File naming conventions

This page defines the filename and directory naming conventions used in lab repositories. These conventions apply to new repositories and new files. Existing repositories are not retroactively renamed unless there is a strong practical reason to do so.

General principles

- Use descriptive names.
- Use only lowercase letters (a-z), digits (0-9), hyphens (-), and underscores (_).
- Avoid spaces and other special characters.
- Use **hyphens (-)** for named entities intended primarily for human reading.
- Use **underscores (_)** for technical identifiers intended primarily for software, programming languages, databases, GIS, and other computational environments.

Documentation and manuscripts

Documentation files, manuscripts, figures, tables, LaTeX macros, and other document-related files use **hyphens** to separate words.

Examples:

```
main.tex
cover-letter.tex
cover-letter.pdf
review-response.tex
response-macros.tex
fig-study-basin.tex
tab-algorithms.tex
software-availability.tex
```

Hyphens improve readability and match conventions commonly used in documentation, URLs, and open-source repositories.

Command-line tools

Command-line interface (CLI) programs use **hyphens** in executable names because they are user-facing commands.

Examples:

```
flow-direction
longest-flow-path
mesh-builder
```

Workflow scripts

Workflow scripts are treated as command-line tools and therefore also use **hyphens**.

If the execution order matters, a numeric prefix may be used.

Examples:

```
1-download-files.sh
2-preprocess-files.sh
3-run-model.sh
4-postprocess-results.sh
```

Directory names

Directory names use **hyphens**, unless the directory represents a software package that must follow programming naming rules.

Examples:

```
submission-docs/
review-response/
fig-study-area/
build-scripts/
```

Generated files

Generated files should normally keep the same base name as their source files.

Example:

```
cover-letter.tex → cover-letter.pdf
```

Software source code

Software source code files use **underscores** to separate words. This ensures compatibility with programming identifiers and module imports.

Examples:

```
flow_direction.py
longest_flow_path.py
mesh_builder.cpp
grid_utils.h
```

Technical identifiers

Technical identifiers use **underscores** to separate words. This ensures compatibility with databases, GIS layers, raster algebra, and similar computational environments.

Examples:

```
gcn10_amc_i.tif
nm_hydro_basins.gpkg
```