

# RTEMS Project Ecosystem

**Joel Sherrill, Ph.D.**

[Joel.Sherrill@oarcorp.com](mailto:Joel.Sherrill@oarcorp.com)

OAR Corporation  
Huntsville Alabama USA

December 2014

# Ecosystem Background

- Objectives:
  - provide a fully repeatable build from source to a tested, working application base.
  - lower entry barrier and increase host support
  - consistent user interface
- Definition:
  - framework that provides a collection of tools to complete this process
- Status:
  - ongoing and active work, evolving to meet needs

# Environment Reproducibility

- User produce and test RTEMS and tools from source
  - complete source
  - RTEMS Source Builder
  - RTEMS Tester
- Same infrastructure used by
  - developers for developmental testing
  - project for Continuous Integration Testing
- Plan to publish results so user can compare their results against official results from RTEMS Project

# Ecosystem Deliverables

- Host tools
  - development tools, simulators, etc. with source, patches, and locally built binaries
- Target libraries
  - RTEMS and third party add-on libraries source, patches, and locally built BSP specific binaries
- Reports
  - configuration, build, and test
- Support smooth integration of RTEMS into project specific configuration management

# Hosting and Infrastructure Refresh

- Half-rack of new hardware purchased with donations for GSoC and GCI participation
- Hosting now at Oregon State University Open Source Laboratory (<https://osuosl.org/>)
  - couple of hops off Internet2 backbone, very fast
- Multiple single purpose VMs versus multi-purpose single host
  - this did cause some URLs to break
- Conversion from Bugzilla and Mediawiki to Trac
  - Trac provides release status tracking and reports
  - commit closes ticket, sends email to list, updates github, and sends message to IRC
- Future Plans
  - need another build machine dedicated to building tools
  - need more RAM and storage for machines (purchased with space to grow)
  - Mac for testing as host

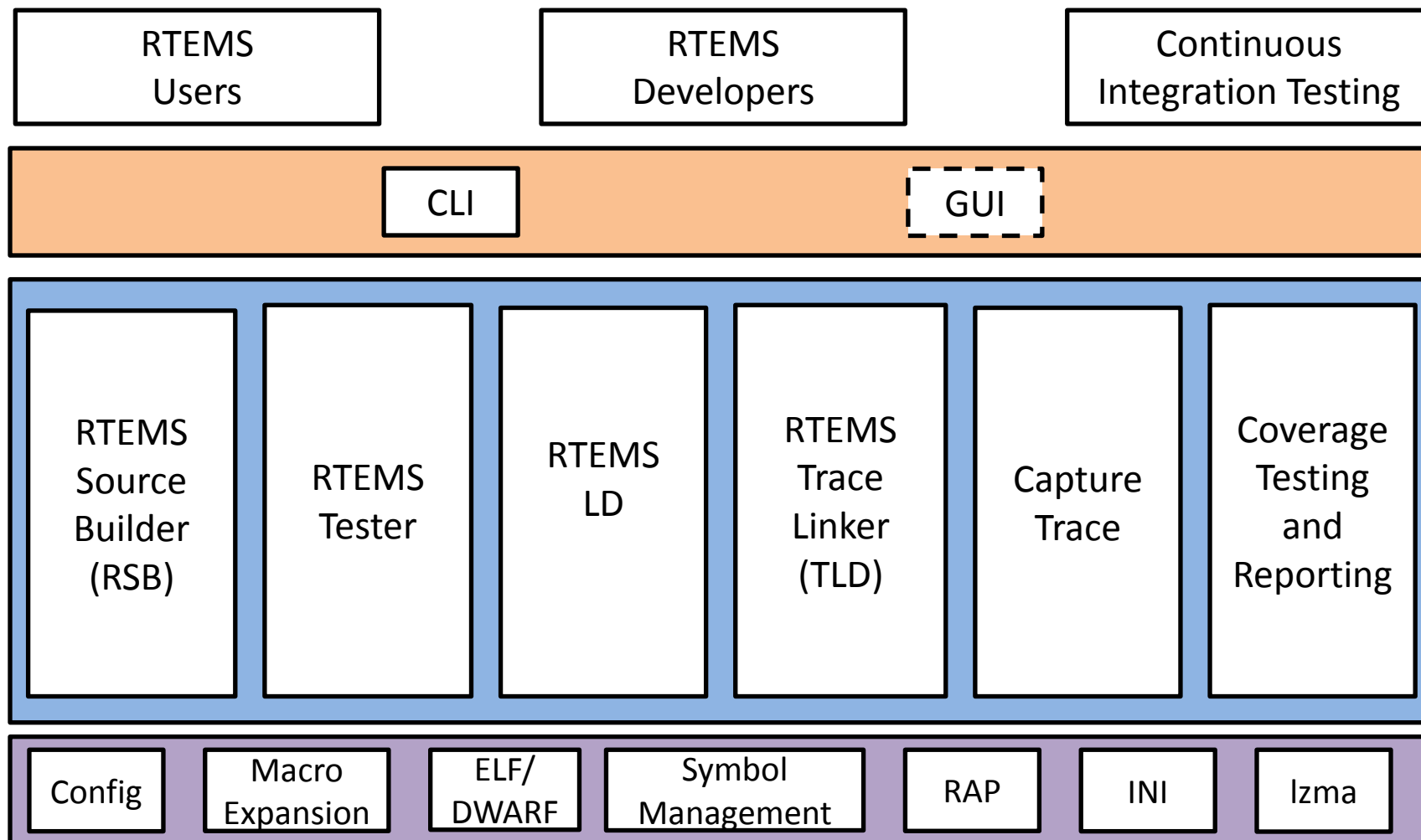
**All volunteer effort! Please pitch in!**

# Process Refresh

- New infrastructure enables new capabilities
- Future plans
  - incorporate web based patch review and submission system
    - eventually a patch will be tested before a human sees it
  - establish continuous integration and testing so every commit results in an appropriate test sweep
  - provide database of test results
  - refresh user documentation in modern tools
    - current effort to review content in Wiki

**Anything else we use computers to help with**

# RTEMS Tools Project



# RTEMS Source Builder (RSB)

- RSB contains “recipes” for building host tools and target packages from source
  - enables configuration management of source and patches
- Host independent way to obtain environment
- Full source and patches
- Use RSB to fetch source and build
  - RTEMS tools
  - cross development environment
  - RTEMS (or by hand)
  - simulators if desired
  - third Party Packages for target system



# RSB Example

- Obtain RTEMS RSB from git or release
  - `git clone git://git.rtems.org/rtems-source-builder.git`
- Check basic host environment
  - `cd rtems-source-builder`
  - `source-builder/sb-check`
- Build SPARC toolset
  - `cd rtems`
  - `../source-builder/sb-set-builder --log=l-sparc.txt \`  
`--prefix=$HOME/development/rtems/4.11 4.11/rtems-sparc`
- Takes approximately 17 minutes on a (modest) quad-core machine

# RTEMS Tester

- Automates testing on simulators and hardware using GDB Machine Interface (MI)
  - direct execution using various simulators
  - takes advantage of multiple cores to parallelize simulator testing
- Execute RTEMS Test Suite

```
~/development/rtems/test/rtems-tools.git/tester/rtems-test \  
  --log=log_sis_run  --rtems-bsp=sis-run \  
  --rtems-tools=$HOME/development/rtems/4.11 \  
  sparc-rtems4.11/c/sis/testsuites
```
- Takes ~24 minutes single threaded but 11.3 minutes with tester on modest 2.4Ghz quad code
  - Note: end up waiting for longest to complete or timeout
- Future plans
  - reduce run-time of longest running tests
  - automatically calibrate performance of simulators and scale the timeouts
  - execute GCC Test Suite
  - execute tests on other components
  - compare results with RTEMS Project
    - requires RTEMS Project to provide a populated test results database

# RTEMS Coverage Testing

- Tools to analyze and report RTEMS Test Coverage
  - primary tool is *covoar*
- Test coverage itself is in good shape
  - Improved reporting and integration into framework is needed
- Current status
  - original scripts not in framework. ESA SOCIS student rewrote initial version for framework
  - original scripts had rigid reporting that was not granular enough. New version addresses that
- Future plans
  - bring framework version up to production standards so they can be integrated
  - finer grained (e.g. by directory) coverage reports

# RTEMS and Tools Milestone

- Tested RTEMS Environment in place
- Provides fully tested application base with
  - full source and binaries
  - configuration reports
  - test reports including coverage
  - build reports
- RTEMS user can use ecosystem to support their project

# RTEMS Continuous Integration Testing

- RSB provides uniform way to build tools, infrastructure, RTEMS and add-on packages
- Eases integration into a Continuous Integration Framework
- Prototype *buildbot* instance demonstration
- Future Plans
  - web based interface for submitting patches
  - patches tested BEFORE seen by reviewers
  - full test results available to entire community

# Target Focused Tools

---

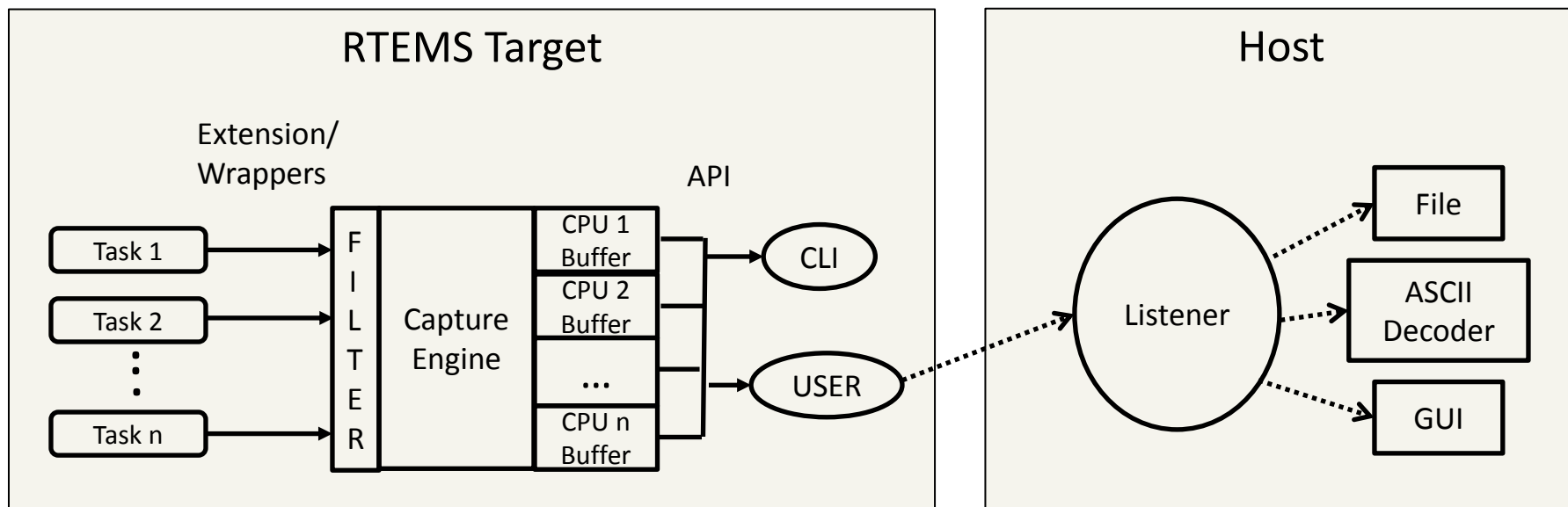
- Event tracing and recording
  - capture engine
  - trace wrapper generation
- Run-Time Loader
- GDB Pretty Printers
  
- Most of these capabilities require combination of target and host side software

# Capture Engine

- Capture trace data from
  - user extensions: thread create, start, restart, delete, switch, begin, exit, and terminate
  - any method invocation with trace wrapper generation
- Trace information is buffered on a per CPU basis to avoid locking during recording
- Can be printed via CLI or transmitted to host computer for further analysis
- Future plans
  - improve trace filtering and triggering

# Capture Engine Usage

- Requires host and target support
- Run-time captures and buffers trace records
- Target has API and CLI to manage trace records
- If buffers transferred to host, other tools can be used to examine records





# Thread Migration Capture Example

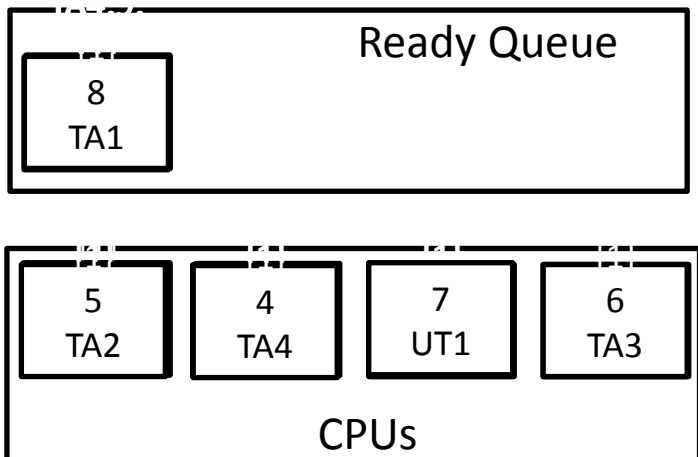
TaskName:Priority:affinity  
 affinity = {cpu,cpu}

(1) UT1:7:{2,3}

(2) TA1:8:{2,3}      (3) TA2:5:{0,1}

(4) TA3:6:{0,3}      (5) TA4:9:{1}

(6) Raise priority of TA4:4:{1} &  
 trigger series of migrations



## Capture Engine Output

```

1 0:00:00.008653000      0a010003 TA02   5 5   5 4096 TASK_RECORD
0 0:00:00.008659000      0a010004 TA03   6 6   6 4096 TASK_RECORD
2 0:00:00.008663000      0a010002 TA01   8 8   8 4096 TASK_RECORD
1 0:00:00.008681000      0 0a010003     5 5           SWITCHED_IN
0 0:00:00.008686000      0 0a010004     6 6           SWITCHED_IN
2 0:00:00.008691000      0 0a010002     8 8           SWITCHED_IN
1 0:00:00.008734000      53000 0a010003     5 5           BEGIN
0 0:00:00.008738000      52000 0a010004     6 6           BEGIN
2 0:00:00.008743000      52000 0a010002     8 8           BEGIN
3 0:00:00.008914000      0a010001 UI1    7 7   7 4096 TASK_RECORD
3 0:00:00.008943000      0 0a010001     7 7           CREATED_BY
3 0:00:00.009015000      0a010005 TA04   9 9   9 4096 TASK_RECORD
3 0:00:00.009041000      98000 0a010005     9 9           CREATED
3 0:00:00.009298000      257000 0a010001    7 7           STARTED_BY
3 0:00:00.009326000      28000 0a010005     9 9           STARTED
3 0:00:01.000432000    991106000 0a010001    7 7           SWITCHED_OUT
1 0:00:01.000452000    991718000 0a010003     5 5           SWITCHED_OUT
3 0:00:01.000456000      24000 0a010004     6 6           SWITCHED_IN
0 0:00:01.000473000    991735000 0a010004     6 6           SWITCHED_OUT
1 0:00:01.000476000      24000 0a010005     4 4           SWITCHED_IN
2 0:00:01.000491000    991748000 0a010002     8 8           SWITCHED_OUT
0 0:00:01.000496000      23000 0a010003     5 5           SWITCHED_IN
2 0:00:01.000514000      0 0a010001     7 7           SWITCHED_IN
    
```

# LTTng/RTEMS Proof of Concept #1

- Aeroflex/Gaisler PoC of MTAPI Demo on NGMP

The screenshot shows the Eclipse IDE interface with the LTTng traces loaded. The main window displays a table of trace events for the data3 trace. The table has the following columns: Timestamp, Source, Type, File, and Content. The trace shows a sequence of events including mtapi\_enter and mtapi\_exit for various nodes (0, 1, 2, 3) and wait\_all events. The Project Explorer on the left shows the trace data3 and its analysis options.

Timestamp	Source	Type	File	Content
<filter>	<filter>	.*mtapi.*	<filter>	<filter>
2966/13170				
01:00:04.522 900 000	0	mtapi_enter	trace	nodeid=0
01:00:04.537 598 000	0	mtapi_enter	trace	nodeid=1
01:00:04.551 215 000	0	mtapi_enter	trace	nodeid=3
01:00:04.564 720 000	0	mtapi_enter	trace	nodeid=2
01:00:04.564 830 000	0	mtapi_wait_all_enter	trace	nodeid=0
01:00:04.565 489 000	0	mtapi_wait_all_exit	trace	nodeid=0, pad=0, result=0
01:00:04.565 496 000	0	mtapi_exit	trace	nodeid=0
01:00:04.565 509 000	0	mtapi_exit	trace	nodeid=1
01:00:04.565 512 000	0	mtapi_exit	trace	nodeid=2
01:00:04.565 516 000	0	mtapi_exit	trace	nodeid=3
01:00:04.566 035 000	0	mtapi_enter	trace	nodeid=3
01:00:04.566 048 000	0	mtapi_enter	trace	nodeid=1
01:00:04.566 051 000	0	mtapi_enter	trace	nodeid=0
01:00:04.566 061 000	0	mtapi_enter	trace	nodeid=2
01:00:04.566 104 000	0	mtapi_wait_all_enter	trace	nodeid=0
01:00:04.566 749 000	0	mtapi_wait_all_exit	trace	nodeid=0, pad=0, result=0
01:00:04.566 755 000	0	mtapi_exit	trace	nodeid=0
01:00:04.566 766 000	0	mtapi_exit	trace	nodeid=2
01:00:04.566 769 000	0	mtapi_exit	trace	nodeid=1
01:00:04.566 772 000	0	mtapi_exit	trace	nodeid=3
01:00:04.567 035 000	0	mtapi_enter	trace	nodeid=3
01:00:04.567 043 000	0	mtapi_enter	trace	nodeid=1
01:00:04.567 047 000	0	mtapi_enter	trace	nodeid=2
01:00:04.567 053 000	0	mtapi_enter	trace	nodeid=0
01:00:04.567 086 000	0	mtapi_wait_all_enter	trace	nodeid=0
01:00:04.567 730 000	0	mtapi_wait_all_exit	trace	nodeid=0, pad=0, result=0
01:00:04.567 736 000	0	mtapi_exit	trace	nodeid=0

# LTTng/RTEMS Proof of Concept #2

Resource - Tracing/Traces/data3/data3\_ - Eclipse Platform

File Edit Navigate Search Project Run File Window Help Window Help

Project Explorer

- Tracing
- Experiments [0]
- Traces [1]
  - data3
    - MTAPI Analysis
    - Test a builtin XML module file
    - Tmf Statistics Analysis
      - Statistics

Outline Task Lis

An outline is not available.

Statistics

Global - data3

Level	Events total	Events in selection
data3	13,170	11,057
Event Types		
dispatch_enter	36 % 4,749	36.3 % 4,019
dispatch_exit	36 % 4,749	36.3 % 4,018
mtapi_enter	9 % 1,188	9.1 % 1,007
mtapi_exit	8.9 % 1,184	9.1 % 1,007
mtapi_wait_all_enter	2.2 % 297	2.2 % 251
mtapi_wait_all_exit	2.2 % 297	2.2 % 251
tick_enter	2.6 % 353	2.2 % 252
tick_exit	2.6 % 353	2.2 % 252

data2 data3

Timestamp	Source	Type	File	Content
<filter>	<filter>	<filter>	<filter>	<filter>
01:00:04.829 299 000	0	dispatch_exit	trace	nodeid=3, pad=0, jobid=4
01:00:04.829 307 000	0	dispatch_enter	trace	nodeid=3, pad=0, jobid=10
01:00:04.829 317 000	0	dispatch_exit	trace	nodeid=2, pad=0, jobid=7
01:00:04.829 325 000	0	dispatch_enter	trace	nodeid=2, pad=0, jobid=11
01:00:04.829 382 000	0	dispatch_exit	trace	nodeid=2, pad=0, jobid=11
01:00:04.829 385 000	0	dispatch_exit	trace	nodeid=1, pad=0, jobid=5
01:00:04.829 390 000	0	dispatch_enter	trace	nodeid=2, pad=0, jobid=17
01:00:04.829 398 000	0	dispatch_exit	trace	nodeid=0, pad=0, jobid=9
01:00:04.829 497 000	0	dispatch_exit	trace	nodeid=3, pad=0, jobid=10
01:00:04.829 732 000	0	dispatch_exit	trace	nodeid=2, pad=0, jobid=17
01:00:04.829 741 000	0	mtapi_wait_all_exit	trace	nodeid=0, pad=0, result=0
01:00:04.829 747 000	0	mtapi_exit	trace	nodeid=0
01:00:04.829 750 000	0	dispatch_enter	trace	nodeid=2, pad=0, jobid=13
01:00:04.829 753 000	0	dispatch_enter	trace	nodeid=3, pad=0, jobid=13
01:00:04.829 755 000	0	dispatch_enter	trace	nodeid=1, pad=0, jobid=13
01:00:04.829 758 000	0	mtapi_exit	trace	nodeid=2
01:00:04.829 761 000	0	mtapi_exit	trace	nodeid=3
01:00:04.829 764 000	0	mtapi_exit	trace	nodeid=1
01:00:04.830 008 000	0	tick_enter	trace	nodeid=0
01:00:04.830 013 000	0	tick_exit	trace	nodeid=0
01:00:04.830 036 000	0	mtapi_enter	trace	nodeid=1
01:00:04.830 041 000	0	dispatch_exit	trace	nodeid=1, pad=0, jobid=13
01:00:04.830 048 000	0	mtapi_enter	trace	nodeid=2
01:00:04.830 052 000	0	dispatch_exit	trace	nodeid=2, pad=0, jobid=13
01:00:04.830 055 000	0	mtapi_enter	trace	nodeid=0
01:00:04.830 060 000	0	mtapi_enter	trace	nodeid=3
01:00:04.830 065 000	0	dispatch_exit	trace	nodeid=3, pad=0, jobid=13
01:00:04.830 069 000	0	dispatch_enter	trace	nodeid=1, pad=0, jobid=0

MTAPI View

1970 Jan 01 01:00:04.578000 01:00:04.578500 01:00:04.579000 01:00:04.579500

data3

- Node 0: job 3, job 8, job 9, job 1, Waiting
- Node 1: job 1, job 5, job 12
- Node 3: job 0, job 10
- Node 2: job 2, job 4, job 7

job 3 job 8 job 9 job 1 job 12 job 2 job 4 job 5 job 10 job 7 job 11 job 10

# RTEMS Trace Linker

- Eases linking RTEMS applications
- Automated generation of method wrappers
  - user specifies methods to wrap and trace
  - user specifies pattern for wrapper methods
- Proof of concept has pattern which uses *printk()* to display method invocation and return values
- Future plans
  - wrapper pattern for method trace to capture logs
  - extend filter/trace for arbitrary sets of user traces

# Trace Visualization

- GUI visualization of event timeline
- Current status
  - identified existing FOSS tools focused on this
  - discussed with Linux Trace Toolkit developers how to add support for RTEMS trace visualization
- Future Plans
  - framework for mechanism to get trace to host
  - support Common Trace Format (CTF)
  - provide mapping of RTEMS events/objects to visual representation

# RTEMS Run-time Loader

- Base image plus RTEMS Application (RAP) modules loaded and linked at run-time
  - uses host-based tool and the run-time link editor to perform a similar function to static linking
- Designed for modest target hardware
- File Formats supported by
  - ELF - normal ELF objects and executable
  - RAP - LZ77 compressed modules
  - AR – Object libraries with GNU extensions
- Future Plans
  - Support for most architectures is in place testing and feedback are needed

# GDB Pretty Printers

- GDB now supports Python plugins
- Plugins can examine and analyze program variables and data structures
- Initial prototype to demonstrate feasibility
- Future Plans
  - user useful views of internal RTEMS information
  - methods to check consistency
  - replace RTEMS GDB macros with better versions written in a more capable language

# Conclusion

---

- Work is ongoing and evolving
- Evolution is driven by feedback
- Help us make these tools make your job easier



# Contacts and Acknowledgements

---

## Joel Sherrill, Ph.D.

OAR Corporation  
Huntsville Alabama USA  
[Joel.Sherrill@oarcorp.com](mailto:Joel.Sherrill@oarcorp.com)

### Gedare Bloom, Ph.D.

George Washington University  
Washington DC USA  
[gedare@rtems.org](mailto:gedare@rtems.org)

### Chris Johns

Contemporary Software  
Sydney Australia  
[chrisj@rtems.org](mailto:chrisj@rtems.org)

# Backup Slides

# Ecosystem Framework (figure)

- Bottom bar is framework
  - Features: configuration, macro expansions, ELF, symbol management, execute for parallelization, RAP (RTEMS Application Package) format, INI file, lzma
- Top bar is user interface, CLI and GUI dashed
- Above top bar: User, Continuous Integration Tester
- Vertical bars: RSB, RTEMS Tester, RTEMS LD, RTEMS Trace Linker (TLD)

# Capture Engine API Example

- Set capture engine to capture and print tasks with priority 0-20 related to TA00 while the application “run\_program” is executing.

```
sc = rtems_capture_open (5000, NULL);
sc = rtems_capture_watch_ceiling (0);
sc = rtems_capture_watch_floor (20);
sc = rtems_capture_watch_global (true);
sc = rtems_capture_set_trigger (
    0,
    0,
    rtems_build_name('T', 'A', '0', '0'),
    0,
    rtems_capture_from_any,
    rtems_capture_switch
);
sc = rtems_capture_control (true);
run_program();
sc = rtems_capture_control (false);
rtems_capture_print_trace_records ( 22, false );
```