Trac, svn and mailing-list

- http://www.tddft.org/trac/libpspio
- http://www.tddft.org/svn/libpspio
- libpspio-devel@tddft.org

Set svn and Trac password:

- Login into www.tddft.org
- `htdigest /server/www/.htpasswd libpspio <username>`

What should `libpspio` do

- Parse a pseudopotentail file
- Store the pseudopotential data internally
- Provide routines to access specific chunks of the psp data

General design considerations

- Autotools
- Error handling: always return control to program
- Documentation (Doxygen?)
- Fortran interface
- Testsuite
- Debug mode?
- Use atomic units internaly

Dependencies

- GSL?
- Libxc?

Related questions:

- Should we return the data on the original grid, or interpolate?
- Internal representation of ixc?

Things to do

- Decide formats to be implemented
- Decide data structures
- Decide API
- Assign tasks
- Code!

Formats to be supported: now and future

- Abinit (format 4, 5, and 6, HGH, GTH, others?)
- FHI98PP
- ATOM (José Luis Martins version)
- SIESTA
- UPF

More information and examples in `trunk/psp_references`

Data structures

Do not reinvent the wheel...

Data structures in APE

```
type atom_t
  private
  integer  :: type
  real(R8) :: z
  character(3) :: symbol
  integer   :: wave_eq
  integer   :: theory_level
  type(xc_t):: xc_model
  type(potential_t) :: potential
  type(mesh_t) :: m
  integer :: nspin
  integer :: n_states, n_sc
  type(state_t), pointer :: states(:,:)
end type atom_t
```

Data structures in APE

```
type ps_generator_t
  integer       :: nspin
  integer       :: scheme
  real(R8)      :: tol
  character(3) :: unbound_trick
  integer       :: n_states, n_sc
  type(qn_t), pointer :: qn(:,:)
  real(R8),   pointer :: rc(:)
end type ps_generator_t
```

Data structures in APE

```
type state_t
  ! General information about the state
  type(qn_t)        :: qn    ! state quantum numbers
  real(R8)          :: occ   ! occupation
  real(R8)          :: ev    ! eigenvalue
  character(len=5)  :: label ! a label to identify the state
  integer           :: wave_eq ! wave-equation used to obtain the w

  ! The wavefunctions
  integer :: np, wf_dim
  real(R8), pointer :: wf(:,:)  ! Wavefunction
  real(R8), pointer :: wfp(:,:) ! Derivative of the wavefunction

  ! Some information about the wavefunctions
  real(R8) :: peak ! outermost peak position
  real(R8) :: node ! outermost node position
end type state_t
```

Data structures in APE

```
type mesh_t
  integer,  private           :: type !mesh type
  real(R8), public            :: a    !mesh parameters
  real(R8), public            :: b    !
  integer,  public            :: np   !mesh number of points
  real(R8), public,  pointer :: r(:) !mesh points

  integer, private :: intrp_method ! Method to interpolate functi
  integer, private :: integ_method ! Method used to calculate int
  integer, private :: deriv_method ! Method used to calculate der
  integer, private :: interp_range
  integer, private :: fd_order
  type(fd_operator_t), private :: deriv
  type(fd_operator_t), private :: deriv2
  type(fd_operator_t), private :: deriv3
end type mesh_t
```

Data structures in APE

```
type potential_t
  private
  integer       :: type
  type(mesh_t) :: m

  type(loc_potential_t), pointer :: vl
  type(sl_potential_t),  pointer :: vsl
  type(kb_projectors_t), pointer :: kb

  !Screening  will be treated as a local potential
  logical :: screened
  integer :: nspin
  type(loc_potential_t), pointer :: vhxc(:)
  type(loc_potential_t), pointer :: vxctau(:)
end type potential_t
```

Data structures in APE

```
type kb_projectors_t
  private
  !Local part
  integer :: l_local
  type(loc_potential_t) :: vl
  !Projectors
  integer :: nc
  type(qn_t), pointer :: qn(:)
  real(R8),   pointer :: e(:)
  real(R8),   pointer :: p(:,:)
  type(spline_t), pointer :: p_spl(:)
end type kb_projectors_t
```

API

```
pspio_parse()
pspio_get_this()
pspio_set_this()
pspio_has_this()
```

Tasks

- Autotools
- Implement data structures
- Parsing
- API
- Documentation
- Fortran interface
- Support for libpspio in Octopus and Abinit