles) op een lege plaats van het achtergrondwindow te plaatsen.
- **Style "*" RandomPlacement**
als dat niet lukt zet hij het window op een willekeurige plaats neer. Als deze twee *Style* opties ontbreken dan komt een nieuw window aan de muis te hangen en moet je door te klikken bepalen waar het window terecht komt.
- **Style "*" ClickToFocus of**
- **Style "*" MouseFocus**
bepaalt of de keyboard input naar het window gaat waar de muis instaat of naar het window waarin je het laatst geklikt hebt. De standaard instelling is dat de focus met de muis meegaat.
- **Style "*lock" NoTitle,NoHandles,Sticky, WindowListSkip**
zorgt ervoor dat alle windows die eindigen op lock (meestal een of andere clock), geen titelbalk, geen omlijsting, niet opgenomen worden in de WindowList maar wel altijd zichtbaar zijn in de viewport, ze plakken aan het scherm.

9.3.2 Popup Menu's

De popup menu's die in beeld komen als je op het achtergrondwindow klikt of op een van de iconen zijn ook volledig door de gebruiker in te stellen. Dit gebeurt in twee fases, eerst definieer je een menu met behulp van de **AddToMenu** functie in **.fvwm2rc** en vervolgens moet je het laten verschijnen van het menu koppelen aan een of andere actie. De syntax voor het definiëren van een menu is eenvoudig:

```
AddToMenu "Naam" "tekst" Actie
+ "tekst" Actie
+ "tekst" Actie
```

Hierin is **'Naam'** de naam waarmee je aan het popup menu kunt refereren, en elke regel met **'tekst' Actie** definieert één regel in het menu met **'tekst'** als tekst en **Actie** de uit te voeren Actie als deze regel gekozen wordt. Een Actie kan een ingebouwde **fvwm2**

functie zijn, een zelfgedefinieerde functie, een extern programma of een ander popup menu. Bijvoorbeeld het **Utilities**-menu dat verschijnt als je de linkermuisknop indrukt op het achtergrondwindow wordt gedefinieerd door:

```
# Root window left button menu
AddToMenu "Utilities" "Utilities " Title
+           "Refresh All" Refresh
+           "Reread X-resources" Function RereadXres
+           "Print Window" Exec xdpr -device ps
+           "" Nop
+           "Modules" Popup Module-Popup
+           "" Nop
+           "Restart fvwm2 " Restart fvwm2
+           "" Nop
+           "Exit Fvwm " Quit
```

De koppeling met een extern programma kun je dus maken door de actie vooraf te laten gaan door **Exec**, het wordt automatisch in de achtergrond gestart. **Popup** correspondeert natuurlijk met het laten verschijnen van het popup menu met de gegeven naam en **Function** voert een zelf gedefinieerde functie uit, zie hieronder.

9.3.3 Functies

Naast de ingebouwde functies van **fvwm2** die merendeels te maken hebben met het openen, sluiten, naar voren of naar achteren plaatsen van windows kun je ook nieuwe functies definiëren met behulp van

```
AddToFunc "Naam" "context" Actie
+ "context" Actie
+ "context" Actie
```

Hierin staat ‘‘context’’ voor één van

"I" van Immediate, de functie wordt onmiddellijk uitgevoerd zodra hij wordt aangeroepen

"M" van Move, de functie wordt na de aanroep uitgevoerd zodra de muis bewogen wordt

"C" van Click, de functie wordt na de aanroep uitgevoerd zodra de muisknop wordt losgelaten

"D" van DoubleClick, de functie wordt na de aanroep uitgevoerd na een dubbelklik.

Meestal wordt de aanroep veroorzaakt door een muisklik of door een keuze te maken in een popup-menu, maar een functie kan ook gekoppeld zijn aan een toets(encombinatie).

Een bijzondere functie is de **InitFunction**, deze wordt automatisch door **fvwm2** uitgevoerd direct na de start. Dit is dan ook de functie die bepalend is voor welke programma's automatisch gestart moeten worden als je **fvwm2** start. De standaard definitie is

```
AddToFunc "InitFunction" "I" Module FvwmButtons
+           "I" Desk 1
+           "I" Module FvwmWinList
+           "I" Exec exec exmh -geom 550x640+4+170
+           "I" Exec exec xrsh fanth xterm -geometry 80x50-5+200 -name fanth
+           "I" Desk 0
+           "I" Exec exec xterm -geometry 60x5-0-0 -C -name Console -sb
+           "I" Exec exec xrsh -c -geometry 80x50+5-5
```

Hierdoor wordt op Desktop 1 `exmh` en een `xterm` op `fanth` gestart, en op Desktop 0 een `console` om foutmeldingen in op te vangen en een `xterm` op de lokale machine (of als je op een X-terminal werkt, een `xterm` op de X-server). Bovendien worden er twee modules gestart, `FvwmWinList` en `FvwmButtons`. Wil je dus bijvoorbeeld altijd een elektronische agenda starten dan kun je die toevoegen aan de `InitFunction`:

```
+          "I" Exec dtcm
```

9.3.4 FvwmButtons

Dit is een handige module die het mogelijk maakt om een knoppenkast te maken waarmee je veelgebruikte commando's door een klik met de muis kunt starten. In je `.fvwm2rc` vind je de volgende setup van `FvwmButtons`.

```
*FvwmButtonsColumns      5
*fVwmButtonsRows         2
*fVwmButtonsGeometry     -1+0
*fVwmButtons(Swallow XBiff 'Exec xbiff &')
*fVwmButtons(Size 60 60 Swallow XClock 'Exec nice -16 xclock -update 1
-geometry 63x71 &')
*fVwmButtons(Title Hosts, Icon rterm.xpm, Action "PopUp Hosts")
*fVwmButtons(Title Tools, Icon toolbox.xpm, Action "PopUp Tools")
*fVwmButtons(Title math, Icon math4.xpm, Action "PopUp Math")
*fVwmButtons(2x1, Swallow "FvwmPager" 'FvwmPager 0 1')
*fVwmButtons(Title X-Tools, Icon toolbox.xpm, Action "PopUp X-Tools")
*fVwmButtons(Title Kill, Icon rbomb.xpm, Action Destroy)
*fVwmButtons(Title Xlock, Icon xlock.xpm, Action "PopUp Lock")
```

De eerste twee regels bepalen de afmeting van het window, hier dus 2 rijen met elk 5 knoppen (ikonen). De `Geometry` regel bepaalt waar het window terecht komt, offset in x en y richting waarbij de linkerbovenhoek (0,0) is. Een negatieve offset is vanaf respectievelijk rechterkant en onderkant. Hier dus 1 pixel van de rechterkant en bovenaan zodat het window rechtsboven terecht komt.

Vervolgens komen er de regels die bepalen wat er rijsgewijs achtereenvolgens in ieder van de knoppen moet gebeuren. Daarbij is het laatste argument telkens de actie die uitgevoerd moet worden als er op de knop geklikt wordt, daaraan vooraf kan een door komma's gescheiden lijst van opties gegeven worden. Bijzonder is daarbij de `Swallow` waarmee je een X-applicatie in een knopje kunt vangen, hier gebruikt voor een klokje, `xbiff` een *mail notifier*, en voor de *pager*. Daarbij zorgt de `2x1` ervoor dat deze twee naast elkaar gelegen knopjes gebruikt. Hierbij moet je voorzichtig zijn dat dit niet de laatste en de eerste van een rij zouden zijn, dat levert `FvwmButtons` onoverkomelijke problemen op met het plaatsen van de knopjes en het programma houdt er dan mee op.