IT-Universitetet i København 2. september 2014

Bygning af en Linux kerne

*** Selve kompileringen af kerne-kildekoden tager ganske lang tid. Det er derfor en god ide at planlægge din tid sådan, at du evt. kan lave andre ting mens der bygges. ***

Baggrund

Kernen i Linux har bl.a. som opgave at være et abstraktionslag til hardwaren.

Når en Linux-distributør laver en CD med Linux har kernen brug for drivere, der styrer de forskellige enheder, til alle tænkelige kombinationer af hardware. Men eftersom det ikke er alle drivere, der er nødvendige i en given computer, lægges driverne over i moduler. Disse kan så hentes ind efter behov. Når det kun er de nødvendige moduler, der hentes ind, kan størrelsen af kernen holdes nede.

En Linux kerne kan bygges af kildekoden af hvem som helst. Alt der behøves er en compiler, f.eks. `gcc`, programmet `make`, GNUs programpakke `binutils`, kerne-kildekoden samt lidt tålmodighed. At bygge en kerne tager tid og der kræves ofte flere forsøg inden kernen fungerer som ønsket.

Inden kernen kan oversættes skal den konfigureres. Den gældende konfiguration skal gemmes i roden af kildekode-folderen under navnet `.config`.

Konfigurationen sker ved hjælp af `make` kommandoen. Reglerne til `make` er gemt i filen `Makefile` i roden af kildekode-folderen. Med et kommandolinje-argument til `make` kan du vælge om du vil konfigurere i tekst-tilstand, tekst-tilstand med grafik eller X11-tilstand. I nedenstående opgave antages at du bruger tekst-tilstand med grafik (altså grafisk i shell vinduet, med valg fra tastaturet).

Bygning og installation af din egen Linux kerne

Bemærk, at den følgende vejledning er for 3.13.x kerner. Hvis du vil bygge en nyere kerne er fremgangsmåden ikke nødvendigvis den samme og du bør søge vejledninger på nettet (se f.eks. links under litteratur på ugeseddel 2). Det antages her desuden, at versionsnumrene `3.13.0` og `3.13.0-32` for kerne-kildekoden bruges. Disse skal du skal erstatte med dine egne versionsnumre.

0. Installere nødvendige pakker.

Begynd med at installere følgende pakker på din virtuelle maskine:

```
sudo apt-get install linux-libc-dev fakeroot kernel-wedge build-essential
```

1. Downloade kildekoden.

Jeg foreslår her, at du vælger en kerne-kildekode, der er den samme (elle kun er få versioner nyere) end den der allerede kører på din virtuelle maskine. Ellers vil der være rigtig mange nye ting som du skal konfigurere inden kernen bygges og afprøves.

Find ud af hvilken version af kerne du kører, f.eks. ved at skrive `uname -a`:

```
root@bosc:~# uname -a
Linux bosc 2.6.24-26-server #1 SMP Tue Dec 1 18:26:43 UTC 2009 x86_64 GNU/Linux
```

Download herefter en kildekoden ved først at browse efter den version du ønsker på `http://www.kernel.org/pub/linux/kernel/` og derefter f.eks. bruge `wget`:

```
wget www.kernel.org/pub/linux/kernel/v3.x/linux-3.13.tar.gz
```

Bemærk, at overførslen godt kan tage noget tid (ofte vil det være hurtigere at downloade filen til jeres bærbare og overføre den til den virtuelle maskine med `scp`).

Nu pakkes kerne-kildekoden ud, f.eks. i en ny folder `uge2`:

```
mkdir ~/uge2; tar -xvf linux-3.13.tar.gz --directory ~/uge2
```

2. Konvertere den gamle konfiguration.

Når der allerede findes en korrekt konfigureret Linux installeret - nemlig den som allerede kører - så behøver du ikke at gå gennem hele konfigurationsprocessen og svare på alle spørgsmål. Det er tilstrækkeligt at opdatere konfigurationen med de ændringer, der forekommer mellem den nye og den gamle version.

Først kopieres den gamle konfiguration til folderen med kerne-kildekoden:

```
cp /boot/config-3.13.0-32-generic ~/uge2/linux-2.6.24/.config
```

Herefter bruges `make` til at tilpasse den gamle `.config` til den nye kerne-kildekode.

```
make oldconfig
```

Du anvender herved de gamle instillinger og konfigurerer kun de nye instillinger, som ikke fandtes i den gamle kenre. Tryk 'Y'+'enter' for at tilvælge en ny feature.

3. Bygge og installere kernen.

Du kan herefter kompilere kernen med:

```
make
```

Bygges kernen helt fra bunden tager dette lang tid ca. 30-60 min. Hvis du har flere beregningskerner i din CPU/ flere CPUer/ eller understøttelse for hyperthreading (i din virtuelle hardware) kan du kompilere hurtigere ved at anvende `make -j #`, hvor `#` er et tal, der angiver hvor mange tråde der skal køres.

Det er nu tid til at installere kernen. Dette gøres i flere skridt. Først skal du kopiere kernen (vmlinuz), dens konfiguration og dens moduler, samt filen `System.map`, til folderen `/boot` ved at bruge `make` af to omgange:

```
sudo make modules_install
sudo make install
```

Til sidst skal du opdatere den såkaldte grub boot-loader, så du får mulighed for at vælge din nye kerne når den virtuelle maskine genstartes. Ubuntu holder menuen skjult for at gøre tingene 'brugervenligt', så vi skal lige gøre den synlig først. Find filen:

```
/etc/default/grub
```

Find linjen med 'GRUB_HIDDEN_TIMEOUT=0' og sæt en '#' foran. Gem. Det kræver superbruger rettigheder via 'sudo' at rette i denne fil. Endeligt skrives:

```
sudo update-grub
```

I opstartsmenuen du får når du reboot findes din kerne under 'advanced options for Ubuntu'.

4. Gen-bygning af en kerne.

Har du ændret et sted i kerne-kildekoden kan du bygge kernen på ny ved at gøre som i afsnit 3.

5. Slette en kerne.

Hvis du ønsker at slette en kerne du har installeret, kan det gøres ved manuelt at fjerne følgende filer og foldere:

```
/boot/vmlinuz-<version>
/boot/initrd.img-<version>
/boot/System.map-<version>
/boot/config-<version>
/lib/modules/<version>/
```

Dette kan gøres med kommandoerne (du kan med fordel lave dette til et bash-script):

```
rm /boot/vmlinuz-3.13.0
rm /boot/initrd.img-3.13.0
rm /boot/System.map-3.13.0
rm /boot/config-3.13.0
rm -r /lib/modules/3.13.0
```

Herefter opdateres grub boot-loaderen:

```
update-grub
```

Behold altid mindst én kerne, som du ved fungerer!

Husk, at du behøver ikke slette en kerne hvis du blot har ændret i den og vil gen-installere; her udfører du blot skridt 3 igen.