

Progress with Bilder and Documentation

How we build and document FACETS

S. Vadlamani S. Kruger A. Hakim J. Cary

Feb. 24, 2011/Facets Winter Meeting

Outline

- 1 Building
 - What is Bilder
 - Building Caveats
- 2 Documenting
 - Defining documentation
 - Using Sphinx
 - Trac sites

Outline

- 1 Building
 - What is Bilder
 - Building Caveats
- 2 Documenting
 - Defining documentation
 - Using Sphinx
 - Trac sites

Bilder wants to simplify building on most platforms.
Can't we all just get along?

Bilder wants to simplify building on most platforms. Can't we all just get along?

- Building complicated, interdependent packages on multiple platforms, consistently, is important.
 - We all want to run FACETS on our favorite machine.

Bilder wants to simplify building on most platforms. Can't we all just get along?

- Building complicated, interdependent packages on multiple platforms, consistently, is important.
 - We all want to run FACETS on our favorite machine.
- Bilder does a great job of managing packages used in the FACETS tool chain.

Bilder wants to simplify building on most platforms. Can't we all just get along?

- Building complicated, interdependent packages on multiple platforms, consistently, is important.
 - We all want to run FACETS on our favorite machine.
- Bilder does a great job of managing packages used in the FACETS tool chain.
 - Bilder helps configure packages and many platforms with many compilers.

Bilder wants to simplify building on most platforms. Can't we all just get along?

- Building complicated, interdependent packages on multiple platforms, consistently, is important.
 - We all want to run FACETS on our favorite machine.
- Bilder does a great job of managing packages used in the FACETS tool chain.
 - Bilder helps configure packages and many platforms with many compilers.
 - It also checks if a packages should be rebuilt because a dependency was rebuilt.

Bilder wants to simplify building on most platforms. Can't we all just get along?

- Building complicated, interdependent packages on multiple platforms, consistently, is important.
 - We all want to run FACETS on our favorite machine.
- Bilder does a great job of managing packages used in the FACETS tool chain.
 - Bilder helps configure packages and many platforms with many compilers.
 - It also checks if a packages should be rebuilt because a dependency was rebuilt.
 - We have made an attempt at standardizing the build and install of packages on a wide variety of machines

Bilder wants to simplify building on most platforms. Can't we all just get along?

- Building complicated, interdependent packages on multiple platforms, consistently, is important.
 - We all want to run FACETS on our favorite machine.
- Bilder does a great job of managing packages used in the FACETS tool chain.
 - Bilder helps configure packages and many platforms with many compilers.
 - It also checks if a packages should be rebuilt because a dependency was rebuilt.
 - We have made an attempt at standardizing the build and install of packages on a wide variety of machines
 - The scripts written using Bilder functionality have many optional flags that offer more flexibility.

The FACETS world is accessible from one repository.

Only need the facetsall repo

```
svn co https://ice.txcorp.com/code/facetsall/trunk facetsall
```

- Tarballs will not be downloaded unless needed.
 - These include petsc, hdf5, openmpi, etc.
 - The tarballs will be downloaded in a directory called *numpkgs*.

Building is now very simple.

We want ease of build on the LCFs and on your local machine.

```
./mkfcall-default.sh
```

```
cd facetsall
```

```
./mkfcall-default.sh -f <extra args file> -w <nobuild file> -n
```

- “-n” nohups and will send output to declared file
- “-f” wants a file that holds comma delimited arguments found by doing `./mkfcall.sh -h`.
- “-w” wants a file that holds comma delimited packages **not to build**.
- “-h” shows options
- The default install is in `$HOME/software` and the builds are in `*/facetsall/builds` directory.

“Options” is freedom!

My favorite extra args in *extraArgs.txt*

```
-u -D -F -e email@this.place
```

- “-u”: Do an svn up on facetsall
- “-D”: build documentation
- “-F”: install packages with local modifications
- “-e” comma delimited list of email addresses to send results of this build/install process
- You can find these options by doing `./mkfcall.sh -h`.

Customizing builds is quite simple

Leave off the toric, fmcfm builds (*nobuilds.txt*)

```
toric,fmcfm
```

More options for `./mkfcall-default.sh`:

- “-m” specifies a machine (cray.pgi, stix.pppl.gov.gnu, darwin9)
- “-l” Install in /Users/\$HOSTNAME
 - default: project directory on LCFs
- “-k” Look for stored tarball installations in /contrib
 - on LCFs install tarballs and use them
- “-t” Run tests
 - limit install to successfully tested packages
- “-p”: print out the real command to be executed
 - This is your “**test my command**” flag.

Outline

- 1 Building
 - What is Bilder
 - Building Caveats
- 2 Documenting
 - Defining documentation
 - Using Sphinx
 - Trac sites

We need input from users.

- It will help for all to start using the `./mkfcall-default.sh` script.
 - Always mail the developers list with any concerns and issues.
- Look in `bilder/machines` to get an idea of what is needed to handle difficult machines.
 - We will help get it building on your machine.
 - Sending the output from “env” always helps.
- Try this on LCFs to help sort out any issues.

Outline

- 1 Building
 - What is Bilder
 - Building Caveats
- 2 Documenting
 - Defining documentation
 - Using Sphinx
 - Trac sites

Documentation helps people use FACETS.

- There are two primary groups that we should inform about FACETS.
 - *Users* need their own manual.
 - This is more of a “how to use” guide.
 - If will be helpful if it filled with examples.
 - *Developers* want more of a technical manual.
 - They love API documentation.
 - Maybe even like dealing more details about the infrastructure.
 - This is a great place for documenting discretization schemes, algorithms and other technical details.
- Some information should appear in both categories, such as the equations we are solving.
- Passing a `-D` to the build script (`./mkfcall.sh`) will build the documentation of packages.
 - These are built in `builds/<package>/webdocs`
 - All packages' documentation will eventually install in

Convenient locations for documentation helps people use it.

- <http://ice.txcorp.com/doc/facets/> is the url built nightly on iter (and rsynced).
- We write documentation in *facetsall/facets/docs*.
 - There are two directories for the two different manuals: userdocs and develdocs.
- We will be cleaning these up to establish a good standard/structure for documentation.

Outline

- 1 Building
 - What is Bilder
 - Building Caveats
- 2 Documenting
 - Defining documentation
 - Using Sphinx
 - Trac sites

Documenting should be fun.

- **Sphinx** is a Python based documenting package,
 - Sphinx uses the **reStructuredText** markup language.
 - **Example** of FACETS developers documentation .
 - Click on the rhs “Show source” button to see the rST syntax.

We want pretty math.

- **Mathjax** is an open source JavaScript display engine for mathematics that works in all browsers.
 - Here is an **example** detailing transport equations.
 - Math is written in LaTeX.
 - A *facetsall* build will provide a local Mathjax installation.

Outline

- 1 Building
 - What is Bilder
 - Building Caveats
- 2 Documenting
 - Defining documentation
 - Using Sphinx
 - Trac sites

Wikis should be updated often

We have many wikis that should offer the latest information.

- The Trac site : <http://ice.txcorp.com/trac>
- You must use svn login/password to update wikis.

Summary

- Use `./mkfcall-default.sh` to build FACETS.
 - It gets better as more people try and post successes and challenges on mailing lists.
- Developers should find Sphinx usable and it may encourage those to write documentation.
- Updating the trac sites will help a border range of users.