Ontwikkeling van besturingssystemen onder Linux

Niek Linnenbank

19 mei 2009

Linux wordt veelvuldig gebruikt als ontwikkelplatform voor allerlei soorten toepassingen. De stabiliteit, snelheid en gebruiksvriendelijkheid van Linux bieden een stevige basis voor ontwikkel- en programmeertaken van diverse systemen. Van simpele scripts to intelligente webapplicaties, met de juiste tools is het een fluitje van een cent. Om deze reden is Linux dan ook uitstekend geschikt voor ontwikkeling van nieuwe besturingssystemen.

Het bouwen van een besturingssysteem vergt een ongelofelijke berg tijd, energie, kennis en vooral geduld. De code die de programmeur schrijft, zal direct communiceren met de onderliggende hardware. In de praktijk betekent dat snelheid, leesbaarheid en vooral correctheid van de code uitermate belangrijk zijn voor het functioneren van de gehele machine. Een fout die in een gewone applicatie meestal herstelbaar is, kan in een besturingssysteem regelrecht catastrofaal zijn. Hierdoor vergt ontwikkeling van besturingssystemen het absolute uiterste qua kennis en creativiteit van de programmeur, en kan het jaren duren voordat een besturingssysteem stabiel genoeg is voor gebruik.

Waarom

Er zijn verschillende redenen om een nieuw besturingssysteem te ontwikkelen. De eerste en vaak onderschatte reden is dat het een unieke kans is om alles te leren over hardware, software, en computersystemen in het algemeen. In veel gevallen helpt het enorm bij het volledig begrijpen van een systeem als je het zelf een keer hebt gebouwd. Dit maakt het ontwikkelen van een besturingssysteem een avontuur op zich, omdat er altijd wat nieuws te leren valt. Het kan een leuke hobby zijn voor in de vrije tijd. Immers is Linux door Linus Torvalds ooit ook begonnen als hobby project, welk hij ontwikkelde op Minix.

Ten tweede kan er in een nieuw besturingssysteem worden geëxperimenteerd met nieuwe implementatiemethodes en -technieken. Resultaten en inzichten die hieruit behaald worden kunnen gebruikt worden in al bestaande besturingssystemen, of andersom. Zo werd Flask door NSA in de eerste plaats gebouwd in verschillende experimentele prototypes en een research besturingssysteem genaamd Fluke. Pas later werd de nieuwe technologie in Linux geïntegreerd, onder de naam SELinux.

Ten derde kunnen fouten uit het verleden, verouderde en overbodige software uitgefaseerd worden. Met een schoon en fris ontwerp van een nieuw systeem is compatibiliteit geen probleem meer, wat kan leiden tot een simpeler, sneller en flexibeler systeem.

Linus Torvalds ontwikkelde Linux in 1991 als hobby project op Minix.

Informatiebronnen

Het is een vrij lastige klus zijn om een begin te maken met een prototype besturingssysteem, waarop men verder kan bouwen. Allereerst is het van belang om een aantal goede informatiebronnen en voorbeelden bij de hand te hebben. Tegenwoordig zijn er op internet talloze websites met informatie, instructies, artikelen, hints, tips en forums over het ontwikkelen van een besturingssysteem. Zo bestaat er `http://www.osdev.org`, welk een uitgebreide

wiki heeft met nuttige informatie voor zowel beginners als experts. Beschikbare programmeertalen, hardwarearchitecturen met specificaties, theorie over de interne organisatie van besturingssystemen, en beschrijvingen van verscheidene ontwikkeltools zijn er allemaal te vinden. Boeken met theorieën en uitleg over besturingssystemen kunnen bovendien handig zijn voor programmeurs. Een echte aanrader is Operating Systems Design and Implementation (3e editie), door Andrew S. Tanenbaum.

Wanneer de ontwikkeling van een besturingssysteem verder vordert, zijn websites en boeken vaak niet voldoende meer. Veel moderne hardware bevat een door de fabrikant geïmplementeerde interface waar het besturingssysteem gebruik van kan maken. Hierbij is het meestal niet meteen duidelijk voor programmeurs, hoe deze aangestuurd kan worden, welk gedrag de hardware vertoont bij welke inputs, en welke outputs het besturingssysteem kan verwachten. Met een beetje geluk is de fabrikant zo vriendelijk geweest om de technische specificaties te publiceren, maar dat is niet altijd het geval. Een goed voorbeeld waarbij dit wel gebeurd is zijn de SCSI controllers van LSI, die daarover uitgebreide technische informatie publiekelijk beschikbaar heeft gemaakt.

Ontwerpkeuzes

Uiteraard moeten programmeurs van besturingssystemen verschillende ontwerpkeuzes maken, die van grote invloed zijn van de interne werking van een besturingssysteem. Een van de belangrijkste keuzes wordt gemaakt bij het kiezen van een architectuur. De bekendste vormen zijn een monolitische en microkernel architectuur. Bij een monolitisch ontwerp (zoals Linux) wordt het besturingssysteem in een groot programma gebouwd, waarbij alle functies binnen het programma elkaar ongehinderd kunnen aanroepen. Microkernels zijn zo klein mogelijk en implementeren besturingssysteem functionaliteit aan als aparte server processen. De belangrijkste voordelen van monolitische besturingssystemen is dat het ontwerp simpeler is en het minste overhead met zich meebrengt. Microkernels zijn veiliger en minder gevoelig voor fouten. Deze afweging heeft in het verleden tot interesstante discussies geleid tussen ontwikkelaars, met name de beroemde Torvalds vs. Tanenbaum discussie in comp.os.minix

Even zo belangrijk is de keuze van de programmeertaal voor het implementeren van het besturingssysteem. Niet elke taal is geschikt voor ontwikkeling van besturingssystemen. Daarnaast is de keuze van het type licentie bepalend of het wel of niet toegestaan is om broncode te delen, kopieren en/of te veranderen. Gelukkig heeft Linux de GPL licentie, waardoor iedereen van harte is uitgenodigd om de Linux broncode te lezen, kopiëren en te veranderen. Het is verstandig om ook van tevoren te bepalen welke hardware ondersteund gaat worden, afhankelijk van de gebruikers applicaties die gedraaid gaan worden. Meestal wordt ervoor gekozen om functionaliteit die gedefineerd is in standaarden te implementeren, zodat zoveel mogelijk hardware ondersteund kan worden. Enkele voorbeelden hiervan zijn PCI, USB en WiFi.

Testomgevingen

Een programmeur zonder een goede testomgeving is als een kapitein zonder schip. Het besturingssysteem moet ergens van op kunnen starten, ook wel booten of bootstrapping genoemd. Dat kan vanaf DVD, CDROM, harde schijf, USB sticks en zelfs vanaf het netwerk. Meestal wordt ervoor gekozen om een LiveCD te bouwen, waardoor het nieuwe besturingssysteem opgestart kan worden zonder het al geïnstalleerde systeem te beïnvloeden. Deze methode wordt bovendien ook toegepast door een groot aantal Linux distributies, om een nieuwe versie uit te proberen of te installeren. Onder Linux zijn er een aantal beschikbare tools waarmee een LiveCD op eenvoudige wijze te maken is, waaronder genisoimage, mkisofs en Ubuntu's remastersys.

Een tweede vraagstuk dat komt kijken bij het booten van een besturingssysteem is welke machines geschikt zijn. In veel gevallen zijn virtuele machines een goede eerste testomgeving. Onvermijdelijk komt het voor dat een nieuw besturingssysteem kritieke fouten bevat, waardoor opnieuw opstarten de enige uitweg is. Virtualizatie kan daarom zeer tijdsbesparend zijn, omdat het herstarten van virtuele machines aanzienlijk minder tijd kost dan het opnieuw opstarten van fysieke hardware. Bovendien kunnen ernstige fouten in experimentele besturingssystemen geen hardware beschadigen in virtuele machines. Linux heeft een aantal open source

virtualisatie oplossingen die gebruikt kunnen worden voor het booten van experimentele besturingssystemen, zoals Qemu, Xen, Virtualbox en Bochs.

Onmisbaar bij ontwikkeling van besturingssystemen is een goede debugger, waarmee het besturingssysteem tijdens uitvoering geïnspecteerd kan worden. Het oplossen van meestal onverklaarbaar gedrag van een experimenteel besturingssysteem kan voor de programmeur dagen zoek- en speurwerk betekenen, en soms blijft een probleem dan nog onvindbaar. Zonder debugger tast een programmeur van besturingssystemen dan letterlijk in het duister. Een zeer bekende debugger onder Linux is de GNU Debugger (gdb). Met simpele commando's kan het besturingssysteem stap voor stap worden doorlopen en geheugen en registers worden geïnspecteerd. Sommige virtuele machines, zoals Qemu en Bochs, ondersteunen een directe koppeling met gdb, tot grote vreugde van vele besturingssysteem ontwikkelaars.

Automatiseren

Net als bij gebruikers applicaties kan het ontwikkelproces van besturingssystemen worden geautomatiseerd. Een veel toegepaste vorm is het automatiseren van compiler commando's. Omdat besturingssystemen aan hoge prestatie eisen moet voldoen worden deze veelal geschreven in compilertalen, zoals C en C++. Linux heeft daarvoor de GNU C Compiler (gcc) en respectievelijk de GNU C++ Compiler (g++).
Het kan een tijdrovend karwei zijn om gcc en/of g++ telkens handmatig aan te roepen, en daarvoor zijn zogeheten makefiles bedacht. Daarin wordt beschreven welke compiler commando's nodig zijn om het hele besturingssysteem te bouwen. GNU Make kan onder Linux worden gebruikt voor het uitvoeren van makefiles.

GNU ontwikkelprogramma's zoals gcc, g++ en make zijn krachtige hulpmiddelen onder Linux.

Behalve het bouwproces kan ook het versiebeheer geautomatiseerd worden onder Linux. Besturingssystemen zijn groot, complex en worden vaak ontwikkeld door meerdere programmeurs. Geen versiebeheer leidt vrijwel zeker tot chaos. Er zijn verscheidende versiebeheer oplossingen onder Linux, bijvoorbeeld subversion, git, bazaar, mercurial en CVS. Elk van deze systemen heeft zijn voor- en nadelen. De belangrijkste afweging die men hierbij maakt is dat men een systeem kiest waarmee nog lange tijd doorontwikkeld kan worden, omdat besturingssystemen vaak projecten zijn van lange duur. Linux gebruikt voor haar eigen broncode git als versiebeheer systeem, tevens ook ontwikkeld door Linus Torvalds.

Een wat minder toegepaste vorm van automatisering in de ontwikkeling van besturingssystemen is het genereren van documentatie. De broncode wordt hierbij voorzien van speciale commentaarregels, die herkenbaar zijn voor een documentatie parser. Deze kan vervolgens een overzichtelijk HTML, PDF of LaTeX bestand genereren met alle informatie over gebruikte functies, variabelen en datastructuren in het systeem. In veel gebruikersapplicaties is dit de normale gang van zaken, en soms zelfs vereist. Bij besturingssystemen neigen programmeurs documentatie vaak apart handmatig te schrijven, in de vorm van boeken, artikelen of losse commentaarregels in de broncode. Doxygen maakt het voor ontwikkelaars onder Linux mogelijk om documentatie te genereren van broncode in talen als C, C++, Java, PHP en Python.

Voorbeeld

In mijn vrije tijd programmeer ik mijn eigen experimentele besturingssysteem, genaamd FreeNOS (Free Niek's Operating System). FreeNOS is een microkernel gemaakt in C++, en is op moment van schrijven in staat om met een LiveCD een simpele shell uit te voeren. Experimentele ondersteuning voor geheugenbeheer (virtueel en fysiek), scheduling van processen, filesystemen (ext2, procfs, tmpfs en vfs), hardware drivers (COM poort, keyboard, VGA) en applicatie libraries (libc, libposix) zijn beschikbaar. De complete broncode is gedocumenteerd in Doxygen en is beschikbaar via een subversion repository op `http://www.freenos.org/`. Linux heeft zich daarbij uitstekend bewezen als krachtig ontwikkel- en testplatform.