TECH-X

SIMULATIONS EMPOWERING
YOUR INNOVATIONS

Using Bilder to Build Trilin

T. Austin, S. Kruger, R. Pundaleeka

Goal: Get You to Using Trilinos Today

- Some people can be overwhelmed with complexity of build systems for scientific software.

- Building third-party libraries (MPI, lapack, SuperLU) and getting the dependencies correct can be a nightmare (especially true for Windows).

- Goal is to help you avoid nightmare step by making the build and install process for Trilinos as easy as possible.

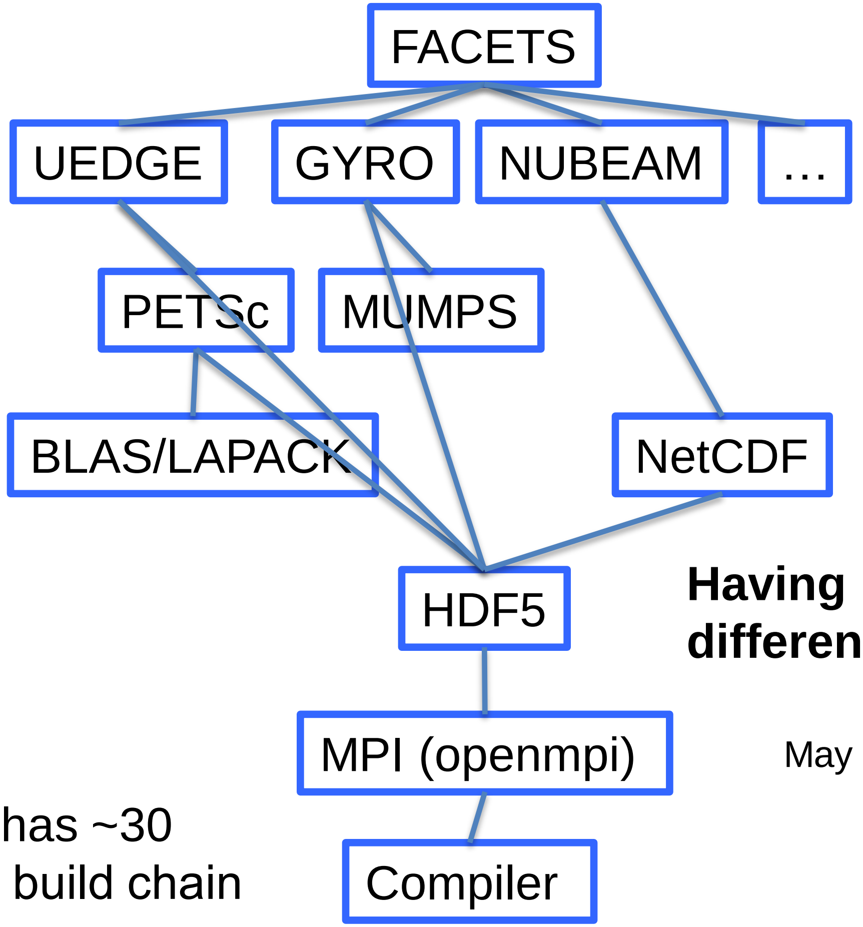
- Let Bilder do all of the work for you so you can solve real problems rather than getting tangled up in frustrating compiler and library issues.

What is Bilder?

A meta-build system
for scientific software

- It deals with code *packages*, not code source.
- It is geared to building *chains* of dependencies of arbitrary length.
- It is cross-platform with no compiler assumptions.
- It is hosted at sourceforge (http://sourceforge.net/p/bilder).
- It is for scientific software
  - Fortran is special
  - MPI is special
  - Handle the diamond-structure dependencies commonly found in scientific software
    - IO libraries (netcdf, HDF5) and math libraries (blas,lapack) are common dependencies
- It is **not** a package manager system like Gentoo’s portage or MacPorts.

Originally developed to solve problems with FACETS: code-coupling framework in the fusion community



Legacy fusion codes:
generally crappy, but contains lots of knowledge that we want to save

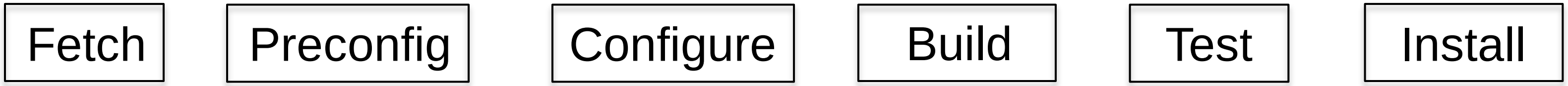
Having individual build systems find different HDF5 libraries is very bad!

May or may not be part of build chain

FACETS really has ~30 packages in it's build chain

What are the common features in building a package?

Bilder: Controls the step of building and installing individual packages



Fetch

Preconfig

Configure

Build

Test

Install

- **Fetch**: Tarball or use repo?
  - Tarballs come from “numpkgs” repo at Tech-X
- **Preconfig**: Do we need to patch for a special system?
- **Configure**: Install tarballs in one location and repos in another?
- **Build**: Do we have to do something special?
- **Test**: Is the build working properly?
- **Install**: Anything to do afterwards, like fix permissions?

Using Bilder to build Trilinos
Step 1: Setup

❑ Make sure you have your target machine ready:

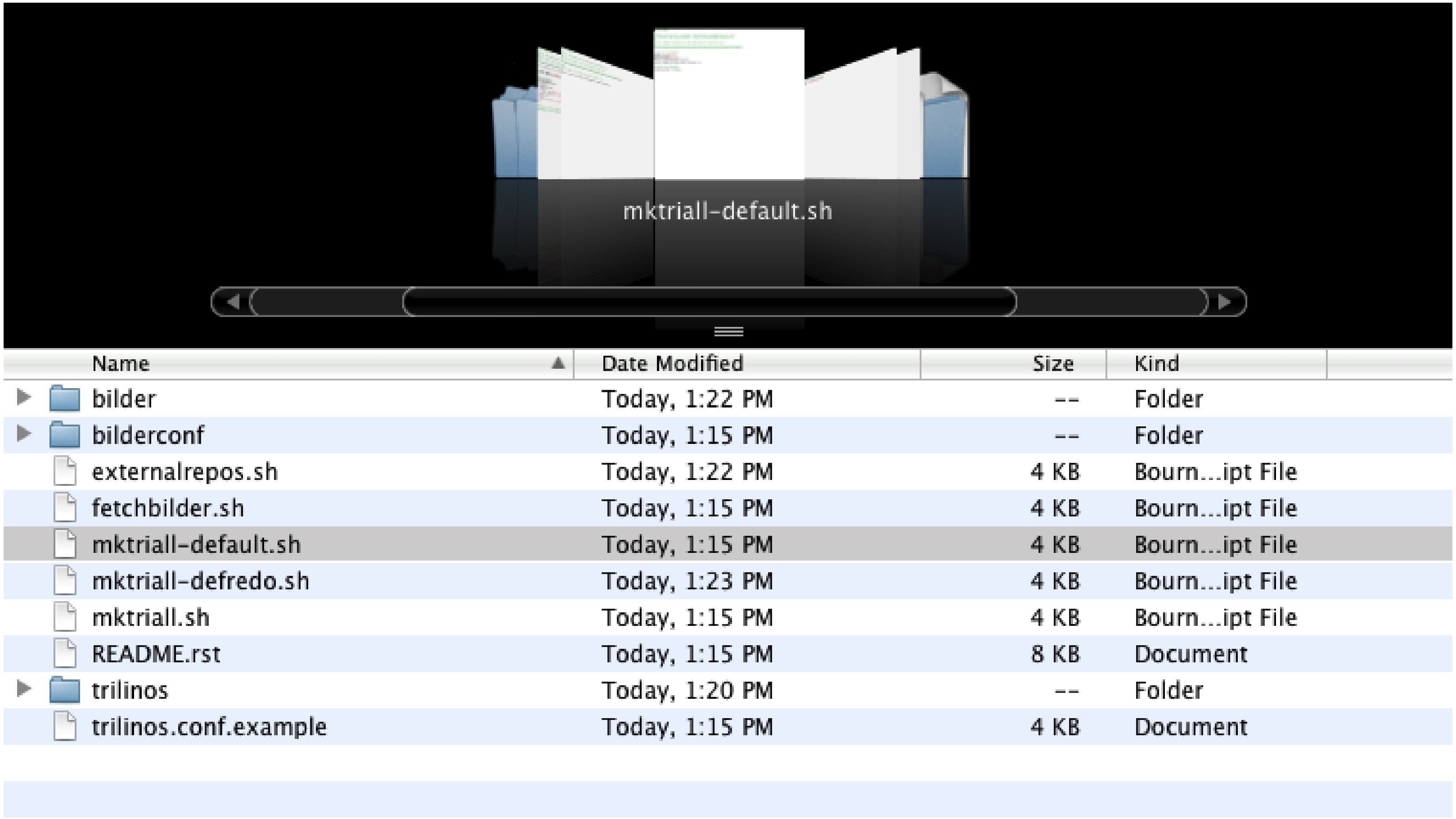
http://sourceforge.net/p/bilder/wiki/Preparing%20your%20machine%20for%20Bilder/

❑ Obtain an account on github (open to anyone):

❑ Make the following calls from the command line (bash shell):

```
% git clone https://USERNAME@github.com/Tech-XCorp/trilinosall.git trilinosall
% cd trilinosall
% ./externalrepos.sh
# Obtain a recent version (11.0.3) of Trilinos or get the repo from publicTrilinos
```

Using Bilder to build Trilinos
Step 1: Setup



Name	Date Modified	Size	Kind
bilder	Today, 1:22 PM	--	Folder
bilderconf	Today, 1:15 PM	--	Folder
externalrepos.sh	Today, 1:22 PM	4 KB	Bourn...ipt File
fetchbilder.sh	Today, 1:15 PM	4 KB	Bourn...ipt File
mktriall-default.sh	Today, 1:15 PM	4 KB	Bourn...ipt File
mktriall-defredo.sh	Today, 1:23 PM	4 KB	Bourn...ipt File
mktriall.sh	Today, 1:15 PM	4 KB	Bourn...ipt File
README.rst	Today, 1:15 PM	8 KB	Document
trilinos	Today, 1:20 PM	--	Folder
trilinos.conf.example	Today, 1:15 PM	4 KB	Document

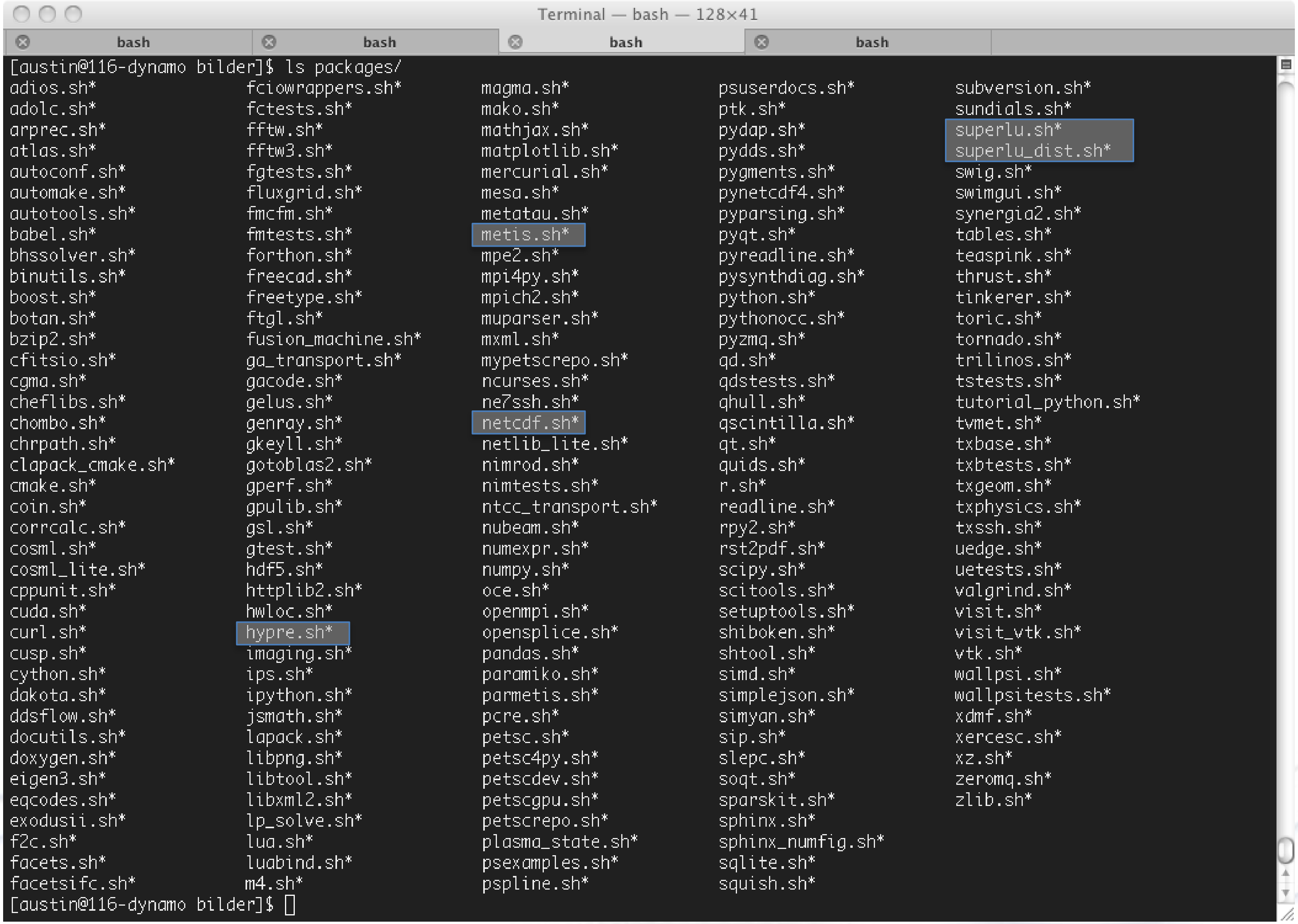
Note: rst2html.py README.rst > README.html generates HTML instructions as well. See file:///Users/austin/Projects/Trilinos/trilinosall/README.html.

Packages relevant to Trilinos

Name	Version	Windows
HDF5	1.8.7-9	Yes
Qt	4.8.1	Yes
Thrust	1.6.0	Yes
Zlib	1.2.6	Yes
PETSc	3.2 or 3.3	Yes
Dakota	5.2	?
Boost/Boostlib	1_47_0 (1_50_0)	Yes
netcdf	4.1.12	Yes
SuperLU	4.1	Yes
SuperLUDist	3.2	Yes
METIS		Yes
HYPRE		Yes

To see all packages supported: `ls bilder/packages`

Packages relevant to Trilinos



```
[austin@116-dynamo bilder]$ ls packages/
adios.sh*               fciowrappers.sh*        magma.sh*               psuserdocs.sh*          subversion.sh*
adolc.sh*               fctests.sh*             mako.sh*                ptk.sh*                 sundials.sh*
arprec.sh*              fftw.sh*                mathjax.sh*             pydap.sh*               superlu.sh*
atlas.sh*               fftw3.sh*               matplotlib.sh*          pydds.sh*               superlu_dist.sh*
autoconf.sh*            fgtests.sh*             mercurial.sh*           pygments.sh*            swig.sh*
automake.sh*            fluxgrid.sh*            mesa.sh*                pynetcdf4.sh*           swimgui.sh*
autotools.sh*           fmcfm.sh*               metatau.sh*             pyparsing.sh*           synergia2.sh*
babel.sh*               fmtests.sh*             metis.sh*               pyqt.sh*                tables.sh*
bhssolver.sh*           forthon.sh*             mpe2.sh*                pyreadline.sh*          teaspink.sh*
binutils.sh*            freecad.sh*             mpi4py.sh*              pysynthdiag.sh*         thrust.sh*
boost.sh*               freetype.sh*            mpich2.sh*              python.sh*              tinkerer.sh*
botan.sh*               ftgl.sh*                muparser.sh*            pythonocc.sh*           toric.sh*
bzip2.sh*               fusion_machine.sh*      mxml.sh*                pyzmq.sh*               tornado.sh*
cfitsio.sh*             ga_transport.sh*        mypetscrepo.sh*         qd.sh*                  trilinos.sh*
cgma.sh*                gacode.sh*              ncurses.sh*             qdstests.sh*            tstests.sh*
cheflibs.sh*            gelus.sh*               ne7ssh.sh*              qhull.sh*               tutorial_python.sh*
chombo.sh*              genray.sh*              netcdf.sh*              qscintilla.sh*          tvmet.sh*
chrpath.sh*             gkeyll.sh*              netlib_lite.sh*         qt.sh*                  txbase.sh*
clapack_cmake.sh*       gotoblas2.sh*           nimrod.sh*              quids.sh*               txbtests.sh*
cmake.sh*               gperf.sh*               nimtests.sh*            r.sh*                   txgeom.sh*
coin.sh*                gpulib.sh*              ntcc_transport.sh*      readline.sh*            txphysics.sh*
corrcalc.sh*            gsl.sh*                 nubeam.sh*              rpy2.sh*                txssh.sh*
cosml.sh*               gtest.sh*               numexpr.sh*             rst2pdf.sh*             uedge.sh*
cosml_lite.sh*          hdf5.sh*                numpy.sh*               scipy.sh*               uetests.sh*
cppunit.sh*             httplib2.sh*            oce.sh*                 scitools.sh*            valgrind.sh*
cuda.sh*                hwloc.sh*               openmpi.sh*             setuptools.sh*          visit.sh*
curl.sh*                hypre.sh*               opensplice.sh*          shiboken.sh*            visit_vtk.sh*
cusp.sh*                imaging.sh*             pandas.sh*              shtool.sh*              vtk.sh*
cython.sh*              ips.sh*                 paramiko.sh*            simd.sh*                wallpsi.sh*
dakota.sh*              ipython.sh*             parmetis.sh*            simplejson.sh*          wallpsitests.sh*
ddsflow.sh*             jsmath.sh*              pcre.sh*                simyan.sh*              xdmf.sh*
docutils.sh*            lapack.sh*              petsc.sh*               sip.sh*                 xercesc.sh*
doxygen.sh*             libpng.sh*              petsc4py.sh*            slepc.sh*               xz.sh*
eigen3.sh*              libtool.sh*             petscdev.sh*            soqt.sh*                zeromq.sh*
eqcodes.sh*             libxml2.sh*             petscgpu.sh*            sparskit.sh*            zlib.sh*
exodusii.sh*            lp_solve.sh*            petscrepo.sh*           sphinx.sh*
f2c.sh*                 lua.sh*                 plasma_state.sh*        sphinx_numfig.sh*
facets.sh*              luabind.sh*             psexamples.sh*          sqlite.sh*
facetsifc.sh*           m4.sh*                  pspline.sh*             squish.sh*
[austin@116-dynamo bilder]$
```

Packages relevant to Trilinos

superlu.sh

```
#!/bin/bash
######################################################################
# Version
######################################################################

SUPERLU_BLDRVERSION=${SUPERLU_BLDRVERSION:-"4.1"}

######################################################################
# Other values
######################################################################
if test -z "$SUPERLU_BUILDS"; then
  SUPERLU_BUILDS=ser,sersh
fi
SUPERLU_DEPS=cmake,atlas,lapack,clapack_cmake
SUPERLU_UMASK=002

######################################################################
# Launch superlu builds.
######################################################################
buildSuperlu() {
  if bilderUnpack superlu; then
    if bilderConfig -c superlu ser; then
      bilderBuild superlu ser
    fi
    if bilderConfig superlu sersh “-DBUILD_SHARED_LIBS:BOOL=ON" ; then
      bilderBuild superlu sersh
    fi
  fi
}
.
.
```

Using Bilder to build Trilinos
Step 2: Invoking Bilder

The two main scripts are:

- ❑ mktriall.sh

  Main bilder script that fine-tunes many of the build aspects.

- ❑ mktriall-default.sh

  Bilder script for handling default parameters for simplifying the builds, including the default locations at LCFs.

❑ For both scripts, ``-h`` or ``--help`` commands will show options.

❑ To build trilinos with all the default builds and third party dependencies, first *print* what the default will do::

```
./mktriall-default.sh –p
```

```
source /Users/austin/Projects/Trilinos/trilinosall/bilder/runnr/runnrfcns.sh
Command is
./mktriall.sh -k /Users/austin/software -i /Users/austin/software –e austin@txcorp.com
runBilderCmd exiting with 0.
```

Understanding Bilder output: Terminology

- PROJECT_DIR

  This is the directory location of this file.
- INSTALL_DIR

  This is where trilinos will be installed (./mktriall.sh –i INSTALL_DIR)
- CONTRIB_DIR

  This is where TPLs from tarballs will be installed (-k CONTRIB_DIR)

  This may equal the INSTALL_DIR
- BUILD_DIR

  This is where the builds are location (-b BUILD_DIR)

  Typically ``$PROJECT_DIR/builds``

For example, we have by default trilinosall/builds where we would see SuperLU and SuperLU_Dist builds.

Typically use ~/Software as INSTALL_DIR and CONTRIB_DIR.

Understanding Bilder output: Key files

- Key output files:

  $BUILD_DIR/mktriall.log

  $BUILD_DIR/mktriall-summary.txt

  $BUILD_DIR/trilinos-chain.txt

- For each package (e.g., trilinos)

  `$BUILD_DIR/trilinos/<build>/<hostname>-<pkg>-<build>-<step>.txt`

  - E.g., `$BUILD_DIR/trilinos/ser/iter.txcorp.com-trilinos-ser-build.txt`
  - To debug, it is help to use the scripts that generated the build:

What is wrong? `cd $BUILD_DIR/trilinos/ser`

Can I fix? `cat iter.txcorp.com-trilinos-ser-build.txt`

Did it work? `vi iter.txcorp.com-trilinos-ser-build.sh`

`iter.txcorp.com-trilinos-ser-build.sh`

Customizing Trilinos builds

- To set up necessary builds and third party dependencies, create a configuration file called ``trilinos.conf`` in $PROJECT_DIR
  - `cp trilinos.conf.example trilinos.conf`
- Key variables:
  - TRILINOS_BUILDS

    Which types of builds do. Possible choices are ser,par,sersh,parsh

    where the sh suffice refers to shared builds
  - TRILINOS_DEPS

    To turn on and off TPL dependencies.
    Needs to be coordinated with TRILINOS_ADDL_ALLARGS potentially
  - TRILINOS_ADDL_ALLARGS

    Arguments used by all builds.
    Generally used to turn on and off trilinos packages and TPL.
  - TRILINOS_<BUILD>_OTHER_ARGS

    Arguments for the individual builds.

Sample trilinos.conf

```
TRILINOS_BUILDS="ser,par”
TRILINOS_DEPS="swig,openmpi,boost,hdf5”
TRILINOS_ADDL_SHARGS="-DTrilinos_ENABLE_Amesos:BOOL=ON"
```

Building other packages

- Bilder has other packages that you may want to build.
- mktriall.sh can take as an argument a different package
- For example, ipython has a pretty long build chain that includes almost all useful scientific python packages

```
mktriall-default.sh –n – ipython
```

will build the ipython build chain in the default locations

Conclusions and further work

- Bilder is a useful tool for building dependency chains on different platforms
- We have “bilderized” trilinos to make it easier for people to build the trilinos build chain
- Customizing your build to choose your dependencies is possible with the trilinos.conf file
- Bilder documents all the steps thoroughly to allow debugging of any problems that arise.
  - Any problems can be sent to developer@txcorp.com
- We welcome feedback and suggestions for improvements