Сборка собственной прошивки для XO

Первоисточник находится тут.

Сборка прошивки разделена на два больших этапа:

1. создание репозитария с собственными пакетами
2. генерации прошивки с помощью Piligrim на базе собственного + OLPC репозитариев

Опыт сборки прошивки с собственным ядром

Сначала необходимо собрать собственный rpm с ядром. В моём случае это была голова из ветки wireless-testing

```
git clone git://git.kernel.org/pub/scm/linux/kernel/git/linville/wireless-testing.git
```

Затем из git OLPC мною был взят конфиг для их ядра и доделан до возможности делать make oldconfig в выбранной мною ветке.

Сборка ядра
В процессе сборки ядра возникали следующие проблемы:

1. исходно конфиг OLPC содержит директиву применять расширение 3DNOW, в виду чего для запуска образа в эмуляторе требуется процессор производства компании AMD. Я оного не имею, поэтому в menuconfig переключил тип процессора на Pentium/i686 вместо какого-то странного названия, которое влекло появление инструкций 3DNOW. Т.е. на самом деле опция конфигурационного файла с включением/выключением инструкций 3DNOW не является первостепенной и может быть отключена изменением типа целевого процессора
2. при загрузке на XO первое ядро сообщило, что не смогло найти модули *ide-core.ko* и *piix.ko* Думаю что для XO данные модули важны, поэтому нашел опции в конфигурационном файле и включил их.
3. так как в репозитариях OLPC пакет kernel имеет архитектуры от i386 до i586, то на всякий случай в спецификации была указана архитектура i686.

Далее делаем отдельный репозитарий для нашего пакета. В моём случае он назывался **kernel-2.6.27-5.olpc.i686.rpm**. Чтобы сделать репозитарий достаточно в каталоге где лежит пакет написать **createrepo .** После выполнения, в каталоге появиться поддиректория repodata, в которой собственно и хранится описание нашего репозитария для yum.

Следующий шаг состоит в включении новоиспеченного репозитария в конфиг, чтобы на стадии сборки прошивки yum подцепил наше ядро взамен представленных в репозиатриях OLPC.

Включение собственного репозитария содержит несколько тонкостей:

1. нигде не написано, однако при инсталяции pilgrim использует **ТОЛЬКО** репозитарии, имена которых начинаются с “olpc”, так что незабудьте добавить префикс olpc к названию собственного репозитария

2. стоит учесть, что для вашего ядра могут понадобиться(в следствии зависимостей) другие пакеты, поэтому я не долго думая добавил репозитарий 9 Федоры... Что в свою очередь выявило следующий косяк:
3. всегда ставте пакет NetworkManager из репозитариев OLPC. У них он очень забавный и содержит какие-то непонятные питоновские скрипты и утилиты, в том числе и для настройки mesh. Конкретно не смотрел, но факт на лицо.

Таким образом конфиг для yum у меня стал выглядеть следующим образом:

```
[olpc-fedora]
name=Fedora 9 - i386
#baseurl=http://download.fedoraproject.org/pub/fedora/linux/releases/9/Every
thing/$basearch/os/
mirrorlist=http://mirrors.fedoraproject.org/mirrorlist?repo=fedora-9&arch=i3
86
enabled=1
gpgcheck=0
exclude=NetworkManager

[olpc-local]
name=zps
baseurl=file:///home/zps/workspace/OLPC/repo/i386/os
enabled=1
gpgcheck=0

[olpc_development]
name=OLPC Development repo, based on koji tag dist-olpc3-devel.
#baseurl=http://koji.fedoraproject.org/static-repos/dist-olpc3-build-current
/i386/
#baseurl=http://xs-dev.laptop.org/~cscott/repos/dist-olpc3-devel/
baseurl=file:///home/zps/workspace/OLPC/repo/xs-
dev.laptop.org/~cscott/repos/dist-olpc3-devel/
enabled=1
gpgcheck=0
exclude=kernel

[olpc-joyride]
name=OLPC 'Joyride' Repository
#baseurl=http://xs-dev.laptop.org/~cscott/repos/joyride/
baseurl=file:///home/zps/workspace/OLPC/repo/xs-
dev.laptop.org/~cscott/repos/joyride/
enabled=1
gpgcheck=0
olpc-development-yum-install.conf 103 utf-8 0x  48,1 Внизу
```

Для ускорения я скачал репозитарии себе на машину, потому что канал на xs-dev.laptop.org слишком не надежный. Статисчески я выяснил что три из четырех запусков генерации прошиыки падает только потому, что он в процессе не смог чего-то закачать с их репозитария. Учитывая что генерация занимает много времени, лучше лишний раз избежать таких

“приколов”.

После добавления собственных репозитариев достаточно выполнить следующую команду:

```
# pilgrim-autobuild --config-dir . --stream olpc-development --dest-dir . --
variant devel_jffs2
```

Выполнение данной команды имеет тоже несколько тонкостей:

1. выполняется она под root
2. вызов скриптов выполняется из /usr/sbin без конкретных/относительных путей поэтому, если вы правите скрипт у себя в домашней директории вы должны вновь его установить, чтобы увидеть ваши правки в действии.
3. во время выполнения лучше не жать Control+C. В процессе выполнения он очень хитро создает файлы с будущим диском и мапит их на /dev/loop5 и /dev/loop6

Просмотр содержимого img файлов

Результатом работы pilgrim будет являться img файлы, которые представляют из себя файловую систему. В варианте готовом для прошивке на XO это jffs, для QEMU — ext3

Для просмотра содержимого ext3-прошивки для эмулятора можно воспользоваться следующей последовательностью команд

```
$ sudo /sbin/losetup /dev/loop5 xo-1-olpc-stream-joyride-
build-24-20081011_1843-devel_ext3.img
$ sudo /sbin/losetup -o 31744 /dev/loop6 xo-1-olpc-stream-joyride-
build-24-20081011_1843-devel_ext3.img
$ mkdir ttt
$ sudo mount -t ext3 /dev/loop6 ttt
```

чтобы отмонтировать:

```
$ sudo umount ttt/
$ sudo /sbin/losetup -d /dev/loop6
$ sudo /sbin/losetup -d /dev/loop5
```

В случае если хочется просмотреть содержимое реальной прошивки то можно воспользоваться этим руководством. *Лично я не смог примонтировать существующий jffs2 образ. Есть подозрение, что в ядре Федоры 9 нет поддержки сжатых jffs2.*