Tecniche di programmazione in chimica computazionale

Global variables & make

Emanuele Coccia

Dipartimento di Scienze Chimiche e Farmaceutiche

Global variables

- ''Info transfer'' between the calling program and the subroutine/function:

Global variables

- ''Info transfer'' between the calling program and the subroutine/function:
  - Formal arguments (as previously seen)

Global variables

- ''Info transfer'' between the calling program and the subroutine/function:
  - Formal arguments (as previously seen)
  - Global variables

Global variables

- ''Info transfer'' between the calling program and the subroutine/function:
  - Formal arguments (as previously seen)
  - Global variables
- Sharing variables through the module

Global variables

- ''Info transfer" between the calling program and the subroutine/function:
  - Formal arguments (as previously seen)
  - Global variables
- Sharing variables through the module
- Access to the variables declared into a module: *use name_module*

Global variables

- ''Info transfer" between the calling program and the subroutine/function:
  - Formal arguments (as previously seen)
  - Global variables
- Sharing variables through the module
- Access to the variables declared into a module: *use name_module*
- "Use" must be inserted before any variable declaration or "implicit"

Global variables

- ''Info transfer" between the calling program and the subroutine/function:
  - Formal arguments (as previously seen)
  - Global variables
- Sharing variables through the module
- Access to the variables declared into a module: *use name_module*
- "Use" must be inserted before any variable declaration or "implicit"
- Example module.f90

Module

- A module can contain subroutines and/or functions

Module

- A module can contain subroutines and/or functions
- No need to pass global variables as arguments

Module

- A module can contain subroutines and/or functions
- No need to pass global variables as arguments
- Explicit (i.e., procedures in a module) versus implicit interface (i.e., procedures not in a module)

Module

- A module can contain subroutines and/or functions
- No need to pass global variables as arguments
- Explicit (i.e., procedures in a module) versus implicit interface (i.e., procedures not in a module)
- Example module2.f90

Make utility

- Complex programs have several source files:

Make utility

- Complex programs have several source files:
  - Modify one or few files

Make utility

- Complex programs have several source files:
  - Modify one or few files
  - Only the last modified files should be recompiled

Make utility

- Complex programs have several source files:
  - Modify one or few files
  - Only the last modified files should be recompiled
  - Make utility automatically recognizes files to be recompiled

Make utility

- Complex programs have several source files:
  - Modify one or few files
  - Only the last modified files should be recompiled
  - Make utility automatically recognizes files to be recompiled
  - Make knows dependencies between files

Make utility

- Complex programs have several source files:
  - Modify one or few files
  - Only the last modified files should be recompiled
  - Make utility automatically recognizes files to be recompiled
  - Make knows dependencies between files
- Mandatory Makefile file:

Make utility

- Complex programs have several source files:
  - Modify one or few files
  - Only the last modified files should be recompiled
  - Make utility automatically recognizes files to be recompiled
  - Make knows dependencies between files
- Mandatory Makefile file:
  - Describing connections between files

Make utility

- Complex programs have several source files:
  - Modify one or few files
  - Only the last modified files should be recompiled
  - Make utility automatically recognizes files to be recompiled
  - Make knows dependencies between files
- Mandatory Makefile file:
  - Describing connections between files
  - Commands to execute to update files

Make utility

- Complex programs have several source files:
  - Modify one or few files
  - Only the last modified files should be recompiled
  - Make utility automatically recognizes files to be recompiled
  - Make knows dependencies between files
- Mandatory Makefile file:
  - Describing connections between files
  - Commands to execute to update files
- Type "make" in the directory containing source files and Makefile

Make utility

- Makefile:

  TARGET ... : DEPENDENCIES ...

  COMMAND

  ...

  ...

Make utility

- Makefile:
  TARGET ... : DEPENDENCIES ...
  COMMAND
  ...
  ...
- TARGET: exe or object name, action (e.g., clean)

Make utility

- Makefile:
  TARGET ... : DEPENDENCIES ...
  COMMAND
  ...
  ...
- TARGET: exe or object name, action (e.g., clean)
- DEPENDENCIES: inputs to generate TARGET

Make utility

- Makefile:
  TARGET ... : DEPENDENCIES ...
  COMMAND
  ...
  ...
- TARGET: exe or object name, action (e.g., clean)
- DEPENDENCIES: inputs to generate TARGET
- COMMAND: command(s) to be applied to DEPENDENCIES

Make utility

- Makefile:
  TARGET ... : DEPENDENCIES ...
  COMMAND
  ...
  ...
- TARGET: exe or object name, action (e.g., clean)
- DEPENDENCIES: inputs to generate TARGET
- COMMAND: command(s) to be applied to DEPENDENCIES
- Example with program.f90, do_op.f90 and output.f90