Writing your own device driver
a.k.a. klus je eigen kernel module

Otto Visser

05-11-2007

1 Inleiding

Bij Linux zitten in principe alle hardware drivers in de kernel. Deze worden meegebakken in de kernel tijdens het compileren. Er is echter ook de mogelijkheid om achteraf drivers toe te voegen aan de kernel zonder deze te hercompileren. Zeker als de code voor de driver nog niet helemaal af is of je weet niet goed hoe je een nieuwe kernel moet maken (welke opties wel/niet?) is het erg handig om een kernel module te maken(hebben) van je driver. Een eenmaal succesvol gecompileerde kernel module kan in de draaiende kernel toegevoegd worden. In dit geval gaat er dus een loadable kernel module geschreven worden voor de accelerometer chip.

2 De grote verschillen met user space programmeren

Je krijgt nu twee delen te programmeren: een kernel module en een user space programma dat met deze kernel module gaat communiceren. Het user space deel is het makkelijkst, dus daar beginnen we maar even mee. In Linux "praat" je tegen device drivers met behulp van de files in de */dev* directory. Voor de sensor is uiteraard op dit moment nog geen file beschikbaar, dus die zal je zelf moeten maken.[1] Denk er wel aan dat je niet zomaar wat kan pakken.[2] Deze file kan je dan openen om je data uit te halen als je kernel module eenmaal werkt.

Het kernel deel is uiteraard wat lastiger. Het grootste deel vd code die je gebruikt hebt om de sensor in user space uit te lezen (middels user_gpio.c) kan je hierin gebruiken, maar vergis je niet: programmeren voor de kernel is anders dan programmeren in user space. In een normaal programma heb je de beschikking over de C-library (libc) met daarin nuttige functies als *printf*, *malloc*/*free*, etc. In de kernel moet je het doen met wat anderen aan functies in de kernel hebben gestopt. Meer informatie vind je op het web.[3] Je gaat een kernel module schrijven voor een kernel versie 2.6; als je documentatie vindt die bedoeld is voor een eerdere versie (2.4), kijk dan even of het in 2.6 nog steeds hetzelfde werkt![4] Hou er rekening mee dat je kernel module als onderdeel van de kernel draait. Dit betekent dus oa. dat van je verwacht wordt dat je aardig bent voor de andere delen (neem niet alle CPU in beslag) en dat als jouw deel iets fout doet, de hele kernel met jouw deel ten onder gaat. Iets anders waar je rekening mee moet houden: je kan niet zomaar een pointer naar een variabele uitdelen aan je user space programma; die mag namelijk niet aan kernel geheugen komen. Als laatste: aangezien

[1] run *man mknod* voor meer info en/of gebruik het setup script.

[2] lees /usr/src/linux/Documentation/devices.txt op een linux systeem, zie het setup script

[3] `http://www.tldp.org/LDP/lkmpg/`

[4] `http://lwn.net/Articles/driver-porting/`

je niet een op zichzelf staand programma maakt, moet je ook je Makefile aanpassen zodat je kernel module niet gelinked wordt. Dit kan bereikt worden met de -c optie voor gcc (zie ook de bijgeleverde Makefile).

Als je kernel module eenmaal af is, moet je hem natuurlijk ook toevoegen aan de kernel op de LART. De commando's die hiervoor van belang zijn zijn *insmod*, *rmmod* en *lsmod*. Lees dus de manpages hierover als je deze commando's nog niet kende.

3 Do's and Dont's

3.1 Do's

- Eerst lezen voordat je uitprobeert. Een klein foutje levert algauw een kernel panic/segfault op en dan kan je herstarten en dergelijke.[5]
- Wees zuinig met de resources, je deelt ze tenslotte met de rest van de kernel en je eigen user space programma moet ook nog wat CPU krijgen.
- maak je kernel module onder de GPL license, dan krijg je geen waarschuwingen over tainted kernels bij het inserten van de module.
- bedenk of er meerdere programma's tegelijk met je module mogen praten en hoe je dat wilt aanpakken/voorkomen.

3.2 Dont's

- include geen standaard header files zoals *stdio.h*; de kernel heeft niet de beschikking over libc.
- Stop niet teveel logica in je kernel module; een device driver is iets low-levels.
- als je user space deel niet draait, hoeft de kernel niet nog steeds je CPU bezig te houden.

[5] `http://www.xml.com/ldd/chapter/book/`