Obtaining the best IPv6 support with Linux

Mauro Tortonesi
Deep Space 6

mauro@deepspace6.net

This is a brief article that explains how to enhance the IPv6 support of your Linux operating system to the best possible status.

1. Introduction

This is a brief article that explains how to enhance the IPv6 support of your Linux operating system to the best possible status. It should be of some interest to you only if you are __seriously__ planning to do some research or experiments with the IPv6 protocol on Linux. If this is not your case, then you probably won't need to follow the procedure described below, and you'll surely find more useful informations in Peter Bieringer's Linux+IPv6 HOWTO (http://www.tldp.org/HOWTO/Linux+IPv6-HOWTO/).

Most of the latest Linux distributions are already IPv6-ready (http://www.bieringer.de/linux/IPv6/status/IPv6+Linux-status-distributions.html), (at least for the default kernel, the initialization scripts and the system libraries). However, the IPv6 support in the kernel and the other basic packages is far from being perfect: the standard linux kernel 2.4.X lacks some advanced IPv6 features and the implementation of some of the extended BSD sockets API in glibc 2.2 is not up-to-date with the latest IETF drafts. In most of the cases, these are only minor problems, but you can get rid of them with the following procedure.

> **Warning**
>
> The instructions below should be executed only by an expert user. Upgrading the kernel and updating basic software like the BSD socket library is a delicate procedure, and potentially dangerous. So, you have been warned: follow the next steps with caution and at your own risk.

2. Installing the USAGI kit

First, you have to download the latest release of the USAGI kit, which contains an enhanced IPv6-enabled version of both the latest 2.4.x Linux kernel and the most important networking software packages for Linux, from the USAGI project website (http://www.linux-ipv6.org).

The USAGI project developers release two different versions of their kit: the stable version, which is released two or three times a year, and the CVS snapshot version, which is supposed to be unstable and is released every two weeks. If you feel very brave, i would recommend you to download and install the unstable version, which is always up-to-date and has never given me any big problem (only a few minor annoyances with a couple of errors in the makefiles - but that was long ago).

So, the time has come to connect to the USAGI project ftp repository (ftp://ftp.linux-ipv6.org), and download the kit. You will find the stable release of the kit in this directory (ftp://ftp.linux-ipv6.org/pub/usagi/kit/stable) and the CVS snapshot version in this directory (ftp://ftp.linux-ipv6.org/pub/usagi/kit/snap).

Next, unpack the package you have downloaded and prepare the kit for compilation with the command:

make prepare TARGET=linux24

2.1. Building and installing a new kernel

Now we're ready to build the kernel. Enter in the kernel/linux24 directory and configure the kernel according to your needs, with:

make xconfig

or:

make menuconfig

If you are compiling a recent snapshot kits (at the moment of writing the last kit has been released on Monday, January 6th 2003), a good kernel configuration for IPv6 networking could be:

```
#
# Networking options
#
CONFIG_PACKET=m
CONFIG_PACKET_MMAP=y
CONFIG_NETLINK_DEV=m
```

```
CONFIG_NETFILTER=y
# CONFIG_NETFILTER_DEBUG is not set
CONFIG_FILTER=y
# CONFIG_NET_NEIGH_DEBUG is not set
CONFIG_NET_RESTRICTED_REUSE=y
CONFIG_UNIX=m
CONFIG_INET=y
# CONFIG_IPSEC is not set
CONFIG_IP_MULTICAST=y
# CONFIG_IP_ADVANCED_ROUTER is not set
# CONFIG_IP_PNP is not set
CONFIG_NET_IPIP=m
CONFIG_NET_IPGRE=m
# CONFIG_NET_IPGRE_BROADCAST is not set
# CONFIG_IP_MROUTE is not set
# CONFIG_ARPD is not set
# CONFIG_INET_ECN is not set
CONFIG_SYN_COOKIES=y
CONFIG_IPV4_IPSEC_TUNNEL=y

#
#   IP: Netfilter Configuration
#
CONFIG_IP_NF_CONNTRACK=m
CONFIG_IP_NF_FTP=m
CONFIG_IP_NF_IRC=m
CONFIG_IP_NF_QUEUE=m
CONFIG_IP_NF_IPTABLES=m
CONFIG_IP_NF_MATCH_LIMIT=m
CONFIG_IP_NF_MATCH_MAC=m
CONFIG_IP_NF_MATCH_PKTTYPE=m
CONFIG_IP_NF_MATCH_MARK=m
CONFIG_IP_NF_MATCH_MULTIPORT=m
CONFIG_IP_NF_MATCH_TOS=m
CONFIG_IP_NF_MATCH_ECN=m
CONFIG_IP_NF_MATCH_DSCP=m
CONFIG_IP_NF_MATCH_AH_ESP=m
CONFIG_IP_NF_MATCH_LENGTH=m
CONFIG_IP_NF_MATCH_TTL=m
CONFIG_IP_NF_MATCH_TCPMSS=m
CONFIG_IP_NF_MATCH_HELPER=m
CONFIG_IP_NF_MATCH_STATE=m
CONFIG_IP_NF_MATCH_CONNTRACK=m
CONFIG_IP_NF_MATCH_UNCLEAN=m
CONFIG_IP_NF_MATCH_OWNER=m
CONFIG_IP_NF_FILTER=m
CONFIG_IP_NF_TARGET_REJECT=m
CONFIG_IP_NF_TARGET_MIRROR=m
CONFIG_IP_NF_NAT=m
CONFIG_IP_NF_NAT_NEEDED=y
CONFIG_IP_NF_TARGET_MASQUERADE=m
CONFIG_IP_NF_TARGET_REDIRECT=m
CONFIG_IP_NF_NAT_LOCAL=y
```

```
CONFIG_IP_NF_NAT_SNMP_BASIC=m
CONFIG_IP_NF_NAT_IRC=m
CONFIG_IP_NF_NAT_FTP=m
CONFIG_IP_NF_MANGLE=m
CONFIG_IP_NF_TARGET_TOS=m
CONFIG_IP_NF_TARGET_ECN=m
CONFIG_IP_NF_TARGET_DSCP=m
CONFIG_IP_NF_TARGET_MARK=m
CONFIG_IP_NF_TARGET_LOG=m
CONFIG_IP_NF_TARGET_ULOG=m
CONFIG_IP_NF_TARGET_TCPMSS=m
CONFIG_IP_NF_ARPTABLES=m
CONFIG_IP_NF_ARPFILTER=m
# CONFIG_IP_NF_COMPAT_IPCHAINS is not set
# CONFIG_IP_NF_COMPAT_IPFWADM is not set
CONFIG_IPV6=m
# CONFIG_IPV6_DEBUG is not set
CONFIG_IPV6_IM=y
CONFIG_IPV6_MODULE_IP_GRE=y
# CONFIG_IPV6_ZONE is not set
CONFIG_IPV6_DROP_FAKE_V4MAPPED=y
CONFIG_IPV6_RESTRICTED_DOUBLE_BIND=y
# CONFIG_IPV6_6TO4_NEXTHOP is not set
CONFIG_IPV6_PRIVACY=y
CONFIG_IPV6_ANYCAST=y
CONFIG_IPV6_ANYCAST_GROUP=y
CONFIG_IPV6_ISATAP=y
# CONFIG_IPV6_PREFIXLIST is not set
# CONFIG_IPV6_SUBTREES is not set
# CONFIG_IPV6_MLD6_ALL_DONE is not set
CONFIG_IPV6_NODEINFO=y
# CONFIG_IPV6_NODEINFO_USE_UTS_DOMAIN is not set

#
#   IPv6: Netfilter Configuration
#
CONFIG_IP6_NF_QUEUE=m
CONFIG_IP6_NF_IPTABLES=m
CONFIG_IP6_NF_MATCH_LIMIT=m
CONFIG_IP6_NF_MATCH_MAC=m
CONFIG_IP6_NF_MATCH_RT=m
CONFIG_IP6_NF_MATCH_OPTS=m
CONFIG_IP6_NF_MATCH_FRAG=m
CONFIG_IP6_NF_MATCH_MULTIPORT=m
CONFIG_IP6_NF_MATCH_OWNER=m
CONFIG_IP6_NF_MATCH_MARK=m
CONFIG_IP6_NF_MATCH_AHESP=m
CONFIG_IP6_NF_MATCH_EUI64=m
CONFIG_IP6_NF_MATCH_LENGTH=m
CONFIG_IP6_NF_MATCH_EUI64=m
CONFIG_IP6_NF_FILTER=m
CONFIG_IP6_NF_TARGET_LOG=m
CONFIG_IP6_NF_TARGET_REJECT=m
```

```
CONFIG_IP6_NF_MANGLE=m
CONFIG_IP6_NF_TARGET_MARK=m
CONFIG_IPV6_IPSEC_TUNNEL=y
# CONFIG_IPV6_IPV6_TUNNEL is not set
```

In order to avoid conflicts with the other kernels installed on your system and to be able to track what version of the USAGI-enhanced kernel you are using, I recommend you to edit the file kernel/linux24/Makefile and change the fourth line from:

```
EXTRAVERSION =
```

to something like:

```
EXTRAVERSION = usagisnap20030106
```

which is a good choice if you are installing the unstable kit released on Monday, January 6th 2003.

Now, you can build and install the kernel in the usual manner. For more information on how to accomplish this task take a look at the Linux Kernel HOWTO (http://www.linuxdoc.org/HOWTO/Kernel-HOWTO/).

Warning

If you configure the IPv6 support as a kernel module, remember to add the following line:

```
alias net-pf-10 ipv6
```

to /etc/modules.conf.

2.2. Installing the networking software

Now, we're going to install the userspace networking software which is included in the kit. Switch to the usagi directory and configure the software with:

./configure --enable-bindtest --enable-ndp --enable-v6p

Then, build the software with:

make

and (as root) install it with:

make install

The software will be installed by default in /usr/local/v6 (which is a very good choice).

2.3. Installing libinet6 as a shared library

If you are a developer, or simply if you wish to obtain the best IPv6 compliance for your system, you may want to install the libinet6 library included in the USAGI kit as a shared library (when we built the userspace software of the USAGI kit, we also built the static version of the libinet6 library), and let all the software installed on your system make use of it.

In fact, libinet6 provides an implementation of the Extended BSD Socket API which is up-to-date with the latest internet-drafts.

First, you have to uncomment the following lines from the file usagi/libinet6/Makefile:

line 94:

```
#shared: includes include_glibc$(usagi_libc) $(SLIBS)
```

lines 106-108:

```
#install-shared: shared include_glibc$(usagi_libc)_install
#         $(INSTALL_DIR) $(slibdir)
#         $(INSTALL_LIBRARY) $(SLIBS) $(slibdir)
```

Then, you have to build and install the software:

make shared make install-shared

This will build a shared version of the library and install it in /usr/local/v6/lib (which, again, is a wise choice - so i suggest you to keep the default installation path).

Next, we have to replace some of the include files installed in your system with the ones shipped with libinet6. This is very simple. Digit:

make install-includes

and the task will be accomplished in a few milliseconds.

> **Warning**
>
> BEWARE: make install-includes will overwrite some files in /usr/src/include. It is a VERY GOOD idea to make a backup copy of /usr/src/include before installing libinet6.

Now that you've installed libinet6, you have to force the software installed on your system to make use of it. All you have to do is to add the following line to /etc/ld.so.preload:

```
/usr/local/v6/lib/libinet6.so
```

and the following line to /etc/ld.so.conf:

```
/usr/local/v6/lib
```

Now, as soon as you'll have reconfigured the dynamic linker with:

ldconfig

this will force the load of libinet6 in the address space of every process at the execution time. In result, all the software installed in your system will automatically make use of libinet6 and you won't need to do any recompilation.

Interesting links

[USAGI] The USAGI project (http://www.linux-ipv6.org) .

[KERNELHOWTO] *The Linux Kernel HOWTO (http://www.tldp.org/HOWTO/Kernel-HOWTO/)* , Linux Documentation Project, Brian Ward and Alavoor Vasudevan.

[IPv6HOWTO] *The Linux+IPv6 HOWTO (http://www.tldp.org/HOWTO/Linux+IPv6-HOWTO/)* , Linux Documentation Project, Peter Bieringer.