1. Comments on Yac_2o_2's *READMEv1*.1 file — Date: February 16, 2015.

Project: Yac_2o_2 — Multi-threaded lr(1) Compiler/compiler system
Distributed under license: Mozilla Public License, v. 2.0.
Distribution Date: February 16, 2015
Distribution version: 1.1
Comments: Currently for the Unix flavoured Platforms
Author: Dave Bone
Support email: cc.yacco2@yahoo.ca
Contributors list:
Dave Bone

The distribution contains the programs: compiler/compiler O_2 and its linker O_2*linker*, along with its runtime library *yacco2*. There are build scripts for platforms: Apple, GNU, Solaris. Various documents, grammars, reference manual, quality assurance suites, and generated c++ source code are included. These items are elaborated upon further within this document.

2. License.

Yac_2o_2's distributed source code is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, you can obtain one at http://mozilla.org/MPL/2.0/.

As best as I could, the "Literate programming" w type files contain a reference to this license along with their generated documents. The c++ source files generated from the w files **do not and will not have any comments referencing this license but are also subject to its terms**. The *ctangle* program generating the c++ code does not emit programmer c type comments. It emits "#line" macro references to the w file and only c comments to its gened document's section number. Any discovered license omissions required to be referenced within Yac_2o_2's Project files are still covered by this license and will be corrected in future distributions.

3. Version *v1*.1 — minor fixes.

⇒ Regenerating Yacco2 c++ code emitted improper license comments that produced compile errors
⇒ Renamed diagrams+etc folder to diagrams

4. README literate programming.

This "READMEv1.1.w" file is run thru the *cweave* progam to generate a tex file: cweave "READMEv1.1.w". The **v1.1** suffix in its name is the distribution's version. Then the program *pdftex* is run to create its pdf document: pdftex "READMEv1.1.tex". Literate programming at the install documentation level that u are now reading.

5. Comments on 1st attempt at "Open Source".

Project Yac_2o_2 is my first attempt at "Open Source" software. There are propably faux pas made by my lack of experience but I'm willing to learn from the community. My ears, eyes, and grey cells are open to your suggested refinements. Harnessed non-determinism thru threading opens up parallel activities. The thought experiment stepping out with arbitration deciding multiple outcomes. Interesting when a thread becomes a computational universe having time displaying its trails of activities...

6. Some installation concerns.

Here is a list of questions/answers to help u install and get things going.

7. How is Yac_2o_2 installed?.

Grab a copy of the system and place it on your computer. It comes as a zipped file named **yacco2.zip**. Depending on your run platform, either double click the zipped file or run from the command-line-interface using an unzip program like gunzip. Once unzipping its contents, the **yacco2** folder contains subdirectories of text files aka grammars and "literate programs", the c++ source code, binary static libraries of Yac_2o_2, and the 2 executeable programs *o2* and *o2linker* which are the compiler/compiler and its linker.

8. Where is Yac_2o_2 installed?.

The **yacco2** folder should be copied to the **/usr/local/** folder.

This distribution contains the executeables and libraries **prebuilt for an Apple laptop running Maverick**. Other platforms will need to be built using the instructions below. This comment also holds for older Apple platforms like the "Cat" series.

A "drag and drop" attitude it is to installing Yac_2o_2 placed in **/usr/local/**. It is self contained. To thrown it out just drag the "/usr/local/yacco2" folder into the trash bin and delete it or use the Unix "rm" utility to delete it.

For example if u unzipped Yac_2o_2 into your desktop folder, then the following *Bash* CLI move command on the Apple platform is:

1) cd ~/DeskTop # example Apple desktop holding yacco2 folder
2) mv -f yacco2 /use/local/ # move the yacco2 folder

Substitute your own folder name containing yacco2 for the above "cd" command. U can also use a "drag-and-drop" to do it. For the Apple platform, u can unhide the hidden folders shown by "Finder" or use Finder's "cmd-shift-G" key combination and type in the "/usr/local/" to see its contents in the Finder's screen. Now "drag-and-drop" the **yacco2** folder into **/usr/local/**. Other platforms have their version of "drag-and-drop" to move Yac_2o_2.

9. What C++ compiler does it require to compile/install Yac_2o_2 system?.

The Apple environment needs Xcode or a development system like GNU. For the other platforms, a development system with a C++ compiler and a run time library supporting multi-threading.

This distribution already has the *o2* and *o2linker* programs installed along with the *yacco2* library for the latest Maverick Apple platform. So u can stop the balance of the installation for the Apple if u are running Maverick. For older Apple Operating systems like the "Cat" series, they require u to do some adjustments to the Apple scripts below. There are *bash* comments in these scripts on the variables to set.

10. How to install Yac_2o_2.

Once you have placed Yac_2o_2 system in **"/usr/local/"** on either your laptop or server, you will find the following Bash scripts where each script's suffix name indicates the platform to run the script under.

1) bld_bash_APPLE
2) bld_bash_GNU
3) bld_bash_SOLARIS

You run one of the above scripts with the *bash* command-line-interface(CLI). Each script is documented with references to other scripts of **makefile_xxx** where xxx is the platform to build the subcomponents of Yac_2o_2. Have a read, they don't bite.

11. Possibly out-of-date — GNU and Solaris scripts.
Possibly points 2 + 3 could be out-of-date as I'm now developing exclusively on the Apple MacPro / Xcode environment. These scripts should clue u into what is required to get them running on a more up-todate platform. Once upon a time they worked but time has changed the Apple script and not them! So use the Apple script as a guide to get the other scripts working. I'm sure the C++ compiler and its supporting libraries will be out-of-date.

12. Run the installation script.
Here are the bash commands to install the complete system on the Apple MacPro system:

1) cd /usr/local/yacco2
2) . *bld_bash_APPLE*

The script has commented out other options that u can use for other things like gening the *Cweb* files to building debug executeables. Ditto on the other Operating system scripts.

Note:
In the ./bin folder, there are the latest executeables for the Apple MacPro "OS X 10.9.2/ Xcode 5.1.1: O_2 and O_2linker along with the libraries: *yacco2* in "/usr/local/yacco2/library/lib/Release" for u to link your own compiler to, and *o2grammars* in "/usr/local/yacco2/compiler/grammars/lib/Release" for the Yac$_2o_2$ programs to build against. Ready baked for u to try if u are on an Applelite.

13. Other OPs: GNU, SOLARIS for bld_bash_XXX and makefile_XXX scripts.
To provide consistency, scripts per run platform were written rather than depend on a specific development framework like Apple's Xcode. If u use a development framework, these scripts will clue u to the ingredients needed to map Yac$_2o_2$'s components to your framework.

You will probably need **to edit** the GNU or Solaris scripts regarding the C++ libraries versions needed. The referenced **makefile_APPLE** scripts per subfolder like **/usr/local/yacco2/library** in the **bld_bash_APPLE** script uses the latest 64 bit LLVM compiler toolset under Apple's latest operating Maverick version "10.9.2" and **Xcode** Version 5.1 (5B130a). The later Xcode (6.+) versions were tried but at the time of preparing this distribution were unstable and crashing. Possiblely now Xcode's latest version is stable but for now i'm staying with this earlier Xcode version. Each **makefile_APPLE script per subfolder are up-todate** and can be run individually within their subfolder. There are equivalent scripts to the mentioned APPLE scripts within the various folders with suffix GNU or SOLARIS indicating the operating system supported.

Their use are the same under each operating platform. For example to regenerate only the Yac$_2o_2$'s library, run the following *bash* commands:

1) cd /usr/local/yacco2/library
2) make Genw -f makefile_APPLE # gen c++ code from w files
3) make Rlse -f makefile_APPLE # compile c++ code and create library

All other port scripts **should be eye-balled** against each *makefile_APPLE* script for any adjustments needed when compiling/linking re: C++ compiler and **ld** libraries used: *pthread*, *C++*. For the above example with the *Rlse* parameter inputted, you will see it as a label within the bash script with its appropriate compile and create the library commands. These scripts are reasonably documented for your eyes and editing.

Note: the **makefile_APPLE** script can be modified to support an older **Xcode** version. Yac$_2o_2$ has been developed on Xcode versions from 3 and greater on Apple's "Cat" platforms. You'll have to adjust your script for the appropriate C++ compiler and C++ runtime support library.

14. Grammar Testsuite.
Added to the distribution in folder **/usr/local/yacco2/grammar-testsuite** are some grammars from published papers testing out lr(0), lr(1), and lalr ness by running its "test-out-grammars.sh *bash* script. All platform scripts run O_2 against the testsuite to confirm the distribution validity. It also runs self against self with one of its own grammars: "eol.lex".

15. A translator for your learnings from the Grammar Testsuite.

To confirm that the distribution works, one of the published grammars: "pager_1.lex" has a translator *testout* that is compiled, linked against the built Yac_2o_2 library, and run against some input with some trace parameters turned on: see **1lrtracings.log**. Have a peek in its folder to see how easy it is to build a compiler/translator: script makefile_testout_APPLE.. Of course my leanings are towards "literate programming" use and hence the slant on its residues of generated documents.

16. Quality Assurance.

In **/usr/local/yacco2/qa** folder, there are a series of batch files that tests out the "quality assurance" of Yac_2o_2. Within the various tests are more grammars testing lr(1) compliance. For the curious or doubters have a look or try them out yourself. They might leave bite marks but should be a bit more convincing than my verbage.

```
cd /usr/local/yacco2/qa
ls *sh # list of bash scripts to be run
```

17. Challenge to the Open Source / Computer Science Community.

Depending on the C++ code, *cweave* emits unstructured code. In the **o2book** document there is a footnote as to why this is occuring. I raise the challenge to u to correct this in improvements to the “Open Source” evolution.

Internet donations is a newer money raising model helping not-for-profit foundations with support staff but for the majority of Open Source projects, their life expectancy becomes short lived when others don’t pick up the torch towards bug corrections etc. Letting the author “just do it” leads to a no-growth attitude for the next generation. Enough Dave of your squawking! So is the challenge taken up from possibly the computer science students as a class assignment? Let’s hope the future tales whisper challenge accepted/taken/completed!

18. Unix *man*'s quick overview to Yac_2o_2's programs and API.
In **/usr/local/yacco2/bin/man/man1** are the Unix style references for:

o2 compiler/compiler
o2linker Yac_2o_2's linker
yacco2 API/library

Either set up a link to this folder in your startup file like **.bash_profile** to exercise them, explicitly reference their path when running *man*, or adjust your **/private/etc/man.conf** file.

Command-line-interface examples for Yac_2o_2's references.
The examples show variations depending on how u set up the path reference:

man /usr/local/yacco2/bin/man/man1/yacco2.1 #explicit reference to API library
man -M /usr/local/yacco2/bin/man/man1: yacco2 # use man's path list
man o2linker # o2linker and man.conf file has Yac_2o_2's reference's link
man o2 # o2 where MANPATH has link

19. Documents to read — U have edocs Sir!.
All docs are in the "/usr/local/yacco2/docs" folder in "pdf" format.

1) Grammars in their published format — examples of literate programming using *Cweb*
2) O_2's grammars in /usr/local/yacco2/compiler/grammars/*.lex in raw form.
3) O_2 and O_2linker documented source programs.
4) wlibrary — Yac_2o_2's documented library.
5) Some unpublished essays — p1, p2 discussing the why of O_2's language paradigm.
    Relatively relevent though early in O_2's development cycle.
6) Unpublished Tugboat-o2 essay on "literate programming" as applied to O_2. A must read.
7) Draft copy of reference "o2book.pdf" to rev u up.
8) "testout.pdf" and grammars "test*.pdf, pager_1.pdf" comments on writing your own compiler.
9) "testdriver.pdf" and "test_components.pdf" — lexical quality assurance testing.
10) possiblely published book — "Yac_2o_2 and some practical uses".

The unpublished essayes listed above should be read first as they describe in a light way the Why of O_2. For those that don't have time, at least speed read the "Tugboat-o2" essay. *testout* program documentation should be read along with the "o2book". There are comments to developing your own compiler regarding the Terminal / Error Vocabulary, and Error tracing. At least there are some clues/comments/references to Yac_2o_2, and its use.

"Yac_2o_2 and some practical uses..." will be a "how to" set of observations/steps when i developed a retargetting Pascal translator. It deals with observations made in grammars evolution, Terminal/Error Vocabulary development, symbol table management controlled from grammars' logic points, to tree building and their walkings within semantic routines. The project's post evaluation gives observations, mistakes made, and potential warnings for your future language development. Hopefully to u a necessary companion with Yac_2o_2's reference "o2book".

20. Comments: Typesetting, “Literate programming”, and Drawing.
The principle software driving the typesetting of documents is TEX which should be on your system. Metapost is the graphics drawing system used to sketch grammars. By combining the 2 components: C++ code and typesetting directives into one file, u now have a “Literate program”. This is the principle used by O_2’s grammars. O_2 emits its own C++ code files. For its grammar documents, O_2 maps the grammar into a *Cweb*’s template form called a “literate grammar” and also emits Metapost grammar type files. These become the fodder for *Cweb*’s *cweave* program and Metapost’s *mpost* digestion. The TEX *pdftex* program turns their outputed files into a pdf document. All said, to use O_2 does not require TEX, *Cweb* or Metapost systems to be installed but you are losing out on one of the gems in O_2’s tiarra: document generation.

Caveat: as the *Cweb* system is required to emit C++ code from “Literate programs”, all of Yac_2o_2’s programs and API library will be very difficult to modify/correct/enhance without this system installed.

Please go visit the **‘www.tug.org’** website. It provides “downloads” for various operating platforms of TEX, *Cweb*, and Metapost systems. Other interesting software gems are there for download. Accompanying these downloads are their “How to use it” manuals/documents. **Please consider joining Tug** as there are a lot of “Open Source projects” that need your support to keep improving.
A recap on programs use:

1) *cweave* creates a “.tex” file from a “.w” program for document creation.
2) *pdftex* generates a pdf document from cweave’s outputted ‘.tex’ file.
3) *ctangle* generates the C++ program from the same ‘.w’ program.
4) *mpost* is used to draw the grammar’s productions generated from O_2.

All Yac_2o_2’s grammars and programs can be regenerated using these above programs. To generate the grammars’s documents, they are run thru O_2 with the ‘-p’ parameter that generates a ‘.w’ file and accompanying mpost grammar diagrams for *mpost* and *cweave* digestion outputting the “.tex” file for *pdftex* to generate a “pdf” document. Running O_2 without or with possible parameters “-t”, “-err” generates the “.cpp” files to be compiled by your C++ compiler.

21. Bash/batch files to help u do the dirty work.
Have a look at ‘**o2grammars.sh** bash script in ‘/usr/local/yacco2/compiler/grammars/’ folder showing “how to” generate either a C++ or typeset document for each grammar. ‘**gen1grammar.sh** bash script has the same capability but is applied to 1 interactively selected grammar.
Here’s how to gen your grammars.

```
cd /xxx # your folder containing your grammar *.lex files
. /usr/local/yacco2/compiler/grammars/o2grammars.sh Rlse # gen c++ code
cd /xxx # your folder containing your grammar *.lex files
. /usr/local/yacco2/compiler/grammars/o2grammars.sh Genw # gen grammar docs
mv *.pdf /xxx/docs # optional: move docs from your grammars folder into your docs folder
```

22. Potential problems installing O_2.
This is a list of possible problems u might run across. As the system gets more exposure / use, this list will probably grow. Hopefully not too fast!

23. Why is Yac_2o_2 installed at the /usr/local level?.
My 1st virtual attempt to get Yac_2o_2 out to the "Open Source Community" and my laziness had it installed at the root level but this has been changed. "/usr/local" was chosen as this seems to be the crowd's concensus to installing foreign software. I contemplated using Apple's "Framwork" approach but this seems to be local to Apple though appropriate. The original include statements for Yac_2o_2's library and API have been adjusted.

24. File Permissions.
After u installed the /usr/local/yacco2 folder make sure u can read / modify its contents. chmod utility might be needed or by running the script suffixed to the sudo utility.

chmod +a "staff allow read" /usr/local/yacco2

or

sudo chmod +a "staff allow read" /usr/local/yacco2

or

sudo chmod 755 /usr/local/yacco2

25. Comments on compiling, linking, and *Cweb* line macro referencing file's "w" code lines.
Files makefile_* like **makefile_APPLE** or **makefile_GNU** are found in /usr/local/yacco2's subfolders of "compiler/grammars/, compiler/o2/, library/, or o2linker/" that compile the individual C++ programs and possibly build a library, or create a runable program. These *bash* scripts are Unix flavoured. Cast your eyes on some of these files as the appropriate *bash* statements with comments show "how to" roll your own within these various Operating system contexts. U should note the **C++'s compiler options used**, and ditto on the **linker's parameters**. They will clue u into what i needed to bootstrap Yac_2o_2 onto different Operating system platforms.

In some of the subfolders you will find a *bash* script "**makefile_Edit_CWEB**" to comment out links to the CWEB code lines in the C++ generated programs. This is helpfull when u want to debug, set break points to the C++ lines of code rather than the ".w" line of code using your interactive debugger. Possibly the newer version of CWEB's **ctangle** program has an option whether to emit the "c" line macro or not. My version does not. Hence the reason for the script to "post process" *ctangle*'s emitted C++ code.

26. Where's libstdc++ or libc++ library?.

Note: The below points also apply to the other computer scripts: SOLARIS and GNU.

These comments are more appropriate for older Xcode environments and other Unix platforms. One problem u might experience is where does the libstdc++ library reside for building both programs O_2 and O_2linker if u are not using a development framework. Or possibly u might want to use a different C++ compiler/set of libraries then the one your system defaults to. For an Apple computer, *bash* scripts "**/usr/local/yacco2/compiler/o2/makefile_APPLE**" and "**/usr/local/yacco2/o2linker/makefile_APPLE**" might need editing along with the c++ compiler to be used due to your older Xcode environment: eg. g++ or gcc compiler and libstdc++ library. Find out where the libstdc++ resides on your system. Running these below *bash* commands to look for libstdc++ should find it. For other C++ library, run the same commands below with the appropriate library name to be searched:

```
find /usr/ | grep libstdc++
```

Or more general:

```
find / | grep libstdc++
```

Once found using your text editor, adjust the **STDCC_dir** variable with the found library location, the library **CC_lib** variable, and compiler **CC** variable in the above 2 script files. The scripts are commented so u should not have any touble editing them. Now rerun the *bash* script to rebuild Yac_2o_2:

```
. bld_bash_APPLE
```

27. An inconvenience on getting the typesetting to work.

An apology due to this minor inconvenience: the "**cwebmac.tex**" file might need to be modified so that the grammar's file listings can be included into their generated documents.

Warning: Try generating a grammar document to see if your system needs to be adjusted.
Warning: The instructions are an example of what I had to do to get it working on my Apple laptop.
Warning: I am not responsible if they do not work covered by the license terms.

The below instructions are what I had to do to get it working on my Apple laptop.
Please make appropriate backups before trying these instructions on your computer if required.

To do this, u must find where the **.../eplain/eplain.tex** file is installed on your computer from the Tex installation. In my computer, it is in the folder "**/usr/local/texlive/2014/texmf-dist/tex/eplain/eplain.tex**" where the year of the distribution in this example is 2014 which u would substitute with your own distribution year. Possibly non Apple platforms could be distributed differently and would require you to use **its folder name instead of this example**. Then u must edit "**.../cweb/cwebmac.tex**" file from the *Cweb* installation with the following line added at the beginning of this file:

```
\input /usr/local/texlive/2014/texmf-dist/tex/eplain/eplain
```

On my computer this file is found here:

```
/usr/local/texlive/2014/texmf-dist/tex/plain/cweb/cwebmac.tex
```

U can use your prefered text editor but make sure u have write permission to this read-only file. I used vim text editor with sudo to obtain write privileges.

```
sudo vim /usr/local/texlive/2014/texmf-dist/tex/plain/cweb/cwebmac.tex
```

Here are its edited contents with my TEX's % comments:

```
%%% Date: 23 Oct 2014 EDT modified cwebmac.tex file to fetch eplain tex macros
%%% needed is the \listing tex macro to generate grammar documents
\input /usr/local/texlive/2014/texmf-dist/tex/eplain/eplain
```

28. Happy Ya$c_2o_2$ing.
Bonne chance with your language development and use of Yac_2o_2.

29. Index.

READMEV1.1