FPE.doc

COLLABORATORS

	TITLE : FPE.doc	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 30, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

FPE.doc

1.1 FPE.doc

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

 Created by: fjc (Frank Copeland)
  $Revision: 1.6 $
    $Author: fjc $
      $Date: 1995/01/25 23:29:42 $

Copyright © 1994, Frank Copeland.
```

FPE is a programmer's environment. It is designed to assist a programmer in organising the source code for a programming project and to integrate programming tools from different sources into a single system. It is inspired by the Turbo Pascal style of integrated environment, but it is designed to deal with tools that were not originally intended to work together.

FPE is distributed with the Oberon-A compiler package, but it is by no means restricted to creating Oberon programs. It was originally used with a Modula-2 compiler and there is no reason why it cannot be used by C, Pascal or Assembler programmers. It is highly configurable and creating and loading custom environments is quick and simple.

History Release history

1.2 Distribution and Copyright

FPE is part of Oberon-A and is:

See Oberon-A.doc for its conditions of use and distribution.

1.3 Running FPE from the CLI

Here is a formal description of the FPE program, in the style of the offical AmigaDOS documentation:

Format: FPE [<path>][<project>]

Template: PROGRAM

Purpose: To provide a source code manager and tool integrator.

Path: Oberon-A/FPE

FPE can be run without parameters. If it is passed a parameter, the parameter must be either a directory name or a program name. Either may be a full path name, either absolute or relative to the current directory.

1.4 Running FPE from the Workbench

First select the FPE program icon. Next shift-select either a directory icon or another program icon and double-click on it.

Alternatively, double-click on the FPE icon.

1.5 Getting started with FPE

After reading the rest of this document, work through the following tutorial. It is assumed that you have only just installed FPE and are running from the Workbench. For installation instructions, see Oberon-A.doc.

First, double-click on the FPE icon to run the program. A requester should appear saying something like: "Could not load setup file". Ignore this and continue. Another requester should appear saying: "Could not load FPE.prg". Again, ignore this. You should now see the FPE dialog window. The project name gadget should contain the string "FPE", which

should also appear as the only item in the module list gadget. The tool buttons should all be disabled and ghosted.

Select the "Select" item from the "Project" menu. You should then be presented with a file requester. Use this to navigate through the sub-directories in the Oberon-A directory. Find the "Source/FPE" directory and enter it. Select "FPE" from the file list or type "FPE" into the file name gadget and click on "OK". The disk drive will activate and a list of module names will appear in the module list gadget. These are the modules that make up the FPE program.

Select "Project-Select" again and find the "Amiga" directory. Enter it and make sure the file name gadget is empty before clicking on the "OK" button. A list of Amiga operating system modules will appear in the module list gadget.

Now select the "Setup-Load-Select" menu item. This will display the file requester again, showing the contents of the "FPE:S" directory. Select "DefOberon.fpe" and click on the "OK" button. The tool buttons should now be redrawn and activated. Select the "Setup-Save-Default" menu item to save the setup again as "FPE:S/Default.fpe". This setup will now be automatically loaded whenever the program is run. If you don’t believe me, quit FPE and run it again.

Repeat the same process to load "AltOberon.fpe" and use "Setup-Save-Alternate" to save it as "FPE:S/Alternate.fpe". Use the "Setup-Load-Default" and "Setup-Load-Alternate" menu items to switch between these two setups.

Now, go back to the "Sources/FPE" directory, select FPE as the project name and make sure you are using the default setup. Select the "Project-Create Directory" menu item to create a "Code" sub-directory in the "Source/FPE" directory. Select the file gadget marked "mod". Select each module in the module list in turn, from first to last, and click on the "Compile" button once for each module. Each time "Compile" is clicked, a window will open and the compiler will run in it to compile the selected module. The symbol and object files will be written to the "Code" directory. When all the modules are successfully compiled, click on the "Pre-Link" button. When the pre-linker is finished, click on the "Link" button. When that is finished, examine the "Source/FPE" directory where you should find a new copy of the FPE program. Click on the "Run" button to run the new copy.

The final step before you can start to write and compile your own programs is to install an editor. Select the "Setup-Load-Default" menu item to select the default settings. Select the "Setup-Buttons-Edit" menu item to launch the tool editor dialog. Click the "Active ?" check box to activate the button. Enter the full path name of the editor you wish to use in the "Command" input gadget. The "Arguments" input gadget will already contain the string "!F". Edit this to add any other arguments your editor will require. The "Console" input gadget should be left inactivated and empty. Edit the "Stack" gadget if your editor requires more than 4000 bytes of stack. When you are satisfied, click on the "Accept" button. Repeat the process for the "Errors" tool button, then save the setup. Select the "Setup-Load-Alternate" menu item to select the alternate settings. Edit the alternate setup in the same way and save it as well.

What is the difference between the default and alternate setups? The default setup is designed for managing the development of a specific program like FPE. The alternate setup is designed for dealing with library modules like those found in the "Amiga" directory. The default setup directs symbol and object files to the Code subdirectory of the project directory. The alternate setup directs compiler output to the OLIB: directory and does not provide access to the linker.

You are now ready to set off on your own. Good luck !

1.6 Making use of FPE

FPE works on the basic assumption that a project consists of one or more seperate modules, each of which may be made up of one or more files. A project can be a program, or a library of modules. If the project is a program, it has the same name as the program. If it is a library of modules it is unnamed, but is identified by the name of the directory containing the modules. Each module is named and each file making up a module is named as "<module name>.<extension>". FPE can handle any number of module names and up to four file extensions.

All the files belonging to a given project are kept together in the same directory. FPE always sets its current directory to be the directory containing the project it is currently managing. The current directory when the program starts is remembered and is restored when the program exits.

The FPE window contains a list view for module names, check boxes for file extensions and buttons for tools. The current module is always highlighted in the list view. The project name, current module and current files are used as parameters to the various operations bound to the tool buttons. Which combination of these applies to an operation depends on the operation, and is completely configurable.

Operations that affect the entire project (such as linking the program associated with it) typically take the project name as a parameter. Operations that affect a module, or a specific file belonging to a module, take the module name as a parameter. These might include compiling a module (assuming the source is in the file "<module>.mod"), or determining the dependants of a redefined module. Operations that apply to specific files (such as editing them) use a combination of the module name and the selected file extensions.

A typical sequence of operations might be editing a module's source code and the documentation for it, recompiling it and relinking the program it is part of. The user would first select the project using the "Project-Select" menu. Then she would select the name of the module in the list view, and select the ".mod" and ".doc" check boxes. Clicking on an "Edit" button would run her favourite editor. This would be passed the names of two files, constructed from the module name and the two selected extensions. After making any changes and exiting the editor, she would click on a "Compile" button to recompile the module's source code. This would only be passed the module name, as it already assumes that the file has a ".mod" extension. Assuming no errors, she

would then click on the "Link" button to re-link the program. This would be passed the name of the project, from which the linker can determine the files it needs to include.

See Configuring FPE for details on how operations are passed parameters.

1.7 Projects

A named project is used when managing the modules for a program. The program's executable will have the same name as the project. The list of modules consists of those modules specific to the program, excluding library modules that are re-used in multiple programs. The project must contain at least one module, with the same name as the project. This is the main program module, where execution of the program will start.

The list of modules is read from and saved to a file called <project>.prg in the project's directory.

1.8 Library modules

A project can consist of a collection of library modules which are intended to be re-used in multiple programs. Such a project is unnamed, and is identified by the name of the directory in which the modules are kept.

The list of modules is read from and stored in a file called ".prg" in the project's directory.

1.9 Program reference for FPE

1.10 The main program window

The main program window contains a number of gadgets that are used to control the operation of the program, along with the menus described below.

At the top of the window is a recessed bevelled box labelled "Project" in which is displayed the name of the current project. This may be empty.

Below this on the left of the window is a list view gadget labelled "Modules". This will display a list of module names. The currently

selected module is highlighted. A scroll bar may be used to scroll through the list if it is too large to fit completely in the gadget.

Below this is a group of four check box gadgets collectively labelled "Files". Each gadget is individually labelled with a file extension which may be edited (see "Customising FPE" below). If any of these gadgets is selected, the corresponding extension is combined with the name of the current module and the resulting string may be passed to a program bound to a tool button. If more than one gadget is selected, one file name per gadget is created, seperated by spaces.

In the right half of the window is a double row of text buttons. Each of these buttons may have a program bound to it which is run whenever the button is clicked. The text in the button, as well as the details of the program bound to it, may be edited (see "Customising FPE" below).

1.11 Menus

* FPE

  * About

    This displays a notice giving information about FPE.

  * Quit (right-Amiga Q)

    This item causes the program to exit.

* Project

  * Create Directory ...

    A string dialog is displayed requesting the name of the directory to be created. If a name is entered in the string gadget and the "Accept" button clicked, FPE attempts to create the directory named, in the current directory. If it fails, a message will be displayed informing the user of this. If the "Cancel" button is clicked, no directory is created.

  * Select Project ...

    A file requester is displayed showing the contents of the current directory. The file name gadget is initially emptyj. The user may use this requester to move about within the directory structure and select a project. If the "Accept" gadget is selected, the current directory will be changed to the directory displayed in the file requester and the current project name will be changed to the contents of the file name gadget. If the "Cancel" gadget is selected, the state of the project remains unchanged.

  * Add Module ...

    A string input dialog is displayed, allowing the user to specify the name of a module to be added to the list of modules. The name of the module to be added is entered in the input gadget. Selecting

the "Accept" gadget confirms the module name and adds it to the list of modules. Selecting the "Cancel" gadget exits the dialog without changing the module list.

* Remove Module ...

Removes the currently highlighted module name from the list of modules. A requester is displayed first to confirm that the module should be removed.

* Save Module List

The list of module names is saved in the file "<project name>.prg" in the current directory. If no project name has been selected, it is saved as ".prg" in the current directory.

* Setup

* Load Setup

* Default Setup

Loads the default setup information from the file "Default.fpe". The program searches for this file in the following directories in turn:

the current directory
S
FPE:S
S:

The program remembers where it finds the setup file for the benefit of the Save-Default menu.

* Alternate Setup

Loads the alternate setup information from the file "Alternate.fpe". It searches the same directories as for the default setup file.

* Select Setup ...

Displays a file requester allowing the user to select a setup file to be loaded.

* Save Setup

* Default Setup

Saves the current setup in the file "Default.fpe" in the directory from which the file was last loaded. The default is the "FPE:S" directory.

* Alternate Setup

Saves the current setup in the file "Alternate.fpe" in the directory from which the file was last loaded. The default is

the "FPE:S" directory.

* Select Setup ...

Displays a file requester allowing the user to specify the directory and file in which to save the current setup.

* Edit Tool Buttons

Selecting the name of a tool button from the list of sub-items brings up a dialog allowing the user to edit the details of the program bound to the button. See "Customising FPE" below.

* Edit File Extensions

Selecting a file extension from the list of sub-items displays a string dialog allowing the user to edit the file extension. Changes are reflected in the labels attached to the file gadgets.

1.12 Configuring FPE

The user can customise the tool buttons and file extensions used by FPE. Custom setups can be saved and reloaded very easily.

A tool button can be edited by selecting its current name from the sub-items of the "Setup-Edit Tool Buttons" menu item. This will bring up a dialog window showing the current settings for the button. This dialog contains a number of input gadgets, check boxes and buttons which are explained below.

The "Button" input gadget is used to set the text that appears in the button displayed in the FPE window. The "Command" input gadget is used to specify the name of the program bound to that button. This should be the full path name of the program. The "Arguments" input gadget is used to specify the arguments to be passed to the program when it is run. The "Console" input gadget is used to specify a window to be opened and used as the standard input and output for the program. It is ignored unless the check box gadget next to it is selected. The "Stack" input gadget is used to specify the size of the stack to be allocated for the program. If the "Active" gadget is not selected, the button is disabled. The "Accept" button is used to accept any changes that have been made; the "Cancel" button is used to exit without making any changes.

The "Command", "Arguments" and "Console" input gadgets may contain special escape sequences which are used to include the project, module and/or file names in the string when the tool button is used. These escape sequences consist of a "!" character followed by a "D", "F", "M" or "P". "!D" is replaced in the string by the name of the current work directory. "!P" is replaced by the project name. "!M" is replaced by the currently selected module. "!F" is replaced by a sequence of file names, one per file extension currently selected.

1.13 Who is responsible for THIS?

FPE 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 :-).

1.14 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 FPE you are using.

1.15 Release History

0.0 - 0.4 Early versions written in Modula-2 and named M2Org.

0.5 The first Oberon version, renamed to FPE.

0.6 General re-arrangement of the source code and some improvements.
Improved handling of tool buttons and updated tool button editor.

1.0 Start of revision control. First public release.

1.1 Added "Project-Create Directory" menu item and removed references
to the "Make Code" button in FPE.doc.

1.2 Bug-fixes and minor improvements.

1.3 More bug-fixes and minor improvements.

1.4 arp.library is now only used if dos.library < V37, or asl.library
cannot be opened. This fixes some problems caused by conflicts
between arp.library and dos.library.

1.5 Updated to reflect changes made in Oberon-A Release 1.4.

1.6 Minor modifications.