Linux introductie

D. Leeuw

10 april 2025
v.1.1.2

Over dit Document

Dit document behandeld Linux voor het middelbaar beroepsonderwijs in Nederland, maar kan breder ingezet worden, daar het gericht is op het behalen van het LPI Linux Essentials examen. De doelgroep is niveau 4 van het MBO, met enige kennis van computers.

Versienummering

Het versienummer van elk document bestaat uit drie nummers gescheiden door een punt. Het eerste nummer is het major-versie nummer, het tweede nummer het minor-versienummer en de laatste is de nummering voor bugfixes.

Om met de laatste te beginnen als er in het document slechts verbeteringen zijn aangebracht die te maken hebben met type-fouten, websites die niet meer beschikbaar zijn, of kleine foutjes in de opdrachten dan zal dit nummer opgehoogd worden. Als docent of student hoef je je boek niet te vervangen. Het is wel handig om de wijzigingen bij te houden.

Als er flink is geschreven aan het document dan zal het minor-nummer opgehoogd worden, dit betekent dat er bijvoorbeeld plaatjes zijn vervangen of geplaatst/weggehaald, maar ook dat paragrafen zijn herschreven, verwijderd of toegevoegd, zonder dat de daadwerkelijk context is veranderd. Een nieuw cohort wordt aangeraden om met deze nieuwe versie te beginnen, bestaande cohorten kunnen doorwerken met het boek dat ze al hebben.

Als het major-nummer wijzigt dan betekent dat dat de inhoud van het boek substantieel is gewijzigd om bijvoorbeeld te voldoen aan een nieuw kwalificatiedossier voor het onderwijs of een nieuwe versie van Linux Essentials van de LPI. Een nieuw major-nummer betekent bijna altijd voor het onderwijs dat in het nieuwe schooljaar men met deze nieuwe versie aan de slag zou moeten gaan. Voorgaande versies van het document zullen nog tot het einde een schooljaar onderhouden worden, maar daarna niet meer.

Document ontwikkeling

Het doel is door middel van open documentatie een document aan te bieden aan zowel studenten als docenten, zonder dat hier hoge kosten aan verbonden zijn en met de gedachte dat we samen meer weten dan alleen. Door samen te werken kunnen we meer bereiken.

Bijdragen aan dit document worden dan ook met alle liefde ontvangen. Let u er wel op dat materiaal dat u bijdraagt onder de CC BY-NC-SA licentie vrijgegeven mag worden, dus alleen origineel materiaal of materiaal dat al vrijgegeven is onder deze licentie.

De eerste versie is geschreven voor het ROC Horizon College.

Versienummer	Auteurs	Verspreiding	Wijzigingen
1.1.0	Dennis Leeuw		Installatie opdracht voor CentOS verplaatst naar opdrachten document.
1.0.0	Dennis Leeuw		Veel taalfouten verwijderd en herschrijvingen om het een beter leesbaar document te maken.
0.9.0	Dennis Leeuw		Eerste release na splitsing GUI document in een Intro en een GUI document

Tabel 1: Document wijzigingen

Inhoudsopgave

Hoofdstuk 1

Inleiding

Deze Linux cursus beoogt aan te sluiten bij het Linux Essentials examen van de LPI (Linux Professional Institute) en dient als voorbereiding op het MBO ICT Systems and Devices Expert examen. Voor het leren gebruiken van de grafische interface en de command line maken we gebruik van CentOS en om kennis te maken met het gebruik van Linux als server/command line installeren we Debian. De keuze om CentOS als werkstation te installeren en Debian als server is volledig willekeurig. Het doel is dat de studenten kennis maken met de zowel rpm/dnf en de apt package managers en leren dat het ene Linux systeem het andere niet is.

Alle Linux systemen zullen geïnstalleerd worden als virtuele machines. Door gebruik te maken van virtuele machines zijn we niet afhankelijk van de onderliggende hardware. De keuze van de virtuele omgeving is aan de gebruiker. Ons advies zou zijn om gebruik te maken van VirtualBox en VMware Workstation, beide zijn gratis en kunnen op de meest gangbare operating systems gebruikt worden.

Voor de CentOS machine is 15G vrije schijfruimte nodig en voor het Debian systeem 5G, wat een totaal aan 20G vrije schijfruimte vereist. Voor elke machine hebben we 2G RAM nodig, dus een totaal van 4G RAM moet vrij beschikbaar zijn.

Hoofdstuk 2

Wat is Linux?

Om de vraag te kunnen beantwoorden wat Linux is moeten we een stukje terug in de tijd en wel naar het eind van de jaren '60 uit de vorige eeuw. Een groepje programmeurs werkend aan een project genaamd Multics bij Bell Labs van AT&T zaten op een dood spoor of beter Bell Labs vond het een dood spoor en cancelde het project, dus schreef een van die programmeurs in ongeveer een maand tijd in 1969 een simpeler alternatief. De ontwikkelaar was Ken Thompson die samen met Dennis Ritchie een team leidde.

Een ander lid van het team Brian Kernighan schijnt met de naam Unics gekomen te zijn als woordgrap op Multics. Hoe de naam ooit Unix is geworden is niet bekend.

De gedachte achter Unix is ooit in 1979 mooi beschreven door Dennis Ritchie: "What we wanted to preserve was not just a good environment in which to do programming, but a system around which a fellowship could form. We knew from experience that the essence of communal computing, as supplied by remote-access, time-shared machines, is not just to type programs into a terminal instead of a keypunch, but to encourage close communication." Het ging dus om een systeem waarop samengewerkt kon worden en dan vooral geprogrammeerd.

Uit de wens vanuit de organisatie om tekst te kunnen verwerken werd het systeem uitgebreid met tekstverwerkingsfuncties en een eerste tekst editor en tekst formatter. Hieruit blijkt meteen de filosofie achter Unix. Maak kleine tools die één ding goed doen en gebruik ze gezamenlijk om complexe dingen te doen. Er is dus een editor en een formatter. De eerste tekst formatter heette roff en deze werd als snel opgevolgd door troff, die je nog steeds terug vindt op de systemen. De unix manual-pages, waarover later meer, worden opgemaakt met behulp troff.

De editor is inmiddels verschillende keren verbeterd, wat een ander voordeel is van dit "small is beautiful" principe. Je kan kleine stukjes aanpassen

terwijl wat goed is behouden blijft.

De eerste versies van Unix waren geschreven in assembly. Assembly is een programmeertaal die volledig toegespitst is op de onderliggende hardware. Iets dat in assembly geschreven is werkt dan ook alleen op één type machine. In 1974 kwam Unix versie 4 uit die bijna volledig herschreven was in de taal C. C is niet machine afhankelijk, er is een compiler nodig om de taal om te zetten naar de machine afhankelijke machinetaal. Door het gebruik van C werd het mogelijk Unix ook op andere systemen te draaien dan het PDP systeem waar het oorspronkelijk voor geschreven was. Unix kon de wereld gaan veroveren.

Bell Labs mocht door juridische afspraken met de Amerikaanse overheid geen commerciële zaken ondernemen buiten de telefonie, daardoor werd de vraag naar Unix beantwoord door alleen geld te rekenen voor de media (tapes) en het verzenden van het systeem, en niet voor het product zelf.

Omdat het systeem goedkoop verkrijgbaar was was het een ideaal systeem voor universiteiten. Studenten konden er relatief goedkoop op leren programmeren. Door studenten en andere gebruikers werden er in de loop van de tijd meer en meer programma's geschreven en verbeteringen gemaakt. Deze verbeterde stukken software werden met elkaar gedeeld. Van verkoop was nog geen sprake. Het voelde als hobby en leer projecten. Soms waren de wijzigingen zo groot dat er een variant van het oorspronkelijk Unix werd gedeeld (gedistribueerd). Deze collecties van programma's werden dan ook distributies genoemd. Eén van de meest bekende van deze distributies is die van University of California die ontwikkeld werd door de Berkeley Computer Systems Research Group, de Berkeley Software Distribution of afgekort BSD.

Een klein bedrijfje in de US was een van de eerste bedrijven die Unix naar de microcomputer, ofwel de PC, bracht. Tot dan draaide het eigenlijk alleen op grote servers. Hun Unix versie heette Xenix en het bedrijf stond bekend als Microsoft. Grote bedrijven als IBM (AIX) en HP (HPUX) hebben hun eigen Unix versie voor grote machines.

De vele verschillende systemen die ontstonden waren voor de eindgebruikers een ramp. Wat op het ene systeem aanwezig was was op een andere niet aanwezig en omgekeerd. Een Unix-applicatie op de ene distributie werkte werkte niet zomaar ook op een andere. Gebruikers wilden standaardisatie. Software moest uitwisselbaar zijn. De standaard die ontstond in 1988 is de POSIX standaard van de IEEE. Het beschrijft welke software minimaal aanwezig moet zijn en ook aan welke functionalitiet die software minimaal moet voldoen. Extra functionaliteit is dus geen probleem. Als een systeem aan de eisen van POSIX-standaard voldoet heet die dan ook POSIX-compliant.

In 1983 bracht Richard Stallman een nieuw project in de wereld genaamd

het GNU Project, waarbij GNU staat voor GNU's not Unix! Dit om aan te geven dat het gebaseerd is op Unix maar geen Unix componenten bevat. GNU is een "from scratch"geschreven systeem dat volledig open source is. From-scratch betekent dat alle code opnieuw geschreven is en Open Source wil zeggen dat van het hele systeem alle broncode openbaar beschikbaar is, hierdoor kan het gebruikt worden als studiemateriaal. Het heeft tevens het voordeel dat iedereen fouten uit de software kan halen en aan de software kan bijdragen om het beter te maken. Dit is dan ook altijd de strijd van Richard Stallman geweest en zijn daarvoor opgerichte organisatie The Free Software Foundation, software moest vrij zijn en voor iedereen toegankelijk. Om dit te bewerkstelligen stelde Stallman het copyleft in, in plaats van het copyright, en maakte een licentie genaamd de GPL, GNU Public License, die ervoor moest zorgen dat de gemaakte software niet zomaar weer gesloten kon worden.

De software werd door vele verschillende distributies in die tijd gebruikt om software van Unix geheel of gedeeltelijk te vervangen. Missend onderdeel was echter jarenlang een kernel. Het stukje software dat ligt tussen de gebruikersinterface en de hardware.

In 1991 schreef een Finse student met de naam Linus Torvalds een berichtje in een news-group dat hij bezig was met zo'n kernel. Niets groots zei hij, maar wel onder de GPL-licentie van de Free Software Foundation. Velen werden aangetrokken door dit project en gingen Linus helpen zijn kernel verder uit te breiden. Die kernel ging naar de maker heten en werd Linux genoemd.

Linux is zoals gezegd een kernel; een abstractie laag tussen de hardware en de gebruikers interface. Voor een compleet systeem is er veel meer nodig. De overige software op wat wij nu een LInux systeem noemen is dan ook veelal afkomstig van het GNU-project. Fanatiekelingen willen dan ook graag dat je spreekt van GNU/Linux, maar de hele wereld spreekt meestal gewoon over Linux als we een systeem bedoelen met een Linux kernel.

De Linux kernel kan natuurlijk gecombineerd worden met vele software pakketten en dat gebeurt ook. En net als bij Unix werden de systemen die ontstonden distributies genoemd. Eén van de eerste distributies was Slackware, later kwamen SuSE, Debian, Red Hat, CentOS, Ubuntu en nog vele vele anderen.

Het meest gebruikte Unix desktop systeem is ongetwijfeld Darwin het open source besturingssysteem van Apple dat de basis vormt voor Mac OS X en op de telefoon is dat Android het systeem van Google dat een Linux kernel en andere open source tools onderwater gebruikt.

En zo komen we na een lang verhaal aan het einde van deze simpele geschiedenis over Unix en Linux. Waarin hopelijk duidelijk is geworden waarom

Linux Linux wordt genoemd, duidelijk is waarom Linux open source is en waarom delen zo'n belangrijk onderdeel is van de cultuur rond Unix en Linux systemen.

Hoofdstuk 3

Waarom Open Source?

Wat Linux en het GNU project bijzonder maken is het feit dat alle code open source is. Je mag er mee doen en laten wat je wilt, je mag het aanpassen, je kan het inzien en je mag het natuurlijk gewoon gebruiken. Voor het merendeel is de software nog gratis ook. Dat is natuurlijk bijzonder in een wereld die draait om commercie. Daarom willen hier iets dieper duiken in de wereld van open source.

De eerst versies van Unix zoals geschreven door Ken Thompson, Dennis Ritchie en de overige leden van het team bij Bell Labs was geschreven is de assembly language. Assembly language is een taal die heel dicht ligt bij wat computers snappen en daarmee altijd hardware afhankelijk is. In de tijd dat Unix werd ontwikkeld geloofde men dat je assembly language nodig had om een besturingssysteem snel genoeg te laten zijn. Dennis Ritchie nam de taal B, ontwikkeld door Ken Thompson, maakte verbeteringen en kwam met C in 1972. In 1973 kwam Unix versie 4 uit die voor een groot deel herschreven was in C en daarmee aantoonde dat een hogere programmeertaal gebruikt kon worden om besturingssystemen in te schrijven die snelgenoeg waren, maar belangrijker nog omdat er een hogere programmeertaal werd gebruikt was Unix opeens overdraagbaar naar andere hardware en dat had voor de ontwikkeling van software grote voordelen. Software kon opeens geschreven worden op het ene systeem en gebruikt worden op een totaal ander systeem.

Voor het programmeren in C heb je een C-compiler, een linker en een C-Library nodig. Library is het Engelse woord voor bibliotheek. Een C-bibliotheek bevat een aantal standaardfuncties die je kan gebruiken in een programma. Een heel simpel programma als Hello World ziet er in C zo uit:

```
#include <stdio.h>

int main() {
        printf("Hello, World!");
```

```
        return 0;
}
```

De printf functie die de woorden "Hello World!" op het scherm laat zien is zo'n standaard functie uit de C-Library. De C-bibliotheek bevat een aantal standaardfuncties die je kan gebruiken in een programma, zo hoeft niet elke programmeur de printf functie te programmeren. Functies in een library zijn dus kleine stukjes code die je met elkaar delen kan zodat iedereen in zijn programma deze functies gebruiken kan mits de library op het systeem aanwezig is.

De compiler is verantwoordelijk voor het omzetten van de C-code in machinetaal. Machinetaal is binair, daar computers werken met 1 en 0 en niets anders. Het proces om C-code om te zetten in binaire code heet dan ook compileren. De compiler vertaalt alles wat er geschreven is in C in 1-en en 0-en zodat de computer ermee kan werken. Maar omdat je ook functies gebruikt uit de C-library moet ook daar nog iets mee gebeuren, daarvoor zorgt de linker. De linker zorgt ervoor dat de functie uit de C-library gelinkt wordt aan het programma dat je geschreven hebt. Een compiler is niet alleen hardware afhankelijk, maar ook operating systeem afhankelijk. Een compiler voor Mac OS X maakt van C machinetaal voor Mac OS X en een C-compiler voor Windows maakt machinetaal voor Windows.

Er zijn twee manieren waarop de linker ervoor kan zorgen dat bijvoorbeeld de printf-functie gelinked kan worden met je programma. Het kan statisch en dynamisch. Statisch betekent dat een kopie van de functie toegevoegd wordt aan je programma. Bij dynamisch linken betekent het dat er in je programma een verwijzing komt te staan naar de binaire printf-functie in die specifieke binaire C-library. Je C-programma wordt zo afhankelijk van deze specifieke versie van de C-library die aanwezig is op je systeem.

Het voordeel van statisch linken is dat het programma onafhankelijk is van de C-library, het nadeel is dat het binaire-programma vele malen groter wordt omdat dat alle code uit de C-library (of andere bibliotheken) ook aan het programma wordt toegevoegd. Bij dynamisch linken is dit juist omgekeerd. Het programma blijft kleiner, laadt daardoor sneller van disk en neemt minder geheugen ruimte in, maar dat gaat tenkoste van de portabiliteit, kortom het kan alleen nog gebruikt worden op systemen die exact dezelfde versies van de libraries geïnstaleerd heeft. Dat laatste is geen probleem zolang je het programma gebruikt op systemen die dezelfde libraries hebben als jij, zoals het geval is bij mensen die dezelfde distributie gebruiken als jij. Ook als er een kleine wijziging gemaakt wordt in de printf-functie die geen invloed heeft op de binaire syntax van de printf-functie dan kan je programma gelijk gebruik maken van deze verbetering doordat je alleen de

C-library update. Je hoeft je programma dan niet opnieuw te compileren. Er zijn dus vele voordelen aan dynamisch linken en daarmee is het dan ook de meest gebruikte methode op Linux systemen.

Dat dynamisch linken klinkt allemaal vreselijk complex en dat is het ook. Er zijn momenten waarop er zoveel wijzgingen zijn in bijvoorbeeld een nieuwe C-library dat jij je programma opnieuw moet compileren om het te laten werken met die nieuwe C-libary, maar kleine wijzigingen in de C-library kunnen ervoor zorgen dat dat niet hoeft en dan blijft je programma gewoon werken. Dat verschil heeft te maken met de API (Application Programming Interface) en ABI (Application Binary Interface). De API van een functie is de syntax van de functie, als deze verandert dan moet je je programma opnieuw compileren en als het tegenzit moet je zelfs je code aanpassen. Als echter de API blijft zoals hij is en er zijn alleen kleine wijzigen zodat de ABI niet veranderd, dan heeft je programma geen last van de wijziging.

Programma's die werken op je telefoon, je Windows systeem of op Mac OS X worden je vaak aangeboden als binary, kortom ze zijn al door iemand gecompiled. Deze applicaties kan je direct gebruiken, maar alleen op het systeem waarvoor ze gecompiled zijn. Je kan een Windows .exe niet gebruiken op een Mac OS X systeem.

Als je (ook) de beschikking hebt over de broncode dan kan je die code ook compilen op je eigen computer en zorgen dat die ook werkt op jou systeem. Je bent dan niet meer afhankelijk van een leverancier die jou systeem moet ondersteunen. Soms moet je wel wat aanpassingen maken om het geheel goed te laten werken. De grafische interface van Windows is heel anders dan die van Mac OS X, en daar zit dan ook een heel andere library onder. Dus als je een Windows applicatie op een Mac wil compileren zul je wel wat programmeer werk moeten doen. Maar als een applicatie is geschreven op een Debian systeem dan kan deze meestal zonder enige wijziging gecompileerd worden op een CentOS systeem.

En daar zit de kracht van open source. Met het delen van de broncode wordt de reikwijdte van die software groter. Zoals gezegd moet je soms wel wijzigen maken om het te laten werken en dus is het bijna een eis dat je de software ook mag aanpassen en die eisen zijn in open source licenties vastgelegd.

Hoofdstuk 4

Gebruiksrechten en licenties

Het auteursrecht stamt uit 1710 en was oorspronkelijk bedoeld om de drukker te beschermen. Later is dit over gegaan op de auteur. Omdat ook een programmeur een schrijver is geldt er voor broncode dezelfde rechten als op boeken. Maar wat houdt auteursrecht nu eigenlijk in.

Volgens de wet:

"Het auteursrecht is het uitsluitend recht van de maker van een werk van letterkunde, wetenschap of kunst, of van diens rechtverkrijgenden, om dit openbaar te maken en te verveelvoudigen, behoudens de beperkingen, bij de wet gesteld." (Artikel 1 Auteurswet)

Dit zegt dus iets over het openbaar maken en het kopiëren (verveelvoudigen) van het gemaakte werk. De auteur mag dit doen en niemand anders (uitsluitend recht van de maker). Als een programmeur software schrijft en dit aan iemand anders geeft, dan mag deze de software gebruiken maar niet kopiëren en verder verspreiden.

Voor de auteur zijn de regels dus in de wet vastgelegd, welke rechten een gebruiker van de software heeft dat mag de auteur zelf bepalen. Deze rechten worden vastgelegd in een gebruikerslicentie. Elke auteur kan zijn eigen licentie maken, wat veel bedrijven dan ook doen. De meest bekende is waarschijnlijk Microsoft's EULA; End-Users License Agreement. De EULA zegt in het kort dat Microsoft niet verantwoordelijk is voor de gemaakt software en eventuele fouten die het mocht bevatten, dat je het op 1 machine mag gebruiken, je mag geen kopieën mag maken en het mag maar door 1 persoon tegelijkertijd gebruikt worden.

Omdat Unix ontstaan is door software met elkaar te delen zijn er andere licenties ontstaan. De meestvoorkomende zullen we in dit hoofdstuk behandelen.

4.1 Gebruikslicenties

4.1.1 MIT

De MIT (Massachusetts Institute of Technology) licentie geven we hier in zijn geheel weer omdat hij de basis vormt voor veel van de erop volgende licenties.

Copyright <YEAR> <COPYRIGHT HOLDER>
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Eigenlijk zegt deze licentie dat je alles met de software mag doen, behalve het weghalen van de copyright en de licentievoorwaarden. Ook is er een duidelijk statement dat de maker niet verantwoordelijk is voor wat de software doet. Deze laatste kom je in bijna elke licentie tegen. Ook in de EULA van Microsoft staat dit voorbehoud.

4.1.2 BSD

De BSD licentie, of beter de 3-clause BSD-licentie, voegt daar wat regels aan toe:

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

De belangrijkste afwijking is de laatste regel, namelijk dat de naam van de schrijver van de software niet gebruikt mag worden voor promotionele doeleinden zonder schriftelijke toestemming. Dit was om te voorkomen dat een gewijzigde versie verspreid zou worden onder de naam van de oorspronkelijke auteur alsof deze de wijzigingen goed gekeurd zou hebben. Het was dus belangrijk dat er een onderscheid kwam tussen de oorspronkelijke code en de ervan afgeleide producten.

4.1.3 GPL

De GPL van de Free Software Foundation voegt er vele regels aan toe. De belangrijkste regel hierin is dat als je wat wijzigt in de software dan moet je die gewijzigde broncode ook weer openbaar maken. Wat bij de vorige twee licenties mogelijk is is de code te nemen, er wijzigingen in aan te brengen en dan alleen een gecompileerde (binaire) versie te leveren zonder de broncode. Dat mag met de GPL niet meer. Elke wijziging die je niet alleen voor je zelf maakt, daarvan moet je ook de broncode weer publiek beschikbaar stellen.

Als het goed is, althans dat is het doel van deze licentie, wordt elke verbetering zo openbaar en wordt de software beter. Je kunt een probleem dat je opgelost hebt dus niet meer alleen voor jezelf houden.

4.2 Creative Commons

Naast software wordt er natuurlijk ook documentatie geschreven voor de software, worden er logo's gemaakt voor software en worden er websites ontworpen, soms worden er ook filmpjes gemaakt. Kortom we willen vaak meer creatieve uitingen vangen in een licentie dan alleen de software. In het begin werden de software licenties ook voor deze zaken gebruikt, maar dat bleek niet altijd toereikend. De oplossing is uiteindelijk gevonden in een set van voorwaarden die bekend staan als de Creative Commons.

Een Creative Commons (CC) licentie zegt dat je werk hergebruikt mag worden. Met wat toevoegingen aan de licentie kan je zelf bepalen wat er wel

niet met je werk gedaan mag worden. De meest eenvoudige vorm is de CC BY. Deze zegt dat iedereen van alles met je werk mag doen, maar dat ze daarbij altijd aan naamsvermelding moeten dien. Een beetje zoals de MIT licentie.

Zaken die je toe kan voegen zijn:

ND staat voor dat er Geen Afgeleide werken gemaakt mogen worden. Een logo mag dus wel gebruikt worden maar mag niet gewijzigd worden. ND is een afkorting voor No Derivatives.

NC wat staat voor Niet Commercieel (Non Commercial). Het product of een afgeleide ervan mag niet commercieel gebruikt worden.

SA staat voor dezelfde licentie (Share Alike). Het product of een afgeleide daarvan mag dan niet van licentie veranderen. Dit lijkt erg op de GPL licentie.

Dit boek is uitgebracht onder de CC BY NC SA. Dat wil dus zeggen dat als er een gewijzigd werk gemaakt wordt van dit document dan moeten daar de namen van de auteurs van dit document bij vermeld worden, mag het niet commercieel uitgegeven worden en moet het onder dezelfde licentie verspreid worden. Je mag dus wel geld vragen voor het feit dat je een boek bijvoorbeeld gedrukt hebt.

4.3 Open standaarden

Een verwarring die weleens wil ontstaan is het verschil tussen open source en open standaarden en toch is daar een wezenlijk verschil.

Open standaarden beschrijven hoe bijvoorbeeld data uitgewisseld kan worden. Protocollen als SMTP, POP3, IMAP, Telnet en FTP zijn allemaal beschreven in documenten die vrij op Internet toegankelijk zijn. Iedereen, dus ook de open source wereld, kan deze standaarden implementeren en er dus voor zorgen dat verschillende systemen, open source en commercieel, met elkaar kunnen communiceren. Gesloten protocollen die door een bedrijf zijn bedacht kunnen alleen door dat bedrijf gebruikt worden, hoewel door luisteren op het netwerk er natuurlijk ook gekeken kan worden hoe het protocol werkt.

Ook voor het uitwisselen van data is het van belang dat er open standaarden zijn. Het feit dat je een document dat je in Microsoft Word maakt alleen in Word kan lezen is natuurlijk een enorme beperking van je vrijheid. Je document zou in elke willekeurige tekstverwerker te openen en wijzigen moeten

zijn, het is tenslotte jouw tekst. Toch is er pas sinds 2005 een open standaard voor office documenten. In 2005 werd door Organization for the Advancement of Structured Information Standards (OASIS) de OpenDocument standaard goedgekeurd. Deze standaard beschrijft hoe office documenten eruit moeten zien en omdat het een open standaard is kan iedereen de standaard implementeren, daarmee zou het mogelijk moeten zijn dat elke tekstverwerker een OpenDocument tekstdocument moet kunnen lezen en schrijven.

4.4 Open data

In navolging van open source en open standaarden kwam er ook steeds meer de vraag op hoe dat zit met data. Kan data ook open zijn?

Is onderzoeksdata van een universiteit van de universiteit, van de onderzoeker of van de overheid (of instantie) die het onderzoek betaald heeft? En als die data gedeeld wordt met de wereld wat mag je er dan mee doen. Mag je de data van een onderzoek wijzigen?

Al dit soort zaken zijn het domein van de open data. Het is dus belangrijk dat bij data ook een keuze gemaakt wordt wat er wel en niet mee mag gebeuren.

4.5 Open Source Business Model

Een veel gehoord tegenargument bij (tegen) het gebruik van een open source licentie bij de ontwikkeling van nieuwe software is dat er geen geld te verdienen zou zijn op deze manier en dat is gedeeltelijk waar. Het één keer ontwikkelen van software, het daarna eindeloos kopiëren en er geld voor vragen, dat werkt niet meer. Maar je mag nog altijd geld vragen voor de distributie, je mag (installatie) hulp aanbieden (helpdesk), cursussen aanbieden, of adviesuren verkopen. Er blijven dus voldoende middelen over om geld te verdienen.

Hoofdstuk 5

Linux Distributies

Een Linux distributie is een collectie van software samen met de Linux-kernel. Veel van de software is afkomstig van het GNU-project. Makers van een distributie maken hun eigen keuzes welke software zij belangrijk vinden. Daarom zijn er ook veel verschillende distributies omdat er zoveel mogelijk is met open source in iedereen iets anders belangrijk vindt. We kunnen je in dit document niet kennis laten maken met alle bestaande distributies, maar we kunnen wel een paar van de belangrijkste distributies voor je beschrijven.

De software die meegeleverd wordt met een distributie is allemaal voor gecompileerde software, je krijgt dus binairies net als bij Windows en Mac OS. Je hoeft de software niet meer zelf te compileren. Om die voorgecompileerde software te kunnen installeren is er een package manager nodig. Een stukje software dat de binaries allemaal op de juiste plek op de harddisk zet en er eventueel voor zorgt dat benodigde extra software, zoals libraries, ook geïnstalleerd worden. Verschillende distributies gebruiken verschillende package managers.

Een van de allereerste distributies was het Softlanding Linux System (SLS) door Peter MacDonald in 1992, deze Linux distributie bevatte, als eerste, een grafische interface. Het stond bekent om zijn buggy character en er ontstonden dan ook al snel opvolgers zoals Yggdrasil en Slackware van Patrick Volkerding. Slackware kwam in 1993 uit en is de oudste nog steeds bestaande distributie. Ook Debian is een afgeleide van SLS.

5.1 Debian

Debian is een op SLS gebaseerde distributie. Debian is een distributie die ooit is opgezet door Ian Murdoch, in 1993, en de distributie is vernoemd naar hem en zijn vrouw Debra; Deb-Ian ofwel Debian. Debian wordt ontwikkeld

zoals ook open source software wordt ontwikkeld, door vrijwilligers op een volledig open en transparante manier. Ze hebben zelfs hun eigen voorwaarden waaraan iedereen die mee ontwikkeld moet voldoen. Debian is de distributie met de meest beschikbare software pakketten.

De package manager van Debian is apt.

Debian vormt zelf weer de basis voor heel veel andere distributies. Ubuntu en Linux Mint zijn gebaseerd op Debian. Ubuntu en Linux Mint zijn vooral populair als desktop omgevingen, terwijl we Debian vaker terug vinden is in de serverruimte.

5.2 Red Hat

Red Hat is de grootste commerciële leverancier van Linux en tegenwoordig onderdeel van IBM. Red Hat wordt in het bedrijfsleven veel gebruikt. Omdat Red Hat commercieel is moet je betalen voor gebruik. Gelukkig voor ons is er een community based version genaamd CentOS die we gratis kunnen gebruiken. CentOS is dan ook de versie die we in deze serie gebruiken als het om de dekstop omgeving gaat.

Red Hat gebruikt de Red Hat Packet Manager rpm welke door veel distributies ook gebruikt wordt.

Ook Red Hat kent verschillende afgeleide systemen zoals CentOS en Scientific Linux (gemaakt door Fermilab, CERN, DESY en ETH Zuric).

5.3 OpenSuSE

De SuSE Linux distributie komt oorspronkelijk uit Duitsland en is een afgeleide van Slackware. De naam was in het begin een afkorting S.u.S.E. die stond voor: Software- und SystemEntwicklung. De eerste release was in 1994 en dat maakt SuSE tot een van de oudste commerciële distributies. In 2003 is SuSE gekocht door Novell dat toen nog groot was in de server-markt. Novell is degene geweest die OpenSUSE, de niet commerciële versie op de markt gezet heeft in 2005. Daarna is SuSE nog verscheidene malen overgenomen door ander bedrijven. Op dit moment is het in handen van EQT, een Europese investeringsmaatschappij.

SuSE is vooral bekend geworden door de YaST, Yet Another Setup Tool. De interface waarmee de Linux distributie op een eenvoudige manier beheerd en software geïnstalleerd kan worden. SuSE gebruikt rpm voor de installatie van pakketten. Voor de gebruiker is Zypper ontwikkeld als commando om software te installeren en deinstalleren.

Index