Title	Manual - vtty
No.	

Rev.	Date	By	History
0.0.1	2003-12-4	UIS	Initial draft

This datasheet describes how to install the Unix software and run ttyd daemon. The software (vtty-1.0.tar.gz) is available at:

http://www.kanda.com/support/

1 Introduction

vtty consists of **sena_vpty**, **ttyd** and **vtty_manager**.
sena_vpty is a device driver module that is dynamically loaded into the Linux kernel. **ttyd** is a terminal server application. **vtty_manager** is a wrap-up application that provides an in-hand way of using the driver module, pseudo-tty devices and terminal server application.

2 Compile and installation

Get and copy `vtty-1.0.tar.gz` to somewhere in your Linux machine.

```
#tar -xvzf vtty-1.0.tar.gz
#cd vtty
#make
#make install
```

Note: All the binaries are installed in "`/usr/local/bin`" and the compiled kernel module is copied to "`/usr/local/vpty`" at the install time. An initialization script "vtty init" is appended to the end of "`/etc/rc.local`" as well.

3 Kernel driver module

After installation, with first execution of "`vtty init`" or "`vtty add {arguments..}`", the driver module for pseudo-tty devices is loaded into the kernel with its name "`sena_vpty`". The major numbers of devices are 226 and 227 respectively for master and slave node each. Users can add up to 255 devices and the device files are managed by "**vtty_manager**".

4 Device files

Once users add a node using "`vtty add [SUFFIX]`" command, there generated a pair of master and slave devices: `/dev/vpty[SUFFIX]` for master and `/dev/tty[SUFFIX]` for slave. The "**vtty_manager**" finds the lowest available minor number ranging from 0 to 255 and assign it for the devices.

5 Database

"**vtty_manager**" utilizes a local database for its data management including nodes(device files), serial options for terminal server side and node status. The database is generated as "`/var/vtty/vtty.db`".

6 Syntax

```
vtty add TTYSUFFIX ip_address tcp_port [serial_option]
```

```
vtty { enable | disable | remove } TTYSUFFIX
vtty status [TTYSUFFIX]
```

Note : The TTYSUFFX is a suffix for the device file name and at the same time, it becomes a management keyword for the vtty database

```
serial_option: -b <baudrate> -p <port_setting>

     <baudrate>: one of the following

     75, 150, 200, 300, 600, 1200, 1800, 2400, 4800,
     9600, 19200, 38400, 57600, 115200, 230400

     <port_setting>: specified by a string of one of more of the
     following concatenated together with no intervening spaces

     8: 8 bits/character
     7: 7 bits/character
     6: 6 bits/character
     5: 5 bits/character
     N: No parity
     E: Even parity
     O: Odd parity
     C0: No hardware flow control
     C1: Hardware flow control
     S0: No software flow control
     S1: Software flow control

If the serial option is not specified, the serial option defaults to :
baudrate = 9600, port setting = 8NC0

Example:
     vtty add mysuffix 192.168.1.1 6001 -b 9600 -p 8NC0
     vtty enable mysuffix
     vtty status
```

Add:

```
vtty add mysuffix 192.168.1.1 6001 -b 9600 -p 8NC0
```

Adds a node information on the local database and makes two device files: master(/dev/vptymysuffix) and slave(/dev/ttymysuffix). And then, invokes a terminal server process "ttyd" and the through-connection to the remote host is ready.

Users might want to use slave device for their application.

The status of the newly added device is "enabled".

If there happens any problem listed in the following in running the vtty, the command is discarded.

- Device add limit exceeded (No more minor number for the new device is available)
- The suffix already exists in the local database
- Failed in loading driver module
- A device file with the same name already exists (Either of master or slave device file)
- Any failure in making device file
- "ttyd" process invocation failure

Note: If there, in the kernel, is not loaded driver module, "add" or "init" command checks the module and loads it into the kernel.

Remove:

```
vtty remove mysuffix
```

Removes the node information from the local database and deletes two device files: master(/dev/vptymysuffix) and slave(/dev/ttymysuffix). And then, kills the relevant "ttyd" process running.

Enable:

```
vtty enable mysuffix
```

If the state of the node is "disabled", then the command sets the node status "enabled" and invokes a "ttyd" process to be ready to connect. Otherwise, this command has no effects.

Disable:

```
vtty disable mysuffix
```

If the state of the node is "enabled", then the command sets the node status "disabled" and kills relevant "ttyd" process. Otherwise, the command has no effects.
Note: The "disable" command does not delete device files.

Status:

```
vtty status [TTYSUFFIX]
```

Displays the status of the node(s). If TTYSUFFIX is given, it displays the status of the specified node. Else, it shows status of all nodes registered in the local database.
Note: STATUS field in the result is categorized into CONNECTED, DISCONNECTED and DISABLED.

Init:

```
vtty init
```

Initializes the vtty by loading driver module and running "ttyd" processes marked "enabled" in the local database. This command does not have effect twice or more.
Note: This command is appended to the end of "/ect/rc.local" file so to run at the boot-up time of the machine.
Note: Users don't need to use this command.

Remote TTY Daemon for Unix/Linux

Application Guide

Version 1.0

2003-10-24

Technical Support
Sena Technologies, Inc.
Tel: (+82-2) 573-7772
Fax: (+82-2) 573-7710
E-Mail: support@sena.com

Contents

.

1. Overview

This user guide documents the **ttyd** remote modem daemon. The **ttyd** daemon is fully telnet compatible allowing programs to connect to any remote device such as networked modems and terminal servers as if they were local devices as long as the device utilizes the telnet protocol. The **ttyd** daemon also supports raw TCP protocol allowing programs to connect to any remote device through the raw TCP protocol.

2. Operation of Daemon

After opening a Master `Pseudo tty` device, the daemon will wait for some other process to open the corresponding slave device. Once the slave device is opened, a connection will be made, using the telnet or raw TCP protocol, to a remote server. The remote end may be any server supporting the telnet or raw TCP protocol such as a terminal server or network modem.

3. Unix O/S support

Currently the supported system types are AIX, FreeBSD, Linux, OSF, SCO5 and Sun/OS.

SCO	- SCO 5 host
OSF	- DEC host
AIX	- AIX host
LINUX	- LINUX host
FreeBSD	- Free BSD host
SunOS	- Sun host

4. Synosis

`ttyd –d pty-device [option] host port`

Where options

-b baud-rate :

Sets the initial baud-rate of the device.

-d pty-device :

Master Pseudo tty device to open and wait for a connection on. This option is not optional, it must be provided

-p port-settings :

Configures the port as specified by port settings.

If port-settings is present, the port will be configured to them.

Otherwise, the current settings will be used. The settings are specified by a string of one or more of the following concatenated together with no intervening spaces:

8 8 Bits / Character

7 7 Bits / Character

6 6 Bits / Character

5 5 Bits / Character

N No Parity

E Even Parity

O Odd Parity

C0 No H/W Flow Control

C1 H/W Flow Control

S0 No S/W Flow Control

S1 S/W Flow Control

-n :

No detach, do not run as a background process.

The **host** and **port** parameters specify the IP address of host and port number to connect to when the Slave Pseudo device is opened.

For reference, in Linux the Master Pseudo `tty` devices are the devices **`/dev/pty[a-ep-z][0-9a-f]`** and the slave devices are **`/dev/tty[a-ep-z][0-9a-f]`**.
Other Unices may have different naming schemes.

5. Examples

Let's assume IP address of the host and port number to connect as **192.168.100.3** and **7001** respectively. And also assume user wants to use 0-th Pseudo tty device as a local serial device on the Linux host.

If user runs the following command in the shell,

```
# ttyd -d /dev/ptyp0 -b 9600 -p 8NC0 192.168.100.3 7001
```

Then the daemon will wait for some other process to open the corresponding slave device(`/dev/ttyp0`). Once the slave device is opened, a connection will be made, using the telnet or raw TCP protocol, to a remote server. Users may make some script for batch process to connect multiple connection.