

# File Formats: Overview

Micael Oliveira

CECAM Tutorial: Basic techniques and tools for development and  
maintenance of atomic-scale software

February, 11-18 2008, Lyon

## Some basic ideas

- A file format is a particular way to encode information for storage.
- It is about the structure of the information:
  - layout
  - type of data (string, integer, ...)
  - relationships between the different elements
- Allows programs to recognize, access, or store data within a file.

# Why do we want file formats?

Basically... **to share information!**

- For the same or similar applications
- Between different applications
  - The output of one application can be the input of another one (e.g. pseudo-potential generator → DFT code)
  - Post-processing: visualization, analysis...

This is only possible with file format specifications.

## Important Issues

- Portability
- Flexibility and extensibility
- Performance
- Software portability/maintainability

# Portability

- Different platforms
  - Endianness
  - 32/64 bit
- Character encoding
- Programming language  
(e.g. try to parse a tree structure with Fortran 77...)

# Flexibility and extensibility

- Optional or conditional content.
- Possibility to access specific chunks of the data.
- File format suitable for storing different kinds of data  
(e.g. using the same format to store a wave-function or a potential)
- Possibility to extend or change the format while ensuring backward compatibility.
- Ideally: self-describing formats.

- Volume of data to be stored/transferred.
- Reading/writing speed.

## Software Portability/Maintainability

File Format Specification = data sharing between different applications

### Really?

- Implementing a new format in a program is not always easy.
- Formats may change -> programs have to be updated.

### Solution

- Libraries + Application Programming Interface (code reuse!)

File Formats + API/Libraries = Happy Developers

# Types of File Formats

## Text

- Plain text
- Tagged text (e.g. HTML, XML, T<sub>E</sub>X)

## Binary

- Raw binary
- Structured binary: header or records
- Portable binary: specific byte order and access method

## Text vs. Binary

Binary	Text
More compact (usually)	Human-readable (editable)
Not all data is textual in nature	Portable
Faster to access	

# File Formats and the ETSF/NANOQUANTA

- One of the main goals of NANOQUANTA was to achieve effective code interoperability and portability of essential data file.
- Global solution: formats that deal appropriately with the self-description issue, i.e. XML and NetCDF.

## NetCDF

- Crystallographic data
- Densities and potentials
- Wavefunctions

(see Caliste)

## XML

- Pseudopotential and PAW

## Case study: pseudo-potentials

### The jungle of pseudo-potentials...

- psf
- upf
- hgh
- fhi
- cpi
- ncpp
- vdb
- ...

Clearly too many!

## Case study: pseudo-potentials

### The jungle of pseudo-potentials... and codes

- psf
- upf
- hgh
- fhi
- cpi
- ncpp
- vdb
- ...
- FHI98
- APE
- opium
- atom
- siesta
- abinit
- octopus
- PWscf
- ...

## Case study: pseudo-potentials

### Problems

- Difficult to share pseudo-potentials.
- Lots of code duplication (both to write and to read the files).

### How to make things worse:

- Not all the formats contain the same information.
- Not all the codes expects/use exactly the same information
- Large database of existing pseudo-potentials.

## Case study: pseudo-potentials

### Key issues for the new format

- Should be easy to implement and maintain.
- Should be flexible (remember: not all the codes expects/use exactly the same information)
- Performance is not important.

### Solution

- Long-term: standard XML format.
- Short-term: pseudo-potential converter