Megatest/Logpro Training

Using the Megatest Regression & Automation Engine and the Logpro log file analysis tools to do robust QA and automation.

Matt Welland, 2013

Megatest Information

- Main development site: http://chiselapp.com/user/kiatoa/repository/megatest
- Mirror

  http://chiselapp.com/user/kiatoa/repository/megatest
- SourceForge Page

  http://sourceforge.com/projects/megatest

Training Overview

- Background on Megatest
- Getting started
  - Running tests and managing runs
  - Creating a Megatest area
  - Creating tests/tasks
  - Getting information about runs and tests
  - How to write Logpro files
- Advanced Megatest topics
- Future Megatest development

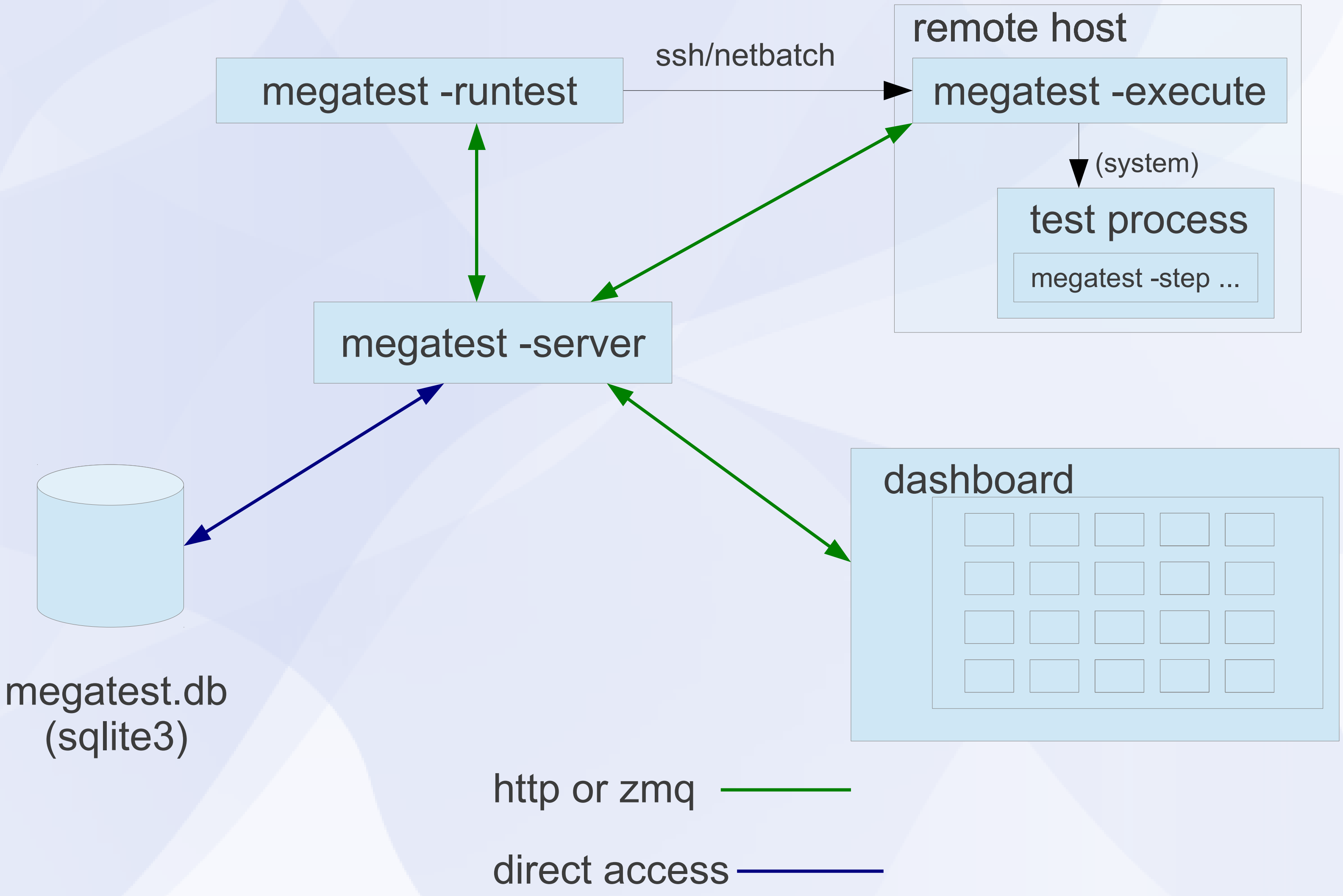
What Can Megatest Do?

- Run tests or tasks with
  - one or many steps
  - dependencies on other tests
  - on multiple hosts
  - iterated over variables
- Report, record and roll up
  - PASS, FAIL, WARN, CHECK
  - Test generated data details
- Organize runs by project specific variables

Megatest Architecture

- config files
  - megatest.config
  - runconfigs.config
  - tests/<testname>/testconfig
- SQL database
  - megatest.db
- Tools
  - megatest (command line), dashboard (gui), and logpro (log file analysis via rules)

How it Works



remote host

ssh/netbatch

megatest -runtest

megatest -execute

(system)

test process

megatest -step ...

megatest -server

dashboard

megatest.db
(sqlite3)

http or zmq

direct access

Terminology

target	one or more “keys” separated by “/”, used to organize runs hierarchically; examples include platform, architecture, stage (e.g. development, final QA, alpha, beta) and so forth. E.g target = x86/centos/dev where the keys are ARCHITECTURE, OS, and RELEASE
run name	a unique name (within a single target grouping) for a run, a common idiom is to use week and day numbers: date +%V.%u
run	a group of tests run under a single target and run name
test or task	a self-contained area with scripts and data to achieve some testing or automation goal
iterated test	a single test run multiple times with variables iterated over a range of values
state	the state of a test; NOT_STARTED, RUNNING, COMPLETED etc.
status	the current status of this test given its state; PASS, FAIL, n/a

Megatest Design Philosophy

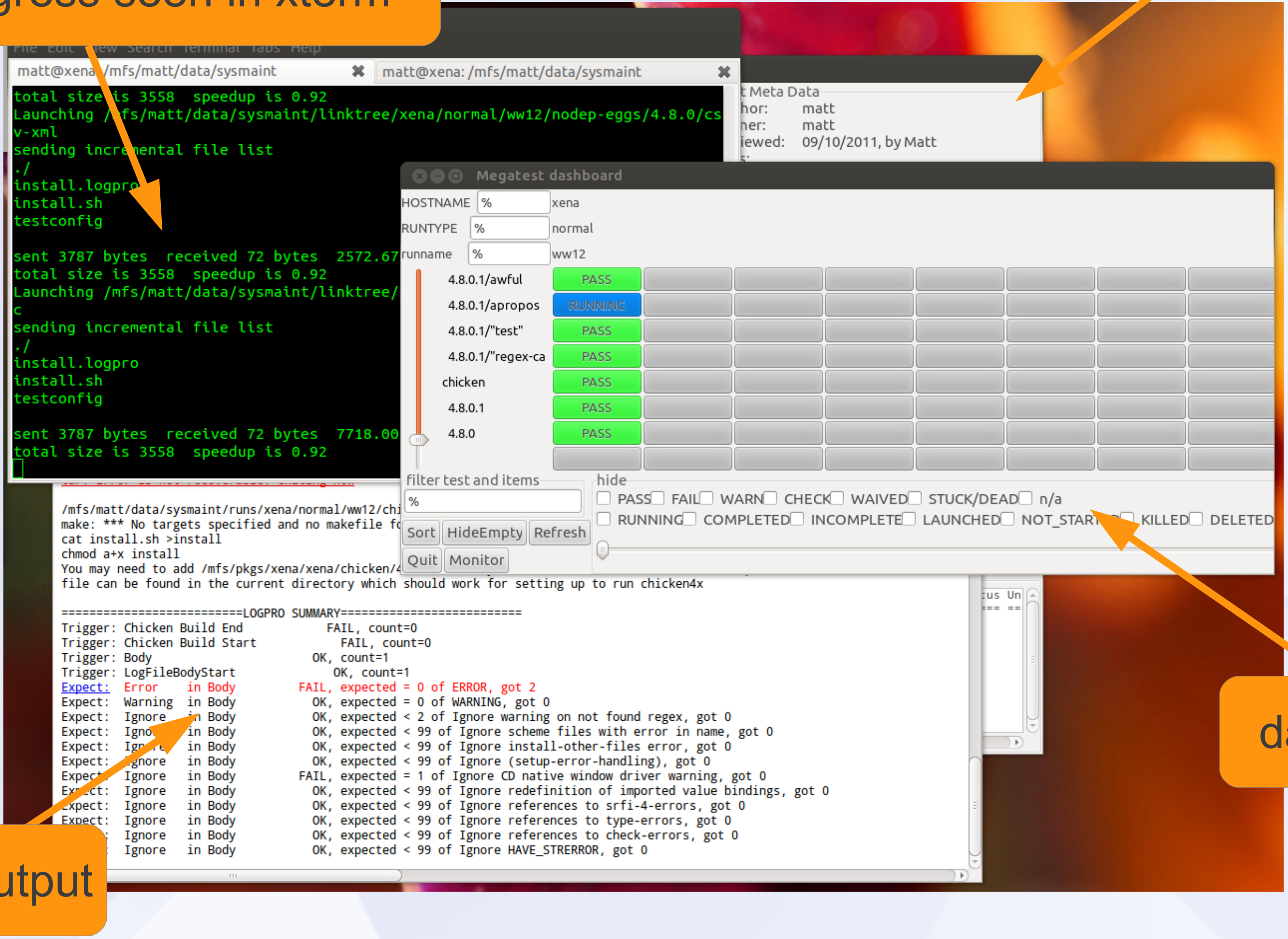
Deployable	anyone on the team, at any site, at any time can run the tests
Immutable	once this test is run it cannot be easily overwritten or accidentally modified.
Repeatable	this test result can be recreated in the future
Self-checking	strive for directed or self-checking test as opposed to delta based tests
Traceable	environment variables, host OS and other possibly influential variables are captured and kept recorded.
Encapsulated	the area where the test was run is self-contained and all inputs and outputs to the test can be found in the test run area.
Relocatable	the test area can be checked out and the tests run anywhere

Wisdom is knowing when it is ok to bend or break the rules.

Megatest strives to make it straightforward to do things right but still possible to get the job done when the rules must be bent or broken.

(yes, it is true, the highlighted letters don't spell anything meaningful!)

A Day in The Life ..



run progress seen in xterm

test control panel (in background)

dashboard

logpro output

File Edit View Search Terminal Tabs Help
matt@xena: /mfs/matt/data/sysmaint
matt@xena: /mfs/matt/data/sysmaint

```
total size is 3558  speedup is 0.92
Launching /mfs/matt/data/sysmaint/linktree/xena/normal/ww12/nodep-eggs/4.8.0/cs
v-xml
sending incremental file list
./
install.logpro
install.sh
testconfig

sent 3787 bytes  received 72 bytes  2572.67
total size is 3558  speedup is 0.92
Launching /mfs/matt/data/sysmaint/linktree/
c
sending incremental file list
./
install.logpro
install.sh
testconfig

sent 3787 bytes  received 72 bytes  7718.00
total size is 3558  speedup is 0.92
```

t Meta Data
hor: matt
ner: matt
iewed: 09/10/2011, by Matt

HOSTNAME % xena
RUNTYPE % normal
runname % ww12

```
4.8.0.1/awful       PASS
4.8.0.1/apropos     RUNNING
4.8.0.1/"test"      PASS
4.8.0.1/"regex-ca   PASS
chicken             PASS
4.8.0.1             PASS
4.8.0               PASS
```

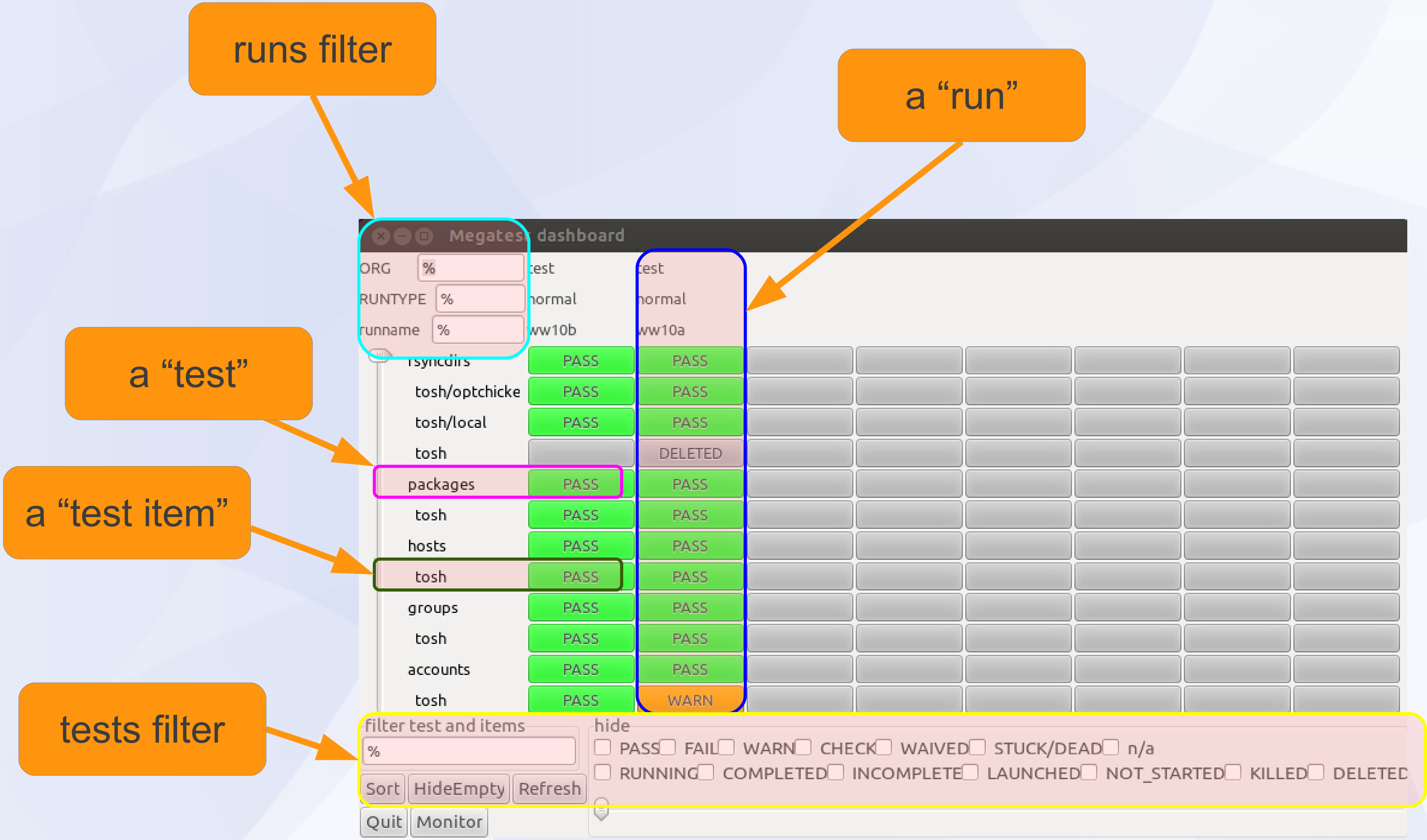
filter test and items
%

hide
□ PASS □ FAIL □ WARN □ CHECK □ WAIVED □ STUCK/DEAD □ n/a
□ RUNNING □ COMPLETED □ INCOMPLETE □ LAUNCHED □ NOT_STARTED □ KILLED □ DELETED

```
/mfs/matt/data/sysmaint/runs/xena/normal/ww12/chi
make: *** No targets specified and no makefile fo
cat install.sh >install
chmod a+x install
You may need to add /mfs/pkgs/xena/xena/chicken/4
file can be found in the current directory which should work for setting up to run chicken4x

=========================LOGPRO SUMMARY=========================
Trigger: Chicken Build End            FAIL, count=0
Trigger: Chicken Build Start            FAIL, count=0
Trigger: Body                         OK, count=1
Trigger: LogFileBodyStart               OK, count=1
Expect:  Error    in Body           FAIL, expected = 0 of ERROR, got 2
Expect:  Warning  in Body             OK, expected = 0 of WARNING, got 0
Expect:  Ignore   in Body             OK, expected < 2 of Ignore warning on not found regex, got 0
Expect:  Ignore   in Body             OK, expected < 99 of Ignore scheme files with error in name, got 0
Expect:  Ignore   in Body             OK, expected < 99 of Ignore install-other-files error, got 0
Expect:  Ignore   in Body             OK, expected < 99 of Ignore (setup-error-handling), got 0
Expect:  Ignore   in Body           FAIL, expected = 1 of Ignore CD native window driver warning, got 0
Expect:  Ignore   in Body             OK, expected < 99 of Ignore redefinition of imported value bindings, got 0
Expect:  Ignore   in Body             OK, expected < 99 of Ignore references to srfi-4-errors, got 0
Expect:  Ignore   in Body             OK, expected < 99 of Ignore references to type-errors, got 0
Expect:  Ignore   in Body             OK, expected < 99 of Ignore references to check-errors, got 0
Expect:  Ignore   in Body             OK, expected < 99 of Ignore HAVE_STRERROR, got 0
```

dashboard



runs filter

a “run”

a “test”

a “test item”

tests filter

ORG %

RUNTYPE %

runname %

	test	test
	normal	normal
	ww10b	ww10a
rsyncdirs	PASS	PASS
tosh/optchicke	PASS	PASS
tosh/local	PASS	PASS
tosh		DELETED
packages	PASS	PASS
tosh	PASS	PASS
hosts	PASS	PASS
tosh	PASS	PASS
groups	PASS	PASS
tosh	PASS	PASS
accounts	PASS	PASS
tosh	PASS	WARN

filter test and items

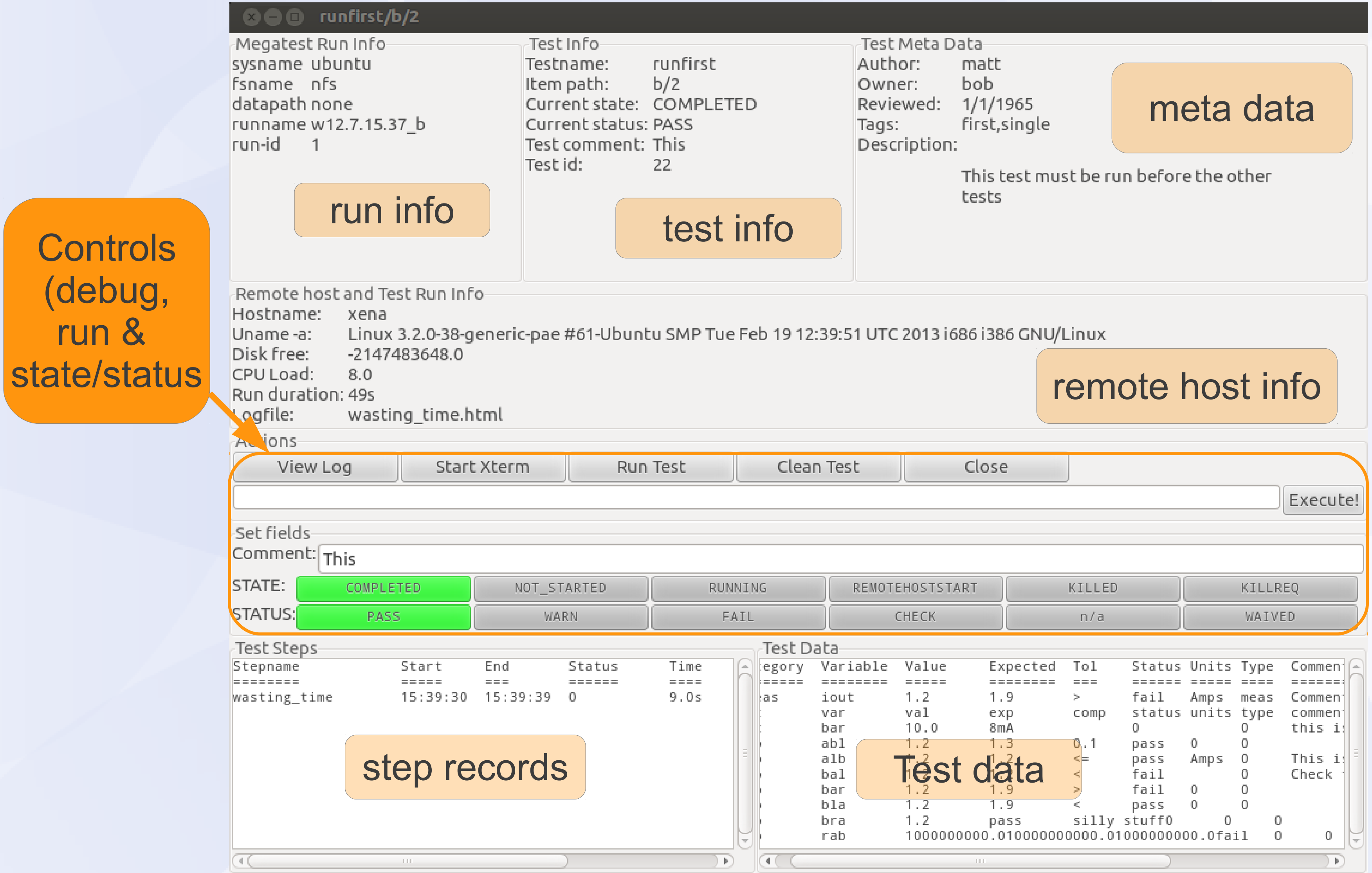
%

hide

□ PASS□ FAIL□ WARN□ CHECK□ WAIVED□ STUCK/DEAD□ n/a

□ RUNNING□ COMPLETED□ INCOMPLETE□ LAUNCHED□ NOT_STARTED□ KILLED□ DELETED

test control panel



runfirst/b/2

Megatest Run Info

```
sysname  ubuntu
fsname   nfs
datapath none
runname  w12.7.15.37_b
run-id   1
```

Test Info

```
Testname:       runfirst
Item path:      b/2
Current state:  COMPLETED
Current status: PASS
Test comment:   This
Test id:        22
```

Test Meta Data

```
Author:      matt
Owner:       bob
Reviewed:    1/1/1965
Tags:        first,single
Description:

             This test must be run before the other
             tests
```

Remote host and Test Run Info

```
Hostname:     xena
Uname -a:     Linux 3.2.0-38-generic-pae #61-Ubuntu SMP Tue Feb 19 12:39:51 UTC 2013 i686 i386 GNU/Linux
Disk free:    -2147483648.0
CPU Load:     8.0
Run duration: 49s
Logfile:      wasting_time.html
```

Actions

Comment: This

STATE: COMPLETED

STATUS: PASS

Test Steps

```
Stepname             Start     End       Status    Time
========             =====     ===       ======    ====
wasting_time         15:39:30  15:39:39  0         9.0s
```

Test Data

```
egory   Variable   Value      Expected   Tol      Status Units Type  Commen
=====   ========   =====      ========   ===      ====== ===== ====  ======
as      iout       1.2        1.9        >        fail   Amps  meas  Commen
        var        val        exp        comp     status units type  commen
        bar        10.0       8mA                 0            0     this i
        abl        1.2        1.3        0.1      pass   0     0
        alb        1.2        1.2        <=       pass   Amps  0     This i
        bal        1.2        1.2        <        fail         0     Check
        bar        1.2        1.9        >        fail   0     0
        bla        1.2        1.9        <        pass   0     0
        bra        1.2        pass       silly stuff0        0       0
        rab        1000000000.01000000000.01000000000.0fail   0       0
```

Controls (debug, run & state/status

run info

test info

meta data

remote host info

step records

Test data

Run Management

- Launching runs
  - command line: “megatest -runtests %”
  - test control panel: push “run” then “execute”
- Removing runs
  - “megatest -remove-runs”
- Rolling up runs
  - “megatest -rollup”

note: all these commands require the use of additional selector parameters such as -target and :runname

Task/Test Management

- Killing jobs
  - In the gui set status to “KILLREQ” and the job will be killed.
  - Command line example:

```
megatest -set-state-status KILLREQ,FAIL -target ubuntu/nfs/none \
         :runname w10.2a -testpatt %/% :state RUNNING
```

- Changing state and status of tests
  - Use -set-state-status, see example above.
- Add “-rerun FAIL” to your launch command line to force the re-run of failed jobs

Test Selectors

- -testpatt testpatt/itempatt

  “%” is wild card

  “%” is synonymous with “%/%”

  comma separate multiple patterns (OR)

  “%/” is tests with no items

Getting information

- -list-runs pattern
  - lists runs with runname matching pattern.
- -extract-ods
  - creates an open-document spreadsheet
- Miscellaneous queries

  -list-disks

  -list-targets

  -list-db-targets

  -find-files, -find-paths

Config File Syntax

The config file syntax was designed to be:

- simple and forgiving to syntax mistakes
- easy to understand and trace where values originated
- expressive enough for complex needs.

	Example	description of the example
Sections	[setup]	Variables defined on subsequent lines will be in the “setup” section
Variables	ABC 1	Variable “ABC” will have the value “1”
[ ] directives	[include a.txt]	include file “a.txt”, see manual for all directives
#{ } text substitutions	#{shell ls $PWD}	replace the #{ … } with the output of the ls $PWD command. Note that newlines are replaced with spaces.

Config File Text Substitutions

NOTE: [ ] substitutions can be deferred by megatest and executed just before launching a test but #{ } substitutions are done as each line is read.

[include filename]	Includes filename. Ignores if filename does not exist
[system command]	replaced with output from command
#{shell command}	replaced with output from command
#{system command}	replaced with the exit code of command
#{scheme schemecode}	replaced with the result of schemecode
#{getenv VAR}	replaced with the value of environment variable VAR
#{get section var}	replaced with the value of var from section

Creating a Megatest Area

- Required Config files
  - megatest.config
  - runconfigs.config
- Tests
  - testconfig
  - How to write a test

Setup Megatest Area (configs)

- Config files
  - megatest.config
    - Target A/B/C ...
      - One or more “keys” (the “A”, “B” and “C”)
      - Choose carefully! They cannot be changed after your megatest.db is created
    - links area (the link tree to all your tests)
    - runs disk (can add more over time)
      - Lowest space used first
      - Link tree symlinks point into run areas
  - runconfigs.config
    - can be empty initially

Required Config Files

megatest.config

```
[fields]
PLATFORM TEXT
OS       TEXT

[setup]
# Adjust max_concurrent_jobs to limit parallel jobs
max_concurrent_jobs 50

# This is your link path, best to set it and then not change it
linktree #{getenv PWD}/linktree

# Job tools control how your jobs are launched
[jobtools]
useshell yes
launcher nbfind

# You can override environment variables for all your tests here
[env-override]
EXAMPLE_VAR example value

# As you run more tests you may need to add additional disks
# the names are arbitrary but must be unique
[disks]
Disk0 #{getenv PWD}/runs
```

runconfigs.config

```
[default]
ALLTESTS see this variable

# Your variables here are grouped by targets [SYSTEM/RELEASE]
[SYSTEM_val/RELEASE_val]
ANOTHERVAR only defined if target is SYSTEM_val/RELEASE_val
```

Setup Megatest Area (tests)

- Tests
  - tests/<yourfirsttest>/testconfig
- Can use the helper “wizards”

```
megatest -gen-megatest-area
megatest -gen-megatest-test
```

Example testconfig

testconfig

```
# Add additional steps here. Format is "stepname script"
[ezsteps]
step1 step1.sh
step2 step2.sh

# Test requirements are specified here
[requirements]
waiton setup
priority 0

# Iteration for your tests are controlled by the items section
[items]
COMPONENT parser datastore transport analyzer

# test_meta is a section for storing additional data
# on your test
[test_meta]
author matt
owner  matt
description An example test
tags tagone,tagtwo
reviewed never
```

Writing a Test “checkspace”

- Write a test that checks for available space
  - tests can “waiton” this test before running.
- Our test will use this simple script, checkspace.sh:

```
#!/bin/bash -e
freespace=`df -k $DIRECTORY | grep $DIRECTORY | awk '{print $4}'`
if [[ $freespace -lt $REQUIRED ]];then
  echo "ERROR: insufficient space on $DIRECTORY"
  exit 1
else
  echo "There is adequate space on $DIRECTORY"
fi
```

Note: Files for this example can be found in “example” dir in Megatest distribution

Writing a Test “checkspace”

- Commands to create test “checkspace”
  - mkdir -p linktree runs tests/checkspace
  - cd tests/checkspace
  - vi checkspace.sh
  - chmod a+x checkspace.sh
  - vi testconfig

```
# Add steps here. Format is "stepname script"
[ezsteps]
checkspace checkspace.sh

# Iteration for your tests are controlled by the items section
[itemstable]
DIRECTORY     /tmp    /opt
REQUIRED   1000000 100000
```

Writing a test “checkspace”

- Write a logpro file to analyze your results

```
(expect:error in    "LogFileBody" = 0 "Any error"         #/err/i)
(expect:required in "LogFileBody" = 1 "Sucess signature" #/adequate space/)
```

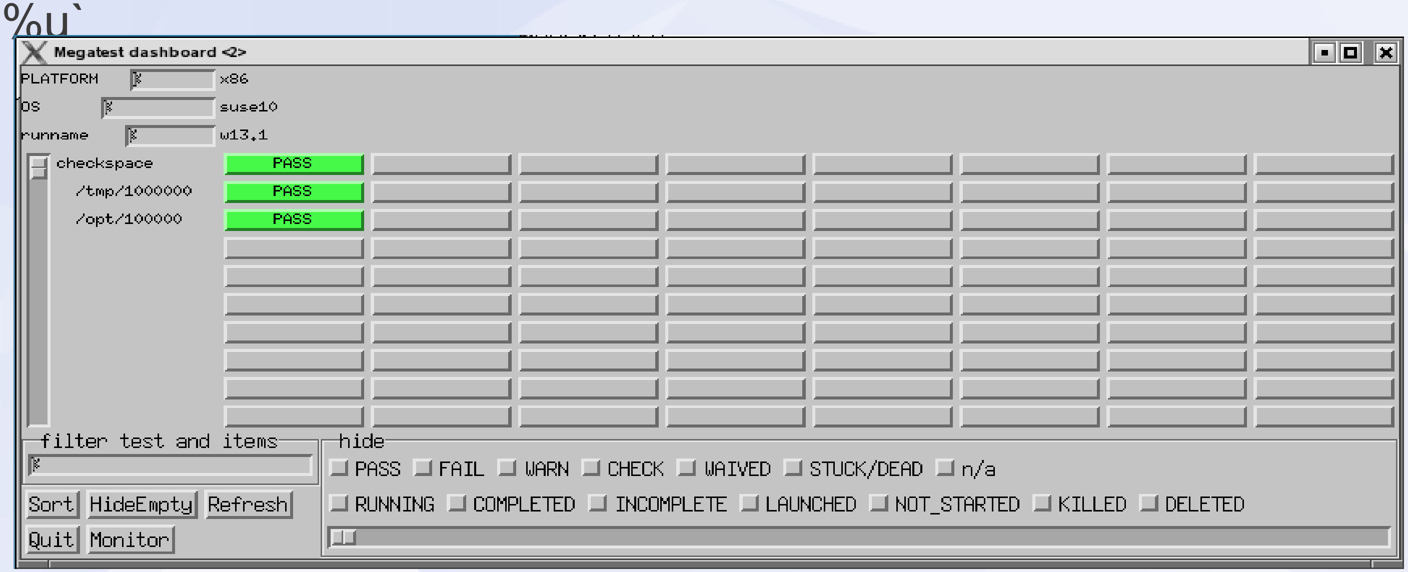
```
.
|-- megatest.config
|-- megatest.db
|-- monitor.db
|-- runconfigs.config
`-- tests
    `-- checkspace
        |--
checkspace.logpro
        |-- checkspace.sh
        `-- testconfig
```

Runing the “checkspace” Test

Run your test

From the directory where “megatest.config” exists run these commands:

```
dashboard &
megatest -runtests % -target x86/suse10 :runname w`date +%V.%u`
```



Megatest dashboard <2>

PLATFORM x86

OS suse10

runname w13.1

```
checkspace      PASS
  /tmp/1000000  PASS
  /opt/100000   PASS
```

filter test and items
hide
□ PASS □ FAIL □ WARN □ CHECK □ WAIVED □ STUCK/DEAD □ n/a
□ RUNNING □ COMPLETED □ INCOMPLETE □ LAUNCHED □ NOT_STARTED □ KILLED □ DELETED
Sort
HideEmpty
Refresh
Quit
Monitor

The “checkspace” Test Directories

```
|-- linktree
|   `-- x86
|       `-- suse10
|           `-- w13.1
|               `-- checkspace
|                   |-- opt
|                   |   `-- 100000 -> /nfs/ch/disks/ch_unienv_disk005/qa_mrwellan/interim/src/megatest/example/runs/x86/suse10/w13.1/checkspace//opt/100000
|                   |-- testdat.db
|                   `-- tmp
|                       `-- 1000000 -> /nfs/ch/disks/ch_unienv_disk005/qa_mrwellan/interim/src/megatest/example/runs/x86/suse10/w13.1/checkspace//tmp/1000000
|-- runs
|   `-- x86
|       `-- suse10
|           `-- w13.1
|               `-- checkspace
|                   |-- opt
|                   |   `-- 100000
|                   |       |-- NBFAKE-2013WW13.1_09:57:48
|                   |       |-- checkspace.html
|                   |       |-- checkspace.log
|                   |       |-- checkspace.logpro
|                   |       |-- checkspace.sh
|                   |       |-- megatest.csh
|                   |       |-- megatest.sh
|                   |       |-- mt_launch.log
|                   |       |-- testconfig
|                   |       `-- testdat.db
|                   `-- tmp
|                       `-- 1000000
|                           |-- NBFAKE-2013WW13.1_09:57:49
|                           |-- checkspace.html
|                           |-- checkspace.log
|                           |-- checkspace.logpro
|                           |-- checkspace.sh
|                           |-- megatest.csh
|                           |-- megatest.sh
|                           |-- mt_launch.log
|                           |-- testconfig
|                           `-- testdat.db
```

Setup for Run “Flavors”

- runconfigs.config

```
[default]
VARS here are inherited by all runs

[some/target]
VARS here inherited in some/target runs
```

- NB// the last specified definition overrides prior definitions.

Setup Tests/Tasks

- A test or task is a set of scripts and data designed to do something or test something.
- Create in tests directory
- Test name limitations
    - No spaces or special characters
    - [a-zA-Z0-9_] and “-” are ok.

The testconfig file [setup]

- [setup]
  runscript scriptname.sh
  - The script must exist in the testconfig directory and be executable
  - Output from the script is NOT captured by Megatest directly
  - The script can be an executable or written in any scripting language

The testconfig file [ezsteps]

- [ezsteps]
  step1 script1.sh
  - The script “script1.sh” will be executed and its output redirected to the file step1.log.
  - If a logpro file step1.logpro exists it will be used to process the logfile step1name.log and generate the PASS/FAIL/WARN status.

The testconfig file [items]

```
[items]
VAR1 value11 value12 value13 …
VAR2 value21 value22 value23 ...
```

  - This will iterate this test with all possible combinations of VAR1 and VAR2 values.

- Results:
  - value11/value21, value11/value22, value11/value23, value12/value21, value12/value22, value12/value23 ...

The testconfig file [itemstable]

```
[itemstable]
VAR1    value11    value12  …
VAR2    value21    value22  …
```

  - This will iterate over the test with only aligned value combinations.

- Result:
  - value11/value21, value12/value22 …

NOTE: You can combine items and itemstable but they work independently and the result may not be what you expect.

The testconfig file [requirements]

[requirements]

waiton <testname … >

- this test will not be launched until the listed tests are COMPLETED and PASS, WAIVE or SKIP.

jobgroup <groupname>

- this test will be added to the named job group and the relevant max concurrent jobs will apply

mode toplevel

- this test will proceed once all it waiton tests are completed with any status.

The testconfig file[test_meta]

- author matt
- owner bob
- description The description can run to multiple lines but subsequent lines must be indented with spaces.
- tags first,single
- reviewed 09/10/2011, by Matt

Megatest Calls in Tests

- -step stepname
  - mark the start or end of a step
- -test-status

  set the state and status of a test
- -setlog logfname

  set the path/filename to the final log relative to the test directory.
- -set-toplog logfname

  set the log for a series of iterated tests

Other Megatest calls

- -summarize-items

  for an itemized test create a summary html (usually called automatically)

- -m comment

  insert a comment for this test, can be used with any of the above calls but only one comment is stored per test

- -test-files or -test-paths

  Use the database to search for files or paths in the test run area

Example Megatest in-test calls

- -step

```
$MT_MEGATEST -step step1 :state start :status
running -setlog step1.html
```

- -test-status

(Mark a test as completed and trigger a rollup to the parent test of overall status)

```
$MT_MEGATEST -test-status :state COMPLETED :status
AUTO
```

- -test-path

```
export EZFAILPATH2=`$MT_MEGATEST -test-paths -target
  $MT_TARGET :runname $MT_RUNNAME -testpatt
  runfirst/a%`
```

Environment Variables

MT_TARGET	Contains the target for this run
MT_RUNNAME	The run name
MT_MEGATEST	Full path to megatest executable
MT_TEST_RUN_DIR	The area where the test itself runs
MT_TEST_NAME	The name of the current test
MT_ITEM_INFO	Data on the iteration
MT_RUN_AREA_HOME	The base area for this regression
MT_CMDINFO	Used internally by megatest
MT_DEBUG_MODE	Used to propogate debug mode to underlying megatest calls.

Additional Features

- Run locking
  - Prevents removing or adding tests to a run

    -lock

    -unlock

Logpro

- Logpro syntax

  Logpro uses scheme calls directly and the full power of scheme is available. However 99% of logpro rule files will not need anything other than the base logpro rules.

- Documentation at: http://www.kiatoa.com/fossils/logpro

Rule	Example	Purpose
expect:error	(expect:error in “Logf” = 0 “Err desc” #/err1/i)	Flags errors matching the pattern err1
expect:ignore	(expect:ignore in “Logf” < 10 “Err desc” #/err2/i)	Ignore errors matching the pattern err2
expect:warning	(expect:warning in “Logf” = 0 “Desc” #/warn1/i)	Lines matching pattern warn1 flagged as warning
expect:required	(expect:required in “Logf” = 1 “Desc” #/reqrd/i)	Line matching pattern reqrd must exit in log file
expect:waive	(expect:waive in “Logf” = 0 “Err desc” #/err3/i)	Waive error matching pattern err3
expect:value	(expect:value in “Logf” 10 1 “Err desc” #/(\d+)/i)	The number matched must be 10 +/- 1
trigger	(trigger “start” #/Start logfile/)	Set trigger “**start**” on line with “Start logfile” string.
section	(section “Logf” “start” “end”)	Section **Logf** starts at trigger **start**, ends at **end**
hook:add	(hook:add “err1” “err1.pl #{msg}”)	On err1 call the err1.pl script with msg as param

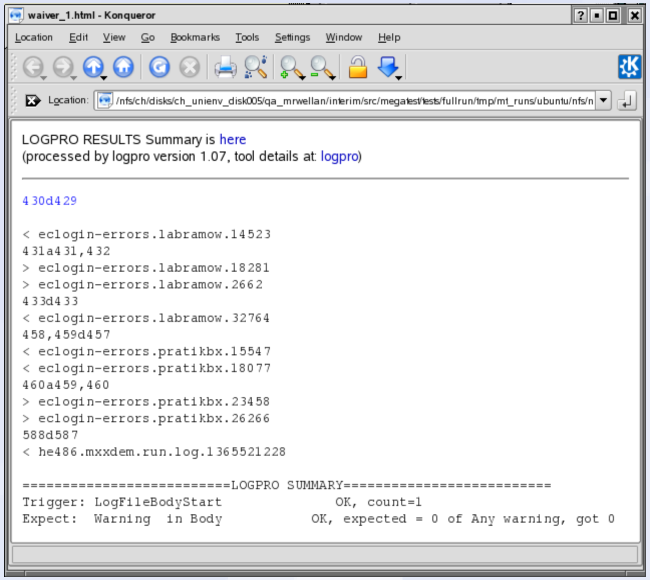
Advance Logpro Usage

- Data collection
  - Capturing with logpro
  - Rolling up with Megatest

Waiver Propagation

This test failed and was manually set to WAIVED in the next run

This test uses diff and logpro to determine if ok to propagate WAIVED


waiver_1.html - Konqueror
Location Edit View Go Bookmarks Tools Settings Window Help
Location: /nfs/ch/disks/ch_unienv_disk005/qa_mrwellan/interim/src/megatest/tests/fullrun/tmp/mt_runs/ubuntu/nfs/n

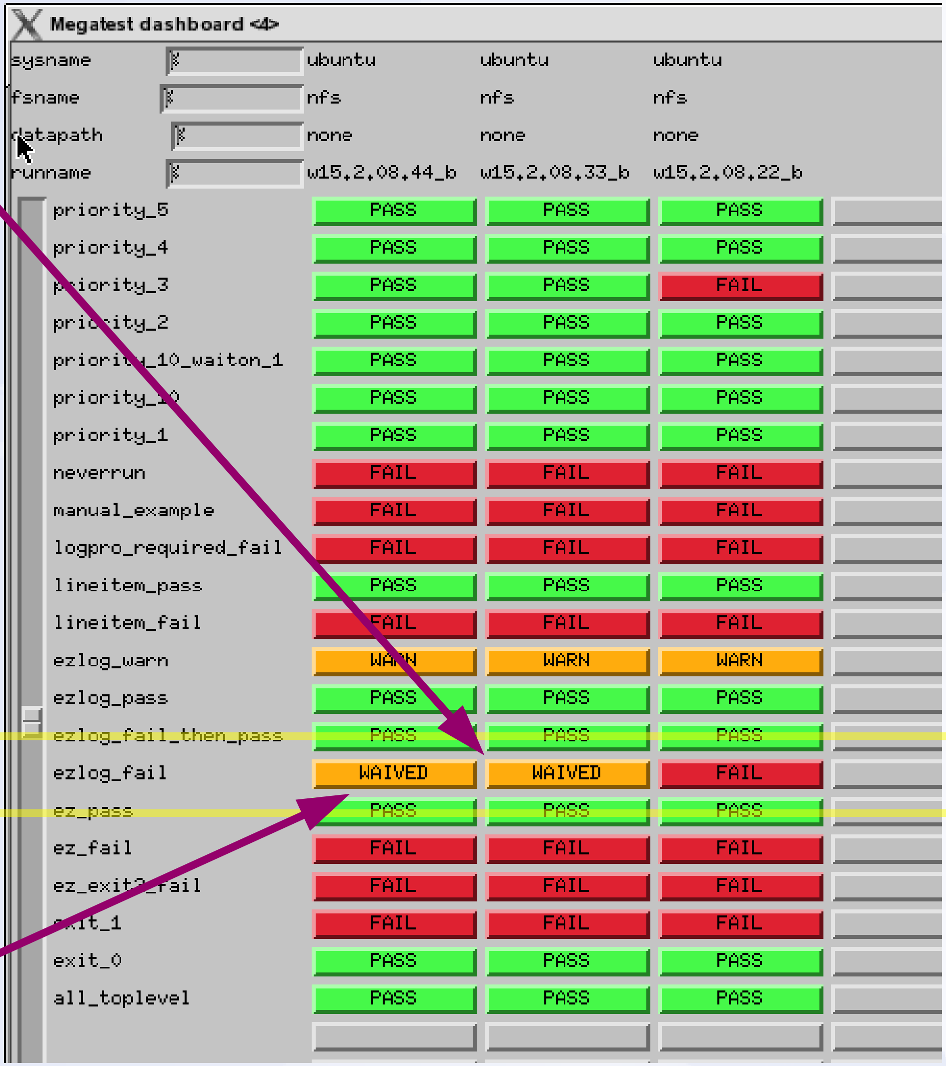
LOGPRO RESULTS Summary is here
(processed by logpro version 1.07, tool details at: logpro)

```
430d429

< eclogin-errors.labramow.14523
431a431,432
> eclogin-errors.labramow.18281
> eclogin-errors.labramow.2662
433d433
< eclogin-errors.labramow.32764
458,459d457
< eclogin-errors.pratikbx.15547
< eclogin-errors.pratikbx.18077
460a459,460
> eclogin-errors.pratikbx.23458
> eclogin-errors.pratikbx.26266
588d587
< he486.mxxdem.run.log.1365521228

=========================LOGPRO SUMMARY=========================
Trigger: LogFileBodyStart              OK, count=1
Expect:  Warning  in Body           OK, expected = 0 of Any warning, got 0
```

The WAIVED status was propagated because the criteria set in testconfig were all met



sysname	ubuntu	ubuntu	ubuntu
fsname	nfs	nfs	nfs
datapath	none	none	none
runname	w15.2.08.44_b	w15.2.08.33_b	w15.2.08.22_b
priority_5	PASS	PASS	PASS
priority_4	PASS	PASS	PASS
priority_3	PASS	PASS	FAIL
priority_2	PASS	PASS	PASS
priority_10_waiton_1	PASS	PASS	PASS
priority_10	PASS	PASS	PASS
priority_1	PASS	PASS	PASS
neverrun	FAIL	FAIL	FAIL
manual_example	FAIL	FAIL	FAIL
logpro_required_fail	FAIL	FAIL	FAIL
lineitem_pass	PASS	PASS	PASS
lineitem_fail	FAIL	FAIL	FAIL
ezlog_warn	WARN	WARN	WARN
ezlog_pass	PASS	PASS	PASS
ezlog_fail_then_pass	PASS	PASS	PASS
ezlog_fail	WAIVED	WAIVED	FAIL
ez_pass	PASS	PASS	PASS
ez_fail	FAIL	FAIL	FAIL
ez_exit2_fail	FAIL	FAIL	FAIL
exit_1	FAIL	FAIL	FAIL
exit_0	PASS	PASS	PASS
all_toplevel	PASS	PASS	PASS

Waiver Propagation

waiver name

waiver rule type

file to apply rule

```
# logpro_file  input_glob
# matching file(s) will be diff'd with previous run and logpro applied
# if PASS or WARN result from logpro then WAIVER state is set
#
[waivers]
waiver_1 logpro lookittmp.log

[waiver_rules]

# This builtin rule is the default if there is no <waivername>.logpro file
# diff   diff %file1% %file2%

# This builtin rule is applied if a <waivername>.logpro file exists
# logpro diff %file1% %file2% | logpro %waivername%.logpro %waivername%.html
```

example rules

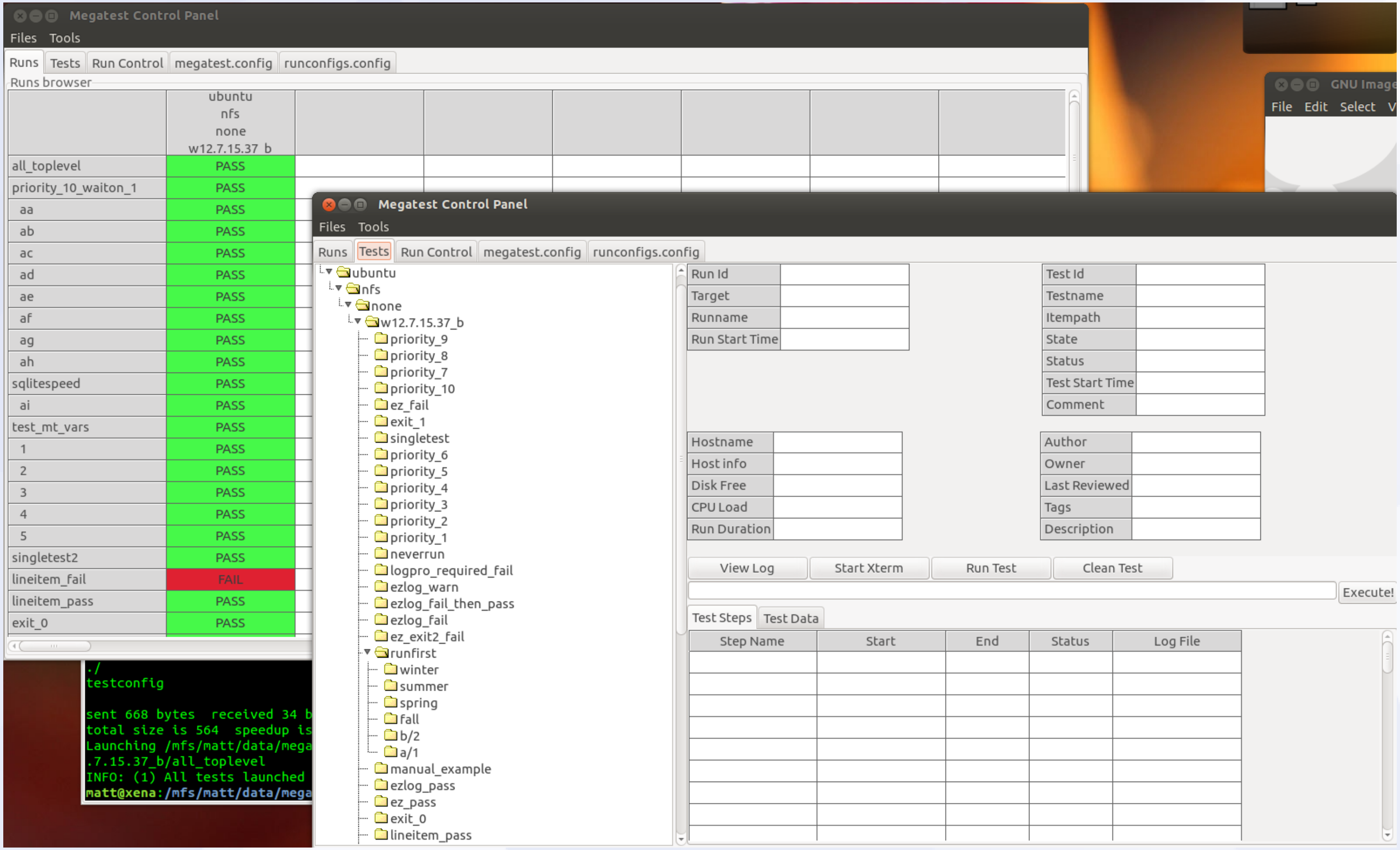
Direct Access to Megatest Functions

- -repl
  - This will start a read-eval-print loop allowing you to directly call Megatest calls.
- -load test.scm
  - This will load the scheme source code and exectute it in the Megatest context.

New Features in v1.55

- Task/Test search path
  - organize your tests in different directories
  - reuse tests from other flows
- Automatic SKIP handling
  - Crontab friendly runs (can overlap)
- “itemmatch” mode
  - iterated tests block only on previous same-named iteration

Future Megatest Development



	ubuntu nfs none w12.7.15.37_b						
all_toplevel	PASS						
priority_10_waiton_1	PASS						
aa	PASS						
ab	PASS						
ac	PASS						
ad	PASS						
ae	PASS						
af	PASS						
ag	PASS						
ah	PASS						
sqlitespeed	PASS						
ai	PASS						
test_mt_vars	PASS						
1	PASS						
2	PASS						
3	PASS						
4	PASS						
5	PASS						
singletest2	PASS						
lineitem_fail	FAIL						
lineitem_pass	PASS						
exit_0	PASS						

```
./
testconfig

sent 668 bytes  received 34 b
total size is 564  speedup is
Launching /mfs/matt/data/mega
.7.15.37_b/all_toplevel
INFO: (1) All tests launched
matt@xena:/mfs/matt/data/mega
```

```
ubuntu
  nfs
    none
      w12.7.15.37_b
        priority_9
        priority_8
        priority_7
        priority_10
        ez_fail
        exit_1
        singletest
        priority_6
        priority_5
        priority_4
        priority_3
        priority_2
        priority_1
        neverrun
        logpro_required_fail
        ezlog_warn
        ezlog_fail_then_pass
        ezlog_fail
        ez_exit2_fail
        runfirst
          winter
          summer
          spring
          fall
          b/2
          a/1
        manual_example
        ezlog_pass
        ez_pass
        exit_0
        lineitem_pass
```

Run Id	
Target	
Runname	
Run Start Time	

Test Id	
Testname	
Itempath	
State	
Status	
Test Start Time	
Comment	

Hostname	
Host info	
Disk Free	
CPU Load	
Run Duration	

Author	
Owner	
Last Reviewed	
Tags	
Description	

Step Name	Start	End	Status	Log File

Advanced Topics