1 Hardware

The department of Meteorology and Geophysics has access to 3 servers (**SRVX1**, **SRVX8**, **AURORA**) as well as one cluster (**JET**). Additionally the department has access to private nodes on **VSC 4/5**. As a **staff** member, **PhD** student or **external** researcher you will have **access to all** of these resources if necessary. **Students** get access to the **TeachingHub**. Master students can get access to everything as well. **Please talk to your supervisor.**

Servers & Clusters

Name	Location	# Nodes	Purpose
SRVX1	Arsenal	1	teaching
SRVX8	Arsenal	1	services
AURORA	Arsenal	1	R&D, compute
JET 01	Arsenal	1	R&D, visual
JET 02	Arsenal	1	R&D, jupyter
JET	Arsenal	19	compute
VSC4	Arsenal	5	compute
VSC5	Arsenal	11	compute
VSC5	Arsenal	1	GPU

CPU Architectures

Node	CPU	# Cores	RAM
jet0**X**	Intel(R) Xeon(R) Gold 6148	2x20	768GB
jet1**X**	AMD EPYC 9454P	2x48	1152GB
srvx1	Intel(R) Xeon(R) Gold 6148	4x20	768GB
srvx8	Intel(R) Xeon(R) CPU E5-2697	2x14	510GB
aurora	AMD EPYC 7773X	2x64	2TB

2 Storage

Please familiarize yourself with the storage options. We have two large file systems (**srvfs** and **jetfs**), with the former being a local storage array (raid fault protection) and the latter being a global storage array (raid fault protection). Because we have **multi-user** systems, there needs to be some **quotas**.

File systems & Quotas (Default)

mountpoint	space	quotas	fs
/srvfs/home	400*TB*	200*GB*	local
/srvfs/scratch	400*TB*	no	local
/jetfs/home	100*TB*	100*GB*	global
/jetfs/scratch[a]	3.6*PB*	no	global

[a]Note `/srvfs/scratch` and `/jetfs/scratch` are shared by all users. Please use responsibly.

Please remember that **storage is always limited** and it is necessary to delete things after some time. Data that can easily be recomputed should not be stored forever. Remember that **source code and important things** should go to your **HOME** and not so much important things as well as **large data sets or temporary data** should go to **SCRATCH**.

Directories and Purposes

mountpoint	fs	**purpose**	**backup**
/srvfs/home	srvfs	personal files, code	X
/srvfs/scratch	srvfs	data	-
/srvfs/tmp	srvfs	data	-
/srvfs/data	srvfs	data	X
/srvfs/lehre	srvfs	teaching	-
/srvfs/shared	srvfs	share data	-
/srvfs/webdata[a]	srvfs	share web	-
/jetfs/home	jetfs	personal files, code	X
/jetfs/scratch	jetfs	data	-
/jetfs/shared-data	jetfs	share data	-

[a]Accessible for everyone, unless requested, under webdata.wolke.img.univie.ac.at

Please note that unless explicitly stated, there is no disaster backup available on these directories. **Backup is done once a day, late in the evening.**

2.1 Sharing data

It can be very important to share data with colleagues within and outside of the department. Please use one of the following methods:

- u:cloud or `userservices ucloud` (personal: 50GB)
- `userservices filesender` (250GB)
- webdata (`/srvfs/webdata/`) (depends)

Sharing via u:cloud is easy via the web interface or u:cloud client or using the userservices. Permissions can be given to persons or via a link. Using the ACOnet filesender requires you to login (*u:account*) and create an API secret (docs). Sharing via the webdata is quick from the servers, but access is not restricted by default. Transfer of files is limited by uplink speed and system performance.
More details are available on u:wiki / Tutorials / share data

3 Access

An IMGW server account can be requested by your supervisor. Please connect to the servers using one of the following suggestions:

- `ssh` available on Linux, iOS, Windows
- `MobaXterm` or `Putty` or `Bitvise` available on Windows

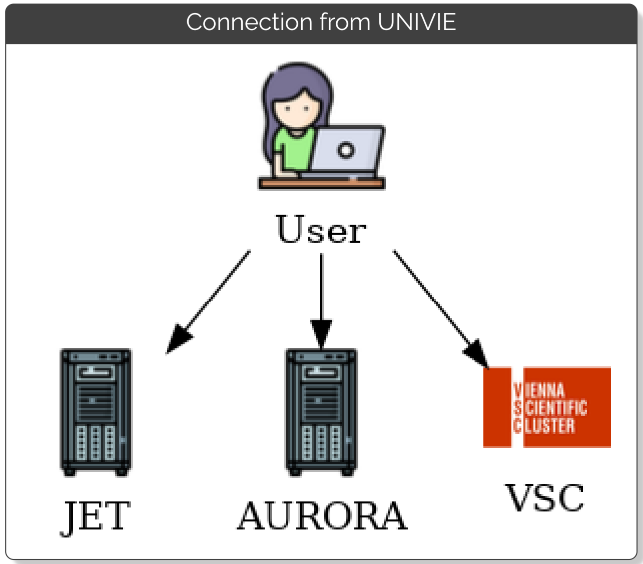
any other software is fine too.

Password Manager

Please to use a password manager e.g. Bitwarden, KeepassXC, ... You will have at least these accounts:

- `u:account`
- `w:account` (Wolke, IMG)
- `vsc:account`

3.1 Connection from IMGW



Of course there are numerous web services, but secure shell access (ssh) is still the most common:

Access via secure shell

```
ssh <w:account>@login.img.univie.ac.at
ssh <w:account>@aurora.img.univie.ac.at
ssh <w:account>@jet01.img.univie.ac.at
ssh <w:account>@jet02.img.univie.ac.at
ssh <vsc:account>@vsc4.vsc.ac.at
ssh <vsc:account>@vsc5.vsc.ac.at
```

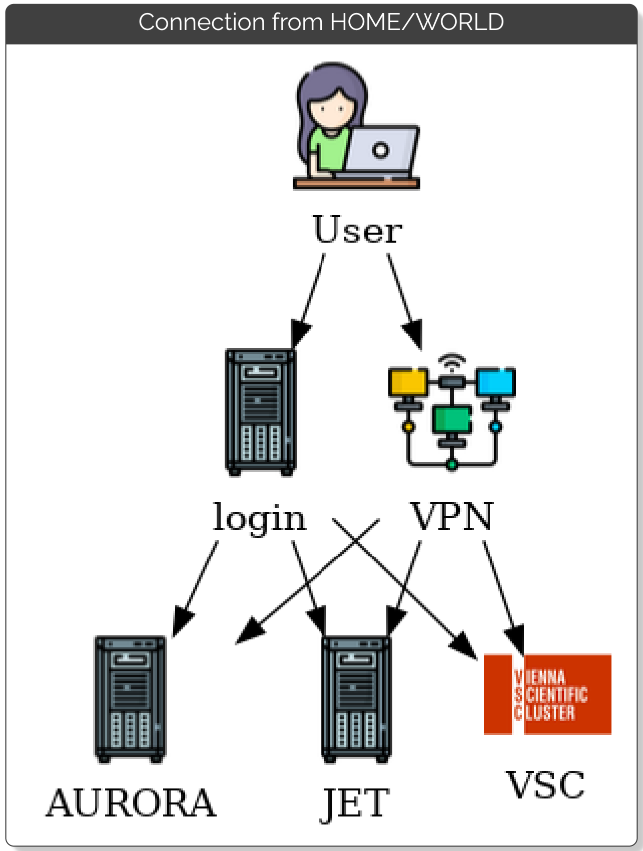
3.2 Connection from outside

If you are **outside** the university network, e.g. at home, EDUroam, outside AT, … it is required to login via a jump host to most of the servers.

Access via jump host

```
ssh -J <w:account>@login.img.univie.ac.at
<w:account>@jet01
```

Alternatively, it is also possible to use the **VPN** from the University of Vienna. More information can be viewed at the ZID, but this requires a *u:account*.



4 Help & Documentation

There is extensive information available on these sites, depending on your access level. The most convenient is to use the **documentation** available on Wolke, which is available as well on GitLab. For staff members there are information on the u:wiki

https://wolke.img.univie.ac.at/documentation/general
https://wiki.univie.ac.at/display/theomet
https://gitlab.phaidra.org/imgw

Area of Expertise Contact

IT support	it.img-wien@univie.ac.at
HPC list	rechnen.imgw@lists.univie.ac.at
IT HPC	michael.blaschek@univie.ac.at
VSC support	service@vsc.ac.at
Gitlab support	support.phaidra@univie.ac.at

Chat with us on Mattermost (🐛). VSC offers a **service desk** (`support.vsc.ac.at`) to handle problems or requests.

5 Software

All servers at the department as well as all HPC systems have a software stack that is based on **modules**. These so called environment modules allow **dynamic loading of different versions of the same software and dependencies**. Common commands are shown here:

`module av`	Show available modules
`module load <name>`	Load module <name>
`module unload <name>`	Unload module <name>
`module purge`	Unload all module
`module list`	List loaded modules
`module show <name>`	Show module <name>

5.1 Advanced software management - spack

On VSC, JET and many other HPCs () the software is installed using spack which is a more than just a package manager and might replace environment modules as well. You can use spack as module replacement like this:

`spack find`	Show available packages
`spack load <name>`	Load packages
`spack load -list`	Show loaded packages
`spack unload -a`	Unload all packages
`spack info <name>`	Info on package

Spack can create software environments, where you can choose to install **libraries** and **software in a user environment**. However, this will create a **lot of small files** and is not recommended on GPFS file systems. **If you require an additional software or library package please get in touch with IT or VSC service desk**. More infos in section 10

5.2 Software Stacks

There are software stacks usually grouped by **compiler**:

- GNU Compilers (`gcc, gfortran, g++`)
- Intel Compilers (`icc, ifort, icpc`)
- Intel Oneapi Compilers (`icx, ifx, icpx`)
- AMD Compilers AOCC (`clang, flang, clang++`)

and then there are usually different version of these compilers and message parsing interface (**MPI**) implementations

- `openmpi`
- `intel-mpi`
- `mpich`

As you can see this evolves really quickly into a large tree with a lot of different options and configurations. A simple example:

Loading a software stack

```
# Load the netcdf fortran version 4.5.3 with GNU compiler
8.5.0
module load module load netcdf-fortran/4.5.3-gcc-8.5.0
# show what is loaded
module list
Currently Loaded Modulefiles:
 1) ucx/1.11.2-gcc-8.5.0    5) netcdf-c/4.7.4-gcc-8.5.0
 2) openmpi/4.0.5-gcc-8.5.0 6) netcdf-fortran/4.5.3-gcc-8.5.0
 3) hdf5/1.10.7-gcc-8.5.0   7) gcc/8.5.0
 4) parallel-netcdf/1.12.2-gcc-8.5.0
```

This example shows you the dependencies of some packages and hence the complexity these software installations can be. Imagine that all of these libraries change version regularly and how many packages we would end up with. **Therefore installing all versions is not feasible not desirable.**

How to compile a Fortran program?

```
# Load the module
module load gcc-8.5.0
# Compile the program
gfortran -x -o test.exe test.f90
# unload all modules
module purge
# load intel compiler
module load intel-oneapi-compiler
# Compile the program
ifort -x -o test.i.exe test.f90
```

6 Job scheduler

Most HPC systems use a job scheduler. **VSC and JET use slurm**, which uses the following commands to control jobs:

`sinfo`	partition information
`squeue`	queue information
`sqos`	quality of service information
`scontrol show job <jobid>`	show job information
`salloc`	request an allocation
`srun -N<n> <cmd>`	run a cmd
`sbatch <job file>`	run job script
`scancel <jobid>`	cancel a job
`scontrol update job <jobid> <option>`	Change job settings while running, e.g. TimeLimit
`seff <jobid>`	show efficiency of job

Example script on JET

```
#!/bin/bash
# SLURM specific commands
#SBATCH -job-name=test-run
#SBATCH -output=test-run.log
#SBATCH -ntasks=1
#SBATCH -mem=1MB
#SBATCH -time=05:00
#SBATCH -mail-type=BEGIN
#SBATCH -mail-user=<email@address.at>

# Your Code below here
module load miniconda3
# Execute the miniconda Python
# use /usr/bin/time -v [program]
# gives statistics on the resources the program uses
# nice for testing
/usr/bin/time -v python3 -v
```

Common sbatch options:

`--job-name=<name>`	job name
`--nodes=<n>`	number of nodes
`--ntasks=<n>`	number of tasks
`--ntasks-per-node=<n>`	tasks in parallel on a single node
`--ntasks-per-core=<n>`	tasks on a single core
`--mem=<mem>`	max memory: 1MB, 1GB
`--time=<time>`	estimated run time: D-HH:MM:SS

Please note that when you do not supply the `--output=` option, a file called `slurm-<jobid>.out` will be created by default.

Examples - resource control inside the job script

```
# openmp
# -ntasks=1 -cpus-per-task=n
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
# -ntasks=n
export OMP_NUM_THREADS=$SLURM_NTASKS
<exe>

# MPI
module purge
module load openmpi/xx.yy.zz
mpirun <exe>
```

Interactive job

```
# Request resources from slurm (-N 1, a full Node)
salloc -N 1 -p <partition> -qos <only on VSC> -no-shell
# Once the node is assigned / job is running
# Check with
squeue -u $USER
# connect to the Node with ssh
ssh [Node]
# test and debug the model there.
```

6.1 JET special commands

On the Jet cluster there are a few special commands to make your life easier.

`jobinfo <jobid>`	Shows information on a running job
`jobinfo_remaining`	Shows how long the current jobs lasts
`nodeinfo`	Shows usage of JET compute nodes
`queueinfo`	Shows queue information per node
`watchjob <jobid>`	Monitor a running job

JET slurm partitions

`hub`	jet03-jet06
`compute`	jet04-jet09
`amd`	jet10-jet19
`all`	jet03-jet19
`time limit`	no

more information on the details can be found in the documentation, check the help section.

JET jobs

Jobs can share nodes on JET. **Not on VSC.**

7 User services

All department servers have **special scripts** that are meant to make the users life easier. Please find a list of useful commands here:

`userservices <service>`	Master function for all services

All of these services have a help (`-h`) and example section.
List of Services:

`containers`	Show available apptainer containers
`fetch-sysinfo`	Display system information
`filesender`	Transfer files to ACONET
`fix-permissions`	fix file/directory permissions
`modules`	list environment modules
`numfiles`	Check number of files in all sub directories
`quota`	Report user quotas
`server-schedule`	Show Server schedule
`transfersh`	Transfer files/directories
`ucloud`	Upload files/directories to your u:cloud
`weather`	Retrieve weather information
`yopass`	Send messages/small files encrypted
`vnc`	Manage a VNC session (SRVX8)
`vnc-geometry`	Change VNC desktop resolution (SRVX8)

Other useful commands:

`ncdu -x <dir>`	Show disk usage of <dir> (man)
`htop`	Show running processes

Learn more about linux commands from **explainshell.com** or check the **man.cx** for manual pages of linux commands.

8 Vienna Scientific Cluster (VSC)

The **VSC is Austria's university HPC system** (part of EuroCC), which the department has private nodes and **researchers can request projects to get more resources.** Please note that on VSC only full nodes can be requested, whereas on JET nodes can be shared. Check their wiki

Currently available VSC HPC clusters:

1. VSC4 since 2019
2. VSC5 since 2022

Connect to VSC

```
ssh <vsc:account>@vsc4.vsc.ac.at
ssh <vsc:account>@vsc5.vsc.ac.at
ssh -J <w:account>@login
<vsc:account>@vsc5.vsc.ac.at
```

VSC Node Setup

HPC	#	CPU	# Cores	RAM
VSC4	5	Intel(R) Xeon(R) Platinum 8174	2x24	384GB
VSC5	11	AMD EPYC 7713	2x64	512GB
VSC5	1	GPU Nvidia A100	2x64	512GB

Useful commands:

`sbatch <job_script>`	Add job script to queue
`scancel <jobid>`	Cancel job
`sinfo`	Show node/queue status
`sqos -M vsc5 -x`	Show qos for VSC5
`sqos -M vsc5 -x`	Show qos for VSC5
`sqos -M vsc4 -x`	Show qos for VSC4
`squeue -p p71386_0512`	Show queue for IMGW only
`squeue -u \$USER`	Show queue for USER
`scontrol show jobid <jobid>`	Show job details
`sprio -j <jobid>`	Show job priority in queue

VSC Quality of Service (QOS)

`p71386_0384`	VSC4 qos
`skylake_0384`	VSC4 partition
`p71386_0512`	VSC5 qos
`p71386_a100dual`	VSC5 GPU qos
`zen3_0512`	VSC5 partition
`zen3_0512_a100x2`	VSC5 partition

Most queues on VSC have a walltime limit of 10 days.

VSC Storage shared VSC4/VSC5

name	space	# files	comment
HOME	200GB	2.000.000	global fs
DATA	100TB	2.000.000	global fs

VSC Storage

Please remember that storage is **shared among all IMGW users**.

Example VSC job

```
#!/bin/bash
#
#SBATCH -J TEST_JOB
#SBATCH -N 2
#SBATCH -ntasks-per-node=64
#SBATCH -ntasks-per-core=1
#SBATCH -mail-type=BEGIN
#SBATCH -mail-user=<email@address.at>
#SBATCH -partition=zen3_0512
#SBATCH -qos=p71386_0512
#SBATCH -account=p71386
#SBATCH -time=<time>

# when srun is used, you need to set (Different from Jet):
<srun -l -N2 -n64 a.out >
# or
<mpirun -np 64 a.out>
```

8.1 JET and VSC

Since Oktober 2023 the JET cluster storage file system is available on VSC5. Every VSC and JET user can find their files on both systems, with slightly different paths

JET file system on VSC

cluster	path
JET	`/jetfs/...`
VSC5	`/gpfs/jetfs...`

Please make sure that files and directories on JET have the correct permissions, since everybody on VSC is represented in the third column of linux permissions.

8.2 Linux file system permissions - reminder

Linux file permissions

```
+-----+---+--------------------------+
| rwx | 7 | Read, write and execute  |
| rw- | 6 | Read, write              |
| r-x | 5 | Read, and execute        |
| r-- | 4 | Read,                    |
| -wx | 3 | Write and execute        |
| -w- | 2 | Write                    |
| --x | 1 | Execute                  |
| --- | 0 | no permissions           |
+------------------------------------+
+------------+------+-------+
| Permission | Octal| Field |
+------------+------+-------+
| rwx------  | 700  | User  |
| ---rwx---  | 070  | Group |
| ------rwx  | 007  | Other |
+------------+------+-------+
```

Some examples:

`chmod u=rwx,g=rwx,o=rx`	For world executables files
`chmod 775`	For world executables files
`chmod u=rwx,g=rx,o=`	For executables by group only
`chmod 750`	For executables by group only
`chmod u=rw,g=r,o=r`	For world readable files
`chmod 644`	For world readable files
`chmod u=rw,g=r,o=`	For group readable files
`chmod 640`	For group readable files
`chmod u=rw,go=`	For private readable files
`chmod 600`	For private readable files
`chmod u=rwx,go=`	For private executables
`chmod 700`	For private executables

JET and VSC Warning

Remember **other** users (linux) now means including VSC users. **Do not give other users write permission to any of your directories!!!**

9 ECMWF

It is possible for you to get an account on **ECMWF Bologna HPC** system via your supervisor.

Connect to ECMWF

```
# requires teleport
module load teleport
# start ssh-agent
startagent
# run browserless login
python3 -m teleport.login
# Check if ssh keys are known to the agent
ssh-add -l
# login using your ECMWF credentials
ssh -J <user>@jump.ecmwf.int <user>@ecs-login
```

Sometimes it is necessary to kill the ssh-agent, run: `ssh-agent -k`.
On all department servers a module called `ecaccess-webtoolkit` is installed, that can be used to **monitor, submit jobs, transfer files.** For convenient transfer use `ectrans` to transfer files from and to ECMWF. Predefined **associations** can be configured using boaccess.ecmwf.int.
Other useful ECMWF services:

- confluence.ecmwf.int - Documentation
- desktop.ecmwf.int - Virtual Machine
- ECcharts - Maps + JupyterNotebooks

9.1 ECGateway

The department runs an EC Gateway server on JET02 (ecaccess.wolke.img.univie.ac.at), which can be accessed from ECMWF's servers, allowing transfer between JET and ECMWF. You need to login with your **ec:account** and create *Association* in *ECtrans Setup*. Giving your login credentials for JET (**w:account**) and specifying your **SCRATCH** space. Then you can use ectrans to transfer files between the two systems.

10 Spack

Spack is an open source project that offers a package management framework and tool for installing complex scientific software. It is designed to support multiple versions and configurations of a software on many different platforms and environments.
All the libraries and compilers are installed using spack and different version can be installed as you go. There is also the possibility as a user to use spack and install and build applications that have specific requirements. **However, this is rather complex.**

`spack list`	List and search all avail. packages
`spack find`	List all installed packages
`spack info <pkg>`	Show information on a package
`spack spec -I <pkg>`	Show dependencies
`spack install <pkg>@<version>`	install a package
`spack compiler list`	List avail. compilers
`spack load <pkg>`	Load module
`spack unload <pkg>`	Unload module

spack specs

spec	meaning	example
`@`	custom version	`mpileaks@3.3`
`+/-/`	build options/-variants	`mpileaks@3.3 +threads`
`target=`	Set CPU architecture	`target=cascadelake`
`^`	dependency information	`mpileaks ^mpich@3.2`
`/`	specify by hash	`spack load /h4jqiw`

spack user environment

```
# init spack
# For bash/zsh/sh
. $SPACK_ROOT/share/spack/setup-env.sh
# create a spack directory
mkdir $HOME/myspack
# create spack environment
spack env create -d $HOME/myspack
# activate spack environment
spack env activate $HOME/myspack
# install a package to your new environment
spack install gcc@10.3.1
# deactivate
spack env deactivate
```

more information can be found in the spack.readthedocs.io documentation. Ask IT to give you some guidance.

spack files/storage

Remember that when using spack a lot of files are created. Please keep in mind what you do and the impact this might have on others. Especially on VSC.

Spack can be used to build a containerized version of your library stack. Ask IT for soome guidance as well.

11 Python / Conda

On all servers `conda` is installed via the module system. Load one of the available modules, e.g. `miniconda3` or `micromamba` and start developing your Python code. **Micromamba** is a C version of conda and much faster.

Setup a Python environment

```
module load miniconda3
# install a python version
conda create -name myenv python=3.11 <other pkg>
# install package with version
conda install -n myenv scipy=0.18
# show what environment you have
conda env list
```

`conda`	Master conda process
`conda env list`	List conda environments
`conda install -n <env> <pkg>`	Install a pkg into env
`conda update`	Update packages

More information can be found in the documentation
It is possible to create a kernel for the **TeachingHub** or the **ResearchHub** from your environment. Just install `conda install -n <env> ipykernel` into your environment and you should be able to select it for notebooks. Check by running: `jupyter-kernelspec list`

12 Singularity / Apptainer

Singularity/Apptainer is a container technology for HPC that is designed to execute applications at high performance while being secure, portable, and reproducible. **It is installed on all servers and VSC.** Some applications are installed as containers, e.g. `userservices containers`
Usually commands `apptainer` and `singularity` can be used interchangeable.
Some common commands:

`apptainer help`	Show help
`apptainer run docker://alpine`	Run alpine linux from dockerhub
`apptainer pull docker://alpine`	Pull image from dockerhub
`apptainer build image.sif docker://alpine`	Build image and save in file
`apptainer shell image.sif`	Interact with the container
`apptainer exec image.sif date`	Execute `date` inside container
`apptainer run image.sif`	Run default app from container

More complex things can be done, and are documented a bit here: gitlab.phaidra.org/imgw/singularity

13 Recipes

13.1 Build an app with modules

When developing an application, there are usually some **dependencies or libraries** that you do want to use, but not deliver. Other people shall install these themselves. This is very common in software development. **And a big mess.** HPC centers use **environment modules** to make sure that libraries are available in different versions. These modules set the following environment variables:

- `LIBRARY_PATH`
- `INCLUDE` or `CPATH`

Therefore you should include these in your `Makefile` and make use of these paths during the build process.

Makefile with environment modules

```
# use the environmental variable $CPATH
# split the paths separated by :
INC = $(subst :, ,$(CPATH))
# add a -I/path/to/include
INC := $(INC:%=-I%)
# use the environmental variable $LIBRARY_PATH
LIBS = $(subst :, ,$(LIBRARY_PATH))
LIBS := $(LIBS:%=-L%)
```

13.2 GNU Fortran compiler options

`-fdefault-real-8`	double precision real
`-fbounds-check`	check array bounds
`-fbacktrace`	call chain traceback
`-fconvert=big-endian/little-endian`	convert little/big endian
`-O0`	default optimization
`-O3`	highest optimization
`-frecord-marker=8`	Length of record marker for unformatted files

13.3 Intel Fortran compiler options

`-r8`	double precision real
`-check`	check array bounds
`-backtrace`	call chain traceback
`-convert big_endian/little_endian`	convert little/big endian
`-O2`	default optimization
`-O2 or -O3 or -fast`	highest optimization
`-mcmodel=medium`	Memory Model