Common development tasks

Unknown macro: 'html'

Unknown macro: 'html'

Introduction

This sections lists several typical tasks that users and developers will perform frequently. It implies that the SIT environment has already been setup as explained in Environment setup.

Our standard development machine is `psdev`, all commands below are expected to work from this machine. Analysis farm machines should not be used for development.

Currently our repository is setup on AFS, to access it we use ssh tunnel to `yakut` machines. Before running SVN commands make sure that you can connect from `psdev` to `yakut` with ssh – try to run `ssh yakut`. To avoid password prompts from svn commands you may need to run `kinit` on `psdev` to obtain AFS token. SVN repository is accessible to accounts in a special AFS group (g-lusi), if you are not a member of this group ask Andy or Igor to add you account to this group.

Tasks

Using different release versions

User release directories involve a mix of packages from the user and system. scons will make sure the version associated with each is the same. Steps to upgrade a user release are below. When just using the system release - for example running psana from the home directory, you can easily switch the release version as follows:

To switch to "newest" release:

> *sit_setup ana-current*

To switch to a numbered release and use debug build:

> *sit_setup 1.2.3 dbg*

Several environment variables that start with SIT_ will be changed by the above commands.

Switching to a different release or build options

For a user release directory, you may want to switch build options, or upgrade the release directory.

To use debug build

```
sit_setup dbg
```

Switch to other release permanently

```
relupgrade <new-release-name>
sit_setup
scons -c
scons
```

Creating test release based on some numbered release

All available releases can be found in the directory `SIT_RELDIR`

```
cd <test-area>
newrel 1.2.3 test-1.2.3
cd test-1.2.3
sit_setup
```

Check out package from repository

To check out package HEAD

```
addpkg MyPackage HEAD
```

To check out the same tag as in the release

```
addpkg MyPackage
```

To check out specific tag

```
addpkg MyPackage V00-00-00
```

Check the status of the files in package

Run this command often to see if you forgot to add any local files to repository

```
svn status MyPackage
```

this command shows the local status of the package with respect to the last `svn commit` (check out) command.

```
svn status -u (or --show-updates)
```

shows files which will be updated if you run `svn update` command.
See more info in svnbook.

Check tags of local packages

The tags of packages located in the release directory can be listed by the command

```
relinfo
```

The tags of packages of any release can be listed by the command

```
relinfo <release-name>
```

How to see which new tags were added on top of the specified release (for example ana-0.13.3)

```
kinit

relinfo -n -P -f /reg/g/psdm/sw/releases/buildbot/tags/tags-ana REPO > latest-tags

diff latest-tags /reg/g/psdm/sw/releases/buildbot/tags/ana-0.13.3
```

Create completely new package

This will create basic structure for a regular package - package directory, `SConscript` file, `doc/README`, and `doc/ChangeLog`.

```
newpkg MyNewPackage
```

This command does not change anything in the repository, it only creates local directory and files.

Create new package in Subversion

```
psvn newpkg MyNewPackage
```

and check it out

```
addpkg MyNewPackage
```

Remove package from release directory

In order to get rid of traces of the package in the release directory, all binary files need to be cleaned by the command `scons -c` before removing the package. For example,

```
cd <release-directory>
sit_setup
scons -c
rm -rf <package-name>
```

Add files to repository

```
svn add <file-or-dir> ...
```

If the argument is a directory then all files in that directory will be added too.

Display modifications to a package

Display all local modification in a working copy

```
svn diff <package>
```

Display all local modification to a particular file

```
svn diff <package>/<file>
```

Display diffs between local copy and the HEAD of the package in repository

```
svn diff -r HEAD <package>
```

Display changes happened in repository between time when the package was checked out and now:

```
svn diff -r BASE:HEAD <package or path to module>
```

where

- BASE is the revision of the package when it was checked out

- HEAD is the latest revision of the same directory in SVN

For example

```
svn diff -r 7810:HEAD ParametersV1.h
```

To see ALL modifications made to a <package-name> since the specified SVN revision (ex.: 8968)

```
svn diff -r 8968:HEAD https://pswww.slac.stanford.edu/svn-readonly/psdmrepo/<package-name>/trunk
```

For more complex cases consult SVN documentation.

Building the release

```
scons
```

or

```
scons TRACE=1
```

Use higher TRACE numbers for verbose output.

To compile all unit tests in the release

```
scons test
```

or to build test for particular package:

```
scons test <package>
```

Remove or rename files

```
svn rm <filename>
```

```
svn mv <filename> <new-filename>
```

Committing changes to a package

```
svn status <package>
svn commit -m "Log message for this commit" <package>
```

List existing tags for a package

```
cd <package>
psvn tags
```

or

```
psvn tags <package>
```

Creating new tag for a package

Before you create new tag run 'svn update' command:

```
```

```
cd <package>
svn update
psvn tag V01-02-03
```

or

```
svn update <package>
psvn tag -p <package> V01-02-03
```

Problems can arise if your working directory is checked out from a tagged release in the repository (you cannot make changes into something that is tagged - it has been finalized). If this is the case, do the following

```
svn info #this prints the URL you have checked the package out from, is /tags/ in the path?
svn switch <path to to trunk location for the package>
svn update # get the latest from trunk
svn ci ... # commit any local changes you wish to trunk, update doc/ChangeLog in your commit
psvn tag V01-02-03
```

If the psvn tag command doesn't work, it may be because your are having svn copy code from the working directory rather than from a revision of trunk into a tag. The command

```
psvn tag V00-28-08
```

is equivalent to

```
svn copy . $SIT_SVN/package/tags/V00-28-08
```

which probably does not do exactly what you intend. TO fix this you need to copy trunk to tags. This can be done in couple of ways:

1. Find revision of the files that you committed (appears in output of svn commit, say it is 6404), then use it with psvn tag:

```
psvn tag -e 6404 V00-28-08
```

2. Use svn copy:

```
svn copy $SIT_SVN/package/trunk $SIT_SVN/package/tags/V00-28-08
```

Package sub-directory names convention

Command

```
newpkg MyNewPackage
```

creates the sub-directory tree with a minimal set of directories/files in it. For most projects the list of sub-directories need to be extended. For example, **for C++ projects** it is convenient to add sub-directories with pre-defined names:

```
cd <your-release-directory>
mkdir <package>/include
mkdir <package>/src
mkdir <package>/app
mkdir <package>/test
```

for C++ header (**.h), source (**.cpp), application (**.cpp), and test-application (**.cpp) files, respectively.
By default all files in `src` with `include` will be compiled and put in the package library. The files in `app` will be also compiled and moved in the `<release>/arch/<architecture>/bin` - default release `bin` directory. In order to compile and run modules from `test` directory use commands:

```
scons test
```

or

```
scons test <package>
```

then, to run:

> *build/<architecture>/<package>/<test-module>*

These modules will not be moved to the release `bin` directory.

Updating svn property variables

Every module under svn control of may heve/use associated property variables, which may be updated at `svn commit` command. The default property variables are listed in table:

Variable	Example
$Revision$	$Revision: 8146 $
Id	$Id: <module-name> 8146 2014-05-05 16:33:57Z <login-name>@SLAC.STANFORD.EDU $
$Author$	$Author: <login-name>@SLAC.STANFORD.EDU $
$HeadURL$	$HeadURL: https://pswww.slac.stanford.edu/svn/psdmrepo/<package-name>/trunk/src/<module-name>$
$Header$	
$LastChangedDate$	$LastChangedDate: 2014-05-05 09:33:57 -0700 (Mon, 05 May 2014) $
$Date$	$Date: 2014-05-05 09:33:57 -0700 (Mon, 05 May 2014) $

Usage of these variables should be allowed in `~/.subversion/config` file,

To update all keywards in the module before commiting use command:

> *psvn mktxtprop <path-to-modeule(s)>*

which is a wrapper on `svn propset` command. To update revision only, use command:

> *svn propset svn:keywords "Revision" <path-to-modeule(s)>*

Adding package in release build

For RPM-based build - edit the file **ana-tags** in the directory /reg/g/psdm/sw/releases/buildbot/tags/

For conda-based build - update the file psana-conda-svn-pkgs in the directory /reg/g/psdm/sw/conda/manage/config/

Examples

Create a new package

This example shows how to (1) create a new release, (2) to create new package, (3) add directories and modules, (4) commit to svn, and (5) set a tag.

```
ssh psdev
cd <directory-with-your-favorite-releases>
newrel ana-current <your-release-directory>
cd <your-release-directory>
sit_setup

# create local package
newpkg <new-package>

mkdir <new-package>/src
codegen -l pyana-module <new-package> <module1>
   or
cd <new-package>
mkdir <new-directory>
cp <path>/<module2> <new-directory>/<module2>
...
edit <module1> <module2>
...

# create package in SVN and check it out
psvn newpkg <new-package>
addpkg <new-package>

# add files to repository
svn add <new-directory>
   or
svn add <new-directory>/<module1>
svn add <new-directory>/<module2>

svn commit -m "Log message for this commit" <new-package>
psvn -p <new-package> tag V01-00-01
```

Edit existing package

```
ssh psdev
cd <directory-with-your-favorite-releases>
newrel ana-current <your-release-directory>  # or use already existing release directory
cd <your-release-directory>

sit_setup            # To use the same release
   or
sit_setup newest     # To switch to "newest" release
   or
sit_setup 1.2.3 dbg  # To switch to a numbered release and use debug build

addpkg <existing-package> HEAD       # To check out package HEAD (latest version)
cd <existing-package>
svn update

   [Edit, add remove modules and directories, build ({{scons}}), and test applications, etc...]

svn commit -m "Log message for this commit"
svn update                           # To account for possible modifications from other developers
psvn tag V00-00-06                   # Incremented package version tag
```