LPC COMPUTER CONSULTING INC.

http://lpccomp.bc.ca/remserial/

Remserial

The remserial program acts as a communications bridge between a TCP/IP network port and a Linux device such as a serial port. Any character-oriented Linux /dev device will work.

The program can also use pseudo-ttys as the device. A pseudo-tty is like a serial port in that it has a /dev entry that can be opened by a program that expects a serial port device, except that instead of belonging to a physical serial device, the data can be intercepted by another program. The remserial program uses this to connect a network port to the "master" (programming) side of the pseudo-tty allowing the device driver (slave) side to be used by some program expecting a serial port. See example 3 below for details.

The program can operate as a server accepting network connections from other machines, or as a client, connecting to remote machine that is running the remserial program or some other program that accepts a raw network connection. The network connection passes data as-is, there is no control protocol over the network socket.

Multiple copies of the program can run on the same computer at the same time assuming each is using a different network port and device.

Some examples:

Give access to a RS232 device over a network.

The computer with the serial port connected to the device (such as a data aquisition device) runs the remserial program:

```
remserial -d -p 23000 -s "9600 raw" /dev/ttyS0 &
```

This starts the program in daemon mode so that it runs in the background, it waits for connections on port 23000 and sets up the serial port /dev/ttyS0 at 9600 baud. Network connections to port 23000 from any machine can then read and write to the device attached to the serial port.

This can be started from /etc/rc.local or as an entry in /etc/inittab or set up as a system service with a file in /etc/rc.init/.

Connect an RS232 device to a specified server.

The computer with the serial port connected to the device (such as a data aquisition device) runs the remserial program:

```
remserial -d -r server-name -p 23000 -s "9600 raw" /dev/ttyS0 &
```

This would be used with case number 1 above creating an end-to-end serial port connection. What goes in the serial port on one machine would come out the serial port of the other machine. The ports could be running at different baud rates or other serial port settings.

Connect a Linux program that needs a serial port to a remote serial port.

Some programs are written to communicate directly with a serial port such as some data aquisition programs. The remserial program can use pseudo-ttys to fool the program into thinking that it is talking to a real serial port on the local machine:

```
remserial -d -r server-name -p 23000 -l /dev/remserial1 /dev/ptmx &
```

This creates a file called /dev/remserial1 which can be used by the data aquisition application as its serial port. Any data sent or received is passed to the remote server-name on port 23000 where a computer configured in case number 1 above passes it to a real serial port.

The remserial program uses the special pseudo-tty master device /dev/ptmx (see man ptmx) which creates a slave device that looks like a normal serial port named /dev/pts/something. Unfortunately, the actual device name created isn't consistent, so the remserial program creates a symbol link from the device name specified with the -l option to the /dev/pts/ name that was created allowing the other application to be configured with a consistent device name.

Server farm console control.

Assuming multiple Linux servers (such as web servers) are set up to have a serial port as their console instead of a monitor/keyboard, their serial ports could be connected to a control server using a multi-port serial board. On the control server, a copy of remserial is run for each server: remserial -d -p 23000 -s "115200 raw" /dev/ttyS0 & remserial -d -p 23001 -s "115200 raw" /dev/ttyS1 & remserial -d -p 23002 -s "115200 raw" /dev/ttyS2 & remserial -d -p 23003 -s "115200 raw" /dev/ttyS3 & etc.

From any computer on the local network, use a telnet program to connect to the control server on the appropriate port:

```
telnet control-server-name 23002
```

This would connect through the associated serial port to the desired server's console. This example would then give the user console access to the 3rd server.

Careful scripting such as using the Linux "expect" program could allow batches of commands to be run on each server.

Other Linux program useful with remserial

nc

The netcat program is similar to remserial except that it creates connections between network ports and command line standard input and output.

For example, with case number 1 above, the following command run on another computer will send the contents of the named file out the serial port used by the remserial program:

```
nc server-name 23000
```

```
Similarily, the following command will store incoming serial data in a file
until the program is manually interrupted:
```

```
nc server-name 23000 >file-name
```

telnet

The telnet program is normally used to log into a remote computer,
but when used with network ports other than number 23, it operates in a
raw data mode.

For example, with case number 1 above, the following command will allow
the user of the telnet program to see incoming serial port data and
type data on the keyboard to send to the serial port:

```
telnet server-name 23000
```

This is ideal for controlling the device connected to the serial port
if it has some sort of command line interface usable over the serial port.

remserial Usage

```
remserial [-r machinename] [-p netport] [-s "stty params"] device

-r machinename          The remote machine name to connect to.  If not
                        specified, then this is the server side.
-p netport              Specifiy IP port# (default 23000)
-s "stty params"        If serial port, specify stty parameters, see man stty
-d                      Run as daemon programs
-x debuglevel           Set debug level, 0 is default, 1,2 give more info
-l linkname             If the device is /dev/ptmx, creates a symbolic link
                        to the corresponding slave pseudo-tty so that another
                        application has a static device name to use.
-w                      Wrote-only mode, no reading from the device.
device                  Character oriented device node such as /dev/ttyS0.
```

Download the source code. To compile,
just do make remserial.

Download a pre-compiled binary for i386 Linux with
glibc.

Please send feedback to
pdavis@lpccomp.bc.ca

Change Log

Version 1.4

- Added -m option to allow multiple client connections. This allows more than one client to share the port on the server at the same time. For example, if a weather instrument sends regular data, more than one program can listen at the same time. Another use would be if a serial console on a server is attached to computer sharing the serial port with remserial, muliple telnet clients could attach to the server at the same time and issue commands and monitor output.

Version 1.3

- Fixed compile on OSX and FreeBSD with the termio CBAUD, XCASE, IUCLC and ILCUC flags.
- Fix compile with conflicting devname variable name.
- Add -w write-only option.