rina-cat notes
Steve Bunch, 23 July 2017 — Alpha Version

The tar file:

The tar file contents can be installed in …/stack/rina-tools/src/rina-cat, a new directory, where it can become part of the normal user-level build process, or can be built outside the tree using the included makefile if desired. The contents include:

.cc and .h source files

build-changes

A git diff showing the two one-line edits to add rina-cat to the build process.

Makefile.am

The config file to build rina-cat as part of the normal build.

makefile

A quick-and-dirty makefile to allow building rina-cat outside the irati build process. You may have to edit it to locate the tclap header files, and/or may have to run as root to reach them (comments included should be self-explanatory). (Not used if installed in the tree.)

rina-cat Operation:

Overview

By default, rina-cat copies bidirectionally between its own standard input and standard output files to/from a RINA flow. rina-cat can be used in either client (initiator) or server (listener/target) mode. rina-cat can be used like netcat(nc), but since it copies in both directions, it has many other uses. rina-cat can instead launch applications when run in either client or server mode, exec'ing a separate process whose standard input and output are connected to a RINA flow. In server mode, it can launch multiple simultaneous processes, one for each incoming flow request. Using standard UNIX/Linux commands, rina-cat can implement a bi-directional chat, perform file copying, launch server instances, and perform many other functions without writing custom programs. See the Examples. To get a list of the command-line options, type;

```
rina-cat --help
```

Client Mode

In client mode, the default, rina-cat attempts to create a RINA flow to a specified destination RINA application. If successful, in its simplest mode of operation, it then copies its own standard input to the flow, and simultaneously reads from the flow and copies to its standard output file. An EOF indication or error on any file terminates the execution. Optionally, rather than using its own standard input/output, the client can launch a separate command, provided on the rina-cat command line using the -c option. rina-cat will connect the flow to the command's standard input and output, wait for it to complete, then either terminate or persist (see the -p option).

Server Mode

In server(listen) mode, designated using the -l option, rina-cat registers an application name, then awaits incoming flow requests. When one is received, it vets it (more below

on that) and if happy with the request, accepts the flow. In its simplest mode of operation, it then copies data from the flow to its standard output, simultaneously reads from its standard input and copies it to the flow, and when EOF or error is reached on any file, terminates. In server mode, rather than copying to/from its own standard output/input, rina-cat can instead launch a separate application and attach the flow to its standard input and output, and let that application handle the flow and terminate when the client application completes. If a persistence option is included, the server rina-cat will instead remain operating indefinitely, launching and monitoring up to the specified maximum number of simultaneous applications as flow requests arrive. Flow requests above the command-line-set number will be refused.

Misc. Details

When launching a separate Linux application in either client or server mode, rina-cat never redirects the standard error output file, so errors from the rina-cat program as well as any children will NOT go over the RINA flow. (Note that stderr for launched commands can be redirected on its command line.)

Commands launched by rina-cat are placed into the same process group as rina-cat, so signals such as SIGTERM (control-C) sent to rina-cat will also be sent to any running command(s). (If this behavior is not desired, the commands are free to detach themselves, or we can in future provide an auto-detach option to rina-cat.)

The command line provided to rina-cat via the -c option is sent to the current default shell ($SHELL) for execution, using its -c command option. That is,

$SHELL -c “string provided using rina-cat -c option”

Therefore, the command line can use environment variables, pipes, redirection, or any other typical shell command line function. Some of the command line options of rina-cat as well as information about incoming connections to a server are provided to commands via environment variables (see example below). Users need to be careful of quoting conventions, etc., when using complex commands such as those including environment variables to be expanded.

Run rina-cat —help to get a list of the command options.

Examples:

Simple two-way “chat” session between a client and server: Copy input from standard input on one system to standard output on another, and vice versa.

Client:

```
# rina-cat -a dest_app -d normal.DIF
```

Server:

```
# rina-cat -l -A dest-app -d normal.DIF
```

Input typed to the standard input of the client instance will be sent over the RINA flow, and will be sent to standard output on the server instance, and vice versa. (The -d DIFNAME option may not be needed.) An EOF (control-D) at either end will end the

conversation.

To do a file copy, just redirect the input and output to files:
Client:

```
# rina-cat -a dest_app -d normal.DIF <filename
```

Server:

```
# rina-cat -l -A dest-app -d normal.DIF >filename
```

Note that the copy could have instead been done in the other direction, or both directions simultaneously, as both stdin and stdout are redirected at both ends.

Copy a video stream from a Raspberry Pi camera to the display of another Raspberry Pi.

Client:

```
# raspivid -fps 10 -w 800 -h 600 -t 0 -o - | rina-cat -a display -d
normal.DIF
```

Server:

```
# rina-cat -l -A display -d normal.DIF | mplayer -fps 10 -cache 32 -vo
x11 -
```

Run a persistent video server (an improvement over the last example, as it doesn't buffer data in a pipe, and allows player commands to be fed back to the source, in cases where that's useful).

Client:

```
# rina-cat -a server_app -c "mplayer -fps 10 -cache 32 -vo x11 -"
```

Server (persistent, allowing only one active flow at a time):

```
# rina-cat -l -A server_app -p 1 -c "raspivid -fps 10 -w 800 -h 600 -t
0 -o -"
```

(NOTE: the raspivid program sometimes enters a state where it quits sending video, probably when the network can't keep up. The pipe version above, or running it over a network with more buffering, can avert this problem somewhat, but it's a bug to be worked around at this time.)

Commands executed by rina-cat have access to information about the rina-cat instance that ran them via environment variables. In particular (subject to change):

```
  RINACAT_A           Name of this rina-cat instance (-A argument)
  RINACAT_a           Name of other rina-cat instance (-a argument, or
incoming flow)
  RINACAT_d           Name of dif to use (-d argument)
  RINACAT_sdusize     Specified SDU size (-sdusize argument)
```

```
  RINACAT_l           If running as a server, “l”, else null (-l
option)
  RINACAT_v           Verbosity (0 = none, 1 = most, 2 = all) (-v and
-V options)
```

Client (on razzycam2):

rina-cat `-a razzycam4.rina-cat`

Server (on razzycam4):

rina-cat `-l -c 'echo -A $RINACAT_A -a $RINACAT_a -d $RINACAT_d -sdusize $RINACAT_sdusize -l $RINACAT_l -v $RINACAT_v'`

Output at client:

```
# rina-cat -a razzycam4.rina-cat
-A razzycam4.rina-cat -a razzycam2.rina-cat -d -sdusize 4096 -l l -v 0
#
```

(Note that in this example some parameters, such as the -A application name, have been allowed to default.)

TO-DO, as of 23 July 2017 (alpha) version:

rina-cat has been tested with TCP sockets, but for now it has run with Irati rina-api only a few times on Raspberry Pi, where the kernel crashes when data is sent down the flow. Testing on a non-Pi Irati system is next. If you would like to build it with TCP support instead of RINA, let me know and I'll provide you the substitute i/o library.

In client mode, rina-cat doesn't yet allow setting flow properties — it requests a reliable, in-order, stream flow for maximum compatibility with existing Linux commands.

rina-cat lets a server specify the sole application name that is allowed to connect to it (using the optional -a option), and will refuse flow requests from any other app. This is untested. It should include some wild-card name flexibility, and include setting requirements on incoming flow specs (e.g., reliable vs unreliable), via command-line options since only the person specifying the application receiving rina-cat's output knows what it needs. Details will be finalized when the behavior of the rina-api library with application and AE names and instances becomes clearer.

rina-cat does not modify default signal behavior for rina-cat or launched commands.

Exception handling is minimal, so any exceptions thrown by internal libraries used by the RINA API will be minimally documented in error output.