

Модульная анатомия

Цель

Разобраться в структуре загружаемых модулей Linux, больше узнать об устройстве ядра.

Ресурсы

Ноутбук Acer Aspire 3680 (старенький), дистрибутив Debian, голова и интернет.

Действия

Первый пункт

Нам нужно скачать, скомпилировать исходники ядра, с которым мы будем работать. Пользуемся возможностями Debian и получаем исходники ядра из репозитория Debian (всегда можно воспользоваться и <http://kernel.org>)

- `sudo apt-get update`
- `sudo apt-get install linux-doc-2.6.32 linux-manual-2.6.32 linux-source-2.6.32`
- `cd /usr/src/`
- `tar jxf /usr/src/linux-source-2.6.32.tar.bz2`
- `sudo apt-get install build-essential fakeroot kernel-package`
- `make menuconfig`
- `sudo make-kpkg clean`
- `sudo fakeroot make-kpkg --initrd --append-to-version=-mine kernel_image kernel_headers`
- `sudo dpkg -i linux-image-2.6.32-mine-10.00.Custom_i386.deb`
- `sudo dpkg -i linux-headers-2.6.32-mine-10.00Custom_i386.deb`
- `sudo update-initramfs -c -k 2.6.32-mine`
- `sudo shutdown -r now`

Дальше, если ядро сконфигурировано нормально, то загружаемся и вроде все, кстати `update-initramfs` нужен только, если `initrd.img-2.6.32-mine` не создавался на лету, когда устанавливался пакет

*1 советую обновить `gcc` (если пользуетесь `stable`, то обновлять с `testing`, в противном случае ядро не соберется, так как не будет нужных заголовочных файлов), вообще стоит использовать новые версии всех требуемых пакетов

*2 возможно придется доставить некоторые другие пакеты (см `/usr/share/doc/kernel-package/Kernel.htm`)

*3 ядро можно собрать и не “в стиле Debian”, а обычным образом - нет никакой разницы

*4 в данной версии ядра пришлось поправить файл /usr/src/linux-source-2.6.32/Documentation/lguest/lguest.c, в нем нужно было убрать строку #include <sys/eventfd.h> (21 строка), в противном случае оно просто отказывалось компилироваться, хотя я не понял почему (потому что такое же ядро на другом компьютере собралось без проблем), но погуглив нашел, что такая проблема не только у меня, и что такое решение используют и другие, после такого решения мы получаем при компиляции implicit декларацию функции, короче если в этом месте будет ошибка, то когда она вылезет непонятно

Второй пункт

Проверим, что все работает, для этого напишем какой-нибудь бесполезный модуль, скомпилируем его и посмотрим, что получится

```
1 //hello-1.c
2 #include <linux/module.h>
3 #include <linux/kernel.h>
4
5 int init_module(void) {
6     printk(KERN_INFO "Hellow world\n");
7     return 0;
8 }
9
10 void cleanup_module(void) {
11     printk(KERN_INFO "Godbye world\n");
12 }
```

Теперь Makefile

```
1 obj-m += hello-1.c
2 all:
3     make -C /usr/src/linux-source-2.6.32 M=$(shell pwd) modules
4 clean:
5     make -C /usr/src/linux-source-2.6.32 M=$(shell pwd) clean
```

Теперь из каталога, в котором лежит исходник и Makefile делаем

- sudo make

Должны получить примерно следующее:

```
1 make -C /usr/src/linux-source-2.6.32
M=/home/mirovingen/Interested/kernel_programming/src/Hellow_World modules
2 make[1]: Entering directory `/usr/src/linux-source-2.6.32'
3 CC [M]
/home/mirovingen/Interested/kernel_programming/src/Hellow_World/hello-1.o
4 Building modules, stage 2.
5 MODPOST 1 modules
```

```
6 CC
/home/mirovingen/Interested/kernel_programming/src/Hellow_World/hello-1.mod.o
7 LD [M]
/home/mirovingen/Interested/kernel_programming/src/Hellow_World/hello-1.ko
8 make[1]: Leaving directory `/usr/src/linux-source-2.6.32'
```

Если все получилось, у нас в каталоге должен появиться файл hello-1.ko, далее загружаем и выгружаем модуль:

```
1 sudo insmod hello-1.ko
2 sudo rmmod hello-1.ko
3 sudo cat /var/log/messages | grep -i -e "world"
```

Должны получить примерно следующее:

```
1 Jan 28 22:32:25 debian kernel: [ 6196.992494] Hellow world
2 Jan 28 22:40:17 debian kernel: [ 6669.012286] Goodbye world
```

From:
<http://wiki.osll.ru/> - **Open Source & Linux Lab**

Permanent link:
<http://wiki.osll.ru/doku.php/etc:users:kernel?rev=1264708695>

Last update: **2010/01/28 22:58**

