
Dit document bevat algemene informatie over computers, en meer in het bijzonder over het operating systeem UNIX/Linux. Een # betekent: voor beginners; en *: voor gevorderden. Het document wordt gebruikt bij het eerstejaars college *Programmeermethoden*, Universiteit Leiden, najaar 2018, zie

www.liacs.leidenuniv.nl/~kosterstwa/pm/

en BlackBoard voor de Python-variant Programmeermethoden NA.

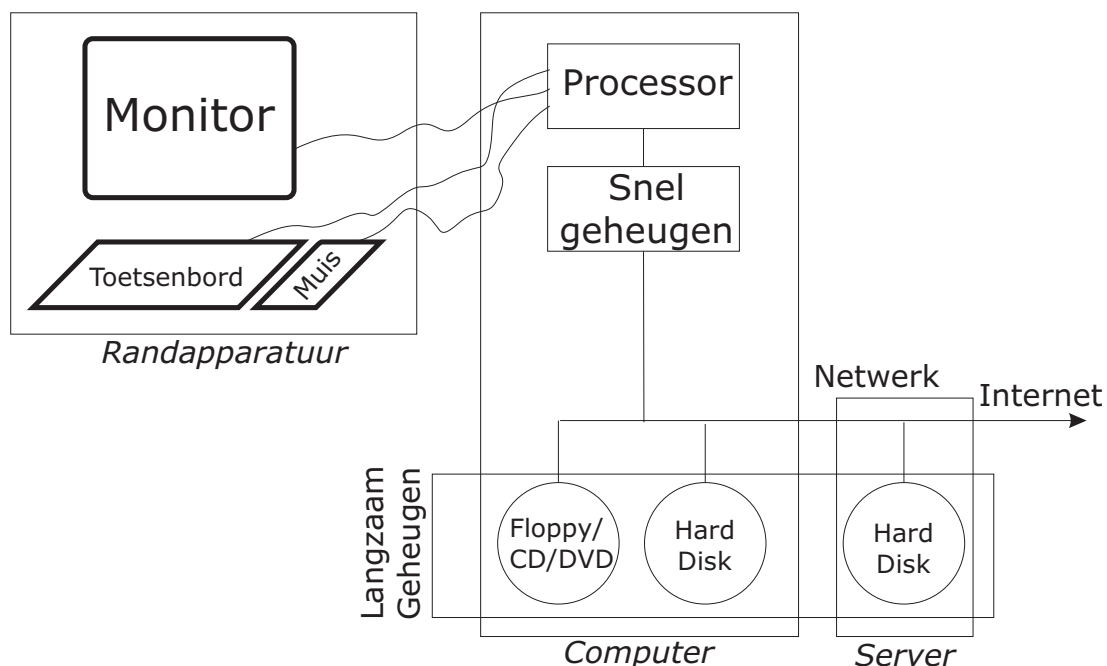
Met dank aan allen die aan deze tekst hebben bijgedragen.
Walter A. Kusters, Leiden, 31 augustus 2018.

1 Computers en programmeren

Dit deel van het dictaat biedt enige elementaire kennis van computers. Met deze kennis als bagage is het gemakkelijker te begrijpen waar het vak Programmeermethoden over gaat en hoe het zich verhoudt tot andere vakken.

1.1 De computer algemeen

De computer en alle erop aangesloten apparatuur is als volgt voor te stellen:



Figuur 1: Computer overzicht

Het beeldscherm en het toetsenbord, maar ook de muis, luidsprekers, enzovoort, vormen de randapparatuur van de computer. Via deze apparatuur communiceren de computer en de menselijke gebruiker met elkaar.

In de computer zelf zitten verschillende onderdelen. De belangrijkste daarvan zijn de processor, het geheugen en de diskdrives. Daarnaast bevat ze nog andere onderdelen, die ervoor zorgen dat de bouwblokken van de computer met elkaar kunnen communiceren.

De *processor* heeft de centrale rol in de computer. Deze chip, die ook wel CPU (central processing unit, ofwel: centrale rekeneenheid) genoemd wordt, verricht het echte (reken)werk. Ook zorgt deze chip ervoor dat alle andere onderdelen van de computer aangestuurd worden. Bekende voorbeelden van processoren zijn de Intel Core, Intel Pentium, Intel Xeon (van het merk Intel), de Athlon en Ryzen (van AMD), de SPARC (van SUN) en de ARM.

Processoren verrichten rekenwerk door een stappenplan uit te voeren. Dit stappenplan is volgens bepaalde afspraken in nullen en enen opgesteld. Intel en AMD processoren houden zich aan dezelfde afspraken; beide processoren kunnen dezelfde stappenplannen uitvoeren. SUN en ARM processoren houden zich aan andere afspraken. Het is daarom niet mogelijk een stappenplan dat voor een Intel processor bedoeld is, op een SUN of ARM uit te voeren. Computers die Intel stappenplannen uit kunnen voeren, worden PC's (Personal Computers) genoemd.

Omdat in het Engels een stappenplan vaak met “program” wordt aangeduid, wordt in het Nederlands een computerstappenplan meestal een *programma* genoemd. Dat is vanwege de lengte van het woord ook wel zo gemakkelijk. De opdrachten waaruit een programma bestaat — je kunt dan denken aan “tel a en b op” — worden *instructies* genoemd.

Om de instructies in het programma bij te houden, zijn computers uitgerust met een (snel) geheugen (*memory*) waar de processor direct bij kan. De processor kan dit geheugen ook gebruiken om aantekeningen tijdens het rekenen bij te houden.

Het nadeel van het snelle geheugen is dat hetgeen er in opgeslagen is slechts bewaard blijft zolang de computer aanstaat. Omdat er over het algemeen een heleboel programma's zijn (een Internetbladerprogramma (browser), een tekstverwerker, een rekenmachine, enzovoort), en mensen daarnaast ook fors veel informatie via hun computer op willen bergen (werkstukken, muziek, films), hebben computers ook een veel groter, maar langzamer, geheugen. Dit langzame geheugen is heel divers. In de computer zelf zit meestal een harddisk. Daarnaast kun je een CD/DVD of USB-stick in je computer stoppen, zodat de informatie op de CD/DVD/stick beschikbaar komt voor de processor.

Als de gebruiker een programma wil gebruiken dat in het langzame geheugen is opgeslagen, wordt het programma van het langzame geheugen in het snelle geheugen gekopieerd zodat de processor daarna de instructies uit kan voeren.

Een andere mogelijkheid om bij informatie te komen, is deze informatie via een netwerk van kabels (buiten de computer) uit de harddisk van een andere computer te halen. Computers waaruit je informatie kunt halen, of waarin je informatie kunt opslaan, worden *servers* genoemd. Sommige servers stellen hun harddisk zodanig beschikbaar aan andere computers, dat het op daarop net lijkt of de netwerkharddisk in de “locale” computer zit.

1.2 De C++ compiler

In essentie bestaat een stappenplan dat de computer uit kan voeren dus uit nullen en enen (*binaire code*). Om zo'n programma te maken (oftewel: om te *programmeren*), moet een opeenvolging van nullen en enen gecreëerd worden, waarbij de afspraken van de processor strikt nageleefd moeten worden. Omdat het een erg vervelende bezigheid is om dit met de hand te doen (voor elke processor opnieuw), hebben informatici gezocht naar methoden om het programmeren te vergemakkelijken. Eén van de oplossingen die ze bedacht hebben is de *compiler*. Een compiler is een programma dat een tekst (opgeslagen in langzaam geheugen, bijvoorbeeld een harddisk) omzet in binaire code (dat ook weer wordt opgeslagen in langzaam geheugen). Zodoende hoeft de programmeur niet te weten hoe nullen en enen gecomponeerd dienen te worden. Verder is het

zo mogelijk om dezelfde tekst voor zowel een SUN of Mac, als een PC onder Windows of Linux te compileren — dat levert allemaal verschillende executables op.

Hoe ziet nu de tekst eruit die aan de compiler gegeven wordt? Eigenlijk is dat ook een stappenplan, alleen dan opgesteld in een andere, makkelijkere taal dan die van nullen en enen. Er zijn veel verschillende van deze talen (Pascal, Java, Python, . . .) waarvan C++ er één is. C++ is de opvolger van de taal C. Een compiler kan maar één taal omzetten naar binaire code (*compileren*). In C++ zijn echter ook de “oude” instructies uit C nog toegestaan. In het vak Programmeermethoden zullen we leren hoe een stappenplan (ofwel: *programma*) eruit moet zien zodat een C++-compiler die tekst kan compileren.

We zagen hierboven dat het woord “programma” nogal algemeen is. Om een goed onderscheid te kunnen maken, wordt een programma dat uit nullen en enen bestaat ook wel een *executable* of een *binary* genoemd. De tekst van een programma wordt ook wel de broncode (of: *sourcecode*) genoemd.

1.3 De Python interpreter

Net als bij C++, moet ook in het geval van Python een opeenvolging van nullen en enen gecreëerd worden die de computer kan uitvoeren. Echter wordt er bij Python geen gebruik gemaakt van een compiler, maar van een interpreter. De interpreter leest een tekst (wederom opgeslagen in langzaam geheugen), interpreteert deze als programma en voert deze direct uit. De interpreter voert het programma eigenlijk stap voor stap uit. Dit in tegenstelling tot C++, waar een compiler eerst het gehele programma vertaalt en het daarna kan worden uitgevoerd. De Python-interpreter bevat in wezen al een binaire code voor de verschillende stappen die in het programma kunnen voorkomen.

Bij Python is het dus zo dat het uitvoerbare bestand niet uit nullen en enen bestaat, maar uit een (tekstuele) Python broncode. Een dergelijk programma dat direct door een interpreter kan worden uitgevoerd wordt over het algemeen een “script” genoemd. Er bestaan overigens veel verschillende scripttalen: Perl, Ruby, JavaScript, shell script, enz.

1.4 Operating Systemen

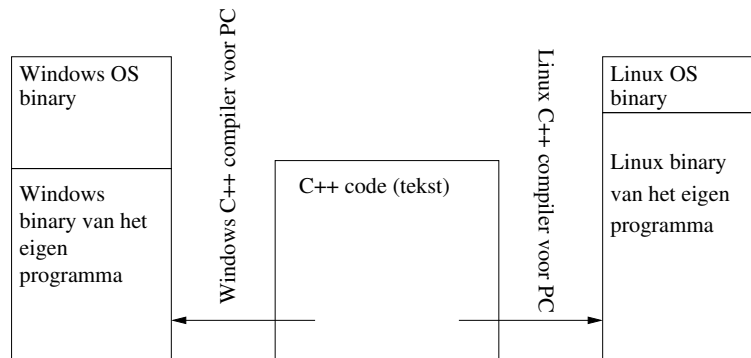
Door de meeste programma's moet een grote hoeveelheid aan taken uitgevoerd worden — denk hierbij aan: zaken afdrukken op het scherm, het toetsenbord uitlezen, gegevens opslaan op de harde schijf, etc. Het lijkt om meerdere redenen niet praktisch al deze “standaard” taken mee te compileren met elk afzonderlijk programma:

- alle binaries hebben dan voor een groot deel dezelfde inhoud, waardoor het geheugen snel vol komt te zitten;
- ook al hebben computers dezelfde processor, vaak hebben ze niet hetzelfde toetsenbord, muis, beeldscherm, harddisk, netwerk, enzovoort. Om alle verschillende soorten apparaten aan te kunnen sturen, zou de binary heel groot moeten zijn.

De voor de hand liggende oplossing waarvoor gekozen is, is de computer uit te rusten met een binary die direct bij het aanzetten van de computer in het geheugen geplaatst wordt: het *Operating System* (OS). Voor veel voorkomende taken kunnen programma's dan gebruik maken van de functionaliteit die deze binary biedt.

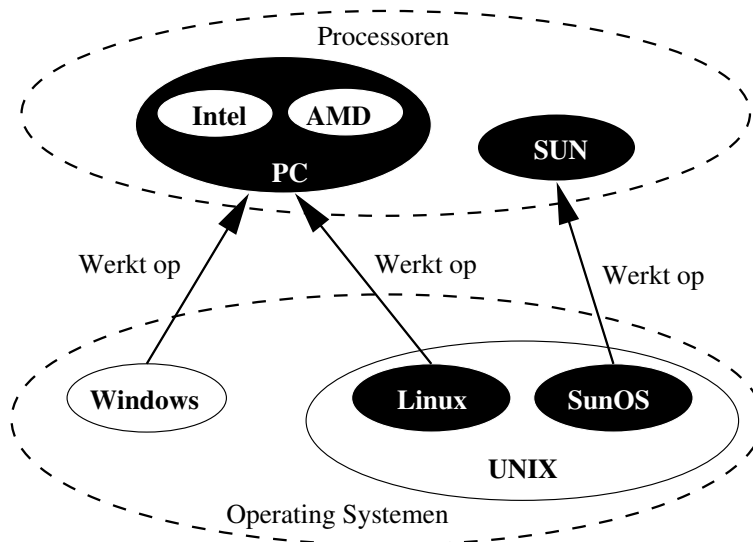
Een probleem (maar eigenlijk ook een voordeel) is dat er per processor vaak meerdere operating systems bestaan. Zo zijn er op de Snellius PC's twee operating systemen geïnstalleerd: Linux en Windows. En er zijn ook nog Mac's van Apple. Als je een programma compileert onder

één van die operating systemen, heeft de compiler daarbij in gedachten dat het OS bepaalde functionaliteit biedt (zodat hij die bijvoorbeeld niet in de executable hoeft te stoppen). Omdat verschillende operating systems vaak andere standaard-functionaliteiten bieden is een gevolg hiervan dat een C++-binary die onder Linux gemaakt werd, niet onder Windows werkt — en andersom — (zie Figuur 2).



Figuur 2: Het operating systeem en de compiler

Een ander gevolg is dat sommige code maar op één van de operating systemen gecompileerd kan worden. In Windows, bijvoorbeeld, kan een programma gebruik maken van schuifbalken, knoppen en menu's die in de operating systeem binary zitten. Niemand heeft echter deze zelfde schuifbalken, knoppen en menu's in een Linux binary gestopt (ook al probeert men dat wel). Het is daardoor onmogelijk deze typische Windows-programma's onder Linux te hercompileren. Dat verklaart ook waarom veel Windows-programma's niet onder Linux bestaan, en andersom. Aan de andere kant zijn er ook operating systemen die zoveel van elkaar weg hebben dat vrijwel elk programma dat op het ene OS compileert, ook op het andere compileerbaar is. Een voorbeeld hiervan is het duo Linux en SunOS/Solaris, die beide tot de klasse van Unix operating systemen behoren. Op veel plaatsen in dit dictaat zullen deze operating systemen daarom samen behandeld worden. Het belangrijkste verschil ertussen is de processor waarvoor ze ontworpen zijn. Eén en ander is samengevat in Figuur 3.



Figuur 3: Verhoudingen tussen de verschillende OS's

Wij zullen ons bij het vak Programmeermethoden voornamelijk toeleggen op het schrijven van programma's die op alle computers en op alle operating systemen te compileren zijn. Het voordeel daarvan is dat het in principe mogelijk is om de opdrachten bij het vak op alle computers te maken. Het nadeel is dat je de specifieke voordelen van een operating systeem niet kunt benutten.

1.5 Andere termen

Er zijn verschillende andere termen die je bij computers vaak zult tegenkomen. Hieronder zijn deze kort nader toegelicht.

Editor: een programma waarmee je een tekst zonder opmaak in kunt typen. Onder Windows is Notepad een editor, Word is geen editor. C++- en Python-code zal in een editor ingetypt worden.

Terminal/Console/Shell: een programma waarin je opdrachten kunt intypen die het operating systeem direct uit moet voeren. Je kunt in dit programma bijvoorbeeld opdracht geven om een tekst van de USB-stick naar de (netwerk)harddisk te kopiëren, of om een programma te starten.

Algoritme: een algoritme is een stappenplan om een bepaald probleem op te lossen. Het stappenplan hoeft nog niet in een programmeertaal uitgewerkt te zijn en kan nog Nederlandse of Engelse zinnen bevatten.

Grafisch/GUI programma: een programma dat niet alleen met de gebruiker communiceert door middel van tekst en toetsenbord.

Secundaire opslag (secondary storage): het langzame geheugen van de computer.

Account: alle onderdelen van het netwerk (delen van een harddisk, passwords, ...) die bij een bepaalde gebruiker horen.

Linken: vaak is een programma zo groot, dat het overzichtelijker is de broncode niet in één bestand te stoppen. Deze onderdelen kunnen wel apart gecompileerd worden, maar moeten uiteindelijk toch aan elkaar geplakt worden in één binary. Dit proces van aan elkaar plakken heet *linken*. Veel compilers hebben standaard-binaries die aan elk programma toegevoegd dienen te worden; ook om die in een programma te stoppen moet er gelinkt worden. Dat verklaart waarom programma's met één broncode-bestand vaak ook gelinkt moeten worden.

1.6 Een klein C++-programma

Een klein C++-programma dat gebruikt zal worden als illustratie is het volgende:

```
#include <iostream>
using namespace std;

int main ( ) {
    cout << "Dit komt op het scherm" << endl;
    return 0;
} // main
```

Voorbeeldprogramma 1: Dit komt op het scherm, C++-versie

Het bovenstaand programma doet niet veel meer dan de volgende tekst op het scherm afdrukken:

Dit komt op het scherm

In onderstaande paragrafen is terug te vinden hoe dit kleine voorbeeld-programma (en andere, moeilijkere programma's) op de verschillende operating systemen te creëren en compileren zijn.

1.7 Een klein Python-programma

Een klein Python-programma dat gebruikt zal worden als illustratie is het volgende:

```
# Dit is een regel commentaar (en wordt overgeslagen door de interpreter)

print("Dit komt op het scherm")
```

Voorbeeldprogramma 2: Dit komt op het scherm, Python-versie

Het bovenstaand programma doet niet veel meer dan de volgende tekst op het scherm afdrukken:

```
Dit komt op het scherm
```

In onderstaande paragrafen is terug te vinden hoe dit kleine voorbeeldprogramma (en andere, moeilijkere programma's) op de verschillende operating systemen kunnen worden uitgevoerd.

1.8 Windows computers

In dit hoofdstuk wordt het gebruik van Windows-computers om te programmeren kort toegelicht. We nemen daarbij aan dat de lezer al vaker met Windows gewerkt heeft. We gaan verder niet in op het onderscheid tussen de verschillende Windows varianten. Op de wiskunde-computers van het Snellius is overigens geen Windows versie aanwezig.

1.8.1 Inloggen en bladeren

Zorg er eerst voor dat de PC het Windows OS gestart heeft. Mocht de PC op Linux staan, herstart de PC dan via shutdown⇒restart, en kies voor Windows. Om in te loggen onder Windows moeten de gebruikersnaam en het Windows password (oftewel wachtwoord) worden ingetypt. Druk pas op ENTER nadat zowel de gebruikersnaam als het wachtwoord is ingevuld. Met TAB kun je van het ene invoerveld naar het andere gaan.

Met de Windows Explorer (te vinden in het startmenu) is het mogelijk rond te bladeren op het systeem. Aan de linkerkant van de explorer staan de verschillende stations (cdrom, ...) aangegeven waarover de computer de beschikking heeft. Eén van de stations heet Z:\. Dit is een onderdeel van een netwerkharddisk. Deze Z:\ is via elke Windows computer in het gebouw te benaderen. Bestanden die op locale harde schijven zijn opgeslagen, kunnen alleen achter de desbetreffende computer bekeken worden. Voor het programmeren in C++ valt het aan te raden de harde schijven te gebruiken bij het compileren van de programma's (vanwege de grote tijdelijke bestanden die gemaakt worden tijdens het compileerproces) en na het werken de benodigde bestanden (alleen de .h en de .cpp/.cc bestanden zijn echt noodzakelijk) op Z:\ op te slaan voor later gebruik.

1.8.2 Programmeren in C++

Voor het schrijven van C++-programma's onder Windows is op universitaire computers soms Microsoft Visual C++ 6.0 geïnstalleerd, of Code::Blocks.

Ga in het startmenu naar Programs⇒Microsoft Visual C++ 6.0⇒Microsoft Visual C++ 6.0 toe om Visual C++ te starten. Sluit de "Tip of the Day".

We gaan eerst de broncode intypen. Selecteer daartoe File⇒New en dan bij Files (dat is een tabblad, bovenaan): C++ Source File. Geef de filenaam op (deze maakt nu niet zoveel uit, maar

laten we kiezen voor eerste), de harde schijf waar het bestand wordt opgeslagen en klik op OK. Het nieuwe bestand wordt nu opgeslagen eindigend op `.cpp`, zodat Visual Studio het herkent als een C++ bestand. Typ in het venster dat verschijnt het voorbeeld-programma uit Hoofdstuk 1.6. Om het bestand te compileren, dient in het Build menu de optie Compile gekozen te worden. Visual C++ vraagt om een bijbehorend project te openen. Door op OK/Yes te klikken wordt dit automatisch gedaan. Vervolgens wordt de broncode gecompileerd; onderin het scherm verschijnen waarschuwingen en foutmeldingen bij dat compileren. Door op de foutmelding te dubbel-klikken kan de cursor verplaatst worden naar de plaats in de broncode waar zich een fout voordeed. Op deze manier kunnen hopelijk alle C++-fouten verbeterd worden. Vervolgens dient het programma gelinkt te worden (zelfs als het, zoals hier, slechts uit één bestand bestaat, zie Hoofdstuk 1.5). Doe dit door op Build⇒Build te klikken. Tenslotte kan het programma gestart worden door Build⇒Execute te selecteren. Nu alle compiler- en build-stappen eenmaal zijn uitgevoerd zullen in het vervolg alle voorgaande stappen indien nodig herhaald worden wanneer er op het rode uitroepteken (Execute) geklikt wordt. Klik daar nu op. Op het beeldscherm verschijnt dan (zoals verwacht) in een zwart venster:

Dit komt op het scherm

Je kunt nu beginnen met het schrijven van grotere programma's. Gefeliciteerd! Denk eraan de bestanden naar `Z:\` te kopiëren als je het voor later wilt bewaren.

Als je in een later stadium een bestaand programma weer verder wilt bewerken, en je alle bestanden bewaard hebt kun je het beste het "project" ophalen. In bovenstaande situatie zou dat project `eerste.dsw` heten, en is het op te halen via File⇒Open Workspace.

1.8.3 Thuis in C++ programmeren met Windows

Om thuis C++-programma's onder Windows te kunnen schrijven, zijn er een heleboel verschillende mogelijkheden. Hieronder staan er enkele op een rijtje.

Microsoft Visual C++: Dit programma wordt in nieuwere versies tegenwoordig ondergebracht in `.net`. Een demo-versie van (de ook goede) versie 6.0 zit soms bij het boek van Savitch.

Code::Blocks: Dit programma is kosteloos te downloaden van

www.codeblocks.org

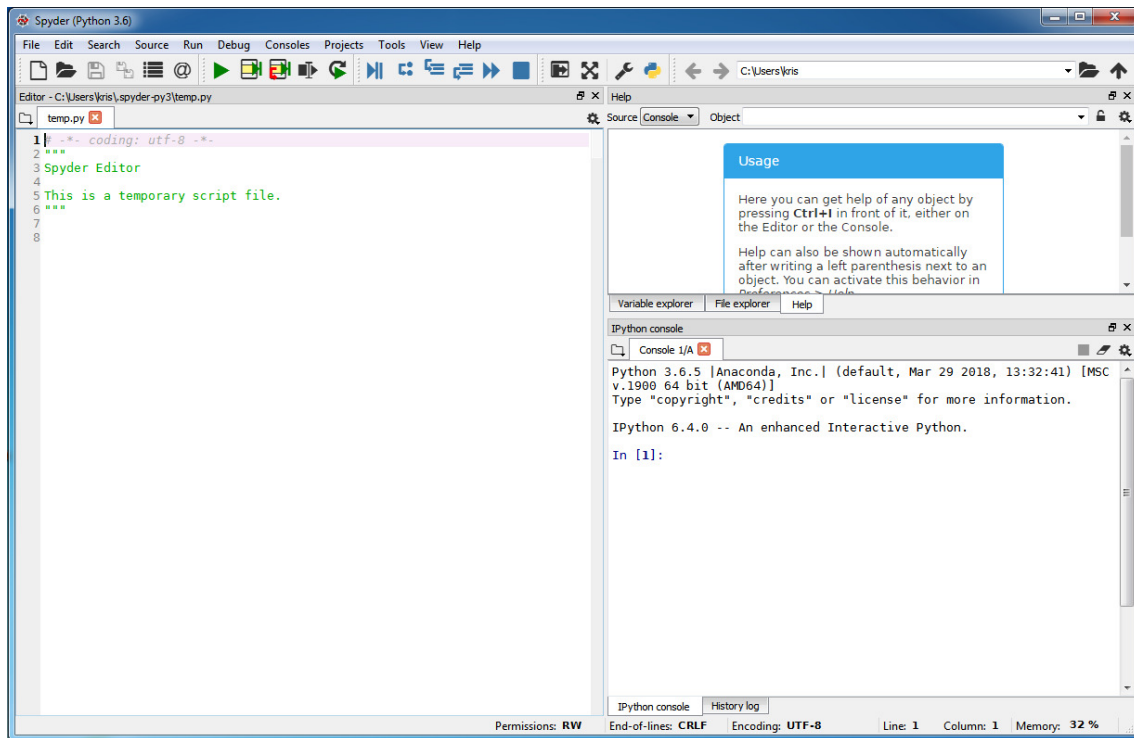
Net zoals bij Visual C++ zijn de compiler en editor geïntegreerd. Het mist echter enkele opties voor het schrijven van grafische programma's.

De werking lijkt op die van Visual C++. Kies aan het begin Bestand⇒Nieuw⇒Broncode, type weer dezelfde C++-tekst in als boven, en compileer en executeer het programma.

Tip: zet bij "Settings > Compiler" waarschuwingen aan met `-Wall`.

1.8.4 Programmeren in Python

Voor het schrijven van Python-programma's onder Windows is op universitaire computers Anaconda Python geïnstalleerd. Naast de "kale" interpreter, bevat Anaconda ook een collectie van veel gebruikte modules, Python Notebooks en een handige ontwikkelomgeving (Spyder). In de sectie over thuis in Python programmeren verderop in de tekst kun je lezen hoe je thuis Anaconda kunt installeren. We zullen nu kort omschrijven hoe je in Spyder een eerste Python-programma schrijft.



Figuur 4: De Spyder-ontwikkelomgeving

Ga in het startmenu naar All Programs⇒Anaconda3 (64-bit) ⇒Spyder toe om de Spyder ontwikkelomgeving op te starten. Sluit het venster “Spyder updates” in het geval dat verschijnt. Je krijgt een venster te zien als in Figuur 4.

We gaan eerst de broncode intypen. Er is aan de linkerkant al een nieuw bestand voor je geopend, zichtbaar als een tabblad. Laten we deze eerst opnieuw opslaan met een duidelijke naam. Gebruik hiervoor File⇒Save As. Geef de bestandsnaam op, bijvoorbeeld `eerste.py`. Kies een locatie waar je het bestand op wilt slaan. Klik vervolgens op Save. Typ nu in het tabblad het voorbeeldprogramma uit Hoofdstuk 1.7.

Om het bestand uit te voeren klik je op de groene “Play-knop” bovenaan het venster. De eerste keer zal Spyder vragen hoe het bestand moet worden uitgevoerd. De standaard keuze is “Execute in current console” en dat is prima. Klik op Run. In de console rechts onderin wordt het programma nu uitgevoerd. Je ziet hier nu als het goed is de uitvoer verschijnen.

Als er een fout in het programma zit, krijg ook weer in het venster rechts onderin te zien wat het probleem is. Tevens wordt in de editor de regel met de fout gemarkeerd met een geel driehoekje met uitroeptekens. Wanneer je dit driehoekje met de muis aanwijst, wordt ook aangegeven wat het probleem is. Je kunt dan alle fouten verbeteren en het programma opnieuw proberen uit te voeren door weer op de Play-knop te klikken.

Je kunt nu beginnen met het schrijven van grotere programma’s. Gefeliciteerd! Denk eraan de bestanden naar P:\ te kopiëren als je het voor later wilt bewaren (indien je werkt op je eigen computer, kies dan zelf een geschikte locatie).

Als je Spyder later opnieuw opstart, worden standaard de oude tabbladen weer heropend. Als dit niet het geval is kun je altijd via File⇒Open het opgeslagen bestand weer openen.

Je kunt er later ook nog voor kiezen om het programma niet in de standaard console uit te voeren, maar in een aparte console. Hiervoor kies je voor Run⇒Configuration per file. Kies hier een andere optie, bijvoorbeeld “Execute in a dedicated console”. Klik op OK en daarna op de Play-knop. Het programma wordt nu in een apart tabblad rechts onderin uitgevoerd.

1.8.5 Thuis in Python programmeren met Windows

Op Windows-systemen wordt standaard geen Python-interpreter geïnstalleerd. Het is daarom noodzakelijk om dit zelf te doen. Gelukkig bestaan er verschillende Python “distributies” die naast de Python-interpreter zelf ook handige uitbreidingen bevatten, zoals standaard modules om mooie plots te maken en ontwikkelomgevingen.

Binnen dit college en de opleiding Natuurkunde wordt er gebruik gemaakt van **Anaconda Python**. Dit pakket is kosteloos te downloaden, let er op dat je kiest voor Anaconda **Python 3.6 version**:

<https://www.anaconda.com/download>

Bij deze download is de ontwikkelomgeving Spyder inbegrepen. Zie de sectie hierboven voor een korte uitleg hoe je deze ontwikkelomgeving moet gebruiken.

1.9 Unix computers

In het Snellius is op de meeste PC's Linux aanwezig. Vrijwel alle PC's kunnen zowel onder Windows als onder Linux draaien.

1.9.1 Beginnen op de Linux PC's

Zorg er eerst voor dat de PC het Linux OS gestart heeft. Als de PC op Windows staat, geeft deze PC meestal de boodschap “Press CTRL+ALT+DELETE to log on”. Om naar Linux te gaan, moet eerst het drietal toetsen CTRL+ALT+DELETE tegelijk ingedrukt worden en daarna Shutdown⇒Shutdown and Restart gekozen worden. De computer start dan opnieuw op en Linux kan gekozen worden.

Als Linux opgestart is, is het eerst noodzakelijk zich aan het systeem bekend te maken. Doe dit door de username en het password in te typen die het systeembeheer gegeven heeft: het ULCN-account. Ga zonodig met TAB of de muis van het ene invoerveld naar het andere. Druk daarna op Enter — en wacht (de eerste keer soms lang).

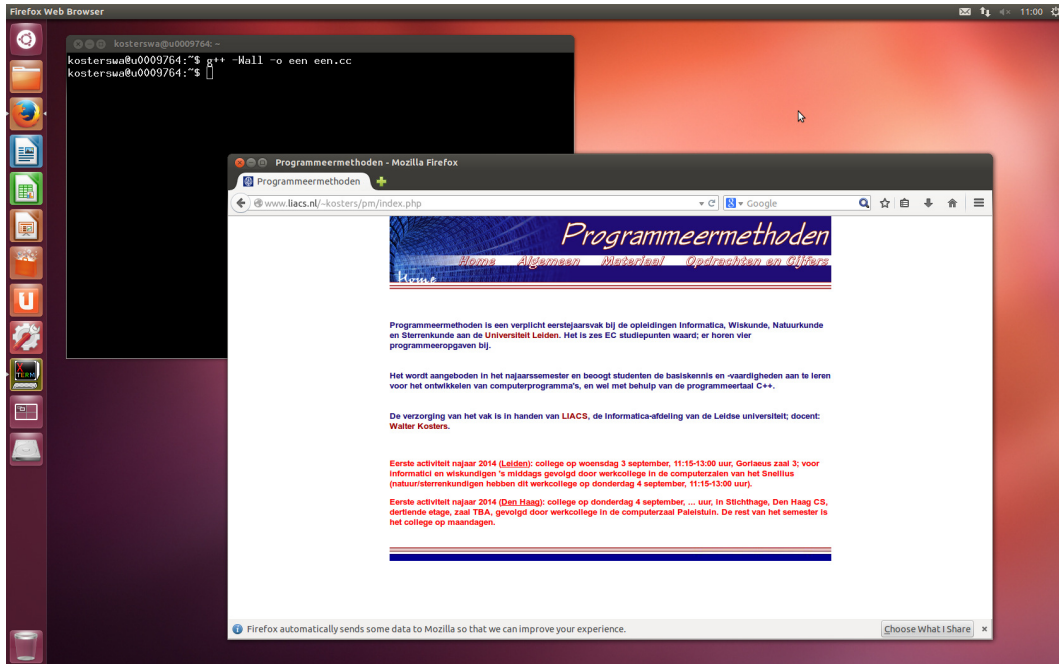
Het beeldscherm dat dan verschijnt, kan er op een heleboel verschillende manieren uitzien (dat is onder andere afhankelijk van het gekozen sessie-type en van de versie). Eén mogelijkheid is dat het scherm er ruwweg uitziet als Figuur 5: Ubuntu!

De getoonde icoontjes kunnen er anders uitzien of op een andere plaats staan; je kunt zelf van alles wijzigen. De programma's zijn te starten door op het bijbehorende icoontje te klikken. Probeer eerst een Internet browser te gebruiken. De werking van de programma's is ongeveer gelijk aan die van bekende Windows programma's. Vraag om hulp indien zich problemen voordoen.

Het belangrijkste is te ontdekken hoe de terminal (shell, console) opgestart wordt. Meestal is daar wel een aparte icoon voor, met een schelp of een beeldscherm. Of via de menu's. In de rest van het dictaat zullen we de terminal vaak gebruiken om aan te geven hoe je een programma start.

1.9.2 De Unix terminal, programma's starten en bestandsbeheer

In deze paragraaf zullen we kort het gebruik toelichten van de “Unix terminal”. In Hoofdstuk 2 is een uitgebreid overzicht te vinden van diverse commando's die in de Unix terminal gebruikt kunnen worden. De Unix terminal wordt overigens ook wel eens “xterm” genoemd. In feite is het gewoon een window waarin je Unix-opdrachten kunt geven.



Figuur 5: Linux: Ubuntu

Bestandsbeheer

Op de Unix systemen start de terminal altijd in jouw eigen *homedir*. Dit is het beginpunt van je bestandssysteem. Om de *homedir* te bekijken kan het commando

```
ls
```

ingetypt worden. Of, voor iets meer bestandsinformatie:

```
ls -lrt
```

(direct achter de min staat de kleine letter l).

Net als bij de Windows systemen heeft ook een Unix systeem een directory structuur om bestanden te organiseren. Om van directory te wisselen gebruik je

```
cd directory
```

waarbij *directory* als volgt voorgesteld kan worden:

<code>..</code>	Ga een directory “hoger”
<code>~</code>	Ga naar je <i>homedir</i> (“tilde”)
<code>naam</code>	Ga naar directory <i>naam</i>

Overigens staan de slashes in bestands- en directorynamen net andersom dan in Windows: / in plaats van \.

Om een nieuwe directory te maken gebruik je `mkdir`:

```
mkdir naam
```

Als je één van de bestanden wil verwijderen dan kan dat met `rm`:

```
rm naam
```

Als je één van de bestanden wil verplaatsen dan kan `mv` gebruikt worden:

```
mv oudenaam nieuwenaaam
```

Als je opnieuw `ls` uitvoert, heeft het oude bestand nu een nieuwe naam en is identiek aan het origineel van daarnet. Als je bij *nieuwe naam* ook een nieuwe directory invoert zal het bestand ook van directory veranderen.

Als je van één van deze bestanden een kopie wilt maken, kan `cp` gebruikt worden:

```
cp oudenaam nieuwenaaam
```

Als je opnieuw `ls` uitvoert, staat er nu een nieuw bestand met een nieuwe naam dat identiek is aan het origineel van daarnet.

Om een bestand naar een USB-stick te kopiëren (bij PC's) kan

```
cp denaam /media/???
```

gebruikt worden. Er wordt dan een kopie met dezelfde bestandsnaam op de USB-stick ??? gezet. Geef wel daarna het commando `sync` om te “synchroniseren”. Gebruik eventueel “TAB-completion” om de naam van de USB-stick aan de weet te komen.

Grafische programma's starten vanaf de terminal

Bij het starten van grafische programma's — tekstverwerking, webbrowser — vanaf de terminal is het noodzakelijk om de opdracht die het programma start te laten volgen door een `&`. Op die manier kan de terminal gebruikt blijven worden voor het uitvoeren van andere opdrachten.

Surfen op Internet

Om de Firefox webbrowser te starten kan (behalve door te klikken op het desbetreffende icoontje) ook het volgende commando worden ingetypt:

```
firefox &
```

Of zoek Chrome op ...

Een simpele tekst editor

Om een tekstbestand aan te maken kan de editor `gedit` gebruikt worden — er zijn trouwens nog vele andere editors. `gedit` wordt gestart door het volgende commando in te typen:

```
gedit &
```

Om een bestand met betekenisvolle kleurtjes weer te geven, kan in het Preferences menu de optie `Highlight Syntax` aangezet worden (als de file-extensie goed is, gaat dit vanzelf). Om elk bestand altijd met kleurtjes weer te geven, moet de juiste optie in het Preferences⇒Default Settings menu aangezet worden en vervolgens `Save Defaults` gekozen worden. Om voor elke regel een regelnummer af te beelden, kan in het Preferences menu de optie `Show Line Numbers` aangezet worden. Er zijn nog hier meer handige instellingen aan te zetten.

Typ bij wijze van proef een kleine tekst in het venster, en schrijf deze weg door naar `File⇒Save As...` te gaan. Typ achter `/home/iemand/` (iemand is jouw gebruikersnaam) een bestandsnaam zonder spaties in en klik op `OK`. Het bestand wordt hiermee ergens op jouw account opgeslagen (in je homedir).

1.9.3 Programmeren in C++

Open `gedit` en voer het programma van Hoofdstuk 1.6 in. Bewaar deze code onder de naam `eerste.cc` (het is onder Unix gebruikelijk C++-bestanden een naam te geven die op `.cc` eindigt). Typ daarna het volgende commando in:

```
g++ -Wall -o eerste eerste.cc
```

De compiler (g++) wordt hiermee gestart. Achteraan het rijtje staat het bestand dat je wil gebruiken als bron voor het nieuw te maken programma. Na de optie `-o` staat de naam die het nieuwe programma, de “executable”, moet krijgen (dit hoeft niet hetzelfde te zijn als (het beginstuk van) de naam van het bronbestand!). Als deze optie niet gebruikt wordt, krijgt de executable de naam `a.out`. Met `(-Wall)` geven we aan dat we alle waarschuwingen over C++-fouten te zien willen krijgen. Als het voorbeeldbestand goed gemaakt en weggeschreven is, zegt de compiler niets en kun je vervolgens het nieuwe programma starten. Typ als commando:

```
./eerste
```

Op het beeldscherm verschijnt dan (zoals verwacht):

```
Dit komt op het scherm
```

Dit betekent dat de C++-code goed was, gecompileerd kon worden en dat de processor het simpele programma uit kon voeren. Gefeliciteerd!

1.9.4 Programmeren in Python

Open gedit en voer het programma van Hoofdstuk 1.7 in. Bewaar deze code onder de naam `eerste.py`. Het is gebruikelijk dat Python-bestanden altijd een naam krijgen die op `.py` eindigt. Typ daarna het volgende commando in:

```
python3 eerste.py
```

De Python-interpret wordt hiermee gestart en deze zal het programma direct uitvoeren. Op het beeldscherm verschijnt dan (zoals verwacht):

```
Dit komt op het scherm
```

Dit betekent dat de Python-code goed was en dat het interpreter het simpele programma uit kon voeren. Gefeliciteerd!

Als je dat prettiger vindt kun je ook gebruik maken van de ontwikkelomgeving Spyder. Deze is op te starten door het volgende commando te geven in een terminal:

```
spyder3 &
```

Links kun je de editor vinden met een tabblad per bestand. Een programma uitvoeren gaat door te drukken op de groene “Play-knop”. De uitvoer van het programma verschijnt dan rechtsonder. Zie ook Programmeren in Python in het hoofdstuk Windows computers voor wat meer informatie over Spyder.

1.9.5 Thuis programmeren met Linux *

Het is goedkoop en “eenvoudig” om Linux op je PC thuis te gebruiken. Wellicht de eenvoudigste methode is om gebruik te maken van “(K)Ubuntu”. Dit zijn versies van Linux die je vanaf CD of DVD of USB kunt starten (een “live CD”); je hoeft het niet meteen op je computer te installeren. Je kunt de CD eenvoudig van het web halen: <http://www.ubuntu.org/>. Denk er aan een “image” op te halen. Stop de CD in je computer, en zorg ervoor dat deze vanaf CD kan opstarten. Je komt dan vanzelf in het window-systeem terecht. Je kunt een en ander ook vanaf het internet direct laten doen.

Ietsje ingrijpender is het om Linux thuis op je PC te installeren. Je zult dan namelijk de harddisk in je computer moeten *partitioneren*: de harddisk moet zodanig worden ingedeeld dat een vast deel ervan door Linux gebruikt kan worden. Vanuit Windows kun je dat deel van je harddisk echter niet meer gebruiken. Het is daarom aan te raden eerst vertrouwd met Unix te raken. Raadpleeg <http://www.linux.org/>, of een van de talloze andere websites, of vriendjes.

Er zijn verschillende fabrikanten of groepen mensen die een versie van Linux in elkaar zetten. Eén zo'n versie wordt een distributie genoemd, en elke distributie is afgestemd op een bepaalde gebruikersgroep.

Een degelijke, maar ook zeer technische, distributie is Debian GNU/Linux: goedkoop, in feite zelfs gratis, en eenvoudig te verkrijgen. Minder technisch (maar daarom ook minder geliefd bij sommige computeraars) zijn SUSE en RedHat/Fedora. Ook deze zijn min of meer gratis van het Internet te halen. Voor extra informatie over Linux zijn de boeken van uitgeverij O'Reilly, zoals "Linux in a nutshell" (ook in het Nederlands), meestal hun geld meer dan waard. Bij de meeste distributies wordt software meegeleverd om de harddisk te partitioneren.

2 Werken met Unix

2.1 Unix commando's *

Hier wordt meer uitgebreid uitgelegd hoe Unix (terminal)commando's er uitzien en wat ze doen. Er wordt slechts een heel klein, maar wel nuttig gedeelte van dit gigantische systeem behandeld.

2.1.1 Syntax van een commando

Een commando bestaat uit een of meer woorden, gescheiden door spaties. Een woord is een string zonder spaties. Het eerste woord van een commando is de naam, de rest zijn parameters en argumenten. Bijvoorbeeld:

```
lpr -Peenprinternaam rommel.txt
```

Hier is `lpr` de naam van het commando (output naar een printer), `-P` is een parameter (om de naam van de gewenste printer aan te geven), “`eenprinternaam`” is het argument van “`P`”, terwijl “`rommel.txt`” het argument van het commando is om de naam van de te printen file mee aan te geven. Parameters zoals `P` die voorafgegaan worden door een “`-`” heten “switches” en bieden de mogelijkheid te kiezen tussen de verschillende opties van het commando. In dit speciale geval: als de switch afwezig is, wordt de file geprint op de “default” printer — en die is in principe voor iedereen anders. Als dat toevallig een matrix-printer is, en de file is een PostScript-file, haal je je overigens een hoop narigheid op de hals.

In het vervolg worden parameters of argumenten die niet verplicht zijn tussen vierkante haakjes gezet, met “[...]”; als er een waarde voor het een of ander gespecificeerd moet worden, wordt dit genoteerd met “<het-een-of-ander>” (de < en > niet intikken!).

2.1.2 Procesbeheer

Met Unix kun je allerlei processen (programma's) tegelijkertijd draaien. Zo draait er altijd al een fors aantal processen zonder dat je daar erg in hebt, dit om de computer zijn werk te laten doen: een klokje op het scherm te zetten, een CD te lezen, naar email te kijken, enzovoorts. Soms geeft een commando een eigen window waarin het programma zich verder ontwikkelt, soms niet. De volgende commando's zijn in staat op allerlei manieren processen te manipuleren:

<code>CTRL-c</code>	stopt een lopend proces
<code>CTRL-z</code>	interrumpeert een lopend proces, dat later weer verder kan gaan; het proces blijft bestaan
<code>ps eaf</code>	geeft overzicht van lopende processen; <code>ps -u Clinton</code> doet dit voor gebruiker Clinton; op sommige systemen <code>ps aux</code> of <code>ps -eaf</code>
<code>kill <PID></code>	gooit proces met process-id PID uit de lijst met processen
<code>kill -HUP <PID></code>	gooit iets netter proces met process-id PID uit de lijst met processen (via een “hangup-signaal”)
<code>exit</code> of <code>logout</code>	verlaat een shell of command window; ook: <code>CTRL-d</code>
<code>passwd</code>	verander je eigen password

En wat meer gevorderde commando's:

<code>bg</code>	stuurt een proces gestopt met CTRL-z naar de achtergrond, dat wil zeggen: ga er mee door zonder op stoppen te wachten
<code>fg</code>	haalt het huidig lopende proces van achtergrond naar voorgrond
<code>jobs</code>	laat de huidige achtergrond-processen zien
<code>top</code>	laat de processen actief op de machine zien, ook processen van andere gebruikers
<code><command> &</code>	start een commando op de achtergrond, dat wil zeggen zonder op stoppen te wachten
<code>CTRL-s</code>	stopt output naar het scherm
<code>CTRL-q</code>	gaat verder met uitvoer naar het scherm

Een voorbeeld. Stel je wilt een (misschien nieuwe) file `nieuw.cc` editen met het programma `gedit`. Je tikt in `gedit nieuw.cc`, gevolgd door een druk op de Enter-toets. Je kunt dan in een nieuw window aan het editen slaan, maar stel dat je in het oorspronkelijk window weer wat wilt doen. Dat kan nu helaas niet, want de controle is helemaal overgegeven aan het `gedit`-proces. Dat kun je tijdelijk staken door CTRL-z in het oorspronkelijke window te geven (het proces wordt dan *gesuspend*; nu kun je voorlopig niet meer editen), en als je verder wilt met editen: `bg`. Het was eenvoudiger geweest om meteen aan het begin `gedit nieuw.cc &` te geven, waardoor je tevens in het oorspronkelijke window gewoon door kunt werken. Overigens zijn er ook programma's die zelf die `&` er als het ware bijdoen: `gvim nieuw.cc` bewerkt, met de editor `gvim`, de file `nieuw.cc` — een `&` is overbodig.

2.1.3 Files

Unix heeft een boomstructuur met als bladeren de *files*, en als (interne) knopen de *directories* — ook wel mappen genoemd. Elke directory kan andere directories en files bevatten. Aan de wortel staat altijd `/`. De directory waarin je meteen na het inloggen terecht komt, heet je “home-directory”, doorgaans met een `~` aangegeven (dit is trouwens hetzelfde als `/home/iemand` als je `iemand` bent). Alle files en directories die je zelf maakt horen hieronder te zitten. Elke file wordt gekarakteriseerd door:

1. het pad er naar toe (bijvoorbeeld `/home/clinton/geheim/`)
2. de filenaam (bijvoorbeeld `dagboek.txt`)
3. de eigenschappen (properties): eigenaar, grootte, toegangsrechten, soort en dergelijke

Om naar een file te verwijzen moet je of de volledige naam geven (waarbij zeer lange namen, eventueel met punten erin, kunnen voorkomen — en let er ook op dat kleine letters en hoofdletters verschil maken!): `/home/clinton/geheim/dagboek.txt` of — doorgaans eenvoudiger — de plaats met betrekking tot de huidige “working directory” (cwd) (geen `/` in het begin!): `geheim/dagboek.txt` als je op dit moment in directory `/home/clinton` zit.

Afkortingen voor directories en files:

.	huidige working directory (cwd)
..	ouder-directory van cwd
~	je eigen home-directory (een “tilde”)
*	staat voor elke serie karakters in een filenaam (bijvoorbeeld <code>rm *.c</code> gooit alle files die eindigen op <code>.c</code> weg uit de cwd)
?	staat voor precies één karakters in een filenaam (bijvoorbeeld <code>lpr data?</code> print alle files waarvan de naam begint met <code>data</code> , en waar nog één karakter achter komt)

En commando's om files en directories te manipuleren:

<code>cat <file(s)></code>	output file(s) op beeldscherm (zie ook “re-direction” van output)
<code>less <file(s)></code>	eenvoudige output van file(s), pagina voor pagina
<code>lpr <file(s)></code>	output naar een printer
<code>rm <file(s)></code>	verwijder file(s) — oppassen!
<code>cp <file1> <file2></code>	kopieer file1 naar file2
<code>ls [-al] [<directory>]</code>	laat informatie zien over de inhoud van een directory; <code>-a</code> laat ook “dot files”, zoals <code>.login</code> ; <code>-l</code> laat meer informatie zie (eigenaar, datum, grootte, ...)
<code>pwd</code>	laat huidige working directory (cwd) zien, doorgaans ook in de prompt genoemd
<code>cd [<directory>]</code>	verander cwd in directory (cd zonder directory maakt home-directory de cwd)
<code>mkdir <directory></code>	maakt binnen cwd een nieuwe directory aan geheten directory
<code>mv <oudenaam> <nieuwenaam></code>	als nieuwenaam een bestaande directory is, wordt oudenaam naar deze directory verplaatst; hernoem anders oudenaam in nieuwenaam
<code>rmdir <directory/ies></code>	verwijder (lege) directory/ies
<code>chmod [aug] [+ -] [rwx] <file(s)></code>	wijzig de rechten van file(s), waarbij de eerste parameter vertelt om wie het gaat (a=all, u=user, g=group, o=others), de tweede parameter of toegang wordt verstrekt (+) of afgenomen (-), en de derde welke toegangsvorm wordt gewijzigd (r=read, w=write, x=execute). Oude stijl: <code>chmod 644 *</code> of <code>chmod 750 *</code>
<code>wc <file(s)></code>	telt aantal regels, woorden en bytes van file(s)
<code>grep [<patroon>] [<file(s)>]</code>	doorzoekt file(s) op patroon, dat een (wellicht kryptisch) reguliere expressie zoals <code>'[kK][a-zA-Z]*[sS]'</code> kan zijn

Een voorbeeld: gebruik `rm core` om zogeheten core-dumps weg te gooien (grote files in nood-

gevallen door het systeem gemaakt); gebruik `cd ..` (en niet `cd .`) om naar de ouder-directory van de `cwd` te gaan.

2.1.4 Input/Output

Veel Unix-commando's maken gebruik van de drie standaard *devices*, te weten: standard-input, standard-output en standard-error-output, die normaal gekoppeld zijn aan toetsenbord (de eerste) en beeldscherm (de tweede en de derde). Ze kunnen ook eenvoudig worden ge-“redirect” naar files (de middelste `>`, `>>` en `<` wel intikken):

<code><commando> > <file></code>	redirect output van commando naar file <code>file</code>
<code><commando> >> <file></code>	append (hang) output van commando achter file <code>file</code>
<code><commando> < <file></code>	haal input voor commando uit file <code>file</code>

2.1.5 Pipelining

Een zeer nuttige eigenschap van Unix is de mogelijkheid tot “pipelining”. Dat is de uitvoering, achter elkaar, van verscheidene commando's, waarbij de output van ieder commando dient als invoer voor het volgende. Het pipelining symbool is `|`.

Enkele voorbeelden:

```
ps -uObama | grep 15
```

retourneert (onder meer) informatie over alle processen van Obama, gestart tussen 3:00 pm en 3:59 pm en de processen met 15 in hun process-id. Het `grep`-commando grijpt namelijk alle regels waar 15 in voorkomt uit de lijst met processen van Obama.

```
find . -name '*.cc' | xargs grep -i "Bush"
```

zoekt in de huidige directory en daaronder naar files met extensie `.cc`, selecteert daaruit die files die de string `Bush` bevatten, niet lettend op het verschil tussen kleine letters en hoofdletters, en zet de betreffende regels met filenaam erbij op het scherm.

2.1.6 Geschiedenis

De shell heeft een mechanisme dat gegeven commando's bijhoudt. Als je `history` als commando geeft (h werkt vaak ook) krijg je een genummerde lijst met de laatste commando's. Met behulp van `!` kun je oude commando's of gedeeltes daarvan gebruiken in je volgende commando. Als voorbeeld: `!c` wordt het laatste commando dat begon met een `c`. Gebruik de pijltjes-toetsen om vorige commando's te krijgen en de `TAB`-toets om je commando's af te maken. Bijvoorbeeld, als je de `TAB`-toets gebruikt na `rm fil` — voor de `Enter`-toets te gebruiken — krijg je een lijst van alle filnamen die beginnen met `fil` (als zulke files bestaan); dit werkt ook met begingedeeltes van commando's. Doorgaans worden beginstukken aangevuld tot het systeem verschillende vervolgmogelijkheden ziet.

2.2 De editor vi in een notepad *

Het programma `vi`, de “visual editor”, is een zeer krachtige editor die op elk Unix-systeem aanwezig is (soms overigens ook `vim` geheten). Het is een alternatief voor het simpelere `gedit`. Voor liefhebbers van grafische user interfaces is er ook een grafische variant, `gvim`, beschikbaar. Sommige mensen vinden het lastig om met `vi` te werken, maar eigenlijk is het heel simpel. Het enige waar je op moet letten is dat er een *command-mode* is en een *input-mode*. In de

command-mode (de naam zegt het al) wordt alles wat je intikt opgevat als een commando. In de input-mode wordt alles opgevat als tekst.

Als je vi aanroept met `vi <file-naam>` kom je in de command-mode. Met de volgende commando's kom je in de input-mode:

i	voeg tekst toe voor het huidige character
A	voeg tekst toe achter de huidige regel

Met de ESC-toets verlaat je weer de input-mode en kom je terug in de command-mode.

Er zijn een aantal manieren om vi te verlaten. De belangrijkste zijn:

ZZ	verlaat vi en save de file
:q!	verlaat vi zonder de file te saven (noodsprong)

De cursor kan doorgaans worden verplaatst met behulp van de pijltjes op het toetsenbord, of — altijd — met de letters h, j, k en l in command-mode. Tekst verwijderen gebeurt in de command-mode met de volgende commando's:

x	delete het huidige character
dd	delete de huidige regel

Soms werkt ook de backspace-toets.

Het kopiëren van stukken tekst kan op de volgende manier:

- ga met de cursor naar de beginregel van de te kopiëren tekst;
- geef het aantal regels dat je wilt kopiëren;
- “yank” die regels met behulp van het commando Y;
- ga vervolgens naar de regel waar je de tekst wilt toevoegen;
- geef het commando p (put).

Het vlot doorlopen van je tekst doe je — behalve met Page Up en Down — met het volgende commando:

CTRL-f	ga een pagina omlaag
CTRL-b	ga een pagina omhoog
:27	ga naar regelnummer 27

Uiteraard heeft vi nog veel meer commando's, bijvoorbeeld voor het zoeken naar strings:

/McCain	zoek naar string McCain
---------	-------------------------

Nog een voorbeeld: een punt in command-mode herhaalt het laatste commando. Als je in command-mode een : intikt zit je overigens in *ex-mode*. Je kunt meer informatie vinden in de systeem-documentatie.

Als je gvim gebruikt, is het handig om een file `.gvimrc` (let op de punt aan het begin) in je eigen home-directory te zetten met daarin je eigen standaardinstellingen.

2.3 Slotopmerkingen *

Dit overzicht is bij lange na niet compleet. Maar er staat voldoende in om voorlopig aardig uit de voeten te kunnen.

Het is een goed idee om je eigen file `.bashrc` (als je tenminste bash als shell gebruikt, anders misschien `.tcshrc`; let op de punt aan het begin van de filenaam: het betreft een zogeheten “dot-file”) in je home-directory eens te bekijken. Deze file bevat persoonlijke informatie en wordt bij het opstarten van het systeem uitgevoerd. De file kan door de gebruiker zelf aangepast worden, bijvoorbeeld door eigen “aliases” (afkortingen) voor veelgebruikte commando’s toe te voegen, of het zoekpad te wijzigen (zo zit vaak de huidige directory niet in dit pad, vandaar dat je `./eerste` moet zeggen om de executable eerste uit `.`, de huidige directory, uit te voeren). Veel Unix-programma’s hebben zogeheten “resource-configuration” (rc) files voor persoonlijke instellingen.

Meer informatie over Unix-commando’s kan worden gevonden via de online handleidingen. Type man `<command>` om dit te bekijken, bijvoorbeeld man `ls`. Als je de naam van een commando niet (meer) weet kun je proberen man `-k <woord>` in te tikken: er verschijnt dan een lijst met commando’s die iets met “woord” te maken hebben, bijvoorbeeld: man `-k process` of man `-k debug`.

Niet alle commando’s zijn gedocumenteerd met man; het GNU project (de makers van `gcc/g++`) gebruikt bijvoorbeeld `TeXinfo`. Probeer ook maar eens man `ddd` om informatie te krijgen over de fraaie Data Display Debugger DDD, die op diverse systemen beschikbaar is.

Een ander probleem is het volgende. Stel dat je wilt weten hoeveel tijd je C++-programma doorbrengt in zijn verschillende functies. Je bent dan op zoek naar een “profiler”. Op sommige systemen bestaat `gprof`. Compileer eerst je programma, zeg eerste.`cc`, met

```
g++ -o eerste eerste.cc -pg
```

run daarna `./eerste`, en laat dan `gprof` zijn werk doen:

```
gprof eerste gmon.out > eerste.prof
```

De file `eerste.prof` bevat nu interessante informatie.

In de Unix-wereld worden regelovergangen met één symbool, de LF oftewel LineFeed (karakter 10) verzorgd. In de Windows-wereld zijn dat er twee: een CR oftewel CarriageReturn (karakter 13), en een LF. Sommige editors hebben hier problemen mee. Onder Linux kun je het ene formaat in het andere omzetten met `dos2unix` en `unix2dos`.

En hoe kun je files “zippen”, en meerdere files tegelijk per email versturen? Gebruik

```
tar cvfz naam.tgz *cc plus.txt
```

om een gezippt “archief” naam.tgz te maken dat alle files waarvan de naam eindigt op `cc en` de file `plus.txt` uit de huidige directory. Het archief kan worden uitgepakt (“extracten”) met `tar xvfz naam.tgz`.

En waar komen al deze programma’s vandaan? Probeer `which`, bijvoorbeeld `which g++`. Of waar is `time.h`? Gebruik `locate`, bijvoorbeeld `locate time.h`.

Het is heel eenvoudig om vanuit huis op het Unix-systeem in te loggen. Gebruik daar het kleine Windows-programma `putty` voor, gratis op te halen bij

<http://www.chiark.greenend.org.uk/~sgtatham/putty/>

Je moet dan “ssh-en” (ssh staat voor “secure shell”) naar `sshgw.leidenuniv.nl`. (Als je daar echt wilt werken moet je weer doorloggen naar een machine als `huisuil03`.) En als je files heen en weer wilt transporteren, gebruik dan het aldaar ook te verkrijgen ftp-programma `psftp`. Overigens, op een Linux-systeem met internet kun je (user `clinton` zijnde) gewoon intikken

```
ssh clinton@sshgw.leidenuniv.nl
```

en files heen en weer transporteren met scp. Zo kopieert

```
scp clinton@sshgw.leidenuniv.nl:dagboek.txt .
```

de file dagboek.txt uit de home directory van de gebruiker clinton op server sshgw.leidenuniv.nl naar de huidige directory op de machine waar je dit commando intikt.

Het World Wide Web (WWW) bevat allerlei online informatie. Zoeken met de bekende zoekmachine www.google.com levert al heel wat op, of speur naar files met Frequently Asked Questions, de zogeheten FAQs. Een goede Unix-introductie is beschikbaar als

<http://www.liacs.leidenuniv.nl/~kosterstwa/4ltrwrdr-1.pdf>

En misschien ben je geïnteresseerd in het produceren van prachtig vormgegeven output: op zoek naar $\text{T}_{\text{E}}\text{X}$ of $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$? Probeer

<http://www.liacs.leidenuniv.nl/~kosterstwa/stuva1/lshort.pdf>

Of wil je meer weten van reguliere expressies en Perl ... zoek dan zelf bij

<http://www.google.com/>