

# Megatest/Logpro Training

Using Megatest and Logpro for creating flows and automation for software, EDA, or Unix infrastructure at the unit, functional, and regression levels.

Matt Welland, 2016

# Training Overview

- Background
- Getting started
  - Dashboard/command line (existing flow)
    - Running tests and managing runs
  - Creating a flow
    - configs: megatest, runconfig
    - tests/tasks: testconfig, logpro
  - Getting information about runs and tests
- Preview of advanced Megatest topics
- Future Megatest development

# What does Megatest do?

- Run tests or tasks with
  - one or many steps
  - dynamic test dependency calculation
  - on multiple hosts
  - multi-level iteration
- Report, record and roll up state, status and data
  - state: RUNNING, COMPLETED
  - status: PASS, FAIL, WARN, CHECK
  - data: slew rate, count of failed assertions etc.
- Organize “runs” by project specific variables

# Megatest Design Philosophy

## Factors for Sustainable Automation

|                       |  |
|-----------------------|--|
| <b>S</b> elf-checking | write directed or self-checking tests (avoid delta based tests)  |
| <b>T</b> raceable     | environment variables, host OS, etc. captured and recorded.      |
| <b>I</b> mmutable     | once run do not modify, reuse or overwrite tests.                |
| <b>R</b> epeatable    | this test result can be recreated in the future                  |
| <b>R</b> elocatable   | the test area can be checked out and the tests run anywhere      |
| <b>E</b> ncapsulated  | test run area is self-contained with all inputs and outputs kept |
| <b>D</b> eployable    | anyone on the team, at any site, at any time can run the tests   |

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.

# Dashboard/Test Control Panel

- dashboard
  - browse runs
  - filtering
    - target
    - runname
    - test pattern
    - state/status
  - launch runs
- test control panel
  - xterm
  - view log
  - cleanrunexecute

The screenshot shows the Megatest dashboard interface. The main window displays a grid of test results for various test patterns across different releases and iterations. The test patterns listed include: toprun, testpatt\_envvar, testpatt, test2, runconfig-tests, rollup, rerunclean, listruns-tests, itemwait, itemmap, fullrun, envvars, dependencies, and chained-waiton. The results are color-coded: green for PASS, red for FAIL, and orange for KILLED.

A detailed view for the 'listruns-tests' test is open, showing the following information:

- Megatest Run Info:** release v1.62, iteration 7172, runname ww41.5a, run-id 69, run-date 2016ww41.5 22:59
- Test Info:** Testname: listruns-tests, Item path: listruns-tests, Current state: COMPLETED, Current status: PASS, Test comment: Test id: 2070012, Test date: ww41.5 23:06
- Test Meta Data:** Author: tyardeni, Owner: tyardeni, Reviewed: tyardeni, Tags: rerunclean, Description: Test preclean/rerun behavior
- Remote host and Test Run Info:** Hostname: xena, Disk free: 70048920, CPU Load: 1.86, Run duration: 1m 8s, Logfile: chk-filt.html, Top process id: 23010, Uname -a: Linux 4.4.0-38-generic #57-Ubuntu SMP Tue Sep 6 15:42:33 UTC 2016 x86\_64 x86\_64 GNU/Linux
- Actions:** View Log, Start Xterm, Run Test, Clean Test, CleanRunExecute, Kill All Jobs, Archive Test, Close, Execute!
- Set fields:** Comment:
- STATE:** COMPLETED, NOT\_STARTED, RUNNING, REMOTEHOSTSTART, LAUNCHED, KILLED, KILLREQ, STUCK, ARCHIVED
- STATUS:** PASS, WARN, FAIL, CHECK, n/a, WAIVED, SKIP, DELETED, STUCK/DEAD, ABORT
- Steps Test Data:**

| Step Name | Start    | End      | Status | Duration | Log File      | Comment |
|-----------|----------|----------|--------|----------|---------------|---------|
| copy      | 23:06:11 | 23:06:13 | 0      | 2s       |               |         |
| clean     | 23:06:13 | 23:06:17 | 0      | 4s       | clean.html    |         |
| run-ex    | 23:06:17 | 23:06:51 | 0      | 34s      |               |         |
| run-filt  | 23:06:51 | 23:07:13 | 0      | 22s      |               |         |
| chk-filt  | 23:07:13 | 23:07:17 | 0      | 4s       | chk-filt.html |         |

# Terminology

|               |   |
|---------------|---|
| target        | one or more “keys” separated by “/”, used to organize runs hierarchically; examples include platform, release, 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. A target is a context. |
| run name      | unique name (within a single target grouping) for a run, a common idiom is to use week and day numbers: e.g w41.6 (use unix command: 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  |

# Architecture

- config files, static state, human input
  - megatest.config
  - runconfigs.config
  - tests/<testname>/testconfig
- SQL database, dynamic state
  - megatest.db
- Tools
  - megatest (command line), dashboard (gui), and logpro (log file analysis via rules), refdb (text based data base)

# Getting Help

- Command line help:  
    `megatest -h`  
    or try: `viewscreen "megatest -h |& less"`
- The user manual:  
    `megatest -manual`



# dashboard

runs filter

a "run"

a "test"

a "test item"

tests filter

| ORG            | test   | test    |  |  |  |  |  |  |  |
|----------------|--------|---------|--|--|--|--|--|--|--|
| RUNTYPE        | normal | normal  |  |  |  |  |  |  |  |
| runname        | 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  
%  
Sort HideEmpty Refresh  
Quit Monitor

hide  
 PASS  FAIL  WARN  CHECK  WAIVED  STUCK/DEAD  n/a  
 RUNNING  COMPLETED  INCOMPLETE  LAUNCHED  NOT\_STARTED  KILLED  DELETED

Do live demo of dashboard here.

# test control panel

Controls  
(debug,  
run &  
state/status)

The screenshot shows a web-based test control panel for a test named 'runfirst' with item path 'b/2'. The interface is divided into several sections:

- Megatest Run Info:** sysname ubuntu, fsname nfs, datapath none, runname w12.7.15.37\_b, run-id 1. Callout: **run info**
- Test Info:** Testname: runfirst, Item path: b/2, Current state: COMPLETED, Current status: PASS, Test comment: This, Test id: 22. Callout: **test info**
- Test Meta Data:** Author: matt, Owner: bob, Reviewed: 1/1/1965, Tags: first,single, Description: This test must be run before the other tests. Callout: **meta data**
- 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. Callout: **remote host info**
- Actions:** View Log, Start Xterm, Run Test, Clean Test, Close, Execute!
- Set fields:** Comment: This
- STATE:** COMPLETED (highlighted), NOT\_STARTED, RUNNING, REMOTEHOSTSTART, KILLED, KILLREQ
- STATUS:** PASS (highlighted), WARN, FAIL, CHECK, n/a, WAIVED
- Test Steps:** Table with columns Stepname, Start, End, Status, Time. Row: wasting\_time, 15:39:30, 15:39:39, 0, 9.0s. Callout: **step records**
- Test Data:** Table with columns Category, Variable, Value, Expected, Tol, Status, Units, Type, Comment. Rows include iout, val, bar, abl, alb, bal, bra, rab. Callout: **Test data**

Do live demo of test control panel here.

# Run Management

- Launching runs
  - command line: “megatest -run ...”
  - test control panel: push “run” then “execute”
- Removing runs
  - command line: “megatest -remove-runs ...”
- Archiving runs
  - command line: “megatest -archive ...”

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,LAUNCHED
```

- 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 testpattern/itempattern
  - wild card is “%”
    - % synonymous with %/%
    - %/ toplevel tests (no items)
- comma separate multiple patterns (OR)
  - %,%/a/b All toplevel + any items matching a/a

# 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

# 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 evaluating (schemecode)   |
| #{getenv VAR}          | replaced with the value of environment variable VAR   |
| #{get section var}     | replaced with the value of var from section           |
| #{rget var}            | use runconfig rules to get a variable                 |



# Creating a Megatest Area

- Required Config files
  - megatest.config
  - runconfigs.config
- Tests
  - testconfig
- Can use the helper “wizards”
  - megatest -create-megatest-area
  - megatest -create-test <testname>

(demo of -create-megatest-area and -create-test)

# Setup Megatest Area (Review)

- 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 usage disk 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 MT_RUN_AREA_HOME}/linktree

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

# 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 MT_RUN_AREA_HOME}/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
```

# 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

[logpro]
step1 ;;
    (expect:error in "LogFileBody" = 0 "No errors" #/err/i)

# 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
```

# Megatest Information

- Main development site

<http://www.kiatoa.com/fossils/megatest>

<http://www.kiatoa.com/fossils/logpro>

- Mirror

<http://chiselapp.com/user/kiatoa/repository/megatest>

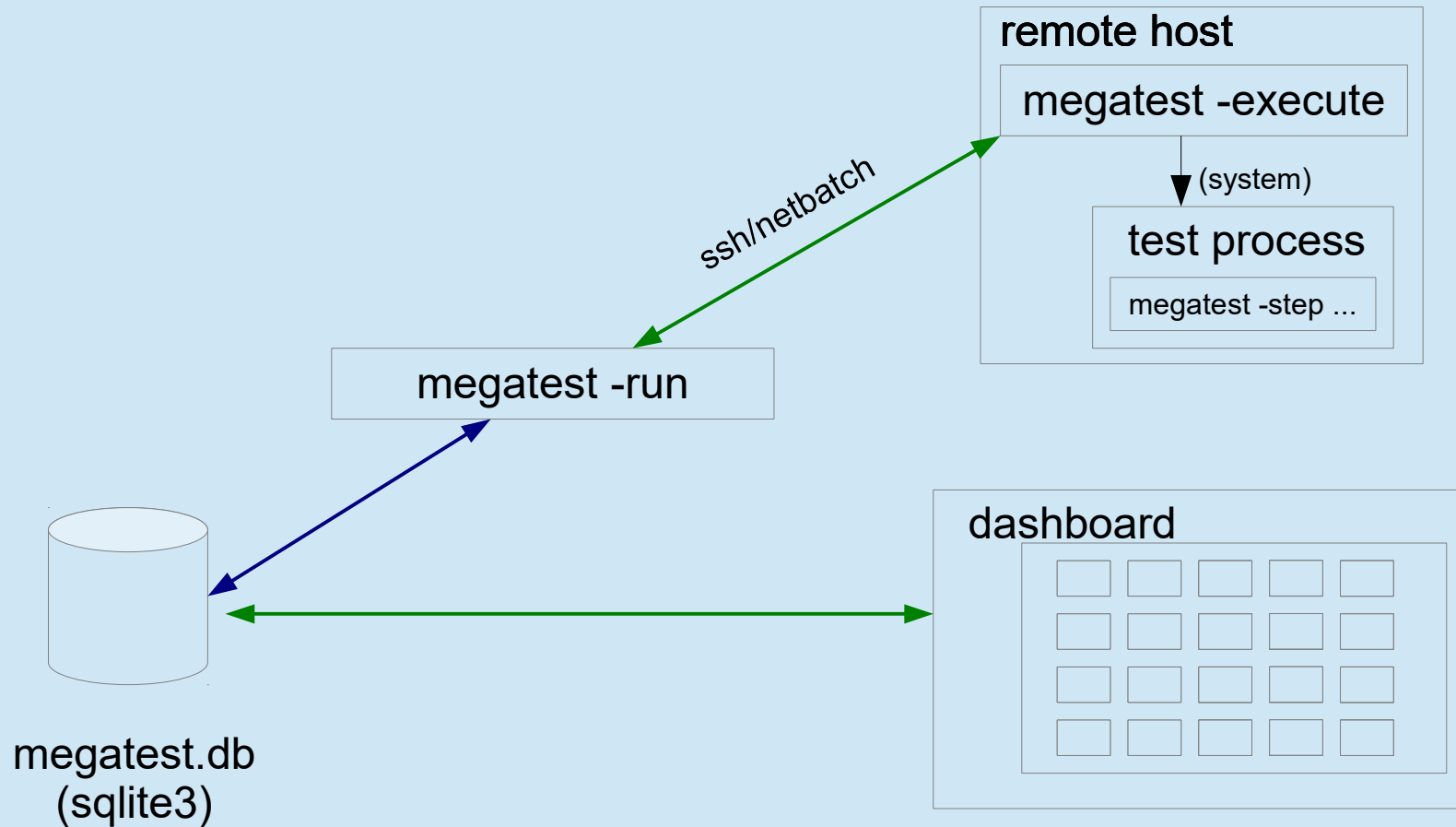
<http://chiselapp.com/user/kiatoa/repository/logpro>

- SourceForge Page

<http://sourceforge.com/projects/megatest>

# Backup

# How it Works



# A Day in The Life ...

test control panel  
(in background)

run progress seen in xterm

The image shows a terminal window and a Megatest dashboard window. The terminal window displays the following output:

```
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

/mfs/matt/data/sysmaint/runs/xena/normal/ww12/ch
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
```

The Megatest dashboard window shows the following test results:

| Test Name         | Status  |
|-------------------|---------|
| 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    |

The dashboard also includes a filter test and items field, a hide checkbox, and a list of test statuses: PASS, FAIL, WARN, CHECK, WAIVED, STUCK/DEAD, n/a, RUNNING, COMPLETED, INCOMPLETE, LAUNCHED, NOT\_STARTED, KILLED, DELETED. There are also buttons for Sort, HideEmpty, Refresh, Quit, and Monitor.

logpro output

dashboard



# 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
```



# 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
- **-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. |
| MT_LINKTREE      | Full path to the link tree, use to find tests              |

# 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 exist 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 " <b>start</b> " on line with "Start logfile" string.       |
| section         | (section "Logf" "start" "end")                     | Section <b>Logf</b> starts at trigger <b>start</b> , ends at <b>end</b> |
| 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

```
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.15947  
< 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
```

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

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

# Waiver Propagation

waiver name

waiver rule type

file to apply rule

example rules

```
# 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 <waivename>.logpro file
# diff diff %file1% %file2%

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

# 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 execute 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