Organizing your Workflow
Keeping your workspace organized
Motivation: Reproducibility and Traceability

- Can I **reproduce** my workflow
  - Tomorrow?
  - 1 month from now?
  - 1 year from now?
- Give a set of analysed results – can I **trace back** my workflow all the way to the beginning?
Your environment and .bashrc

- Tip: try to keep your login environment clean (don’t pollute your .bashrc)
Tip: try to define your environment based on what you are currently working on.

- Use (carefully) python environments
- `source` a script that defines the environment for your current work
# OpenIFS configuration
# Sets correct user environment
# e.g. source ./oifs-config.sh
# If your site uses modules, then load the correct
# module environment first.
module purge
module load buildenv-intel/2018.u1-bare
module load grib_api/1.24.0-nscl-intel-2018a-eb

# OpenIFS version: IFS CYCLE and RELEASE
# Must be upper case. Used to set directories.
export OIFS_CYCLE=CY40R1
echo "Environment for OpenIFS: $OIFS_CYCLE"

# OpenIFS home. Set this to the directory
# containing this file (and 'src', 'bin' etc)
export OIFS_HOME=${PWD}
echo "OpenIFS home directory: $OIFS_HOME"

# OpenIFS data directory.
# Parent directory of the climatology files, vtables, rtables etc.
export OIFS_DATA_DIR=${OIFS_HOME}/ifsdata
echo "OpenIFS top-level data directory: $OIFS_DATA_DIR"

# Location of grib-api (or eccodes) directory (may be set by module)
export OIFS_GRIB_API_DIR=${GRIB_API_DIR}
echo "OpenIFS GRIB_API directory: $OIFS_GRIB_API_DIR"

# OpenIFS compilation environment
export OIFS_COMP=intel_mkl_impi  # gnu, intel or ccb (see make/cfg)
export OIFS_BUILD=opt  # opt, noopt, or nansC (see make/cfg)

# PATH.
# Add OpenIFS supplied FCM and grib-api to user path
export PATH=$PATH:$OIFS_HOME/fcm/bin:$OIFS_GRIB_API_DIR/bin
echo "Added OpenIFS FCM and GRIB_API to user path"

# Create some useful aliases
echo "Creating command aliases for OpenIFS..."
alias omake="fcm make -v -j4 -f $OIFS_HOME/make/oifs.cfg"
alias omakenew="fcm make --new -v -j4 -f $OIFS_HOME/make/oifs.cfg"
alias oenv='env|grep OIFS_'
Remember tab completion

- Tab completion means it is practical (recommended) to use descriptive names for files and folders
- Tips for file naming
- What not to do (file naming)
- Tips for folder structures
File, folder organization and annotation

- Tip: Try to avoid reusing run folders

```
my-run-folder  ── output-01
               ▼
               |   ▼
edit        my-run-folder  ── output-01
               |   |   ▼
               |   |   |   ▼
               |   |   |   |   ▼
               |   |   |   |   |   ▼
               |   |   |   |   |   |   ▼
               |   |   |   |   |   |   |   ▼
               |   |   |   |   |   |   |   |   ▼
               |   |   |   |   |   |   |   |   |   ▼
               |   |   |   |   |   |   |   |   |   |   ▼
               |   |   |   |   |   |   |   |   |   |   |   ▼
               |   |   |   |   |   |   |   |   |   |   |   |   ▼
               |   |   |   |   |   |   |   |   |   |   |   |   |   ▼
               |   |   |   |   |   |   |   |   |   |   |   |   |   |   ▼
               |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   ▼
               |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   ▼
```

```
output-01
output-02
output-03
```
File, folder organization and annotation

my-run-folder-01 — output-01

copy & edit

my-run-folder-01 — output-01
my-run-folder-02 — output-02

copy & edit

my-run-folder-01 — output-01
my-run-folder-02 — output-02
my-run-folder-03 — output-03
File, folder organization and annotation

- Tip: Don’t worry about deep folder structures
  - You can always use **symbolic links** or **aliases** to help navigate efficiently
Tip: It’s never too late
  ▪ `mv` (rename) within a file system is an efficient, atomic operation
File, folder organization and annotation

- **Tip**: get into the habit of annotating everything – be kind to the future you
  - Comment your code
  - Comment your scripts (incl. run scripts)
  - Add README files in your folders
  - ...
Version control tools can be helpful, especially in shared workspaces

- If you don’t have any particular preferences, consider git
- If you are not familiar with git check out Software Carpentry (or Code Refinery in the Nordics)
Workflow management
If you follow a sequence of steps when working you have a workflow.

If you repeatedly follow the same steps (workflow) it can pay to manage your workflow.

- repeatability
- convenience
- sharing
Workflow management: Tools

- A notebook
  - paper
  - electronic
- shell script
- Jupyter notebook
Workflow management: Tools

- Kepler (fusion research)
- Pegasus
- Galaxy (life sciences)
- Cylc, Autosubmit, ecFlow (Earth sciences)
- Taverna, COMPSs (Astrophysics)