

# Introductie UNIX voor A2/PC

Dit is een korte introductie UNIX voor de basisvaardigheden die nodig zijn voor het practicum van de vakken A2 (Bedrijfssystemen) en PC (Processen & Concurrency). Het heeft niet de pretentie een volledige manual te zijn.

Aspecten die aan bod komen zijn het werken met de “command line shell”, wat basiscommando’s van het filesysteem, en het editen en compileren van C programma’s, eventueel met behulp van een zogenaamde **Makefile**.

Wat je voor het practicum moet kunnen is wat in sectie 4 aan bod komt. Wat daarna komt wordt voorlopig niet gebruikt.

Bij sommige opdrachten zijn de instructies specifiek voor de UNIX workstations in practicumzaal TK 8/9 en de window manager die daarop draait; onder een andere Linux of UNIX installatie, of een andere window manager (bijv. Gnome of KDE) zal het mogelijk net wat anders werken.

## 1 Inloggen op de UNIX workstations

1. *Vóór je inlogt, selecteer eerst onder Options in het submenu **Session de Common Desktop Environment (CDE)**. Hiermee kies je welke window manager je gebruikt en hoe je “desktop” er dus uitziet.*
2. *Log nu in gewoon door je login en password in te tikken.*
3. *Klik het ‘User Registration Window’ weg door met de muis op de balk bovenaan te staan, de rechtermuisknop in te drukken, en Close te selecteren.*
4. *Doe hetzelfde met het ‘Help Viewer’ window.*
5. *Zet het irritante piepje af door de rechtermuisknop in te drukken als de cursor er-gens voor de achtergrond staat, en vervolgens **Tools** → **Desktop controls** te selecteren; dubbelklik met de linkermuisknop op de **Beep Style Manager** en zet het volume op nul.*

Klik openstaande windows weer weg zoals bij 3 hierboven.

Hieronder wat algemene tips over het omgaan met deze workstations:

- Je hebt vier verschillende desktops tot je beschikking: door op **One**, **Two**, **Three**, of **Four** te klikken in het midden onderaan verander je van desktop. Mocht je merken dat je opeens allerlei dingen van je desktop kwijt bent, dan ben je mogelijk naar een andere desktops verhuisd.
- **Uitloggen** kan door op **EXIT** te klikken in het controle paneel in het midden onderaan het scherm.

- Een **terminal window** kun je opstarten door de rechtermuisknop in te drukken, en `Hosts` → `This Host` te selecteren.
- Een **editor** kun je opstarten door onder de rechtermuisknop `Applications` → `Text Editor` te selecteren.

## 2 De Shell

Onder Linux en UNIX wordt vaak gewerkt vanuit een zogenaamd **command line shell** programma, dat in een **terminal window** draait. Hier worden dan commando's ingetikt, steeds afgesloten door Enter (Net zoals vroeger in MS-DOS, voor het geval je daar nog wel eens mee gewerkt hebt.)

Er zijn goede window interfaces voor UNIX of Linux zoals degene die je nu gebruikt, waar de gewone huis-tuin-en-keuken gebruiker alles kan doen wat hij of zij wil zonder ooit de ouderwetse shell te gebruiken. Toch is het nuttig om met de shell overweg te kunnen:

- Er zijn dingen die je eenvoudig vanaf de command line kunt doen die moeilijk of niet te doen zijn met grafische interfaces.
- Vanuit de shell kunnen we het operating system goed observeren, zonder dat bepaalde aspecten worden verborgen door de window manager. Dit is vooral interessant als we willen weten wat er “onder de motorkap” nou precies allemaal gedaan wordt door het operating system.
- De shell werkt op alle UNIX en Linux systemen vrijwel hetzelfde.
- Bij remote inloggen over een netwerk naar een ander UNIX of Linux machine zul je geconfronteerd worden met een shell.
- Tot slot, geoefende gebruikers kunnen veel sneller werken met de shell dan welke grafische interface dan ook. Dit is zeker het geval als je gebruik maakt van de mogelijkheden om handige ‘afkortingen’ te introduceren voor rijen commando's die je vaak wilt herhalen. Eén zo'n mogelijkheid waar we in detail op ingaan zijn zogenaamde `Makefile`'s.

Het oorspronkelijke shell programma bij UNIX heette `sh`. In de loop van de tijd zijn er steeds geavanceerdere versies ontwikkeld. De meest gebruikte shell programmas tegenwoordig zijn `tcsh` en `bash`. De verschillen zijn minimaal, zolang je niet allerlei geavanceerdere features gebruikt.

### Password veranderen

Als je wilt kun je je password veranderen.

*Start een terminal window op, door de rechtermuisknop in te drukken, en `Hosts` → `This Host` te selecteren.* Als je in een terminal window wil tikken, zul je soms dit window eerst selecteren door met de rechtermuisknop op het terminal window te klikken.

Het eerste wat je nu moet doen is je password veranderen. *Tik in een terminal window het command*

```
passwd
```

*in, gevolgd door een Enter.* Je moet nu één keer je oude password en twee keer je nieuwe password intikken.

Het kan zijn dat je nieuwe password geweigerd wordt, omdat het te makkelijk te raden is; kies in dat geval een ander password, totdat het geaccepteerd wordt.

Slechtgekozen passwords zijn een notoire zwakke plek in de beveiling van computersystemen. Een goede truuk om goede passwords te bedenken én te onthouden is om een zin te nemen, bijv. “ik betaal 5 dollar plus 10 % aan Piet Venema”, en dan de eerste letter van elk woord te nemen, `ib5$+10%aPV`.

### 3 Het UNIX filesystem

Deze sectie introduceert wat basiscommando's in UNIX/Linux om met het filesystem om te gaan.

Een belangrijk verschil tussen het filesystem onder UNIX en het filesystem zoals je het waarschijnlijk gewend bent op je PC, is dat UNIX een multi-user systeem is, waar meerdere gebruikers hun files opslaan.

*Open een terminal window, als je er nog geen hebt.*

1. *Geef het commando “cd”.*

Hiermee ga je naar je zogenaamde **home-directory**; `cd` is een afkorting voor “change directory”. Je home-directory is de plek waar je als gebruiker je eigen bestanden opslaat. Het is te vergelijken met de map “Mijn Documenten” die je op een Windows PC daarvoor gebruikt.

2. *Geef het commando “pwd”.*

Je krijgt nu te zien waar je momenteel bent in het het filesystem. `pwd` is een afkorting voor “path working directory”, dwz. het pad van de directory waar je momenteel aan het werken bent. Als je loginnaam `klaas` is, dan is je home-directory waarschijnlijk iets als `/home/infstud/klaas`.

3. *Geef het commando “ls”.*

Je krijgt nu de inhoud van je home-directory te zien; `ls` is een afkorting voor “list”. Als het goed is staat er nog niet veel in je home-directory.

4. We gaan nu een nieuw bestand (file) aanmaken in je home-directory met behulp van een editor:

- Start een editor. Op de UNIX machines gaat dit door onder de rechtermuisknop **Applications** → **Text Editor** te selecteren.
- Creëer een bestand met daarin wat willekeurige tekst. Bewaar dit bestand onder de naam `testfile` in je home-directory, met **File** → **Save As**.
- Als je in het terminal window weer het commando `ls` geeft, moet er nu een bestand `testfile` bij zijn gekomen. Zoniet, probeer het dan nog een keer. Zoja, sluit de editor af.

5. *Geef het commando “mv testfile testfile2”.*

Bekijk met `ls` de nieuwe inhoud van je home-directory. Met het commando `mv` kun je de naam van een file veranderen; het bestand “testfile” heet nu “testfile2”. `mv` is een afkorting van “move”.

6. Geef het commando “`cp testfile2 testfile3`”.

Bekijk met `ls` de nieuwe inhoud van je home-directory. Met het commando `cp` kun je een file kopiëren; `cp` is een afkorting van “copy”. Het nieuwe bestand “testfile3” is identiek aan “testfile2”; bekijk ze allebei maar eens met de editor.

7. Verwijder nu het bestand `testfile3` met het commando “`rm testfile3`”.

`rm` is een afkorting van “remove”. Waarschijnlijk wordt je gevraagd of je het bestand daadwerkelijk wilt verwijderen; hierop moet je “y” (voor “yes”) antwoorden.

8. Tik “`cp t`” in en druk dan op de TAB toets. Je zult zien dat er dan vanzelf `testfile2` wordt ingevuld. Dit is een handige truuk die veel tikwerk kan besparen. Maak het `cp` commando af, door bijv. `testfile3` of een andere naam als tweede argument in te tikken en een return te geven.

Nu wat commando’s om met directories te werken:

1. Geef het commando “`mkdir testdirectory`” in het terminal window, en bekijk met `ls` de nieuwe inhoud van je home-directory. Er is nu een nieuwe directory (oftewel map of folder) genaamd `testdirectory`; `mkdir` is een afkorting voor “make directory”.
2. Geef het commando `ls testdirectory` om de inhoud van de `testdirectory` te zien.
3. Geef het commando `ls testdirectory` om de inhoud van de `testdirectory` te zien.
4. Geef het commando “`mv testfile2 testdirectory`”. Bekijk met `ls` de nieuwe inhoud van je home-directory, en met `ls testdirectory` de nieuwe inhoud van de directory `testdirectory`. Het bestand “testfile2” is verplaatst naar de directory “testdirectory”.

NB. merk op dat we `mv` eerder gebruikt hadden om de naam van een bestand te veranderen. `mv` kan dus zowel de naam van een file veranderen als een file naar een andere directory verplaatsen!

5. Geef het commando “`cd testdirectory`”. `cd` is een afkorting van “change directory”. Bekijk wat het resultaat van `pwd` en van `ls` nu is: We zitten nu in de directory `testdirectory`.
6. Geef het commando “`cd ..`”, en bekijk het resultaat van `pwd` en `ls` nu. In UNIX/Linux betekent `..` één directory omhoog in de directoryboom.
7. Geef weer het commando “`cd testdirectory`” om naar de directory `testdirectory` te gaan.
8. Geef het commando “`ls ..`”. Hiermee krijg je de inhoud van een directory omhoog te zien. Geef het commando “`mv testfile2 ..`”, en bekijk het resultaat van `ls` en `ls ..` daarna. Je zult zien dat `testfile2` één directory omhoog is verhuisd.

NB. een verschil tussen het filesystem onder UNIX en het filesystem zoals je het wellicht gewend bent op je PC, is dat UNIX een multi-user systeem is, waar alle gebruikers hun files opslaan. Alle gebruikers van het facultaire UNIX systeem hebben hun files op hetzelfde filesystem staan. Je mag zelf alleen files opslaan in je eigen home-directory, en in sub-directories daarvan.

1. Geef het commando “`cd`” om terug te gaan naar je home directory, en kijk met `pwd` nog even wat het pad van je home directory was.
2. Geef het commando “`cd ..`” om terug één directory hoger te gaan, en bekijk het resultaat van “`ls`” en “`pwd`”.

Als het goed is zie je dat alle informatica studenten een home-directory in `/home/infstud` hebben. In principe kun je ook kijken wat er in de home-directory van iemand anders staat. Probeer maar eens.

Je kunt als gebruiker bepalen of anderen in jouw home-directory kunnen kijken, of zelfs per file of subdirectory opgeven of anderen erin mogen lezen of schrijven. Dit wordt uitgelegd in Sectie 7.

Er is ook een grafische file manager, die je kunt opstarten door de rechtermuisknop in te drukken als je op de achtergrond staat, en `Files` → `File Manager` selecteren.

Nog wat opdrachten om te oefenen:

**Opdracht 3.1** *Probeer de volgende operaties eens uit, in een willekeurige volgorde*

```
cd /
cd ..
cd
```

*en kijk steeds na elke operatie met `pwd` waar je ergens terecht bent gekomen op het filesystem.*

**Opdracht 3.2** *Probeer de volgende commando's eens uit*

```
history
!!
ls
ls -l
!l
!h
!c
```

*in verschillende volgordes.*

*Probeer eens op de pijltje-omhoog en pijltje-omlaag toets te drukken. Zo kun je alle oude commando's die je hebt gegeven terugzien. Probeer door op Enter te drukken zo'n oud commando te herhalen.*

*Druk eens op pijltje-omhoog en vervolgens een paar keer op pijltje-links of op Backspace. Zo kun je het vorige commando iets wijzigen. Dit is vooral handig als je een keer een tikfout in een commando hebt gemaakt.*

## 4 Compileren en Makefile's

Deze sectie legt uit hoe je een C programma kunt editen en compileren en hoe je hierbij gebruik kunt maken van zgn. Makefile's en het commando `make`.

### 4.1 De gnu C compiler gcc

Eerst gaan we een simpel C programmaatje editen en compileren:

1. Start de webbrowser Netscape op: klik hiervoor met de linkermuisknop op het driehoekje boven de aardbol om het bijbehorende menu te openen, en klik vervolgens met de linkermuisknop op Web Browser.
2. Open `http://www.cs.kun.nl/~erikpoll/A2` of `http://www.cs.kun.nl/~erikpoll/PC`. met File → Open Page.
3. Klik door naar de *practicumpagina* en vervolgens de *UNIX introductie cursus*.
4. Download hier de files `helloworld.c` en `Makefile`. Dit doe je met je de rechtermuisknop. Zet deze files in dezelfde directory, bijv. je home directory of een sub-directory hiervan.
5. Open `helloworld.c` in een editor.  
Je kunt een editor opstarten onder de rechtermuisknop Applications → Text Editor te selecteren, en vervolgens `helloworld.c` openen.
6. Wijzig nu de tekst “Hello World” in `helloworld.c` in iets anders, en sla de gewijzigde file op.
7. Compileer de gewijzigde file met het commando

```
gcc -o helloworld helloworld.c
```

Met `-o helloworld` vertel je de compiler `gcc` dat de gecompileerde file moet worden opgeslagen in een bestand `helloworld`.

8. Executeer nu de gecompileerde file `helloworld` door het commando “`helloworld`” te geven.

### 4.2 Een simpele Makefile

Zeker als je veel files moet compileren, wordt het benodigde tikwerk irritant. Dit is gelukkig te vermijden door gebruik te maken van een `Makefile` bestand en het bijbehorende commando “`make`”:

- Bekijk het bestand `Makefile`. Hierin staan de instructies voor het compileren van `helloworld.c`. Er staan ook nog enkele andere instructies in de file.

Merk op dat de instructies voor het compileren van `helloworld.c` geïndenteerd zijn met een “tab”, en dat deze onder een kopje `helloworld:` staan, wat niet geïndenteerd is.

Als je in een shell het commando `make hallo` geeft, dan worden de instructies onder het kopje `hallo` uitgevoerd. (NB `make` kijk in de Makefile in de huidige working directory.)

Probeer de file `helloworld.c` nog maar eens te wijzigen en vervolgens met `make helloworld` te compileren.

- Er staan nog meer “kopjes” in Makefile. Probeer maar eens uit wat het effect van de commando’s `make clean`, `make realclean`, en `make gethello` doen.

### 4.3 Een ingewikkeldere Makefile

Je kunt Makefile’s gebruiken voor allerlei doeleinden. Voor ingewikkeldere Makefiles biedt `make` wat extra opties, namelijk het gebruik van afkortingen en afhankelijkheden. Om een voorbeeld te zien, ga naar de practicumwebpagina en download daar de C files `child.c` en `parent.c` en de bijbehorende Makefile.

Deze Makefile maakt gebruik van afkortingen. Er wordt bijvoorbeeld een afkorting `CC_OPTS` gedefinieerd voor de opties van de compiler `gcc`, en een afkorting `TARGETS` voor de lijst van C files die we hebben. Het is een conventie om afkortingen in hoofdletters te schrijven. Als je een afkorting gebruikt moet je er `$(...)` omheen zetten.

Er kunnen twee soorten afhankelijkheden worden gedeclareerd in Makefile’s, namelijk afhankelijkheden tussen commando’s in de Makefile, en afhankelijkheden tussen commando’s in de Makefile en files.

- Een voorbeeld van de eerste soort is

```
all: $(TARGETS)
```

wat een afkorting is voor

```
all: parent child
```

Dit zegt dat bij `make all` zowel `make parent` als `make child` moet worden uitgevoerd.

- Een voorbeeld van de tweede soort is

```
parent.o: parent.c
```

```
...
```

Dit zegt dat `make parent.o` alleen hoeft worden uitgevoerd als `parent.c` nieuwer is dan `parent.o`.

Probeer maar eens uit door meerdere keren het commando `make parent.o` te geven: alleen de eerste keer gebeurt er iets, de volgende keren krijg je de boodschap dat ‘`parent.o`’ is up to date.

Het netto effect van al deze afhankelijkheden is dat bij het commando `make all` alleen de C files die gewijzigd zijn opnieuw gecompileerd worden.

## 5 Meer UNIX commando’s

In deze sectie staan wat handige truuks die het gebruik van de shell een stuk interessanter maken.

## 5.1 Wildcards

Behalve de TAB toets, is een andere truuk die veel tikwerk kan besparen het gebruik van de speciale karakters \* en ?. Bijvoorbeeld

```
cp * testdirectory
```

kopieert alle bestanden uit de huidige directory naar `testdirectory`, en

```
rm testfile?
```

verwijdert alle bestanden met een naam die begint met `testfile` gevolgd door precies één andere karakter.

## 5.2 Opties

Veel van de commando's die tot nu toe genoemd zijn kunnen allerlei opties krijgen. Probeer eens het commando

```
ls -l
```

Je krijgt dan een uitgebreidere listing (l staat voor lang) te zien, met o.a. het tijdstip dat de files zijn gemaakt, van wie ze zijn, en hoe groot ze zijn, en wie er toestemming heeft om de file te lezen en te schrijven; meer hierover later. Probeer ook eens “`ls -R`” voor een recursieve listing, of “`ls -lR`” voor een uitgebreidere recursieve listing. Met

```
ls -lR /
```

krijg je een uitgebreidere recursieve listing van het *hele* filesysteem te zien. Dit duurt waarschijnlijk erg lang; druk op Ctrl-c om het te onderbreken.

## 5.3 Meer informatie over commando's

Met het commando `man` kun je informatie over commando's, de zogenaamde “manual page”, opvragen. Bijvoorbeeld met `man ls` kun je alle opties van `ls` te zien krijgen. Met PageUp en PageDown kun je door deze informatie op en neer, met “q” (voor “quit”) verlaat je de manual page. Helaas wordt je bij `man` vaak overspoeld door een waslijst aan opties en allerlei informatie waar je helemaal niet in geïnteresseerd bent, en kun je beter op het web zoeken.

Er zijn weliswaar kleine verschillen tussen verschillende versies van UNIX en Linux, zeker wat betreft de meer obscure opties. Maar voor de meest gebruikelijke commando's en opties zul je hier niet snel iets van merken.

## De commando's cat, more, grep en wc

Nog wat handige UNIX/Linux commando's:

- `cat`

Dit drukt een of meer bestanden af in een terminal window. Bijvoorbeeld, `cat testfile testfile2 testfile3` drukt de drie bestanden `testfile`, `testfile2`, en `testfile3` achter elkaar af.

- **more**

Dit doet bijna hetzelfde als `cat`; het enige verschil is dat `more` het afdrucken onderbreekt telkens als het terminal window vol is, en wacht tot de gebruiker een toets indrukt, zodat je pagina voor pagina door het bestand ‘scroll’t. Door op ‘q’ (voor ‘quit’) te drukken beëindig je het scrollen. `more` is erg handig om snel een bestand te bekijken. Net als `cat` kan `more` ook op meerdere files werken, bijvoorbeeld `more testfile testfile2 testfile3`.

- **grep**

Hiermee kun je in bestanden zoeken naar voorkomens van een woord. Bijvoorbeeld, `grep Piet testfile` drukt alle regels uit het bestand `testfile` af waarin het woord ‘Piet’ voorkomt. Net als `cat` en `more`, kan `grep` ook meerdere files werken.

- **wc**

Hiermee tel je het aantal regels, woorden, en karakters in een file. Voor meer info, probeer man `wc`.

Voor al deze commando’s kun je ook wildcards gebruiken. Bijvoorbeeld, met

```
more *.txt
```

bekijk je alle files met een naam die eindigt op `txt`, en met

```
grep Piet */*.c
```

zoek je naar regels waar ‘Piet’ in voorkomt in alle C files in de subdirectories.

## 5.4 Combineren van commando’s met | en >

UNIX/Linux biedt allerlei mogelijkheden om commando’s aan elkaar te knopen. Hierdoor wordt de shell een krachtige tool, en kun je eenvoudig dingen doen die je met een grafische file manager niet zullen lukken.

**Opdracht 5.1** *Ga naar je home-directory met “`cd` ”. Kopieer alle bestanden uit de directory `/vol/practica/a2/UNIXintro` naar je eigen homedirectory (of een subdirectory daarin). Dit kan in een keer met het commando*

```
cp /vol/practica/a2/UNIXintro/* ~
```

*Kijk met `ls` of het gelukt is. Experimenteer wat met de commando’s `wc`, `grep`, `more`, en `cat` op deze files.*

### Output redirection

Veel commando’s – zoals `ls`, `grep`, `cat`, etc. – sturen uitvoer naar het beeldscherm. Met `>`, het ‘groter-dan’ teken, kun je deze uitvoer in een file opslaan. De technische kreet hiervoor is output redirection.

Probeer achtereenvolgens de volgende commando’s in je home-directory

```
ls -l > mijndir
ls -l .. > mijnsuperdir
ls -lR .. > mijnsuperdir-rec
cat mijn* > alle3
```

Bekijk vervolgens – of tussendoor – de inhoud van je home-directory met `ls`, en bekijk de inhoud van de nieuwe files in home-directory met behulp van `more`, bijv. met “`more mijndir`”.

Output redirection met `>` is erg handig als een commando te veel uitvoer naar het beeldscherm stuurt om te bekijken.

## Pipes

Net zoals “`>`” gebruikt kan worden om de output van een commando naar een file te sturen, kan “`|`” gebruikt worden om de output van een commando door te sturen als invoer voor een ander commando. Bijvoorbeeld

```
cat * | wc
```

plakt alle files in een directory achter elkaar en telt vervolgens het aantal karakters, woorden en regels. Probeer dit maar eens uit. Probeer eventueel ook wat van de opties van `wc`.

Je kunt `|` beschouwen als een pijpleiding die de uitvoer van bijv. `cat *` doorsluisst naar `wc`.

Een ander voorbeeld van het gebruik van `|` is

```
ls -R | more
```

waarmee je door lange directory-listings kunt scrollen, of

```
ls -R | grep klaas
```

waarmee je in directory-listings naar een patroon kunt zoeken.

## 6 Processen

Het doel van deze oefeningen is je vertrouwd te maken met het begrip “proces”. Dit is een van de belangrijkste begrippen in de informatica! Alle operating systems zijn georganiseerd rond het begrip proces: *alles* wat op een computer gebeurt is onderdeel van een proces.

Open twee terminal windows naast elkaar. Geef in een van de twee het commando

```
ps
```

Je krijgt een lijst te zien van alle processen die er in dit window draaien. Als het goed is bestaat deze lijst maar uit één proces, namelijk de shell `bash`. Met

```
ps -f
```

krijg je wat meer informatie over dit proces.

Het commando `ps` kan weer allerlei opties meekrijgen. Probeer bijvoorbeeld, als je loginnaam `inf098` is,

```
ps -u inf098
```

Je krijgt nu een hele waslijst met al je processen te zien. Als het goed is zie je hier in elk geval de twee bash shells die in je beide terminal windows draaien. De andere processen die hierbij staan zijn allemaal processen die deel uitmaken van de desktop omgeving.

Afhankelijk van de opties geeft `ps` voor elk proces z'n

- PID: de process-id
- TTY: het virtuele beeldscherm oftewel het terminal window waar het proces is opgestart,
- TIME: de rekentijd die het proces heeft verbruikt,
- CMD: het commando waar dit proces mee is opgestart,
- UID: het user-id van de eigenaar van het proces

Kijk bij `man ps` eventueel voor meer opties en uitleg over `ps`.

Merk op dat processen zoals `bash` niet veel rekentijd gebruiken; de meeste tijd doen deze processen namelijk niets, maar zitten ze enkel te wachten tot er wat wordt ingetikt.

PID's en UID's, unieke nummers om processen en gebruikers te identificeren, spelen een belangrijke rol in de boekhouding van het operating system. Kun je zien wat jouw user-id is?

Geef in een van de terminal windows het commando

```
xcalc
```

Dit opent een calculator. Probeer deze maar eens uit, maar sluit hem niet af.

Kijk in het andere terminal window met het commando `ps -a` welke processen er leven. Probeer ook eens `ps -al`, en kijk in de PPID (parent process id) kolom wat de parent van dit calculator proces is.

Merk op dat er geen prompt verschijnt in het terminal window waar je het commando `xcalc` gaf. Als je probeert hier iets in te tikken zul je merken dat het niet lukt. Dit komt doordat nu het shell proces `bash` tijdelijk op non-actief staat, zolang het `xcalc` proces in de voorgrond draait.

Sluit het `xcalc` window weer af, door op de "X" knop rechtsboven in het window te klikken. Zodra je dit doet, wordt het `bash` proces weer actief, en krijg je de prompt weer te zien.

## 6.1 Opstarten van processen in de achtergrond

We kunnen de calculator ook opstarten met

```
xcalc &
```

Probeer dit eens uit. Kijk met `ps` weer welke processen er leven.

Het verschil tussen "`xcalc`" en "`xcalc &`" is dat met "`xcalc &`" het calculator proces in de *achtergrond* wordt opgestart. Het `bash` proces blijft in dat geval gewoon doordraaien, in de *voorgond*. Met het commando `xcalc` zonder `&` wordt het calculator proces in de voorgrond opgestart. Als we invoer in een window intikken, gaat die invoer naar het proces dat in de voorgrond draait.

## 6.2 Killen van processen

Geef in een terminal window weer het commando

```
xcalc
```

en druk vervolgens op Ctrl-c in datzelfde terminal window. Wat gebeurt er?

Probeer het eventueel nog een keer, en kijk met `ps -a` in een ander window welke processen er leven vóór en na je op Ctrl-c drukt.

Met Ctrl-c kun je processen om zeep helpen. Een andere manier om dit te doen is met het commando

```
kill pid
```

waar *pid* het process-id nummer is van het proces dat we willen afbreken.

Sommige processen zijn wat hardnekkiger en overleven `kill`. In dat geval kun je de optie `-9` meegeven; met

```
kill -9 pid
```

breng je gegarandeerd elk proces om zeep (tenminste, als het een van je eigen processen is).

## 6.3 Suspenden van processen

Geef het commando

```
xcalc
```

en druk vervolgens op Ctrl-z in het terminal window. (Misschien wil je het `xcalc` window eerst even verslepen zodat het niet voor het terminal window hangt).

Waarschijnlijk verschijnt er een boodschap in het terminal window. Probeer de calculator te gebruiken, en probeer het `xcalc` window te resizen. Kijk met `ps` of het `xcalc` proces nog leeft.

Met Ctrl-z is het `xcalc` proces *ge-suspend*. Het proces bestaat nog wel, maar is als het ware “bevroren”, en reageert nergens meer op.

Er zijn twee commando’s om het `xcalc` proces weer tot leven te wekken:

- het commando `fg`, voor “foreground”; het gewekte proces draait dan verder in de voorgrond.
- het commando `bg`, voor “background”; het gewekte proces draait dan verder als achtergrondproces.

Probeer `fg` en `bg` maar eens nadat je een `xcalc` proces ge-suspend hebt met Ctrl-z.

### Alternatieven voor `ps`

Een alternatieve manier om in de gaten te houden welke processen er draaien is het commando `top`. Probeer dit commando maar eens uit. `top` sorteert de processen naar hoeveel procent van de CPU tijd ze verbruiken. Merk op dat `top` zelf ook in de lijst van processen voorkomt. Door `u` in te tikken bij `top` kun je een user opgegeven van wie je de processen wil zien. Door

een ? in te tikken krijg je meer informatie over alle interactieve commando's die je in `top` kunt geven. Met `Ctrl-c` onderbreek je het `top` proces.

Er is op de UNIX machines ook een mooie grafische variant met `top`, namelijk `rechtermuisknop` → `Tools` → `Process Manager`. Probeer deze maar eens. Je zult mogelijk zien dat er ook andere mensen op je machine aan het werken zijn.

### **Opdracht 6.1** *Wat oefeningen met kill:*

1. Geef in een window het commando `xclock`, en probeer vervolgens vanuit een ander window deze klok om zeep te helpen met `kill`.
2. Geef in een terminal window het commando `xclock`. Probeer vervolgens vanuit een ander terminal window het eerste terminal window om zeep te helpen, door het `bash` proces wat daar draait af te schieten met `kill`.
3. Geef in een terminal window het commando `xclock &`. Probeer vervolgens weer vanuit een ander terminal window het eerste terminal window om zeep te helpen, door het `bash` proces wat daar draait af te schieten met `kill`. Is er enig verschil met wat er bij 2 gebeurde?
4. Start Netscape weer op, en probeer Netscape vervolgens met een `kill` om zeep te brengen.
5. Je kunt eens kijken met `ps -Al`, `top`, of het `Process manager` window, welke processen er nog meer draaien op je machine, en proberen ze met `kill` of `kill -9` af te schieten. In de `Process manager` window doe je dit ook door met de rechtermuisknop te klikken. Zolang je als gewone gebruiker bent ingelogd, niet als `root`, moet dit in principe geen kwaad kunnen :-)

*(Misschien moet je wat van de verschillende opties van `ps` uitproberen om erachter te komen welke processen er echt allemaal draaien; zie `man ps`.)*

Het `kill`-en van processen met `kill` is in de praktijk nog wel eens nuttig, bijvoorbeeld als er een bepaald proces is vastgelopen of op hol is geslagen.

## **6.4 Samenstellen van processen**

Processen kunnen worden samengesteld met `;` en `&`. De operatie `;` is *sequentiële compositie*, en je zult deze operatie wel kennen uit programmeertalen. De operatie `&` is waarschijnlijk minder bekend.

### **Opdracht 6.2** 1. *Kijk bijvoorbeeld wat het effect is van de commando's*

```
ls ; ls -l
xclock ; xcalc
```

*Voor het onderste commando zul je de klok moeten afsluiten (met `rechtermuisknop` → `Close`) om te zien wat er daarna gebeurt. Kijk eventueel met `ps` wanneer welke processen er draaien vóór en na het afsluiten van de klok.*

2. De operatie `&` is behalve als unaire post-fix operatie ook als binaire infix operatie gebruikt worden. Probeer de commando's hieronder uit om erachter te komen wat de verschillen tussen `;` en `&` zijn, en wat de commando's `echo` en `sleep` doen.

```
echo slaap; sleep 2; echo wakker
xclock ; sleep 3 ; xcalc
xclock ; sleep 3 ; xcalc &
xclock & xcalc
xclock & xcalc &
sleep 3; xclock & (xcalc ; ls)
```

*Verzin eventueel zelf nog wat variaties.*

3. *Probeer de onderstaande commando's uit*

```
xclock ; xclock & xcalc ; xcalc
sleep 3 ; xclock & xcalc
```

*en probeer uit het gedrag af te leiden hoe we de haakjes moeten lezen.*

4. *Wat is het verschil tussen de volgende twee commando's ?*

```
sleep 3 ; (xclock & xcalc)
(sleep 3 ; xclock) & (sleep 3 ; xcalc)
```

*Wat denk je dat in het algemeen het verschil is tussen `sleep n ; (A & B)` en `(sleep n ; A) & (sleep n ; B)` voor willekeurige `n`, `A` en `B` ?*

De operatie `;` is *sequentiële compositie*, de operatie `&` is *parallele compositie*. In sommige programmeertalen is het ook mogelijk om processen parallel op te starten, vaak op hele andere en ingewikkeldere manieren dan `&`. (Bijvoorbeeld, in Java met threads).

De pipe-operatie `|` die we eerder gezien hebben is ook een vorm van parallele compositie. Probeer maar eens

```
xclock | xclock
```

Dit is natuurlijk een gedegenereerd gebruik van een pipe; het linker proces `xclock` stuurt namelijk geen uitvoer naar het terminal window, en het rechter proces `xclock` krijgt geen file of ander invoer als argument.

De operatie `|` is interessanter dan `&`, omdat er met `|` communicatie tussen de twee processen is. Processen die parallel zijn opgestart met `&` zijn volledig onafhankelijk.

Er is een tak van de informatica, genaamd procesalgebra, die zich bezighoudt met de theorie van het samenstellen van processen met operaties als `;`, `&`, en `|`, en bijbehorende regels voor 'gelijkheid' zoals de regel `sleep n ; (A & B) = (sleep n ; A) & (sleep n ; B)` uit 4. in de opgave hierboven.

## 7 Het File Systeem

Omdat UNIX/Linux multi-user systemen zijn, wordt er voor alle files en directories bijgehouden wie de eigenaar ervan is. Bijvoorbeeld, als `ls -l` het volgende oplevert

```
-rw-r--r--  1 inf099  infstud  145559 Oct 26 14:57 practicum.pdf
drwx-----  1 inf099  infstud   1024  Jan 26 18:13 mail
```

dan zijn `practicum.pdf` en `mail` van user `inf099`.

In de listing die je met `ls -l` krijgt beginnen alle regels met tien speciale karakters, de *file attributen*, bijvoorbeeld voor `practicum.pdf` hierboven `-rw-r--r--`.

De eerste letter geeft aan *wat voor soort file het is*. Meestal is de eerste letter een `-`, voor file, of een `d`, voor directory. In het bovenstaande voorbeeld is `mail` dus een directory, en `practicum.pdf` een file.

In een grafische file manager kun je natuurlijk vaak aan verschillende plaatjes zien met wat voor soorten files je te maken hebt. Je kunt je shell zo instellen dat-ie verschillende kleuren voor directories en bestanden gebruikt.

De volgende 9 letters geven de *file permissies*. Deze bepalen wie er allemaal toestemming hebben om een bepaalde file te lezen, overschrijven, of executeren, en voor directories ook wie er allemaal toestemming hebben om de directory binnen te gaan. Voor files betekent

r	read
w	write
x	execute

`x` permissie om de file te executeren; voor directories betekent `x` permissie om de directory binnen te gaan.

De 9 letters die de *file permissies* aangeven bestaan uit 3 groepen van 3. De eerste groep van drie geeft de permissies van de eigenaar (**user**), de tweede groep van drie geeft de permissies van de groep (**group**), de derde groep van drie geeft de permissies van alle anderen (**other**):

```
- | rw- | r-- | r--  1 inf099  infstud  145559 Oct 26 14:57 practicum.pdf
d | rwx | --- | ---  1 inf099  infstud  1024  Jan 26 18:13 mail
  |     |     |     |
  | user| group| other
```

In het bovenstaande voorbeeld mag iedereen dus de file `practicum.pdf` lezen maar alleen de user `inf099` zelf mag de file schrijven. Alleen `inf099` mag de directory `mail` lezen, schrijven en binnengaan.

Veranderen van de permissies van een file gaat met het commando `chmod`, gevolgd door meerdere karakters **MODE**, gevolgd door de naam van een of meer files of directories

```
chmod MODE file
```

De karakters **MODE** bestaan uit drie stukken:

1. een of meer van de karakters **u**, **g**, **o**, **a**,
2. gevolgd door **+** of **-**, voor “geef” en “neem af”
3. gevolgd door een of meer van de karakters **r**, **w**, **x**.

De betekenis van **r**, **w**, **x** is als in de tabel boven. De betekenis van **u**, **g**, **o**, **a**, is gegeven door de tabel hieronder: Wat voorbeelden

u	user
g	group
o	other
a	all

```

chmod u+w mail geef aan de user (u) write (w) permissie
chmod u-w mail ontnem de user (u) write (w) permissie
chmod g+w mail geef aan de groep (g) write (w) permissie
chmod o+x mail geef aan others (o) execute (x) permissie
chmod a+x mail geef aan iedereen (a) execute (x) permissie
chmod go-r mail ontnem group en others read (r) permissie

```

**Opdracht 7.1** *Creëer in je home-directory bestanden klad, publiek, geheim, definitief, bijvoorbeeld met een editor. De inhoud doet er niet toe.*

*Verander nu de file-permissie van de de bestanden klad publiek geheim definitief zodanig dat*

- *klad door iedereen gelezen en geschreven mag worden*
- *publiek door iedereen gelezen maar enkel door jou geschreven mag worden*
- *geheim enkel door jou gelezen en geschreven mag worden*
- *definitief door iedereen gelezen maar door niemand geschreven mag worden*

*Controleer of dit allemaal goed werkt. Vraag aan je buurvrouw of buurman of ze willen kijken welke van jouw files ze kunnen lezen en schrijven.*

Tip: jezelf schrijf-permissie ontnemen voor een file is in de praktijk erg handig om te voorkomen dat je een file per ongeluk verandert.

Je kunt `chmod` ook een getal als parameter meegeven ipv. iets als `og-rw`. Het idee is dat de permissies `x`, `r`, `w` de volgende waarden hebben: Je kunt deze waarden optellen, bijvoorbeeld

1	execute
2	write
4	read

6 komt overeen met `rw-`, en 7 komt overeen met `rwX`. Vervolgens kun je deze cijfers aan elkaar plakken, in de gebruikelijke volgorde: user, group, other. Dus bijvoorbeeld 761 staat voor `rwX` permissie voor de user, `rw-` permissie voor de group, `r--` permissie voor others.