Introductie Unix

De eerste dag overleven

Raphael 'kena' Poss

2 september 2013

Contents

1 Opdracht

Tijdens deze introductie is van je het volgende verwacht:

1. lees dit hele document door en doe de opdrachten erin. Zorg ervoor dat je alle concepten goed doorgenomen hebt.

2. Volg de volgende tutorials: Tutorial One, Tutorial Two, Tutorial Three, Tutorial Four, Tutorial Five, Tutorial Six.

3. Er is geen #3, je bent dan klaar!

2 Waar ben ik?

Je zit voor je Unix systeem. Je scherm staat aan, toetsenbord reageert goed: je *console* staat klaar voor gebruik.

Iedere Unix gebruiker werkt binnen een *sessie* die hem/haar identificeert. Als je nog geen sessie open hebt, moet je dat nu doen: log in met gebruikersnaam en wachtwoord.

Als je bent ingelogd, zie je een startsscherm. Er zijn veel verschillenden soorten Unixsystemen: sommige presenteren alleen een terminal, met puur tekst als in- en uitvoer. Anderen presenteren een grafische interface. De meest gebruikelijke grafische interfaces zijn:

- de zogenaamde X11 interface, te vinden op meeste werkcomputers zoals de computers bij de UvA;
- de Android interface, te vinden op alle Android toestellen (mobieltjes / tablets);
- de Quartz interface, te vinden op MacOS X.

Voor deze introductie gaan we beginnen met X11. Zoals je zal ontdekken is de grafische interface alleen een dunne laag tussen de gebruiker en het systeem; alle Unix systemen werken min of meer met dezelfde concepten eronder. De kennis die je met deze introductie zal verzamelen is dus grotendeels toepasselijk in andere Unix omgevingen, ongeacht wat je op het scherm kunt zien.

Om die onderliggende, gezamenlijke concepten te leren, gaan we dus de grafische interface grotendeels negeren voor deze introductie. Na de introductie, zal je zelf begrijpen hoe grafische tools zich koppelen aan de rest van het systeem *in het algemeen*. Daarvandaan ben je dan in staat om iedere soort Unix systeem te gebruiken, zelfs als de grafische interface afwijkt van wat je vandaag bij de UvA ziet.

2.1 Een terminal vinden

Je ziet dus een scherm met visuele elementen. Als je al een terminal voor je neus hebt, kun je deze paragraaf overslaan. Anders moet je op zoek naar een terminal. Er is vast ergens op het scherm een menu die je toegang geeft tot alle beschikbare applicaties. Een terminal is gebruikelijk te vinden bij een menu "Hulpprogramma's", "Utilities", "Systeem" of "Development". Al je Unixsysteem een visuele zoekfunctie heeft, kun je deze ook gebruiken: zoek naar "Terminal" of "xterm". Als je een veld kunt vinden om een commando handmatig in te voeren, tik dan "`xterm`" in. Op Linux (maar niet andere Unixsystemen) kun je ook toegang krijgen tot een volscherm terminal door Ctrl+Alt+F1 tegelijk te drukken op het toetsenbord; daar moet je opnieuw een sessie openen door je gebruikersnaam en wachtwoord in te tikken. Je kunt terug naar de grafische omgeving door Alt+F7.

Een terminal ziet er ongeveer zo uit:

Het kleurenschema (zwart op wit, wit of zwart, wit op groen, etc) maakt natuurlijk niet uit. Zorg er alleen voor dat de tekst groot genoeg is om je ogen niet in te hoeven spannen.

Wat je ziet is een plek waar je commando's in kan tikken op het toetsenbord. Dit heet de "command line", of *commandolijn*.

2.2 Shell en programma's

Meerdere programma's delen de terminal om met jou te communiceren. Het "hoofdprogramma" dat je als eerste ziet heet de *shell*: het is het programma die je commando's zal herkennen en uitvoeren.

Aan het begin heb je een soort mini-teksteditor voor maar 1 regel tekst, voor je commando. Hier kan je je eerste commando editen: de lettertoetsen om tekst toe te voegen, pijltjes te navigeren, maar ook:

- Ctrl+A om te navigeren naar het begin van de lijn,
- Ctrl+E naar het eind,
- Alt+F (of [Escape], dan 'F') naar het volgende woord,
- Alt+B (of [Escape], dan 'B') naar het vorige woord.
- Ctrl+H als je "backspace" toets het niet goed doet (gebeurt helaas vaak)

Zodra je een commando valideert met de [Enter] toets, zal de shell zoeken naar de betekenis van je commando, en als het een programma is zal de shell een nieuwe process aanmaken om je programma te draaien. Zodra dit gebeurt, geeft de shell tijdelijk controle van de terminal aan het nieuwe programma, totdat het programma klaar is.

Tik maar bijvoorbeeld in:

```
echo hello; sleep 5
```

(dan [Enter])

Met deze regel vraag je aan je shell om eerst het commando `echo` te draaien om tekst te printen op de terminal, dan het commando `sleep` om 5 seconden te wachten. Zolang de commando's draaien, hebben ze controle over je terminal en zal de shell wachten. Je kunt pas een commando opnieuw intikken als het vorige commando klaar is.

Een terminal en een shell zijn de eenvoudigste interface naar een Unix systeem. Ze vormen een soort sequentiële interactieplan, waar de shell en commando's om en om de terminal met jou delen.

2.3 In... maar ook uit!

Veel introductiecursussen en documentaties laten zien hoe je programma's kunt opstarten, en dingen ermee doen. Wel leuk, maar onze ervaring als studenten is dat het hoofdprobleem is altijd: "hoe kom ik terug naar een bekende situatie?". Stel je voor: je hebt een programma geopend, en je weet niet hoe je hem kunt afsluiten. Wat dan?

Vaak werkt de "Escape" toets *niet*. De toetscombinatie Alt+F4 ook niet. Dan op een terminal is er natuurlijk geen kleine kruis die je kunt klikken met je muis, of vinger op een touchscherm.

Op Unix zijn er meerdere manieren een programma af te sluiten:

1. als je programma een overzichtelijk interface heeft met duidelijk aanwijzing van hoe je hem kunt afsluiten, gebruik dan dat.

2. als je programma een soort commandolijn heeft waar je commando's blijkt in te kunnen tikken, probeer dan de woorden "`quit`" of "`exit`" in te tikken. Soms is de letter `q` alleen al genoeg. Als het niet werkt, probeer dan "`help`". Tik bijvoorbeeld de volgende commando's in je terminal, en test jezelf dat je ze kunt afsluiten met een commandowoord:

   ```
   gdb
   bc
   expect
   telnet
   csh
   gnuplot
   ```

3. zo niet, als je programma meestal gebaseerd is op invoer vanuit een terminal, gebruik dan de toetscombinatie Ctrl+D. Deze combinatie wordt ook beschreven als "`^D`" in documentatieteksten; het betekent "de huidige invoer beëindigen". Tik de volgende commando's in je terminal, en test jezelf dat je ze kunt afsluiten met `^D`:

   ```
   cat
   sort
   uniq
   ```

```
sed
perl

python
```

NB: De meeste commando's die "`exit`" of "`quit`" ondersteunen kunnen ook worden afgesloten door `^D`.

4. als `^D` het niet doet, kun je een "geweldadigere" manier gebruiken: de toetscombinatie Ctrl+C, ook geschreven als "`^C`". Het betekent "een signaal sturen aan het proces om zich af te sluiten", zonder te veel vragen te stellen. Het werkt met bijna alle programma's, tenminste die die de toetscombinatie niet specifiek hebben uitgeschakeld. Tik de volgende commando's in, en zie voor jezelf hoe je ze kunt afsluiten met `^C`:

   ```
   top
   yes
   sleep 10

   hexdump /dev/urandom
   ```

   NB: `^C` werkt ook vaak als een programma lang niet reageert, of als je niet wil wachten.

5. als het je niet lukt met de stappen #1-#4 het programma af te sluiten, heb je misschien te maken met een van "de bijzondere 4":

   - het programma `vi` or `vim` (teksteditor): die wordt afgesloten door [Escape], ':' (dubbele punt), 'q', '!', [Enter] in te tikken.
   - het programma `emacs` (een andere teksteditor): die wordt afgesloten door Ctrl+X, dan Ctrl+C in te tikken.
   - het programma `ssh` (om een andere computer te benaderen): die wordt afgesloten door uit te loggen binnen de ssh sessie. Mocht dit niet lukken, bijvoorbeeld door een netwerkprobleem, kan `ssh` worden afgesloten door [Enter], '~', '.' in te tikken.
   - het programma `telnet`: tik dan Ctrl+] in, dan `exit`.

6. als alles tot nu toe heeft gefaald, kan je de "final solution" gebruiken. Dit werkt op alle processen, op alle commando's, en je moet dus voorzichtig mee omgaan: het commando `kill`.

   Om `kill` te kunnen gebruiken, moet je toegang hebben tot een commandolijn. Natuurlijk, als je `kill` nodig hebt, is de kans groot dat je huidige terminal bezet is door het commando dat je wil beëindigen. Wat dan? Twee oplossingen.

   Eerst kan je kijken of `^Z` het doet. "`^Z`" betekent "het huidige programma pauzeren, sturen naar de achtergrond, en de shell met commandolijn die erachter aan het wachten was terug naar de voorgrond plaatsen". Sommige programma's die niet op `^C` reageren kunnen wel worden geplaatst naar de achtergrond met `^Z`.

   Als `^Z` het niet doet, kun je ook een andere terminal openen op hetzelfde systeem. Zelfs als het dan opnieuw moet inloggen, heb je toegang tot alle processen die je al draaiend hebt in je eerdere sessie.

   Zodra je wel een commandolijn beschikt naast het programma die je wil beëindigen, moet je zijn *proces identificatinummer* weten, ook "PID" genoemd. Dit haal je door het

commando "`ps x`", dat een lijst aangeeft van alle processen die nu draaien van je op het systeem. Het identificatienummer is dan te vinden in de eerste column.

Met het PID kun je dan het commando `kill` intikken, gevolgd door een spatie, dan het PID, dan [Enter]. Dit stuurt hetzelfde signaal als `^C` maar dan ook zelfs als het proces de toetscombinatie heeft uitgeschakeld.

(NB: het was trouwens niet de hele waarheid: alleen het commando `kill` met een PID is soms niet genoeg. Het commando `kill` heeft wel een "ultieme" vorm, maar deze zal je pas later ontdekken.)

2.4 Samenvatting toetscombinaties

Tot nu toe heb je de volgende al geleerd:

Toetscombinatie	**Waar?**	**Beschrijving**	**Wordt ook beschreven als**
Ctrl-A	Shell, Emacs	Begin van de lijn	`^A`, C-a
Ctrl-E	Shell, Emacs	Eind van de lijn	`^E`, C-e
Alt-F / Esc,F	Shell, Emacs	Volgende woord	`ESC F`, M-f
Alt-B / Esc,B	Shell, Emacs	Vorige woord	`ESC B`, M-b
Ctrl+H	Shell, Emacs	Eerste karakter links verwijderen.	`^H`, C-h
Ctrl-D	Tekstinvoer	Tekstinvoer beëindigen	`^D`
Ctrl-C	Shell	Huidige regel negeren, opnieuw beginnen	`^C`
Ctrl-C	algemeen	Huidige programma beëindigen	`^C`
Ctrl-Z	algemeen	Huidige programma pauzeren, achtergrond	`^Z`
Esc, :, q, !	vi, vim	vi/vim beëindigen	
Ctrl-X, Ctrl-C	Emacs	Emacs beëindigen	
Enter, ~, .	ssh	ssh beëeindigen	
Ctrl+], exit	telnet	telnet beëindigen	

Zoals je ziet kan dezelfde toetscombinatie meerdere functies hebben, afhankelijk van welk programma het ontvangt. Daarom is het belangrijk te begrijpen dat de programma's samen de terminal delen, maar dat meestal maar één programma actief is aan het "hoofd" van de terminal.

3 Documentatie zoeken

Het is natuurlijk altijd mogelijk je browser te openen en naar hulp gaan vragen op een forum of zo. Maar het wordt ook verwacht dat je weet dat veel informatie al beschikbaar is binnen je Unix systeem!

3.1 Informatie over commando's

- om te weten wat een bepaald commando doet: gebruik dan "`man`". Dit is een interface naar de zogenaamde "online Unix manual": een collectie tekstpagina's en artikelen. De meeste

commando's hebben allemaal een eigen pagina in de manual.

Tik dus in: `man man` [Enter]

en lees de pagina voor het commando `man` zelf om het leren gebruiken.

Als je `man` draait en als de tekst niet volledig binnen de terminal past, krijg je een soort navigatieinterface. Die bestuur je door je toetsenbord:

- [Spatie] om naar de volgende pagina te navigeren,
- "b" naar de vorige pagina,
- de pijltjes "omhoog" en "omlaag",
- "q" om de navigatie te verlaten,
- "/" en dan een zoektekst om iets te zoeken.

Zoals je snel zal ontdekken, bevatten manual pages vaak meer informatie dan wat je nodig hebt. Gebruik dan "/EXAMPLES" [Enter] om snel de voorbeelden te vinden. Veel pagina's hebben voorbeelden.

Voor meer informatie: https://en.wikipedia.org/wiki/Man_page

- om te herinneren welk commando iets doet, als je de naam van het commando bent vergeten: "`apropos`".

  Stel je voor bijvoorbeeld dat je niet meer weet welk commando je moet gebruiken om een proces te beëindigen. Je denkt aan het woord "terminate" maar dit is geen commando. Dan tik je `apropos terminate` in: `apropos` zoekt in alle beschikbare manual pagina's naar het woord "terminate" en laat je zien de naam van de pagina's die het woord bevatten. Hiermee zie je waarschijnlijk snel weer een referentie naar het programma `kill`.

- om snel te weten welke opties een programma heeft. Je kunt natuurlijk het commando `man` gebruiken, maar de meeste programma's reageren ook op de optie `--help` of `-h`. Probeer bijvoorbeeld:

  ```
  ls --help
  man --help
  grep --help
  ```

  Misschien past de tekst van het programma niet volledig binnen je terminal. Als je terminal wordt weergegeven in een grafische omgeving, beschik je misschien een scrollbar. (Dit is een luxe.) Zo niet, kan je het woord "`|more`" toevoegen aan het eind van je commandolijn, bijvoorbeeld:

  ```
  ls --help | more
  ```

  Dit werkt op alle Unix systemen, ongeacht de omgeving waar je een terminal hebt.

Naast die opties bestaat er nog een bijzondere hulp-systeem, die heet `info`. Weinig programma's maken gebruik van `info`, maar voor de enkele programma's die er wel in staan, is de documentatie daar zeer uitgebreid en vaak van zeer hoge kwaliteit. Je hoeft het nu niet te leren gebruiken, maar onthoud hem voor later, voornamelijk voor je vakken over compiler/vertalerbouw.

3.2 Naar bestanden zoeken

- je bent de naam van een bestand kwijt: gebruik dan `find`. Gebruik `man find` en zoek naar voorbeelden.

  `find` is altijd beschikbaar. Naast `find` is soms het commando `locate` ook beschikbaar. `locate` is minder flexibel dan `find` maar kan sneller zijn.

- je zoekt een bestand die een bepaalde tekst heeft als inhoud: gebruik dan `grep`. Gebruik `man grep` en zoek naar voorbeelden.

3.3 In je web browser

Je wordt aangeraden om de volgende bookmarks bij te houden in je webbrowser.

Introductie Unix/Linux:

- http://www.oliverelliott.org/article/computing/tut_unix/
- http://www.ee.surrey.ac.uk/Teaching/Unix/
- http://tldp.org/LDP/intro-linux/html/index.html
- http://www.science.uva.nl/~arnoud/education/workstations/linux_introduction

Enkele nuttige commando's:

- http://www.science.uva.nl/~arnoud/education/workstations/linux_introduction/deel1/lijst.html

Algemene vragen:

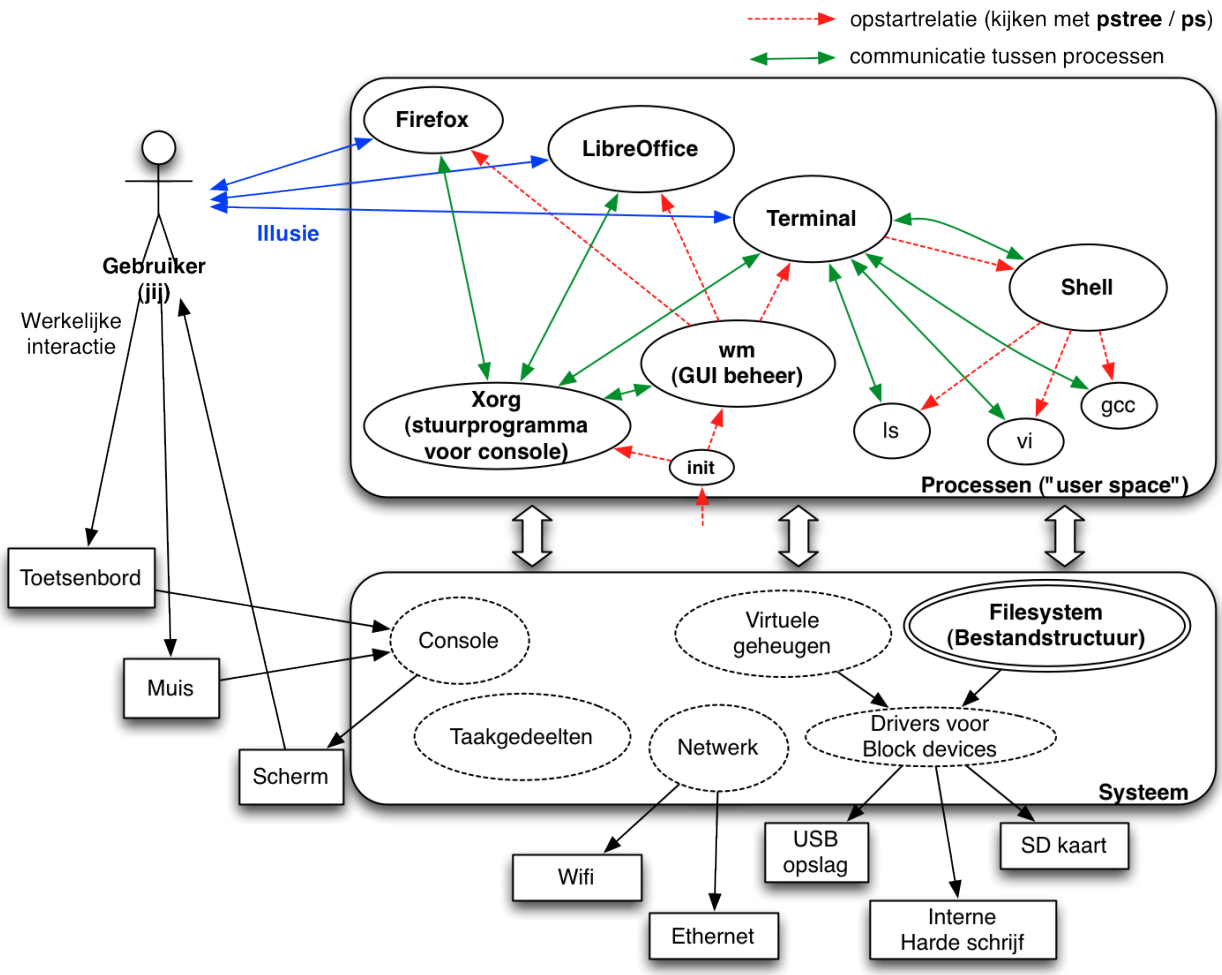
- How to ask a smart question: http://faculty.gvc.edu/ssnyder/121/Goodquestions.html
- How to ask questions the smart way: http://catb.org/~esr/faqs/smart-questions.html
- Stack Exchange, Unix: http://unix.stackexchange.com/
- Stack Exchange, Super User: http://superuser.com/
- Op IRC: https://www.freenode.net/

4 Draaiende programma's

Eerder heb je al geleerd hoe je `^Z` kan gebruiken om een programma tijdelijk te stoppen en sturen naar de achtergrond; heb je ook geleerd hoe je de commando's `ps` en `kill` kunt gebruiken om een programma af te sluiten.

Door `ps` heb je waarschijnlijk veel andere programma's zien draaien die je misschien had verwacht. Een Unix systeem heeft vaak tientallen *processen* draaien in de achtergrond.

Het volgende plaatje laat zien hoe je Unix systeem eruit ziet "van binnen":



4.1 Opstarten, communiceren en eindigen / afsluiten

Ieder proces wordt *opgestart* (rode pijlen) door een ander proces, behalve `init`. Alle processen zijn dus gerelateerd door een ouder-kind relatie in een boomstructuur. Je kunt deze structuur bekijken door het programma `pstree` te draaien.

Hiernaast *communiceren* processen met elkaar (groene pijlen in het plaatje). Bijvoorbeeld, communiceert de shell (voor de commandolijn) met de terminal. Alle commando's opgestart door de shell communiceren ook met de terminal.

Hier moet je letten op het volgende:

> *Als een programma wordt afgesloten, worden zijn kind-programma's (en kleinkinderen) niet automatisch afgesloten. Echter, worden alle programma's die met hem communiceren geïnformeerd dat de communicatiekanaal afgesloten wordt.*

Voor de meeste programma's, als hun communicatiekanaal wordt afgesloten eindigen ze zelf. Dit is de reden waarom alle programma's binnen je sessie automatisch eindigen als je je terminal afsluit.

Maar als een programma niet communiceert met de terminal, is het mogelijk dat het niet beseft dat de terminal wordt afgesloten. Dan blijft het programma draaien in de achtergrond.

4.2 Acties op processen

Als je een commando intikt in de shell, wordt een proces aangemaakt in het systeem om het programma te draaien.

Je hebt eerder gezien hoe je met `kill` een proces kunt beëindigen.

Tot nu toe ziet er alsof je de volgende kunt doen met een proces:

- het opstarten vanuit een shell;
- zijn communicatiekanalen afsluiten:
  - door Ctrl+D op de terminal, of
  - door de processen aan de andere kant van de communicatiekanalen af te sluiten;
- het beëindigen door Ctrl+C;
- wachten totdat het zelf klaar is.

Technisch gezien zijn alle interacties met processen gebaseerd op de volgende principes:

- een proces "van binnen" dupliceren ("fork" in zijn eigen code);
- een proces "van binnen" vervangen ("exec" in zijn eigen code);
- een proces "van binnen" beëindigen ("exit" in zijn eigen code);
- met andere processen communiceren "van binnen" ("read"/"write" in zijn eigen code);
- een *signaal* sturen van een proces naar zichzelf of naar een ander process.

Bijvoorbeeld, als je een commando intikt in een shell, gaat de shell zichzelf dupliceren met "fork", dan in de nieuwe kopie van zichzelf, zichzelf vervangen met "exec" door het programma wiens naam je hebt ingetikt.

Als je Ctrl+D gebruikt, of als je een terminal afsluit terwijl commando's "erin" draaien, dan gebeurt er het volgende: zodra het programma weer probeert te lezen op zijn invoerkanaal, of schrijven op zijn uitvoerkanaal, gaat hij automatisch van het systeem een signaal ontvangen (`SIGHUP`) die zegt "oeps, je communicatiekanaal is weg". Voor de meeste programma's betekent het gewoon "afsluiten".

De toetscombinatie Ctrl+Z stuurt alleen het signaal "`SIGTSTP`" aan het proces op het hoofd van je terminal. De consequentie ervan, een pauze en dan de shell zien terugkomen op de voorgrond, gebeurt door het afhandelen van dit signaal door het proces zelf en de shell erachter.

De commando "`kill`", ondanks zijn naam, is alleen een programma om signalen te sturen aan andere processen. Als je geen signaalnaam aangeeft, stuurt hij automatisch `SIGTERM`.

Je kunt "`man signal`" en "`man kill`" gebruiken om erover meer te leren.

Opdracht: op basis van het plaatje hierboven, en je begrip van signalen, probeer te gokken wat gebeurt als je het signaal `SIGTERM` stuurt aan het proces Xorg (of misschien heet het enkel "`X`" op je systeem).

4.3 Jobbeheer

Je kent nu al `ps`, `kill`, en het concept van signalen.

Een applicatie hiervan is het zogenoemde "job control" protocol. Dit is een feature van de shell en is beschikbaar met bijna alle shells op alle Unix systemen.

Probeer het volgende:

1. Draai het commando: `sleep 300`.
2. Toets `^Z`. Je shell zegt dan "[1]+ Stopped" of iets dergelijks.

   Op dit moment is je proces gepauzeerd in de achtergrond.
3. Draai het commando: `jobs`

   `jobs` laat zien welke processen zijn nu opgestart door je shell.
4. Draai het commando: `fg %1`

   `fg` brengt een proces die op de achtergrond zit terug aan het hoofd van de terminal, en activeert hem weer als het gepauzeerd was. Hier komt dus `sleep` terug.
5. Toets `^Z` weer.
6. Draai het commando: `bg %1`

   `bg` laat een proces in de achtergrond, maar activeert hem weer. Hierdoor blijft het proces draaien, alleen zit hij niet meer aan het hoofd van de terminal. Je kunt dan tegelijk iets anders gaan doen, bijvoorbeeld andere commando's intikken in je shell, of een andere commando terugbrengen aan het hoofd van je terminal.
7. Draai het commando: `vim`
8. Toets `^Z` weer.
9. Draai het commando: `jobs`

   Hier zie je twee processen: `sleep` van eerder, die nog draait; en `vim` die gestopt is.
10. Draai het commando: `kill %1`.

    Dit stuurt `SIGTERM` aan de eerste job om het af te sluiten.
11. Draai het commando: `fg`

    Dit brengt de andere job aan de voorgrond en activeert hem weer.
12. Sluit die laatste job af zelf.

Hiermee heb je geleerd:

Toets/Commando	**Beschrijving**
`^Z`	Proces pauzeren (`SIGTSTP`), sturen naar achtergrond
`jobs`	Lijst van jobs aangeven
`fg %N`	Breng een job terug aan het hoofd van de terminal, en activeer hem
`bg %N`	Activeer een job weer maar laat hem in de achtergrond
`kill %N`	Stuur een signaal aan een job.

Om er meer over te leren: https://en.wikipedia.org/wiki/Job_control_%28Unix%29

5 Je teksteditor kiezen

Als je het nog niet weet: er is een soort "religieuze oorlog" tussen de gebruikers van het programma Vi en de gebruikers van Emacs. Het is al bijna 30 jaar zo, en het blijft waarschijnlijk nog 30 jaar zo.

- https://en.wikipedia.org/wiki/Editor_war
- http://www.c2.com/cgi/wiki?EmacsVsVi

En je wordt ook waarschijnlijk verzocht om een kant te kiezen. :)
Als je helemaal niet tussen Emacs and Vi wil kiezen: `joe`, `nano` en `pico` zijn andere kleine, eenvoudige interactieve teksteditors die bijna overal beschikbaar zijn.
Anders:

- Vi (`vi` of zijn modernere versie `vim`): werkt op basis van twee modi voor interactie: een "controle" modus en een "invoer" modus. Je wordt er een beetje schizofreen van. In de "controlemodus" van `vi` tik je "commando's" om operaties uit te voeren op de tekst: tekst zoeken, verplaatsen, vervangen, etc. In tegenstelling met `emacs` zijn de commando's van `vi`/`vim` redelijk logisch te onthouden. Ook heeft `vi` meestal geen toets *combinatie* nodig, waar je met meerdere vingers tegelijk moet duwen op je toetsenbord.
- Emacs (`emacs`): werkt op basis van zijn eigen besturingssysteem, geprogrammeerd in het eerbiedwaardige programmeertaal LISP. Emacs is dus heel krachtig met heel veel functionaliteit, maar dus ook wat ingewikkelder goed te leren gebruiken. Emacs maakt uigebreid gebruik van toetscombinaties.

Om meer erover te weten:

- VIM:
  - Overzicht: https://en.wikipedia.org/wiki/VIM
  - Leuke tutorial: http://vim-adventures.com/
- Emacs:
  - Overzicht: https://en.wikipedia.org/wiki/Emacs
  - Leuke tutorial: http://david.rothlis.net/emacs/howtolearn.html

Kortom, in de praktijk:

Actie	**In vi/vim**	**In Emacs**
Editor afsluiten	:q / :q!	C-x C-c
Bestand openen	:e <file>	C-x C-f
Bestand opslaan	:w	C-x C-s
Bestand sluiten	:close	C-x k

... continued on next page

Actie	**In vi/vim**	**In Emacs**
Navigeren	Pijlen	Pijlen
Zoeken	/	C-s
Navigeren	h/j/k/l	C-p / C-n / C-b / C-f
Tekst verwijderen	x	C-h
Regel verwijderen	dd	C-k
Annuleren	u	C-_
Tekst toevoegen	i	Gewoon invoeren
Naar controlemodus	Escape	M-x

6 Bestandsstructuur

Bestanden worden georganiseerd in mappen, met mappen binnen mappen. Dit principe ken je al. De bijzonderheden:

- ieder proces heeft een notie van "huidige map", of *working directory*. Als je een bestand noemt zonder te zeggen waar het zich vindt, gaat het proces dat bestand proberen te zoeken in de huidige map.
- het commando om je "huidige map" te controleren: `cd`

  (heet ook "change directory")

  en om je huidige map te identificeren: `pwd`

  ("print working directory")
- Het commando `cd` zonder argumenten brengt je altijd naar je "thuismap" of *home directory*.
- Nieuwe processen hebben dezelfde working directory als hun ouder, maar kunnen het verder veranderen zonder die van de ouder te veranderen.
- een *pad* is een naam naar een map of bestand *vanaf* de huidige map. Bijvoorbeeld "`Documents/hello.txt`" refereert naar het bestand/map `hello.txt` in de map `Documents` vanuit de huidige map. Je kunt een *absolute* map noemen door met een "/" te beginnen. Bijvoorbeeld op de UvA machines refereert altijd "`/scratch`" naar een map met veel ruimte, bijvoorbeeld voor experimenten.

  Opdracht: gebruik `cd` dan `pwd` om het absolute pad naar je thuismap te leren.
- er is maar een grote boom van mappen: geen "schrijfletters". Verschillende opslagelementen worden benaderd door verschillende paden in de boom. Bijvoorbeeld op Android is de SD-kaart vaak te vinden in de map `/sdcard`. Op OSX worden harde schrijven en geheugenkaartjes zichtbaar in de map `/Volumes`. Meestal op andere soorten Unixsysstemen zijn opslagelementen zichtbaar in de map `/mnt`.
- de bijzondere naam "`..`" refereert altijd naar "een map omhoog".
- de bijzondere naam "`.`" refereert altijd naar "deze map".
- de belangrijkste commando's op mappen en bestanden: `find`, `mkdir`, `rmdir`, `ls`, `mv`, `ln`, `rm`, `touch`, `stat`.

  Gebruik `man` om te weten wat ze doen.

7 Vergeet niet voor later

Tijdens de loop van je opleiding zal je veel leren over Java, C, Python, misschien veel andere programmeertalen. Door tutorials, documentatie, je assistenten en medestudenten zal je leren over allerlei Unix commando's die iedereen kent.

Maar er zijn een paar "geheimen" dat zelfs gevorderde gebruikers soms vergeten; je vindt ze vast handig in de toekomst tijdens wetenschappelijke bezigheden bij de UvA:

- `du -k | sort -n` om te weten welke bestanden het meest ruimte in beslag nemen in een bepaalde map;
- `join` om columns uit verschillende databestanden naast elkaar te plakken;
- `cut` om columns te verwijderen;
- `tr` om comma's te vervangen door spaties of andersom;
- het commando `printf` in de shell (niet de functie in C) om iets te printen en op dezelfde regel blijven, in tegenstelling met `echo`;
- `awk` en `sed`: tekstverwerking. Ze kunnen allebei ongeveer dezelfde doen, maar sommige taken zijn eenvoudiger te bereiken met de ene dan de andere;
- De sectie "Parameter Expansion" van de manual page van bash (`man bash`, dan "/" om te zoeken);
- De programma's `tmux` en `screen` om je sessie en draaiende programma's te bewaren zelfs als je je terminal afsluit.
- GNUplot voor grafieken: https://www.gnuplot.info/
- GNU Make om een stappenplan van commando's te definiëren en automatisch uitvoeren: https://www.gnu.org/software/make/manual/make.html
- `telnet towel.blinkenlights.nl` om tijd te doden tijdens saaie colleges;
- Een andere manier om tijd te doden op een manier die eruit ziet alsof je aan het werk bent, omdat je met een terminal bezig bent: https://en.wikipedia.org/wiki/MUD (pas op, dit is verslavend)

Je wordt aangeraden om een lijst "nuttige" commando's met uitleg bij te houden in een schrijft, webpagina of blog. Daar kan je alleen de commando's inventariseren die je zelf leuk vindt en die je in je eigen werk gebruikt.

Note

The latest version of this document can be found online at http://science.raphael.poss.name/intro-unix.html. Alternate formats: Source, PDF.

8 Copyright and licensing

SC fingerprint: `fp:Dr7db84xPkmpTSA6mIlcHG6ELrn6hi0zmca2WDkjVagUQA`