

Extra inspanning blijkt zeker de moeite waard

# Embedding Linux

Bij het ontwikkelen van embedded systemen is de keuze van het operating system (OS) een terugkerend onderwerp. Bij Technolution, een projectenbureau in de technische automatisering, zijn in de loop der tijd al veel verschillende opties overwogen, onderzocht en toegepast. Zo is er de beschikking over een zelf ontwikkeld OS, dat simpel en efficiënt is maar een beperkte functionaliteit heeft. De benodigde functionaliteit van embedded systemen neemt echter sterk toe, zodat zelf ontwikkelen niet meer rendabel is. Linux blijkt dan een prima alternatief te zijn.



Gwendolyn van der Linden en Olaf Peters

Het evalueren van een commercieel embedded operating system is geen eenvoudige taak. Zelfs na een gedegen onderzoek kunnen tijdens het gebruik alsnog tekortkomingen of zelfs bugs naar boven komen, die het succes van een project beïnvloeden. Voor het oplossen van de problemen is men meestal afhankelijk van de leverancier.

Daarnaast zijn er verschillende open source operating systemen beschikbaar, variërend van kleine hobby-projecten tot grote onderzoeksprojecten waarin nieuwe OS-concepten worden toegepast. Toen zich bij het bedrijf Technolution de mogelijkheid voordeed om meer ervaring op te doen met open source software, is deze met beide handen aangegrepen. Het is toegepast in een tweetal lopende projecten, en zal zeker gebruikt worden

in toekomstige projecten. In dit artikel worden de uitgangspunten bij en algemene ervaringen met embedded Linux bij Technolution beschreven, en er worden twee projecten uitgelicht: een klimaatregelsysteem en een meer algemeen ontwikkelplatform.

### Eisen aan het operating system

Voordat er een keuze gemaakt wordt voor een bepaald operating system is het belangrijk om de eisen op een rijtje te zetten. De volgende lijst is samengesteld door naar de eisen van een aantal lopende projecten te kijken: Ondersteuning van meerdere platformen, met name de meer krachtige embedded processoren zoals Intel x86 en Arm-families; verschillende projecten vragen verschillende hardware-oplossingen, maar het is een stuk

eenvoudiger als grotendeels dezelfde software gebruikt kan worden.

Connectivity: Ethernet, USB, Wireless Ethernet, Bluetooth, GSM, TCP/IP, PPP.

Het gebruik van standaarden (POSIX, W3C, RFC's ...) is wenselijk, omdat het de compatibiliteit, overdraagbaarheid en het hergebruik van software verhoogt.

Basiscomponenten en -applicaties moeten al beschikbaar zijn, zoals een HTTP-server, window-manager, web-browser, SMS handling, sound support en MP3-decoders. Er zijn hiervan al vele versies beschikbaar en het zelf ontwikkelen ervan heeft weinig toegevoegde waarde. Een korte time-to-market is cruciaal in vele projecten en het ontwikkelen van embedded systemen is daarop geen uitzondering.

Er moet voldoende support zijn voor extern advies, voor als zich problemen of vragen voordoen.

Continue doorontwikkeling, zoals het ondersteunen van nieuwe hardware.

De sourcecode moet beschikbaar zijn. De ervaring heeft geleerd dat dit cruciaal is bij het opsporen van bugs en het bestuderen van het gedrag van de software.

Royalty-free. Zeker bij kleine, relatief goedkope systemen kunnen te betalen royalty's een aanzienlijk deel van de totale prijs per product vormen.

Real-time extensies. Embedded systemen worden regelmatig ingezet voor tijdkritische taken, wat real-time faciliteiten vereist van het operating systeem.

### De keuze

Gezien de bovengenoemde systeemeisen en eerder opgedane ervaring viel de keuze al snel op Linux, en wel om de volgende redenen:

- Linux ondersteunt vele platformen, zoals de 32-bits- en 64-bits-processoren in de x86, IA-64, Arm (StrongARM, Arm7, Arm9), Alpha, PPC, en M68K-families. De standaard Linux-kernel ondersteunt geen 16-bits-processoren of processoren zonder MMU (memory management unit), maar dat is voor de twee besproken projecten geen probleem.
- De source code is beschikbaar onder een open source licentie en het geheel is royalty-free (zie [www.opensource.org](http://www.opensource.org) voor meer informatie over het begrip open source).
- Linux kent een groot aantal device drivers, zoals voor Ethernet, USB, en FireWire, en bijbehorende protocolimplementaties, zoals TCP/IP en NFS.
- Er is een continue en relatief snelle verbetering en uitbreiding van de functionaliteit. Dit is mogelijk dankzij de vele gebruikers die helpen bij het testen en de vele ontwikkelaars die aanpassingen en uitbreidingen maken. Nieuwe kernelversies verschijnen bijna dagelijks.
- Er zijn veel softwarecomponenten en -applicaties beschikbaar en veel daarvan zijn open source. Een

## Royalty's op commerciële systemen kunnen een groot deel van de prijs vormen

aantal is specifiek geschikt voor embedded systemen. Met het krachtiger worden van embedded platformen kan direct gebruikgemaakt worden van de vele applicaties die bij de standaard Linux-distributies worden meegeleverd.

- Er is een aantal commerciële distributies voor embedded Linux, waaronder Lineo, BlueCat, Red Hat, HardHat en LynuxWorks, voor onder meer PC/104, CompactPCI en enkele ARM targets. Een aantal leveranciers van embedded hardware levert ook aangepaste versies van Linux, die direct op hun hardware draaien.

## Een aantal embedded hardware-leveranciers heeft ook eigen aangepaste Linux-versies beschikbaar

- Linux geeft de keuze uit verschillende kernelseries: de 2.0-, 2.2- en 2.4-series. De oudere zijn compacter, de nieuwere hebben meer functionaliteit.
- Nieuwe kernelfuncties of -bugfixes worden geport naar oudere versies.
- De kernel is configureerbaar, zodat ongebruikte functionaliteit kan worden weggelaten en een kleinere kernel wordt gerealiseerd.

### Linux-architectuur

Een aantal aspecten van de architectuur van Linux is van belang voor het ontwikkelen van embedded applicaties. Met behulp van de MMU-hardware zijn processen onderling gescheiden en gescheiden van de kernel. Dit verhoogt de robuustheid, omdat een procescrash beperkt blijft tot dat proces. De Linux-kernel zelf is echter één geheel (monolithisch), zodat een device driver in de kernel wel het hele systeem kan laten crashen.

De kernel biedt services voor de applicaties, met name memory management, proces scheduling, een file-systeem en networking. Het leeuwendeel van de kernelcode is in C geschreven, met een dunne platformafhankelijke laag in assembly. De kernel is configureerbaar wat betreft de ondersteunde hardware (CPU, bussystemen, devices) en services. De kernel kan geconfigureerd worden met een grafische tool.

Standaard Linux-distributies omvatten alles wat nodig is om een kernel opnieuw te configureren, compileren en installeren. Het ontwikkelen van embedded Linux-sys-

teem- en applicatiesoftware kan met een standaard Linux-systeem. De host en embedded target voelen daardoor vrijwel identiek aan, wat het ontwikkelen een stuk makkelijker maakt.

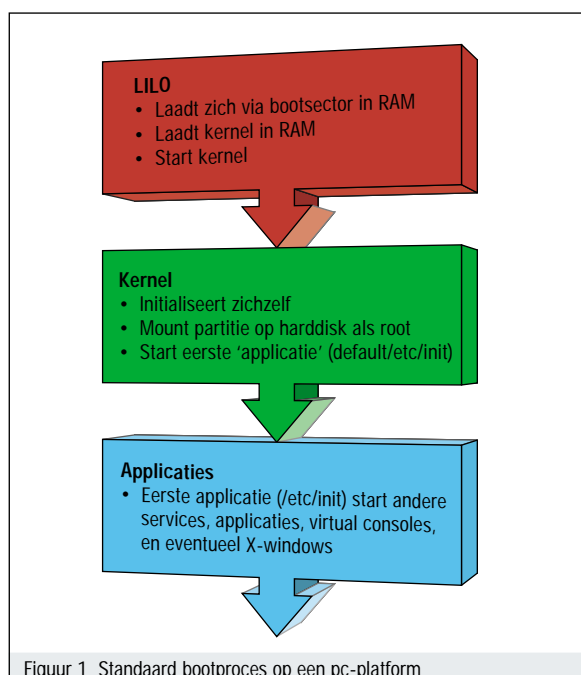
Linux-applicaties draaien als proces, dus executeren in een eigen, afgeschermd omgeving. Ze kunnen geen interrupts ontvangen en hebben beperkt toegang tot de hardware, door gebruik te maken van de kernel-services. Applicaties worden opgeslagen in het file-systeem.

## Een bug in een kernel device driver kan het hele systeem laten crashen

Zoals bij elk Unix-achtig operating system worden vrijwel alle systeemcomponenten benaderd via het file-systeem. Elke Linux-installatie heeft ten minste één file-systeem, het zogeheten root file-systeem. Een file-systeem kan zijn opgeslagen op een floppy, harde schijf, flashgeheugen, RAM-geheugen, of op het netwerk (via BOOTP). Vooral de laatste drie mogelijkheden komen zeer goed van pas bij embedded systemen.

### Booten

De normale procedure voor het booten van een desktop-Linux-systeem (zie figuur 1) is het laden van de kernel in het geheugen met een boot loader (vaak LILO) en het starten van de kernel. De Linux-kernel initialiseert zich,



mount het root file-systeem en start de applicatie /sbin/init. Onder normale omstandigheden start /sbin/init dan weer de rest van de OS-services die als proces draaien.

### Embedding Linux

Het belangrijkste verschil met niet-embedded systemen is dat er bij embedded systemen een beperkte hardware-reconfiguratie is en er minder services nodig zijn. In hoofdzaak blijft een normale Linux-set-up mogelijk. De scheiding tussen kernel en applicatie blijft bestaan, maar er is vaak minder geheugen (RAM, ROM, en/of Flash), meestal geen harddisk en meestal geen videokaart voor console output. In plaats van een harddisk wordt vaak gebruikgemaakt van een Flash-disk. Voor console output wordt meestal een seriële poort gebruikt, waarmee het embedded systeem via een ander systeem benaderd kan worden.

Er zijn verschillende opties voor het installeren van embedded Linux. Afhankelijk van de gekozen hardware is het soms mogelijk om een Software Development Kit (SDK) te kopen bij een embedded Linux-leverancier. Een andere optie is om alles zelf te doen met tools en software die vanaf internet gedownload kunnen worden. Binnen Technolution hebben we voor de laatste optie gekozen, zodat we ervan kunnen leren en het volledig naar wens kunnen aanpassen. Wel hebben we een commerciële SDK (LynuxWorks) gebruikt als referentie.

Het recept voor embedded Linux wordt dan:

Neem een minimale kernelconfiguratie. Zorg voor de benodigde device drivers. De standaard Linux-kernel komt al met een indrukwekkende lijst device drivers, en er zijn via internet veel speciale drivers te vinden. Zelfs als er een device driver zelf geschreven moet worden, kan een bestaande gelijksoortige driver vaak als uitgangspunt dienen. Vervang de Linux Loader (LILO) met een speciaal geschreven embedded boot loader toegesneden op de gebruikte embedded hardware. Bij zelf ontwikkelde hardware moet veelal ook de boot loader zelf geschreven worden.

### Klimaatregelsysteem met embedded 486-board

Het eerste project dat in meer detail wordt besproken is de modernisering van een klimaatregelsysteem. Het nieuwe systeem moet een betere prestatie leveren en meer netwerkfaciliteiten hebben, zoals remote access. De hardware bestaat uit een industrieel embedded AMD486-systeem met 8Mbytes RAM, 8Mbytes Flash, een Ethernet controller en drie seriële poorten (zie foto pag. 34). Het board is pc-compatibel, heeft een power-down-mechanisme, maar geen videokaart. De Ethernet

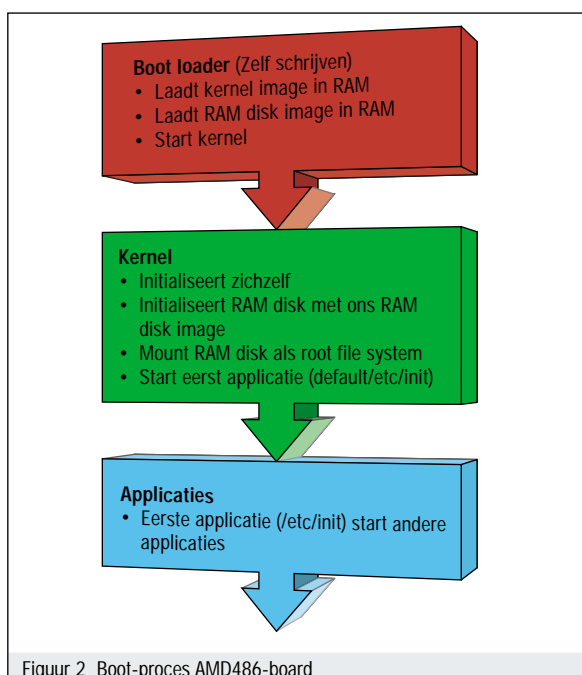
controller en het power-down-mechanisme worden niet door Linux ondersteund.

Als eerste is de kernel source code gedownload van [www.kernel.org](http://www.kernel.org) en bestudeerd, met name het boot-proces. Het opzetten van een ontwikkelomgeving was relatief eenvoudig. Een pc met standaard Red Hat 6.2-distributie voldoet prima en omdat de target en host beide dezelfde processorarchitectuur hebben, is de geïnstalleerde gcc-compiler direct te gebruiken en hoeft er geen cross-compiler te worden geïnstalleerd.

Vervolgens is de geschiktste kernelconfiguratie bepaald en met de normale configuratietool ingesteld:

- Geen videokaart.
- Console over seriële poort.
- Laden van RAM-disk via image in RAM (daar neergezet door de boot loader).
- RAM-disk support (initrd).
- Root file-systeem support (minixfs).
- Netwerk-file-systeem support (NFS).

De boot loader, Ethernet device driver en power-down handler zijn zelf geschreven. Vooral het schrijven van de Ethernet device driver vergt een redelijke inspanning. Indien mogelijk is het dus raadzaam om een reeds ondersteunde Ethernet controller te gebruiken. De boot loader en power-down handler konden gebaseerd worden op reeds bestaande software uit andere (niet-Linux-) projecten. Figuur 2 illustreert het boot-proces en figuur 3 de benodigde geheugen-lay-out hiervoor.



Het opzetten van een root file-systeem is een behoorlijke klus, maar er is wel veel informatie over te vinden. Vooral Linux From Scratch (LFS) ([www.linuxfromscratch.org](http://www.linuxfromscratch.org)) biedt waardevolle informatie. Door gebruik te maken van het loopback file-systeem kan een file worden gemount als een file-systeem, om later weer als binaire image te worden gekopieerd naar het Flash-geheugen. Met een zelfgeschreven shell script worden alle benodigde files uit /etc, /sbin, /dev en /bin naar dit file-systeem gekopieerd, inclusief bash shell, shell utilities, inetd, telnet server, HTTP-server, en NFS-client. Voor het gemak zijn deze direct in binaire vorm van de standaard Red Hat-distributie gekopieerd, aangezien deze geschikt zijn voor een AMD486.

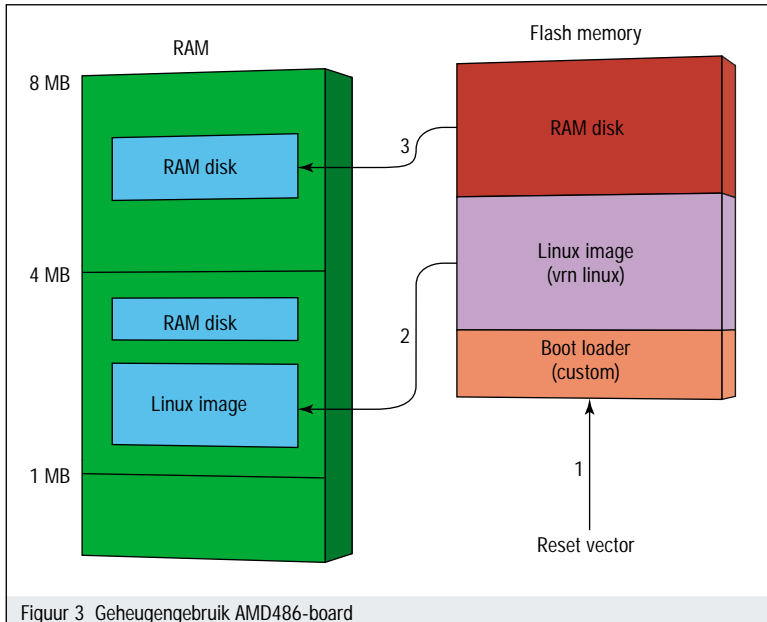
## Voor zelf ontwikkelde hardware moet veelal ook de boot loader zelf worden geschreven

Een belangrijk nadeel van deze aanpak is dat de benodigde geheugenruimte vrij fors is, ook al worden de executables en libraries van debug-informatie gestript met behulp van strip. Met name de standaard libraries zijn groot en nemen met elke nieuwe versie in omvang toe. Een alternatief is het gebruik van speciale embedded utilities die compacte versies van de noodzakelijke applicaties en bibliotheken bundelen, zoals BusyBox ([busybox.lineo.com](http://busybox.lineo.com)). Het installeren daarvan bleek echter niet eenvoudig en alleen zinvol als er echt ruimtegebrek is.

Het run-time geheugengebruik was in totaal 2 Mbytes, waarin zowel de kernel (gedecomprimeerd) als een GoAhead HTTP-server draaide. Het uitbreiden van de functionaliteit vereist al gauw meerdere Mbytes.

### Cirrus-Logic EP7211-evaluatieboard

Het tweede project dat wordt uitgelicht is een onderzoek op basis van een Cirrus-Logic-evaluatieboard, als voorbereiding op het porten van Linux voor een zelf ontwikkeld board. Dit board zal onder meer gebruikt worden in een intelligente smartcard terminal, voor het ontwikkelen van een nieuwe generatie smartcard-applicaties. Het board heeft een Cirrus-Logic EP7211 CPU, die is gebaseerd op een ARM7 processor core. Verder heeft het board 16Mbytes RAM, 8Mbytes Flash, een Ethernet controller, twee seriële poorten, een 5x5 matrix-keyboard en een 2x20 character-LCD. De 16 Mbytes is opgesplitst in twee niet-aaneengesloten 8 Mbytes-gebieden.



Figuur 3 Geheugengebruik AMD486-board

De gebruikte Ethernet controller wordt, net als de niet-aaneengesloten 8 Mbytes-gebieden, niet standaard ondersteund door de 2.2.x Linux-kernels. Cirrus-Logic heeft echter een patch voor een Linux-kernelversie die beide problemen oplost. Een belangrijk verschil met het embedded AMD486-board is dat de target-processor-architectuur nu verschilt van de host (nog steeds een pc met Red Hat). Er moet dus een cross-compiler gebruikt worden en de componenten van Red Hat kunnen niet gebruikt worden op de target.

## Het porten van Linux vraagt veel kennis

De volgende stappen waren nodig om Linux te kunnen gebruiken op het EP7211-board:

- Downloaden van GNU tool chain sourcecode en configureren/compileren als cross-compiler voor de ARM7.
- Downloaden van 2.2.1 kernel patch van Cirrus-Logic.
- Zelf schrijven van boot loader.
- Downloaden en cross-compileren van de benodigde applicaties (zie Linux From Scratch).
- Opzetten van een root file-systeem, als bij het AMD486-board.

### Conclusie

In beide beschreven projecten is Linux met succes toegepast en is veel ervaring opgedaan met het embedden van Linux. Net als bij andere embedded operating systemen vereist het gebruik echter wel de nodige inspanning.

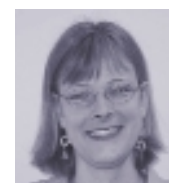
Het porten van een Linux-kernel naar nieuwe hardware is zeker mogelijk, maar het gebruik van niet-ondersteunde hardware vraagt enige inspanning omdat het zelf ontwikkelen van device drivers kennis-intensief is. Zo is het schrijven van een device driver voor een Ethernet-chip zeer leerzaam, maar meestal eenvoudig te vermijden gezien de grote hoeveelheid chips die wél ondersteund worden. Het advies is dan ook om zo veel mogelijk ondersteunde hardware te gebruiken.

Het geheugengebruik van zowel de kernel als de applicaties is aanzienlijk. Gelukkig staan ook op dit gebied de ontwikkelingen niet stil. Er is een continue aanwas van compacte componenten en applicaties uit embedded ontwikkelingen en toepassingen voor PDA's. De standaard kernel heeft minimaal 32 bits-processoren met een MMU nodig, maar dit soort processoren wordt steeds meer gemeengoed in embedded toepassingen. Daarnaast zijn speciale Linux kernel ports zoals  $\mu$ Clinux ([www.uclinux.org](http://www.uclinux.org)) en ELKS ([www.elks.ecs.soton.ac.uk](http://www.elks.ecs.soton.ac.uk)) geschikt voor kleinere processoren.

Het porten van Linux vraagt veel kennis. Een nadeel hierbij is dat de kernel-sourcecode onvoldoende is gedocumenteerd en lang niet altijd voor zichzelf spreekt. Meerdere boeken, vele on line referenties en het doorzoeken van Linux-nieuwsgroepen zijn nodig om het benodigde inzicht te krijgen. Een goed startpunt hiervoor is [www.linuxdoc.org](http://www.linuxdoc.org).

Een belangrijk voordeel is dat de kernel optimaal aan de hardware kan worden aangepast. Daarnaast bieden de kernel en bijbehorende systeemsoftware vele services, die ook in embedded systemen gebruikt kunnen worden. Verder maakt de grote hoeveelheid beschikbare (eenvoudig te porten) applicatiesoftware voor Linux het mogelijk om snel functionaliteit toe te voegen, als eenmaal de systeemsoftware is geport.

Al met al is Linux zeer geschikt voor embedded toepassingen. Technolution verwacht dan ook in de toekomst Linux – of andere open source software – in meer projecten toe te passen.



Gwendolyn van der Linden is consultant bij Technolution bv,

en Olaf Peters is senior consultant bij Technolution bv. Technolution is een innovatief projectbureau in de technische Automatisering. Het bureau adviseert over, specificeert en ontwikkelt hardware en software.