Mellow Panda V1.0.1

16 January, 2010

Guy Cole
gsc@digiburo.com

Abstract – Many contemporary communications or "scanner" radios feature remote control options, typically in the form of a serial (RS-232 or USB) interface. In general, the communication protocol for these radios is a simple ASCII based request/response cycle. Even a simple dumb terminal such as a VT-100 can control these radios. Mellow Panda is a collection of applications which permits control of serial devices via a TCP/IP connection. The Mellow Panda distribution contains a background daemon (pandad) which acts as a TCP/IP to RS-232/USB gateway and a command line client (panda).

Introduction

Mellow Panda is a collection of applications which enable remote control of communications and "scanner" radios via RS-232/USB interfaces.

I created Mellow Panda because most radio manufacturers only support "Brand W" computer operating systems, and I use LINUX (Ubuntu LINUX to be specific). There are already many similar LINUX based applications available, but they did not fit my needs (although they were helpful to review). Mellow Panda was developed on Ubuntu 8.10 desktop and (to date) has been tested w/Ubuntu 8.04.3 desktop, Ubuntu 9.04 server and Fedora 11. I have exercised pandad w/Icom PCR-1000, R-7000, Uniden Bearcat BC-780-XLT and Radio Shack PRO-2052 radios.

Notable features:

- pandad is a UNIX daemon and does not require a controlling terminal. I have several applications which drive my radios automatically, and in these cases a GUI is a liability.
- pandad is accessed via TCP/IP, allowing your radios to be shared by multiple machines.
- Support for asynchronous updates (PCR-1000 family "band scan", etc).
- pandad can simultaneously support radios of different model and manufacture. I routinely drive Uniden Bearcat scanners right along w/my Icom radios using a single instance of pandad.
- Diagnostic options to assist you w/installing new radios.
- Extensive logging support via syslogd(8).
- Did I mention it runs on LINUX?
- There are several demonstration clients:
  - a command line client (in C)
  - //more coming//

The sources and documentation for Mellow Panda are available on Source Forge (http://mellowpanda.sourceforge.net) and released under the BSD license (copyright 2010 by Guy Cole). There is no obligation to contact me, but I am always interested in where my software travels. Please send me an email (gsc@digiburo.com) of your success or suggestions.

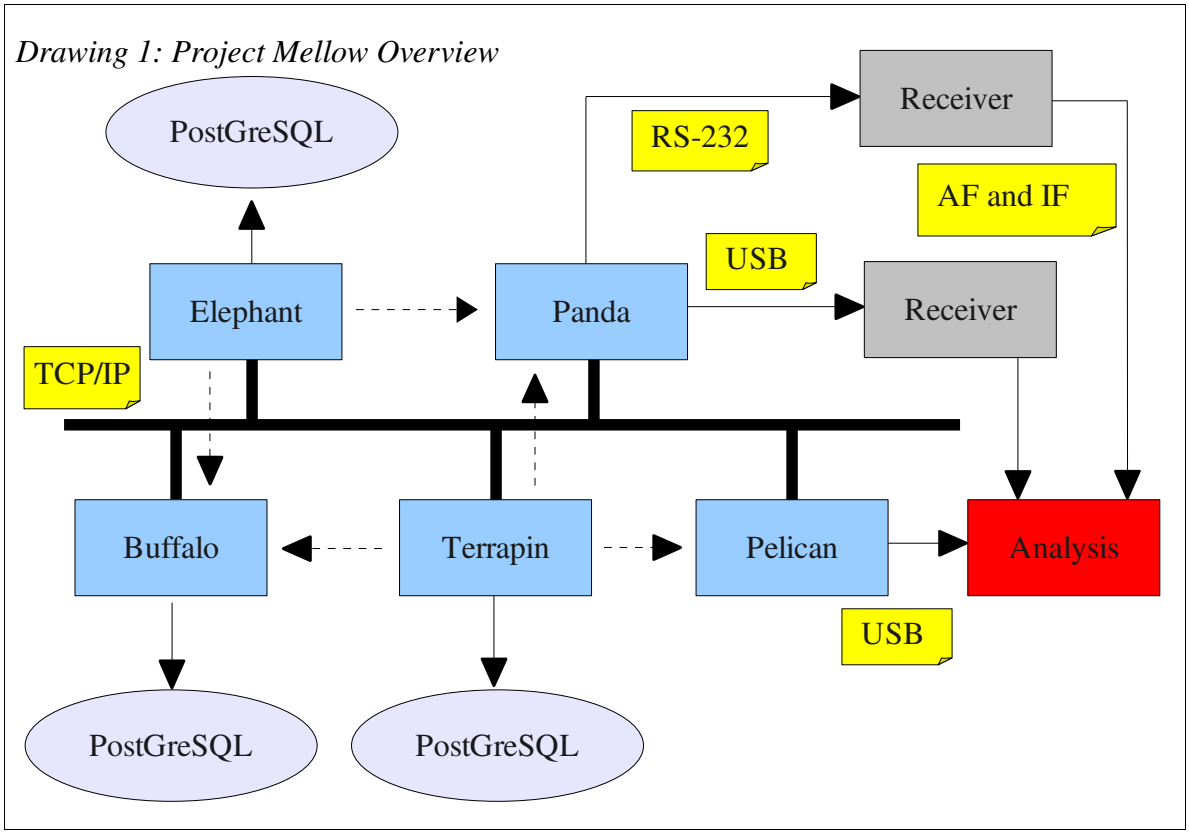
Please Note: I have no affiliation w/any of the radio manufacturers mentioned, apart from being a (more or less) satisfied customer. All information and software is for entertainment purposes only. Use at your own risk, there is no warranty (be fair, there was zero cost to you). Be sure to make back ups, etc. before using Mellow Panda. Thank you for reading.

Project Mellow

Mellow Panda is part of a larger collection of “Mellow” projects, which I am making available as time permits.

- Mellow Buffalo = Common Database
- Mellow Elephant = Energy Search
- Mellow Panda = Receiver Gateway
- Mellow Pelican = Software Defined Radio Lab
- Mellow Terrapin = Manual Collection

Drawing 1: Project Mellow Overview



Mellow Panda

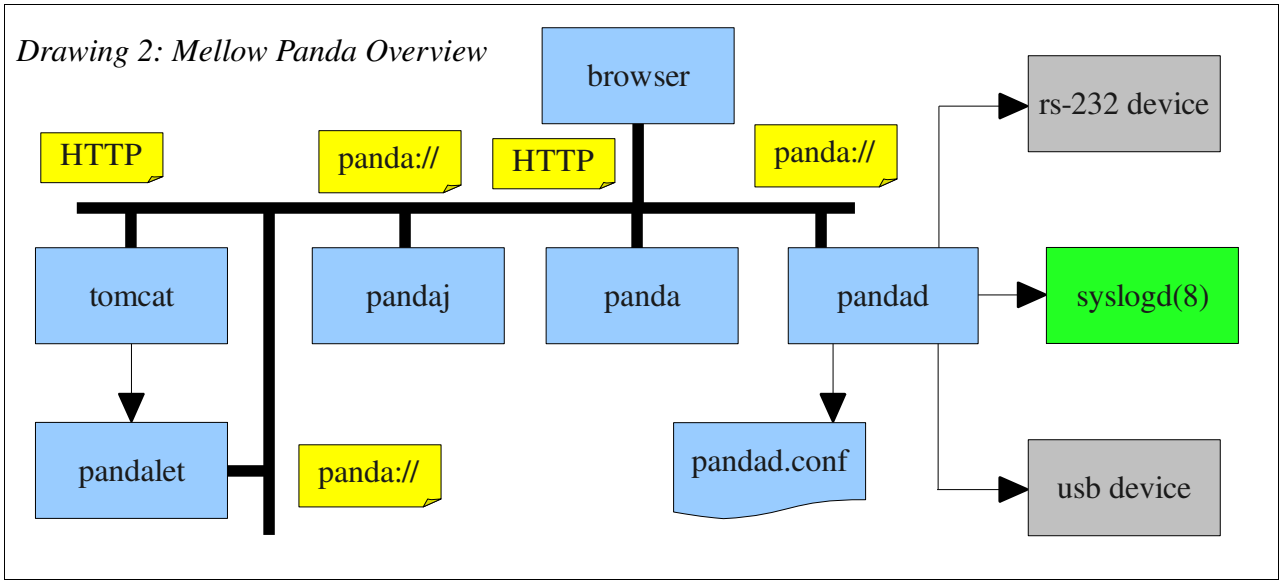
Mellow Panda consists of a background daemon (pandad) and specialized (radio model specific) clients. The distribution contains a general purpose client which employs a command line interface. There are additional clients available as a separate download such as pandaj (nice GUI, implemented in Java) and pandalet (a Servlet).

Mellow Panda Goals:

- Does not require “Brand W”
- Support for remote radio operation (via network).
  - HTTP support - control via curl(1) or wget(1).
- Control radios via scripts, scripting languages.
- GUI for interactive desktop control.
- Support for loading/collecting radio information (scan lists, etc).
- Simultaneous operation of multiple radios.
  - Radios can be of different manufacture/model.
- Control of RS-232/USB devices.
- Diagnostic support for serial device test and integration.
- Maintainable sources.
- Portable build.

One of the design goals for Mellow Panda was to support radios of different model and manufacture. I noticed that although different radios had unique feature sets and commands, they all used simple ASCII strings terminated by a “newline” (0x0a) or a “carriage return” (0x0d). The server could be generic (little abstraction, does not interpret the radio commands, apart from line terminators), and the clients could be radio model specific. As a result, pandad can control any generic serial device which follows the same communications model (write a string command, read the string response).

Drawing 2: Mellow Panda Overview



Panda Daemon (pandad)

The panda daemon (pandad) is a background process which provides a gateway between network clients and serial (RS-232 or USB) devices. Clients write to pandad using a TCP/IP stream socket (called a "command" socket). pandad will relay the command to the radio via RS-232/USB. Any response from the radio will be returned to the client via pandad.

The protocol between client and pandad is a simple string (more fully described below in "panda protocol"). It is possible to control your receivers using pandad and a telnet client to enter panda commands. Panda protocol defines a limited subset of generic commands such as "ping" but most of your activity will be via the "raw" command which sends unadulterated commands to your radio and writes the raw results back to your client.

The simple request/response model is adequate for many, but not all receivers. Some scanners feature asynchronous messages varying from a simple "squelch" cue on the PRO-2052 to the "band scan" (a type of spectral display) supported by the Icom PCR-1000/PCR-1500 radios. To get these asynchronous messages back to the client, pandad creates a UDP socket (called a "update" socket) to broadcast updates for each radio.

Consider the traffic from a radio to Mellow Panda as a stream. Asynchronous messages may be inserted into the stream at any time. While responding to a command, Mellow Panda will pull the next message from the serial port. The next message is usually in response to a command, but it might be something completely out of the blue. To solve this problem, all traffic from the radio is echoed on the "update" socket. The client can collect collect and parse the radio traffic (asynchronous and synchronous) and process accordingly.

pandad handles signals in the usual way. A SIGHUP will cause a full restart (closes all sockets and RS-232/USB ports, reads configuration file and opens the sockets/ports again). A SIGTERM will cause a graceful shutdown.

Configuration information is kept in a external file: pandad.conf (more fully described below). The configuration file maps radio names/types to serial port assignments both RS-232 and USB. A problem is that USB assignments can be rather dynamic and pandad makes no effort to discover USB assignments. If you define a radio as being at /dev/ttyUSB1, pandad expects it at USB1 no matter what the reality might be. (Perhaps this situation will change in the future).

pandad is implemented in C for portability reasons. I wanted painless compilation without first having to collect the dependencies and I didn't want to manage external libraries as they change over time. The sources have a dedicated section below.

Panda Daemon Command Line

pandad command line supports these options:

•-c filename = (configuration) specify non-standard configuration file

•-d = (diagnostic) write status to stdout and exit, useful for verifying configuration

•-f = (foreground mode) run daemon in foreground

•-h name = (host) start daemon using this host name

•-p port = (port) start daemon using this non-standard port

•-x command = (experimental) write a test command to the serial device and exit.

•-V = (version) write version information and exit

-x can be used to test radio connectivty and command processing.

•This example will cause pandad to “ping” radio0bc (defined in the sample pandad.conf as a Bearcat 780):

◦./pandad -x panda://radio/ping/radio0bc

•This example will cause pandad to “tune” radio1pcr to my local airport ATIS on 124.1MHz.

◦./pandad -x panda://radio/raw/radio1pcr/K00124100000020300

Panda Client (panda)

The panda client (panda) is a generic command line client which is useful for testing and scripting. There are several demonstration scripts located in the “demo” directory which illustrate the use of panda client.

Panda Client (panda) “Command Line”

The panda command line client is useful for testing and controlling receivers from scripts. These are the command line arguments:

•-c filename = (configuration) specify non-standard configuration file

•-d = (diagnostic) write status to stdout and exit, useful for testing configuration

•-f = (foreground mode) run daemon in foreground

•-h name = (host) start daemon using this host name

•-k command = (command) write a command to the mellow panda

•-p port = (port) start daemon using this non-standard port

•-r radio = (radio) command target (must match definition in pandad.conf)

•-V = (version) write version information and exit

This example sends a status request to the radio.

```
panda -r radio -k ping
```

This example tunes a bearcat radio to 124.1 MHz.

```
panda -r radio -k raw/RF01241000
```

Panda Protocol

Panda protocol is a simple one line string which describes the target device (by name as referenced within the configuration file) and the operation to perform. A simple telnet client is all you need to speak panda (in fact, I did the initial development using telnet).

Note that panda protocol is completely insecure with no encryption or authentication at all. You should not expose panda to any external network. If you need to use panda w/authentication and encryption, consider using pandalet servlet client and configure the servlet for HTTPS and authentication.

Panda protocol has the following form:

•panda://command/subcommand/device_name/argument

Examples:
•panda://auto_update/radio1bc
•panda://ping/radio1bc
•panda://device/raw/radio1bc/raw_command

Getting Started

1.Obtain latest Mellow Panda distribution from Source Forge.

2.Unpack distribution.

3.Type “./configure”

4.Type “make”

5.Type “make install” (probably requires root permissions)

•These files were installed:

◦/usr/local/etc/pandad.conf (pandad configuration)

◦/usr/local/sbin/pandad (pandad executable)

◦/usr/local/bin/panda (panda command line client)

1.Optional: copy dist/pandad start script to /etc/init.d

2.Optional: configure syslogd

◦Default facility is “local5”

◦Sample syslogd configuration:

■sample

1.Update pandad.conf (configuration file) to reflect your radios.

1.pandad.conf defines the attached radio types, serial port assignments, etc.

2.pandad will read pandad.conf at startup and whenever a SIGHUP is noted.

3.pandad.conf default location is /usr/local/etc but can an alternate location can be specified on the command line (-c filename).

4.pandad.conf has these elements:

1.A hash (“#”) means comment, anything after a hash is ignored.

2.Each radio has a unique index which associates the configurable elements

3.Each radio has a mnemonic name

4.Each radio has a type which is enumerated in utility.h/utility.c

5.Each radio has a RS-232 or serial port.

6.A radio can be marked as active “no” which will cause the configuration to be read (and ignored). To bring a radio online/offline, update pandad.conf and send pandad a SIGHUP to reread the configuration file.

7.Sample file:

•radio.0.name radio0bc

•radio.0.type bc780

•radio.0.serial /dev/ttyS0

•radio.0.active yes

•#

•radio.1.name radio1pcr

•radio.1.type pcr1000

•radio.1.serial /dev/ttyUSB0

•radio.1.active no

1.Test connectivity to your radios

1.use pandad command line to test your radios

2.monitor your progress w/syslogd(8)

2.Start pandad to run in background.

1.Now that the radios work, start pandad to run in background.

1./etc/init.d/pandad start (should require root permissions).

2.You might want to configure pandad to start at system boot.

3.End to end test (panda client to pandad to radio and back).

1.This example uses the command line client to ping your radio via pandad

2../panda -r radio0bc -k ping

3.Review the FAQ section if your radios did not respond to panda.

4.At this point your radios are accesible via panda/pandad. Check out the demo directory for example scripts.

Panda Client FAQ

•//placeholder//

Panda Daemon FAQ

•If you define the hostname as “localhost” in pandad.conf, then pandad will bind to the loopback. Specify the machine name for remote access.

•You can test access to the command socket w/telnet.

•If there are permission problems w/the serial devices then add yourself to the 'dialout' group.

pandad Source Code Companion

At runtime pandad follows the standard UNIX daemon model: i.e. read a configuration file, resolve

command line options then fork(2) to run in background (establish signal handlers, closing file descriptors, etc along the way).

Mellow Panda uses a TCP/IP stream socket ("command" socket) to communicate w/clients and USB/RS-232 ports to drive the receivers. After initialization pandad falls into a select(2) loop which polls the "command" socket and serial file descriptors.

An external command would be read from the command socket (detected by select(2)) and then passed to the specified radio (assuming the command passed format checks). The radio will generate a response which will be repackaged and dispatched back to the client (and echoed on the "update" socket). This is the complete request/response cycle. Done.

It is possible for the radio to generate an asynchronous message which select(2) will detect and write to the update socket.

To reread the configuration file, use SIGHUP and for a graceful exit use SIGTERM.

Panda Java Client (pandaj)

//place holder

Panda Servlet (pandalet)

//place holder

Panda Perl (pandapl)

//place holder

Panda Python (pandapy)

pandapy is a collection of applications implemented in python (v2.5.2). This example is pure python, including the TCP and UDP socket handling. pandapy is a optional download.

Thanks

A big "thank you" to the David Anderson (GM4JJJ) web site which was a great help w/making the PCR-1000 behave.

Also thanks to Ian Johnston and Matthew Cashdollar (the authors of sctl) which was useful for reaching my BC-780 scanners.

About Us

Digital Burro offers professional services and product development for UNIX platforms such as LINUX, Solaris, Mac OS X, iPhone and Google Android. Please contact us if you need assistance for your next project.

“Mellow Panda” doesn't mean anything, I pick these names and create the logos solely for amusement value. Any resemblance between Mellow Panda and a real project or application, living or dead is purely coincidence.