Настройка доступа в Internet через GPRS на Linux используя Siemens S55 и Bluetooth

Alex Ott

25 сентября 2004 г.

Все началось с того, что дома у меня плохая телефонная линия, которую не видит мой модем серии Eline на 57.6kbit (кроме модема ее не видели еще несколько моделей телефонов). Кабель в квартиру мне тоже не светил, поэтому было решено попытать счастья со связью через мобильный телефон. В рамках планового апгрейда телефона, был закуплен Siemens S55 — в качестве решающих факторов было наличие голубого зуба (bluetooth), irda, j2me, ну и конечно поддержки GPRS. Оператором сотовой связи является Beeline.

Со стороны компьютера для общения с телефоном использовался Bluetooth USB Adapter Tekram TM-300. В качестве операционной системы используется ALTLinux Master 2.2, который раз в неделю обновляется из Sisyphus (поскольку я являюсь одним из разработчиков данного дистрибутива). Использовалось ядро Linux 2.4.22 (сборка 2.4.22-std-up-alt13).

Сначала я приступил к настройке bluetooth на Linux. Для работы с ним, в систему были установлены пакеты libbluez 2.4, bluez-sdp 1.5, bluez-pan 1.1, bluez-hciemu 1.0, bluez-utils 2.3. После изучения доступной документации, началась настройка:

1. В файл `/etc/modules.conf` было добавлены строки, которые подключают модули поддержки bluetooth и

   ```
   alias net-pf-31 bluez
   alias tty-ldisc-15 hci_uart
   alias bt-proto-0 l2cap
   alias bt-proto-2 sco
   alias bt-proto-3 rfcomm
   alias bt-proto-5 bnep
   ```

2. Вносим изменения в `/etc/bluetooth/hcid.conf`, чтобы общение по bluetooth шифровалось, а также была указана программа, которая по запросу выдавала pin-код телефону. Это была директива `pin_helper` в разделе `options` и она стала выглядеть следующим образом:

   ```
   pin_helper /etc/bluetooth/bluepin;
   ```

3. Создаем сам файл `/etc/bluetooth/bluepin` следующего содержания:

```
#/bin/sh

echo "PIN:00"
```

4. Тестируем работу USB Bluetooth адаптера. С помощью утилиты **hciconfig**, был получен список bluetooth устройств (в моем случае это был `hci0`).

5. Выполняем сканирование окружающей среды на предмет наличия bluetooth устройств:

```
root@flash:/\>hcitool scan
Scanning ...
        00:01:E3:70:E0:AF       OttAlex
```

   это и был мой телефон.

6. Убеждаемся в том, что телефон поддерживает dial-up networking, что было и сделано с помощью команды:

```
root@flash:/\>sdptool search DUN
Inquiring ...
Searching for DUN on 00:01:E3:70:E0:AF ...
Service Name: Dial-up networking
Service RecHandle: 0x11103
Service Class ID List:
  "Dialup Networking" (0x1103)
  "Generic Networking" (0x1201)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 1
Language Base Attr List:
  code_ISO639: 0x656e
  encoding:    0x6a
  base_offset: 0x100
Profile Descriptor List:
  "Dialup Networking" (0x1103)
    Version: 0x0100
```

   Аналогичным образом можно получить информацию о том, какие сервисы поддерживает телефон с помощью команды sdptool browse

7. и протестируем подключение к телефону с помощью l2ping:

```
root@flash:/\>l2ping 00:01:E3:70:E0:AF
Ping: 00:01:E3:70:E0:AF from 00:0A:94:00:03:EC (data size 20) ...
0 bytes from 00:01:E3:70:E0:AF id 200 time 36.57ms
0 bytes from 00:01:E3:70:E0:AF id 201 time 31.55ms
.............
5 sent, 5 received, 0% loss
```

8. Теперь можно подключиться к телефону. Это делается через **rfcomm**. Для этого используется команда вида:

```
root@flash:/\>rfcomm bind 0 00:01:E3:70:E0:AF 1
```

которая заставляет подключить устройство 0 к bluetooth устройству `00:01:E3:70:E0:AF` к первому каналу.

Можно проверить подключение с помощью команды **rfcomm show**.

В том случае, если у вас нет файлов устройств `/dev/rfcomm*`, то вам необходимо их создать с помощью команды вида:

```
mknod /dev/rfcomm0 c 216 0
```

у меня они уже были созданы при установке системы.

9. Проверяем работу подключения с помощью minicom. Запускаем его, настраиваем на работу с `/dev/rfcomm0` вместо /dev/modem и пробуем набрать команду:

   ```
   ATDT+79031234567
   ```

   вместо `1234567` я указал свой номер, и телефон начал звонить показывая, что я вызываю сам себя :-)

10. Теперь приступаем к настройке **pppd**. Создаем файл `/etc/ppp/peers/gprs` следующего содержания:

    ```
    /dev/rfcomm0 57600
    connect '/usr/sbin/chat -v -f /etc/ppp/chat/gprs'
    noauth
    defaultroute
    lock
    debug
    novjccomp
    nopcomp
    noaccomp
    nodeflate
    novj
    nobsdcomp
    default-asyncmap
    ipcp-accept-local
    ipcp-accept-remote
    usepeerdns
    user beeline
    nodetach
    ```

    При этом отключается всякое сжатие, как указано в рекомендациях сервисной службы beeline, и не происходит отключение от управляющего терминала, так что **pppd** можно остановить с помощью Ctrl-C

    В файл `/etc/ppp/chat/gprs` записываем команды программы **chat**:

```
TIMEOUT 5
ECHO    ON
ABORT   '\nBUSY\r'
ABORT   '\nERROR\r'
ABORT   '\nNO ANSWER\r'
ABORT   '\nNO CARRIER\r'
ABORT   '\nNO DIALTONE\r'
ABORT   '\nRINGING\r\n\r\nRINGING\r'
''              \rAT
TIMEOUT 12
OK              ATH
OK              ATE1
OK              AT+CGDCONT=1,"IP","internet.beeline.ru"
OK              ATD*99***1#
CONNECT
```

И в файл `/etc/ppp/pap-secrets` добавляем строку:

```
beeline  ppp0  "beeline"
```

и все, можно пробовать подключиться к интернету с помощью команды

```
pppd call gprs
```

запущенной из под пользователя *root*.

11. Дополнительная настройка может заключаться в том, что в каталоги `/etc/ppp/ip-up.d` и `/etc/ppp/ip-down.d` можно поместить скрипты, которые будут запускаться при поднятии и закрытии соединения. У меня это команды, которые запускают и останавливают **fetchmail**.