# Programmeren met Linux

Henk Siewert

De reden dat computeraars overstappen naar Linux zijn onder meer de uitgebreide mogelijkheden voor het programmeren van van alles en nog wat. Uiteenlopend van kleine of grote programma's op de computer zelf tot het programmeren en communiceren met microcontrollers als de Raspberry Pico en Arduino's.
In dit artikel gaan we kijken wat daarvoor nodig en beschikbaar is.

## Wat hebben we nodig

Om te beginnen hebben we natuurlijk een computer nodig. Het is aan te bevelen om daar niet de gezinscomputer voor te gebruiken: met het experimenteren kan er veel misgaan. En Murphy zit altijd op de stoel naast de programmeur. Dus het is beter om daar een aparte computer voor te gebruiken. Dat hoeft niet een hele dure supermoderne pc te zijn. De voorbeelden die ik in dit artikel geef zijn uitgevoerd op een pc die ik heb gekocht in de kringloopwinkel voor 35 euro. Het is een HP Pavillion met 4 Gigabyte RAM en een 500 Gigabyte harde schijf, een hele serie USB-2 en -3 uitgangen, een HDMI poort en een dual core processor. Voor het draaien van Linux is dat meer dan genoeg.
Dat gezegd hebbende: ik draai op deze machine Xubuntu 20.04.4. Er is een versie 22 maar die is nog een beetje buggy en wil ik op deze computer niet installeren. Maar omdat nieuwer niet altijd beter is lijkt me dat geen probleem. Je zou zelfs versie 18 kunnen draaien. Werkt ook prima op dit soort systemen.

## Keuze en installeren

Misschien zou je bij het beginnen met Linux snel geneigd zijn om met Ubuntu te beginnen. Dat zou ik je niet aanraden. Ubuntu is leuk voor mensen die van Windows overstappen naar Linux. Maar wil je meer mogelijkheden om zelf dingen naar je eigen hand te zetten en vooral meer snelheid neem dan een versie met een eenvoudiger desktop zoals Xubuntu.

Een ISO is te downloaden van **xubuntu.org**. Brand die op een DVD en je bent klaar om te beginnen. De computer gaat booten van de DVD en geeft je de mogelijkheid om het eerst uit te proberen of direct te installeren. Het installeren gaat vanuit een grafisch programma en is echt niet ingewikkeld. Er zijn wel een paar dingetjes om op te letten. Zo krijg je op een gegeven moment een dialoog met:


Vink het vakje voor 'Installeer programmatuur van derden…' aan. Doe je dat niet dan kan het zijn dat bepaalde hardware-drivers niet worden geïnstalleerd.
Een ander aandachtpunt is het invullen van je naam en wachtwoord.

Bij 'Automatisch aanmelden' kun je aangeven of je bij het opstarten van de computer steeds je wachtwoord wilt intypen of niet.


## Talen

Na het installeren krijg je de desktop te zien en kan het programmeren beginnen. Ten minste als we daar het benodigde gereedschap voor installeren. We beginnen met C, C++, FORTRAN en 'last but not least' BASIC.
C en C++ zullen je waarschijnlijk niet verbazen, maar FORTRAN? Fortran is een oude taal, uit 1957, die nog steeds veel wordt gebruikt in zowel de wetenschap en de techniek, als in de bank- en financiële sector. Het is zelfs zo dat in de bank- en financiële sector het overgrote deel van de software is gemaakt in FORTRAN en COBOL. De New York Times publiceerde onlangs nog een artikel over FORTRAN - 'The Latin of Software Code Is Thriving' en twee jaar geleden 'The Code That Controls Your Money' over COBOL.
COBOL ga ik hier niet behandelen. Die taal is bedoeld voor het werken met grote gegevensbestanden. Heeft niet mijn voorkeur. Trouwens, wist u dat de enorme computers die worden gebruikt voor het opstellen van weersverwachtingen, zoals bij het KNMI, in FORTRAN worden geprogrammeerd? Zelf gebruik ik FORTRAN voor het maken van berekeningen voor één van mijn hobby's: het ontwerpen, bouwen en lanceren van raketten. Als er rekenkracht wordt vereist, is FORTRAN je taal!

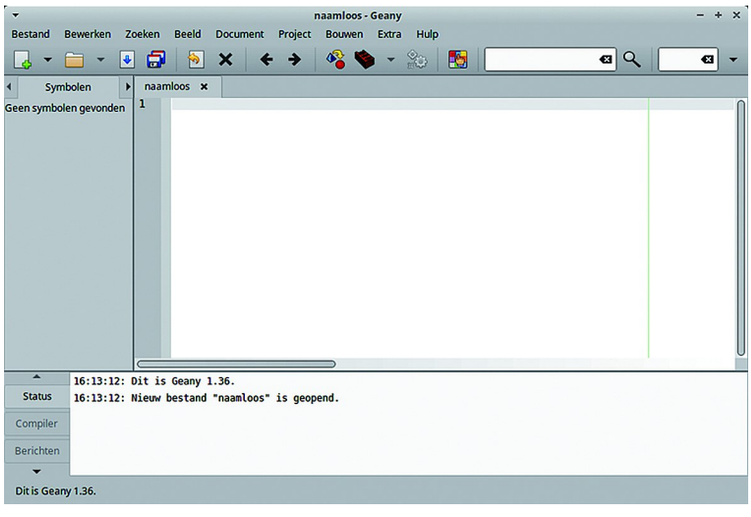
## C

We beginnen met C. Open het terminalprogramma en typ *gcc*. Zeer waarschijnlijk krijg je dan de foutmelding: 'Opdracht '*gcc*' niet gevonden'. OK, je gaat nu de C- en C++-compiler installeren.
Type in het terminalvenster: sudo apt update
Dat is om op te zoeken welke programma's kunnen worden geüpdatet. Dat doen we altijd voordat we een nieuw programma gaan installeren.
Als je de melding krijgt dat er pakketten kunnen worden geüpdatet, type dan: sudo apt upgrade
Na het verschijnen van de prompt kunnen we de compilers installeren. Type daarvoor: sudo apt install gcc, en daarna: sudo apt install g++
Als je nu in het terminalvenster gcc intypt krijg je de foutmelding: *gcc: fatal error: no input files*. Klopt, we hebben nog geen broncode. Die broncode, het programma, gaan we nog maken.

Nu gaan we FORTRAN installeren. Typ in het terminalvenster:
sudo apt install gfortran

## Editor

Mooi, dan ga je nu broncode maken om de compilers te testen. En daar hebben we een editor voor nodig. Een heel geschikte editor voor het schrijven van programmacode in Linux is *Geany*. Maar die is niet standaard geïnstalleerd. Dus ga je weer naar het terminalvenster en typ je:
sudo apt install Geany. Als alles goed is gegaan vind je in het menu onder Ontwikkeling een shortcut naar Geany. Klik hierop om het programma te starten. Je krijgt dan het volgende te zien:




Dit is ons programmeursgereedschap.
Je gaat nu het standaard *hello programma* intypen in de editor:

```
#include<stdio.h>
int main(void)
{
printf("\n Hello World from C-code in Geany.\n");
return 0;
}
```

Sla het programma op onder de naam *hello.c*. Je zou nu in het terminalvenster de opdracht kunnen geven om het programma te compileren. Maar het kan ook direct vanuit Geany. Geany is al uitgerust met de mogelijkheid om de GNU-compilers die je hebt geïnstalleerd, te gebruiken. Dat is wel zo makkelijk. Klik op het compileerknopje (met het gele pijltje, de piramide en het bolletje). Je krijgt dan een vergelijkbare melding als:

*Compilatie met succes beëindigd*

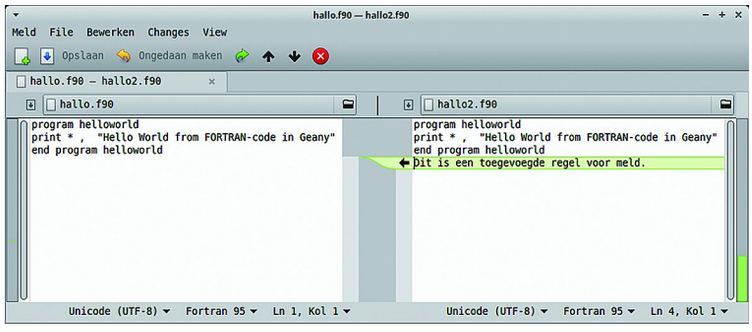
Klik dan op de buildknop – rechts naast de compileerknop - en klik dan op de runknop (met het tandwieltje). Er opent zich dan een terminalvenster met de tekst: *Hello World from C-code in Geany. Succes!* Je hebt nu een C-programma geschreven, gecompileerd en gestart.
Natuurlijk gaan we nu ook een FORTRAN-programma maken.

```
program helloworld
print * ,  "Hello World from FORTRAN-code in Geany"
end program helloworld
```

Opslaan als *hello.f90* en compileren, net als bij het C-programma. De *.f90* extensie geeft aan dat het programma is geschreven volgens de FORTRAN 90-norm. Het resultaat is dan een venster met: *Hello World from FORTRAN-code in Geany.* Goed zo, je kunt nu in twee belangrijke talen programmeren. Maar er zijn natuurlijk nog veel meer talen die beschikbaar zijn in Linux, waaronder BASIC, PASCAL, JAVA, enzovoort. Daar komen we nog op terug.

## Vergelijken

Een ander handig stuk gereedschap voor de programmeur is *Meld*. Meld is een programma om verschillende broncode-bestanden met elkaar te vergelijken. Als je verstandig bent, sla je de broncode na verandering onder een aangepaste naam op. Zodat je altijd op vorige versies kunt terugvallen. Maar op een gegeven moment weet je misschien niet precies meer wat de verschillen zijn tussen de verschillende versies. Dat kun je controleren door de versies in ***Meld*** te laden, waardoor in Meld de verschillen worden aangegeven.
Meld is beschikbaar in het Software Installatie-programma. Dus installeren is heel makkelijk.
Hieronder zie je twee versies van ons FORTRAN-programma en hoe de verschillen worden aangegeven.




## Microcontrollers

Misschien nog wel belangrijker dan het maken van programma's die op de computer zelf draaien, is tegenwoordig het programmeren van externe elektronica. Meestal in de vorm van een microcontroller. De bekendste zijn denk ik wel de Raspberry Pico en de Arduino. Deze microcontrollers moet je niet verwarren met een computer als de Raspberry Pi. De Raspberry Pi heeft een besturingssysteem, Linux bijvoorbeeld, nodig om te kunnen werken. Microcontrollers gebruiken, op een enkele uitzondering na, geen besturingssysteem.

## Arduino

We gaan beginnen met de Arduino. Op de website van de Arduino, **https://www.arduino.cc**, is een editor speciaal voor deze microcontroller te downloaden. Gebruik liever niet de versie die via het Software -programma beschikbaar wordt gesteld. Dat is een oudere versie. Die bovendien in een soort afgeschermd gedeelte draait, waardoor bepaalde microcontrollers die gebruik moeten maken van Python niet gebruikt kunnen worden.
Haal de voor jouw systeem geschikte versie op. Pak het bestand uit – klik rechts op het bestand en kies '*Hier uitpakken*' of '*Uitpakken naar…*' - en open het in de map waarin het bestand is uitgepakt in het terminalvenster. Dat kan ook vanuit de bestandsbeheerder, als je de betreffende map hebt openstaan, door op F4 te drukken. Typ vervolgens ./install.sh en laat dit bash-script zijn werk doen. Als het goedgaat, verschijnt er op de desktop een shortcut naar de ArduinoIDE. Anders staat er in het menu bij 'Ontwikkeling' een shortcut. Open het programma. Je gaat nu het *hallo equivalent* voor de Arduino openen. Ga naar '*Bestand*', '*Voorbeelden*', '*Basics*' en '*Blink*'. Open het programma. (zie afbelding op volgende pagina)

Om het programma te sturen naar de Arduino, die je natuurlijk inmiddels hebt aangesloten op een USB-poort, moet bij 'Hulpmiddelen' het juiste bordje worden gekozen en daaronder bij poort de juiste USB-aansluiting. Klik vervolgens op de upload-knop en, jawel: een foutmelding! Dat is even vervelend. Zeker als je dit al eens van uit de Raspberry Pi of Windows hebt gedaan. Maar Linux is net even anders. Daar mag niks zonder dat je daar toestemming voor hebt gegeven.

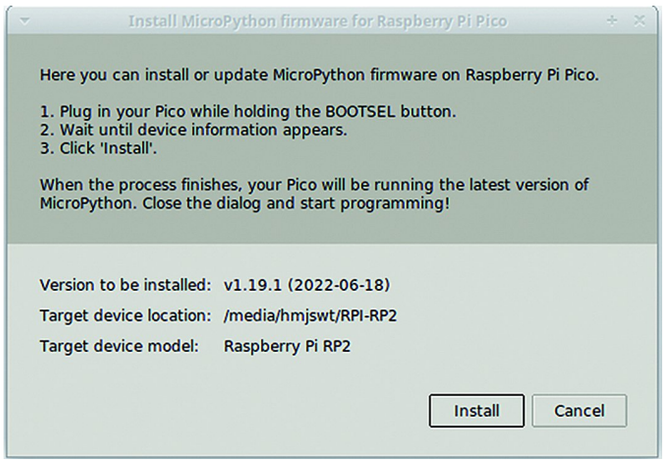

En dat heb je nog niet. Maar daar ga je verandering in brengen. Open het terminalvenster en typ:
sudo usermod -a -G tty <inlognaam> && sudo usermod -a -G dialout <inlognaam>

Voor <inlognaam> gebruik je dan je eigen inlognaam ( zonder de haakjes ). Om deze aanpassing te activeren even opnieuw opstarten. Dit hoef je maar één keer te doen. Als je na het opstarten de ArduinoIDE opent met *blink* en op de upload-knop drukt, zal het wel goedgaan en begint de LED op de Arduino te knipperen.

## Raspberry Pico

Voor het programmeren van een Raspberry Pico kan ik je het gebruik van *Thonny* aanbevelen. Ook hier niet installeren met het 'Software'-programma, dat is weer een oude versie. Open voor het installeren van Thonny het terminalvenster en type:
bash <(wget -O - https://thonny.org/installer-for-linux)

Na het ophalen en installeren van het bestand kun je aan de slag gaan. Sluit je Pico aan op een USB-poort. Er opent zich dan een RPI-RP2 venster. Sluit dit maar. Open vervolgens Thonny.




Thonny opent dan met de mogelijkheid om MicroPython op de Pico te installeren. Klik op '*Install*' en vervolgens op '*Close*'. In de rechterbenedenhoek van het Thonny-venster verschijnt dan de mededeling: '*MicroPython (Raspberry Pi Pico)*'.
Je gaat nu de LED van de Pico laten knipperen:

```
from machine import Pin, Timer
led = Pin(25, Pin.OUT)
timer = Timer()

def blink(timer):
    led.toggle()

timer.init(freq=2.5, mode=Timer.PERIODIC, callback=blink)
```

Sla het programma op onder de naam *blink.py* en start het programma. Als je alles goed hebt gedaan gaat de LED op de Pico knipperen. Let op! Dit gaat alleen op de Pico zonder wi-fi. Op de nieuwe PicoW zit de LED op een andere uitgang. Maar dat komt later in een ander artikel aan de beurt.

## BASIC

Ja, ja, BASIC, er valt niet aan te ontkomen. De meeste wat oudere gebruikers van Linux zijn hoogstwaarschijnlijk ooit begonnen met een Micro Computer die opstartte in BASIC. Als je even snel een stukje code wilt uitproberen of een programmaatje wilt maken, is BASIC nog steeds ideaal.


Om met BASIC aan de slag te gaan is MMBASIC beschikbaar. De speciale versie voor Linux is beschikbaar op:
https://github.com/thwill1000/mmb4l/releases/tag/2022.01.00-a4

Pak het gedownloade bestand uit en start MMBASIC en je kunt aan de slag. Het is mogelijk dat het programma niet wil opstarten.

Het is dan nog niet gewaarmerkt als uitvoerbaar bestand. Dat is op te lossen door in de directory waar *mmbasic* staat een terminalvenster te openen en het volgende in te typen:
sudo chmod +x mmbasic