Home

2019-02-17 - russng v6.8 release on bitbucket.org.

Contents

What is RUSS?

RUSS is an alternative to HTTP/web technologies for services running on UNIX/Linux.

RUSS is a protocol and framework for building service-oriented servers using UNIX/Domain sockets.

RUSS is built on some familiar ideas:

- orthogonal operations: execute, help, list
- service path: /-separated list of strings identifying a service and how to get there
- ordered list of string arguments (aka positional arguments)
- unordered collection of string attritubes as key=value pairs (like environment variables)
- exit/return value
- stream I/O over file descriptors (stdin, stdout, stderr)

The benefits of using UNIX/Domain sockets are:

- performance
- standard part of UNIX/Linux (no kernel modules needed)
- credentials are mediated by the OS
- connection between independent processes (even between different users)
- passing of descriptors between independent processes (even between different users)

More Info

Get started with RUSS v6 - Quickstart Setup.

Further information for users and developers is available in the Documentation section:

- RUSS Specification
- RUSS v6 - User Guide
- RUSS v6 - Tools
- RUSS v6 - Core Servers
- pyruss - RUSS for the Python Programming Language
- goruss - RUSS for the Go Programming Language

Show Me Something

First things

- `+` - the area that system servers register at; usually under `/var/run/russ/services`
- `ruls` - command line tool to list servers/services (think `ls`)
- `ruhelp` - command line tool to get help information (think `man`)
- `ruexec` - command line tool to execute a service
- `pyruss` - Python bindings for the C API

Listing Servers/Services

What's available?

About

Name

RUSS - Services over UNIX Domain Sockets

Requirements

russng (C compiler), pyruss (Python 2), goruss (go, gccgo)
Linux, OSX, UNIX (AIX, FreeBSD)

License

Apache v2

Links

Repository, API Docs

Featured

RUSS v6 - Quickstart Setup
Oct 09, 2019 • updated by John • view change

RUSS v6 - rurun
Feb 17, 2019 • updated by John • view change

RUSS v6 - Server Configuration
Jan 11, 2019 • updated by John • view change

RUSS v6 - ruservice
Oct 20, 2018 • updated by John • view change

RUSS v6 - Tools
Aug 19, 2018 • updated by John • view change

RUSS v6 - ruspawn
Aug 19, 2018 • created by John

RUSS v6 - rumpirun
Aug 19, 2018 • created by John

RUSS v6 - rudial
Aug 19, 2018 • created by John

RUSS v6 - ssh / sshr Server
Aug 19, 2018 • created by John

RUSS v6 - pnet Server
Aug 19, 2018 • created by John

RUSS v6 - User Guide
Feb 07, 2018 • updated by John • view change

```
$ ruls +
debug
exec
proc
set
ssh
tee
```

What services does the debug server provide?

```
$ ruls +/debug
chargen
conn
daytime
discard
echo
env
exit
request
```

Getting Help - Built in Man Page

How do I use the debug services?

```
$ ruhelp +/debug
Provides services useful for debugging. Unless otherwise stated,
stdin, stdout, and stderr all refer to the file descriptor triple
that is returned from a russ_dial call.

/chargen[/...]
    Character generator outputting to stdout; follows the RFC 864
    the RFC 864 protocol sequence.

/conn[/...]
    Outputs russ connection information.

/daytime
    Outputs the date and time to the stdout.

/discard[/...] [--perf]
    Discards all data received from stdin; if --perf is specified,
    performance feedback is provide to stderr, otherwise there is
    none.

/echo[/...]
    Simple echo service; receives from stdin and outputs to stdout.

/env
    Outputs environ entries to stdout.

/exit <value>
    Return with given exit value (between 0 and 255).

/request[/...]
    Outputs the request information at the server stdout.
```

Running a Service

Try the character generator:

```
$ ruexec +/debug/chargen
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefgh
"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghi
#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghij
^C
```

Show "request" information (as received and sent back by the server):

```
$ ruexec -a X=123 -a Y=abc +/debug/request hello there world
protocol string (0010)
spath (/request)
op (execute)
opnum (2)
attrv[0] (X=123)
attrv[1] (Y=abc)
argv[0] (hello)
argv[1] (there)
argv[2] (world)
```

Call the `daytime` service:

```
$ ruexec +/debug/daytime
Friday, February 16, 2018 11:45:50-GMT
```

Call the `daytime` service on another machine "buddy" (`ssh` must work without user interaction):

```
$ ruexec +/ssh/buddy/+/debug/daytime
Friday, February 16, 2018 11:46:55-GMT
```

Call the `daytime` service from Python:

```
$ PYTHONPATH=/usr/lib/russng python2
>>> import pyruss
>>> rv, ev, out, err = pyruss.execv_wait_inouterr_timeout(1000, "+/ssh
/buddy/+/debug/daytime")
>>> print out
Friday, February 16, 2018 11:48:23-GMT
```

Echo a message, hopping through three machines "buddy", "bobby", and "bibby" (as before, `ssh` must work without user interaction):

```
$ echo "hop hop hop" | ruexec +/ssh/buddy/+/ssh/bobby/+/ssh/bibby/+/debug
/echo
hop hop hop
```