Coding standards for the CP-PAW code

Peter E. Blöchl

Contents

Chapter 1

Coding Standards of the PAW project

1.1 SHELL scripting

[**?**]

- specify the SHELL variable: start the script with `#!/bin/bash`. You may specify another shell, but bash is preferred.
- Define a variable `USAGE` describing the function and options. Example:

  ```
  export $USAGE="Usage of $0\n"
  USAGE="$USAGE description \n"
  ```

- pass arguments as variables to options. Analyze options with getopts. Example

  ```
  while getopts :h0b:p: OPT ; do
    case $OPT in
      x)   # executable
        EXCTBLE=$OPTARG
        shift
        ;;
      b)   # directory holding paw executables
        PAWXDIR=$OPTARG
        shift
        ;;
      p)   # project name
        PROJECT=$OPTARG
        echo argument projectname=${NAME}
        shift
        ;;
      0)   # dry run only
        DRYRUN=yes
        echo option dry-run=${DRYRUN}
        shift
        ;;
  ```

```
      h)   # help
        echo -e $USAGE
        exit 1
        ;;
      \?)   # unknown option (placed into OPTARG, if OPTSTRING starts with :)
        echo "error in $0" >&2
        echo "invalid option -$OPTARG" >&2
        echo "retrieve argument list with:" >&2
        echo "$0 -h" >&2
        exit 1
        ;;
      :)    # no argument passed to option requiring one
        echo "error in $0" >&2
        echo "option -$OPTARG requires an additional argument" >&2
        exit 1
        ;;  esac
    esac
  done
```

- check if all mandatory arguments have been passed.
- for every error, exit with a non-zero return code, i.e. by `exit 1`, and issue an error message to “error out”=&2.

  ```
  echo "error in $0: message" >&2
  exit 1
  ```

- finish the script with `exit 0`

1.1.1 List of recommended option id's

c name of the control file

i input file (other than a control file)

o output file

p root name of the paw project

x executable

b directory holding the executables (to select a specific paw distribution)

0 dry run

v verbose

q quiet

h issue help message

1.1.2 Brief description of getopts

The bash command

`getopts` $OPTSTRING OPT

processes an option string OPTSTRING, and returns true or false depending of whether it encountered a valid option in teh calling sequence of the calling bash script. It returns the id of the option as $OPT and it sets the variable OPTARG with the argument of the option. In case of an error OPTARG contains the name of the option, if OPTSTRING starts with a colon ":".

The option string is a string of option letters. An option with an argument is followed by a colon ":". An initial ":" switches getopts into the quiet mode, which also changes the error handling. Therefore capture all errors and work in quiet mode.

- A double dash "–" signifies the end of the options
- Options may be grouped such as `-abc` which is identical to `-a -b -c`.
- Options may only be single letters or numerals.

1.1.3 Default environment

- the current PAW directory can be obtained via

  ```
  export PAWXDIR=$(which paw_fast.x); PAWXDIR=${PAWXDIR%paw_fast.x}
  ```

- The current directory is captured with

  ```
  THISDIR=$(pwd)   # current directory
  ```

Bibliography