ORU.doc

COLLABORATORS			
	TITLE : ORU.doc		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		December 7, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

ORU.doc

1.1 ORU.doc

```
   $RCSfile: ORU.doc $
Description: Documentation for the ORU utility

 Created by: fjc (Frank Copeland)
  $Revision: 2.3 $
    $Author: fjc $
      $Date: 1995/01/25 23:36:25 $

Copyright © 1994, Frank Copeland.
```

ORU is the Oberon-A recompilation utility. It is used when the definition of a module (the symbols exported by it) changes and it becomes necessary to recompile all other modules that directly or indirectly import it. The user specifies the module whose definition has changed and the directories to search ORU produces a text file containing the paths of the source files of all the modules affected by the redefined module.

Distribution	Copyright and distribution

Running ORU ...
- From the CLI
- From the Workbench
- From FPE

The Author	Contacting the author
Bugs & Suggestions	Reporting bugs and suggestions
Changes	Changes since the last release
To Do	Bugs to fix and improvements to make
Release History	Release history of the program

1.2 Distribution and Copyright

ORU is part of Oberon-A and is:

1.3 Running ORU from the CLI

Usage: ORU {option} <module>|ALL

Options: NOTCD
{SYM | SYMBOLS <path>}
{SRC | SOURCE} <path>}
DST | DESTINATION <path>
EXT | EXTENSION <extension>
{WITH <file>}

Purpose: To generate a file listing all source files affected by a redefined module.

Path: Oberon-A:C/ORU

Given the name of a module, ORU attempts to generate a list of dependant modules by scanning the file system for symbol files. It then attempts to locate the source files for the dependant modules. If successful, it generates a batch file named "<module>.bat" in the current directory containing a list of the source files it has located.

If the keyword ALL is given instead of a module name, ORU will determine the dependants of all the modules whose symbol files it finds, as if they had all been redefined.

If the NOTCD parameter is specified, ORU will ignore the current directory when searching for symbol and source files. In this case at least one SYMBOLS and one SOURCE parameter must be specified for the program to have any effect. If it is not specified, ORU will always search the current directory first before attempting to search any other directory.

If a SYMBOLS parameter is specified, ORU will search the given directory for symbol files. If a SOURCE parameter is specified, ORU will search the given directory for source files. If a DESTINATION parameter is specified, ORU will place the batch file it generates in that directory instead of in the current directory.

If an EXTENSION parameter is specified, ORU will append the given string to the name of a module when it searches for the module's source file. If it is not specified, a default extension of ".mod" is used.

If a WITH parameter is given, ORU will open the file and treat its contents as if they were parameters passed on the command line. Parameters may be enclosed in quotes and seperated by whitespace and newlines. WITH and DST parameter are not allowed inside a WITH file.

Example:

Assuming Module1 is imported by Module2 and Module3, the command:

```
ORU Module1 SYM Oberon:Modules/Code/ SRC Oberon:Modules/
  DST Oberon:Modules/Code/ EXT .src
```

might produce the batch file (Module1.bat) below:

```
Module1.src
Oberon:Modules/Module2.src
Oberon:Modules/Module3.src
```

1.4 Running ORU from the FPE utility

A tool button in the FPE window can be configured to run ORU (see FPE.doc). In the button editor, set the Command field to the full path name of the ORU program. Set the Arguments field to "!M" plus any arguments that are desired. Specify a console window as the Console field. Put at least 4000 in the stack field.

For example:

```
Command="DH1:Oberon-A:C/ORU"
Arguments="!M SYM OLIB: DST Code/ EXT .src"
Console="CON:0/11/540/189/ORU working.../CLOSE/WAIT"
Stack=4000
```

A tool button can also be configured to run the compiler using a batch file produced by ORU as input.

For example:

```
Command="DH1:Oberon-A/OC"
Arguments="NS SYM Code/ DST Code/ BATCH !M.bat"
Console="CON:0/11/540/189/Compiling.../CLOSE/WAIT"
Stack=10000
```

To recompile a module and all its dependants:

1. select the module in the Module gadget.
2. select the file extension from the Files gadgets.
3. click on the tool button ORU is bound to.
4. click on the tool button the batch compile option is bound to.
5. go and make yourself a cuppa.

1.5 Running ORU from the Workbench

ORU cannot be run from the Workbench. Sorry. This will come in a future version.

1.6 Who is responsible for THIS?

ORU was written by Frank Copeland.

All bug reports, suggestions and comments can be directed to:

Email : oberonwossname.apana.org.au

Snail Mail :

Frank J Copeland
PO BOX 236
RESERVOIR VIC 3073
AUSTRALIA

Remember the J. It saves a lot of confusion at my end :-).

1.7 Reporting bugs and suggestions

You are encouraged to report any and all bugs you find to the author, as well as any comments or suggestions for improvements you may have.

Before reporting a suspected bug, check the file ToDo.doc to see if it has already been noted. If it is a new insect, clearly describe its behaviour including the actions necessary to make it repeatable. Indicate in your report which version of ORU you are using.

1.8 Release History

0.0 Initial version.

0.1 Bug fixes and improvements

0.2 Added the ALL and WITH options.

1.0 Start of revision control. First public release.

2.0 * Changed to generate a batch file to be used with the BATCH option of the compiler.
* OLIB: is now the default symbol file search path.

2.1 * [bug] Module parameter is no longer case-sensitive.
* Appends "/" to paths not ending in "/" or ":".

2.2 Changed symbol file tag in line with the compiler.

2.3 Cleaned up for release.

2.4 Minor modifications.

2.5 Minor modifications.

2.6 Changed to use new interface to module Strings

2.7 Now uses module Out for console IO