Quick Linux AX.25 HowTo

By 5B8AP

5b8ap(at)qsl.net

Revision History

Date of this revision: 13.05.2010	Date of Next revision: (date)

Number	Date	Summary of Changes	Changes marked
0.1	10/10/2006	Initial version	No
0.2	15/10/2006	Completed sections 6 & 7	No
0.3	24/03/2009	Minor correctiona and additions	No
0.4	13/05/2010	How to set up a KISS device	No

TABLE OF CONTENTS

1 Introduction

Maybe the simplest and cheapest way to get on the air with packet radio is to use a simple home-brew interface like the Baycom Modem. Unfortunately, most of these interfaces were designed to work under DOS and cannot work under Windows. However, they can work well with the Linux operating system.

This document describes the necessary steps you need to follow to set up a Linux system with a Baycom modem for use with some packet radio applications like APRSD or Xastir. It's meant to be used as a "Quick Start" quide. You will have to refer to the "Linux Amateur Radio AX.25 HOWTO" for more details.

What is described below is based on my personal experience and on the "Linux Amateur Radio AX.25 HOWTO" which I found at http://www.faqs.org/docs/Linux-HOWTO/AX25-HOWTO.html

There may be steps that need improvement in what I describe. If you think there's something that needs corrections, please let me know. You can email me at **5b8ap**(at)**qsl.net**

1.1 Required Hardware & Software

One thing I like about Linux and packet radio is that it does not require a high-performance system, so you can "recycle" your old PC. I have used old PCs like Pentium 100 - 166MHz or AMD K6/2 500MHz for this. Of course, the faster your machine, the better.
I've used AX.25 on RedHat Linux 7.2 and 9.0. Apart from the standard Linux distribution you will also need:

- The kernel source
- AX.25 Library
- AX.25 Applications
- AX.25 Tools
- Other software (e.g. APRSD, Xastir etc).

Note that the process of compiling all required software can prove to be quite time consuming. It usually takes me many hours until I manage to correclty compile the kernel and the other AX.25 software. I hope you will be able to do it faster ☺

You can build your own Baycom modem. Schematics can be found on the Web.
You can try:

- http://web.telia.com/~u85920178/use/baycom.htm
- http://www.qsl.net/sv1bsx/Baycom/baycom.html

The one I've built looks like the one in the first link.

Generally, the Baycom interface is easy to build. You may have some difficulties obtaining the TCM3105 IC, but you can most probably order it on-line. Some sites that may have it in stock:

- http://www.futurlec.com/
- http://www.tcm3105.com/

Once you build the modem you can test it using its own software under DOS. You can use an old PC or boot your PC with a DOS floppy. The software can fit on a bootable floppy. Check the web or http://www.baycom.org/ for the Baycom software under DOS.

2 Compiling the kernel

[Most of the information below were taken from the "Linux Amateur Radio AX.25 HowTo" by Jeff Tranter, VE3ICH]

Before you are able to use AX.25 on your Linux system, you need to re-compile the kernel to enable AX.25 support. For more information you can check the "Linux Kernel HowTo". If you search the Web you should be able to find it on many sites.

I extracted the kernel source in /usr/src. After that I ran the configuration script and selected the appropriate options for AX.25 support and for the specific hardware configuration.

 # make xconfig

What worked most of the times was loading one of the existing configuration files for the appropriate machine type from the **configs** directory and changing the configuration parameters as necessary. The i686 file worked for most of my Intel Pentium and AMD PCs.

The options most relevant to an AX.25 configuration are:

```
Code maturity level options  --->
    [*] Prompt for development and/or incomplete code/drivers
    ...
General setup  --->
    ...
    [*] Networking support
    ...
Networking options  --->
    <*> UNIX domain sockets
    ...
    [*] TCP/IP networking
    ...
    [?] IP: tunneling
    ...
Amateur Radio Support --->
    --- Packet Radio protocols
    [*]   Amateur Radio AX.25 Level 2 protocol
    [?]     AX.25 DAMA Slave support
    [?]     Amateur Radio NET/ROM protocol
    [?]     Amateur Radio X.25 PLP (Rose)
    AX.25 network device drivers  --->
    <?> Serial port KISS driver
    <?> Serial port 6PACK driver
    <?> BPQ Ethernet driver
    <?> High-speed (DMA) SCC driver for AX.25
    <?> Z8530 SCC driver
    <?> BAYCOM ser12 fullduplex driver for AX.25
    <?> BAYCOM ser12 halfduplex driver for AX.25
    <?> BAYCOM picpar and par96 driver for AX.25
    <?> BAYCOM epp driver for AX.25
    <?> Soundcard modem driver
    [?]   soundmodem support for Soundblaster and compatible cards
    [?]   soundmodem support for WSS and Crystal cards
    [?]   soundmodem support for 1200 baud AFSK modulation
    [?]   soundmodem support for 2400 baud AFSK modulation (7.3728MHz crystal)
    [?]   soundmodem support for 2400 baud AFSK modulation (8MHz crystal)
    [?]   soundmodem support for 2666 baud AFSK modulation
    [?]   soundmodem support for 4800 baud HAPN-1 modulation
    [?]   soundmodem support for 4800 baud PSK modulation
    [?]   soundmodem support for 9600 baud FSK G3RUH modulation
    <?> YAM driver for AX.25
```

The options I have flagged with a `*' are those that you *must* must answer `Y' to. The rest are dependent on what hardware you have and what other options you want to include.

After completing the configuration, I used the following commands to compile the new kernel:

#make dep
#make clean
#make bzImage

After successful compilation of the kernel, I proceeded with compiling and installing the kernel modules:

#make modules
#make modules_install

Note: If at some point the compilation fails and you want to retry it after changing some settings, it might be a good idea to run **make mrproper** before. I've had cases when building of the modules failed at first but was successful after running **make mrproper**.

After the modules are also compiled and installed successfully, you can proceed with installing the new kernel so that the PC can boot from it. From the kernel source directory, copy the kernel boot image to the **/boot** directory. For example:

```
# cp arch/i386/boot/bzImage /boot/bzImage-KERNEL_VERSION
```

The next step is to configure the bootloader. To configure the GRUB bootloader you have to edit the **/etc/grub.conf** file to add the boot option for the new kernel. A sample **grub.conf** file is shown below:

```
#boot=/dev/hda
default=0
timeout=10
title Red Hat Linux (2.4.20-24.9)
       root (hd0,1)
       kernel /boot/vmlinuz-2.4.20-24.9 ro root=LABEL=/
       initrd /boot/initrd-2.4.20-24.9.img
title AX25 (2.4.20-20.9)
       root (hd0,1)
       kernel /boot/bzImage--2.4.20-20.9 ro root=LABEL=/
```

Another one with a few differences:

```
#boot=/dev/hda
default=0
timeout=10
splashimage=(hd0,0)/grub/splash.xpm.gz
title Red Hat Linux (2.4.20-8)
       root (hd0,0)
       kernel /vmlinuz-2.4.20-8 ro root=LABEL=/ hdc=ide-scsi
       initrd /initrd-2.4.20-8.img
title AX25 (2.4.20-8)
       root (hd0,0)
       kernel /bzImage_AX25_1_9_06 ro root=/dev/hda3
```

After making the changes you need to restart your PC and hope that the new kernel boots ☺
If it doesn't you will have to re-compile it. I usually try to change the Processor type and features or use another of the existing configuration files. Once you are sure that the PC will boot with the new kernel, you can change the **default** option from **0** to **1** so that it boots by default using the AX25 kernel.

3 Compiling the AX.25 Library, Tools & Applications

There are three main packages you need to compile to enable support for AX.25:

- AX.25 Library
- AX.25 Tools
- AX.25 applications

I downloaded the latest versions of the AX.25 tools, AX.25 apps and libax25 from http://sourceforge.net/projects/ax25/

Compiling them should be straightforward if you have the necessary compilers and libraries installed on your system.

To compile and install libax25 you should use a series of commands similar to the following:

```
# cd /usr/src
# tar xzvf libax25-0.0.7.tar.gz
# cd libax25-0.0.7
# ./configure --exec_prefix=/usr --sysconfdir=/etc --localstatedir=/var
# make
# make install
```

If this is a first time installation, that is you've never installed any AX.25 code on your machine before, you should also use the:

```
# make installconf
```

command to install some sample configuration files into the /etc/ax25/ directory from which to work.
You can now build install the AX.25 tools in a similar fashion:

```
# cd /usr/src
# tar xzvf ax25-tools-0.0.6.tar.gz
# cd ax25-tools-0.0.6
# ./configure --prefix=/usr  --sysconfdir=/etc --localstatedir=/var
# make
# make install
# make installconf (if you want to install the configuration files)
```

And finally you can install the AX.25 applications:

```
# cd /usr/src
# tar xzvf ax25-apps-0.0.4.tar.gz
# cd ax25-apps-0.0.4
# ./configure --prefix=/usr  --sysconfdir=/etc --localstatedir=/var
# make
# make install
# make installconf (if you want to install the configuration files)
```

If you get messages something like:

```
gcc -Wall -Wstrict-prototypes -O2 -I../lib -c call.c
call.c: In function `statline':
call.c:268: warning: implicit declaration of function `attron'
call.c:268: `A_REVERSE' undeclared (first use this function)
call.c:268: (Each undeclared identifier is reported only once
call.c:268: for each function it appears in.)
```

then you should double check that you have the ncurses package properly installed on your system. For RedHat 7.2 I had to install ncurses, ncurses4, ncurses-devel and re-run the configuration script to manage to compile AX.25 Apps.

I also noticed that an application required for AX.25, **node** is not included in RedHat. You will have to download and build this manually. I downloaded it from ftp://ftp.hes.iki.fi/pub/ham/linux/ax25/

4 Configuring AX.25

Once the AX.25 packages are compiled and installed, you can proceed with the basic configuration of the sytem.

Each of the AX.25 applications read a particular configuration file to obtain the parameters for the various AX.25 ports configured on your Linux machine. For AX.25 ports the file that is read is the

/etc/ax25/axports file. You must have an entry in this file for each AX.25 port you want on your system.

4.1 Configuring a Baycom Device

To configure a Baycom device on COM1, the /etc/ax25/axports file will look like this:

```
# /etc/ax25/axports
#
# The format of this file is:
#
# name callsign speed paclen window description
#
bcsf0   5B8AP-5       1200    255     2       144.800 MHz (1200  bps)
```

bcsf0 is the network device created by the Baycom driver on COM1. bcsf1 will stand for COM2 and so on.

After creating the entry in the /etc/ax25/axports file, I use the following commands (as root) to start the Baycom device:

```
# setserial /dev/ttyS0 uart none
# sethdlc -p -i bcsf0 mode ser12* io 0x3f8 irq 4
# /sbin/ifconfig bcsf0 hw ax25 [my callsign] up
# /sbin/ifconfig bcsf0 [port IP] netmask 255.255.255.0
```

Usually, the I/O addresses and IRQs of the COM ports are:

- 0x03f8, IRQ4 for COM1
- 0x02f8, IRQ3 for COM2

After configuring your port you should also be able to ping it at the IP address you specified.
At this point, you should be able to transmit packets if you connect the Baycom modem to a radio and your PC and issue the **call** command with the syntax **call [interface] [callsign]**, for example:

call bcsf0 5b8ap

That should key the radio and transmit a valid packet.

(I'm not sure if this is needed in order to use your Baycom device with Xastir or APRSD, but I'm including some information anyway.)

To add the new port to ax25d, I edited the existing /etc/ax25/ax25d.conf file (kept a backup of the original). Basically I commented out all lines I didn't need and left only the configuration parameters for my Baycom device. My ax25d.conf file looks like this:

```
# /etc/ax25/ax25d.conf
#
# ax25d Configuration File.
#
# AX.25 Ports begin with a '['.
#
#[OH2BNS VIA 1]
#NOCALL   * * * * * *  L
#default  * * * * * *  -       root  /usr/local/sbin/ttylinkd  ttylinkd
#
[5B8AP-5 VIA bcsf0]
NOCALL   * * * * * *  L
default  1 10 3 300 10 5  -    root  /usr/sbin/node    node
#
#
#[OH2BNS VIA 2]
#NOCALL   * * * * * *  L
```

```
#default  * * * * * *  -       root  /usr/local/sbin/ttylinkd  ttylinkd
#
```

After configuring the ax25d.conf file, you can start ax25d using

 # /usr/sbin/ax25d

Once you start it, you should be able to see the active AX.25 ports using the command

 # netstat –-ax25

```
$ netstat --ax25
Active AX.25 sockets
Dest      Source     Device  State       Vr/Vs    Send-Q  Recv-Q
*         5B8AP-5    bcsf0   LISTENING   000/000  0       0
$
```

4.2 Configuring a KISS Device

The steps below have been followed in order to configure the system to use an MFJ 1274 TNC.

To configure a KISS device on COM1 at 9600 baud, the /etc/ax25/axports file will look like this:

```
# /etc/ax25/axports
#
# The format of this file is:
#
# name callsign speed paclen window description
#
mfjtnc  5B8AP-5       9600    255     2       144.800 MHz (1200 bps)
```

mfjtnc is the name you will use later to access the device and you can set it to anything you want.

After creating the entry in the /etc/ax25/axports file, you have to bring the TNC into KISS mode. For the specific TNC, I used the **minicom** program to connect to it and issued the following commands:

KISS ON (enter)
RESTART (enter)

I then exited minicom through the **Quit with no Reset** option.
You will have to do this every time you reboot the TNC (unless you know how to make the change permanent).

I used the folowing commands (as root) to start the KISS device:

 # /usr/sbin/kissattach /dev/ttyS0 mfjtnc [IP]
 # /usr/sbin/kissparms -p mfjtnc -f no -l 50 -r 32 -s 320 -t 320

 # /sbin/ifconfig ax0 [IP] netmask [netmask]
 # /sbin/ifconfig ax0 hw ax25 5B8AP-5 up

After configuring your port you should also be able to ping it at the IP address you specified.
At this point, you should be able to transmit packets if you connect the Baycom modem to a radio and your PC and issue the **call** command with the syntax **call [interface] [callsign]**, for example:

call mfjtnc 5b8ap

That should key the radio and transmit a valid packet.

For more information on creating a KISS device, refer to **section 6.1.1** in
http://www.faqs.org/docs/Linux-HOWTO/AX25-HOWTO.html

Note: In case you need to "detach" the TNC from the serial port, you can do it by killing the **kissattach** process.

- Do a **ps axu** to view the list of processes
- There should be a process like **/usr/sbin/kissattach /dev/ttyS0 mfjtnc [IP]**
- Kill the process by issuing the command **kill [process ID]** (as root)

To add the new port to ax25d, I edited the existing /etc/ax25/ax25d.conf file as follows:

```
# /etc/ax25/ax25d.conf
#
# ax25d Configuration File.
#
# AX.25 Ports begin with a '['.
#
#[OH2BNS VIA 1]
#NOCALL   * * * * * *  L
#default  * * * * * *  -      root  /usr/local/sbin/ttylinkd  ttylinkd
#
[5B8AP-5 VIA mfjtnc]
NOCALL   * * * * * *  L
default  1 10 3 300 10 5  -    root  /usr/sbin/node    node
#
#
#[OH2BNS VIA 2]
#NOCALL   * * * * * *  L
#default  * * * * * *  -      root  /usr/local/sbin/ttylinkd  ttylinkd
#
```

After configuring the ax25d.conf file, you can start ax25d using

 # /usr/sbin/ax25d

Once you start it, you should be able to see the active AX.25 ports using the command

 # netstat –-ax25

Note that you can kill the kissattach command, so that you can access the serial port through another application.

5 Compiling Xastir

Xastir is program for receiving and plotting APRS™ position packets. It supports many map formats, has many nice features and is highly customizable, but building it can prove to be tricky if you don't have the necessary libraries installed.

You can download the latest version of Xastir from http://www.xastir.org/
The INSTALL file that comes with the source contains all the necessary information on how to build the package, which are the required libraries and how to install them.

The number of libraries and other packages you will have to install will depend on what you already have installed on your system.

The libraries I installed when I built Xastir on RedHat 9.0 the last time were:

1. freetype-devel-2.1.3-6 (RPM)
2. fontconfig-devel-2.1.9 (RPM)
3. XFree86-devel-4.3.0-2 (RPM)
4. Openmotif-devel-2.2.2-14 (RPM)

After installing these libraries Xastir could be built, but without many features/formats support, so I installed the following additional libraries:

5. shapelib-1.2.10-1 (RPM)
6. ImageMagick-devel-5.4.7-10 (RPM)
7. libtiff-devel-3.5.7-11 (RPM)
8. PCRE-6.3 (built from source)
9. proj-4.4.7-1 (RPM)
10. libgeotiff-1.1.4-1 (RPM)
11. festival-1.4.2-16 (RPM)
12. libjpeg-devel-6b-26 (RPM)
13. libpng-devel-1.2.2-16 (RPM)
14. GDAL-1.3.1 (built from source)
15. gpsmanshp-1.0.2-1
16. gpsman-6.2.1 (as per installation instructions)
17. ImageMagick-perl-5.4.7-10 (RPM)
18. gv-3.5.8-22 (RPM)
19. bzip2-devel-1.0.2-8 (RPM)
20. Upgraded ImageMagick to version 5.5.6-5 (using RPMs)
21. Created a symbolic link from /usr/lib/libgdal.so.1 → /usr/local/lib/libgdal.so.1.9.0

If you get font warnings when you run Xastir, you can set the environment variable LANG="en_US"

 # export LANG="en_US"

You can convert UI-View maps for use with Xastir by creating a **.geo** file using **inf2geo.pl** available in **xastir/scripts** directory.

If you need to create a passcode for your callsign to use with an APRS server, you can use the **callpass** command:

 # callpass [your callsign]

To configure Xastir to work with the Baycom interface, once the program opens:

- Go to **Interface → Interface Control**
- Click **"Add"** to configure a new interface
- Select to add an **"AX25 TNC"**
- For the AX.25 Device name, enter the name of the network device for your Baycom modem (i.e. bcsf0, bcsf1 etc.)
- Configure the other options based on your needs. See screenshot below for a sample configuration

To configure Xastir to work with the MFJ, KISS TNC follow the procedure above, but instead of addina an **"AX25 TNC"**, choose to add a **"Serial TNC"**. Be sure to select the correct TNC port and baud rate.

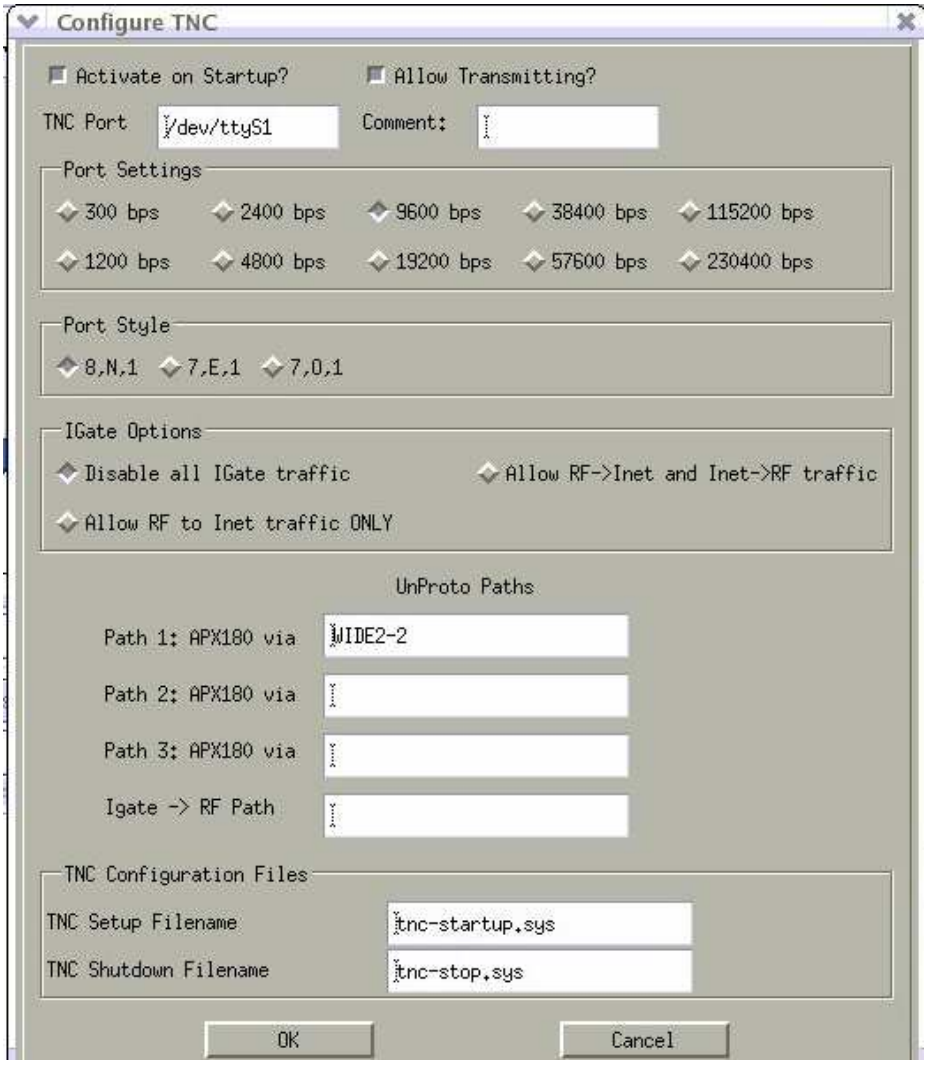

Figure 1 - Sample TNC Xastir port setup for MFJ KISS TNC

Note that you need to run Xastir as root or another user that has access to the serial port you are using.

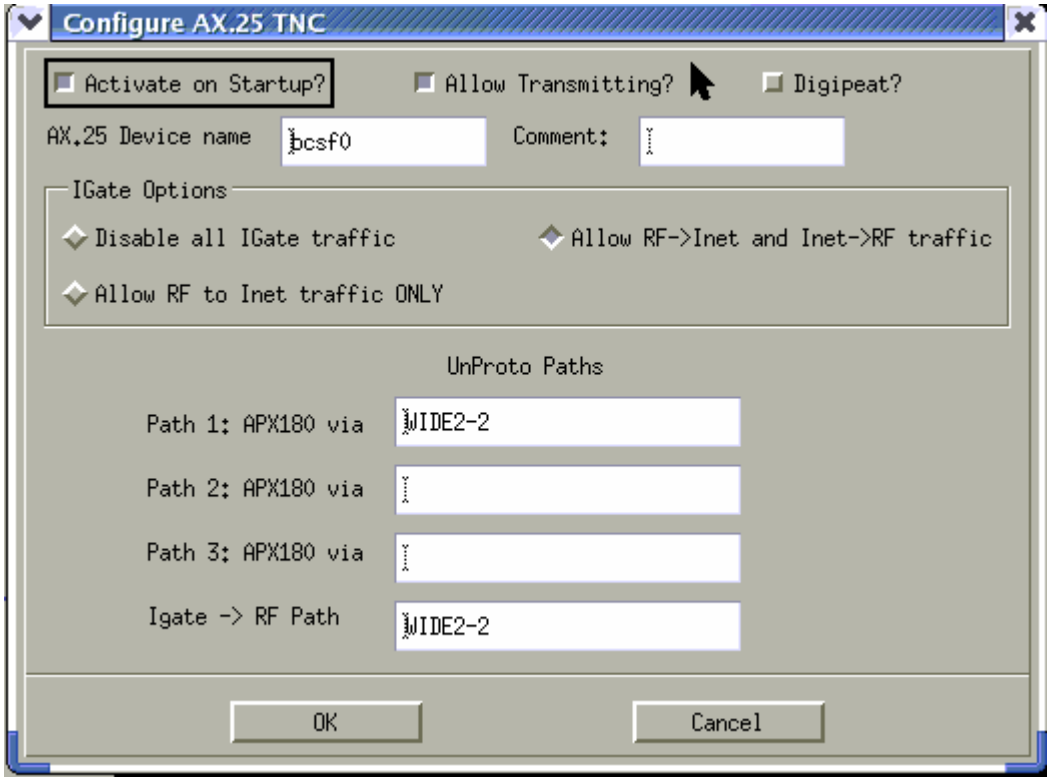

Figure 2 - Sample Xastir port configuration for Baycom modem

6 Compiling APRSD

aprsd is a server daemon that provides Internet gateway and client access to amateur radio APRS packet data. This section lists the steps I followed to make APRSD work with the Baycom interface.

Aprsd can be downloaded from http://sourceforge.net/projects/aprsd/

Extract the files, change to the aprsd directory and follow the instructions in the INSTALL file. Basically the simplest way to compile the package is by running the following commands:

1. ./configure
2. make
3. make check (optional)
4. make install
5. make clean

Of course, you will need to execute these commands with a user with the appropriate access level. It will work if you do it as root, but it might also work if you assign the required privileges to another user. Note that by default aprsd uses the configuration files in the directory in which you extracted it. There must be a way to change this, but I didn't try it.

I was able to build APRSD without any problems. After compiling, I edited the **admin/aprsd.init** file to change the paths to the executables to the directory where aprsd was installed (in my case **/usr/local/bin**)

After that I edited the aprsd.conf file to set the configuration parameters.

Below is a summary of the changes I made. They are in BOLD, in the sample aprsd.conf file, some followed by an arrow and some comments (which of course you should not include in your config file). I don't know if I'm 100% correct, but it worked:

```
# $Id: aprsd.conf,v 1.15 2003/10/01 16:57:16 kg4ijb Exp $
#aprsd 2.2.5 server configuration file
#
```

```
#This file is read ONCE on server startup.
#You must restart aprsd for changes to take effect.
#eg: /etc/rc.d/init.d/aprsd.init stop (then start)
#
#Lines starting with "#" are comments and are ignored
#Key words such as "mycall" and "maxusers" are NOT case sensitive.
#MyCall is the same as mycall.
#
#*** There is no error checking so be careful ******
#
#
#Servercall is the ax25 source call used in packets
#sent from the server to Internet users. (9 chars max)
#Note: Does not go out on the air.
#
servercall 5B8AP-5
#
#MyCall will be over written by the MYCALL string in INIT.TNC
#if "tncport" has been defined.
#
MyCall 5B8AP-5
#
MyLocation Nicosia_Cyprus
#
#This email address will be sent in replies to ?IGATE? queries.
# Also, it will be on the status web pages on port 14501.
MyEmail 5b8ap@qsl.net
#
#Set MaxUsers and MaxLoad to values that your Internet connection can
support.
# Set MaxLoad in bytes/sec.   If either value is exceeded
# new users can't connect.
MaxUsers 15
MaxLoad 100000000
#
#
#Define beacon text. The server will supply the ax25 path header.
#The first number after "NetBeacon" is the time interval in minutes.
#Comment out the line or set time interval to 0 to disable beacon.
#The rest of the line can be any aprs protocol conforming packet.
#
NetBeacon 10 !3508.20NI03318.17E& APRS TEST Server
#
#Define the TNC beacon. The TNC will supply the ax25 path header.
#It's optional and you may use the TNC BTEXT in the INIT.TNC file instead.
#
TncBeacon 10 !3508.20N/03318.17E& Linux APRS Test Server
#
#
#Send 2 extra message acks is addition to each received ack to TNC
#Range 0 to 9
ackrepeats 2
#
#Send extra acks at 5 second intervals
#Range 1 to 30 seconds
ackrepeattime 5
#
#Set history list items to expire in 35 minutes
expire 35
#
#Define the TNC serial port and baud rate.
#Note: This device must have write permissions
#If undefined all TNC related functions are disabled.
#Permissable baud rates are 1200,2400,4800,9600 and 19200.
```

```
#
#tncport /dev/ttyS0
#tncport radio
tncport bcsf0  ← these are the settings for the Baycom modem on COM1
tncbaud 1200
tncport mfjtnc ← these are the settings for the KISS TNC
tncbaud 9600


#Define the path for transmitted packets
#This is only used when using Linux sockets, not the TNC.
#For the TNC, set this in INIT.TNC. Note the format is
#slightly different to the TNC command.
aprspath APRS v WIDE5-5

#
# Allow Internet to RF message passing.
rf-allow yes
#
#Set filterNoGate "yes" to block RFONLY and NOGATE packets
filterNoGate yes
#
#Set history-allow to NO if you do not want users to get history dumps.
history-allow no
#
#TRACE causes the server to append its own callsign to the end
#of the AX25 path of every packet processed.
#To conserver bandwidth this should only be
#done for short periods to track sources of problems.
#
Trace no
#
#Set this to 'yes' if you want to log ALL PACKETS heard on RF to
/home/aprsd2/rf.log
#If 'no' then only packets with your callsign will be logged.
logAllRF yes
#
# Allow the insecure aprs passcodes to be used
# Note: "no" means all users need Linux user names and passwords
#       and aprsd must be run as root for that to work.
aprsPass yes
#
#Set the minimum time between TNC transmit packets in milliseconds
TncPktSpacing 1500
#
# Disallow packets transmitted from our own TNC from
# being igated back to the Internet after being digipeated.
igateMyCall no
#
#This determines if Mic-E packets are converted to classic APRS packets.
#Put 'no' unless you have a very good reason to do conversions.
#This option must also be enabled in the SOURCE CODE. To turn it on
#you must edit "constant.h" and change CONVERT_MIC_E from FALSE to TRUE.
#then recompile aprsd.
ConvertMicE no
#
#
#The PASS command.  The callsign supplied in MyCall and this
# passcode allow you to send data to distant servers.
# PASS can be computed from MYCALL with the aprspass program.
# Note: This example is invalid. Use your passcode.
#
pass 123123  ← Use the “callpass” command to get your passcode
#
```

```
#------------------------------
#Server connection definitions
#
#usage: <Server> <host name> < host port> <TYPE-DIR> <optional OmniPort
filter command>
#
#The TYPE-DIR field sets the connecton type and data flow.
# TYPE is  either "SERVER" or "HUB"
#  SERVER connections attempts to maintain a connection to the designated
host
#  HUB connections maintain a connection to only ONE hub and rotate to the
next
#  if the connectin fails.
# DIR is either "RO" or "SR"  RO is Receive Only. SR is Send and Receive.
#Eamples: HUB-RO   HUB-SR  SERVER-RO SERVER-SR
# If you select -SR to send data you must also supply a passcode
# using the "PASS" command. See above.
#
#These commands are NOT case sensitive.
# ** Defaults to Server-RO if TYPE-DIR not specified. **
#
#
#Example send-receive  HUB connections
#Hub is like Server except only ONE hub connection is active at a time.
#If the hub connection fails the next hub is tried in rotation until one
accepts a connection.
#
#Server second.aprs.net 23 hub-sr
Server first.aprs.net 23 hub-sr
#Server 128.143.202.191 23 hub-sr
Server third.aprs.net 23 hub-sr
#Server rotate.aprs.net 23 hub-sr
#
#
#Example Receive-Only HUB connecton
#Server first.aprs.net 23 hub-ro
#
#Example Send-Receive SERVER connection.  The SERVER type maintains a
connection to
#the specified server.  No rotation.
#Server first.aprs.net 23 server-sr
#
#Example of OmniPort connection to receive the local stream (tnc).
#Note: OmniPort is currently available only on aprsd 2.1.5
#
#Server atlanta.aprs.net 14600 server-ro portfilter(local)
#
#Receive the full stream.
#Server atlanta.aprs.net 14600 server-ro portfilter(full)
#
#
#These servers are provided as a reference. Some may no longer be active.
#You really only need the two or three HUB connections above unless
# you are doing something unusual.
#
#server second.aprs.net 1313 server-ro
#server third.aprs.net 1313 server-ro
#server socal-igate.aprs.claremont.edu 1313 server-ro
#server first.aprs.net 23
#server atlanta.aprs.net 1313 server-ro
#server aprsdkw.dyndns.org 1313
#server aprs.k9iu.ampr.org 1313
#server atlanta.aprs.net 1313
#server baltimore.aprs.net 1313
```

```
#server calgary.aprs.net 1313
#server concord.aprs.net 1313
#server cosprings.aprs.net 1313
#server dayton.aprs.net 14439
#server elansing.aprs.net 1313
#server gw.officine.it 1313
#server hamgate.cs.usu.edu 1313
#server kb2ear.aprs.net 1313
#server kd6wxd.dynip.com 10148
#server marconi.ece.cmu.edu 14579
#server michigan.aprs.net 1313
#server milwaukee.aprs.net 1313
#server montreal.aprs.net 1313
#server mulvey.dyndns.com 1313
#server netherlands.aprs.net 1313
#server newyork.aprs.net 1313
#server ontario.aprs.net 1313
#server orlando.aprs.net 1313
#server radio.artsfac.csuohio.edu 1313
#server radio-active.net.au 1313
#server sandiego.aprs.net 1313
#server santabarbara.aprs.net 1313
#server saopaulo.aprs.net 1313
#server swiss.aprs.net 14579
#server temple.aprs.net 1313
#server tucson.aprs.net 1313
#server wv.aprs.net 1313
#--------------------------------
#
#Define server listen ports
#Read ports.html for more info.
#
rawtncport 14580
localport 14579
mainport 10151
mainport-nh 10152
linkport 1313
msgport 1314
udpport 1315
sysopPort 14500
httpport 14501
ipwatchport 14502
errorport 14503
omniport 14600
#
#define trusted users of the UDP port.
#usage: trust <ip address> <subnet mask>
#trust 208.148.145.151
#trust 208.148.145.144 255.255.255.240
#
#Selected call signs which are always gated to RF
#if they are not seen locally. All packets from
#these are gated in real time. Do not use unless
#you really need real time data.  Consider posit2rf below.
#They are case sensitive! Use upper case. Up to 64 may be defined.
#As of version 2.1.5 the * wild card character is supported.
#All characters beyond the * are ignored.
#ie: WA4* would match ALL call signs beginning with "WA4"
#
# This has been enhanced in 2.2.5-15 to include
#
#   Instead of only having "*" you can use
#
#   $ B for one single alpha character
```

```
#   # B for one single numeric character
#   . B for one single punctation character
#   ? B for one single alphanumeric character
#   * B for any number and kind of characters (like before)
#
#   Example:
#   Patterns like B "D$#$*" only allow german callsigns (starting with D,
#   followed by 1 alpha, 1 num, 1 alpha and anything else) but it does not
#   match "DALLAS" or something like that which occurs without this change
#   when configuring just "D*"
#
gate2rf 5B* SW* SV*  ← callsigns you want to gate to RF
#
#Call signs of stations whose posits are gated
#to RF every 15 minutes.  Only posit packets are
#gated.  Posits are taken from the history list.
#They are case sensitive! Use upper case.
#
#posit2rf K4HG-8
#
#Define a list of message destination call signs or aliases
#to gate to RF full time.  Note: the CQGA example
#below is CQ GA (Georgia). Edit to suite your locale.
#Up to 64 of these may be defined. They are case sensitive.
#
msgdest2rf SCOUTS KIDS CQGA 5B* SV* SW*
#
#end
```

You can also edit the welcome.txt file to change the welcome message that appears when somebody connects to your server.

You can check the status of your server by pointing a browser to
http://[your server's IP or hostname]:14501/

In a similar fasion, you can check the status of the servers to which you selected to connect (in this example **first.aprs.net** and **third.aprs.net** to see if your callsign is listed in the client connections.

Once you have it running, you can connect to your server on port 14580.

7 Utilities

Sometimes APRSD crashes and has to be restarted. In order to avoid extended downtime of the Igate, in case I fail to notice, I wrote a script that checks if APRSD is running and if not tries to start it. The script is scheduled as a **cron** job and runs every 15 minutes. Notification functionality is also included in the script.

The script is something like the below: (parts in red are optional)

```
if ps ax | grep aprsd | grep -v grep
then
echo "APRSD is running"
else
echo "APRSD is down"
#call a Web address that will notify you that APRSD is down. This can be a servlet,
#PHP script etc. You can also change this to send you a notification by email.
wget -b "http://[address to call when APRSD is down]"&
echo "trying to enable APRSD"
#you will have to change the path accordingly
cd /home/andreas/ax25/aprsd-2.2.5-15/admin/
./aprsd.init start
sleep 60
#check if APRSD has been enabled
if ps ax | grep aprsd | grep -v grep
then
echo "APRSD has been enabled"
#call a Web address that will notify you that APRSD is running. This can be a
#servlet, #PHP script etc. You can also change this to send you a notification by
#email.
wget -b "http://[address to call when APRSD has been enabled]"&
#remove the temporary files greated by wget
#You will have to change the paths to match your installation
#Additional temporary files may be created,
#based on the addresses you call with wget
rm /home/andreas/ax25/wget-log*
rm /home/andreas/ax25/aprsd-2.2.5-15/admin/wget-log*
fi
fi
```

Schedule the above to run as a **cron** job under root or another user that has access to the serial port and can start **APRSD**.

```
# crontab -l
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.30301 installed on Fri May  7 22:28:33 2010)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
0,15,30,45 * * * * /home/andreas/ax25/chkAPRSD
```

For more details on cron, see the cron man page.

8 Where to get more information

More information about the tasks described in this document can be found in:

- Linux Amateur Radio AX.25 HOWTO
  - http://www.faqs.org/docs/Linux-HOWTO/AX25-HOWTO.html
- APRSD Home page
  - http://sourceforge.net/projects/aprsd/
- Xastir Home Page
  - http://www.xastir.org/
- AX.25 library and other software
  - http://sourceforge.net/project/showfiles.php?group_id=24545
- Baycom Modems
  Below are some links I found on the Web. You should be able to find more related info if you search a little.

- http://www.qsl.net/sv1bsx/Baycom/baycom.html
- http://web.telia.com/~u85920178/use/baycom.htm
- http://users.belgacom.net/hamradio/schemas/baycom_on1dht.gif
- http://home.concepts-ict.nl/~cybersake/baycom.htm

- If all else fails… try Google ☺