Using MELCOR on Linux and Open Source tools

Petr Vokáč, petr.vokac@ujv.cz
ÚJV Řež, a.s.

10th EMUG, 25.-27. 4. 2018, University of Zagreb, Zagreb, Croatia

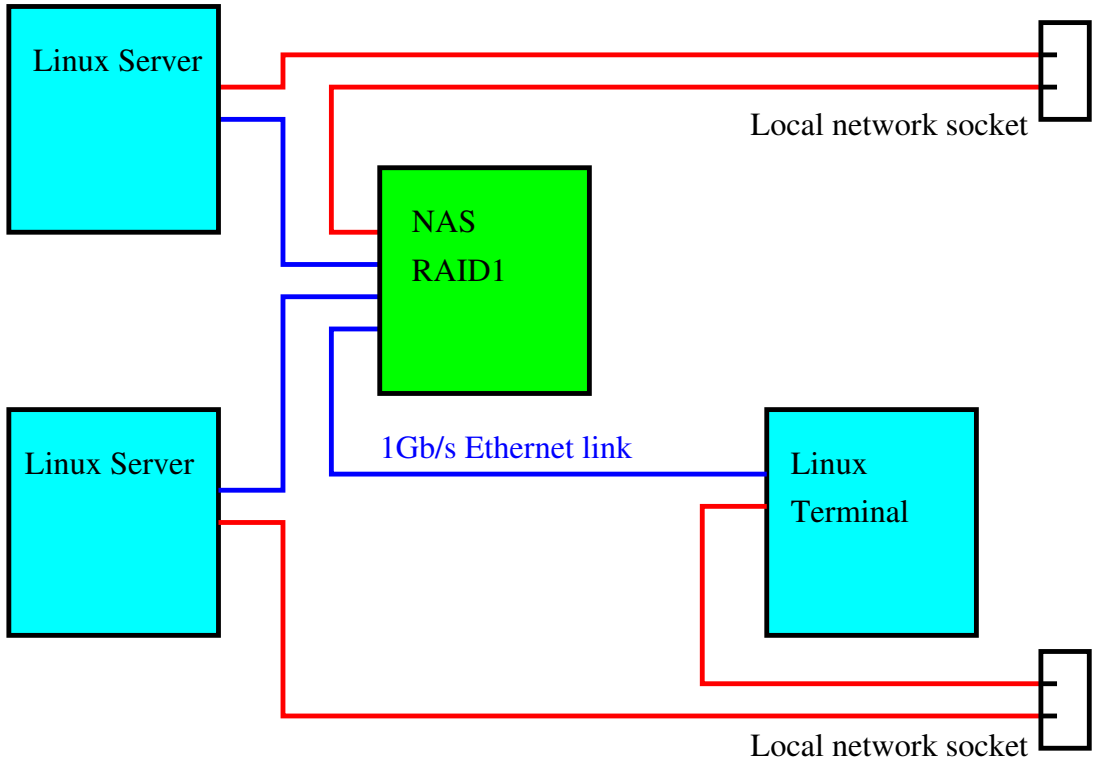
Contents

- Linux environment for team work
- Open source tools (ten years after)
- Input format discussion

Linux environment for team work — version 1

- NAS with RAID1 used to share common data (NFS for Linux, SAMBA for Windows)
  4x 1Gb/s Ethernet card
- Linux Servers provide shared calculation power
  2x 1Gb/s Ethernet card
- Dedicated 1Gb/s Ethernet connection between each server and NAS
- Each machine conected to the local network

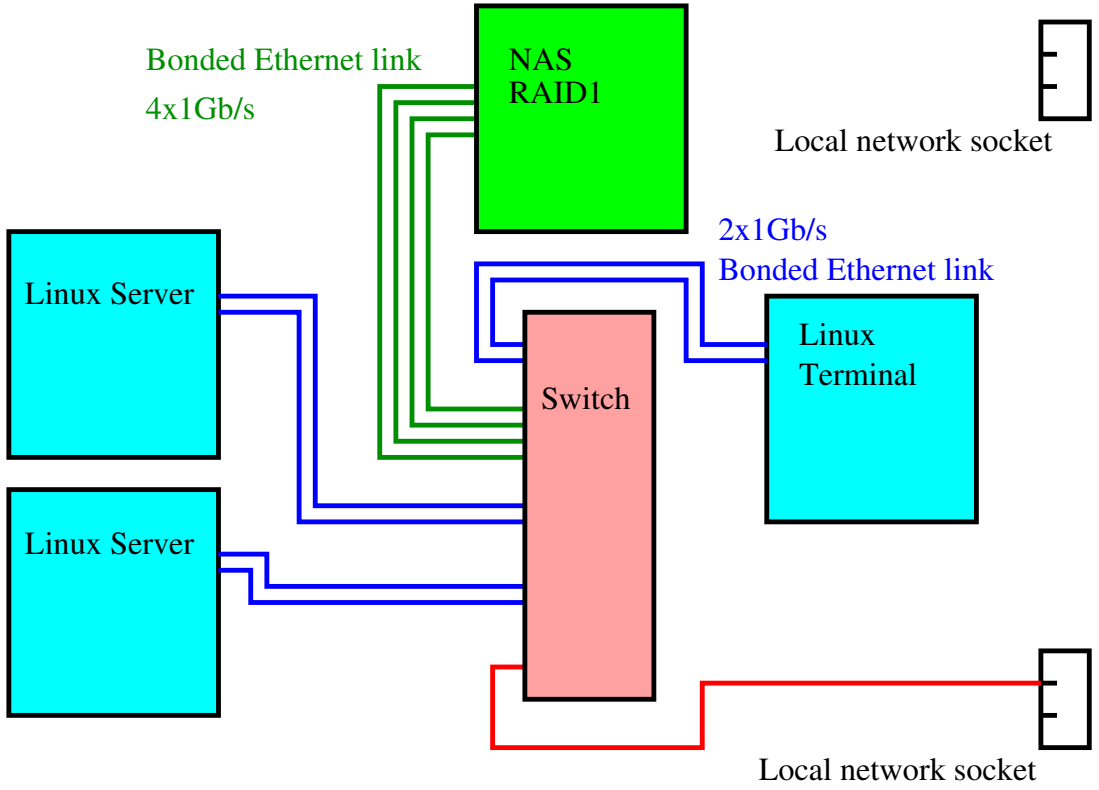
Spurious errors occured: MELCOR failed to write output to temporarily unavailable network storage.



Linux environment for team work — version 2

- NAS with RAID1 used to share common data (NFS for Linux, SAMBA for Windows)
  4x 1Gb/s Ethernet card
- Linux Servers provide shared calculation power
  2x 1Gb/s Ethernet card
- **Switch**

- Each machine is conected to switch using agregated Ethernet links
- Single connection to the local network



Finally, it works well.

Open source tools for MELCOR on Linux

Development started in 2008 and the first version was presented at MCAP 2009.

Vingt ans après, Autor: unknown – Internet Archive, Public domain

What is new?

2017: https://github.com/petrvokac/meltools:

- packages (`*.tar.gz`) for download
- documentation in Wiki

To make local copy type (in empty folder):

```
git clone https://github.com/petrvokac/meltools
```

Readptf&Tranptf

Problem with missing information about total number of values in record solved ⇒

- when the last variable is a vector, it is interpreted correctly
- output produced by `Tranptf` is compatible with other tools (e.g.: ATLAS).

Open issues:

- Can be structure of MELPTF records changed (e.g.: after restart)?

  In this case `Readptf&Tranptf` would not work correctly.
- Names of components introduced in MELCOR2 is supported just for CVH and FL (and this feature was not tested sufficiently).

  Each component type need specific coding, which is cumbersome.
- `Tranptf` errors were recently experienced for plotfiles larger ∼15GB

Python tools

- all sripts were converted to Pyhton3 (currently I am using 3.5)
- no installation procedure is provided but:
  - each package has similar structure: main "executable" script(s) and subfolder with modules
  - it is enough to copy everything somewhere on `PATH` (I use `/usr/local/bin` on server)

Decay heat input preparation

Structure changed:

- current package: `melendf-170915.tar.gz`
- main script `melendf/melendf.py`
- data subfolders:

  data-endf — raw ENDF data downloaded from https://t2.lanl.gov/nis/data/endf/decayVII.1.html

  data-pickle — filtered data ENDF stored in Python pickle format — it does not provide any performance benefit — data from original format are used directly

- configuration file examples in subfolder `tests`

Reader

It is not distributed — it has not reached user friendly state — it newer will, it has meaning only for Python programer as API (and currently it is not stable).

I expected that with MELCOR2 the input structure will be more uniform — unfortunately it is not the case. Are multiple variants of input format really needed?

Anyway, for me it provides programing framework to:

- generate variable nodalization (mainly for COR and associated CVH/FL/HS):
  VVER-440/213, OECD-SFP, NUGENIA Air-SFP,
  QUENCH-11 (under preparation)
- analyse and modify input prepared by others
- M1.8.6 → M2.x conversion

Discussion on user input formats

Most of tools I use need some user input data for configuration — format of these data differs though it is generaly the same ⇒

- burden of repetitive programing
- burden on user to learn different input format for each tool

After little googling → JSON found:

- It is standardized.
- Libraries to read/write data in the format are available for most computer languages (including Python and Fortran).

JSON example

Download: json-example-180410.tar.gz

- `fjw.f`: Fortan code to generate `example.json`
- `pjm.py`: Python code to read, modify data and write back `example.json`
- `fjr.f`: Fortan code to read and interpret `example.json`

Note additional libraries needed: FoBiS-2.2.8, json-fortran-6.1.0, simplejson (I use py35-simplejson @3.13.2)

Conclusion: I will use JSON to standardize config files for all (new) Python tools.
Question: wouldn't it be nice if the MELCOR input is formated in a similar way?

```
{
  "inputs": {
    "t0": 0.10000000000000001E+0,
    "tf": 0.11000000000000001E+1,
    "x0": 0.9999E+4,
    "integer_scalar": 787,
    "integer_array": [
      2,
      4,
      99
    ],
    "names": [
      "aaa",
      "bbb",
      "ccc"
    ],
    "logical_scalar": true,
    "logical_vector": [
      true,
      false,
      true
    ],
    "float_vector": [
      0.18E+1,
      0.10000000000000001E-4,
      0.7E+1
    ]
  }
}
```