Snelle Linux-overstap begint bij toepassing

Linux combineert de kracht van Unix met de vrijheid van open source. Het besturingssysteem is daardoor erg geschikt voor moderne, krachtige embedded systemen. Toch is overstappen op Linux niet eenvoudig. Vele valkuilen kunnen de kostenvoordelen van open source tenietdoen. Albert Mietus van PTS vertelt hoe u de kosten van de overstap naar Linux in de hand kunt houden.

ALBERT MIETUS

Er komen steeds meer succesvolle embedded Linux-systemen op de markt. Zo is het mogelijk om files te vermijden dankzij verkeersregulatiesystemen gebaseerd op dit open-source besturingssysteem en om de weg te vinden met een populair, op Linux gebaseerd navigatiesysteem. Thuis neemt de digitale recorder, opgebouwd met Linux, een film op. Die kunnen we via het internet – ook daar veel Linux – op afstand programmeren. Kortom, iedereen is omgeven door Linux. Niet door de studentikoze versie van jaren geleden, maar ingebouwd in hoogwaardige, commercieel verkrijgbare producten.

Linux is blijkbaar geschikt voor embedded systemen. Alle technische problemen lijken opgelost. Waarom ondervinden ontwikkeltrajecten dan toch nog steeds grote problemen als ze Linux gaan gebruiken? In algemene zin constateer ik dat ontwikkelaars er te laat achter komen dat dit besturingssysteem heel anders werkt dan de traditionele OS'en die ontwikkelaars gewend zijn. De architectuur is anders en de begrippen zijn of betekenen bij Linux wat anders. Bijna per definitie ontbreekt het de eerste keer aan ervaring.

Beginnen met de toepassing

De meest voor de hand liggende manier om embedded Linux te gebruiken, is niet altijd de verstandigste. Vaak starten ontwikkelaars door het besturingssysteem eerst uit te proberen op het target. Dit is veelal zelfontwikkelde of specifieke hardware. Linux ondersteunt dat niet *out of the box*. Uiteindelijk zal het lukken om het OS daarop te draaien. Het kost alleen meer tijd dan verwacht. Hierdoor begint het poorten van de eigenlijke toepassing pas laat, terwijl deze tweede stap veel belangrijker is. Immers, het zijn de applicaties die het geld straks binnenbrengen, het OS is voor klanten onbelangrijk.

Het poorten van de toepassing naar Linux kan zowel heel eenvoudig als heel complex uitvallen. Dit is afhankelijk van kleine details. Een strikt toegepaste softwarearchitectuur is vaak een voordeel. Maar soms passen het ontwerp en de Linux-grondbeginselen (zie kader 'Linux-grondbeginselen') gewoon niet goed. Bijvoorbeeld als er weinig abstractie is tussen applicatie en hardware. Dat is heel gebruikelijk in traditionele embedded software, maar het kan een showstopper zijn voor Linux.

Soms eindigt een project daar. Linux draait prima op de hardware, maar zonder toepassing. Het gevolg: een team boordevol frustraties over Linux. Het is dan de vraag of de gekozen volgorde een verstandige was.

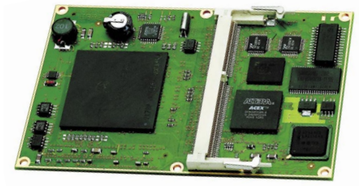
Linux-grondbeginselen

Linux gebruikt andere concepten dan traditionele OS'en. Een opsomming van de belangrijkste verschillen voor embedded programmeurs:

- **Linux hanteert strikte grenzen tussen verschillende delen: tussen processen onderling, maar ook tussen hardware, besturingssysteem en toepassingen.**
- **Linux is een generiek OS, dat geschikt is voor een breed scala aan computers. Programma's gebruiken altijd een abstract 'device': een printer, een harde schijf, een beeldscherm. Hierdoor werkt software op alle hardware, maar is het gebruiken van specifieke hardware-eigenschappen niet eenvoudig.**
- **Linux-systemen zijn modulair opgebouwd, waarbij er veel alternatieve (concurrerende) componenten zijn. (Systeem)ontwikkelaars mogen en moeten zelf kiezen.**
- **De ontwikkeling van Linux is in handen van onafhankelijke bedrijven en individuen. Goede ideeën nemen ze vaak over, maar een algeheel erkende organisatie die de richting bepaalt, ontbreekt.**
- **Iedereen heeft het recht om Linux te gebruiken, te veranderen en door te geven, maar niet het recht om die rechten te beperken. Dit is de kern van de GPL.**

Verstandiger is het om te beginnen met de belangrijkste en meest risicovolle delen van een project: het poorten van de toepassing. Dankzij de Linux-grondbeginselen is dat heel goed mogelijk. Omdat Linux de applicatie en de hardware strikt van elkaar scheidt, is het niet nodig om te beschikken over de uiteindelijke hardware om met de toepassing te kunnen spelen.

Net als bij Unix, zijn Linux-applicaties hardwareonafhankelijk (zie kader 'De



wereld is een file'). Ze functioneren min of meer op elk platform. Slechts een zeer beperkt deel van de code is geoptimaliseerd voor specifieke hardware. Dit geldt zowel voor normale Linux-systemen als voor ingebedde varianten. Een embedded toepassing is snel over te zetten naar Linux, gewoon op een standaard pc . Pas in tweede instantie is het nodig om Linux te laten draaien op het echte target. Deze 'omgekeerde' volgorde pakt de grootste risico's direct aan en levert snel bruikbare informatie over de projectkosten. Soms al na één dag.

De aanpak

Meestal is een toepassing zonder al te veel moeite over te zetten naar Linux. Dat betekent dat de applicatie de abstracte hardware-interfaces van het besturingssysteem kan gebruiken. Daardoor is het ook mogelijk om tijdelijk andere hardware te gebruiken, bijvoorbeeld een oude pc. Een 386 is vaak al prima.

De computer met ingebedde Linux-variant kunnen we inzetten als 'embedded pc'. Op dag twee kunnen we al beginnen met het poorten van onze toepassing naar dit platform. Typisch gaat dat door te proberen de applicatie met GCC te compileren op dit platform. We mogen niet verwachten dat dit meteen werkt, maar het levert wel veel informatie op over de moeite die het gaat kosten om het geheel werkend te krijgen.

Vaak kunnen we in korte tijd een draaiende toepassing op Linux maken met hier een daar wat *stubs* (niet-functionele dummy-interfaces). Omdat zowel de hardware als de software niet volledig zijn, zal dat systeem niet functioneren, maar het levert wel gedetailleerde informatie waarop we een planning kunnen baseren.

Met deze korte experimentele onderzoekjes leggen we een basis om verder te gaan. Zo hebben we de grootste risico's in kaart gebracht. Daarnaast is er een technische baseline om op voort te borduren. Impliciet is het project ook opgedeeld in hapklare deeltaken, zoals het invullen van elke stub en drivers schrijven voor overige hardware. Veel van die taken zijn onafhankelijk van elkaar uit te voeren. Ook het testen is vaak eenvoudig, dankzij de referentie in de vorm van de embedded pc.

Terwijl de toepassing stap voor stap gaat werken op Linux, kunnen we het target voorzien van het open-source besturingssysteem. Dat kan zelfs gerichter, omdat we meer inzicht hebben in wat de applicatie verwacht van het OS. Als we op dat systeem bijvoorbeeld drivers testen, kunnen we gebruikmaken van eerder opgedane ervaringen.

Als laatste moeten we alle delen integreren. De hardware, Linux, de drivers en de toepassing moeten samenkomen in één embedded systeem. Hoewel dat vaak een moeilijke stap is, kan ook hier de kracht van Linux helpen. Er zijn heel veel Linux-ontwikkeltools beschikbaar

die ook bruikbaar zijn op een ingebed systeem. Zeker in het lab.

Belangrijk bij deze verbeterde aanpak is de vroegtijdige inzet van Linux. Liefst een versie die het gehele project is te gebruiken, dus eentje die geschikt is voor embedded. Omdat Linux onder de GNU-licentievoorwaarden (GPL) valt, hebben we automatisch het recht om de veranderingen te maken die tijdens het project nodig blijken. Dat geldt echter niet voor alle commerciële gereedschappen, maar er zijn voldoende open-source tools beschikbaar.

Duidelijk is dat Linux-obstakels zijn te omzeilen door snel te starten. Installeer bijvoorbeeld eens een geschikte embedded Linux-versie op een pc en probeer de toepassing daar eerst werkend op te krijgen. Maak verder gebruik van de ervaring die er al is. Bijna alles is beschikbaar, je moet het alleen weten te vinden.

Albert Mietus is consultant bij PTS Software en de architect van Embedded Quickstart Linux (EQSL), PTS' huisselectie op het gebied van embedded Linux.

De wereld is een file

In tegenstelling tot veel traditionele OS'en houdt Linux er een strikte en conceptuele scheiding op na tussen hardware en applicaties. Deze 'device-interface' is in hoge mate abstract. De programmeur heeft geen weet van typische hardware-eigenschappen. Hij gebruikt vooral de generieke schrijf-naar- en lees-van-interfaces, dezelfde als voor het werken met bestanden. Ook de aansturing van beeldscherm, muis en toetsenbord verloopt via die filefuncties, evenals de communicatie met flashgeheugens, harde schijven, netwerken, seriële verbindingen en bijna alle andere hardware. Unix ziet alles als een file. Voor embedded-ontwikkelaars is dat even wennen. Het beste is dan dat de kennismaking zo snel mogelijk gebeurt, want als de abstracte interface echt niet te gebruiken is, is het maar de vraag of Linux een goed(kop)e keuze is. Mocht dat de conclusie zijn, dan is het onnodig om dit besturingssysteem aan de praat te krijgen op het target.