A/UX: How To Use Prototyper as Development Tool

Article Created: 12 April 1989

Article Change History

08/31/92 - REVIEWED
- For technical accuracy.

TOPIC ---

These notes explain how to use Prototyper 2.0 as a development platform for A/UX.

DISCUSSION ---

Introduction

Prototyper 2.0 is a screen layout and code generator that produces most of the code and resources necessary for the Macintosh user interface part of the program. Prototyper release 2 includes things like MPW C 3.0 code generation, pop-up and hierarchical menus, and linking icons and menu entries to various other windows and dialog boxes. We hope these notes help people get up to speed when using Prototyper as a development tool for A/UX programs that make use of the Toolbox.

As stated above, Prototyper is a very versatile tool. A programmer saves a lot of time and energy when using Prototyper for the base program. The use of prototyping tools and object-oriented methodologies speeds up the boring parts of the programming job.
In general, one can define a simple interface and produce code within 10 minutes. Depending on the level of complexity, the design phase takes much longer.

How To Use Prototyper

Prototyper runs well under A/UX. There are some things that make the program bomb, but, in general, the program is quite stable. When you design anything with Prototyper, you start with the user interface. This phase usually takes much longer than anticipated.

Prototyper saves this frozen user interface in a special file. You are able to

cold-run this interface during the prototyping phase as much as you want. This is especially helpful because it saves time compared with endless testing-recompilation phases and sleepless nights.

You can produce either resource code (in .RSRC and .R format, .R stands for the old RMaker program), or both resource and code. In our case, we will choose MPW 3.0 code because this best reflects the A/UX environment.

The only worry when Prototyper produces source and resource code is that the tab settings are a bit unusual for the A/UX environment. This is easy to change with some clever sed programming. Another thing to think about is the 14-character limit of filenames. Prototyper tries happily to produce filenames that reflect the names of windows and dialogs used inside the code, and many times the names are longer than 14 characters. You can define your own names when Prototyper produces the C source code files.

Remember to strip out the AppleSingle/Double stuff from the top of the file before you continue in the A/UX file system with UNIX tools.

Resources

The .RSRC format can be translated back to the rez/derez format that rez/derez uses within A/UX. The following shell script does the job handily:

```
if [ -f ${$1}.res] then
        echo Output file \(${1}.res\) exists!
        echo No action taken
        exit
fi
echo Doing derez of file $1.RSRC - new resource file will be $1.res derez
-i/usr/lib/mac/rincludes ${1}.RSRC types.r > ${1}.res
```

What you then need to do with the produced .res file is to include a statement type:

```
#include "types.r"
```

at the top of the file for the ultimate rez of the resource file. This is for the make, so it will find the resource type templates in the A/UX system.

Prototyper makes a special OpenResFile call inside the produced C code that opens a special resource file. Comment this because we will use the AppleDouble style resource files.

C Code

The C code produced by Prototyper is very ANSI-C-oriented. Thus, some parts of the code have to be changed so it will compile under A/UX. The most annoying things to change are the so-called function prototypes, usually found in the .h files.

For instance, if Prototyper produces function declarations like:

```
void Foo(WIndowPtr MyWindow, ControlHandle CItem);
```

we have to change this to:

```
void Foo()
```

in the .h files, and sometimes in the .c files. Note that these are function declarations, and not the actual function headers. For me, it is annoying because the function prototypes help a lot when you have written code that wants to call a function with the totally incorrect data type. ANSI C function prototypes complain about it during the compilation; normal K&C compilers like pcc happily accept the variables.

Another issue is the string used in MPW and A/UX. In MPW, you can declare Pascal strings with the handy \p in front of the string, as in:

```
DrawString("\pThis is a Pascal String");
```

This won't work with A/UX, so you have to use the small letter Toolbox calls that know about C strings, like:

```
drawstring("This is my String");
```

The best thing is to define:

```
#define DrawString drawstring
```

in a special header file. This is also true with string functions, such as GetIText and SetItext.

Another thing to change are the filenames for the Toolbox modules. They usually have capital letters in their names, so you need to change names such as QuickDraw.h to quickdraw.h.

Note that all this could be done with #ifdef statements in the code; that is, you could use the same base for both Macintosh OS and A/UX code.

In general, the code produced is stable, but not much is optimized. It also leaves dangling handles that you need to keep track of and remove if you want to have really bug-free code.

Prototyper produces code with comments, so it is very easy to dive into the code and change or modify things. Still, you need to know the fundamentals of Macintosh programming if you want to change or enhance the C code.

Don't forget to use select(2) to poll for interrupts on the Macintosh Toolbox device; otherwise, you spend too many resources for the Macintosh program itself. For more details, read the term sources.

Makefile

The easiest way is to copy the example one found under /usr/lib/mac/examples and tweak this to reflect the source files.

End Result

Well, after about 10 minutes and with a clever idea, you have a working base program and can continue with the rest of the UNIX programming. After a while, you realize that with Prototyper the user interface is 80% of the job, and the function coding is a small portion of the creative work. I would highly recommend this product if you want to write A/UX tools and applications that make use of the Macintosh side of A/UX.

If you are unsure (as I am) about scroll bars in windows, check the latest code examples from Phil and Dave's Excellent CD. There is a really good example that works under A/UX, as well.

Keywords: <None>

==========

19960215 11:05:19.00
Tech Info Library Article Number: 4723