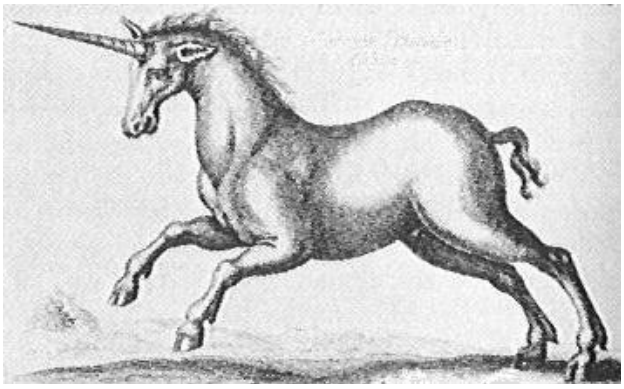
Ui: a Unicon Development Environment

Clinton L. Jeffery and Hani Bani-Salameh
March 11, 2010
Unicon Technical Report #12

Abstract

Ui is a simple integrated development environment (IDE) for the Unicon programming language. Ui makes it simple to edit, compile, run, and debug Unicon programs. Ui runs on both UNIX-based and Windows platforms. Its source code provides various examples of extending and customizing a graphical user interface generated using the IVIB interface builder tool.



http://unicon.org/utr/utr12.html

Department of Computer Science
University of Idaho
Moscow, ID 83844

Contents

1. Introduction

This report describes a simple integrated development environment for the Unicon programming language[1], called Ui. The basic goal for Ui is to make it as simple as possible to edit, compile, and execute Unicon programs. By design and intention, Ui is much smaller and easier to learn than industry behemoths such as Eclipse and Visual Studio. Power users who want a much more ambitious Unicon IDE are directed to Ui's big brother project which is under construction at http://cve.sourceforge.net". That IDE, which includes multi-user collaboration facilities, borrows much from Ui's code and adds many features.

Even Ui's very "simple" goals are complicated by the fact that the tool needs to run on any machine where Unicon runs with graphics facilities enabled. At this time, this primarily consists of UNIX-based systems that run the X Window System, such as Linux, Solaris, Mac OS X, and Microsoft Windows-based machines. There is a practical minimal screen resolution limit, probably around 800x600 for this tool.

Ui is a product of the Unicon Project and is a standard part of the Unicon programming language distribution, developed under the GPL, hosted on Source Forge, and downloaded from the web site at http://unicon.org. If you are not already familiar with Unicon you may wish to consult other technical reports from that site, such as UTR #7[2], along with this one. Windows Unicon and the Ui environment are based on the volunteer work of many people; final responsibility for this release rests with Clinton Jeffery of University of Idaho. Send requests, and bug reports to `jeffery@cs.uidaho.edu`.

2. Editing, Compiling, and Executing Programs

Double-click the Windows Unicon icon to launch *Ui*, the Unicon IDE (integrated development environment). *Ui* is written in Unicon and allows you to edit, compile, and execute programs from a graphic user interface. Ui is designed to run the same on UNIX and Linux as it does on Windows. Ui's documentation (this file) may be accessed on-line through its Help menu. To start, you must select the name of a file to edit, in following dialog:

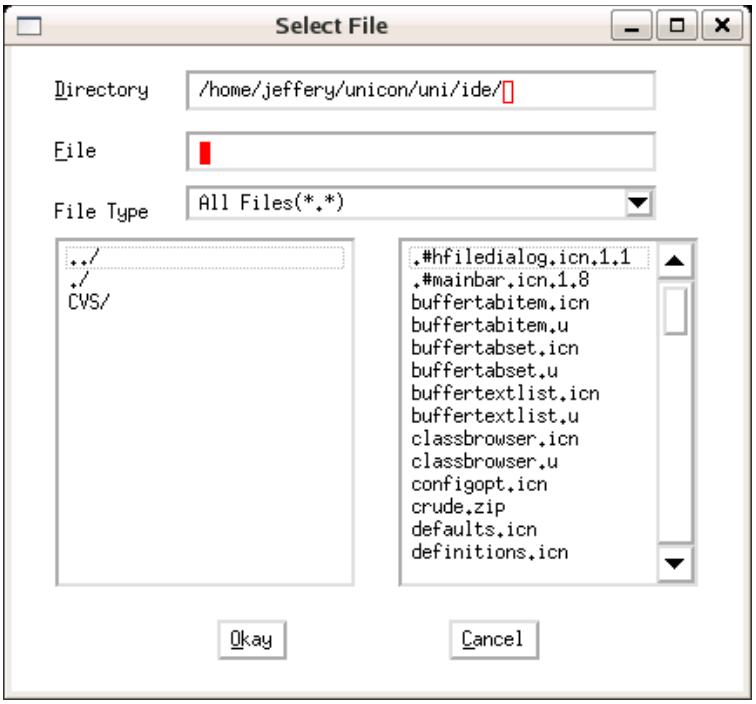


Figure 1: Opening a File within Ui

You can easily select an existing Unicon source file, or name a new one. If you click "Open" without choosing a name, you will be given the default name of "noname.icn". Unicon source files generally must use the extension `.icn` and should be plain text files without line numbers or other extraneous information. Editing your program occurs within the main *Ui* window, which might look like this:

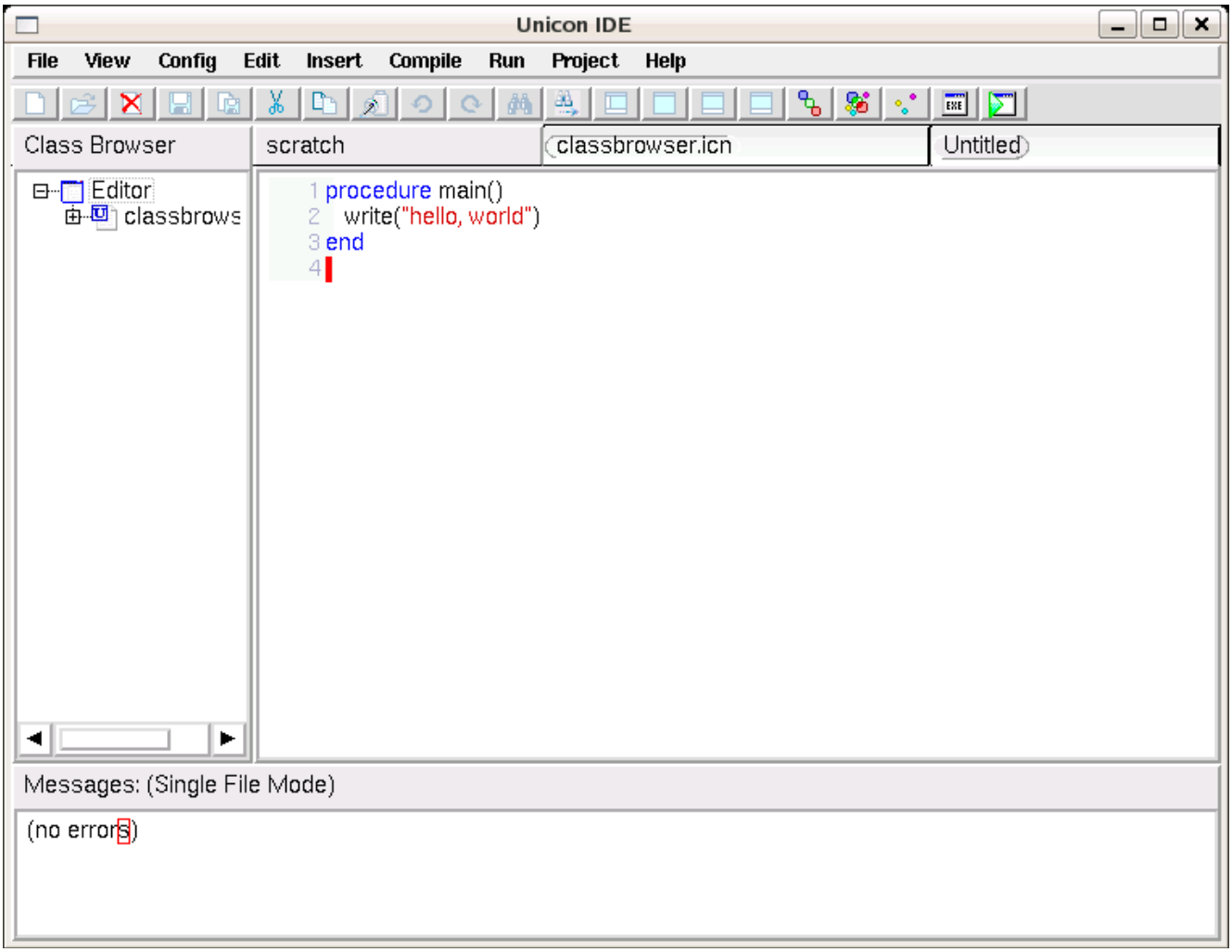


Figure 2: Ui's Editing Interface

The top area shows program source code, while the bottom portion shows messages such as compiler errors. You can change the font and the number of lines used to show messages from the Edit menu.

When you are done editing your program, you can save it, compile it, or just "make" (save, compile and link an executable) and run your program with menu options. The Arguments command in the Run... menu let's you specify any command-line arguments the program should be given when it is executed.

3. Error Handling

Compile errors result in a message in which the editor highlights the line at which the error was detected, like this:

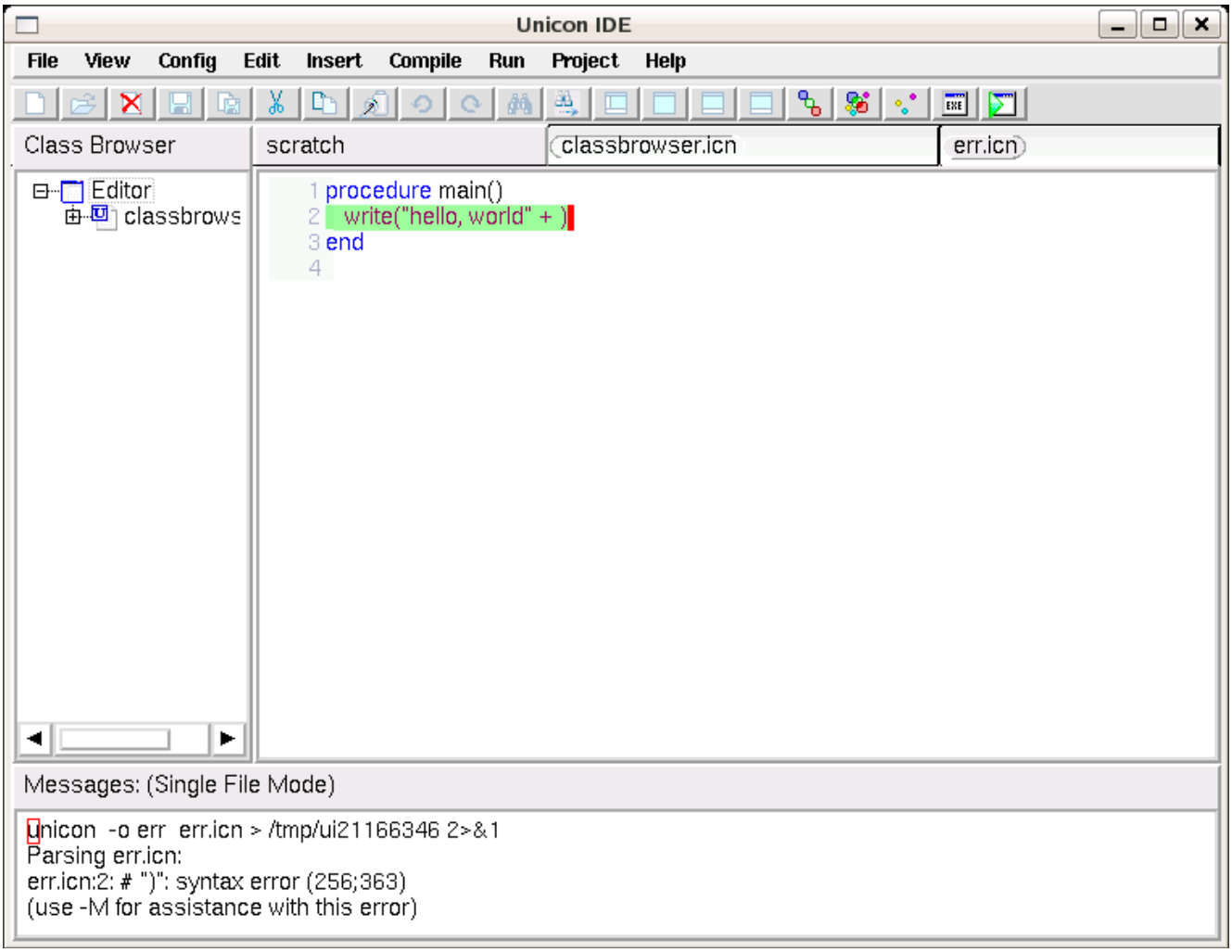


Figure 3: Reporting Compiler Errors in Ui

Run-time errors also result in a message for which the source line is highlighted. The message for a run-time error includes Unicon's standard traceback of procedures from `main()` to the procedure in which the error occurred. When the error messages get long, you can either increase the number of lines for the message window (as was done here) or scroll through the message window's entire text using the scrollbar.

Figure 4: A Run-time Error in Ui. The message window's size is adjustable.

4. Projects

Ui works on programs comprised of any number of source files. By default, Ui assumes a single file program consisting of the current source file. If you are working on a single-file program, and you open a new `.icn` source file, Ui switches the editor and compile and link commands to work on a different program.

Project files are plain text files with extension `.icp`. They list source files, one file per line. When you open a project file, Ui goes into "project mode", and adds the source files in the project to your File menu, allowing you to switch easily between files in the project. If you subsequently open a source file not in the project, Ui asks if you want to add that source file to the project, or switch out of project mode and edit that file as a separate program. In general, project files allow you to "make" large projects efficiently. Underneath the covers, Ui invokes the other Unicon program executables to do the work of compiling and running programs, described below.

When Ui "makes" a program executable, it recompiles those modules listed in the project file whose modified time is newer than their corresponding object files. On the other hand, Ui knows nothing about link declarations embedded in source files, and recompilation will not be triggered by such dependencies. When you use Ui, you should generally use link statements for library files (such as Unicon Program Library modules), and use `.icp` files for your own sources. Files listed in the .icp file must not also be referenced in a link statement; linking the same module twice causes link errors. For projects, the executable `.bat` that is produced by "make" is named after the first program in the project file.

5. Ui Options

Ui supports command-line options to be passed to wicont with the Compiler Options... menu item in the Compile menu. Command-line arguments passed to the Unicon program when it is run are supplied via the Arguments... menu item in the Run menu.

The font in the edit window can be picked from the Font... command in the Edit menu. The number of lines to use for messages can be picked with the Message window... menu item in the Edit menu.

The font and message line options, as well as a default window width and height, may be specified in a file called `ui.ini` that Ui reads when it starts up. The file `ui.ini` from the current directory is used, unless a UNICONINI environment variable is set, in which case it is taken to be the pathname of the initialization file that Ui is to use. An example `ui.ini` file is the following:

```
width=800
height=800
font=times,28
msglines=3
```

6. History and Related Work

> Many Bothan spies died to bring us this information. -- Mon Mothma

In the early 1990's, a prominent member of the Icon community (Bob Goldberg, if I recall correctly) did a very rough prototype IDE for Icon on Windows, which whetted our appetite for this category of tool. An M.S. student at UTSA, Steve Schiavo, did a second prototype IDE in Borland Delphi. These efforts featured basic editing, compiling, and executing but were not feature complete and never released.

Clint Jeffery ported Icon's graphics facilities to Windows while at UTSA, as part of his agreed contribution for the book "Graphics Programming in Icon"[3]. An IDE (along with a setup.exe installer) was considered essential for the Windows Icon distribution. In order to minimize external language- and compiler-dependencies, and in order to test and demonstrate the functionality of Windows Icon, Schiavo's IDE was reimplemented in Icon by Clint Jeffery as a program called Wi (Windows icon). In order to provide an attractive, native-looking GUI, Windows95 native dialogs (the Win* functions) were added to the language specifically for use in Wi. The Wi program was modified slightly to form Wu (Windows unicon) when that language was invented.

Although the Wu program was nice, it was limited to only run on Windows, and made extensive use of Windows-only native GUI functions. Even on Windows it had some nasty limitations, such as a 32kb file size limit imposed by the classic native windows text editor widget. After Robert Parlett contributed his wonderful GUI class library, which supported fonts and colors, it became ever more desirable to build a Unicon IDE that would run on all Unicon platforms. There were multiple failed attempts, such as Bryan Ross's RUDE (Ross' Unicon Development Environment).

Finally, the Ui program was developed by Clint Jeffery, using Parlett's IVIB interface builder and adapting code from Wu. Unfortunately, various elements of Wu were not replicated immediately in Ui, and it has only gradually progressed towards a complete or usable feature set. Its toolbar was added by Nolan Clayton. Syntax coloring was added by Luis Alvidrez. Hani bani Salameh ported it to version 2 of the GUI classes. It is by now the work of several people, and remains very small and lacks features compared with mainstream IDE's. However, it continues to evolve.

7. Conclusions and Future Work

The Ui program is a relatively small multi-platform IDE for Unicon. Its original author (Jeffery) is perpetually embarrassed by its lack of features, and is grateful for the help he has received towards making Ui less of a toy. Ui remains perpetually in need of more and better "polish", and the list of planned features is long.

- integration of IVIB and Ui -- like other languages' IDE's.
- Emacs mode -- ground work for programmable keybindings has been laid.
- integration of UDB

References

1. Clinton Jeffery, Shamim Mohamed, Ray Pereda, and Robert Parlett, Programming with Unicon, http://unicon.org/ub/ub.pdf, 2005.

2. Clinton Jeffery, Version 11 of Unicon for Microsoft Windows, http://unicon.org /utr/utr7.html, 2006.

3. Gregg M. Townsend, Ralph E. Griswold, and Clinton L. Jeffery, Graphics Facilities for the Unicon Programming Language Version 9.3, The Univ. of Arizona Icon Project Document IPD281, 1996.