IPv6: Linux IPv6 Gateway (6to4 Tunnel)

Unfortunately many low cost routers deployed with broadband subscriptions do not support IPv6. There are more advanced routers that already feature the new generation IP protocol. If you happen to have a Cisco router you may follow **the Cisco router 6to4 tutorial** to enjoy the IPv6 Internet.

In case you do not have an IPv6 capable router you can still get IPv6 Internet connection by deploying a Linux system as your Internet gateway for IPv6. It does not necessarily need to replace your existing installation but can be deployed as an add-on. There is one requirement that needs to be met: The Linux system must have a public IPv4 address on one of its interfaces.

Objective

The goal is simple: IPv6 connectivity from the local network to the Internet. With a 6to4 tunnel (automatic transport of IPv6 in IPv4) every public IPv4 address opens up a /48 IPv6 prefix which provides 80 bit for host addressing in the local network (compare this number to the 32 bit of the entire IPv4 address space). Everyone on this planet who is connected to the Internet can build a huge IPv6 network by setting up a 6to4 gateway which is exactly what this tutorial is about.

6to4 Tunnel Intro

Few service providers offer IPv6 connectivity yet. To have access to IPv6 sites it is therefore necessary to tunnel IPv6 traffic through the IPv4 Internet. To prevent a manually configured tunnel for each site, 6to4 tunnels "automatically build" these tunnels.

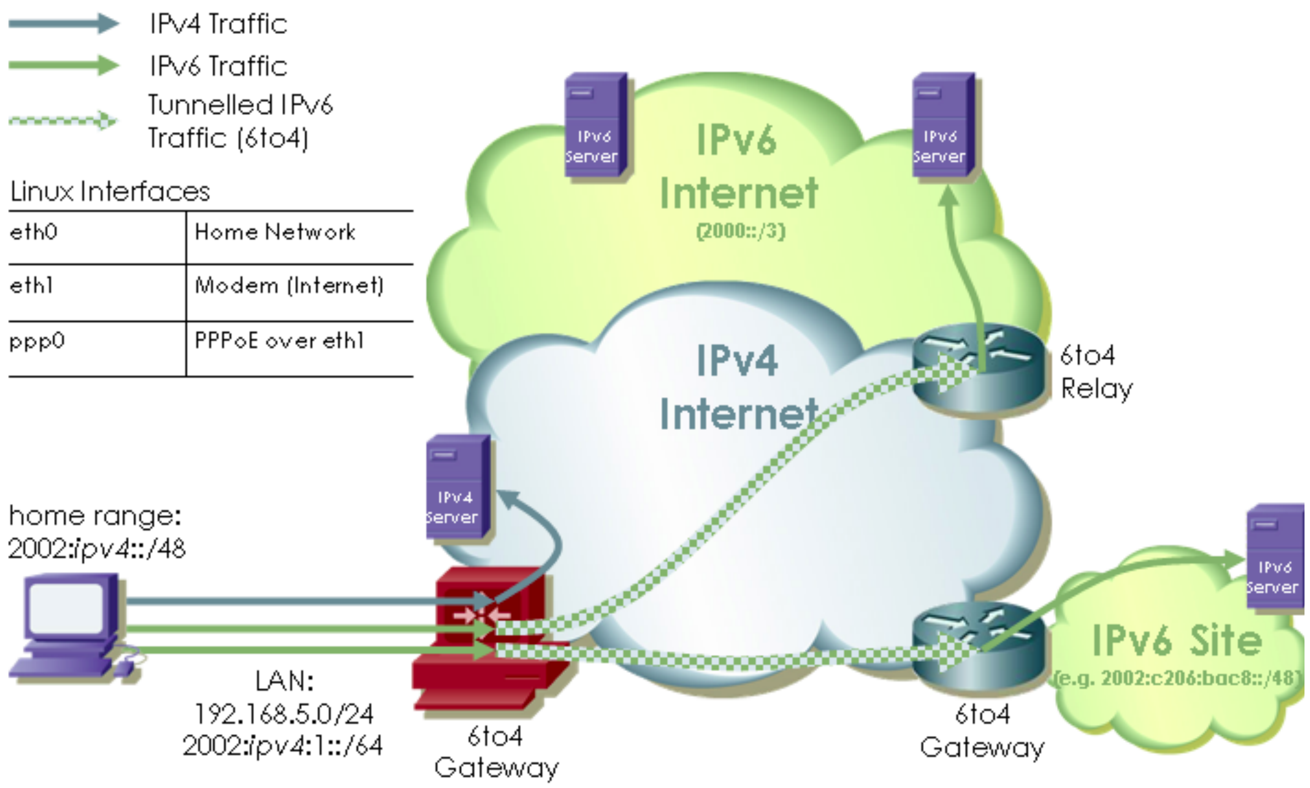
The trick is simple: The tunnel endpoint is embedded in the IPv6 address. A global prefix, assigned by IANA, indicates to a router that it is a 6to4 address and it therefore must encapsulate the IPv6 packet and forward it to the embedded IPv4 address.

A 6to4 local network has the following prefix: 2002:`ipv4-address`::/48. The embedded IPv4 address must be routable to reach the endpoint of the tunnel. This is the reason why this address must be a public (official) one.

To reach the entire IPv6 Internet a device is required that accepts your automatic tunnels and is able to forward the encapsulated IPv6 packets to the IPv6 Internet. This is the task of a 6to4 relay. They have been widely deployed and are reachable on 192.88.99.1 everywhere you are. Other relays are available too but you need to know their IPv4 addresses.

Network Layout

This network sketch shows the required communication paths for IPv6 Internet access. 6to4 sites are directly tunnelled to by using the embedded IPv4 address. The rest of the IPv6 address space can be reached through the 6to4 relay.



Linux Interfaces

eth0	Home Network
eth1	Modem (Internet)
ppp0	PPPoE over eth1

Prerequisites

The following steps configure the (red) Linux router. It is assumed that the Linux system already is configured for:

- IPv4 connection to the Internet with a public IPv4 address (the configurations below use PPPoE to the service provider).
- Name server entries in *`resolv.conf`* (either static or dynamic).
- An interface to the LAN side of your local network (the tutorial uses the network 192.168.1.0/24 with address 192.168.1.1 configured on the Linux interface).

The Linux distribution used in this tutorial is Debian Etch. Ubuntu should be the same but it has not been tested.

Configuration

The following steps are required:

- Configure Linux to forward (route) IPv6.
- Create the 6to4 tunnel.
- Assign IPv6 addresses to the interfaces on the Linux system.
- Configure IPv6 routing.
- Advertise IPv6 prefixes to the end systems.
- Configure the Linux as a DNS proxy.

Each task is described in the following sub-chapters. At the end you will find a script that does all the steps as soon as the WAN interface comes up.

IPv6 Routing

By default, routing is disabled on Linux systems. IPv4 routing is not really required to meet our goal but it is enabled anyway in case you also want to use the system as your IPv4 gateway to the Internet.

```
# sysctl -w net.ipv4.conf.default.forwarding=1
# sysctl -w net.ipv6.conf.default.forwarding=1
```

To make it permanent update the following file (the configuration lines are already prepared by default but are commented).

```
/etc/sysctl.conf
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.conf.default.forwarding=1

# Uncomment the next line to enable packet forwarding for IPv6
net.ipv6.conf.default.forwarding=1
```

6to4 Tunnel

To create the tunnel we need to know the IPv4 address of the WAN connection.

```
# ip -4 addr show dev ppp0
19: ppp0:  mtu 1460 qdisc pfifo_fast qlen 3
   inet 203.211.79.44 peer 202.74.206.11/32 scope global ppp0
```

With the IPv4 address in hand the 6to4 tunnel can now be created. The tunnel type is called SIT (Simple Internet Transition) on Linux. The created interface will have the name `tun6to4`.

```
# ip tunnel add tun6to4 mode sit remote any local 203.211.79.44
```

The tunnel is now ready to be activated. At the same time the MTU for the tunnel is set. 8 byte is used for the PPPoE header already. The IPv6 packets are encapsulated in IPv4 on the tunnel. The standard size of an IPv4 header is 20 bytes. The MTU is therefore the normal Ethernet MTU (1500) minus the headers used on the tunnel.

```
# ip link set dev tun6to4 mtu 1472 up
```

The following display shows the tunnel up and running.

```
# ip link show dev tun6to4
21: tun6to4@NONE:  mtu 1472 qdisc noqueue
   link/sit 203.211.79.44 brd 0.0.0.0
```

Addressing

In fact an IPv6 routing device doesn't necessarily need IPv6 global addresses at all because every interface already has an IPv6 address once they become active (the link-local address). The link-local addresses are used in the forwarding process of a router.

It is however quite convenient if an global IPv6 address is available on the interface. Management of the device is the main reason for this but our Linux may also have additional tasks in the future where it needs to have a global address (e.g. VPN concentrator).

Therefore global addresses are assigned to the interfaces. Even two global addresses go on the local LAN interface: the 6to4 derived address and a unique local unicast address. This type of address is like a private address that can be used within the site.

The 6to4 address is a combination of the 6to4 prefix (`2002`::/16), the IPv4 address of the WAN interface (`203.211.79.44` -> hexadecimal ::`CBD3:4F2C`::), the site-level aggregator (SLA or "subnet"; we chose `0` for the WAN interface and `1` for the LAN interface) and the host. The prefix is /64.

```
# ip -6 addr add dev tun6to4 2002:cbd3:4f2c:0::1/64
# ip -6 addr add dev eth0 2002:cbd3:4f2c:1::1/64
```

The unique local unicast starts with the prefix FC00::/7, the 8th bit must be 1 for locally assigned addresses followed by 40 random bits (see RFC 4193). After this /48 prefix we again add the SLA and the host.

```
# ip -6 addr add dev eth0 fdcb:3fb1:918:1::1/64
```

FC00::/7 became FD because the 8th bit must be 1.

Looking at the interfaces we now see these addresses configured. In addition to the manually configured addresses, the link-local and the tunnel mapped address are also present.

```
# ip -6 addr show dev tun6to4
21: tun6to4@NONE:  mtu 1472
   inet6 2002:cbd3:4f2c::1/64 scope global
     valid_lft forever preferred_lft forever
   inet6 ::203.211.79.44/128 scope global
     valid_lft forever preferred_lft forever
# ip -6 addr show dev eth0
10: eth0:  mtu 1500 qlen 1000
   inet6 fdcb:3fb1:918:1::1/64 scope global
     valid_lft forever preferred_lft forever
   inet6 2002:cbd3:4f2c:1::1/64 scope global
     valid_lft forever preferred_lft forever
   inet6 fe80::208:74ff:fee5:aa4f/64 scope link
     valid_lft forever preferred_lft forever
```

IPv6 Routing

As long as the local LAN only consists of directly connected LAN segments only two routes are required: a route to other 6to4 sites and a default route. Actually the default route is sufficient but routing to other 6to4 sites is not as efficient in this case.

The route to other 6to4 sites just points to the tunnel interface. So does the default route but the next hop for it is the mapped IPv4 address which is the anycast address used for the 6to4 relay.

```
# ip -6 route add 2002::/16 dev tun6to4
# ip -6 route add ::/0 via ::192.88.99.1 dev tun6to4 metric 1
```

The IPv6 routes now look as follows:

```
# ip -6 route show
::/96 via :: dev tun6to4  metric 256  ...
2002:cbd3:4f2c::/64 dev tun6to4  metric 256  ...
2002:cbd3:4f2c:1::/64 dev eth0  metric 256  ...
2002::/16 dev tun6to4  metric 1024  ...
default via ::192.88.99.1 dev tun6to4  metric 1  ...
```

Reaching other IPv6 hosts should now be possible. Try the anyweb website (2002:3cea:4704::1).

```
# ping6 2002:3cea:4704::1
PING 2002:3cea:4704::1(2002:3cea:4704::1) 56 data bytes
64 bytes from 2002:3cea:4704::1: icmp_seq=1 ttl=64 time=224 ms
```

Advertise Prefixes

Clients on the local LAN should be able to autoconfigure. To do so they must receive information from the Linux router (prefix, next hop, lifetimes, etc.). There are several possibilities on Linux of which *radvd* (router advertisement daemon) is setup here.

```
# apt-get install radvd
```

Unfortunately Debian Etch comes with version radvd 1.0 which lacks the possibility to just tell it to announce

all IPv6 prefixes configured on the interface (should be possible in version 1.1). Therefore we need to add the prefixes explicitly in the configuration file.
For 6to4 addresses there is the possibility to make the announcements dependent on an IPv4 address on an interface. Because this address most likely changes frequently this feature is very welcome.

```
/etc/radvd.conf
```

```
interface eth0 {
    AdvSendAdvert on;
    MinRtrAdvInterval 5;
    MaxRtrAdvInterval 15;
    prefix fdcb:3fb1:918:1::/64 {
    };
    prefix 0:0:0:1::/64 {
        AdvOnLink off;
        AdvAutonomous on;
        AdvRouterAddr on;
        Base6to4Interface ppp0;
        AdvPreferredLifetime 30;
        AdvValidLifetime 60;
    };
```

radvd announces the unique local unicast address with just the default values. For the 6to4 derived prefix the lifetimes are changed to account for the dynamic IPv4 address (and therefore the change of the 6to4 prefix). To announce a 6to4 prefix the first 48 bits in the prefix statement must be 0, followed by the SLA. In addition, the parameter *`Base6to4Interface`* must be coded pointing to the WAN interface.

Start radvd to fetch the new configuration and start advertising.

```
# /etc/init.d/radvd start
```

The clients now receive the advertisements from the router and configure their interfaces. The following is an example of a Linux client on the LAN network.

```
user@client:~$ ip -6 addr show
2: eth0:  mtu 1500 qlen 1000
   inet6 fe80::21e:c9ff:fe02:6036/64 scope link
     valid_lft forever preferred_lft forever
   inet6 2002:cbd3:4f2c:a:21e:c9ff:fe02:6036/64 scope global dynamic
     valid_lft 52sec preferred_lft 22sec
   inet6 fdcb:3fb1:918:1:21e:c9ff:fe02:6036/64 scope global dynamic
     valid_lft 2591992sec preferred_lft 604792sec
```

Note the different lifetimes for the unique local address and the 6to4 address.

DNS Proxy

This is an optional step and is not required if your clients have IPv4 connectivity to the DNS server. However in this tutorial the clients are (almost) pure IPv6 hosts but some of the clients are not able to configure an IPv6 address as their DNS server (e.g. WinXP). With a DNS proxy on the Linux router they can locally resolve DNS queries through IPv4.

Again, there are many options to setup a DNS proxy. This tutorial uses *`pdnsd`*.

```
# apt-get install pdnsp
```

The configuration of *`pdnsp`* is very simple. Comment all the *`server`* directives in the provided configuration file and add the following instead.

```
/etc/pdnsd.conf
```

```
server {
    file=/etc/resolv.conf;
}
```

Start the proxy DNS and point your clients' DNS server to the Linux router (192.168.1.1 in this example). Also, of course, the clients' interface must have an IPv4 address out of network 192.168.1.0/24.

```
# /etc/init.d/pdnsd start
```

Now your clients have full connectivity to the IPv6 internet. But before we test this we combine all the steps we just went through in a single script and start the script each time the WAN link comes up.

Scripted Configuration

When a PPP connection becomes active the *`/etc/ppp/ip-up`* script is executed, which in turn executes the scripts in the directory *`/etc/ppp/ip-up.d`*. That is where the script must be stored.

```
/etc/ppp/ip-up.d/init6to4
```

```
#!/bin/bash
# The name of the 6to4 tunnel link
TUN_INTF=tun6to4
# Initialize the SLAs (Site-Level Aggregator).
# SLA_INTF specifies the interface
# SLA_SUBN specifies the prefix to the corresponding SLA_INTF
# Use SLA_INTF=() if only the tunnel must be created
# To have IPv6 address on the tunnel:
# -> SLA_INTF=(${TUN_INTF} eth0) SLA_SUBN=(0 10)
SLA_INTF=(${TUN_INTF} eth0)
SLA_SUBN=(0 10)
# 6to4 relay address
TUN_RELY=192.88.99.1

# Clean up before adding the tunnel
ip -6 route flush dev ${TUN_INTF}
ip link set dev ${TUN_INTF} down
ip tunnel del ${TUN_INTF}

# Configure the 6to4 sit tunnel (Simple Internet Transition)
ip tunnel add ${TUN_INTF} mode sit remote any local ${PPP_LOCAL}
# IPv6 in IP has an additional IPv4 header (20 bytes). 8 bytes for PPPoE.
ip link set dev ${TUN_INTF} mtu 1472 up

# Add the routing.
ip -6 route add 2002::/16 dev ${TUN_INTF}
ip -6 route add ::/0 via ::${TUN_RELY} dev ${TUN_INTF} metric 1

# Configure IPv6 addresses on local interfaces (as defined in SLA_INTF)
I=0
for INTF in ${SLA_INTF[@]}; do
        # Before new 6to4 addresses are configured, delete old ones
        ip -6 addr flush to 2002::/16 dev ${INTF}
        # Add the 6to4 IPv6 address (SLA from SLA_SUBN)
        ip -6 addr add $(printf "2002:%02x%02x:%02x%02x:%04x::1/64" \
        $(echo "${PPP_LOCAL} ${SLA_SUBN[$I]}" | tr '.' ' ')) dev ${INTF}
        let I++
done

# Restart router advertisement daemon
```

```
if [ -x /etc/init.d/radvd ]; then
        /etc/init.d/radvd restart
fi
# Restart Proxy DNS
if [ -x /etc/init.d/pdnsd ]; then
        /etc/init.d/pdnsd restart
fi
```

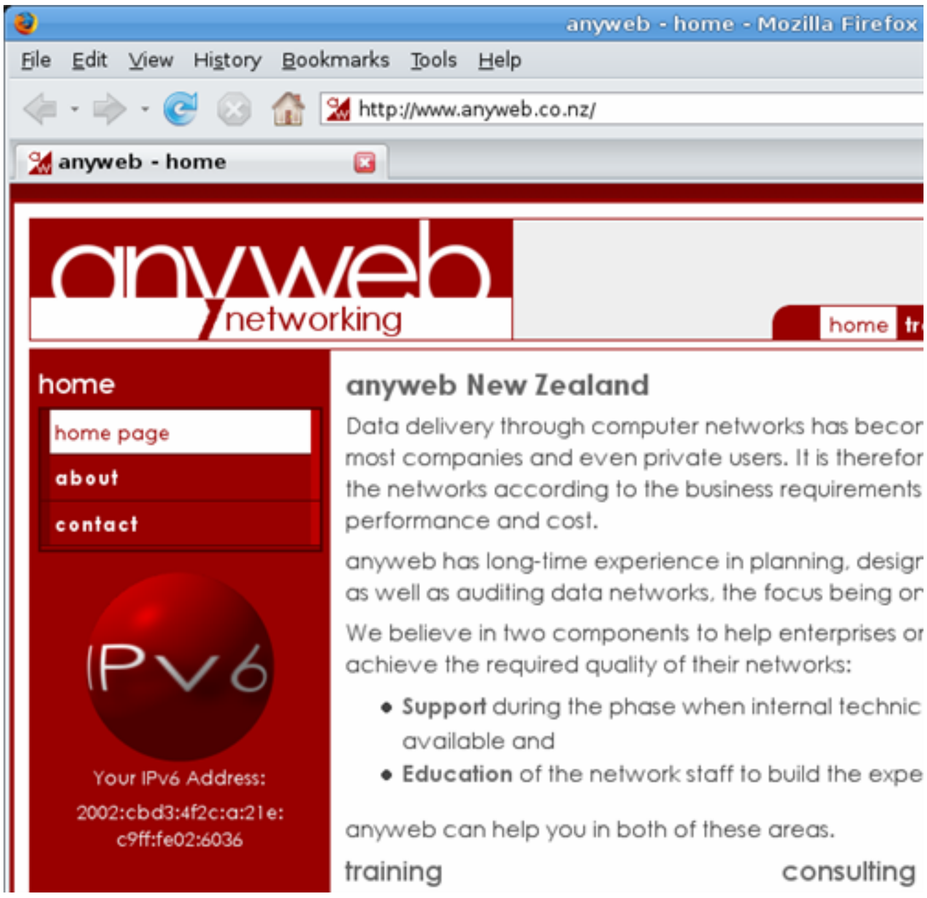
The script cleans up any configuration that may be on the system from previous connections. Therefore only a start script but no stop script is required which is an advantage when the WAN link goes down - the local LAN addresses are still available and connectivity assured.

The variable *PPP_LOCAL* is set by the parent script */etc/ppp/ip-up* and contains the IPv4 address assigned to the Linux router by the service provider.

Verify

From one of your clients attached to the local LAN you can now connect to IPv6 Internet sites.

anyweb's website is reachable through IPv6. If the web application detects a client IPv6 address it displays it in the sidebar together with a IPv6 icon.



Your IPv6 Address: 2002:cbd3:4f2c:a:21e:c9ff:fe02:6036

anyweb New Zealand

Data delivery through computer networks has becor
most companies and even private users. It is therefor
the networks according to the business requirements
performance and cost.

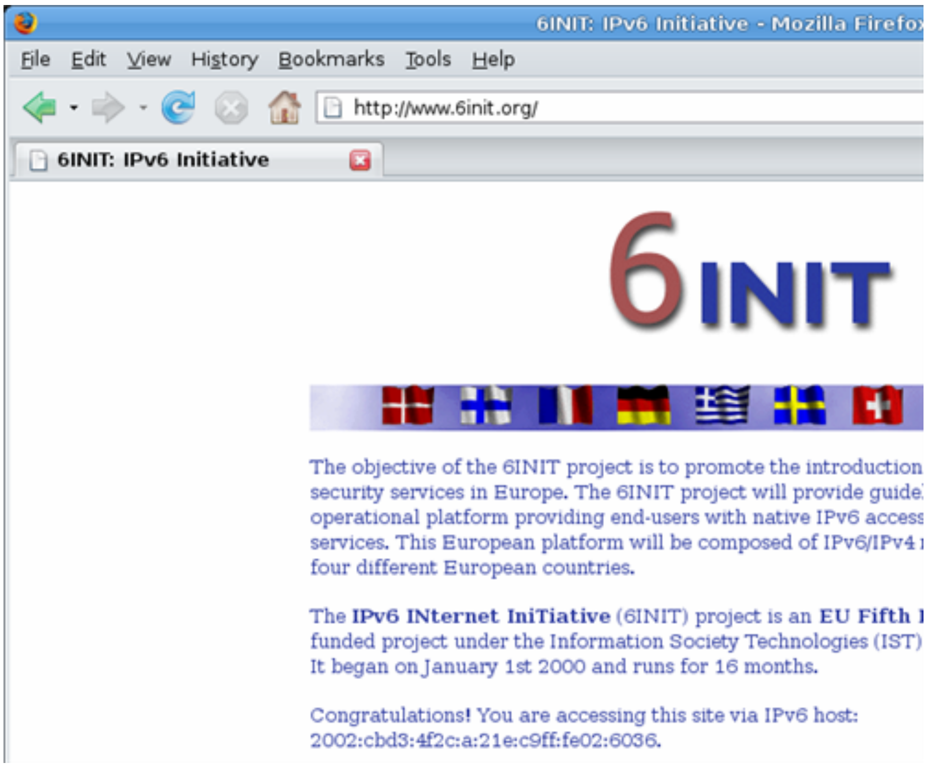
anyweb has long-time experience in planning, desigr
as well as auditing data networks, the focus being or

We believe in two components to help enterprises or
achieve the required quality of their networks:

- **Support** during the phase when internal technic available and
- **Education** of the network staff to build the expe

anyweb can help you in both of these areas.

Another example of an IPv6 enabled site is 6INIT. It also shows the client's IPv6 address. As a special effect the flags at the top of the page flutter with IPv6 while they are still when connecting using IPv4.


6INIT: IPv6 Initiative - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.6init.org/
6INIT: IPv6 Initiative
6INIT

The objective of the 6INIT project is to promote the introduction
security services in Europe. The 6INIT project will provide guide
operational platform providing end-users with native IPv6 access
services. This European platform will be composed of IPv6/IPv4
four different European countries.

The **IPv6 INternet IniTiative** (6INIT) project is an **EU Fifth**
funded project under the Information Society Technologies (IST)
It began on January 1st 2000 and runs for 16 months.

Congratulations! You are accessing this site via IPv6 host:
2002:cbd3:4f2c:a:21e:c9ff:fe02:6036.

The addresses used within the site (unique local unicast) are of course usable as well. For example, you can now connect from the client to the Linux router that was just configured as the 6to4 gateway.

```
user@client:~$ ssh fdcb:3fb1:918:1::1
user@fdcb:3fb1:918:1::1's password:
Linux 6to4gw 2.6.18-xen #1 SMP Fri Jun 1 15:01:20 BST 2007 i686

Last login: Tue Apr 29 17:03:22 2008 from 192.168.1.100
user@6to4gw:~$
```