LINUX PLIP MINI–HOWTO

Table of Contents

LINUX PLIP MINI−HOWTO

Andrea Controzzi, `controzz@cli.di.unipi.it`

v2.1, 12 March 1998

This HOWTO will hopefully let you build and use a Parallel Line Interface Protocol.

NOTE

This is a new release. There are many changes and lots of enhancements, but there will be still grammar and spelling errors (english is not my native language) and, unlikely but possible, some wrong or outdated info. PLEASE let me know about any errors to help me provide the correct information for everybody.

The biggest changes in the release are:

- SGML format
- a general answer to the "PLIP with win95" most asked question (the answer is negative, sorry)
- bigger and better FAQ section (thanks to the reports of countless users: many of them will find their comments there, as I promised!)
- removal of the Quick PLIP Installation files, that was useless and outdated.
- updated Dos−Linux PLIP link addendum

For any question, error correction, comment and/or suggestion, my E−Mail address is: controzz@cli.di.unipi.it. Feel free to mail me any time you need help. Sometimes I won't answer immediately, but I'll answer. If it happens that you don't receive anything from me after 2 weeks, this means that our mail system has trouble: do not hesitate to mail me again, it's my duty to help you. I can guarantee I'll solve your problems, but I'll try. If after many mails you don't receive answer, check your return address. Several times my answers did not arrive due to delivery problems, usually because the return address was root@myhost.

Before sending mail read the FAQ, my answer to a question already present in the FAQ will not be better (but likely worse or less complete) than the answer you can already read.

For questions about PLIP with DOS and Win95 please send mail to the authors of these chapters, I can't help you.

First of all, a lot of technical information come from the net−2−HOWTO, by Terry Dawson. This mini−HOWTO is not supposed to cover other aspects and/or replace the net−2−HOWTO: my goal is to give you a way to install a PLIP permanent connection quickly, *ONLY* this. All the other info come from my personal experience and the help of many users that sent me comments and information.

Read the net−2−HOWTO and the other docs for the general information about the network and the config files I suggest to change.

---

1. Introduction: what is PLIP and why should I use it?

There are many ways to create a connection between multiple hosts. PLIP, like SLIP, allow a local connection between two machines, but uses the parallel ports.

Parallel ports transfer more than one bit at a time, this means it is possible to achieve higher speeds than with a serial interface.

The speed achieved depends completely on your hardware (CPU and parallel port) and system load, in general it may be from 5 Kb/sec up to even 40 Kb/sec.

The PLIP interface is fast enough to allow some decent tcp/ip functions, like NFS. So, you may have a computer with all your Linux stuff and another with only the minimal system, where you can mount all the rest from the main machine.

The disadvantage is that most users have only one parallel port, this means that you won't be able to print and use PLIP together. Even with two parallel ports it is impossible to print and use PLIP without using kernel modules.

This disvantage can be also eliminated, if you have two or more parallel ports, applying a patch that you can find in this Mini−HOWTO.

Finally I am now able to give a hopefully good way to set up a PLIP link between DOS and Linux.

I won't stress it enough: so far nobody reported a successful link between Linux and Windows95.

---

2. Hardware required to use PLIP.

The hardware required to set up a PLIP interface is (obviously) a free parallel port in both the machines and the cable. If you can configure it with your BIOS, set it at least as "bi−directional", but if possible in ECP or EPP mode.

About the cable, this is what is written in the plip.c file, in the kernel 2.0.33 source:

```
The cable used is a de facto standard parallel null cable -- sold as
a "LapLink" cable by various places.  You'll need a 12-conductor cable to
make one yourself.  The wiring is:
  SLCTIN      17 - 17
  GROUND      25 - 25
  D0->ERROR   2 - 15          15 - 2
  D1->SLCT    3 - 13          13 - 3
  D2->PAPOUT  4 - 12          12 - 4
  D3->ACK     5 - 10          10 - 5
  D4->BUSY    6 - 11          11 - 6
Do not connect the other pins.  They are
  D5,D6,D7 are 7,8,9
  STROBE is 1, FEED is 14, INIT is 16
  extra grounds are 18,19,20,21,22,23,24
```

But I strongly advice you to read the /usr/src/linux/drivers/net/README1.PLIP and README2.PLIP files for more info about the cable.

In my opinion you should avoid building your own parallel null cable. A self−made cable may save very little money, but can add lots of headaches. If you wish to build your parallel cable, remember that you're doing it at your own risk, I reported exactly what is written in plip.c but I don't give warranties.

A final word about cable length: long cables (i.e. more than 10 feet or 3 meters) may bring problems due to radio interference. If you need long cables you should use good and well shielded cables, but very long cables are not recommended: I think the maximal cable lenght should be 15 meters (50 feet).

Anyway, someone mailed me that his/her 100 feet (30 meters) cable works fine; if someone really wants to try a PLIP connection between the office and his/her home (200 meters away), and has the money to spend, can try it, but is at his/her risk.

---

3. Reconfigure the kernel.

You're supposed to already know how to configure and compile the kernel, otherwise you must get some doc (kernel−howto or other guides). Thanks to the cool work made by the kernel guys, recompiling the last kernels is a really easy jobs also for "common" people, so just do it. Anyway, for the sake of completeness, here is a quick summary of what you must do:

NOTE: I suppose you are using the 2.0.xx kernel series. Now there is no need to keep the 1.2.xx kernels. There are no istructions about the 2.1.xx kernel series, since they are for development.

I will suppose that you use menuconfig to set up the kernel options, but the other tools are equivalent. I'll show how to do it with menuconfig:

```
#make menuconfig
```

I strongly advice to select

```
Loadable module support  --->
```

and enable the

```
[*] Enable loadable module support
```

and, if possible (i.e. you have modules.2.0.0) the

```
[*] Kernel daemon support (e.g. autoload of modules)
```

Then go back and choose

```
Networking options  --->
```

where you should choose at least

```
[*] Network firewalls
[*] TCP/IP networking
[*] IP: forwarding/gatewaying
```

The go back and choose at least

```
[*] Network device support
<M> PLIP (parallel port) support
```

If you use modules I definitely advice you to set up PLIP as a module. If you do so you can also, if you need to use a printer, go to

```
Character devices  --->
```

and set up as a module the

```
<M> Parallel printer support
```

Now you have enabled the kernel support for PLIP. If it's the first time that you compile the kernel look at all the other options then save and exit.

Finally compile with

```
#make dep ; make clean
#make zlilo
```

And, if you use modules

```
# make modules
# make modules_install
```

Now reboot your system.

4. Kernel messages about the PLIP interface.

After you've reconfigured and compiled the kernel with PLIP support enabled, when you boot the system, if the kernel supports PLIP directly, or when you load (later, see below) the PLIP module if you compiled PLIP as modules, you should get something like this (numbers may differ):

```
NET3 PLIP version 2.2 gniibe@mri.co.jp
plip1: Parallel port at 0x378, using assigned IRQ 7.
```

Depending upon your klogd and syslogd configuration the plip message could have been stored in your system log files: don't panic if you don't see the above message. If you compiled PLIP as a module and lsmod shows that the plip module is loaded, then it's enough.

Please take notice of the interface name. Usually is plip1, but may be plip0 or even plip2, plip3, and so on. It depends on the IO Address.

5. Setting up the configuration files.

NOTE: Some distributions, like Debian, use different config files. If you have a standard installation and you don't find the rc.inet* files, look for (different) config files in the /etc/init.d directory.

First of all remember to backup all the files you will change,

```
#cp rc.inet1 rc.inet1.BACKUP
```

may be a good idea.

Now, if you don't have it done already, you must choose the IP addresses of the two machines. In my examples I'll use a couple of example IPs for the IPs that you'll write, in the standard xxx.xxx.xxx.xxx format.

In the /etc/rc.d/inet1.rc file of both the machines add this (better if in the last part of the file):

```
/sbin/route add −net ${NETWORK} netmask ${NETMASK}
```

Where NETWORK and NETMASK should be set up previously. If you don't know how to do it, please read the NET−2−HOWTO.

If after this route command you get a message like this:

```
SIOCADDRT: network unreachable
```

then use this instead:

```
/sbin/route add −net ${NETWORK} netmask ${NETMASK} dev plip1
```

where, as usually, you'll have to use the interface name reported by the kernel messages (see above).

You may safely ignore these variables only in the following case:

If you only want to connect two machines on a standalone network, you may pick−up any IP address, say 200.0.0.1 and 200.0.0.2 respectively. In this case you can safely put NETWORK="200.0.0.0" and NETMASK="255.255.255 .0". These are the example IPs that I use in my Quick PLIP Installation (see below).

NOTE: 200.0.0.1 and 200.0.0.2 are only example IPs, I advice not to use these numbers definitively because they could be the addresses of real hosts on Internet!

I strongly advice to choose your address between the "private address" intervals:

```
10.0.0.0        -   10.255.255.255
172.16.0.0      -   172.31.255.255
192.168.0.0     -   192.168.255.255
```

In the file /etc/hosts of both the machines you should add the entries with the IP of the machines that you connect via PLIP. In my example, the entries are:

```
200.0.0.1       one             # this is the "one" IP address
200.0.0.2       two             # this is the "two" IP address
```

Where one and two are the names you have chosen for the two hosts.

If you want to activate the NFS, beside answering yes during the kernel configuration, you must add in /etc/exports the entries that describe the directories that you wish to export. In my example, to be able to mount the directory /usr, you should add this entry:

/usr	two (ro)

For more informations about NFS, please read the specific documentation; don't report me problems with the NFS, I won't be able to help.

Now reboot your system.

6. Activate the PLIP link.

Finally, these are the commands, that must be executed with root rights, that activate the PLIP interface (of course the cable must be already plugged correctly).

NOTE: If something unexpected happens, please doublecheck the cable and the spelling of the commands. If you followed the istructions correctly but the are still errors, read the FAQ paragraph, a lot of answers are already available.

First of all confirm that there is no lp device present:

```
# cat /proc/devices
```

You mustn't see any reference to lp like this:

```
 6 lp
```

If you see it, please remove (temporanely) the lp device before going on, if PLIP works then you can try it with lp later. To remove the lp device you'll have to use the rmmod if it's a module; if instead it's built in the kernel, you'll need to recompile the kernel with lp as a module (a much wiser idea).

Again I use the name one and two, as example. On one you'll have to do the following steps.

If you don't have the module automounter daemon and you compiled PLIP as a module, you must mount it:

```
# insmod plip
```

NOTE: if your parallel port is on an IRQ different from 7 and/or is on a IO Address different from 0x378, then you'll have to tell it to insmod. Find your real IRQ and IO Address (the DOS command MSD is likely to be ok, but don't trust it too much) and write something like this:

```
# insmod plip io=0x278 irq=5
```

Usually IRQ is 7 or 5, while IO Address is 0x378, 0x278 or 0x3bc. It is important that you check that the address and IRQ match the hardware settings (jumpers on old boards, BIOS on modern motherboards).

If you are paranoic check that the module has been loaded with:

```
  # lsmod

Module:         #pages:  Used by:
plip                3           0
```

Take notice of the interface name (plip0, plip1, and so on; for more details read the kernel messages chapter above), then set up the PLIP interface:

```
# ifconfig plip1 one pointopoint two up
```

NOTE: if your parallel port is on an IRQ different from 7 and/or is on a IO Address different from 0x378, then you'll have to tell it to ifconfig. Use the same IRQ and IO Address reported by the kernel messages and write something like this:

```
# ifconfig plip1 irq 7
# ifconfig plip1 io_addr 0x3bc
```

Usually IRQ is 7 or 5, while IO Address is 0x378, 0x278 or 0x3bc.

Now check that it worked...

```
# ifconfig
```

```
 .....
 .....
 plip1     Link encap:10Mbps Ethernet  HWaddr FC:FC:C8:00:00:01
           inet addr:200.0.0.1  P-t-P:200.0.0.2  Mask:255.255.255.0
           UP POINTOPOINT RUNNING NOARP  MTU:1500  Metric:1
           RX packets:0 errors:0 dropped:0 overruns:0
           TX packets:0 errors:0 dropped:0 overruns:0
           Interrupt:7 Base address:0x378
```

Add the route to two...

```
# route add two plip1
```

And, if you want also the NFS for two:

```
# rpc.portmap
# rpc.mountd
# rpc.nfsd
```

On "two" the commands are the same, but you must write one instead of two and vice versa.

One of your machines is likely to have only the PLIP connection, if this is true and that machine is two, you may also type:

```
# route add default gw one
```

on that machine. In my example above, two is a laptop with only a PLIP connection with one, so I type the above line on two.

Finally check with a

```
# ping two
```

from one and a

```
# ping one
```

from two to see that all is working.

Of course you may want to have all these commands automatically done by a script or at boot time. You must only create a script that execute these commands: now you may invoke it as root when you need, or you may add a command (in /etc/rc.d/rc.inet2) that calls it at boot time.

To tune your PLIP, you can use the plipconfig command, see the man page for more informations.

To shutdown PLIP, you need only to do:

```
# ifconfig plip1 down
```

which removes also the route entries. If you don't have the automounter daemon, then remove also the module:

```
# rmmod plip
```

---

7. FAQ.

This section will (hopefully) solve your problems. If you have any other question, feel free to mail me anytime.

- I get these messages at boot time (or when I load the plip module):
  ```
  SIOCSIFADDR: No such device
  SIOCADDRT: Network is unreachable
  ```

  and when I try to set up the link as written above, I get again error messages like:

  ```
  SIOCSIFADDR: No such device
  SIOCSIFDSTADDR: No such device
  SIOCADDRT: Network is unreachable
  mount clntudp_create: RPC: Port Mapper failure − RPC: Unable to send
  ```

  - ♦ The kernel, for some reason, hasn't PLIP support enabled. This could be due to:
    - ◊ You didn't answer yes to "PLIP support? " during kernel configuration.
    - ◊ You answered yes to "Printer support? " during kernel configuration.
    - ◊ You compiled PLIP as a module, so you must load it.
    - ◊ You are addressing the wrong port, i.e. you wrote for instance plip1 instead of plip0.
- Is there a way to support both PLIP and LP, beside modules, perhaps with two parallel ports?
  - ♦ Yes, so far there are two ways, described in the "Patches to make PLIP and LP live together":
    - ◊ You can apply a patch to make the kernel support both.
    - ◊ You can apply another patch to make the kernel use a parallel port for PLIP and another for LP.
- I have created the script that connects my 2 computers. I set up the link automatically in my rc.inet2, where I call a script that creates the link and enables NFS. My "two" hosts mounts some "one"'s directories; I have added the correct entries in "two"'s /etc/fstab. If I boot "two" when "one" is down, "two" halts for some minutes on the "mounting remote file systems...".
  - ♦ This happens because "two" waits to mount the "one" filesystems, but if "one" is down you must wait until "two" is bored of waiting. To avoid this, you may:

◊ Comment out in rc.inet2 the command that mounts the remote filesystems
◊ Remove the entry in "two"'s /etc/fstab and mount the remote filesystems manually when and if you need.
◊ A better solution would be for "two" to detect upon booting whether "one" is up, and mount the filesystem if it is. This can be accomplished by replacing the mount command in rc.d or whereever with something like the following:

```
if ping -c 5 one ; then
  mount one:/.....
fi
```

* My link is up, but ping fails. I receive the following message from the kernel:

```
plip1: timed out (1, 89)
```

or similar messages.

  * This means that the "your side of the link" is working, your machine sends the signal, but the "other side" isn't answering or your side is not waiting at the proper IRQ/IO Address. This is the most common problem and, alas, has a lot of possible reasons, usually bad cable or wrong IRQ and/or IO Address. The wrong IRQ is the source of over 60% of the problems, so it's very likely that changing it will remove the problem. Here is a detailed list of possible reasons:
    ♦ The cable isn't plugged properly or is broken or is wrong. Check it, if possible, between two Linux hosts which already work with PLIP. If it is not possible, then at least test the cable with a tester. The fact that the cable worked/not worked with DOS/win95 is a good/bad omen but is not a proof.
    ♦ The "other side" machine has not PLIP up.
    ♦ You are linked with a notebook with a not proper parallel port, see below.
    ♦ You have a really cheap parallel port that is a simple "printer" port, so can send and not receive.
    ♦ Your parallel port is not set as (at least) bi−directional. Do it in the BIOS configuration. Advanced parallel port settings like EPP or ECP are ok.
    ♦ The parallel ports have different irq, so you have to load the plip module (or the lp module) with a different irq. Go back to the chapter "Activate the PLIP link" and choose a different irq.
    ♦ Some other device may have shared your irq (which usually is irq 7), it may be a sound card. Do not trust DOS programs like MSD, instead try to load the plip module with a different irq.
* I put the right IRQ and IO Address, but it still doesn't work. I got the addresses from the MSD command.
  * I got a report from MSD giving wrong port addresses. Try to use this program: http://www.cs.caltech.edu/ huny/para13.zip.
* My link is up, and ping works. I sometimes receive the following message from the kernel:

```
plip1: timed out (1, 89)
```

or similar messages.

  * This means that the other side has not answered before the timeout. If all is working, you can ignore these messages: usually means that the other side is much slower than yours, either due to older hardware or more load. You can try to tune PLIP with the plipconfig command.
* I have installed the PLIP connection but if I ping I get 100% data loss. I connected my desktop with a notebook.

    - Some notebook's parallel ports aren't good for PLIP, because they are only "printer ports", i.e. they can only transmit but not receive the data. So far I don't know if there is a way to make them work. The only hope is:
        - Look at your notebook setup, perhaps there is a way to configure the parallel port as a parallel port instead of a printer port. Usually is called "parallel enhanced mode".
        - Try plip mode 0. Alas I don't know how to do it and/or if it works or is still available in the last kernels.
- What speeds can I achieve with PLIP?
    - This is an hard question to answer to, because there are MANY factors that can change deeply your performance:
        - The CPU speed on both the sides of the link.
        - The parallel port type and settings.
        - The system load.
        - What do you use PLIP for.

      Just to give a rough idea, you should achieve about 40Kbytes/sec, much faster than any serial rate and near to a low−level ethernet card.
- What happens if I need to ifconfig up and ifconfig down many times plip1?
    - Seems that you need to add a −arp to the ifconfig command, except for the first time after each boot. I don't need, but perhaps someone does.
- I have read the IP numbers reserved for private networks and your 200.0.0.1 and 200.0.0.2 are not in these ranges. Shouldn't they be changed?
    - Yes, they should. But as I underline since the beginning I choose these IP addresses only because of their simplicity, you are free to change them as you wish. Here is a cut from the net−2−howto:

```
RFC1597 has specifically reserved some IP addresses for private
networks.  You should use these as they prevent anything nasty
happening if you accidentally get connected to the Internet. The
addresses reserved are:

  10.0.0.0        -   10.255.255.255
  172.16.0.0      -   172.31.255.255
  192.168.0.0     -   192.168.255.255
```

- Is there a way to fine tune PLIP parameters without editing the source code?
    - Yes, there is. Try the /sbin/plipconfig command. See the man page for more info.
- I'm running Debian GNU/Linux, and under Debian, the files /etc/rc.d/rc.inet1 and 2 do not exist. Where must I write the plip configuration commands?
    - In Debian GNU/Linux you must edit /etc/init.d/network, where you have to put all the commands that should stay in rc.inet1 and 2.
- I have some problems linking two hosts with PLIP. The first has the latest kernel, the second still uses the 1.0.x PLIP version: is this a problem?
    - Yes, it's much better, where is possible, to have the same PLIP version on both ends. In the plip.c is written that the actual PLIP cannot work with the 1.0.xx PLIP.
- Right now PLIP works with 4 bits, what about the 8 bit PLIP I've read in the kernel docs? I think is called Mode 1.
    - This Mini−HowTo is for configuration, for technical informations please read the /usr/src/linux/drivers/net/README*.PLIP files or contact the author. What I know is only this: the standard PLIP uses "null printer" cables and is the Mode 0 (don't confuse it with plip0, which is the interface name), which uses 4 bits; Mode 1 uses 8 bits and should be available already, but will need an handmade cable and will work only between 2 Linux hosts. I don't know, once you got the cable, how to set up the Mode 1 PLIP link; if somebody does, please let me know.

---

8. Patches to make PLIP and LP live together.

The best way to make PLIP and LP live together is to use kernel modules: you can load plip.o and unload it when you need to print or vice versa. If you do really need to use both PLIP and LP, try the following patches.

8.1 PLIP and LP together on the same port.

If for some reason you wish PLIP and LP supported directly by the kernel, you can try these patches.

You must modify the following pieces of code, but *backup* the files before:

```
******** modifications to linux/drivers/char/lp.c ***********************
struct lp_struct lp_table[] = {
        { 0x3bc, 0, 0, LP_INIT_CHAR, LP_INIT_TIME, LP_INIT_WAIT, NULL,
NULL, },
/*      { 0x378, 0, 0, LP_INIT_CHAR, LP_INIT_TIME, LP_INIT_WAIT, NULL,
NULL, },
        { 0x278, 0, 0, LP_INIT_CHAR, LP_INIT_TIME, LP_INIT_WAIT, NULL,
NULL, },
*/
};
#define LP_NO 1

******** modifications to linux/drivers/net/Space.c ********************
#if defined(PLIP) || defined(CONFIG_PLIP)
    extern int plip_init(struct device *);
    static struct device plip2_dev = {
        "plip2", 0, 0, 0, 0, 0x278, 2, 0, 0, 0, NEXT_DEV, plip_init, };
    static struct device plip1_dev = {
        "plip1", 0, 0, 0, 0, 0x378, 7, 0, 0, 0, &plip2_dev, plip_init, };
/*    static struct device plip0_dev = {
        "plip0", 0, 0, 0, 0, 0x3BC, 5, 0, 0, 0, &plip1_dev, plip_init, };
*/
#   undef NEXT_DEV
#   define NEXT_DEV     (&plip1_dev)
#endif  /* PLIP */
```

Of course there is the standard disclaimer: *I received these patches and I put them "as I got them". This means that you try them at your own risk.* Anyway, your biggest trouble should be only restore the original files and recompile.

8.2 PLIP and LP on different ports.

If you have at least 2 parallel ports you can try these patches, that should allow you to use PLIP on a port and LP on the other.

1. Comment out one line in kernel source file, drivers/char/lp.c.

```
struct lp_struct lp_table[] = {
{ 0x3bc, 0, 0, LP_INIT_CHAR, LP_INIT_TIME, LP_INIT_WAIT, NULL, NULL, },
{ 0x378, 0, 0, LP_INIT_CHAR, LP_INIT_TIME, LP_INIT_WAIT, NULL, NULL, },
```

```
/* { 0x278, 0, 0, LP_INIT_CHAR, LP_INIT_TIME, LP_INIT_WAIT, NULL, NULL, }, */
};
    3 -> 2
```

2. Kernel configuration
```
        PLIP (parallel port) support (CONFIG_PLIP) [n] y

        Parallel Printer support [y] y
```
3. Kernel message at startup
```
        lp1 at 0x0378, using polling driver
        .....
        NET3 PLIP version 2.0 gniibe@mri.co.jp
        plip2: Parallel port at 0x278, using assigned IRQ 5.
```

Again the standard disclaimer, like section 8.1.

9. A PLIP link between DOS and Linux

After the first release of this Mini−HowTo many people wrote for info about a link between Linux and DOS (or Windows) computer. The general interest lead me to add this chapter, I hope will be of help to everybody.

This section comes from an article I've found on Linux Gazzette by James McDuffie <mcduffie@scsn.net> . It covers the basic installation of a PLIP link between Linux and a DOS computer using Windows and Trumpet WinSock and gives the address of a cool program that let's you run X−Windows programs on Windows.

The last section is an addendum sent by James Vahn jvahn@short.circuit.com where he describes deeply how to set up this link and how to solve many problems.

For any questions about this chapter please contact him, not me.

9.1 DOS−Linux link.

I suppose you have already set up properly the PLIP support on the Linux side and you have got the right cable, else go back to the previous chapters.

Now, for the DOS side, you need first of all a packet driver. It can be found here:

ftp://ftp.crynwr.com/drivers/plip.zip

The program runs under DOS and acts like a Ethernet Packet driver. If you want to use PLIP with Windows you need also Trumpet Winsock. This serves as the TCP/IP interface. Otherwise, you can probably find TCP/IP software for DOS.

Now go back to the Linux computer and add the DOS computer address to /etc/hosts. If your DOS computer does not have a registered IP address you may choose any address (remember the warning of chapter 3 about IP addresses).

Now let's suppose you chose the name linux for the Linux computer and dos for the DOS one. You have to type:

```
ifconfig plip1 linux pointopoint dos arp up
route add dos
```

Of course if you want to have this done every time you boot the linux computer you may add these lines to the file /etc/rc.d/rc.inet1:

```
/sbin/ifconfig plip1 linux pointopoint dos arp up
/sbin/route add dos
```

This sets up the interface and then adds a route to it. Of course if you are using the second parallel port you have to write plip2 instead.

Go back to the DOS/Windows computer and edit autoexec.bat, you have to add the following lines.

```
c:\plip\plip.com 0x60
c:\tcpip\winsock\winpkt.com 0x60
```

Of course I suppose you put plip.com (the packet driver) in the directory c:/plip and the winpkt.com in c:/tcpip, else you need to put the right path.

This sets the plip.com program on packet vector 0x60 and then loads the winpkt.com program that comes with trumpet winsock on the same vector. If the cable is something other than lpt1 you will have to tell plip.com the irq number and io address. Also, winpkt.com needs to run to make the packet vector avaliable to Windows. From here we go to the actual setup under Trumpet Winsock. All you have to do is unselect SLIP or PPP and enter 60 into the box labeled Packet vector. Then tell it the IP address you gave it, the IP address of the Linux computer as the default gateway and the Name Server as either you computer's ip or your ISP's address for its nameservers if your going to connect it to the Internet (more on this later). Close the setup and re−run Winsock and you should have it! Put winsock in your startup group and you have everything setup automatically!

If you want to access the Internet through the Linux computer on the Windows computer you will need to set up IP Masquerading, for info on this see the NET−2−HOWTO. This simply masquerades the Windows computer with your Linux computer's IP address.

Also I have found a program that lets you run X−Windows programs under Windows! It is located at:

http://www.tucows.com/

Set it up according to directions and then all you have to do is telnet in from the Windows computer then set the display to the Windows computer (`DISPLAY=duncan:0.0` for instance) and run the program desired. There is nothing cooler than running xv under Windows! Hope all this helped.

9.2 A DOS−Linux PLIP link experience.

NOTE: I received this document from James Vahn jvahn@short.circuit.com. I put it here unchanged. This means that **for any question about this section he's much better qualified than me so please mail to him than to me**. His experience with a PLIP connection of a floppy−only DOS computer to a Linux one is the perfect example of how to work−around common problems.

Last Update 11 July 1996

My floppy−only DOS box is networked via PLIP to the second printer port on the Linux machine. The first Linux printer port has a printer on it, both are permanently connected and the DOS box is telnet'd into Linux. These are my notes on what I did to accomplish this.

When the kernel probes for printer ports, it will grab all of them unless you remove one from the probe. Otherwise PLIP will get nothing. One method is to load the drivers as modules when needed...

<gniibe@mri.co.jp> writes:

I keep recommending using PLIP/LP as kernel module, since

- modules are flexible for change of configuration
- (re)compiling the kernel is not easy for novice users
- co−existing PLIP and LP is only feasible by the modules

With PLIP/LP as kernel module, you can specify which port is PLIP and which port is LP. Here is example:

```
# insmod lp.o io=0x378
# insmod plip.o io=0x278 irq=2
```

Even you can use two parallel ports:

```
# insmod plip.o io=0x278,0x3bc irq=2,5
```

In the example above,

plip0 is assigned on 0x278 and it's irq is 2,

plip1 is assigned on 0x3bc and it's irq is 5, respectively.

Using modules certainly sounds like the way to go. The following method shows how to patch the kernel to allow both a printer and PLIP on different ports, without modules. If you are unfamiliar with the module concept, you might find this quicker to set up.

You will need to modify two files in the kernel source tree. I'm using kernel 1.2.13 and found some changes were needed in ../linux/drivers/net/Space.c to accommodate my system. Look at around line 205 for the PLIP definitions to make sure your port and IRQ match, and make a note of which driver it will be (plip0, plip1, plip2). In my case port 0x278 uses IRQ 5 (the card is jumpered that way) but Space.c defined it with IRQ 2. I made the changes here, rather than opening up the box and changing jumpers. The alternative is to specify the IRQ through ifconfig later on, but the kernel will boot up with the wrong IRQ for PLIP and it may annoy you. It is a simple (single character) change.

The next, and more difficult step:

In .../drivers/char/lp.c you will find the following at around line 38:

```
struct lp_struct lp_table[] = {
    { 0x3bc, 0, 0, LP_INIT_CHAR, LP_INIT_TIME, LP_INIT_WAIT, NULL, NULL, },
    { 0x378, 0, 0, LP_INIT_CHAR, LP_INIT_TIME, LP_INIT_WAIT, NULL, NULL, },
/*  { 0x278, 0, 0, LP_INIT_CHAR, LP_INIT_TIME, LP_INIT_WAIT, NULL, NULL, },
 * 0x278 reserved for plip1
 *
 * };
 * #define LP_NO 3
 */
};
#define LP_NO 2
```

Notice the changes to make− one port is commented out, so now only 2 ports are defined. Port 0x3BC will probably not work for PLIP− the IRQ line is usually broken on these ports, as found on old monochrome adapters (MDA).

You made backups of these file before you changed them, right? Now make a new kernel with printer, net, dummy, and plip support.

Configure the system. This is my /etc/rc.d/rc.inet1 file:

```
#!/bin/bash
#
/sbin/ifconfig lo 127.0.0.1
/sbin/route add −net 127.0.0.0

/sbin/ifconfig dummy 200.0.0.1
/sbin/route add −net 200.0.0.0 netmask 255.255.255.0
/sbin/ifconfig plip1 arp 200.0.0.1 pointopoint 200.0.0.2 up
/sbin/route add 200.0.0.2
/sbin/ifconfig dummy down
```

Notice that arp is used for the DOS−to−Linux connection, apparently not used on Linux−to−Linux connections.

And in /etc/hosts you can add these, just to give the two machines names:

```
200.0.0.1       console1
200.0.0.2       console2
```

The DOS box is console2. Note Andrea's warning about these, better to use official numbering schemes.

Reboot so all of these changes and the new kernel will take effect. During the boot sequence (or by running dmesg) if you made the patches, otherwise when the modules are loaded:

```
lp0 at 0x03bc, using polling driver
lp1 at 0x0378, using polling driver
[....]
NET3 PLIP version 2.0 gniibe@mri.co.jp
plip1: Parallel port at 0x278, using assigned IRQ 5.
```

The "route" command shows this:

```
Kernel routing table
Destination     Gateway         Genmask         Flags MSS    Window Use Iface
console2        *               255.255.255.255 UH    1436   0      136 plip1
loopback        *               255.0.0.0       U     1936   0      109 lo
```

And "ifconfig plip1" shows:

```
plip1     Link encap:10Mbps Ethernet  HWaddr FC:FC:C8:00:00:01
          inet addr:200.0.0.1  P-t-P:200.0.0.2  Mask:255.255.255.0
          UP POINTOPOINT RUNNING  MTU:1500  Metric:1
          RX packets:132 errors:0 dropped:0 overruns:0
          TX packets:136 errors:0 dropped:0 overruns:0
          Interrupt:5 Base address:0x278
```

Look at /etc/inetd.conf and see if telnet is enabled. You might want to read the man page for tcpd, and the use of /etc/hosts.allow (ALL: LOCAL) and /etc/hosts.deny (ALL: ALL). You should be able to "telnet localhost".

Linux is done, now the DOS side. Again, be suspicious of port 0x3BC if one is present.

I'm using NCSA's telnet and Crynwr's PLIP driver found at these sites:

ftp://ftp.ncsa.uiuc.edu/Telnet/DOS/ncsa/tel2308b.zip

ftp://ftp.crynwr.com/drivers/plip.zip

Be sure to use NCSA's version 2.3.08 telnet and version 11.1 of Crynwr's PLIP driver. Please find and read Crynwr's SUPPORT.DOC located elswhere.

The CONFIG.TEL file. Most of it is the default and to save some space I've tried to cut it back here to just the info you need (hopefully). The second port on this machine is setup as 0x278 on IRQ 5.

```
myip=200.0.0.2
netmask=255.255.255.0       # subnetting mask
hardware=packet             # network adapter board (packet driver interface)
interrupt=5                 # IRQ which adapter is set to
ioaddr=60                   # software interrupt vector driver is using
#
#[...lots unchanged...]
#
# at the end of the file, put this line:
name=console1 ; hostip=200.0.0.1 ; nameserver=1 ; gateway=1
```

(console1 is the name of the Linux machine, you can use whatever you like)

I made a 12 foot null cable between both machines, and (after initially finding it miswired) there have been no problems. A standard 11−wire null printer cable should work too. The Linux plip.c source shows the wiring. Although my cable has the 17−17 connection, I don't think it is used for anything and was not present on a ready−made cable.

```
@echo off
plip.com 0x60 5 0x278
telbin -s console1
```

That should connect you to the Linux box on /dev/ttyp. NCSA's telnet provides for 8 virtual screens and also acts as an ftp server. The PLIP interface provides a fair throughput, I'm getting 6.5K/s file transfers with my antiques. Let's hope you can do better. :−)

---

10. PLIP between Linux and Windows 95.

This section is empty. I use windows 95 for nothing serious but games, so I don't try and don't care about a PLIP link with Linux. The questions about such a link have won the most asked question contest, so I give here a (so far) definitive answer.

No, so far nobody reported me a successful link between Linux and Windows 95. if somebody succeds in setting up this link, please let me know immediately: thousand of PLIP users await these news!

---

11. Questions? Comments? Send me feedback.

For any questions and comments you can find me via e−mail at the address controzz@cli.di.unipi.it

Feedback is welcome, any error report is precious. The next release will have an even larger FAQ section, if you send questions and, of course, the answers if you find them by yourself.

Please do not send questions already present in the FAQ.

If you have to ask me for help, please be sure to let me know any information that can help me, at least: kernel version, commands used, error messages, the cable you used and any other system message related to PLIP.

Please remember not to send me any question about PLIP with DOS/Windows 3.1/Windows 95, I can't help you. These questions should be sent to James Vahn jvahn@short.circuit.com, who sent me the DOS addendum. Again: it's useless to ask him or me about PLIP with Windows 95.

---

12. Where to find new releases of this mini−howto.

This mini−HOWTO is maintained by the HOWTO coordinators and is posted monthly on comp.os.linux.answers and can be found in the HOWTO directory at sunsite and at sunsite's mirrors.

Another way to find the mini−HOWTO (and to contact me) is on my Home Page,

http://www.cli.di.unipi.it/~controzz/intro.html (italian language)

http://www.cli.di.unipi.it/~controzz/intro_e.html (english language)

13. Credits.

Many thanks to:

- Rick Lim <ricklim@freenet.vancouver.bc.ca> for the patches to make PLIP and LP live together.
- Takeshi Okazaki <GBA03552@niftyserve.or.jp> for the patches to use PLIP and LP on two different parallel ports.
- Jim Van Zandt <jrv@vanzandt.mv.com> for some advice on the "tutorial" part of this HOWTO.
- Fernando Molina <fmolina@nexo.es> for useful comments about IRQs and IO Addresses.
- James Vahn <jvahn@short.circuit.com> for the cool addendum on the PLIP between DOS and Linux chapter.
- To all the users that posted PLIP−related articles on the linux newsgroups and/or mailed me. The list of all the people that helped me with info and comments could be longer than the Mini−HOWTO itself: thank you all!

14. Copyright message.