Proposal for Modules Contexts

Felix Friedrich Florian Negele

June 25, 2008

1 Motivation

The source code of the current A2 system consists of over a thousand modules of which one third belongs to the legacy Oberon sub-system. In order to distinguish their membership, some names of the modules belonging to the newer A2 system were prefixed by “Aos” (its previous name). Unfortunately this naming convention has several drawbacks:

- The membership of modules with unprefixed names is not recognisable at first sight and confuses new users.
- The existing prefixes do not reflect and even reverse the intented priority of the modules within the system.
- New modules have to be prefixed as most names are already taken by modules that belong to Oberon.

As the AOS system was currently renamed to A2, modules have again to be renamed. We therefore propose the introduction of a more generic concept that avoids all of these shortcomings.

2 Contexts

A *Context* acts as a single-level namespace for modules. It allows modules with the same name to co-exist within different contexts. Each module belongs to exactly one context. The pseudo-module SYSTEM is available in all contexts but does not belong to any of them. There are currently two contexts intended to be available for the user[1]: *Oberon* and *A2*.

2.1 Language extensions

As modules should be able to import modules from different contexts at the same time, classifications based on a compiler-switch or different source-code file names are not sufficient. Therefore the programmer should be able to specify the context of a module within its code. We propose the following fully backwards-compatible syntax-extension:

[1]Although this concept is more generic than actually needed, regular application programmers shall actually not be encouraged to define and use their own contexts in the medium term.

```
Module := 'MODULE' Identifier ['IN' Identifier] ';' ...
```

The optional identifier after keyword `IN` specifies the name of the context this module belongs to.[2] The context defaults to A2 if it is omitted. We additionally suggest the following syntax-extension for the import section of a module:

```
Import := Identifier [':=' Identifier] ['IN' Identifier].
```

The optional context specification tells the compiler in which context to look for modules to import. This allows to use A2 modules from within Oberon and vice versa. The context defaults to the context of the module if it is omitted by the programmer.

2.2 Run-time extensions

For the execution of commands, the runtime-environment implicitly specifies the correct context. Only the modules within the same context shall be considered when a command is searched for and executed. This also avoids the annoying problem of loading the complete Oberon system when some text displayed in A2 is middle-clicked accidently.

2.3 Naming conventions

The filenames of module files and their corresponding object-files are prefixed by the name of their context followed by a dot. As most of the files will belong to the default A2 context, this prefix shall be omitted for a better overview.

Prefixing module files helps the programmer to be able to distinguish the membership by looking at a filename instead of having to browse its contents. The prefix for object-files is needed by the compiler and the runtime-system in order to dynamically load the correct modules.

3 Conclusion

The introduction of the context concept requires only a few and very simple modifications of the language, compiler and the runtime-system and is fully backwards-compatible to the previous solution. It even offers a more generic solution the actual problem asked for. It could therefore even be used to assemble other big software packages like GUI applications and libraries in the long term.

[2]This notation was chosen because it clearly reflects the semantics and it does not introduce yet another keyword.