# The Initfile Manual

Simon Praetorius

May 30, 2012

AMDiS is equipped with a parser for parameter files to control simulations, e.g. set material parameters, define the timestep length, set meshes for each problem and configure the number of global refinements. These parameter files (or init-files) are simple text files in a special syntax, that is described here.

The .xml-file `amdis.xml` in the `tools/misc` sub-directory of AMDiS is a syntax-file for the Kate editor, that should be copied to the user directory `~/.kde/share/apps/katepart/syntax` in order to activate the highlighting scheme.

## 1 General notation

`%...`

Init-files have the suffix `.dat.Xd`, where `X` is in $\{1, 2, 3\}$. The general definition of a parameter is done by

```
parameter_name:  parameter_value  % a comment
```

where the '`:`' sign is the delimiter between parameter name and its value and the '`%`' sign indicates a starting comment and is not allowed as part of a parameter-name or value.

`#include`

You can include other init-files by

```
#include <filename>  % or
#include "filename"
```

where `filename` must be relative to the path of the calling program. All parameters in the included init-file overwrites the parameters with the same name defined before the include statement.

`${...}`

One can use parameters, defined previously, as variables:

```
parameter1:  value1
parameter2:  $parameter1
parameter3:  ${parameter1}
parameter4_${parameter1}:  value2  % variable in parameter-name
% ⇒ parameter4_value1 = value2
```

where you have to use brackets '{' and '}', if the variable name contains signs other than {a-z, A-Z, 0-9, _}. Variable replacement is simple string replacement, no cast or evaluation of the variable is performed before it is inserted!

If you set a parameter twice, the last value is the one that is return on evaluation:

```
parameter1:  value1
parameter1:  value2  % ⇒ parameter1 = value2
```

## 2 Value format

The values of the parameters can be numbers, strings, vectors or arithmetic expression, or some combination of that:

```
parameter1:  1.0  % double
parameter2:  1  % int
parameter3:  Hello World!  % string
parameter4:  [1,2,3]  % vector<int>
parameter5:  [1,2,3; 4,5,6; 7,8,9]  % vector<vector<int>>
parameter6:  1/2  % arithmetic expression ⇒ 0.5
parameter7:  2*${parameter6}+sin(0.5)  % complex arithmetic expr.
parameter8:  m_pi+m_e  % some constants(π + e)
% now all together:
parameter9:  [2*m_pi, sin((${parameter7})*m_e)/2]
```

Sometimes it is necessary to evaluate an expression before it is used as variable in an other parameter, e.g. if you want to insert a numeric value into a string, e.g. a filename, it is better to use ”0.5” than ”1/2”, since the last one is interpreted as sub-directory delimiter. To insert the value of ”1/2” into a string it has to be evaluated before. This can be done, by: `$(...)`

```
parameter1:  1/2 + 1/2
parameter2:  Hello ${parameter1} World!
% ⇒ parameter2="Hallo 1/2 + 1/2 World!"
parameter3:  Hello $(${parameter1}) World!
% ⇒ parameter3="Hallo 1 World!"
parameter4:  Hello $(1/2 + 1/2) World!
% ⇒ parameter4="Hallo 1 World!"
```