Kompilace kernelu

 **Tento návod je určen pro pokročilé uživatele**

Proč kompilovat?

Důvodů pro kompilaci je několik:

- Ve vašem počítači se nachází zařízení, které není podporováno standardním jádrem distribuce, případně takové zařízení potřebujete občas připojit, stává se to kupříkladu u modemů, wifi atp.
- Novější jádro obsahuje nějakou vlastnost, kterou prostě musíte mít.
- Standardní jádro distribuce je podle vás příliš obecné, podporuje zbytečně mnoho hardwaru a tato podpora zbytečně zvětšuje jeho velikost (nejen na disku, ale i v paměti) a také ho značně zpomaluje.
- Jste hračička nebo guru, který kompiluje z principu vše, co kompilovat jde.

Důvodů proti je také několik:

- Jádro distribuce obsahuje obvykle řadu záplat zlepšujících jeho vlastnosti, nebo přidávajících podporu pro některá zařízení. Kompilací nového jádra pak přijdete o tyto vylepšení, pokud nepoužijete upravené zdrojové kódy jádra distribuce a nebo neaplikujete lepší patche potřebné právě pro váš počítač.
- Systém vám jede k úplné spokojenosti. Netřeba rýpat do něčeho, co jede…
- Jste absolutní linuxový začátečník, zelenější než tráva.
- Na pomalejším PC bude kompilace trvat i několik desítek hodin (záleží na konfiguraci jádra)

Něco málo o kernelu

Kernel také často označovaný českým termínem jádro a nebo také linuxové jádro je nejdůležitější a nezbytnou součástí sytému. Je zodpovědné mj. za zajištění bezpečného přístupu k počítačovému hardwaru, i ke spouštění procesů. Základním jádrem je takzvané **vanilla jádro**, jehož nejnovější řada je označena číslem 3.10. Dnes se většinou používají jádra z řady 3. Dříve bylo pravidlem, že pokud **druhé číslo** verze jádra bylo sudé, znamenalo to stabilní řadu a v případě lichého vývojovou. Momentálně tvůrci jádra ustoupili od této metody a rezignovali na vytvoření řady 2.7 a místo toho vyvíjejí jádro v řadě 2.6. Oficiálně od verze 2.6.11 byla do čísla verze přidána ještě čtvrtá číslice, ale poprvé použito u verze 2.6.8.1. **Třetí číslice** se mění pouze v případě přidání podpory nového zařízení a nebo přidání nové větší funkce do jádra a **čtvrtá číslice** se mění v případě drobnějších úprav.

Na základní jádro existuje mnoho záplat (patchů), které přidávají nové funkce a nebo opravují chyby či zvyšují stabilitu. Příkladem záplatovaného jádra je právě distribuční jádro, které je součástí Ubuntu.

Jádro můžete používat zkompilované na míru bez modulů anebo mít jádro, kde vybrané části, jsou zaváděny jako moduly dle aktuální potřeby. Nemodulární jádro s podporou jen vybraného hardwaru je bezesporu bezpečnější a výkonnější věc a rozhodně preferovanou volbou v případě serverových řešení, avšak v případě desktopu se jedná o nepraktickou volbu a to především k potřebě měnit hardwarovou konfiguraci při práci a v případě nemodulárního jádra by jste s přidáním určitého hardwaru museli, rekompilovat i samotné jádro, avšak pakliže vám kompilace nedělá problém a v případě optimalizovaného a ořezaného jádra zabere minimum času, tak samozřejmě můžete volit i jádro nemodulární.

Kde jej získat?

Pro kompilaci jádra je potřeba obstarat si zdrojové kódy. Je možné použít ty, které jsou přibaleny k distribuci, které jsou upraveny o vybrané záplaty a pokud nechcete ztratit některé funkce a vlastnosti pak je tato volba pro vás, anebo je možné použít i zdrojové kódy jádra zcela čisté z http://www.kernel.org/, bez přídavných záplat a doplňků a toto čisté jádro můžete dále obohatit o vybrané záplaty.

Distribuční jádro

V Ubuntu se samotné binární, to je již zkompilované, jádro se nachází v balíku **linux-image-3.X.Y-Z-architektura**, kde X, Y a Z jsou číslice upřesňující o jakou verzi jádra jde a architektura nám říká na jakou procesorovou řadu je jádro obsažené v balíku zkompilováno, např pro architekturu 686. Dnes je již k dispozici jen jádro pro 386 a jádro takzvané „generic“ univerzální pro všechny architektury x86/x86_64. Nakonec místo architektury může být i „server“, označení že jádro je optimalizováno pro server.

```
linux-image-2.6.X-Y-generic    (metabalík obsahující aktuální řadu "linux-image-generic")
```

S jádrem se instalují i jeho moduly, jejichž binární balíček se v Ubuntu jmenuje linux-restricted-modules-architektura a tudíž pro 686 architekturu:

```
linux-restricted-modules-2.6.X-Y-generic    (metabalík obsahující aktuální řadu "linux-restricted-modules-generic")
```

A samotné spouštěné jádro se nachází v adresáři **/boot** a moduly v adresáři **/lib/modules**. Předchozí balíky musí mít každý nainstalovaný pokud mu funguje systém, avšak ty nám umožňují používat linuxové jádro, ale již nám ho neumožňují měnit, avšak i zdrojové kódy distribučního jádra můžeme instalovat pomocí balíku:

```
linux-source
```

balík bez patchů a samotné patche můžete získat pouze pomocí „apt-get source linux-source-2.X.Y“

A nakonec pakliže chceme jen kompilovat externí moduly do našeho Ubuntu jádra, nepotřebujeme celé zdrojové soubory jádra, ale jen balíček **linux-headers-2.6.X-Y-architektura** například

```
linux-headers-2.6.X-Y-generic  (metabalik obsahujici aktualni radu "linux-headers-generic")
```

linux-headers se instalují již v rozbaleném stavu do příslušného adresáře v **/usr/src** a každopádně **linux-source obsahuje i linux-headers a pokud máte již nainstalován linux-source NEPOTŘEBUJETE linux-headers**, ale pro kompilaci externích modulů nám stačí jen headers a tak je zbytečné stahovat několikanásobně větší balík všech zdrojových kódů.

Zdrojové kódy stačí jen nainstalovat Synapticem/Adeptem/apt-get install(em) například:

```
sudo apt-get install linux-source-2.6.X (metabalík obsahující aktuální řadu "linux-source")
```

Stažené jádro se vám nahraje do adresáře **/usr/src/** a dokumentace do **/usr/share/doc/**. Samotné zdrojáky jádra se uloží pouze ve zkomprimovaném souboru a to:

```
/usr/src/linux-source-2.6.X.tar.bz2
```

Samotné rozbalení můžeme provést takto:

```
cd /usr/src
sudo tar -xvjf linux-source-2.6.X.tar.bz2
```

A po tomto příkazu již budeme mít zdrojáky v adresáři /usr/src/linux-source-2.6.12 rozbalené.

Vanilla kernel

Tzv. *vanilla kernel*, nebo také česky vanilkové jádro, tedy čistě jen to, co odkývá Linus. Toto jádro získáte z http://www.kernel.org/. Stáhnete si archiv, který má okolo 40MB (dávejte pozor, co stahujete, jsou tam různá jádra, stáhněte si raději nejnovější stabilní z řady 2.6 a také dejte pozor, ať nestáhnete jen nějaký patch, to poznáte podle velikosti, patche jsou malé. (A krom toho je to tam napsané).

Příprava na kompilace

Potřebujeme stáhnout nástroje pro kompilaci:

```
sudo apt-get install build-essential bin86 kernel-package gcc gcc-3.4 libncurses5 libncurses5-dev fakeroot
```

budeme-li využívat grafickou konfiguraci jádra „make xconfig“ tak:

```
sudo apt-get install libqt3-headers libqt3-mt-dev
```

anebo budeme-li využívat grafickou konfiguraci jádra „make gconfig“ tak:

```
sudo apt-get install libglade2-dev
```

Aplikace patchů

Pakliže jsme si stáhli vanilla kernel je vhodné upravit zdrojáky k našemu obrazu a nejlépe o nějaké zajímavé patche.

Patche známých hackerů kernelu:

- Con Kolivas
- Alan Cox
- Andrew Morton

Samotný proces patchování se dělá pomocí utilty **patch** a **cat** (v případě komprimovaného patche pomocí *.bz2* využijeme utilitu **bzcat**).

Spustíme tyto příkazy v adresáři s kernelem:

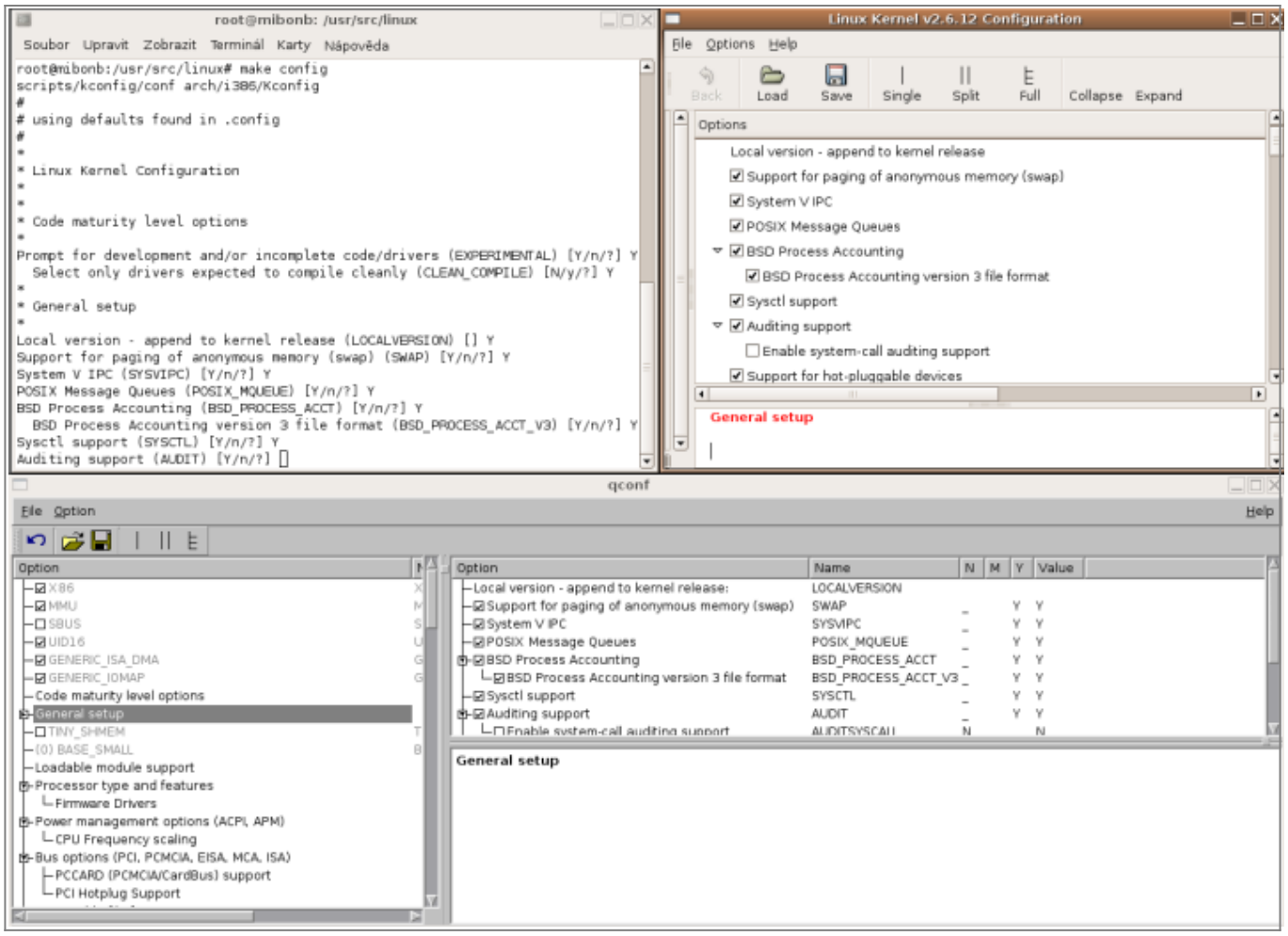
```
bzcat [jmeno patche].bz2 | patch -p1 --dry-run
```

Volba **–dry-run** simuluje aplikaci patche a pakliže vše proběhne bez problému můžeme se vrhnout do samého patchnutí.

```
bzcat [jmeno patche].bz2 | patch -p1
```

Patchování jádra vyžaduje, aby jste měli správný patch pro správné jádro a věděli co děláte a tak berte tento návod spíš jako „nakopnutí“.

Konfigurace jádra



Těsně před samotnou kompilací je potřeba vybrat, co se má kompilovat a jak má výsledné jádro vypadat. O to se starají konfigurační skripty, které můžete vyvolat pomocí příkazů:

```
make config (nejjednodušší)
make menuconfig (pseudografika)
make xconfig (nej pro kde)
make gconfig (nej pro gnome)
make oldconfig (konfigurace dle staršího konfiguračního souboru .config)
```

- Liší se navzájem jen v uživatelském komfortu. (viz obrázek výše) Zatímco make config se postupně ptá YES, NO, MODULAR, make menuconfig vám zobrazuje textové menu s nabídkami. make xconfig a gconfig dokonce využívá grafické prostředí X window system (xconfig na na Qt knihovně a gconfig na GTK+2). make oldconfig se snaží rozpoznat vaše nastavení ze starého konfiguračního souboru a podle něj vyplnit nový. Tedy chcete-li z dejme tomu z jádra 2.6.15 přejít na jádro 2.6.16, stačí nakopírovat .config z /usr/src/linux-2.6.15 do /usr/src/linux-2.6.16 a dát make oldconfig. Nyní budete dotázáni pouze na otázky, které jsou v jádře 2.6.16 nové, ty staré se zachovají dle starého kernelu.
- Při konfiguraci (kromě make oldconfig) máte na výběr, zda součást přikompilovat (YES), ignorovat (NO), nebo ji připravit jako modul s možností dynamické přilinkovaní (MODULE). Středně a málo používané součástí je vhodné kompilovat jako modulární.
- Většinu věcí můžete zakompilovat do jádra buď napevno (YES) nebo jako modul. Moduly lze narozdíl od vlastních částí kernelu odebírat a přidávat za provozu, při jejich zkoušení netřeba pořád restartovat.
- Konfigurace se ukládá do souboru .config a .config používaného jádra naleznete v adresáři **/boot/** v souboru **config-2.6.X-Y-architektura** a tento si můžete přikopírovat před konfigurací

a na něm postavit vaše nové nastavení viz příklad:

```
# cd /usr/src/linux
# cp /boot/config-2.6.12-9-686 .config
```

Pakliže byste teď dali **make oldconfig** tak s využitím aktuálního nastavení jádra se stejně nakonfiguruje nové, no tímto způsobem, ale pokud budete kompilovat zdrojáky k jádru které používáte dostanete nakonec po dalších krocích to samé binární jádro jaké si můžete stáhnout z balíčku. Pakliže ale dáte třeba **make xconfig** uvidíte v přehledném grafickém nástroji jak je vaše jádro právě nastavené a můžete začít měnit. Po dokončení konfigurace doporučuji si schovat pracně vytvořený nový .config.

Vybrané volby jádra

- **Code maturity level options —>** Toto vás zajímá, pokud chcete mít možnost přikompilovat do jádra experimentální, ještě ne úplně stabilní části. Takovéto části pak v dalších menu poznáte podle nápisu EXPERIMENTAL za každou takovouto položkou.
  - Prompt for development and/or incomplete code/drivers
  - Select only drivers expected to compile cleanly
- **General setup —>** Tato sekce se týká všeobecných nastavení komponent jádra. Především však funkcí pro meziprocesorové komunikace. Defaultně je možno vše ponechat jako povoleno.
  - Local version - append to kernel release - Zde je možné nastavit doplňující jméno kernelu, které bude připojeno k verzi jádra (např. 2.6.13-mujlinux1)
  - Support for paging of anonymous memory (swap) - Povolení užívání swapovacího oddílu nebo swapovacího souboru v případech kdy je potřeba více paměti než je celková paměť RAM. Tuto volbu určitě vybrat.
  - System V IPC - Podpora pro meziprocesovou komunikaci kterou využívají například emulátory jiných OS (DOSEMU,..). I když přímo emulátory nebudete používat, doporučuji dát tuhle věc do jádra.
  - POSIX Message Queues - Řazení hlášení dle normy POSIX, z našeho pohledu nepříliš důležitá volba. Nicméně dle defaultní konfigurace necháme v jádře.
  - BSD Process Accounting a BSD Process Accounting version 3 file format - Předávání informací o procesu. Doporučeno dát do jádra.
  - Sysctl support - Podpora pro dynamickou změnu parametrů jádra. Změna parametrů jádra probíhá prostřednictvím adresáře /proc/sys. Velmi doporučená volba.
  - Auditing support a Enable system-call auditing support - Povolení podpory auditu, kterou používají speciální subsystémy (např SELinux, Grsecurity,...). Ponechání této podpory v jádře nás nijak významně při jejím nevyužití neovlivní.
  - Kernel Userspace Events - Mechanizmus pro komunikaci přes netlink socket. Defaultně povoleno pro kompilaci do jádra.
  - Kernel .config support a Enable access to .config through /proc/config.gz [Y] - Při povolení této položky dojde k uložení konfigurace jádra (.config) do samotného obrazu jádra (bzImage). Konfigurační soubor je pak k nalezení v /proc/config.gz.
  - Configure standard kernel features (for small systems) —> Zde je možné při akceptování vybrat různé volby týkající se především optimalizace pro EMBEDDED zařízení.
- **Loadable module support —>** Sekce týkající se podpory pro práci s moduly.
  - Enable loadable module support - Podpora pro nahrávání modulů do jádra které jsou po kompilaci uloženy v /lib/modules/...
  - Module unloading - Další velmi praktická funkce týkající se modulů. Tato funkce slouží k odebrání modulů z jádra, je-li daný modul nepoužívaný.

    - Forced module unloading - Funkce užívaná spíše u vývojářů kernelu. Nicméně se může někdy hodit. Jedná se o odebrání modulu z jádra silou za použití programu rmmod (rmmod -f), tím dojde k odebrání modulu + ostatních modulů které na odebíraném modulu zívisí.
    - Module versioning support (EXPERIMENTAL) - Tímto povolíme podávání informací o modulech. Tato volba má význam pro vývojáře kernelu, proto je pro nás nepotřebná.
    - Source checksum for all modules - Podobný případ jako v předchozím. Pro normální použití nepodstatná fuknce.
    - Automatic kernel module loading - Velmi praktická funkce. Umožňuje automaticky nahrávat moduly do jádra v případě potřeby. V případě absence této funkce by se musely moduly nahrávat manuálně programem modprobe. Určitě volbu akceptovat.
- **Processor type and features —>**- Processor family vyberte podle toho, co máte v počítači, pokud si nejste jisti, můžete zadat Pentium. V případě, že jádro zkompilujete pro nekompatibilní typ procesoru, pravděpodobně nebudete schopni s tímto jádrem nastartovat systém. Obvykle je dobré vypnout High memory support (neznám mnoho lidí, kteří mají více, než 1GiB RAM), zapnout podporu MTRR (pokud máte CPU dle nápovědy) a v případě, že nemáte víceprocesorový stroj, vypnout Symmetric multi-processing support.
- **Power management options (ACPI, APM) —>**
- **Bus options (PCI, PCMCIA, EISA, MCA, ISA) —>**
- **Executable file formats —>**
- **Device Drivers —>**
- **File systems —>** - Chcete-li používat některý z dostupných žurnálovacích souborových systémů (Reiserfs, Ext3), zaškrtněte jejich podporu. Naprostá většina lidí využije DOS FAT fs support pro podporu oddílů s FAT souborovým systémem a VFAT (Windows-95) fs support pro podporu dlouhých jmen souborů na FAT oddílech, ze stejného důvodu je pro ISO 9660 CDROM file system support vhodné přidat i Microsoft Joliet CDROM extensions. Je možná i podpora NTFS (NTFS file system support).
- **Profiling support —>**
- **Kernel hacking —>**
- **Security options —>**
- **Cryptographic options —>**
- **Library routines —>**
- **Cluster Support —>**

Zdroje:

- http://kernel.xc.net/ - Linux Kernel Configuration Archive

Kompilace ala Debian (Ubuntu)

Takže po úspěšné konfiguraci zůstáváme v adresáři **/usr/src/linux** a zadáme příkaz:

```
sudo make-kpkg clean
```

Tento příkaz se postará o odstranění všech zbytků po předchozích kompilacích a následně zadáme příkaz zodpovědný za vlastní kompilaci:

```
sudo make-kpkg --stem linux --revision=vlastni.1.0 kernel_image
```

a nebo s initrd, neboli inital ramdisk (počáteční ramdisk). Je to předskokan jádra. Při bootu je natažen před samotným kernelem. Obsahuje predevším věci, které chceme mít použitelné před nahráním samotného jádra. To je často tzv. bootsplash (fullscreen obrázek při bootu). Také to může být nějaký ovladač, například tehdy pokud jsme ovladače hlavního souborového systému zkompilovali jako moduly. Kernel bez initrd je plně provozuschopný, avšak na některých strojích nemusí fungovat. Originální jádro používá initrd a jeho binárky naleznete také v **/boot/initrd.img-2.X.Y-Z-architektura** a pakliže chceme jádro i s initrd použijeme:

```
sudo make-kpkg --initrd --stem linux --revision=vlastni.1.0 kernel_image
```

Popis použitých a dalších voleb příkazu make-kpkg:

- –initrd - bylo vysvětleno výše
- –stem (–stem nazev) - tato volba umožní v našem případě aby první slovo názvu jádra bylo linux místo kernel
- –append-to-version (–append-to-version=nazev)- funguje identicky jako změna hodnoty v EXTRAVERSION v souboru *Makefile*. Má vliv nejenom na jméno konečného balíčku, ale také na jméno adresáře s moduly. Pravděpodobně při použití –append-to-version=nazev bychom měli použít i volbu modules_image *sudo make-kpkg –initrd –stem linux –revision=vlastni.1.0 kernel_image modules_image*.
- –revision (–revision=nazev)- má vliv pouze na jméno balíčku s jádrem
- Volby výstupu
  - kernel_image - výstupem bude instalační balík jádra i s moduly a s koncovkou .deb
  - modules_image - výstupem bude instalační balík pouze s moduly jádra s koncovkou .deb
  - kernel_headers - výstupem bude instalační pouze hlaviček kompilovaného jádra s koncovkou .deb
  - další volby **man make-kpkg**

Pakliže proběhla kompilace bez chyb měli bychom o adresář níž, tudíž v /usr/src/ najít instalační balíček s naším novým jádrem, který již lehce nainstalujeme pomocí příkazu **dpkg** a jeho odinstalaci zvládneme již přes apt/aptitude/synaptic. Příklad instalace:

```
cd ..
dpkg -i linux-image-2.6.X.Y_vlastni.1.0_i386.deb
```

BTW: make-kpkg má i volbu –config, která v defaultním nastavení spouští oldconfig, avšak můžete třeba –config=gconfig a co to udělá snad už po přečtení tohohle návodu tušíte.

Kompilace ala Linux

Na kompilaci jádra nemusíme využít bezpodmínečně make-kpkg, ale pokud jsme nějak postižení co nejkomplikovanějším postupem tak si můžeme užívat i standardní instalaci, která jde použít i v nedebianních klonech Linuxu. Každopádně musíme nakonfigurovat jádro - to je pro obě kompilace stejné a pak před samou kompilací v adresáři /usr/src/linux zadáme příkaz k odstranění všech zbytků po předchozích kompilacích:

```
make clean
```

Toto makro se stará o odstranění starých souborů. **make clean** maže zejména binární soubory, závislosti (.depend) a konfiguraci (.config) zachovává. Pozor na **make mrproper** (jak již název napovídá je při své „práci“ poctivější, maže vše tak, aby v adresáři zůstaly pouze soubory obsažené v původní distribuci zdrojových kódů) tzn. pokud chcete **make mrproper** tak ho použijte ještě před konfigurací. Před samotnou kompilací doporučuji navštívit soubor /usr/src/linux/Makefile, kde hned na začatku najdete:

```
VERSION = 2
PATCHLEVEL = 6
SUBLEVEL = 12
EXTRAVERSION =
```

a za položku „EXTRAVERSION =“ dopište nějaké označení vašeho jádra, abyste se pak po nějaké době věděli co jste to vůbec kompilovali. Tak to byla odbočka a již můžeme pokračovat v kompilaci, další postup závisí od verze jádra:

- **Jádra 2.4 a nižší:**

```
1. make dep - kontrola závislostí
2. make bzImage - kompilace kernelu
3. make modules - kompilace modulů
```

U příkazu make bzImage máme ještě další volby, make zImage a make zdisk, přičemž zImage vytváři zazipovaný obraz jádra, bzImage zazipovaný velký obraz jádra (pokud zImage nestačí) a zdisk (alt. bzdisk) vytváří přímo image na disketu (podmínkou je vložení naformátované diskety do mechaniky).

- **Jádra 2.6 a vyšší**

```
1. make - kompilace kernelu a modulů
```

Instalace jádra

Instalaci modulů do příslušného adresáře v /lib/modules/ provedeme příkazem:

```
make modules_install
```

Pakliže potřebujeme initrd neboli inital ramdisk (počáteční ramdisk). Je to předskokan jádra. Při bootu je natažen před samotným kernelem. Obsahuje především věci, které chceme mít použitelné před nahráním samotného jádra. To je často tzv. bootsplash (fullscreen obrázek při bootu). Také to může být nějaký ovladač, například tehdy pokud jsme ovladače hlavního souborového systému zkompilovali jako moduly. Kernel bez initrd je plně provozuschopný, avšak na některých strojích nemusí fungovat. Originální jádro používá initrd a jeho binárky naleznete také v **/boot/initrd.img-2.X.Y-Z-architektura** a pakliže chceme jádro i s initrd použijeme:

```
mkinitrd /boot/initrd-2.6.X.Y.EXTRAVERSION 2.6.X.Y.EXTRAVERSION
```

event. pokud v systému není k dispozici příkaz mkinitrd, použijeme:

```
mkinitramfs -o /boot/initrd-2.6.X.Y.EXTRAVERSION 2.6.X.Y.EXTRAVERSION
```

Samotné jádro se nám po kompilaci nachází v adresáři /usr/src/linux/arch/i386/boot/ (případně jiný dle architektury našeho systému v /usr/src/linux/arch/). Pro instalaci provedeme následující kroky:

```
cp /usr/src/linux/arch/i386/boot/bzImage /boot/vmlinuz-2.6.X.Y.EXTRAVERSION
cp /usr/src/linux/System.map /boot/System.map-2.6.X.Y.EXTRAVERSION
ln -s /boot/System.map-2.6.X.Y.EXTRAVERSION /boot/System.map
```

U jádra 2.6 a vyšší můžem samotné kopírování jádra do adresáře /boot provést pomocí make install

Tímto způsobem soubor jádra a System.map umístíme do adresáře /boot/ pod jménem příslušným pro verzi našeho jádra. A nakonec přidáme informaci o našem novém jádře do GRUBU (startovacího programu - zavaděče) a to přidáním položek do souboru /boot/grub/menu.lst pod řádku ### END DEBIAN AUTOMAGIC KERNELS LIST

```
title           Moje nové jádro 2.6.X.Y.EXTRAVERSION
root            (hd0,1)
kernel          /boot/vmlinuz-2.6.X.Y.EXTRAVERSION root=LABEL=/ ro
initrd          /boot/initrd.img-2.6.X.Y.EXTRAVERSION
```

Dodatečné úpravy

Moduly

Jaké moduly běží:

```
lsmod
```

Jak zavést modul:

```
modprobe modul
```

Modul se zadává bez cesty a bez .ko na konci!

Odstranění modulu:

```
rmmod modul
```

Automatické zavádění modulů při startu: Stačí je přidat do /etc/modules

```
sudo gedit /etc/modules
```

Opět se doplňuje jen název modulu, bez cesty a .ko na konci

Nenabootoval jsem, co teď?

No to se snad nestane, snad jste nechal v grubu nějakou záložní verzi jádra??? No, ale kdyby náhodou a nešlo jen modifikovat ručně start grubu. Tak nejlepší je použít nějaké liveCD (Ubuntu live, nUbuntu, Slax...) a zkusit „uchodit“ ještě nějakou tu jinou verzi v grubu a jestli ani to nepomůže, tak rekompilace z chrootu přes liveCD:

```
mkdir /mnt/ubuntu
mount /dev/hdxx /mnt/ubuntu
chroot /mnt/ubuntu
cd usr/src/linux
make menuconfig
```

No a překompilovat to znovu, ale nemyslím si, že pokud si nepřemažete binárky původních verzí jádra, že tato cesta bude vaší cestou.

Odkazy

- http://www.kernel.org/ - Oficiální stránky vanilkového jádra
- http://www.linuxhq.com/ - Neoficiální stránky