''
' Aplomb Scribe
' Martin's Liberty Basic Pre-compiler
' by Martin Truesdell
' parses "include" files in source file
' and other fine things
''

License:

WARNING! Do not use Liberty Basic IDE as your IDE with aplombscribe if and at all possible. AS adds the prefix "DONOTEDIT" so you know not to edit the processed file which is then launched by Liberty Basic. If you edit the DONOTEDIT file your changes will be lost. I personally use vim but you can use geany or any other IDE. Vim and Geany both provide a way of automatically launching AS but I personally use pc.bat 99% of the time.

If anybody sees code in this program that they wrote and want to be credited then let me know (Hooshnik on Liberty Basic Conforums).

I apologize if my documentation is not clear.

Welcome to Aplomb Scribe (AS). With it you can do a few things in a pre-processor way for your Liberty Basic programs. It is spartan and meant to be used by programmers. It has no GUI except the use of the mainwin. Its main feature is the ability to add include files to your program. This program does not support real C++ style include strength and these "include" files are really just macros. It supports a file depth of 2. I was going to add infinit depth but I saw LB has a problem with recursion in functions. So if I have a macro in main.bas and then another macro in that macro that is as far as I can go.

AS will insert the file exactly at the point specified so if you have only functions and subs in it you will want to put it at the end of the file like this:

Right now there is no command in LB that starts with an angled bracket so trying to compile without AS gives an error. Also AS is not compatible with the Liberty Basic Workshop's include system (you can still use it for everything else).

Here is the syntax for the program if you run it without parameters using the console:

```
syntax for this program:
aplombscribe -a -b -s -t [-d | -r] 'bodyfile.bas' 'c:\path_to_liberty_basic\liberty.exe'

example to run program on 64 bit system:
aplombscribe -a -r 'myprogram.bas' 'c:\Program Files (x86)\Liberty BASIC v4.04\liberty.exe'

Liberty Basic specific compiler switches:
-a : close all windows on exit
-d : debug
-r : run

this program's switches:
-b : build mode, don't launch liberty basic just make DONOTEDIT file
-s : enable user defined strict warnings (stop on warning)
-t : turn on test mode (don't write changes to disk, don't launch anything)
-e : turn on expert mode (do not ask me for support for this switch)

strict mode can be slow, using it with test mode recommended
```

NOW! Normally you do not run the exe directly. You want to run it from a batch file so I have provided pc.bat. The typical line to launch AS would be:

```
pc myprogrambody.bas -r
```

or run it with debug:

```
pc myprogram.bas -d
```

If it sees the word "body" in the filename it will remove it in the DONOTEDIT file it creates. Normally I append the word 'body' to my main file. Please edit pc.bat for your needs. You will need to know if you have a 64 bit "Program Files (x86)" or 32 bit "Program Files" windows and whether or not you use LB Pro "Liberty BASIC Pro vX.XX" or Standard "Liberty BASIC vX.XX". Change the line if needed:

```
aplombscribe -a %2 %3 %4 %5 %6 %7 %8 %9 '%1' 'c:\Program Files (x86)\Liberty BASIC Pro v4.04\lbpro.exe'
```

You will notice that the '-a' switch is on by default. It ensures that LB closes with less clicking.

If you have lots of macros and find the standard file existance checks are taking too long try using the expert switch. AS will crash with an ambiguous error message if you have a missing file using the expert switch so watch out.

Well that's all you need to know to get started so don't use strict mode for now.

EXPERT DOCS:

---

warningcheck.cfg is needed for strict mode. I have provided a sample of this file. I will try to explain some lines with tilde '~' as the delimeter. Also see the file itself for more docs (in fact read it first then come back here). Matches are from the beginning of the line and takes indentation into account.

```
~WRITE EVERYTHING IN LOWER CASE EXCEPT FOR THE ERROR MESSAGE
```

standard comment line

```
~gosub~use of gosub depricated~~
```

if gosub is seen give error 'use of gosub depricated'

```
~~call convertir~(test) convertir detected with 'for' before~for~
```

Will do nothing because it is a commented line

```
~call convertir~(test) convertir detected with 'for' before~for~
```

it means look for 2 lines like this:

for (blah blah blah could be anything here)
call convertir

then trigger the error message '(test) convertir detected with 'for' before'. This is a powerful search function. If it had only found the second line it would not have triggered the error message!

~#~possible print without conversion~!call convertir~~mainWindow~configuration

Here is a great one! If a line starts with '#' instead of the usual 'print #' then look at the previous line. If the previous line DOES NOT CONTAIN 'call convertir' then print the error 'possible print without conversion'. You can put an exclamaion point in front to mean not for the before and after fields. The last 2 arguments are the exceptions for the current line and there can be as many as the array can hold (current limit looks like 1000). So if it sees '#mainWindow' it will abort showing the error message.

~#~possible print without conversion~~!call convertir~mainWindow~configuration

Exactly the same as the last one except it will look for NOT 'call convertir' AFTER the current line instead of before. Examine the tildes closely!

The Main Program

The main program code is kind of big to copy paste so download it here:

aplombscribe.bas

- Details
- Download
- 28 KB

Recommended File(s)

pc.bat

```
@echo off

rem aplombscribe pre-Compiler batch file

if "%1"=="" goto case_help
if "%2"=="" goto case_secondargumenterror

rem check for basic file extension
if not "%~x1" == ".bas" goto case_firstargumenterror

rem aplombscribe -a %2 %3 %4 %5 %6 %7 %8 %9 '%1' 'c:\Program Files (x8
6)\Liberty BASIC v4.04\liberty.exe'
aplombscribe -a %2 %3 %4 %5 %6 %7 %8 %9 '%1' 'c:\Program Files (x86)\L
iberty BASIC Pro v4.04\lbpro.exe'
goto end

:case_firstargumenterror
echo.
echo ERROR:  first argument must be a basic file name
:case_secondargumenterror
echo ERROR:  must have a second argument (switch)

:case_help
echo.
echo launches Aplomb Scribe (Liberty Basic Precompiler)
echo.
echo usage:
rem we need the caret (^) for escaping
echo pc filename.bas ^[-r ^| -d^] -s -t
echo.
echo -d : debug
echo -r : run
echo -s : enable user defined strict warnings (stop on warning)
echo -t : turn on test mode (don't write changes to disk, don't launch
 anything)
echo.
echo strict mode can be slow, using it with test mode recommended
echo.

:end
```

Optional File(s)

warningcheck.cfg (lines that had 3 tildes in a row removed due to a bug in the wiki thinking it's a wiki code (in code block or out))

```
~
<c>~winansi windows-1252 marker -> é
<c>~WRITE EVERYTHING IN LOWER CASE EXCEPT FOR THE ERROR MESSAGE
<c>~SEARCHES ARE CASE INSENSITIVE!!!
<c>~IF YOU GET AN ERROR ABOUT BLANK EXCEPTIONS BUT
<c>~HAVE NONE TRY ELIMINATING TRAILING DELIMITERS
<c>~
<c>~comment blocks begin with <c> and the delimeter
<c>~file format :
<c>~the very first line is the delimeter of 1 character
<c>~then you can add comments like this one if you want
<c>~the period at the start of a line is the beginning of a warning ch
eck line
<c>~a delimieter must come in between every argument
<c>~there are 4 mandatory arguments
<c>~1: the search pattern for the current line
<c>~2: the error message
<c>~3: what fragment has to be present
<c>~   on the previous line for a warning, if any
<c>~4: what fragment has to be present on the next line for a warning,
 if any
<c>~   if none for previous, next line put 2 delimiters in a row
<c>~5 and beyond: exceptions immediately after the search pattern
<c>~   for the current line. you can have as many as you want
<c>~write in lower case, everything is case insensitive
<c>~put an ! in front of 3. or 4. to warn if it ISN'T there
<c>~
<w>~gosub~use of gosub depricated~~
<w>~return~use of return depricated~~
<w>~fileio$ = ~seqFile read error~call seqfile fileio$, h1$, "read", 0
~
<c>~<w>~call convertir~(test) convertir detected with 'for' before~for
~
<c>~<w>~call convertir~(test) convertir detected with 'if' before~if~
<w>~call convertir~convertir detected~~
<c>~
<c>~
<c>~<w>~#~print statement used without 'print' in front~~
<c>~<w>~do~(test) found a do loop~~
<c>~
<c>~<w>~#~possible print without conversion~!call convertir~~mainWindo
w~configuration
<w>~print  ~print with 2 spaces after~~
```

```
<c>~
```