Packaging your software for the Life Science Grid

Pieter van Beek
SARA Computing and Networking Services
High Performance Computing and Visualization
e-Science Support

How software is commonly installed...

```
%> tar zxf great_software.tgz
%> cd great_software-0.3/
%> cat README INSTALL HELP
   Hello mindless drones! As root, run:
   $ ./configure
   $ make
   $ make install
   This will install everything in /usr/local/.
   Namasté and good luck!
%>
```

- **(Un)fortunately, you can't do this on the Grid.**

Your GLite Grid Environment

- On a WorkerNode (WN), there are 3 or 4 locations where you can install software:
  - `/tmp` (or whatever `$TMP` points at)
    NEVER DO THIS!
  - `./` (AKA `$PWD` or `` `pwd` ``, your “sandbox directory”)
  - `$HOME` (probably the same as `./`)
  - `$TMPDIR` (the preferred location!)
- None of these have fixed paths.
  - Software that depends on absolute paths MUST be patched.
- Nothing/nobody looks in these directories by default.
  - bash -> PATH
  - ld-linux -> LD_LIBRARY_PATH
  - perl -> PERL5LIB
  - java -> CLASSPATH

Solution in two parts:

- PART I: Make sure your software can run
  - from anywhere on the filesystem
  - or, at least, from $TMPDIR/some_package/
- PART II: Adjust environment variables:
  - by sourcing a bash-script that's part of the package:
    source $TMPDIR/some_package/setenv.sh
  - by using the Modules system

Compiling software that uses autoconf/automake

- Compile the package with a prefix:

```
%> export TMPDIR=`mktemp -d`
%> ./configure --help
%> ./configure --prefix="$TMPDIR/some_package/"
%> make && make install
```

- Many packages depend on the PATH variable:

```
%> export PATH="$TMPDIR/some_package/bin:$PATH"
```

- If the binaries complain about missing libraries, also set the LD_LIBRARY_PATH variable:

```
%> export LD_LIBRARY_PATH="$TMPDIR/some_package/lib"
```

- For dynamic scripting language extensions, you should probably set some environment variable like `PERL5LIB`, `PHPLIB`, `PYTHONPATH`, `RUBYLIB`, `RUBYPATH`
- Try to get some feeling for the software.
- There's no magic cure. If it doesn't work, fix it.
- Write down what you did in a wiki!

Setting the environment I

- Write a nice bash script that sets the environment.
- Not all environment variables are always pre-defined!

```
PATH="$TMPDIR/some_package/lib:$PATH"
if [ -z "$LD_LIBRARY_PATH" ]; then
  LD_LIBRARY_PATH="$TMPDIR/some_package/lib"
else
  LD_LIBRARY_PATH="$TMPDIR/some_package/lib:$LD_LIBRARY_PATH"
fi
export PATH LD_LIBRARY_PATH
```

- Save this bash script as `"$TMPDIR/some_package/setenv.sh"`
- Build tar-ball (and put it somewhere accessible):

```
tar -C "$TMPDIR" -z -c -f some_package.tgz some_package/
```

- On the WN: download and extract the tar-ball and source the bash-script:

```
tar -C "$TMPDIR" -z -x -f some_package.tgz
source "$TMPDIR"/some_package/setenv.sh
```

Setting the environment II

- Alternatively, use the Modules system
- Pros:
  - Modules will handle interdependencies between packages
  - Much more concise
  - Modules can be “unloaded”
  - Lots of examples on the UI nodes
- Cons:
  - Module files are in Tcl/Tk (Eeiw!#$@)
- Example:

```
prepend-path PATH "/opt/vl-e/taverna_1.7/bin"
if [ module-info mode load ] {
    if [ is-loaded java ] {
        module switch java/1.5
    } else {
        module load java/1.5
    }
}
```