Dit document bevat algemene informatie over computers, en meer in het bijzonder over het operating systeem UNIX/Linux. Het document wordt gebruikt bij het eerstejaars college *Programmeermethoden*, Universiteit Leiden, najaar 2024, zie

`www.liacs.leidenuniv.nl/~kosterswa/pm/`

Met dank aan allen die aan deze tekst hebben bijgedragen.
Walter A. Kosters, Leiden, 1 september 2024.

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 C++ compiler

In essentie bestaat een stappenplan dat de computer uit kan voeren (een *programma*) 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 een file) omzet in binaire code (dat ook weer wordt opgeslagen in een file). 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 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 (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.

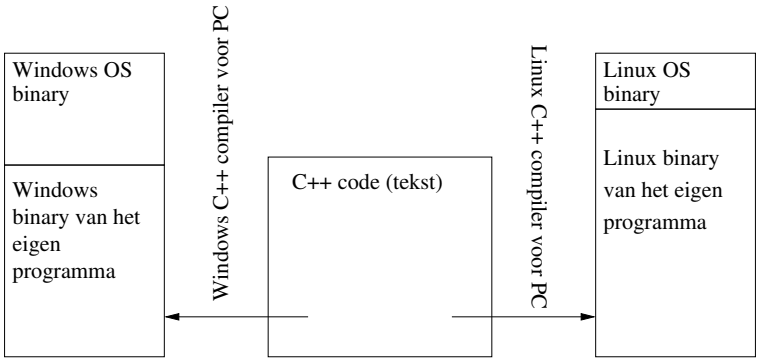
1.2 Operating Systemen

Door de meeste programma's moet een grote hoeveeheid 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 1).



Figuur 1: 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 nagenoeg 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, bijvoorbeeld Unix en de vele Linux-varianten. Het belangrijkste verschil ertussen is eigenlijk de processor waarvoor ze ontworpen zijn.

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, en dat er soms subtiele verschillen zijn bij het draaien op die verschillende systemen.

1.3 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 of Sublime Text een editor, Word is geen editor. C++-code zal in een editor ingetypt worden.

Terminal/Console/Shell: een programma (window) waarin je opdrachten kunt intypen die het operating systeem direct uit moet voeren. Je kunt in dit programma bijvoorbeeld opdracht geven om een file 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.

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 in meerdere bestanden 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. Meestal gebeurt het linken vanzelf of nagenoeg onzichtbaar.

1.4 Een klein programma

Een klein C++-programma ("Hello World") dat gebruikt zal worden als illustratie is het volgende:
Het bovenstaand programma doet niet veel meer dan de volgende tekst op het scherm afdrukken:

```
Dit komt op het scherm
```

Bij het eerste werkcollege, zie

`http://www.liacs.leidenuniv.nl/~kosterswa/pm/pmwc1.php`

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

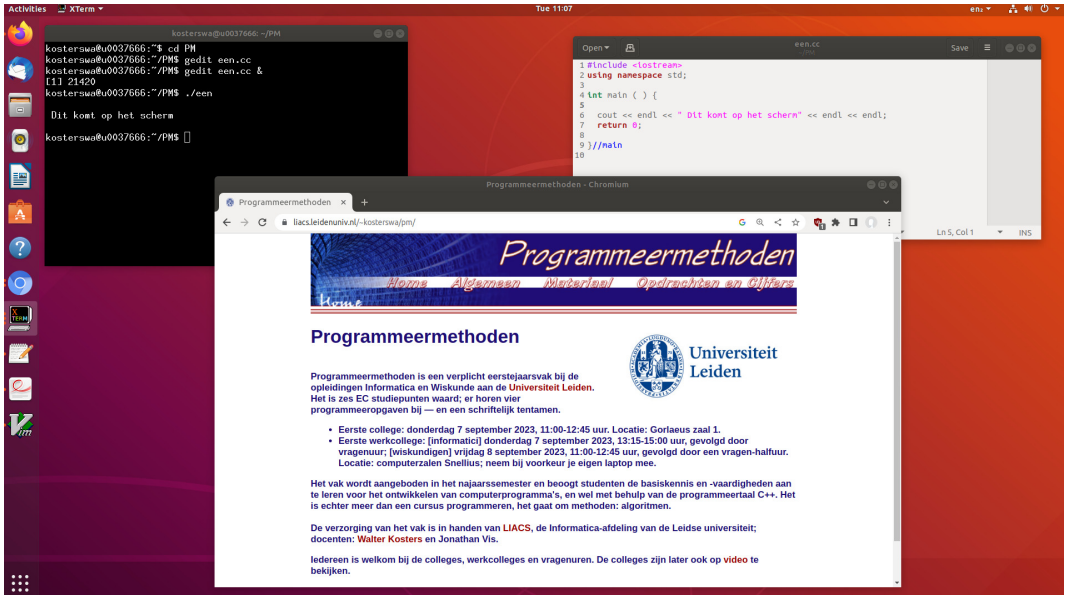
```
#include <iostream>
using namespace std;

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

Voorbeeldprogramma 1: Dit komt op het scherm

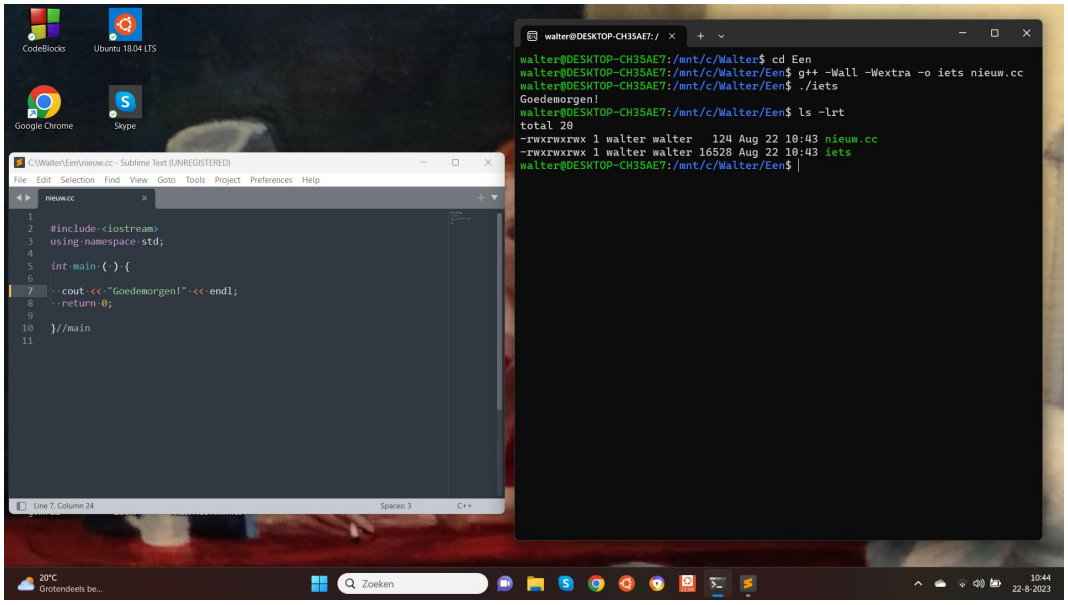
1.5 Unix computers

In het volgende geven we een zeer korte introductie tot Unix/Linux. Het beeldscherm kan er op een heleboel verschillende manieren uitzien. Eén mogelijkheid is dat het er ruwweg uitziet als Figuur 2: Ubuntu, of Figuur 3: Windows Subsystem for Linux.



Figuur 2: Linux: Ubuntu, met terminal, `gedit` en Chrome browser.

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 belangrijkst is te ontdekken hoe de *terminal* (shell, konsole) 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.



Figuur 3: Windows Subsystem for Linux, met `Sublime Text` en terminal.

1.5.1 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.

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 (de ℓ)).
Net als bij de Windows systemen heeft ook een Unix systeem een directory-structuur om bestanden te organiseren. Om van directory = map te wisselen gebruik je

`cd` *directory*

waarbij directory als volgt voorgesteld kan worden:

`..`	Ga een directory "hoger"
`~`	Ga naar je homedir ("tilde")
naam	Ga naar directory *naam*

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 nieuwenaam*

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 nieuwenaam*

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, zoals `Sublime Text` en `gvim`. 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).

1.5.2 Programmeren

Open `gedit` en voer het programma van Hoofdstuk 1.4 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 -Wextra -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!

Zie voor meer details de website van Programmeermethoden.

2 Werken met Unix

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 syteem behandeld.

2.1 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!).

2.2 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:

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

En wat meer gevorderde commando's:

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

2.3 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/biden/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/biden/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/biden` 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 `rm *.c` gooit alle files die eindigen op `.c` weg uit de cwd)
`?`	staat voor precies één karakters in een filenaam (bijvoorbeeld `lpr data?` print alle files waarvan de naam begint met `data`, en waar nog één karakter achter komt)

En commando's om files en directories te manipuleren:

`cat <file(s)>`	output `file(s)` op beeldscherm (zie ook "redirection" van output)
`more <file(s)>`	eenvoudige output van `file(s)`
`lpr <file(s)>`	output naar een printer
`rm <file(s)>`	verwijder `file(s)` — oppassen!
`cp <file1> <file2>`	kopieer `file1` naar `file2`
`ls [-al] [<directory>]`	laat informatie zien over de inhoud van een `directory`; `-a` laat ook "setup" files zien ("dot files"), zoals `.login`; `-l` laat meer informatie zie (eigenaar, datum, grootte, ...)
`pwd`	laat huidige working directory (cwd) zien, doorgaans ook in de prompt genoemd
`cd [<directory>]`	verander cwd in `directory` (`cd` zonder directory maakt home-directory de cwd)
`mkdir <directory>`	maakt binnen cwd een nieuwe directory aan geheten `directory`
`mv <oudenaam> <nieuwenaam>`	als `nieuwenaam` een bestaande directory is, wordt `oudenaam` naar deze directory verplaatst; hernoem anders `oudenaam` in `nieuwenaam`
`rmdir <directory/ies>`	verwijder (lege) `directory/ies`
`chmod [auog][+-][rwx] <file(s)>`	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: `chmod 644 *` of `chmod 750 *`
`wc <file(s)>`	telt aantal regels, woorden en bytes van `file(s)`
`grep [<patroon>] [<file(s)>]`	doorzoekt `file(s)` op `patroon`, dat een (wellicht kryptisch) reguliere expressie zoals `'[kK][a-zA-Z]*[sS]'` kan zijn

Een voorbeeld: gebruik `rm core` om zogeheten core-dumps weg te gooien (grote files in noodgevallen door het systeem gemaakt); gebruik `cd ..` (en niet `cd..`) om naar de ouder-directory van de cwd te gaan.

2.4 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):

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

2.5 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.

2.6 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 filenamen 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.

2.7 De editor vi in een notedop

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:

`/Biden`	zoek naar string Biden

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.

2.8 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 valgrind om informatie te krijgen over de debugger valgrind, die op diverse systemen beschikbaar is. Als je executable eerste heet, run deze dan met

```
valgrind ./eerste
```

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), èn 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 gezipt “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`. Overigens heet `time.h` nu `ctime` .
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 doorlogen 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-syteem met internet kun je (user `biden` zijnde) gewoon intikken

```
ssh biden@sshgw.leidenuniv.nl
```

en files heen en weer transporteren met `scp`.
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/~kosterswa/4ltrwrd-1.pdf`

En misschien ben je geinteresseerd in het produceren van prachtig vormgegeven output: op zoek naar TEX of LATEX? Probeer

`http://www.liacs.leidenuniv.nl/~kosterswa/AI/lshort.pdf`

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

`http://www.google.com/`