Log in to the LRZ Compute Cloud

The windows command line (Start → Enter command) should be used to execute the following command:

- MobaXterm

In the MobaXterm (X server is Embedded) window, please login to the cloud virtual machine via,

  ➢ ssh student@IP

The IP assigned to you is given on your individual desk, and the password is provided on the blackboard. Once you have logged in, you can start additional programs like xterm or editors like vi or emacs.

Compute Cloud Machine
- 4 core virtual machine with 18 GB RAM

Compilers
The following compilers are available for these exercises:

- gfortran (part of GCC, the GNU compiler collection) – version 7 and 8;
  ◦ Usage - module load is not required, simply use e.g., gfortran-7
- ifort (Intel Fortran compiler – commercial) – version 19;
  ◦ Usage - module must be loaded (module load intel)
- nagfor (NAG Fortran compiler - commercial) – version 6.2;
  ◦ Usage – module load is not required

Note that after switching you will need to rebuild previously generated module information files as well as object files.

The command to build coarray programs with gfortran is caf, while for Intel Fortran you should use the mpif90 command and add the compiler switch -coarray.

- OpenCoarrays can be used after loading its module. Please look up the module name by typing “module av opencoarrays”

Debuggers / Tools
The following debuggers and tools are available:

- gdb (GNU debugger) and gdb-ia (Intel-modified version of GNU debugger)
- Valgrind (memory debugger – search via module av valgrind)
- GSL (search via module av gsl)

Note: commercial products may be only used during the course.
Editing sources and makefiles
For editing Fortran source files or Makefiles, using either the vi or emacs editor is recommended. Alternatively, the WYSYWIG editor geany is available.

Further readings and information about the language
For quick reference on selected language elements you may also consult the documentation provided by the compiler vendors, e.g. here for Intel and GNU, but be aware also of non-portable vendor extensions:

https://gcc.gnu.org/onlinedocs/gfortran/

Evaluation procedure
At the end of the course, you will be asked to fill in an electronic evaluation form before you receive your course certificate.
# Table of Contents for the slides

## Slide set days 1-3

### Introductory remarks
- Overview of topics, conventions used, references: 2-7

### Recapitulation: Module features and the environment problem
- Modules, global variables, encapsulation, module procedures, procedure arguments: 9-15
- Threading issues for globals: 16-18
- The environment problem and some of its solutions, particularly reverse communication: 19-30

### Dynamic memory and object-based design
- POINTER and ALLOCATABLE attributes and their usage; some newer features: 32-35
- “Container types” – dynamic type components: 36-39
- Dynamic objects in procedure interfaces, semantics of INTENT: 40-46
- Opinionated recommendation on use of dynamic objects: 47
- Recapitulation of generic interfaces and overloading: 48-49
- A type representing sparse matrices; overloading the structure constructor and the assignment operator; finalizers: 50-56
- Aliasing rules for dynamic objects and TARGETs: 57-66
- ASSOCIATE construct: 67-68
- Library design and the Open Closed Principle: 69-71

### Object oriented programming (I)
- General aims and scope of OO in Fortran: 73-74
- Type extension, inheritance mechanisms, component accessibility, structure constructor: 75-82
- Polymorphism for data and interfaces, polymorphic arrays, allocation mechanisms: 83-88
- Type and Class identification at run time (SELECT TYPE): 89-92
- Unlimited polymorphic objects, ad-hoc type comparison: 93-97

### Object oriented programming (II)
- Type-bound and object-bound procedures, pointers to procedures, invocation of such procedures: 99-106
- Overriding a type-bound procedure, private type-bound procedures, the NON_OVERRIDABLE attribute, use of SELECT TYPE vs type-bound procedures: 107-112
- Passed object: arrays, the PASS and NOPASS attributes: 113-114
- “Interface classes” – abstract types and deferred type-bound procedures: 115-119
- Dependency inversion with submodules – splitting interfaces and implementations: 121-129
- Generic type-bound procedures, additional overloading features: 131-139

### Advanced I/O Topics
- Nonadvancing I/O: 141-143
- User-defined derived type I/O: 145-151
- Asynchronous I/O: 153-159
- Performance and I/O design considerations: 161-166
IEEE Arithmetic and Exception Handling
- IEC 754 Standard, intrinsic modules for IEEE features: 168-171
- Floating point model, inquiry intrinsics for model parameters: 172-175
- Rounding and rounding modes, denormal numbers, inquiry of support for IEEE conforming operations, underflow handling: 176-183
- The five floating point exceptions, example on divide_by_zero: 184-187
- Halting mode, saving and restoring flag and floating point status: 188-191
- Classification of IEEE conforming number values, final comments: 192-194

Parameterized derived types (PDT)
- KIND and LEN parameters, comparing PDT with conventional container types, type parameter inquiry: 196-199
- Assumed, deferred and default type parameters: 200-202
- PDTs and object orientation: 203

Interoperation of Fortran with C in Fortran 2003
- Overview of Fortran 2003 interoperability, concept of companion processor: 205-207
- Interoperable intrinsic types and their use in interfaces, mapping of procedures, linkage: 208-211
- Passing simple arrays between Fortran and C: 212-215
- String handling, interoperable derived types: 216-219
- Handling non-interoperable types – the c_ptr type and the intrinsics c_loc and c_f_pointer: 220-230
- Procedure arguments and pointers – the c_funptr type and the intrinsics c_funloc and c_f_procpointer: 231-235
- Interoperation of global variables and COMMON blocks, enumeration: 236-238

Extended interoperation defined in Fortran 2015 (TS 29113)
- Overview of Fortran 2015 interoperability: 239
- The source file ISO_Fortran_binding.h – the C descriptor for “complex” Fortran dummy objects: 240-242
- Processing array elements in C: 243-245
- Creating Fortran objects within C: assumed shape, ALLOCATABLE, POINTER: 246-252
- Creating subobjects of strings or derived-type objects: 253-254
- Assumed-rank and assumed-type objects: 255-261
- Automatized C include file translation and final remarks: 262-263
- Shared libraries and dynamic loading (“Plug-ins”): 265-271
Slide set days 4-5 (PATC)

Design Patterns (I): Creation and destruction of objects
- Polymorphic type components and polymorphic overloaded constructor, factory procedures with polymorphic dummy arguments: 3-8
- Operator overloading for polymorphic operands and function results: 9-11
- More on finalizers, the IMPURE attribute, finalization of type extensions and layered types: 12-16
- Restrictions on PURE procedures for polymorphic arguments and function results: 17

Design Patterns (II): A solution to the environment problem
- Using an interface class to solve the parameterized integration problem: 19-23
- The Strategy pattern: implementing alternative integration methods without signature changes: 24-28
- The Class Adapter pattern: smooth integration of legacy code: 29-33
- Addendum: Performance considerations for object-oriented programs: 35-38

Design Patterns (III): Error handling in complex applications
- Tracking an exception: Unwinding the stack vs. tracking along objects; the Chain of Responsibility pattern: 40-47
- Throwing and catching exceptions: 48-49
- Restoring an object’s state – the Memento pattern: 50-52

Design Patterns (IV): Maintaining a consistent view of objects
- The Observer pattern – extracting observables from a simulation: 54-57
- The Observer framework: 58-60
- Implementing the simulation: 61-65

Parallel Programming with Coarrays (I): Setting the stage
- Introduction, concept of image, resource mapping, inquiry intrinsics for image management: 67-71
- Data decomposition and local-to-global index mapping, matrix-vector multiplication as illustration: 72-75

Coarrays (II): Basics
- Need for communication, PGAS memory model, CODIMENSION attribute, coindexed accesses to coarrays (Put and Get), local accesses to coarrays: 77-81
- Synchronization requirements and rules, SYNC ALL image control statement: 82-86
- Matrix-vector example completed, analyzing the communication pattern: 87-90
- Varia: Non-trivial coindex-to-image mappings and concept of corank, supporting intrinsics, integration of type system: 91-93

TS 18508: Collective procedures
- Motivation: 95
- Simple and user-defined reduction: 96-97
- Broadcasting data to all images, general rules for collectives: 98-99
Coarrays (III): Dynamic memory
- Symmetric memory, allocatable coarrays and coarray components: 101-105
- Coarrays as TARGETs: 106
- Coarrays of container type and handling of unsymmetric memory: 107-110
- Limitations on intrinsic assignment, coarray subobjects: 111-112

Coarrays (IV): Advanced synchronization
- Image subset synchronization with the SYNC IMAGES image control statement: 114-116
- One-sided synchronization using the TS 18508 image control statements EVENT POST and EVENT WAIT, multiple posting: 117-120
- Mutual exclusion with critical regions and locks: 121-124
- Non-blocking lock and event use (EVENT_QUERY), event and lock subobjects: 125-127
- Memory fence (SYNC MEMORY) and atomic subroutines: 128-134

Coarrays (V): Considerations for type and library design
- Coarrays and object orientation – Polymorphic coarrays, execution of type- and object-bound procedures: 136-139
- Polymorphic coarray subobjects, limitations on assignment and coindexing: 140
- Coarrays as dummy arguments, corank and coindex remapping, image-dependent argument passing of coarrays: 142-143
- Limitations for coarray use in PURE and ELEMENTAL procedures: 144
- Procedure-local coarrays: 145
- Invocation of (serial) procedures with coindexed actual arguments: 146
- Coarray factory procedures: 147
- Working around limitations – Assignment overloading and type-bound communication procedures: 148-150
- Documenting synchronization behavior of library code: 151
- Interoperation of coarrays with MPI – some very basic ideas: 153-155

TS 18508: Teams of images
- Motivation: limitations of flat model for performance and program design, team concept: 157-159
- FORM TEAM statement: 160-161
- CHANGE TEAM block construct – changing execution context: 162-163
- Handling coarray data inside a CHANGE TEAM block and across the context change: 164-166
- Image selector – accessing data across team boundaries: 167
- Synchronization across teams – SYNC TEAM statement, putting things together: 168-169

Coarrays (VI): Program Termination
- The three termination variants: 171
- Error termination and the ERROR STOP statement: 172
- Normal termination via STOP and its synchronization properties, STAT argument on image control statements: 173-174
- Fail-safe execution as an optional feature in TS 18508: Failed images, the FAIL IMAGE statement, identifying failed images via STAT and the FAILED IMAGES intrinsic: 175-179
- Fail-safe execution: use of spare images in a team setup: 180