

TESTING OF HPC SCIENTIFIC SOFTWARE

The IDEAS Team

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Introduction

Why is testing important ?

Granularity of tests

Types of tests

Benefits of testing

3

- Promotes high-quality software that delivers correct results and improves confidence
- Increases quality and speed of development, reducing development and maintenance costs
- Maintains portability to a variety of systems and compilers
- Helps in refactoring
 - ▣ Avoid introducing new errors when adding new features
 - ▣ Avoid reintroducing old errors

How common are bugs?

4

Programs do not acquire bugs as people acquire germs, by hanging around other buggy programs. Programmers must insert them.

- Harlan Mills

- Bugs per 1000 lines of code (KLOC)
- Industry average for delivered software
 - ▣ 1-25 errors
- Microsoft Applications Division
 - ▣ 10-20 defects during in-house testing
 - ▣ 0.5 in released product

Why testing is important: the protein structures of Geoffrey Chang

5

- Some inherited code flipped two columns of data, inverting an electron-density map
- Resulted in an incorrect protein structure
- Resulted in 5 retracted publications
 - ▣ One was cited 364 times
- Many papers and grant applications conflicting with his results were rejected

Why testing is important: the 40 second flight of the Ariane 5

6

- ❑ Ariane 5: a European orbital launch vehicle meant to lift 20 tons into low Earth orbit
- ❑ Initial rocket went off course, started to disintegrate, then self-destructed less than a minute after launch
- ❑ Seven variables were at risk of leading to an Operand Error (due to conversion of floating point to integer)
 - ▣ Four were protected
- ❑ Investigation concluded insufficient test coverage as one of the causes for this accident
- ❑ Resulted in a loss of \$370,000,000.

Why testing is important: the Therac-25 accidents

7

- Therac-25: a computer-controlled radiation therapy machine
- Minimal software testing
- Race condition in the code went undetected
- Unlucky patients were struck with approximately 100 times the intended dose of radiation, ~ 15,000 rads
- Error code indicated that no dose of radiation was given, so operator instructed machine to proceed
 - ▣ Documentation gave no indication that the frequent malfunctions of the machine could place a patient at risk
 - ▣ See also: why documentation is important
- Recalled after six accidents resulting in death and serious injuries

Granularity of tests

8

- Unit tests
 - ▣ Test individual functions or classes
 - ▣ Build and run fast
 - ▣ Localize errors
 - ▣ Usually written before or during code development
 - Prevent faults from being introduced
 - ▣ Example: Can I correctly compute a dot-product?

If a unit test fails, you should know *exactly* what is broken.

Granularity of tests

9

- Integration tests
 - ▣ Test interaction of larger pieces of software
 - ▣ Do not build or run as fast as unit tests
 - ▣ Example: Does the preconditioner class work with the Krylov solver class?

Granularity of tests

10

- System-level tests

- Test the full software system at the user interaction level
- Example: Does my CFD code compute the correct solution?

Types of tests

11

□ Verification tests

- ▣ Does the code implement the intended algorithm correctly?
- ▣ Check for specific mathematical properties
- ▣ Example
 - Solving $Ax=b$ where A has 5 distinct eigenvalues
 - Does my Krylov solver converge in 5 iterations?
- ▣ Can be any granularity

Types of tests

12

□ Acceptance tests

- ▣ Assert acceptable functioning for a specific customer
 - Different from other types of tests, which don't involve customers
- ▣ Generally at the system-level
- ▣ Example: Does my linear solver achieve the correct convergence rate for a particular customer's linear system?

Types of tests

13

- Regression (no-change) tests
 - ▣ Compare current observable output to a gold standard
 - Gold standard frequently comes from previous version of software
 - ▣ Similar to verification tests
 - Must independently verify that the gold standard is correct
 - ▣ Example
 - My Krylov solver took 10 iterations last week; does it still take 10 iterations?
 - Does it achieve the same solution?
 - ▣ Bounded change tests are better for floating point computations

Types of tests

14

- Performance tests
 - ▣ Focus on the runtime and resource utilization
 - ▣ Nothing to do with correctness
 - Orthogonal to other types of tests
 - ▣ Example: It took my code 10s to solve this linear system last week; does it take longer now?

Types of tests

15

- Installation tests
 - ▣ Verify that the configure-make-install is working as expected
 - ▣ Example: Can I build and run a simple driver using my library after the library is installed?

Tpetra verification

16

- Distributed basic linear algebra subroutines
 - ▣ Sparse matrices
 - ▣ Dense matrices
- Check for correct linear algebra
- Check for correct errors
 - ▣ Does the program throw an exception if I try to multiply things with incompatible dimensions?

Belos verification

17

- Krylov solvers
- Use problems with known solutions
 - ▣ Given A and Y , generate $B=AY$
 - Ensures B is in the range of A
 - ▣ Solve $AX=B$
- Some tests use Belos matrix and vector classes
- Some tests use Epetra/Tpetra classes
- Test with and without preconditioning
 - ▣ Left and right

Anasazi verification

18

- Eigensolvers
- Use problems with known solutions
 - ▣ Generated using Matlab's sprand
 - ▣ Problems with analytic solutions
 - Discretization of the Laplace operator
- Measure the residual of the computed eigenvectors
 - ▣ $R = AX - BX\Lambda$
- Number of iterations are compared to a gold standard

Zoltan(2) verification

19

- Graph partitioning
 - ▣ Some Sandia-developed code
 - ▣ Some TPL wrappers
- Gold standard solutions
 - ▣ Labor intensive
 - ▣ Gold standard changes when algorithms change
 - ▣ Upgrades to a TPL such as ParMETIS require gold standard to be updated
- Uses metrics to determine whether the solution is correct
 - ▣ Edge cuts
 - ▣ Balance criteria

How did we motivate people to write 1500 tests?

20

- Tests protect YOU from other people from breaking your work
 - ▣ If someone else's changes break your code, they are responsible for fixing it
- Testing is cheaper and easier than debugging
- We already had some lying around anyway
 - ▣ Drivers for generating conference or paper results
 - ▣ User submitted bugs
 - ▣ Examples

How do we determine what other tests are needed?

21

□ Code coverage tools

▣ Expose parts of the code that aren't being tested

▣ gcov

- standard utility with the GNU compiler collection suite
- counts the number of times each statement is executed

▣ lcov

- a graphical front-end for gcov
- available at <http://ltp.sourceforge.net/coverage/lcov.php>

Continuous integration (CI): a master branch that always works

22

- Code changes trigger automated builds/tests on target platforms
- Builds/tests finish *in a reasonable amount of time*, providing useful feedback when it's most needed
- Immensely helpful!
- Requires some work, though:
 - ▣ A reasonably automated build system
 - ▣ An automated test system with significant test coverage
 - ▣ A set of systems on which tests will be run, and a controller

Continuous integration (CI): a master branch that always works

23

- Has existed for some time
- Adoption has been slow
 - ▣ Setting up and maintaining CI systems is difficult, labor-intensive (typically requires a dedicated staff member)
 - ▣ *You have to be doing a lot of things right to even consider CI*

Cloud-based CI is available as a service on GitHub

24

- Automated builds/tests can be triggered via pull requests
- Builds/tests can be run on cloud systems – no server in your closet. *Great use of the cloud!*
- Test results are reported on the pull request page (with links to detailed logs)
- Already being used successfully by scientific computing projects, with noticeable benefits to productivity
- Not perfect, but *far* better than not doing CI

Travis CI is a great choice for HPC

25

- ❑ Integrates easily with GitHub
- ❑ *Free* for Open Source projects
- ❑ Supports environments with C/C++/Fortran compilers (GNU, Clang, Intel[?])
- ❑ Linux, Mac platforms available
- ❑ *Relatively* simple, *reasonably* flexible configuration file
 - ▣ Documentation is sparse, but we now have working examples

Travis CI live demo

26

- <https://github.com/amklinux/morpheus>

How to use lcov

27

- Compile and link your code with --coverage flag
 - ▣ It's a good idea to disable optimization
- Run your test suite
- Collect coverage data using lcov
- Generate html output using genhtml

Slide 27

KAM3

I will not be offended if someone recommends removing the gcov/lcov tutorial.

Klinvex, Alicia Marie, 8/18/2016

A simple example

```
#include<iostream>
#include "isEven.hpp"

int main()
{
    int num = 8;

    if(isEven(num))
        std::cout << num << " is an even number.\nTEST PASSED";
    else
        std::cout << num << " is an odd number.\nTEST FAILED";

    return 0;
}
```

```
bool isEven(int x)
{
    if(x%2 == 0)
        return true;
    return false;
}
```

A simple example

29

- **Compile and link with --coverage flag**
 - ▣ `g++ --coverage evenExample.cpp -o evenExample`
 - ▣ This creates a file called `evenExample.gcno`
- **Run the test**
 - ▣ `./evenExample`
 - ▣ This creates a file called `evenExample.gcda`
- **Collect coverage data using lcov**
 - ▣ `lcov --capture --directory . --output-file evenExample.info`
 - ▣ This creates `evenExample.info`
- **Generate html output using genhtml**
 - ▣ `genhtml evenExample.info --output-directory evenHTML`
 - ▣ This generates html files in the directory `evenHTML`

A simple example

30

LCOV - code coverage report

Current view: **top level** - [/home/amklinv/IDEAS/testingTalk/examples/simpleExample](#)

Test: **evenExample.info**

Date: **2016-05-24 14:13:07**

	Hit	Total	Coverage
Lines:	9	11	81.8 %
Functions:	4	4	100.0 %

Filename	Line Coverage ↕	Functions ↕
evenExample.cpp	85.7 % 6 / 7	100.0 % 3 / 3
isEven.hpp	75.0 % 3 / 4	100.0 % 1 / 1

Generated by: [LCOV version 1.12-4-g04a3c0e](#)

This is the file we're testing

A simple example

31

LCOV - code coverage report

Current view: [top level](#) - [home/amklinv/IDEAS/testingTalk/examples/simpleExample](#) - [IsEven.hpp](#) (source / functions)

Test: [evenExample.info](#)

Date: 2016-05-24 14:13:07

	Hit	Total	Coverage
Lines:	3	4	75.0 %
Functions:	1	1	100.0 %

Line data

Source code

1	1	: bool isEven(int x)
2		: {
3	1	: if(x%2 == 0)
4	1	: return true;
5		:
6	0	: return false;
7		: }

We never tested this line of code
(which activates when x is odd)

Let's add another test

```
#include<iostream>
#include "isEven.hpp"

int main()
{
    int num = 7;

    if(isEven(num))
        std::cout << num << " is an even number.\nTEST FAILED";
    else
        std::cout << num << " is an odd number.\nTEST PASSED";

    return 0;
}
```

```
bool isEven(int x)
{
    if(x%2 == 0)
        return true;
    return false;
}
```

A simple example

33

- Compile and link with `--coverage` flag
 - ▣ `g++ --coverage oddExample.cpp -o oddExample`
 - ▣ This creates a file called `oddExample.gcno`
- Run the test
 - ▣ `./oddExample`
 - ▣ This creates a file called `oddExample.gcda`
- Collect coverage data for BOTH TESTS using `lcov`
 - ▣ `lcov --capture --directory . --output-file twoExamples.info`
 - ▣ This creates `twoExamples.info`
- Generate html output using `genhtml`
 - ▣ `genhtml twoExamples.info --output-directory totalHTML`
 - ▣ This generates html files in the directory `totalHTML`

A simple example

34

LCOV - code coverage report

Current view: [top level](#) - /home/amklinv/IDEAS/testingTalk/examples/simpleExample

Test: twoExamples.info

Date: 2016-05-24 15:17:38

	Hit	Total	Coverage
Lines:	16	18	88.9 %
Functions:	7	7	100.0 %

Filename	Line Coverage ↕			Functions ↕	
evenExample.cpp		85.7 %	6 / 7	100.0 %	3 / 3
isEven.hpp		100.0 %	4 / 4	100.0 %	1 / 1
oddExample.cpp		85.7 %	6 / 7	100.0 %	3 / 3

Generated by: [LCOV version 1.12-4-g04a3c0e](#)

This is the file we're testing

A simple example

35

LCOV - code coverage report

Current view: [top level](#) - [home/amklInv/IDEAS/testingTalk/examples/simpleExample](#) - [IsEven.hpp](#) (source / functions)

Test: [twoExamples.info](#)

Date: 2016-05-24 15:17:38

	Hit	Total	Coverage
Lines:	4	4	100.0 %
Functions:	1	1	100.0 %

Line data

Source code

1	2	:	bool isEven(int x)
2		:	{
3	2	:	if(x%2 == 0)
4	1	:	return true;
5		:	
6	1	:	return false;
7		:	}

We tested every line of this function

A real example - xSDKTrilinos

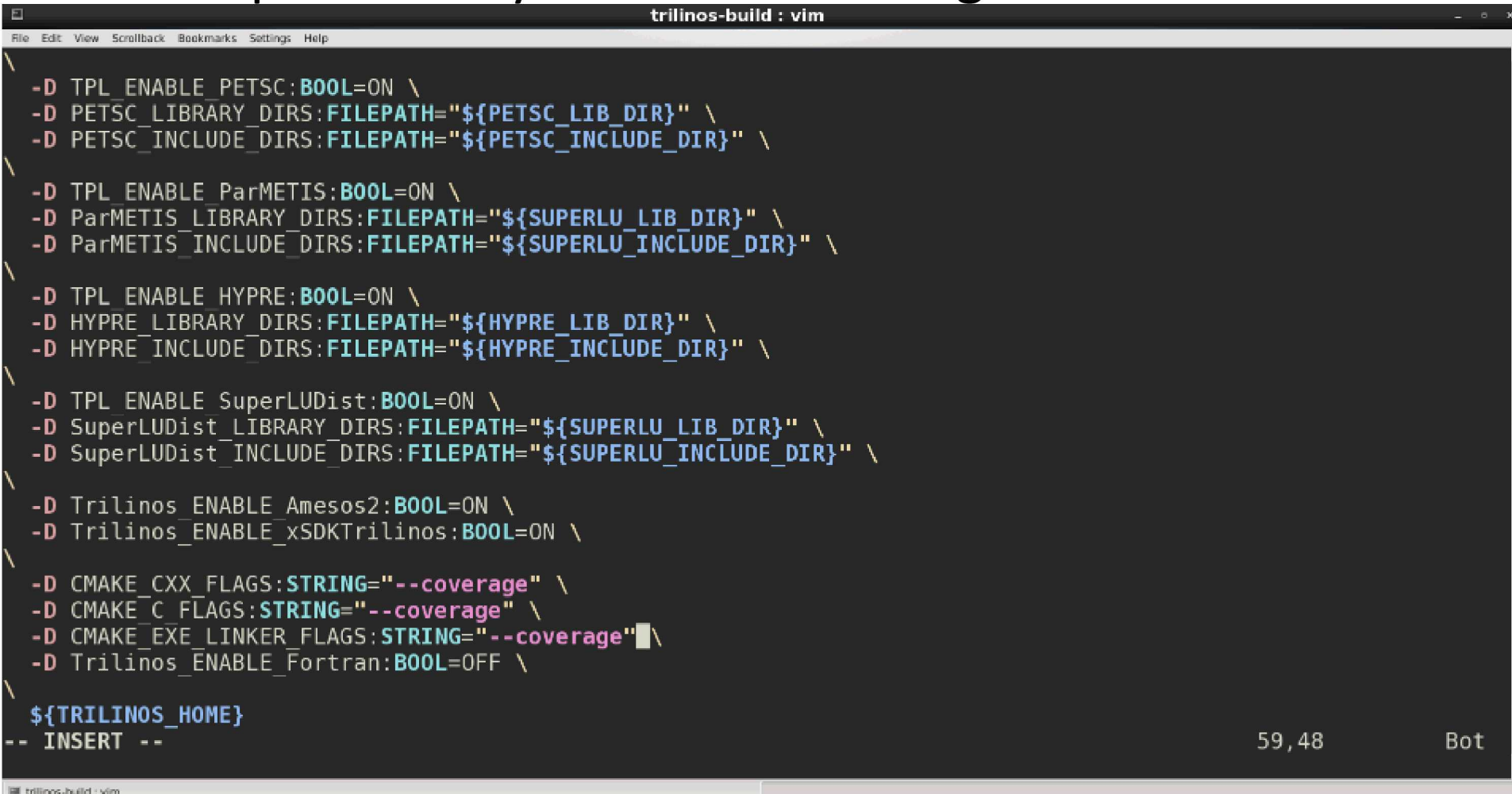
36

- ❑ Part of the Trilinos library, developed at SNL as part of the IDEAS project
- ❑ Contains the interfaces between Trilinos, PETSc, and hypre
- ❑ Available at <https://github.com/trilinos/xSDKTrilinos>
- ❑ Ten automated tests are run nightly
 - ❑ Six are actually examples that were converted into tests
- ❑ Did we leave anything out?

A real example - xSDKTrilinos

37

- Step 1: Modify our CMake configuration file to use



```
trilinos-build : vim
File Edit View Scrollback Bookmarks Settings Help

-D TPL_ENABLE_PETSC:BOOL=ON \
-D PETSC_LIBRARY_DIRS:FILEPATH="${PETSC_LIB_DIR}" \
-D PETSC_INCLUDE_DIRS:FILEPATH="${PETSC_INCLUDE_DIR}" \

-D TPL_ENABLE_ParMETIS:BOOL=ON \
-D ParMETIS_LIBRARY_DIRS:FILEPATH="${SUPERLU_LIB_DIR}" \
-D ParMETIS_INCLUDE_DIRS:FILEPATH="${SUPERLU_INCLUDE_DIR}" \

-D TPL_ENABLE_HYPRE:BOOL=ON \
-D HYPRE_LIBRARY_DIRS:FILEPATH="${HYPRE_LIB_DIR}" \
-D HYPRE_INCLUDE_DIRS:FILEPATH="${HYPRE_INCLUDE_DIR}" \

-D TPL_ENABLE_SuperLUDist:BOOL=ON \
-D SuperLUDist_LIBRARY_DIRS:FILEPATH="${SUPERLU_LIB_DIR}" \
-D SuperLUDist_INCLUDE_DIRS:FILEPATH="${SUPERLU_INCLUDE_DIR}" \

-D Trilinos_ENABLE_Amesos2:BOOL=ON \
-D Trilinos_ENABLE_xSDKTrilinos:BOOL=ON \

-D CMAKE_CXX_FLAGS:STRING="--coverage" \
-D CMAKE_C_FLAGS:STRING="--coverage" \
-D CMAKE_EXE_LINKER_FLAGS:STRING="--coverage" \
-D Trilinos_ENABLE_Fortran:BOOL=OFF \

${TRILINOS_HOME}
-- INSERT --
```

59,48 Bot

A real example - xSDKTrilinos

38

- Build Trilinos (including xSDKTrilinos)
 - ▣ `./do-configure`
 - ▣ `make -j`
- This will create a whole bunch of .gcno files
- This will also build the xSDKTrilinos tests because the configure file included
 - ▣ `-D Trilinos_ENABLE_TESTS:BOOL=ON`
 - ▣ `-D Trilinos_ENABLE_EXAMPLES:BOOL=ON`
 - ▣ `-D Trilinos_ENABLE_ALL_OPTIONAL_PACKAGES=ON`

A real example - xSDKTrilinos

39

□ Run the tests using ctest

```
trilinos-build : ctest
File Edit View Scrollback Bookmarks Settings Help
[amklinv@s995692 trilinos-build]$ ctest
Test project /home/amklinv/IDEAS/testingTalk/trilinos-build
  Start 1: Amesos2_KLU2_UnitTests_MPI_4
1/18 Test #1: Amesos2_KLU2_UnitTests_MPI_4 ..... Passed    1.46 sec
  Start 2: Amesos2_SuperLU_DIST_Solver_Test_MPI_4
2/18 Test #2: Amesos2_SuperLU_DIST_Solver_Test_MPI_4 ..... Passed    2.80 sec
  Start 3: Amesos2_SolverFactory_UnitTests_MPI_4
3/18 Test #3: Amesos2_SolverFactory_UnitTests_MPI_4 ..... Passed    1.46 sec
  Start 4: Amesos2_Tpetra_MultiVector_Adapter_UnitTests_MPI_4
4/18 Test #4: Amesos2_Tpetra_MultiVector_Adapter_UnitTests_MPI_4 ... Passed    1.36 sec
  Start 5: Amesos2_Tpetra_CrsMatrix_Adapter_UnitTests_MPI_4
5/18 Test #5: Amesos2_Tpetra_CrsMatrix_Adapter_UnitTests_MPI_4 ..... Passed    1.42 sec
  Start 6: Amesos2_Epetra_MultiVector_Adapter_UnitTests_MPI_4
6/18 Test #6: Amesos2_Epetra_MultiVector_Adapter_UnitTests_MPI_4 ... Passed    1.35 sec
  Start 7: Amesos2_Epetra_RowMatrix_Adapter_UnitTests_MPI_4
7/18 Test #7: Amesos2_Epetra_RowMatrix_Adapter_UnitTests_MPI_4 ..... Passed    1.35 sec
  Start 8: Amesos2_CrsMatrix_Adapter_Consistency_Tests_MPI_4
8/18 Test #8: Amesos2_CrsMatrix_Adapter_Consistency_Tests_MPI_4 ..... Passed    1.47 sec
  Start 9: xSDKTrilinos_PETScAIJMatrix_MPI_4
9/18 Test #9: xSDKTrilinos_PETScAIJMatrix_MPI_4 ..... Passed    1.42 sec
  Start 10: xSDKTrilinos_PETSc_Amesos2_example_MPI_4
10/18 Test #10: xSDKTrilinos_PETSc_Amesos2_example_MPI_4 ..... Passed    1.42 sec
  Start 11: xSDKTrilinos_PETSc_Anasazi_example_MPI_4
11/18 Test #11: xSDKTrilinos_PETSc_Anasazi_example_MPI_4 ..... Passed    2.71 sec
  Start 12: xSDKTrilinos_PETSc_Ifpack2_example_MPI_4
12/18 Test #12: xSDKTrilinos_PETSc_Ifpack2_example_MPI_4 ..... Passed    1.47 sec
  Start 13: xSDKTrilinos_PETSc_MueLu_example_MPI_4
```

A real example - xSDKTrilinos

40

□ All tests passed. Yay!

```
trilinos-build : ctest
File Edit View Scrollback Bookmarks Settings Help
Start 10: xSDKTrilinos_PETSc_Amesos2_example_MPI_4 ..... Passed 1.42 sec
10/18 Test #10: xSDKTrilinos_PETSc_Amesos2_example_MPI_4 ..... Passed 1.42 sec
Start 11: xSDKTrilinos_PETSc_Anasazi_example_MPI_4 ..... Passed 2.71 sec
11/18 Test #11: xSDKTrilinos_PETSc_Anasazi_example_MPI_4 ..... Passed 2.71 sec
Start 12: xSDKTrilinos_PETSc_Ifpack2_example_MPI_4 ..... Passed 1.47 sec
12/18 Test #12: xSDKTrilinos_PETSc_Ifpack2_example_MPI_4 ..... Passed 1.47 sec
Start 13: xSDKTrilinos_PETSc_MueLu_example_MPI_4 ..... Passed 2.34 sec
13/18 Test #13: xSDKTrilinos_PETSc_MueLu_example_MPI_4 ..... Passed 2.34 sec
Start 14: xSDKTrilinos_example_TpetraKSP_MPI_4 ..... Passed 1.50 sec
14/18 Test #14: xSDKTrilinos_example_TpetraKSP_MPI_4 ..... Passed 1.50 sec
Start 15: xSDKTrilinos_example_EpetraKSP_MPI_4 ..... Passed 1.37 sec
15/18 Test #15: xSDKTrilinos_example_EpetraKSP_MPI_4 ..... Passed 1.37 sec
Start 16: xSDKTrilinos_HypreTest_MPI_4 ..... Passed 1.42 sec
16/18 Test #16: xSDKTrilinos_HypreTest_MPI_4 ..... Passed 1.42 sec
Start 17: xSDKTrilinos_Hypre_Belos_example_MPI_4 ..... Passed 1.38 sec
17/18 Test #17: xSDKTrilinos_Hypre_Belos_example_MPI_4 ..... Passed 1.38 sec
Start 18: xSDKTrilinos_Hypre_Solve_example_MPI_4 ..... Passed 1.36 sec
18/18 Test #18: xSDKTrilinos_Hypre_Solve_example_MPI_4 ..... Passed 1.36 sec

100% tests passed, 0 tests failed out of 18

Label Time Summary:
Amesos2      = 12.67 sec (8 tests)
xSDKTrilinos = 16.39 sec (10 tests)

Total Test time (real) = 29.11 sec
[amklinv@s995692 trilinos-build]$
```

A real example - xSDKTrilinos

41

- Collect coverage data for the tests using lcov
 - ▣ `lcov --capture --directory . --output-file xSDKTrilinos.info`
 - ▣ This creates xSDKTrilinos.info
 - ▣ lcov processes 634 gcda files in this step, so this does take a few minutes

A real example - xSDKTrilinos

42

- Generate html output using genhtml
 - ▣ `genhtml xSDKTrilinos.info --output-directory xSDKTrilinos`
 - ▣ This generates html files in the directory xSDKTrilinos
 - ▣ This step takes a few minutes too

A real example - xSDKTrilinos

43

LCOV - code coverage report

Current view: [top level](#) - xSDKTrilinos/petsc/src

Test: xSDKTrilinos.info

Date: 2016-06-02 15:36:10

	Hit	Total	Coverage
Lines:	342	420	81.4 %
Functions:	77	117	65.8 %

Filename	Line Coverage ↕		Functions ↕	
BelosPETScSolMgr.hpp		84.7 % 166 / 196	68.2 %	30 / 44
Tpetra_PETScAIJGraph.hpp		75.3 % 67 / 89	62.5 %	20 / 32
Tpetra_PETScAIJMatrix.hpp		80.7 % 109 / 135	65.9 %	27 / 41

Generated by: [LCOV version 1.12-4-g04a3c0e](#)

Let's take a look at the solver interface.

```

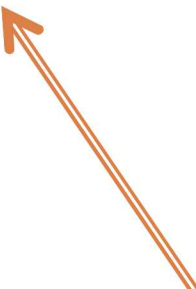
766 : //=====
767 : template<class ScalarType, class MV, class OP>
768 192 : PetscErrorCode PETScSolMgr<ScalarType,MV,OP>::applyPrec(PC M, Vec x, Vec Mx)
769 : {
770 :     using Teuchos::RCP;
771 :     typedef PETScSolMgrHelper<ScalarType,MV,OP> Helper;
772 :
773 :     PetscErrorCode ierr;
774 :     const PetscScalar * xData;
775 :     PetscScalar * MxData;
776 :     void * ptr;
777 :
778 :     // Get the problem out of the context
779 192 :     ierr = PCShellGetContext(M,&ptr); CHKERRQ(ierr);
780 192 :     LinearProblem<ScalarType,MV,OP> * problem = (LinearProblem<ScalarType,MV,OP>*)ptr;
781 :
782 :     // Rip the raw data out of the PETSc vectors
783 192 :     ierr = VecGetArrayRead(x, &xData); CHKERRQ(ierr);
784 192 :     ierr = VecGetArray(Mx, &MxData); CHKERRQ(ierr);
785 :
786 :     // Wrap the PETSc data in a Trilinos Vector
787 192 :     RCP<MV> trilinosX, trilinosMX;
788 192 :     Helper::wrapVector(const_cast<PetscScalar*>(xData), *problem->getLHS(), trilinosX);
789 192 :     Helper::wrapVector(MxData, *problem->getLHS(), trilinosMX);
790 :
791 :     // Perform the multiplication
792 192 :     if(problem->isLeftPrec()) {
793 192 :         problem->applyLeftPrec(*trilinosX, *trilinosMX);
794 :     }
795 :     else {
796 0 :         problem->applyRightPrec(*trilinosX, *trilinosMX);
797 :     }
798 :
799 :     // Unwrap the vectors; this is necessary if we copied data in the wrap step
800 192 :     Helper::unwrapVector(MxData, trilinosMX);
801 :
802 :     // Restore the PETSc vectors
803 192 :     ierr = VecRestoreArrayRead(x,&xData); CHKERRQ(ierr);
804 192 :     ierr = VecRestoreArray(Mx,&MxData); CHKERRQ(ierr);
805 :
806 192 :     return 0;
807 : }
808 :

```

A real example - xSDKTrilinos

45

```
791      :    // Perform the multiplication
792      192 :    if(problem->isLeftPrec()) {
793      192 :        problem->applyLeftPrec(*trilinosX, *trilinosMX);
794      :    }
795      :    else {
796      0 :        problem->applyRightPrec(*trilinosX, *trilinosMX);
797      :    }
```



Oops. I never tested the RIGHT preconditioning branch.

A hands-on gcov tutorial

46

- <https://amklinv.github.io/morpheus/index.html>

Coverity scan

47

- A free cloud-based static analysis product for open source code
 - ▣ Also available for non open source code, but not free
- Analyzes over 4000 open source projects
- Used to analyze
 - ▣ Sudden unintended acceleration of Toyota vehicles
 - ▣ Large Hadron Collider software
 - ▣ Mars Curiosity rover flight software
 - ▣ Libre Office

Coverity scan

48

- Automatically looks at all the different paths
- Finds
 - ▣ Resource leaks
 - ▣ Dereferences of null pointers
 - ▣ Use of uninitialized data
 - ▣ Memory corruptions
 - ▣ Control flow issues
 - ▣ Use of resources that have been freed
 - ▣ And more!

How to use coverity scan

49

- Create a project
- Tell it about your open source license
- Options for performing the scan:
 - ▣ Upload a tarball
 - ▣ Point it to a URL
 - ▣ Use TravisCI ([https://scan.coverity.com/travis ci](https://scan.coverity.com/travis_ci))

Coverity analysis of Trilinos

50

- <https://scan.coverity.com/projects/1680>

Methodology

Evaluating project needs

Devising testing regime

Examples from Alquimia, Amanzi and Trilinos

Evaluating Project Needs

52

Why not use the most stringent testing always ?

- Effort spent in devising tests and testing regime are a tax on team resources
- When does the tax become too high
 - ▣ The team is not able to meet its code use objectives
- When is the tax paid too low
 - ▣ Necessary oversight not provided
 - ▣ Defects in code sneak through

Evaluating Project Needs

53

- Objectives
 - ▣ Proof of concept
 - ▣ Limited research use
 - ▣ Library
 - ▣ Production – simulations and analysis
- Team
 - ▣ Number of developers
 - ▣ Background of developers
 - ▣ Geographical spread
- Lifecycle stages
- Lifetime
 - ▣ New code versus some legacy components
- Complexity

Commonalities

54

- Unit testing is always good
 - ▣ It is unlikely to be sufficient
- Verification of expected behavior
- Understanding the range of validity and applicability is always important
 - ▣ Especially for individual solvers

Customizing for Project Needs : Objectives

55

- Proof of concept
 - ▣ Nothing more than the common testing of previous slide
- Limited use
 - ▣ Manually run test-suite before each use may suffice
 - Coverage is still important
- Library
 - ▣ Depends on team and complexity
- Regular simulation and analysis
 - ▣ Depends on team and complexity
 - ▣ Testing coverage needs system level integrated coverage

Customizing for Project Needs : Team

56

- One to two developers – periodic manual testing and verification
- Mid size to large team – automated test suite running regularly
- Subgroups within the team – automated test-suite with tests of different granularity
 - ▣ May also need multiple suites run on their own schedules

Other Factors

57

- Frequency of testing depends upon lifecycle stage
 - ▣ Midsize to large team working on the same code component doing rapid development – ideally continuous integration
 - ▣ Stable mature code - regular automated testing
 - ▣ Refactoring – needs its own strategy
- Complexity and Lifetime
 - ▣ Affect the testing regime being devised
 - ▣ Testing needs and strategy differ when code incorporates legacy components

Devising the Testing Regime

58

- ❑ Ignoring the easy cases
- ❑ Considering complexity and long lifetime
 - ▣ What runs in the regular test-suite?
 - ▣ If there are subgroups, what goes into the separate test-suites?
 - ▣ How often should each test-suite run?
 - ▣ How do you ensure interoperability coverage?
- ❑ The question to answer is how to balance the tax amount for maximum productivity.

Examples

59

- From Alquimia, amanzi and Trilinos
- Focus on different team sizes and objectives
- Different lifetime spans

How is real DOE code tested?

60

	How many developers?	How much code?	How frequent are changes?
Alquimia	< 1 FTE	O(1,000) lines of code	Every few months
Amanzi	About a dozen	O(100,000) lines of code	A few commits every day
Trilinos	A few dozen	O(1,000,000) lines of code	About 12 per day

Slide 60

KAM8

This is the beginning of the examples section (30 min)

Klinvex, Alicia Marie, 8/18/2016

What is Alquimia?

61

- Biogeochemistry API and wrapper library
- Provides a unified interface to existing geochemistry engines
 - ▣ CrunchFlow
 - ▣ PFLOTRAN
- Allows subsurface flow and transport simulators to access a range of functionality
- NOT an implementation of a biogeochemistry reaction library
- Does NOT perform geochemical calculations

How is Alquimia tested?

62

- Continuous integration testing using Travis CI

What is Amanzi/ATS?

63

□ Amanzi

- ▣ A parallel flow and reactive transport simulator
- ▣ Used to analyze multiple DOE waste disposal sites
- ▣ Example application: modeling hydrological and biogeochemical cycling in the Colorado River System
 - Carbon cycling is especially important because of its role in regulating atmospheric CO₂

□ ATS

- ▣ Advanced Terrestrial Simulator
- ▣ Built on Amanzi
- ▣ Adds physics capability to solve equations for ecosystem hydrology

Amanzi/ATS testing practices

64

- 156 tests that can be run via “ctest”
 - ▣ No continuous integration, but developers are expected to run the test suite before committing
 - ▣ Medium and fine-grained tests
- New physics contributions are required to come with new system-level tests
- Various granularity tests

Slide 64

KAM6

David/Ethan: is there any automated testing for Amanzi/ATS?

Klinvex, Alicia Marie, 8/18/2016

Amanzi/ATS testing granularity

65

- Unit tests
 - ▣ Code is highly componentized
- Medium-grained component tests
 - ▣ Discretizations
 - ▣ Solvers
- Coarse-grained system-level tests
 - ▣ Test full capability
 - ▣ Serve as example for new users

What is Trilinos?

66

- A collection of libraries intended to be used as building blocks for the development of scientific applications
- Organized into 66 packages
 - ▣ Linear solvers
 - ▣ Nonlinear solvers
 - ▣ Eigensolvers
 - ▣ And more!
- 10,000+ commits
- 135 contributors (according to github)
- Millions of lines of code

How is Trilinos tested?

67

- Trilinos has 1500 tests between its 66 packages
- Developers are strongly advised to run a checkin test script when committing
- Automated testing on a variety of different platforms

Checkin test script

68

- ❑ Detects which packages were modified by your commits
- ❑ Determines which packages you potentially broke
- ❑ Configures, builds, and tests those packages
 - ▣ On success, pushes to repo
 - ▣ On failure, reports why it failed
- ❑ Useful for ensuring your changes don't break another package
- ❑ May take a while, but many people run it overnight

Why do we do automated testing if everyone uses the checkin script?

69

- May test a different set of packages
- May test different environments
 - ▣ Do your changes work with Intel compilers as well as GNU?
 - ▣ Do your changes work on a mac?
 - ▣ Do your changes work with CUDA?
- Identifies a small set of commits that could have broken a build or test
 - ▣ Average 12 commits per day
 - ▣ Identifies the person who knows how to un-break it
- Bugs are easier to fix if caught early

What if “bad people” don’t use the checkin script?

70

- Their commit doesn’t include the checkin script information

The screenshot shows a GitHub pull request interface for the `trilinos / Trilinos` repository. The pull request title is **Tpetra: Add "compare Maps" utility executable to examples**. The description states: `@trilinos/tpetra` This is useful for #438 and #558, among others. The build/test summary shows enabled packages (TpetraCore) and disabled packages (FEI, PyTrilinos, Moertel, STK, SEACAS, ThreadPool, OptiPack, Rythmos, Intrepid, ROL). The commit message includes MPI_DEBUG and SERIAL_RELEASE test results. The commit is by `mhoemmen` 4 days ago. The interface shows 5 changed files with 226 additions and 0 deletions. The commit hash is `566b0db7a2dac6455644bcc3ebb01d02f47dac3c`.

trilinos / Trilinos

Unwatch 73 Unstar 98 Fork 58

Code Issues 276 Pull requests 28 Wiki Pulse Graphs

Tpetra: Add "compare Maps" utility executable to examples [Browse files](#)

@trilinos/tpetra This is useful for #438 and #558, among others.

Build/Test Cases Summary
Enabled Packages: TpetraCore
Disabled Packages: FEI,PyTrilinos,Moertel,STK,SEACAS,ThreadPool,OptiPack,Rythmos,Intrepid,ROL
0) MPI_DEBUG => passed: passed=100,notpassed=0 (5.17 min)
1) SERIAL_RELEASE => passed: passed=74,notpassed=0 (2.29 min)
Other local commits for this build/test group: [b945495](#)

master

mhoemmen committed 4 days ago 1 parent [b945495](#) commit [566b0db7a2dac6455644bcc3ebb01d02f47dac3c](#)

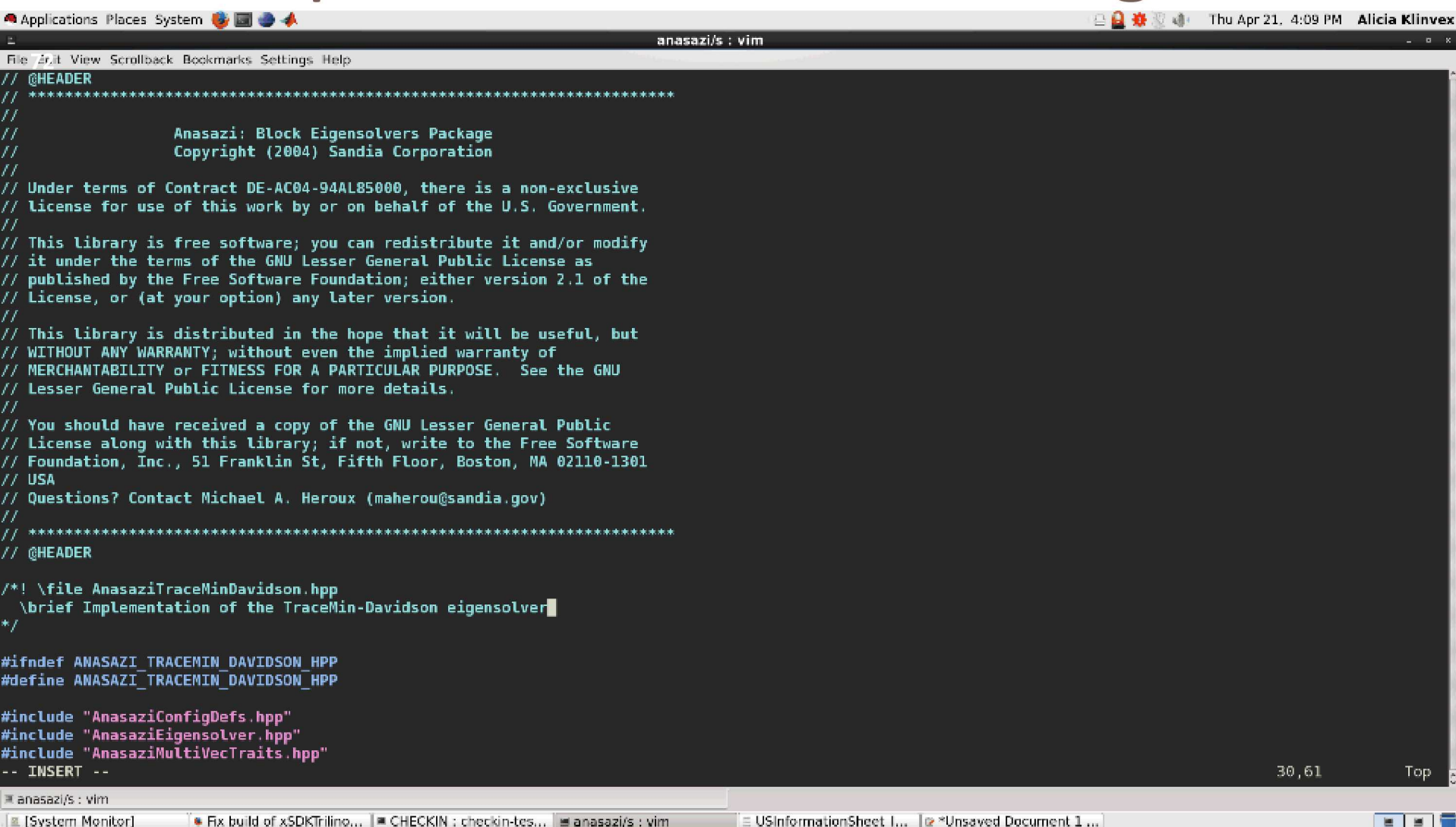
Showing 5 changed files with 226 additions and 0 deletions. [Unified](#) [Split](#)

Checkin test script examples

71

- Example 1: a harmless change to a comment
- Example 2: breaking the build
- Example 3: breaking some tests

Example 1: a harmless change



```
Applications Places System Thu Apr 21, 4:09 PM Alicia Klinvex
anasazi/s : vim
File Edit View Scrollback Bookmarks Settings Help
// @HEADER
// *****
//
// Anasazi: Block Eigensolvers Package
// Copyright (2004) Sandia Corporation
//
// Under terms of Contract DE-AC04-94AL85000, there is a non-exclusive
// license for use of this work by or on behalf of the U.S. Government.
//
// This library is free software; you can redistribute it and/or modify
// it under the terms of the GNU Lesser General Public License as
// published by the Free Software Foundation; either version 2.1 of the
// License, or (at your option) any later version.
//
// This library is distributed in the hope that it will be useful, but
// WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
// Lesser General Public License for more details.
//
// You should have received a copy of the GNU Lesser General Public
// License along with this library; if not, write to the Free Software
// Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301
// USA
// Questions? Contact Michael A. Heroux (maherou@sandia.gov)
// *****
// @HEADER

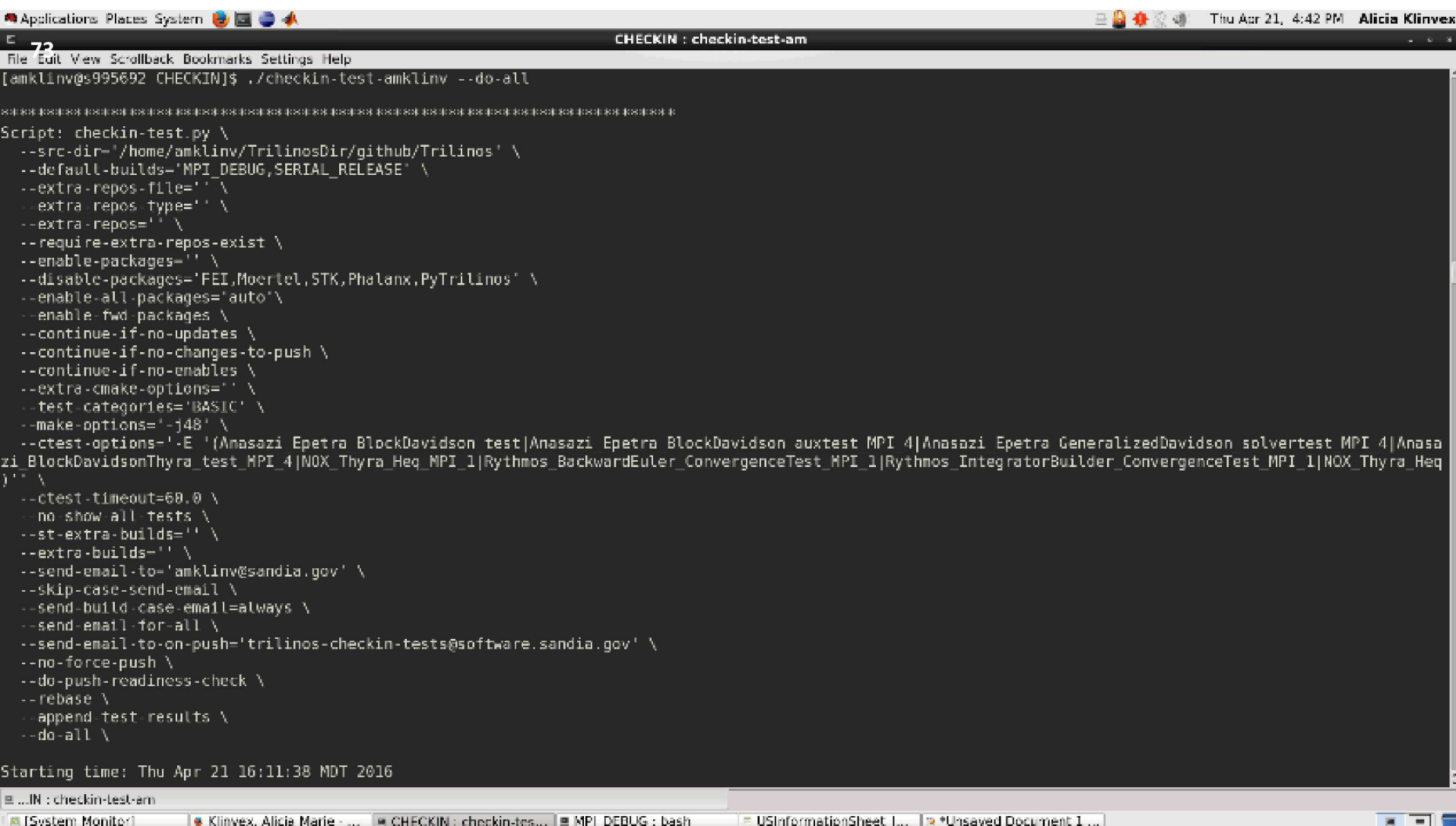
/*! \file AnasaziTraceMinDavidson.hpp
  \brief Implementation of the TraceMin-Davidson eigensolver
  */

#ifndef ANASAZI_TRACEMIN_DAVIDSON_HPP
#define ANASAZI_TRACEMIN_DAVIDSON_HPP

#include "AnasaziConfigDefs.hpp"
#include "AnasaziEigensolver.hpp"
#include "AnasaziMultiVecTraits.hpp"

-- INSERT --
30,61 Top
anasazi/s : vim
[System Monitor] Fix build of xSDKTrilino... CHECKIN : checkin-tes... anasazi/s : vim USInformationSheet I... *Unsaved Document 1 ...
```

Example 1: a harmless change



The screenshot shows a terminal window titled 'CHECKIN : checkin-test-am'. The user is logged in as 'amklinv' on host 's995692'. The command executed is './checkin-test-amklinv --do-all'. The script 'checkin-test.py' is run with various options. The output shows a list of test categories and options, followed by the start time 'Thu Apr 21 16:11:38 MDT 2016'. The terminal window has a menu bar with 'File', 'Edit', 'View', 'Scrollbar', 'Bookmarks', 'Settings', and 'Help'. The status bar at the bottom shows the current directory '...IN : checkin-test-am' and several open files: '[System Monitor]', 'Klinvex, Alicia Marie - ...', 'CHECKIN: checkin-tes...', 'MPI_DEBUG : bash', 'USInformationSheet_L...', and '*Unsaved Document 1 ...'.

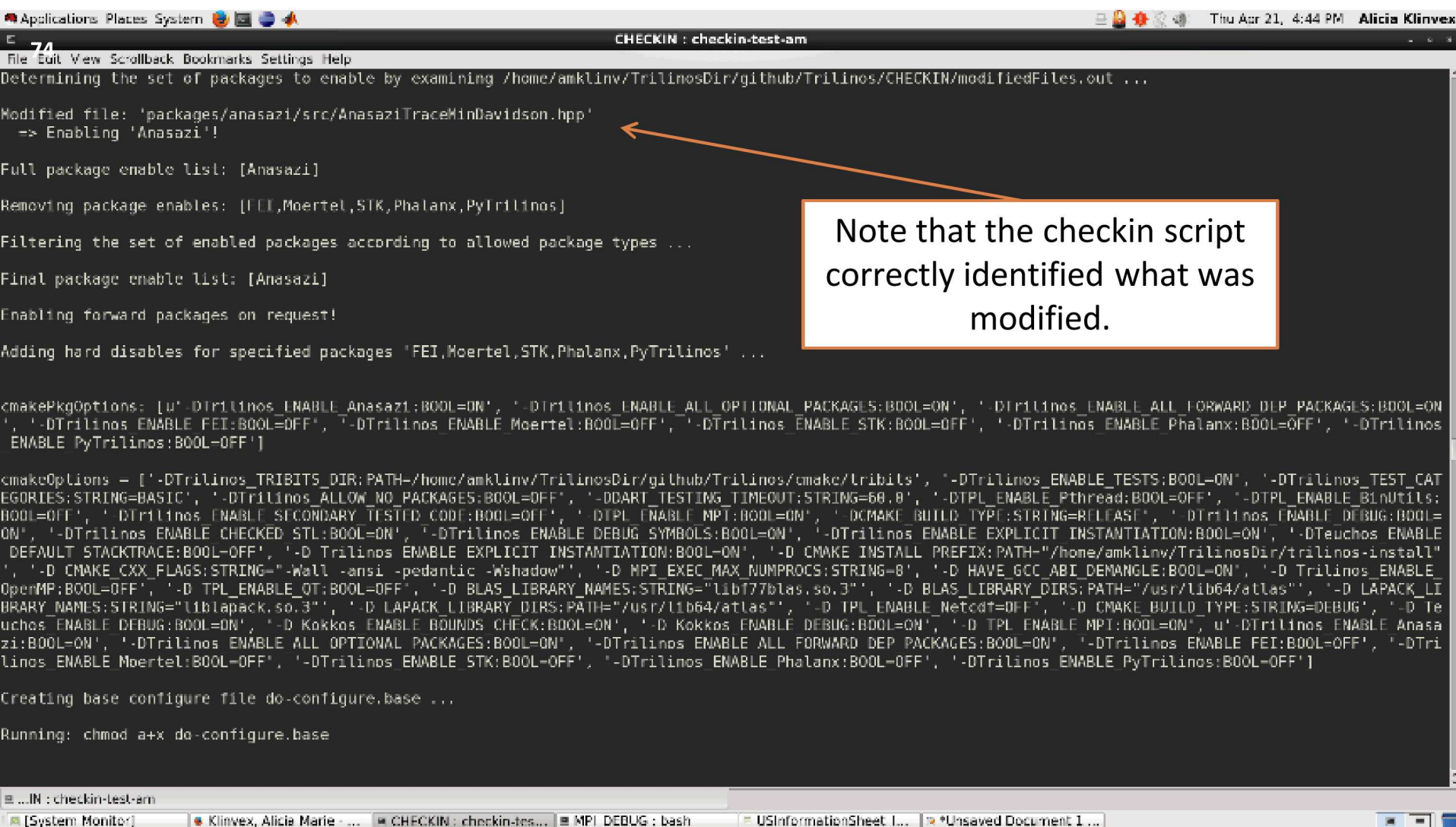
```
Applications Places System Thu Apr 21, 4:42 PM Alicia Klinv...
CHECKIN : checkin-test-am
File Edit View Scrollback Bookmarks Settings Help
[amklinv@s995692 CHECKIN]$ ./checkin-test-amklinv --do-all

Script: checkin-test.py \
--src-dir="/home/amklinv/TrilinosDir/github/Trilinos" \
--default-builds="MPI_DEBUG,SERIAL_RELEASE" \
--extra-repos-file="" \
--extra-repos-type="" \
--extra-repos="" \
--require-extra-repos-exist \
--enable-packages="" \
--disable-packages="FEI,Moortel,STK,Phalanx,PyTrilinos" \
--enable-all-packages="auto" \
--enable-fwd-packages \
--continue-if-no-updates \
--continue-if-no-changes-to-push \
--continue-if-no-enables \
--extra-cmake-options="" \
--test-categories="BASIC" \
--make-options="-j48" \
--ctest-options="-E '(Anasazi Epetra BlockDavidson test|Anasazi Epetra BlockDavidson auxtest MPI 4|Anasazi Epetra GeneralizedDavidson solvertest MPI 4|Anasazi BlockDavidsonThyra_test_MPI_4|NOX_Thyra_Heq_MPI_1|Rythmos_BackwardEuler_ConvergenceTest_MPI_1|Rythmos_IntegratorBuilder_ConvergenceTest_MPI_1|NOX_Thyra_Heq)'" \
--ctest-timeout=60.0 \
--no-show-all-tests \
--st-extra-builds="" \
--extra-builds="" \
--send-email-to="amklinv@sandia.gov" \
--skip-case-send-email \
--send-build-case-email=always \
--send-email-for-all \
--send-email-to-on-push="trilinos-checkin-tests@software.sandia.gov" \
--no-force-push \
--do-push-readiness-check \
--rebase \
--append-test-results \
--do-all \

Starting time: Thu Apr 21 16:11:38 MDT 2016

...IN : checkin-test-am
[System Monitor] Klinvex, Alicia Marie - ... CHECKIN: checkin-tes... MPI_DEBUG : bash USInformationSheet_L... *Unsaved Document 1 ...
```

Example 1: a harmless change



```
Applications Places System Thu Apr 21, 4:44 PM Alicia Klinvex
CHECKIN : checkin-test-am
File Edit View Scrollback Bookmarks Settings Help
Determining the set of packages to enable by examining /home/amklinv/TrilinosDir/github/Trilinos/CHECKIN/modifiedFiles.out ...
Modified file: 'packages/anasazi/src/AnasaziTraceMinDavidson.hpp'
=> Enabling 'Anasazi'
Full package enable list: [Anasazi]
Removing package enables: [FEI,Moertel,STK,Phalanx,PyTrilinos]
Filtering the set of enabled packages according to allowed package types ...
Final package enable list: [Anasazi]
Enabling forward packages on request!
Adding hard disables for specified packages 'FEI,Moertel,STK,Phalanx,PyTrilinos' ...

cmakePkgOptions: [u'-DTrilinos_ENABLE_Anasazi:BOOL=ON', '-DTrilinos_ENABLE_ALL_OPTIONAL_PACKAGES:BOOL=ON', '-DTrilinos_ENABLE_ALL_FORWARD_DEP_PACKAGES:BOOL=ON', '-DTrilinos_ENABLE_FEI:BOOL=OFF', '-DTrilinos_ENABLE_Moertel:BOOL=OFF', '-DTrilinos_ENABLE_STK:BOOL=OFF', '-DTrilinos_ENABLE_Phalanx:BOOL=OFF', '-DTrilinos_ENABLE_PyTrilinos:BOOL=OFF']

cmakeOptions = ['-DTrilinos_TRIBITS_DIR:PATH=/home/amklinv/TrilinosDir/github/Trilinos/cmake/tribits', '-DTrilinos_ENABLE_TESTS:BOOL=ON', '-DTrilinos_TEST_CATEGORIES:STRING=BASIC', '-DTrilinos_ALLOW_NO_PACKAGES:BOOL=OFF', '-DDART_TESTING_TIMEOUT:STRING=60.0', '-DTPL_ENABLE_Pthread:BOOL=OFF', '-DTPL_ENABLE_BluTils:BOOL=OFF', '-DTrilinos_ENABLE_SECONDARY_TESTED_CODE:BOOL=OFF', '-DTPL_ENABLE_MPI:BOOL=ON', '-DCMAKE_BUILD_TYPE:STRING=RELEASE', '-DTrilinos_ENABLE_DEBUG:BOOL=ON', '-DTrilinos_ENABLE_CHECKED_STL:BOOL=ON', '-DTrilinos_ENABLE_DEBUG_SYMBOLS:BOOL=ON', '-DTrilinos_ENABLE_EXPLICIT_INSTANTIATION:BOOL=ON', '-DTeuchos_ENABLE_DEFAULT_STACKTRACE:BOOL=OFF', '-DTrilinos_ENABLE_EXPLICIT_INSTANTIATION:BOOL=ON', '-D CMAKE_INSTALL_PREFIX:PATH=/home/amklinv/TrilinosDir/trilinos-install', '-D CMAKE_CXX_FLAGS:STRING=-Wall -ansi -pedantic -Wshadow', '-D MPI_EXEC_MAX_NUMPROCS:STRING=8', '-D HAVE_GCC_ABI_DEMANGLE:BOOL=ON', '-D Trilinos_ENABLE_OpenMP:BOOL=OFF', '-D TPL_ENABLE_QT:BOOL=OFF', '-D BLAS_LIBRARY_NAMES:STRING=libf77blas.so.3', '-D BLAS_LIBRARY_DIRS:PATH=/usr/lib64/atlas', '-D LAPACK_LIBRARY_NAMES:STRING=liblapack.so.3', '-D LAPACK_LIBRARY_DIRS:PATH=/usr/lib64/atlas', '-D TPL_ENABLE_Netcdf:OFF', '-D CMAKE_BUILD_TYPE:STRING=DEBUG', '-D Teuchos_ENABLE_DEBUG:BOOL=ON', '-D Kokkos_ENABLE_BOUNDS_CHECK:BOOL=ON', '-D Kokkos_ENABLE_DEBUG:BOOL=ON', '-D TPL_ENABLE_MPI:BOOL=ON', u'-DTrilinos_ENABLE_Anasazi:BOOL=ON', '-DTrilinos_ENABLE_ALL_OPTIONAL_PACKAGES:BOOL=ON', '-DTrilinos_ENABLE_ALL_FORWARD_DEP_PACKAGES:BOOL=ON', '-DTrilinos_ENABLE_FEI:BOOL=OFF', '-DTrilinos_ENABLE_Moertel:BOOL=OFF', '-DTrilinos_ENABLE_STK:BOOL=OFF', '-DTrilinos_ENABLE_Phalanx:BOOL=OFF', '-DTrilinos_ENABLE_PyTrilinos:BOOL=OFF']

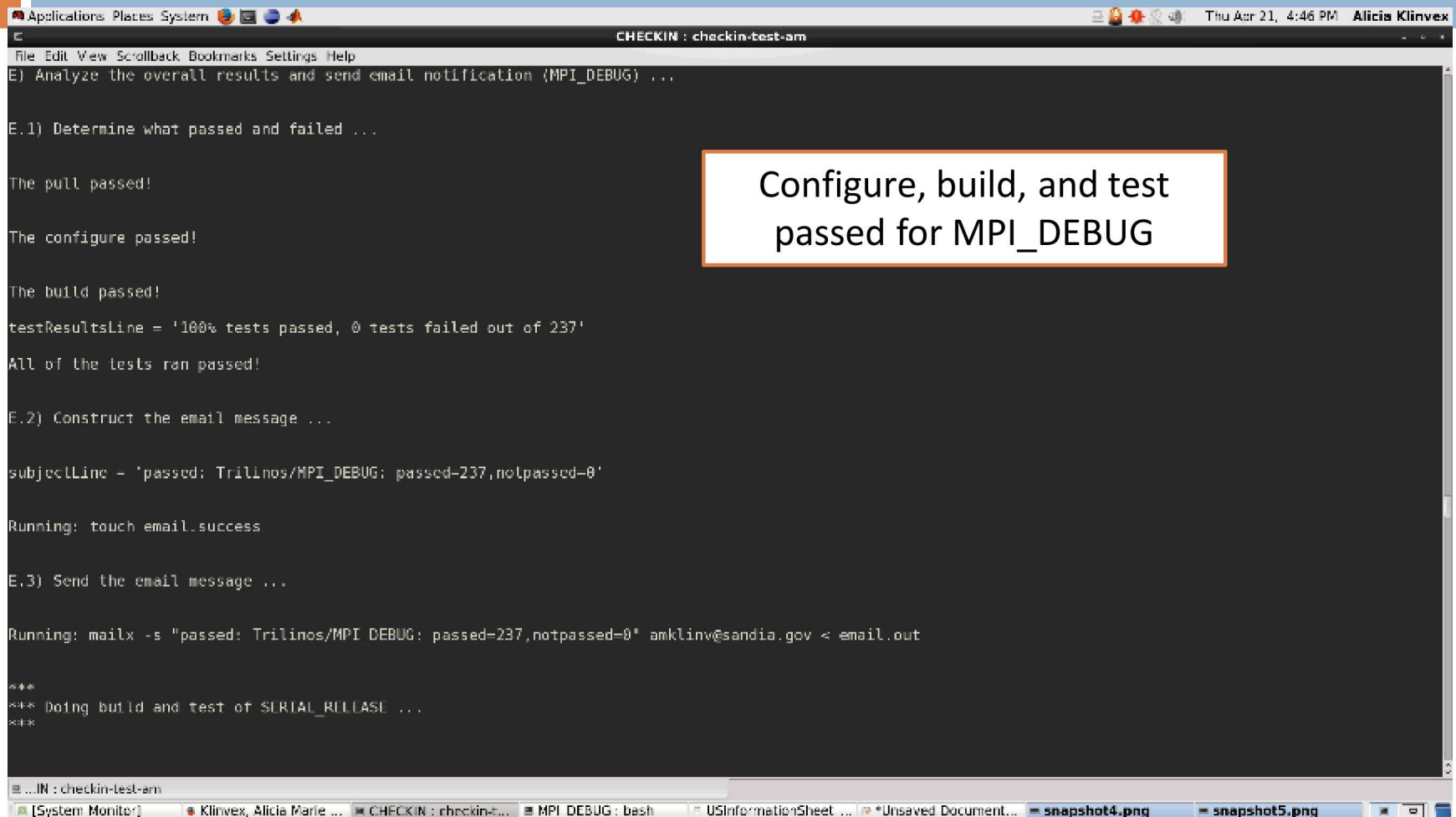
Creating base configure file do-configure.base ...
Running: chmod a+x do-configure.base

...IN : checkin-test-am
[System Monitor] Klinvex, Alicia Marie - ... CHECKIN : checkin-test-am MPI_DEBUG : bash USInformationSheet_L... *Unsaved Document 1 ...
```

Note that the checkin script correctly identified what was modified.

Example 1: a harmless change

75



```
Applications Places System
CHECKIN : checkin-test-am
File Edit View Scrollback Bookmarks Settings Help
E) Analyze the overall results and send email notification (MPI_DEBUG) ...

E.1) Determine what passed and failed ...

The pull passed!

The configure passed!

The build passed!

testResultsLine = '100% tests passed, 0 tests failed out of 237'
All of the tests ran passed!

E.2) Construct the email message ...

subjectLine = 'passed: Trilinos/MPI_DEBUG: passed=237,notpassed=0'

Running: touch email.success

E.3) Send the email message ...

Running: mailx -s "passed: Trilinos/MPI_DEBUG: passed=237,notpassed=0" amklinv@sandia.gov < email.out

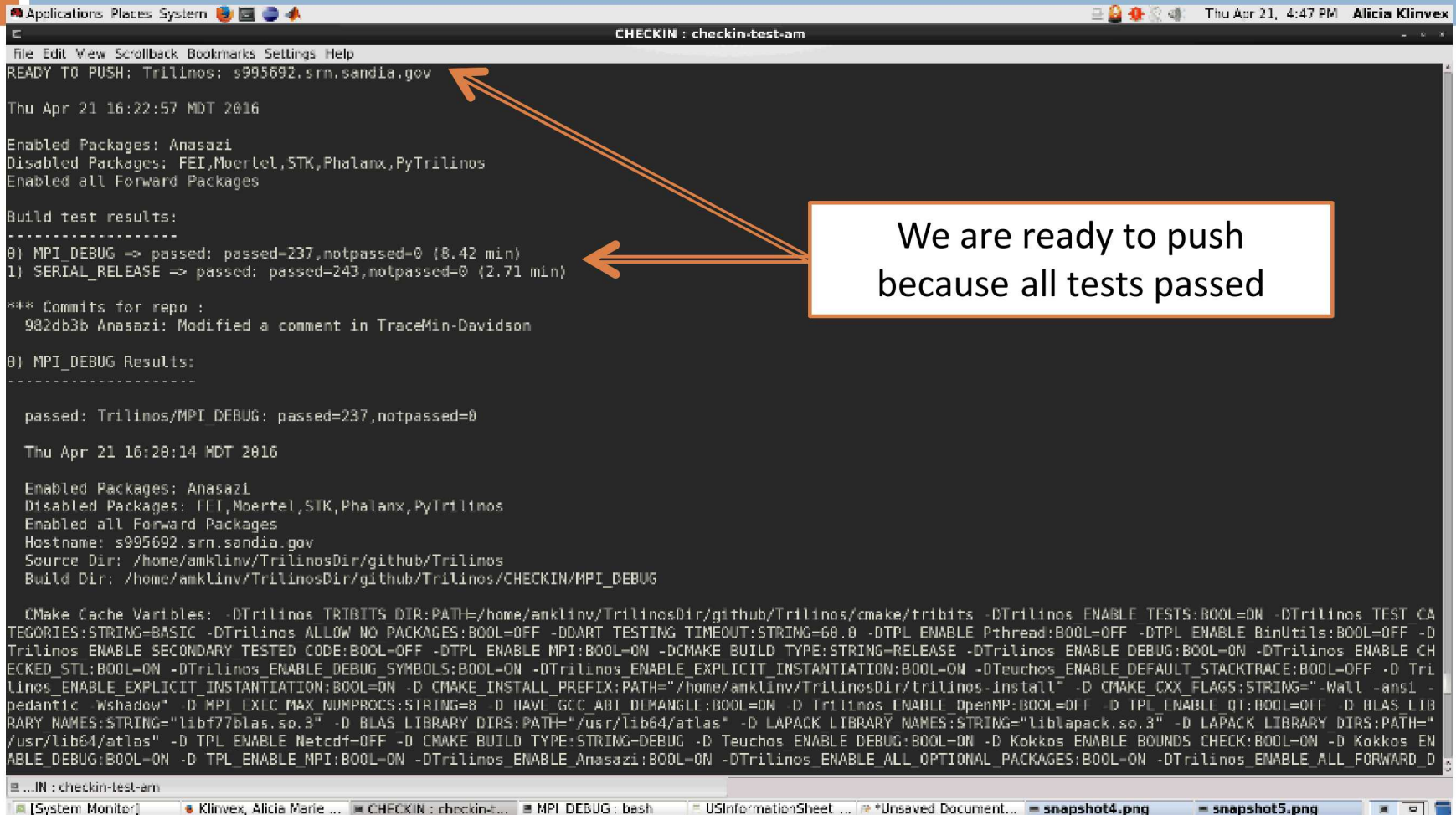
***
*** Doing build and test of SERIAL_RELEASE ...
***

...IN : checkin-test-am
```

Configure, build, and test
passed for MPI_DEBUG

Example 1: a harmless change

76

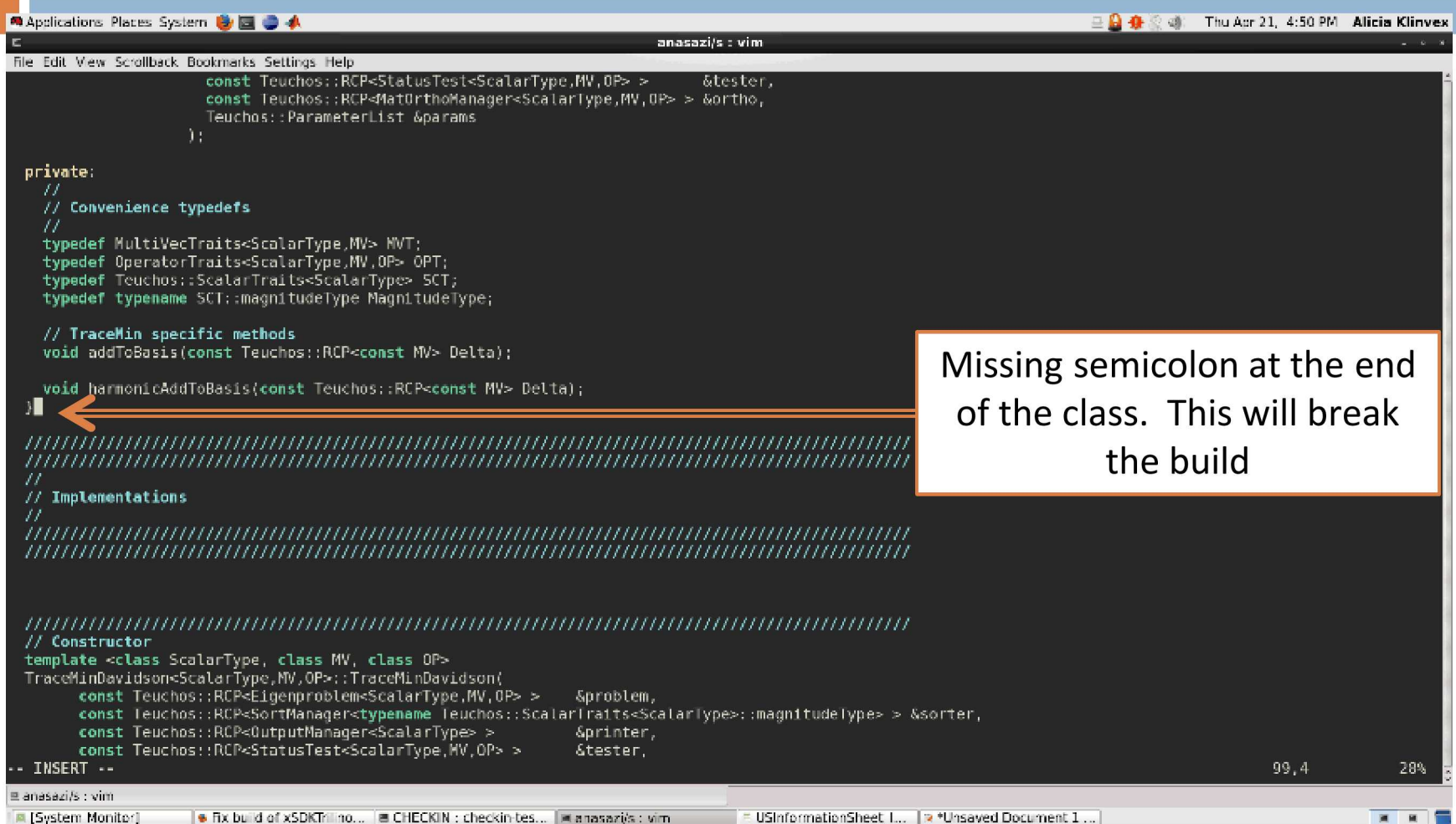


```
Applications Places System Thu Apr 21, 4:47 PM Alicia Klinvex
CHECKIN : checkin-test-am
File Edit View Scrollback Bookmarks Settings Help
READY TO PUSH: Trilinos: s995692.srn.sandia.gov
Thu Apr 21 16:22:57 MDT 2016
Enabled Packages: Anasazi
Disabled Packages: FEI,Moertel,STK,Phalanx,PyTrilinos
Enabled all Forward Packages
Build test results:
-----
0) MPI_DEBUG => passed: passed=237,notpassed=0 (8.42 min)
1) SERIAL_RELEASE => passed: passed=243,notpassed=0 (2.71 min)
*** Commits for repo :
    982db3b Anasazi: Modified a comment in TraceMin-Davidson
0) MPI_DEBUG Results:
-----
passed: Trilinos/MPI_DEBUG: passed=237,notpassed=0
Thu Apr 21 16:28:14 MDT 2016
Enabled Packages: Anasazi
Disabled Packages: FEI,Moertel,STK,Phalanx,PyTrilinos
Enabled all Forward Packages
Hostname: s995692.srn.sandia.gov
Source Dir: /home/amklinv/TrilinosDir/github/Trilinos
Build Dir: /home/amklinv/TrilinosDir/github/Trilinos/CHECKIN/MPI_DEBUG
CMake Cache Variables: -DTrilinos TRIBITS_DIR:PATH=/home/amklinv/TrilinosDir/github/Trilinos/cmake/tribits -DTrilinos ENABLE_TESTS:BOOL=ON -DTrilinos TEST CA
TEGORIES:STRING=BASIC -DTrilinos ALLOW NO PACKAGES:BOOL=OFF -DDART TESTING TIMEOUT:STRING=60.0 -DTPL ENABLE Pthread:BOOL=OFF -DTPL ENABLE BinUtils:BOOL=OFF -D
Trilinos ENABLE SECONDARY TESTED CODE:BOOL=OFF -DTPL ENABLE MPI:BOOL=ON -DCMAKE BUILD TYPE:STRING=RELEASE -DTrilinos ENABLE DEBUG:BOOL=ON -DTrilinos ENABLE CH
ECKED_STL:BOOL=ON -DTrilinos ENABLE DEBUG SYMBOLS:BOOL=ON -DTrilinos ENABLE EXPLICIT_INSTANTIATION:BOOL=ON -DTeuchos ENABLE DEFAULT_STACKTRACE:BOOL=OFF -D Tri
linos ENABLE EXPLICIT_INSTANTIATION:BOOL=ON -D CMAKE_INSTALL_PREFIX:PATH="/home/amklinv/TrilinosDir/trilinos-install" -D CMAKE_CXX_FLAGS:STRING="-Wall -ansi -
pedantic -Wshadow" -D MPI EXEC MAX NUMPROCS:STRING=8 -D HAVE_GCC_ABI_DLMANGL:BOOL=ON -D Trilinos INABLE OpenMP:BOOL=OFF -D TPL ENABLE QT:BOOL=OFF -D BLAS LIB
RARY NAMES:STRING="libf77blas.so.3" -D BLAS LIBRARY DIRS:PATH="/usr/lib64/atlas" -D LAPACK LIBRARY NAMES:STRING="liblapack.so.3" -D LAPACK LIBRARY DIRS:PATH="
/usr/lib64/atlas" -D TPL ENABLE Netcdf:OFF -D CMAKE BUILD TYPE:STRING=DEBUG -D Teuchos ENABLE DEBUG:BOOL=ON -D Kokkos ENABLE BOUNDS CHECK:BOOL=ON -D Kokkos EN
ABLE_DEBUG:BOOL=ON -D TPL_ENABLE_MPI:BOOL=ON -DTrilinos_ENABLE_Anasazi:BOOL=ON -DTrilinos_ENABLE_ALL_OPTIONAL_PACKAGES:BOOL=ON -DTrilinos_ENABLE_ALL_FORWARD_D
```

We are ready to push
because all tests passed

Example 2: broken build

77



The screenshot shows a Vim editor window with a dark background. The code is a C++ source file. At the top, there are some global definitions and a class definition. The class definition is for a template class. The code is as follows:

```
const Teuchos::RCP<StatusTest<ScalarType,MV,OP> > &tester,
const Teuchos::RCP<MatOrthoManager<ScalarType,MV,OP> > &ortho,
Teuchos::ParameterList &params
};

private:
//
// Convenience typedefs
//
typedef MultiVecTraits<ScalarType,MV> MVT;
typedef OperatorTraits<ScalarType,MV,OP> OPT;
typedef Teuchos::ScalarTraits<ScalarType> SCT;
typedef typename SCT::magnitudetype MagnitudeType;

// TraceMin specific methods
void addToBasis(const Teuchos::RCP<const MV> Delta);

void harmonicAddToBasis(const Teuchos::RCP<const MV> Delta);
}

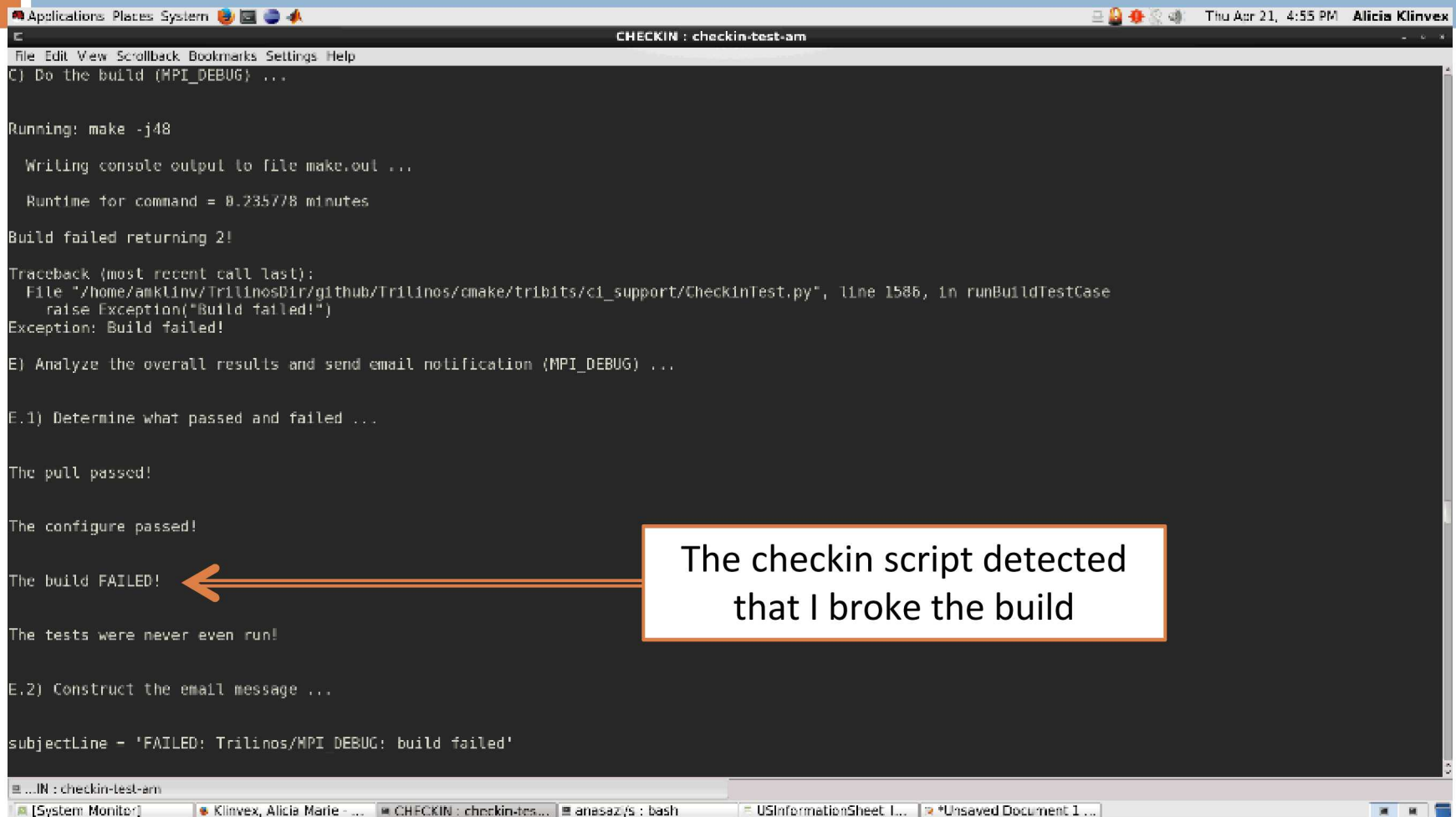
// Implementations
//
// Constructor
template <class ScalarType, class MV, class OP>
TraceMinDavidson<ScalarType,MV,OP>::TraceMinDavidson(
const Teuchos::RCP<Eigenproblem<ScalarType,MV,OP> > &problem,
const Teuchos::RCP<SortManager<typename Teuchos::ScalarTraits<ScalarType>::magnitudetype> > &sorter,
const Teuchos::RCP<OutputManager<ScalarType> > &printer,
const Teuchos::RCP<StatusTest<ScalarType,MV,OP> > &tester,
-- INSERT --
```

A red arrow points from the text box to the closing brace of the class definition, indicating the missing semicolon.

Missing semicolon at the end of the class. This will break the build

Example 2: broken build

78



```
Applications Places System
CHECKIN : checkin-test-am
File Edit View Scrollback Bookmarks Settings Help
C) Do the build (MPI_DEBUG) ...

Running: make -j48

Writing console output to file make.out ...

Runtime for command = 0.235778 minutes

Build failed returning 2!

Traceback (most recent call last):
  File "/home/amklynv/TrilinosDir/github/Trilinos/cmake/tribits/ci_support/CheckinTest.py", line 1586, in runBuildTestCase
    raise Exception("Build failed!")
Exception: Build failed!

E) Analyze the overall results and send email notification (MPI_DEBUG) ...

E.1) Determine what passed and failed ...

The pull passed!

The configure passed!

The build FAILED!

The tests were never even run!

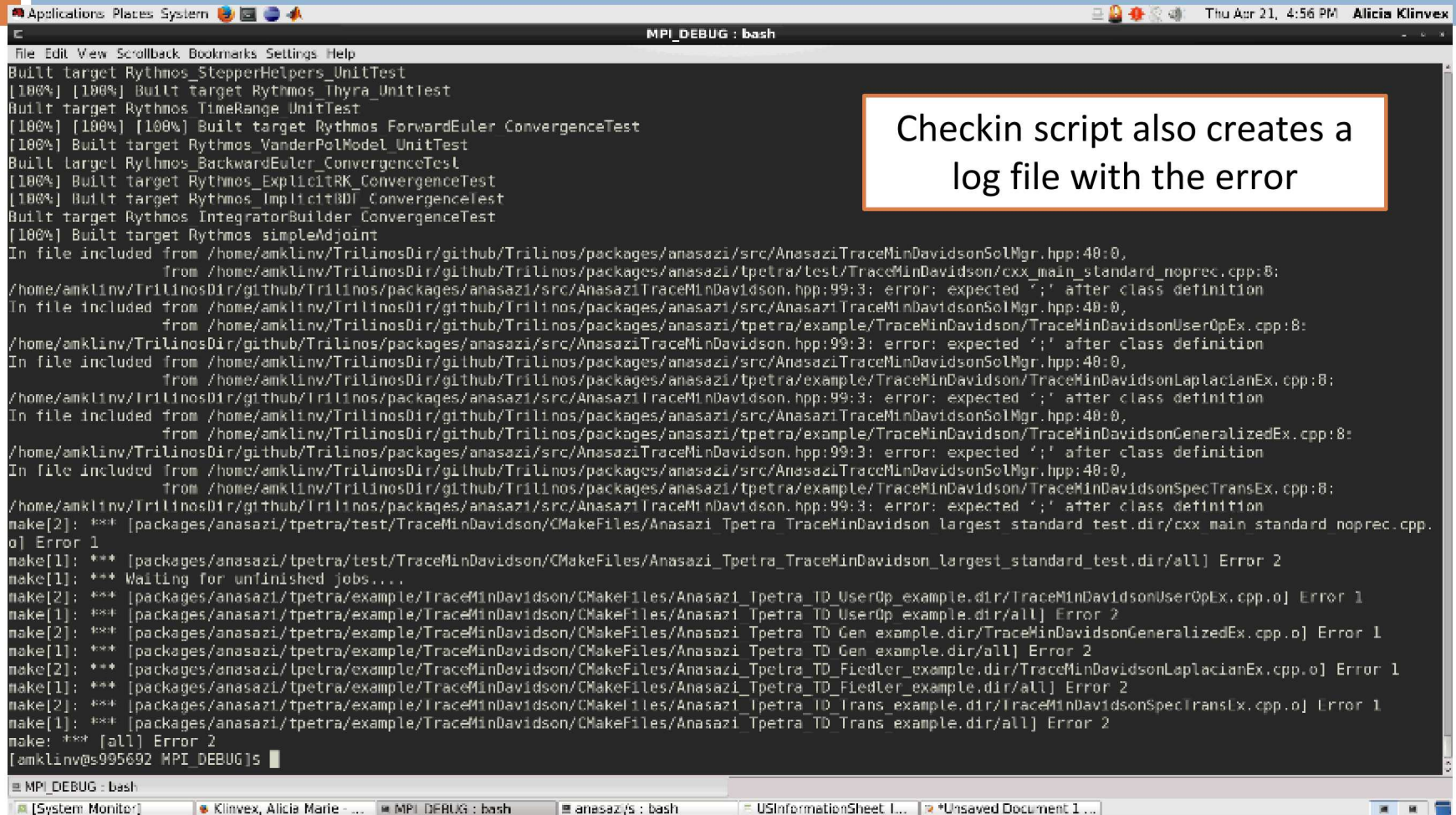
E.2) Construct the email message ...

subjectLine = 'FAILED: Trilinos/MPI DEBUG: build failed'
```

The checkin script detected that I broke the build

Example 2: broken build

79



```
Applications Places System Thu Apr 21, 4:56 PM Alicia Klinvex
MPI_DEBUG : bash
File Edit View Scrollback Bookmarks Settings Help
Built target Rythmos_StepperHelpers_UnitTest
[100%] [100%] Built target Rythmos_Thyra_UnitTest
Built target Rythmos_TimeRange_UnitTest
[100%] [100%] [100%] Built target Rythmos_ForwardEuler_ConvergenceTest
[100%] Built target Rythmos_VanderPolModel_UnitTest
Built target Rythmos_BackwardEuler_ConvergenceTest
[100%] Built target Rythmos_ExplicitRK_ConvergenceTest
[100%] Built target Rythmos_ImplicitRK_ConvergenceTest
Built target Rythmos_IntegratorBuilder_ConvergenceTest
[100%] Built target Rythmos_simpleAdjoint
In file included from /home/amklinv/TrilinosDir/github/Trilinos/packages/anasazi/src/AnasaziTraceMinDavidsonSolMgr.hpp:40:0,
      from /home/amklinv/TrilinosDir/github/Trilinos/packages/anasazi/tpetra/test/TraceMinDavidson/cxx_main_standard_noprec.cpp:8:
/home/amklinv/TrilinosDir/github/Trilinos/packages/anasazi/src/AnasaziTraceMinDavidson.hpp:99:3: error: expected ';' after class definition
In file included from /home/amklinv/TrilinosDir/github/Trilinos/packages/anasazi/src/AnasaziTraceMinDavidsonSolMgr.hpp:40:0,
      from /home/amklinv/TrilinosDir/github/Trilinos/packages/anasazi/tpetra/example/TraceMinDavidson/TraceMinDavidsonUserOpEx.cpp:8:
/home/amklinv/TrilinosDir/github/Trilinos/packages/anasazi/src/AnasaziTraceMinDavidson.hpp:99:3: error: expected ';' after class definition
In file included from /home/amklinv/TrilinosDir/github/Trilinos/packages/anasazi/src/AnasaziTraceMinDavidsonSolMgr.hpp:40:0,
      from /home/amklinv/TrilinosDir/github/Trilinos/packages/anasazi/tpetra/example/TraceMinDavidson/TraceMinDavidsonLaplacianEx.cpp:8:
/home/amklinv/TrilinosDir/github/Trilinos/packages/anasazi/src/AnasaziTraceMinDavidson.hpp:99:3: error: expected ';' after class definition
In file included from /home/amklinv/TrilinosDir/github/Trilinos/packages/anasazi/src/AnasaziTraceMinDavidsonSolMgr.hpp:40:0,
      from /home/amklinv/TrilinosDir/github/Trilinos/packages/anasazi/tpetra/example/TraceMinDavidson/TraceMinDavidsonGeneralizedEx.cpp:8:
/home/amklinv/TrilinosDir/github/Trilinos/packages/anasazi/src/AnasaziTraceMinDavidson.hpp:99:3: error: expected ';' after class definition
In file included from /home/amklinv/TrilinosDir/github/Trilinos/packages/anasazi/src/AnasaziTraceMinDavidsonSolMgr.hpp:40:0,
      from /home/amklinv/TrilinosDir/github/Trilinos/packages/anasazi/tpetra/example/TraceMinDavidson/TraceMinDavidsonSpectransEx.cpp:8:
/home/amklinv/TrilinosDir/github/Trilinos/packages/anasazi/src/AnasaziTraceMinDavidson.hpp:99:3: error: expected ';' after class definition
make[2]: *** [packages/anasazi/tpetra/test/TraceMinDavidson/CMakeFiles/Anasazi_Tpetra_TraceMinDavidson_largest_standard_test.dir/cxx_main_standard_noprec.cpp.o] Error 1
make[1]: *** [packages/anasazi/tpetra/test/TraceMinDavidson/CMakeFiles/Anasazi_Tpetra_TraceMinDavidson_largest_standard_test.dir/all] Error 2
make[1]: *** Waiting for unfinished jobs....
make[2]: *** [packages/anasazi/tpetra/example/TraceMinDavidson/CMakeFiles/Anasazi_Tpetra_TD_UserOp_example.dir/TraceMinDavidsonUserOpEx.cpp.o] Error 1
make[1]: *** [packages/anasazi/tpetra/example/TraceMinDavidson/CMakeFiles/Anasazi_Tpetra_TD_UserOp_example.dir/all] Error 2
make[2]: *** [packages/anasazi/tpetra/example/TraceMinDavidson/CMakeFiles/Anasazi_Tpetra_TD_Gen_example.dir/TraceMinDavidsonGeneralizedEx.cpp.o] Error 1
make[1]: *** [packages/anasazi/tpetra/example/TraceMinDavidson/CMakeFiles/Anasazi_Tpetra_TD_Gen_example.dir/all] Error 2
make[2]: *** [packages/anasazi/tpetra/example/TraceMinDavidson/CMakeFiles/Anasazi_Tpetra_TD_Fiedler_example.dir/TraceMinDavidsonLaplacianEx.cpp.o] Error 1
make[1]: *** [packages/anasazi/tpetra/example/TraceMinDavidson/CMakeFiles/Anasazi_Tpetra_TD_Fiedler_example.dir/all] Error 2
make[2]: *** [packages/anasazi/tpetra/example/TraceMinDavidson/CMakeFiles/Anasazi_Tpetra_TD_Trans_example.dir/TraceMinDavidsonSpectransEx.cpp.o] Error 1
make[1]: *** [packages/anasazi/tpetra/example/TraceMinDavidson/CMakeFiles/Anasazi_Tpetra_TD_Trans_example.dir/all] Error 2
make: *** [all] Error 2
[amklinv@955692 MPI_DEBUG]$
```

Checkin script also creates a log file with the error

MPI_DEBUG : bash

[System Monitor] Klinvex, Alicia Marie - ... MPI_DEBUG : bash anasazi/s : bash USInformationSheet, L... *Unsaved Document 1...

Example 3: broken tests

80

```
Applications Places System anasazi/s : vim Thu Apr 21, 5:02 PM Alicia Klinvex
File Edit View Scrollback Bookmarks Settings Help
useMultipleShifts_ = params.get("Use Multiple Shifts", true);

shiftThresh_ = params.get("Shift Threshold", 1e-2);
useRelShiftThresh_ = params.get("Relative Shift Threshold", true);
shiftNorm_ = params.get("Shift Norm", "2");
TEUCHOS_TEST_FOR_EXCEPTION(shiftNorm_ != "2" && shiftNorm_ != "N", std::invalid_argument,
    "Anasazi::TraceMin::constructor: Invalid value for \"Shift Norm\": valid options are \"2\", \"N\".");

considerClusters_ = params.get("Consider Clusters", true);

projectAllVecs_ = params.get("Project All Vectors", true);
projectLockedVecs_ = params.get("Project Locked Vectors", true);
useRHSR_ = params.get("Use Residual as RHS", true);
useHarmonic_ = params.get("Use Harmonic Ritz Values", false);
computeAllRes_ = params.get("Compute All Residuals", true);

// set the block size and allocate data
int bs = params.get("Block Size", problem->getNEV());
int nb = params.get("Num Blocks", 1);
// setSize(bs,nb);

NEV_ = problem->getNEV();

// Create the Ritz shift operator
ritzOp_ = rcp (new tracemin_ritz_op_type (Op_, MOp_, Prec_));

// Set the maximum number of inner iterations
const int innerMaxIts = params.get ("Maximum Krylov Iterations", 200);
ritzOp_>setMaxIts (innerMaxIts);

alpha_ = params.get ("HSS: alpha", ONE);
}

////////////////////////////////////
// Destructor
template <class ScalarType, class MV, class OP>
TraceMinBase<ScalarType,MV,OP>::~TraceMinBase() {}
-- INSERT --
```

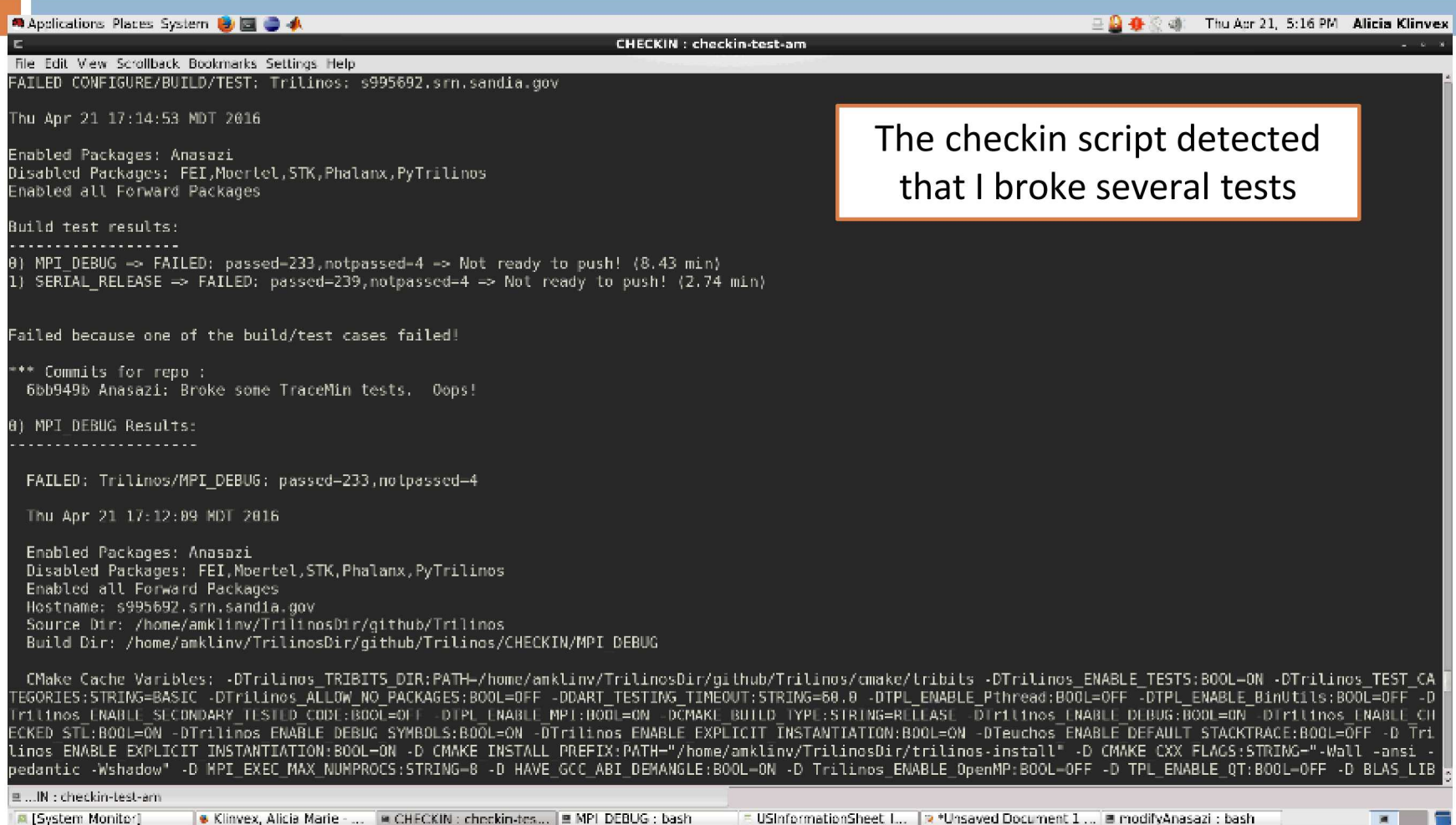
Added a logic error to the code.

746,3 10%

anasazi/s : vim
[System Monitor] Klinvex, Alicia Marie - ... anasazi/s : vim modifyAnasazi: bash USInformationSheet_L... *Unsaved Document 1 ...

Example 3: broken tests

81



```
Applications Places System Thu Apr 21, 5:16 PM Alicia Klinvex
CHECKIN : checkin-test-am
File Edit View Scrollback Bookmarks Settings Help
FAILED CONFIGURE/BUILD/TEST: Trilinos: s995692.srn.sandia.gov

Thu Apr 21 17:14:53 MDT 2016

Enabled Packages: Anasazi
Disabled Packages: FEI,Moertel,STK,Phalanx,PyTrilinos
Enabled all Forward Packages

Build test results:
-----
0) MPI_DEBUG => FAILED: passed=233,notpassed=4 => Not ready to push! (8.43 min)
1) SERIAL_RELEASE => FAILED: passed=239,notpassed=4 => Not ready to push! (2.74 min)

Failed because one of the build/test cases failed!

*** Commits for repp :
    6bb949b Anasazi: Broke some TraceMin tests.  Oops!

0) MPI_DEBUG Results:
-----

FAILED: Trilinos/MPI_DEBUG: passed=233,notpassed=4

Thu Apr 21 17:12:09 MDT 2016

Enabled Packages: Anasazi
Disabled Packages: FEI,Moertel,STK,Phalanx,PyTrilinos
Enabled all Forward Packages
Hostname: s995692.srn.sandia.gov
Source Dir: /home/amklinv/TrilinosDir/github/Trilinos
Build Dir: /home/amklinv/TrilinosDir/github/Trilinos/CHECKIN/MPI_DEBUG

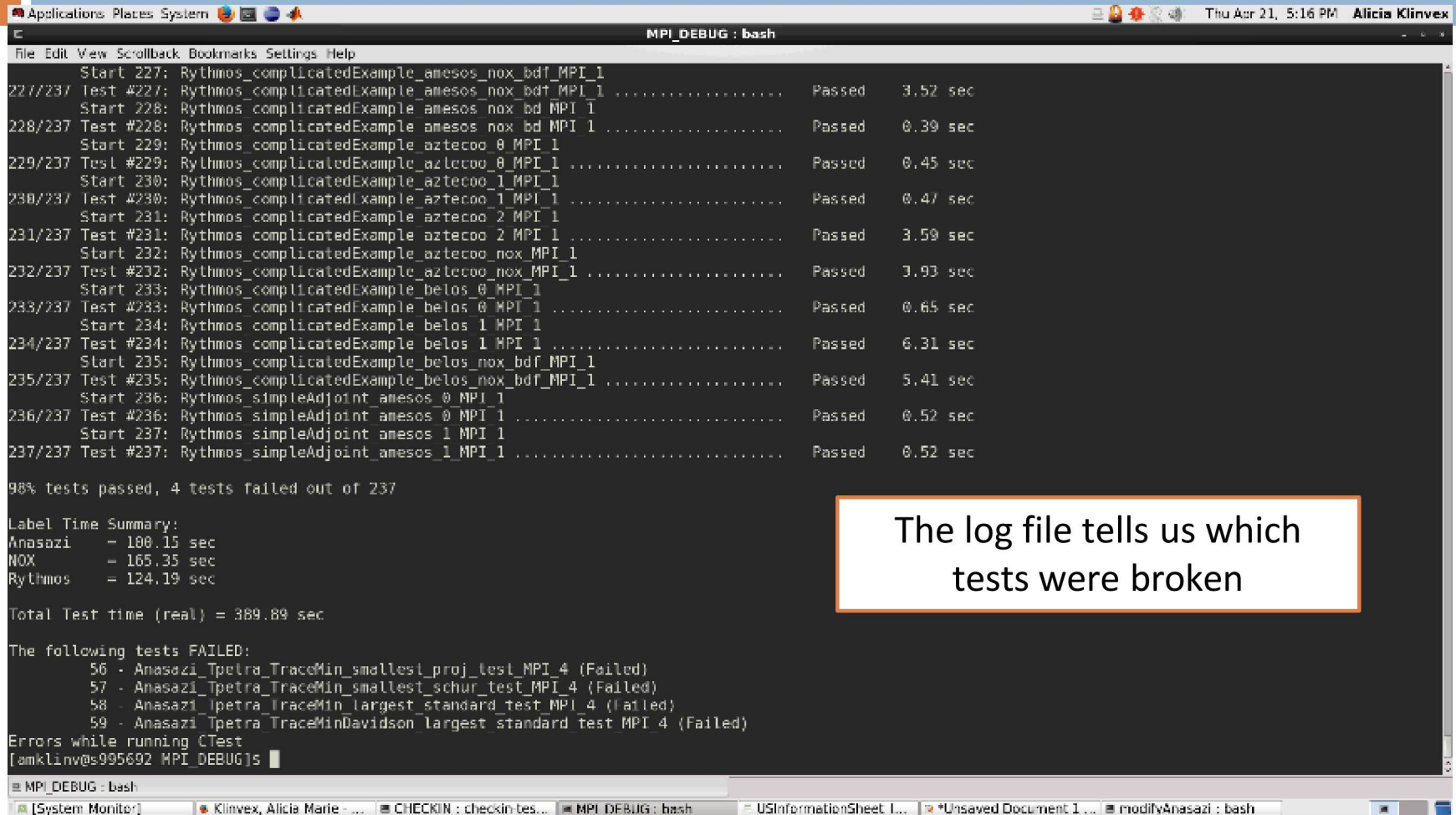
CMake Cache Variables: -DTrilinos_TRIBITS_DIR:PATH=/home/amklinv/TrilinosDir/github/Trilinos/cmake/tribits -DTrilinos_ENABLE_TESTS:BOOL=ON -DTrilinos_TEST_CATEGORIES:STRING=BASIC -DTrilinos_ALLOW_NO_PACKAGES:BOOL=OFF -DDART_TESTING_TIMEOUT:STRING=60.0 -DTPL_ENABLE_Pthread:BOOL=OFF -DTPL_ENABLE_BinUtils:BOOL=OFF -DTrilinos_ENABLE_SECONDARY_TESTED_CODE:BOOL=OFF -DTPL_ENABLE_MPI:BOOL=ON -DCMAKE_BUILD_TYPE:STRING=RELEASE -DTrilinos_ENABLE_DEBUG:BOOL=ON -DTrilinos_ENABLE_CHECKED_STL:BOOL=ON -DTrilinos_ENABLE_DEBUG_SYMBOLS:BOOL=ON -DTrilinos_ENABLE_EXPLICIT_INSTANTIATION:BOOL=ON -DTauchoos_ENABLE_DEFAULT_STACKTRACE:BOOL=OFF -DTrilinos_ENABLE_EXPLICIT_INSTANTIATION:BOOL=ON -DCMAKE_INSTALL_PREFIX:PATH="/home/amklinv/TrilinosDir/trilinos-install" -DCMAKE_CXX_FLAGS:STRING="-Wall -ansi -pedantic -Wshadow" -DMPI_EXEC_MAX_NUMPROCS:STRING=8 -DHAVE_GCC_ABI_DEMANGLE:BOOL=ON -DTrilinos_ENABLE_OpenMP:BOOL=OFF -DTPL_ENABLE_QT:BOOL=OFF -D BLAS_LIB

[Terminal Window]
[System Monitor] [Klinvex, Alicia Marie - ...] [CHECKIN: checkin-test-am] [MPI_DEBUG: bash] [USInformationSheet, L...] [Unsaved Document 1 ...] [modifyAnasazi: bash]
```

The checkin script detected that I broke several tests

Example 3: broken tests

82



```
Applications Places System Thu Apr 21, 5:16 PM Alicia Klinvex
MPI_DEBUG : bash
File Edit View Scrollback Bookmarks Settings Help
Start 227: Rythmos_complicatedExample_amesos_nox_bdf_MPI_1
227/237 Test #227: Rythmos_complicatedExample_amesos_nox_bdf_MPI_1 ..... Passed 3.52 sec
Start 228: Rythmos_complicatedExample_amesos_nox_bd_MPI_1
228/237 Test #228: Rythmos_complicatedExample_amesos_nox_bd_MPI_1 ..... Passed 0.39 sec
Start 229: Rythmos_complicatedExample_aztec00_0_MPI_1
229/237 Test #229: Rythmos_complicatedExample_aztec00_0_MPI_1 ..... Passed 0.45 sec
Start 230: Rythmos_complicatedExample_aztec00_1_MPI_1
230/237 Test #230: Rythmos_complicatedExample_aztec00_1_MPI_1 ..... Passed 0.47 sec
Start 231: Rythmos_complicatedExample_aztec00_2_MPI_1
231/237 Test #231: Rythmos_complicatedExample_aztec00_2_MPI_1 ..... Passed 3.59 sec
Start 232: Rythmos_complicatedExample_aztec00_nox_MPI_1
232/237 Test #232: Rythmos_complicatedExample_aztec00_nox_MPI_1 ..... Passed 3.93 sec
Start 233: Rythmos_complicatedExample_belos_0_MPI_1
233/237 Test #233: Rythmos_complicatedExample_belos_0_MPI_1 ..... Passed 0.65 sec
Start 234: Rythmos_complicatedExample_belos_1_MPI_1
234/237 Test #234: Rythmos_complicatedExample_belos_1_MPI_1 ..... Passed 6.31 sec
Start 235: Rythmos_complicatedExample_belos_nox_bdf_MPI_1
235/237 Test #235: Rythmos_complicatedExample_belos_nox_bdf_MPI_1 ..... Passed 5.41 sec
Start 236: Rythmos_simpleAdjoint_amesos_0_MPI_1
236/237 Test #236: Rythmos_simpleAdjoint_amesos_0_MPI_1 ..... Passed 0.52 sec
Start 237: Rythmos_simpleAdjoint_amesos_1_MPI_1
237/237 Test #237: Rythmos_simpleAdjoint_amesos_1_MPI_1 ..... Passed 0.52 sec

98% tests passed, 4 tests failed out of 237

Label Time Summary:
Anasazi - 100.15 sec
NOX - 165.35 sec
Rythmos - 124.19 sec

Total Test time (real) = 389.89 sec

The following tests FAILED:
 56 - Anasazi_Tpetra_TraceMin_smallest_proj_test_MPI_4 (Failed)
 57 - Anasazi_Tpetra_TraceMin_smallest_schur_test_MPI_4 (Failed)
 58 - Anasazi_Tpetra_TraceMin_largest_standard_test_MPI_4 (Failed)
 59 - Anasazi_Tpetra_TraceMinDavidson_largest_standard_test_MPI_4 (Failed)
Errors while running CTest
[amklinv@s995692 MPI_DEBUG]s
```

The log file tells us which tests were broken

MPI_DEBUG : bash

[System Monitor] Klinvex, Alicia Marie - ... CHECKIN : checkin tes... MPI_DEBUG : bash USInformationSheet_L... *Unsaved Document 1 ... modifyAnasazi : bash

Trilinos automated testing

83

Login All Dashboards

Monday, June 06 2016 08:58:08 MDT



Trilinos

Dashboard

Calendar

Previous

Current

Project

Project

Project	Configure			Build			Test		
	Error	Warning	Pass	Error	Warning	Pass	Not Run	Fail	Pass
Trilinos ▼	1	531	530	0	272	257	0	14	3976

SubProjects

Project	Configure			Build			Test		
	Error	Warning	Pass	Error	Warning	Pass	Not Run	Fail	Pass
Teuchos	0	21	21	0	12	9	0	0	227
ThreadPool	0	1	1	0	0	1			
Sacado	0	2	2	0	2	0	0	0	564
RTOp	0	20	20	0	0	20			
Kokkos	0	19	19	0	0	19	0	0	9
Epetra	0	21	21	0	12	9	0	1	244
Zoltan	0	21	21	0	13	8	0	0	135
Shards	0	1	1	0	0	1			
GlobiPack	0	1	1	0	0	1			

Trilinos automated testing

84

Nightly											
Site	Build Name	Update	Configure		Build		Test			Build Time	Labels
		Files	Error ^	Warn	Error	Warn	Not Run	Fail	Pass		
artemis.srn.sandia.gov	Linux-intel-15.0.2-MPI_RELEASE_DEV_DownStream_ETI_SERIAL-OFF_OPENMP-ON_PTHREAD-OFF_CUDA-OFF_COMPLEX-OFF	68	1	140	0	216	0	3	1256	6 hours ago	(44 labels)
lightsaber.srn.sandia.gov	Linux-GCC-4.7.2-RELEASE_DEV_MueLu_Matlab	69	0	111	0	51	0	0	431	10 hours ago	(25 labels)
enigma.sandia.gov	Linux-GCC-4.8.3-OPENMPI_1.6.4_DEBUG_DEV_MueLu_Basker	69	0	227	0	117	0	0	96	9 hours ago	(25 labels)
hansel.sandia.gov	Linux-GCC-4.4.7-MPI_OPT_DEV_XYCE	121	0	70	0	28	0	0	553	9 hours ago	(13 labels)
enigma.sandia.gov	Linux-GCC-4.8.3-OPENMPI_1.6.4_DEBUG_DEV_MueLu_KLU2	69	0	225	0	91	0	0	73	8 hours ago	(25 labels)
enigma.sandia.gov	Linux-GCC-4.8.3-OPENMPI_1.6.4_DEBUG_DEV_MueLu_ExtraTypes_EI	69	0	227	0	117	0	0	97	8 hours ago	(25 labels)
enigma.sandia.gov	Linux-GCC-4.8.3-SERIAL_DEBUG_DEV_MueLu_ExtraTypes	69	0	227	0	117	0	3	94	7 hours ago	(25 labels)
enigma.sandia.gov	Linux-GCC-4.8.3-SERIAL_RELEASE_DEV_MueLu_Experimental	69	0	227	0	113	0	4	107	6 hours ago	(25 labels)

Trilinos automated testing

85

- Several Amesos2 (direct solver) tests are broken.

SubProject Dependencies										
Project	Configure			Build			Test			Last submission
	Error	Warning	Pass	Error	Warning	Pass	Not Run	Fail	Pass	
Teuchos	0	22	22	0	13	9	0	0	227	2016-06-06 09:01:20
Epetra	0	22	22	0	13	9	0	1	244	2016-06-06 09:02:05
Triutils	0	22	22	0	0	21	0	0	2	2016-06-06 09:02:16
Tpetra	0	20	20	0	18	2	0	0	285	2016-06-06 08:10:13
EpetraExt	0	21	21	0	3	18	0	0	26	2016-06-06 08:11:16
ThreadPool	0	1	1	0	0	1				2016-06-06 02:51:44
Amesos	0	21	21	0	1	20	0	0	41	2016-06-06 08:16:59

- Are any of its dependencies broken?
 - Yes, there is a broken Epetra (basic linear algebra) test
 - Maybe this broke Amesos2

Trilinos automated testing

86

- Which tests were broken in Amesos2?

Testing started on 2016-06-06 07:42:35

Site Name:enigma.sandia.gov

Build Name:Linux-GCC-4.8.3-SERIAL_DEBUG_DEV_MueLu_ExtraTypes

Total time:16s 840ms

OS Name:Linux

OS Platform:x86_64

OS Release:3.10.0-229.4.2.el7.x86_64

OS Version:#1 SMP Fri Apr 24 15:26:38 EDT 2015

Compiler Version:unknown

3 tests failed.

Name	Status	Time	Details	Labels	Summary
Amesos2_Epetra_RowMatrix_Adapter_UnitTests_MPI_4	Failed	1s 860ms	Completed (Failed)	Amesos2	Broken
Amesos2_Epetra_MultiVector_Adapter_UnitTests_MPI_4	Failed	1s 980ms	Completed (Failed)	Amesos2	Broken
Amesos2_Tpetra_CrsMatrix_Adapter_UnitTests_MPI_4	Failed	1s 900ms	Completed (Failed)	Amesos2	Broken

Trilinos automated testing

87

□ If you may have broken something. you will get an



CDash <trilinos-regression@sandia.gov>

4:05 AM (5 hours ago) ☆



to anasazi-regres. ▾

A submission to CDash for the project Trilinos has failing tests.
You have been identified as one of the authors who have checked in changes that are part of this submission or you are listed in the default contact list.

Details on the submission can be found at <http://testing.sandia.gov/cdash/buildSummary.php?buildid=2469557>

Project: Trilinos

SubProject: Anasazi

Site: artemis.srn.sandia.gov

Build Name: Linux-intel-15.0.2-MPI_RELEASE_DEV_DownStream_ETI_SERIAL-OFF_OPENMP-ON_PTHREAD-OFF_CUDA-OFF_COMPLEX-OFF

Build Time: 2016-06-06T03:59:42 MDT

Type: Nightly

Tests failing: 1

Tests failing

Anasazi_Epetra_MVOPTester_MPI_4 (<http://testing.sandia.gov/cdash/testDetails.php?test=33891492&build=2469557>)

New master/develop workflow

88

- Want master branch to remain stable
- All developer changes are now pushed to develop branch
- If changes are “okay”, merge develop to master
 - ▣ Currently a manual process for Trilinos framework team
 - ▣ If no new tests are failing on the dashboard, merge
 - ▣ Will eventually be automated

An important consideration: commits are so frequent, and the test suite is so large, it is not possible to run the test suite after each commit.

Good testing practices

89

- ❑ Test-driven development – acceptance tests are written before the software
 - ▣ Gain clarity on code
 - ▣ Guarantees tests will exist
 - ▣ Useful when testing is viewed as unsustainable tax on resources
- ❑ Provide users a regression test suite
- ❑ Test software regularly, preferably daily

Building a test suite for CSE codes

90

Ideal time is *during development*

- When software is mature, must ensure new code does not break old features
 - ▣ Without regular testing, adding new code is error-prone
 - ▣ Structural changes are tedious without a way to verify ongoing correctness
 - ▣ Regular automated testing can provide a huge savings

New software vs legacy code

91

- Legacy code often has insufficient tests
 - ▣ First step in doing new, nontrivial development: add more tests
 - ▣ The issue: legacy code is not organized for unit tests

The key to working with legacy code is getting it to a place where it is possible to know that you are making changes *one at a time*.

- Michael Feathers, Working Effectively with Legacy Code

New software vs legacy code

92

- Test coverings
 - ▣ Set of tests used to introduce an invariant
 - ▣ Cover a small area of the system
 - Unit tests might not be possible, given legacy code organization
 - ▣ Correct behavior is defined by what the code did yesterday, not an external standard of correctness
 - If the original legacy code was incorrect, that's a separate issue
 - ▣ Build the invariant, then refactor to make the code clear

Development phase

93

- Necessary to devise tests that give confidence the software is behaving correctly
 - ▣ Analyze mathematical properties of numerical algorithm
 - Stability
 - Accuracy
 - ▣ Apply algorithm to physical situation, ensuring physical laws are not broken
 - ▣ Possible statistical or other analysis to evaluate solution quality

Formulating a continuous testing regime

94

- Identify verification needs within software
 - ▣ Defines code-coverage requirements
 - ▣ Pick features of the code necessary for correct behavior
 - Individual units
 - Interaction between units
 - ▣ Know the purpose of testing each feature
 - Reduces wasted effort and computing resources
 - Helps identify the most appropriate type of test
- Identify behaviors of code with detectable response to changes

Maintenance of a test suite

95

- Testing regime is only useful if it is
 - ▣ Maintained
 - ▣ Monitored regularly
 - ▣ Has rapid response to failure
- Maintenance includes
 - ▣ Updating tests and benchmarks
 - ▣ Adjustments to software stack
 - ▣ Archiving and retrieval of test suite output
 - Helpful in tracing change in code behavior

Maintenance of a test suite

96

- Monitoring individual tests manually is unreasonable and should be automated
 - ▣ Manual inspection should be limited to failing tests
 - ▣ For repository code, failure can be correlated to check-ins within a particular time-frame
 - Only certain developers need to be involved

Maintenance of a test suite

97

- Tests should pass most of the time
 - ▣ Easy when code changes are infrequent
 - ▣ Harder when code is large and rapidly changing
 - Difficult to determine cause of failure
 - Pre-commit test suites are a good idea
- Periodically review collection of tests
 - ▣ Look for gaps and redundancies
 - ▣ Pruning is important to conserve testing resources
 - ▣ Deprecated features can be removed
 - ▣ New tests may be necessary when new features are added

Example: SuperLU test suite

98

- SuperLU – sparse Gaussian elimination code
- Test suite
 - ▣ Many unit and integration level tests
 - ▣ Entire suite can be run in a few minutes
 - ▣ Demonstrates validation and acceptance testing, also no-change or bounded-change testing
 - ▣ Demonstrates how to deal with floating point issues

Example: SuperLU test suite

99

- Suite has two main goals
 - ▣ Tests query functions to floating-point parameters
 - Machine epsilon, underflow and overflow thresholds, etc
 - ▣ Provide coverage of all routines
 - Tests all functions of the user-callable routines

Example: SuperLU test suite

100

- Many input matrices are generated
 - ▣ Different numerical and structural properties
- Uses several numerical metrics to assert accuracy of solution
 - ▣ Stable LU factorization
 - ▣ Small forward and backward errors

Example: SuperLU test suite

101

- Performs exhaustive testing of a large number of input parameters

```
For each  $I_1 \in$  set of valid values {  
  For each  $I_2 \in$  set of valid values {  
    ...  
    For each  $I_q \in$  set of valid values {  
      For each matrix type {  
        Generate the input matrix A and rhs b;  
        Call a user-callable routine with input values  $\{I_1, I_2, \dots, I_q\}$ ;  
        Compute the test metrics;  
        Check whether each metric is smaller than a prescribed  
threshold;  
      }  
    }  
    ...  
  }  
}
```

- Runs over 10,000 tests in a few minutes

Use of test harnesses

102

- Essential for large code
 - ▣ Set up and run tests
 - ▣ Evaluate test results
- Easy to execute a logical subset of tests
 - ▣ Pre-push
 - ▣ Nightly
- Automation of test harness is critical for
 - ▣ Long-running test suites
 - ▣ Projects that support many platforms

Automated test harnesses

103

- crontab
 - ▣ Time-based scheduler for Linux
 - ▣ Execute specific command at specific time
- Newer tools...
 - ▣ Allow centralized servers to execute tests on multiple platforms
 - ▣ Assist in load balancing and scheduling on available test resources
 - ▣ Test execution can be triggered by
 - Time
 - An event (such as repository modification)
 - Manual request by developer

Reporting test results

104

- Output results to screen
 - ▣ Appropriate for pre-push testing
- Send email to a mail list
 - ▣ Can be generated by dashboard
- Test results dashboard
 - ▣ Can display results from a range of dates
 - ▣ Can detect changes in pass/fail conditions
 - ▣ Allows results to be sorted and searched
 - ▣ Enhances visibility of failing builds and tests

Example: hypre

105

- Multigrid preconditioner package
- Uses a system of scripts
 - ▣ Send emails to hypre developers indicating issues
 - ▣ Emails point to directories where issues occurred

Policies on testing practices

106

- Must have consistent policy on dealing with failed tests
 - ▣ Issue tracking
 - How quickly does it need to be fixed?
 - Who is responsible for fixing it?
 - ▣ Add regression test afterwards (to avoid reintroducing issue later)
- Someone needs to be in charge of watching the test suite

Policies on testing practices

107

- When refactoring or adding new features, run a regression suite before checkin
 - ▣ Be sure to add new regression tests for the new features
- Make sure at least two people are familiar with every portion of code
- Require a code review before releasing test suite
 - ▣ Another person may spot issues you didn't
 - ▣ Incredibly cost-effective

Policies on testing practices

108

- Avoid regression suites consisting of system-level no-change tests
 - ▣ Tests often need to be re-baselined
 - Often done without verification of new gold-standard
 - ▣ Hard to maintain across multiple platforms
 - ▣ Loose tolerances can allow subtle defects to appear