Plugins: la vendetta ;-)

L. Martin-Samos

Let's recall

- New features inside a "QE" module
  - invasive: old metadyn, constrain dynamics, ...
  - small impact: new metadyn (plumed), ...
- New module that uses QE routines without changing them: NEB, ...
- New module that uses modified QE routines
  - EPW
- New independent module that behaves as PP: yambo, want ...

Let's recall

- New features inside a "QE" module
  - invasive: old metadyn, constrain dynamics, ...
  - small impact: new metadyn (plumed), ...

My changes really need to be invasive?

  - EPW
- New independent module that behaves as PP: yambo, want ...

Specific to the kind of additional features

- **New features inside a “QE” module**
  - invasive: old metadyn, constrain dynamics, ...**NO!**
  - small impact: new metadyn (plumed), ...
- New module that uses QE routines without changing them: NEB, ...
- **New module that uses modified QE routines**
  - EPW
- New independent module that behaves as PP: yambo, want ...

Specific to the kind of additional features

- **New features inside a “QE” module**
  - invasive: old metadyn, constrain dynamics, ...**NO!**
  - small impact: new metadyn (plumed), **environ**, ...
- New module that uses QE routines without changing them: NEB, ...
- **New module that uses modified QE routines**
  - EPW
- New independent module that behaves as PP: yambo, want ...

- Compilation within QE
- diff and patch commands

Compilation within QE

Kind of features

1. C routines
2. Fortran routines
3. Fortran modules
4. Fortran routines and modules

Change the corresponding Makefile to add object files to be compiled and linked by the compiler

What to do?

1. C routines -> sym link routines inside clib dir
2. Fortran routines -> sym link routines inside flib
3. Fortran modules -> sym link routines inside Modules
4. Fortran routines and modules -> sym link inside PW/src, ...

What to do?

1. C routines -> sym link routines inside clib dir
2. Fortran routines -> sym link routines inside flib
3. Fortran modules -> sym link routines inside Modules
4. Fortran routines and modules -> sym link inside PW/src, ...

Change the corresponding Makefile to add object files to be compiled and linked by the compiler

addsonpatch.sh (addsontool.sh)

USAGE :
./install/addsonpatch.sh ADDSON_NAME WHERE_SOURCE WHERE_LINKS (-patch) (-revert)
WHERE_SOURCE is the relative path to the sources of the Addson code
WHERE_LINKS is the relative path to the QE directory where the addson sources have to be linked
 -patch : apply patch to Makefiles
 -revert : revert Makefiles to original

It works for *.f90 sources, sym links in flib or Modules, patch more than one plugin (no revert)

What to do?

1. C routines -> sym link routines inside clib dir
2. Fortran routines -> sym link routines inside flib
3. Fortran modules -> sym link routines inside Modules
4. Fortran routines and modules -> sym link inside PW/src, ...

Change the corresponding Makefile to add object files to be compiled and linked by the compiler

Regen make.depend if needed

makedeps.sh

```
if test $# = 0
then
   dirs=" Modules clib PW/src CPV/src flib PW/tools upftools PP/src PWCOND/src\
        PHonon/Gamma PHonon/PH PHonon/D3 PHonon/FD atomic/src XSpectra/src \
        ACDFT NEB/src Environ/src TDDFPT/src GIPAW/src GWW/pw4gww GWW/gww
GWW/head"

elif
   test $1 = "-addson"
then
   echo "The script for adding new dependencies is running"
   echo "USAGE $0 -addson DIR DEPENDENCY_DIRS"
   echo "$0 assumes that the new dependencies are in $TOPDIR/../"
#    ninput=$#
#    echo "number of input arguments: $ninput"
   dirs=$2
   shift
   shift
   add_deps=$*
   echo "dependencies in $add_deps will be searched for $dirs"
else
   dirs=$*
fi
```

diff and patch commands

diff file1.txt file2.txt >> difffile.patch

patch file1.txt < difffile.patch

file1.txt becomes identical to file2.txt

Reducing impact: HOWTO

- create empty routines when possible
- make the script that make the patch

We can manage for you the automatic download, untar and compilation

WE CAN HELP YOU FOR THE INTEGRATION, CONTACT US!!