OAK RIDGE National Laboratory

SCALE New Documentation Strategy

2021 SCALE Users' Group Workshop

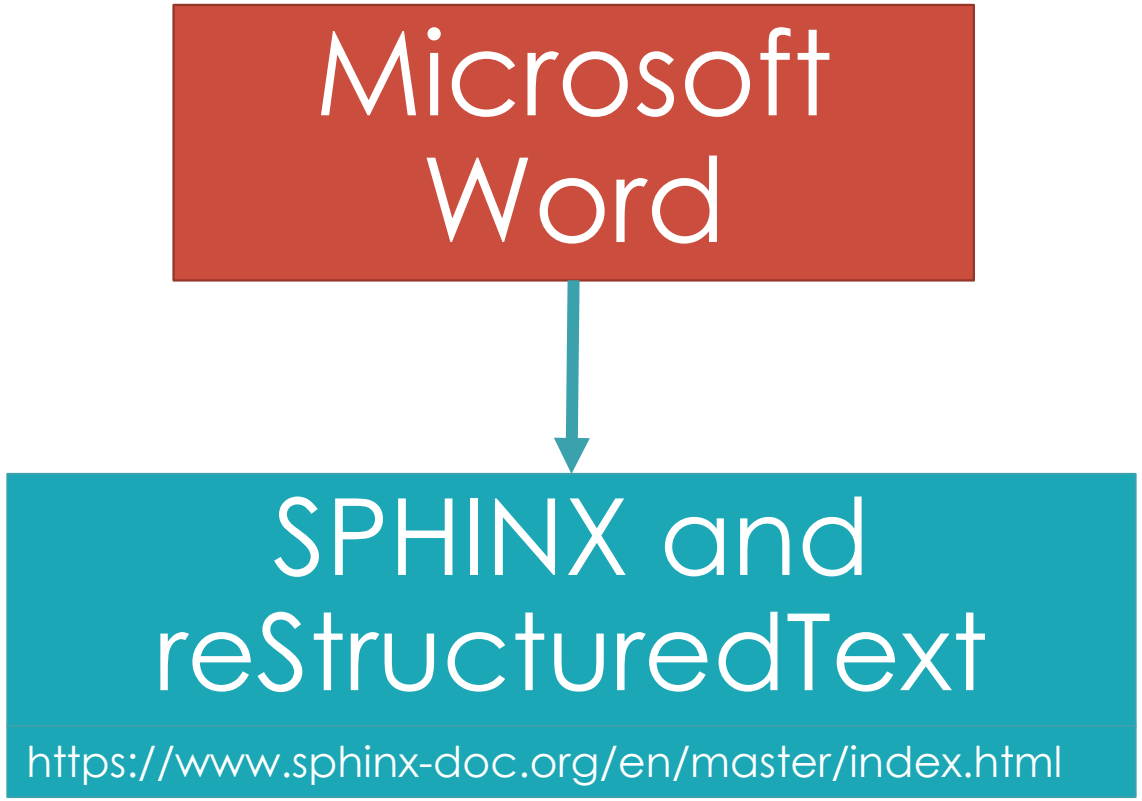
Presenter: W. Wieselquist

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

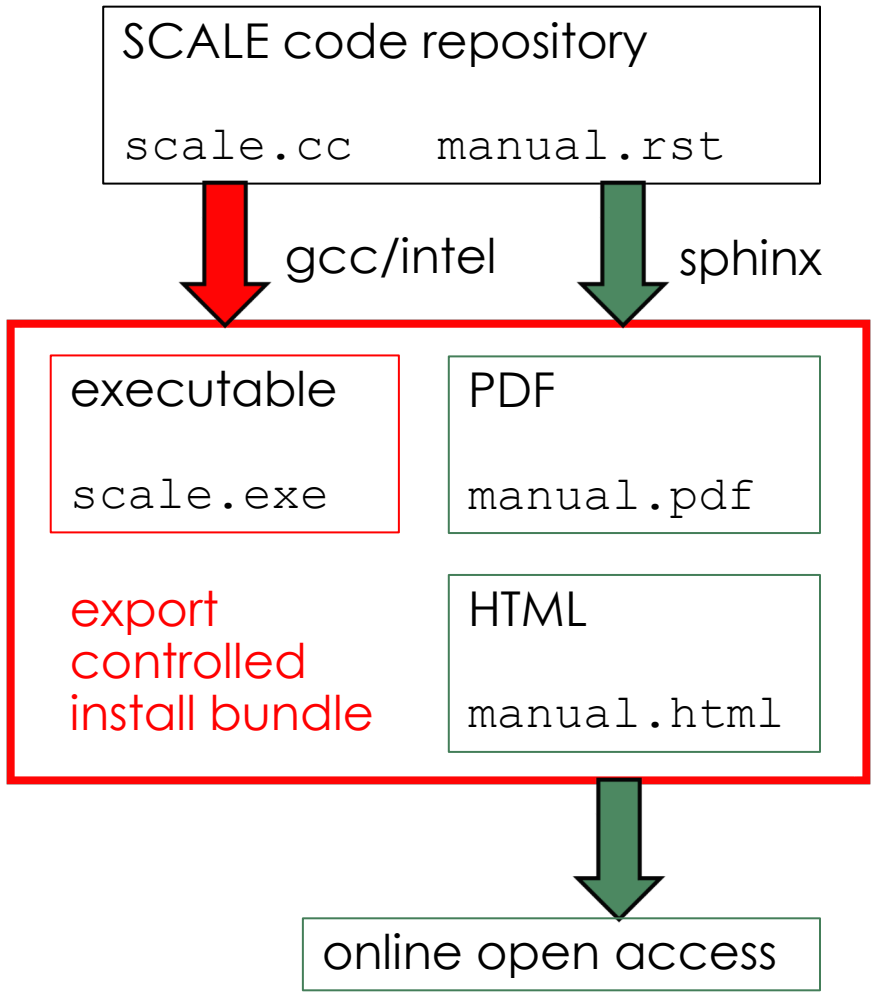
Overview

- Previous situation
  - 2700-page MS Word document
  - managed in sections
  - merging/syncing changes was difficult
  - maintaining section cross links was difficult
- Goals
  - Improve Quality Assurance (QA) processes by integrating code development and documentation updates
  - Be able to deploy manual updates with beta releases



What does ReStructuredText offer?

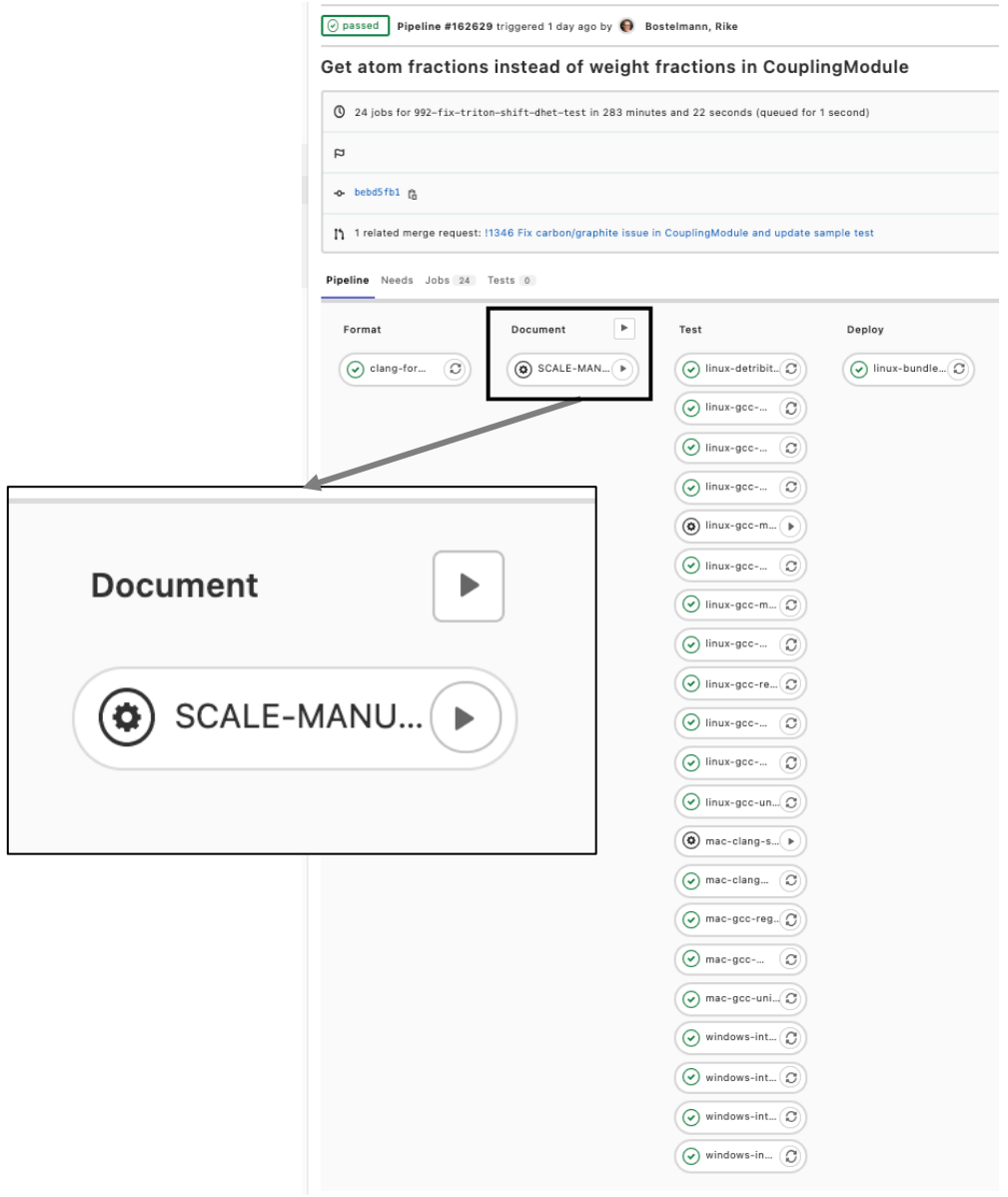
- Used in python documentation and readthedocs.io
- Text-based format which can live in Git repository alongside source code
- Unlike LaTeX (also text-based), simple formatting looks simple (e.g. lists and headers)
- Can produce HTML and PDF from the same source



New development process

- Manual can be updated as part of any QA case
- Developers can review changes to the manual and regenerate the manual as part of normal continuous integration
- Every beta release a new manual will be generated
  (along with fix/enhancement listings)

Gitlab testing pipeline



Moving Forward

- For SCALE 6.3
  - finished converting the manual in early 2021
  - adding new code/feature descriptions
  - keeping basic structure and format of the 6.2 manual
  - develop strategy to post discovered bugs and fixes in 6.3.1, 6.3.2, etc. to open-access website
- For SCALE 7.0
  - integrate "what's new" section into manual based on finished QA cases
  - revisit structure of the manual
  - during 7.0 beta releases, even without SCALE license can view what's new/what's been fixed

Example QA Case

Create initial HDF5 format for TRITON and Polaris nodal data

Description

Create initial HDF5 format for TRITON and Polaris nodal data.

Executive Summary

An initial HDF5 format to store nodal data generated by TRITON and Polaris was created. The output HDF5 includes the same data as the T16 and X16 files but in a more organized, machine-parsable format. The generation of HDF5 files in addition to the traditional T16 files was enabled with Polaris and is planned to be enabled in TRITON in a future enhancement.

Polaris input options have changed:

- `OPT PRINT XFile16` no longer outputs both `X16 and T16` files, but only a `T16` file.
- A new option of `OPT FG ArchiveNodal="NONE"|["MACRO"]` and `OPT FG ArchiveNodalData=["T16"]|"X16"|"H5"` has been added.

Context

The TRITON and Polaris sequences provide few-group cross section data for use in nodal diffusion codes (nodal data) in `T16` (ASCII text) and/or `X16` (binary) files. It is planned to deprecate the `X16` (binary) file and to additionally offer nodal data in an HDF5 format to make it easier for end-users to parse the output and to have an structure that allows for an easy future expansion of data trees/subsections.

This enhancement creates an initial HDF5 format and enables the generation of nodal data in this format for Polaris. Polaris' `PRINT` option was updated so that the user can choose the output formats in a modular, and more intuitive fashion. The generation of HDF5 nodal data files in TRITON is planned as a future enhancement.

Details

Nodal data is hierarchal in nature. The data branches out by depletion step, energy group, isotopes, etc. This makes it appropriate for it to have a structured, hierarchal data format consisted of 2D (and 3D) matrices. Also, the HDF5 format makes it easier to expand the set of data output by Polaris easier (compared to T16).

Create a standalone HDF5 writer/reader class in Fortran, which is then used by `Thunderball/XFILE016.f90` to write the nodal data output from Polaris to HDF5. In `Polaris/Core/Polaris.f90`, the `process` function of `Thunderball/Statepoint_Manager_XFILE016_M.f90` is called, which uses functions in `Thunderball/XFILE016.f90` to output the user-defined file formats.

The output HDF5 follows a tree format with the convention:

```
/
    /metadata (dimensions, burnup history, isotope list)
    /statepoint_set1 (nomenclature `depletion-pass_branch_statepoint`)
        /cross section data (matrices)
    /statepoint_set2 (e.g. 1_0_0, 2_0_0)
    /...
```

A full `h5dump` of the output HDF5 file can be found here. The content of data has not changed from t16 files. A major reorganization (adding / removing which cross sections to include) is planned as future development.

Testing

Testing was done in two tiers:

1. Testing the generic HDF5 writing / reading capability in Fortran
2. Testing the HDF5 output implementation in Polaris