Tutorial Plan

- Review setting up and invoking Bilder for trilinosall.
- Perform a basic configure for a serial program
- Build the serial trilinos with minimal dependencies
- Perform a basic configure for a parallel program
- Customize a particular trilinos version
- Some pretty useful options
- Conclusions
Using Bilder to build Trilinos
Step 1: Setup

- Make sure you have your target machine ready:
  
  [Link to instructions: http://sourceforge.net/p/bilder/wiki/Preparing%20your%20machine%20for%20Bilder/]

- Start with the following commands:
  
  ```
  % git clone https://USERNAME@github.com/Tech-XCorp/trilinosall.git trilinosall
  % cd trilinosall
  % ./externalrepos.sh  # Sets up bilder subdirectory and trilinos subdirectory
  ```

Go to Terminal Window to Download Necessary Repos
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

- If the command looks acceptable, then start up the build process running in the background using *nohup*:
  ./mktriall-default.sh -n -e austin@txcorp.com
Seeing Bilder in Action for Default Serial Builds

% TRILINOS_BUILDS=ser ./mktriall.sh -c

Go to trilinosall-serconf:

- Look at mktriall-summary.txt
- Look at mktriall.log
- Look at trilinos-chain.txt
- Look in numpkgs
- Look at individual build directories.
Seeing Bilder in Action for Default Serial Builds

% TRILINOS_BUILDS=ser ./mktriall.sh –j 2 –i ~/Internal –k ~/Contrib

Go to trilinosall-serbuild and ~/Internal and ~/Contrib:

- Look at what happened in $PROJECT_DIR/build
- Look at ~/Internal and ~/Contrib
- Look at installations.txt
Seeing Bilder in Action for Default Parallel Builds

```
% TRILINOS_BUILDS=par ./mktriall.sh -c
```

Go to trilinosall-parconf:

- Look at mktriall-summary.txt
- Look at mktriall.log
- Look at trilinos-chain.txt
- Look in numpkgs.
- Look at individual build directories.
Seeing Bilder in Action for Default Serial and Parallel Builds

```bash
% TRILINOS_BLDRVERSION=ser ./mktriall.sh -c

Go to trilinosall-serconf

% TRILINOS_BLDRVERSION=ser ./mktriall.sh -j 2 -i ~/Include -k ~/Contrib

Go to trilinosall-serbuild

% TRILINOS_BLDRVERSION=par ./mktriall.sh -c

Go to trilinosall-parconf

% TRILINOS_BLDRVERSION=par ./mktriall.sh -j 2 -i ~/Include -k ~/Contrib

Go to trilinosall-parbuild
```
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` suffix 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.
Customizing trilinos ser build

```bash
###
## Available builds: ser, par, sersh, parsh
#
TRILINOS_BUILDS="ser"
###
## To turn off the dependencies
#
TRILINOS_DEPS="swig, openmpi, boost, hdf5"
###
## Arguments for all static builds
#
TRILINOS_ADDL_ARGS="-DTrilinos_ENABLE_Epetra:BOOL=ON"
TRILINOS_ADDL_ARGS="${TRILINOS_ADDL_ARGS} -DTrilinos_ENABLE_ML:BOOL=ON"
TRILINOS_ADDL_ARGS="${TRILINOS_ADDL_ARGS} -DTrilinos_ENABLE_AztecOO:BOOL=ON"
TRILINOS_ADDL_ARGS="${TRILINOS_ADDL_ARGS} -DTPL_Boost_INCLUDE_DIRS:FILEPATH=$CONTRIB_DIR/boost-${BOOST_BLDRVERSION}/include"
#
# Also TRILINOS_ADDL_SHARGS, TRILINOS_SER_OTHER_ARGS, TRILINOS_SER_OTHER_SHARGS, …
#
```
Some pretty useful options

Specifying the Machine Type

% ./mktriall.sh  –m cygwin.vs9 # Windows-Cygwin for Visual Studio 9
   –m cygwin.vs11 # Windows-Cygwin for Visual Studio 11
   –m bgp.xlc     # Blue Gene/P with xlc compiler
   –m kraken.cray.gnu # Kraken at NICS

Specifying the Builds to Disable

% ./mktriall.sh  –p MY_PATH   # Specify the SUPRA_SEARCH_PATH
                  –A ADDED_PATH # Add this path to SUPRA_SEARCH_PATH

Specifying the Builds to Disable

% ./mktriall.sh  –W lapack,cmake  # Turn off lapack and cmake

Often Bilder finds the right machine type (e.g, Darwin, Cygwin)
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

- Email me questions at austin@txcorp.com or developer@txcorp.com.

- We can create a specialized machine file with compiler for you.

- Let us know if there are any other options that would be useful.