freeWrap

7.03

Documentation

07/04/2026

Build stand-alone TCL/TK executable files. No compiler required!

OR

Use it as a single-file WISH shell

FreeWrap 7.03 Documentation

Table of Contents

FreeWrap 7.03 Documentation

freeWrap License

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Other license information

FreeWrap borrows existing source code from several sources. To meet license terms and Copyright notice requirements associated with some of this code the following information is provided from the original source code distributions.

Dyncall license

If not stated otherwise inside a file, all files here are distributed in terms of:

TCL license

associated with the software unless explicitly disclaimed in individual files.

The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file where they apply.

IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE IS PROVIDED ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

GOVERNMENT USE: If you are acquiring this software on behalf of the U.S. government, the Government shall have only "Restricted Rights" in the software and related documentation as defined in the Federal Acquisition Regulations (FARs) in Clause 52.227.19 (c) (2). If you are acquiring the software on behalf of the Department of Defense, the software shall be classified as "Commercial Computer Software" and the Government shall have only "Restricted Rights" as defined in Clause 252.227-7014 (b) (3) of DFARs. Notwithstanding the foregoing, the authors grant the U.S. Government and others acting in its behalf permission to use and distribute the software in accordance with the terms specified in this license.

TCLLIB license

This software is copyrighted by Ajuba Solutions and other parties. The following terms apply to all files associated with the software unless explicitly disclaimed in individual files.

The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here,

provided that the new terms are clearly indicated on the first page of each file where they apply.

IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY DERIVATIVES THEREOF,

EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE IS PROVIDED ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

GOVERNMENT USE: If you are acquiring this software on behalf of the U.S. government, the Government shall have only "Restricted Rights" in the software and related documentation as defined in the Federal Acquisition Regulations (FARs) in Clause 52.227.19 (c) (2). If you are acquiring the software on behalf of the Department of Defense, the software shall be classified as "Commercial Computer Software" and the Government shall have only "Restricted Rights" as defined in Clause 252.227-7013 (c) (1) of DFARs. Notwithstanding the foregoing, the authors grant the U.S. Government and others acting in its behalf permission to use and distribute the software in accordance with the terms specified in this license.

Tclmtls license

This software is copyrighted by Konstantin Kushnir <chpock@gmail.com>

The following terms apply to all files associated with the software unless explicitly disclaimed in individual files.

The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file where they apply.

IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE IS PROVIDED ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

GOVERNMENT USE: If you are acquiring this software on behalf of the U.S. government, the Government shall have only "Restricted Rights" in the software and related documentation as defined in the Federal Acquisition Regulations (FARs) in Clause 52.227.19 (c) (2). If you are acquiring the

software on behalf of the Department of Defense, the software shall be classified as "Commercial Computer Software" and the Government shall have only "Restricted Rights" as defined in Clause 252.227-7013 (c) (1) of DFARs. Notwithstanding the foregoing, the authors grant the U.S. Government and others acting in its behalf permission to use and distribute the software in accordance with the terms specified in this license.

TK license

This software is copyrighted by the Regents of the University of California, Sun Microsystems, Inc., Scriptics Corporation, ActiveState Corporation, Apple Inc. and other parties. The following terms apply to all files associated with the software unless explicitly disclaimed in individual files.

The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file where they apply.

IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE IS PROVIDED ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

GOVERNMENT USE: If you are acquiring this software on behalf of the U.S. government, the Government shall have only "Restricted Rights" in the software and related documentation as defined in the Federal Acquisition Regulations (FARs) in Clause 52.227.19 (c) (2). If you are acquiring the software on behalf of the Department of Defense, the software shall be classified as "Commercial Computer Software" and the Government shall have only "Restricted Rights" as defined in Clause 252.227-7013 (b) (3) of DFARs. Notwithstanding the foregoing, the authors grant the U.S. Government and others acting in its behalf permission to use and distribute the software in accordance with the terms specified in this license.

TKLIB license

This software is copyrighted by Ajuba Solutions and other parties. The following terms apply to all files associated with the software unless explicitly disclaimed in individual files.

The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file where they apply.

IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE IS PROVIDED ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

GOVERNMENT USE: If you are acquiring this software on behalf of the U.S. government, the Government shall have only "Restricted Rights" in the software and related documentation as defined in the Federal Acquisition Regulations (FARs) in Clause 52.227.19 (c) (2). If you are acquiring the software on behalf of the Department of Defense, the software shall be classified as "Commercial Computer Software" and the Government shall have only "Restricted Rights" as defined in Clause 252.227-7013 (c) (1) of DFARs. Notwithstanding the foregoing, the authors grant the U.S. Government and others acting in its behalf permission to use and distribute the software in accordance with the terms specified in this license.

Tktable license

* COPYRIGHT AND LICENSE TERMS *

(This file blatantly stolen from Tcl/Tk license and adapted - thus assume it falls under similar license terms).

This software is copyrighted by Jeffrey Hobbs <jeff at hobbs org>. The following terms apply to all files associated with the software unless explicitly disclaimed in individual files.

The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file where they apply.

IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE IS PROVIDED ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

GOVERNMENT USE: If you are acquiring this software on behalf of the U.S. government, the Government shall have only "Restricted Rights" in the software and related documentation as defined in the Federal Acquisition Regulations (FARs) in Clause 52.227.19 (c) (2). If you are acquiring the software on behalf of the Department of Defense, the software shall be classified as "Commercial Computer Software" and the Government shall have only "Restricted Rights" as defined in Clause 252.227-7014 (b) (3) of DFARs. Notwithstanding the foregoing, the authors grant the U.S. Government and others acting in its behalf permission to use and distribute the software in accordance with the terms specified in this license.

SPECIAL NOTES:

This software also falls under the bourbon_ware clause v2:

This software is free, but should you find this software useful in your daily work and would like to compensate the author, donations in the form of aged bourbon and scotch are welcome by the author. The user may feel exempt from this clause if they are below drinking age or think the author has already partaken of too many drinks.

TWAPI license

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- The name of the copyright holder and any other contributors may not be used to endorse or promote products derived from this software without specific prior written permission.

Zlib license

Jean-loup Gailly Mark Adler

jloup@gzip.org madler@alumni.caltech.edu

FreeWrap 7.03 Documentation

Overview

The freeWrap program turns TCL/TK scripts into single-file binary executable programs.

The resulting program can be distributed to machines that do not have TCL/TK installed. The executable will also work on machines that have TCL/TK installed but will use its own TCL/TK "image". FreeWrap itself does not need TCL/TK installed to run.

FreeWrap uses the ZIP virtual file system capabilities built into the standard TCL 9 distribution in order to package all necessary files into a single executable file.

Easy, one-step wrapping.
FreeWrap consists of a single executable file. There is no setup required. Wrapping is accomplished with a single command.

Your source and data files can be protected from prying eyes.
FreeWrap can, optionally encrypt all files you wrap into your executable application to provide a secure distribution.

freewrapTCLSH can be used to wrap TCL-only scripts.
FreewrapTCLSH creates a single executable file from one or more TCL scripts. The wrapping syntax is identical to the freeWrap program. This produces a console-only type of program.

freeWrap can be used as a single file stand-alone WISH
Renaming the freeWrap program to some other file name causes freeWrap to behave as a stand-alone, single-file WISH that can be used to run any TCL/TK script. A freeWrap package containing all the files in your application can also be produced by freeWrap and then run using the stand-alone WISH.

freewrapTCLSH can be used as a single file stand-alone TCLSH shell
Renaming the freewrapTCLSH program to some other file name causes freewrapTCLSH to behave as a stand-alone, single-file TCLSH shell that can be used to run any TCL script or a freeWrap package (containing all the files in your application).

Shared libraries can be used with your wrapped programs.
FreeWrapped applications can load TCL/TK shared library extensions that have been compiled with the STUBS interface.

Your wrapped programs can be customized with your own Windows icons.
The Windows version of freeWrap can incorporate your own customized icon into your wrapped application.

No license fees for wrapped programs.
There are no license fees associated with freeWrap. See the freeWrap license.

Cross-platform generation of programs is supported.
The -w "wrap using" option allows cross-platform creation of wrapped applications without the use of the target operating system.

freeWrap includes several Windows-specific commands
These commands can be used to determine the location of Windows' special directories and make for easy creation of file extension associations and shortcuts.

FreeWrap 7.03 is based on TCL/TK version 9.0.4

This version of freeWrap is built with threads enabled.

Availability

FreeWrap 7.03 executable files are freely available for 64-bit versions of Linux and Windows. Instructions and source code for building freeWrap are included in the freeWrap source code distribution.

TCL-only versions of freeWrap are also available for wrapping TCL (non-TK) scripts.

Visit https://sourceforge.net/projects/freewrap/files/freewrap/ to download files.

What's Inside

TCL/TK

The freeWrap binary distributions contain TCL/TK version 9.0.3.

FreeWrap uses an internal virtual file system which contains the files that normally make up a TCL/TK distribution. From within freeWrap the TCL files normally found at ***/usr/local/lib*** can be accessed from the **//zipfs:/app** directory. For example, the **/usr/local/lib/tdbc1.1.10** directory can be accessed as **//zipfs:/app/tdbc1.1.10** within freeWrap.

The following TCL/TK directories are available within freeWrap:

//zipfs:/app/itcl4.3.8
//zipfs:/app/sqlite3.53.0
//zipfs:/app/tcl_library
//zipfs:/app/tcl9
//zipfs:/app/tcllib2.0
//zipfs:/app/tdbc1.1.13
//zipfs:/app/tdbcmysql1.1.13
//zipfs:/app/tdbcodbc1.1.13
//zipfs:/app/tdbcpostgres1.1.13
//zipfs:/app/thread3.0.6
//zipfs:/app/tk_library
//zipfs:/app/tk9.0
//zipfs:/app/tklib0.8

TCL/TK documentation can be found at https://www.tcl-lang.org/man/tcl9.0/.

Combining an Executable and a ZIP Archive

The directory information for most executable formats is at the beginning of the file and the directory information for the ZIP archive format is at the end of the file. This means that you can append extra data to an executable and the operating system will not care and you can add information to the start of a ZIP archive and the program reading the ZIP archive will not care. There is nothing to prevent us from appending the ZIP archive to the executable that contains a TCL interpreter and thereby putting our entire

application into a single standalone file. The freeWrap application does this.

FreeWrap is a compiled C program that creates a TCL interpreter, reads the TCL initialization scripts from the attached ZIP archive then executes a TCL script from the same archive. TCL's **zipfs** command is used by freeWrap to perform all file additions and deletions to the archive portion of freeWrapped applications.

When you execute freeWrapped applications, the operating system loads and runs the first part of the file as the executable. Then the program mounts the ZIP archive attached at the end of the executable so that is can read the application's TCL scripts from the archive.

To build a wrapped application freeWrap simply inserts a user's application files into a copy of freeWrap and configures it to call the user's code at startup.

TCL/TK extensions and libraries

The freeWrap binary distributions also contain the following additional TCL/TK extensions and libraries.

Extension/Library	Version
tcllib	2.0
tklib	0.8
Tktable	2.12.2
TWAPI	5.2

tcllib documentation can be found at https://core.tcl-lang.org/tcllib/doc/trunk/embedded/md/toc.md.

The **tcllib** library files are included within freeWrap at the following virtual directory.
//zipfs:/app/tcllib2.0

tklib documentation can be found at https://core.tcl-lang.org/tklib/doc/trunk/embedded/www/toc.html.

The **tklib** library files are included within freeWrap at the following virtual directory.
//zipfs:/app/tklib0.8

Tktable documentation can be found at http://tktable.sourceforge.net/tktable/doc/tkTable.html.

Tktable support files are included within freeWrap at the following virtual directory.
//zipfs:/app/TkTable

TWAPI documentation can be found at https://twapi.magicsplat.com/v5.0/index.html.

TWAPI support files are included within freeWrap at the following virtual directory.
//zipfs:/app/twapi

FreeWrap 7.03 Documentation

freeWrap as a TCL/TK wrapper program

FreeWrap can wrap TCL/TK applications that consist of multiple script and binary image files. FreeWrap combines all the files into a single executable file. The syntax for wrapping an application is described below.

Calling Syntax:	freewrap MainProg [-debug] [-e encKey] [-f FileDir] [-forcewrap] [-i ICOfile] [-o OutFile] [-p] [-w freewapProg] File1 ... FileN	
where:	**MainProg**	A file path to a single TCL/TK program script
	File1 ... FileN	A list of space-separated text or binary files to include in the wrapped application.
	-debug	Opens a console window so user can see debug messages while wrapping.
	-e	Specifies that the following password (encKey) should be used to encrypt all of the wrapped files in your application.
	-f	The name of a directory containing all the files you want to wrap.
	-forcewrap	Force freeWrap to act as a wrapping program even if it has been renamed.
	-i	Substitute the following named Windows ICO file (ICOfile) as the program application icon.
	-o	Indicates that the name of the produced executable program should be set to OutFile. This may be a full or relative file path.
	-p	Create a freeWrap program package instead of an executable program.
	-w	Specifies that the following named freeWrap program should be used to create the wrapped application. This option allows building an application targeted for execution on a different operating system. Example usage: /apps/freewrap myprog.tcl -w /winapps/win64/freewrap.exe In this example a Linux version of freeWrap is used to produce a wrapped application for 64-bit Windows.
output:	MainProg (Linux) or MainProg.exe (Windows) **Note**: the output file will be placed in the directory from which freeWrap is called.	

Wrapped files are placed in the //zipfs:/app directory of the wrapped application's virtual file system. The resulting executable program can access the wrapped files by referring to them by their full path within the virtual file system. However, it may be easier to just create a TCL package with its own pkgIndex.tcl file and then wrap the package files along with your application. Please see the information below on how to wrap a package extension. Please see Naming and referring to wrapped files for more details on how to refer to wrapped files within the application.

Both text and binary files can be wrapped.

-debug Option

Use of the -debug option will display the freeWrap console window so any warning messages can be viewed in the console while wrapping.

-e Option

Use the -e option to encrypt, using a user specified password, all files contained in your wrapped application.

-f Option

For larger wrap projects, you may want to use freeWrap's -f option to specify a subdirectory that contains files to include in the wrapped application. The subdirectory and its contents will be added to the application under the virtual file system path *//zipfs:/app*. For example:

```
freewrap myApp.tcl -f libfiles
```

will create a virtual file system path of *//zipfs:/app/libfiles* which contains all the files and subdirectories in the *libfiles* directory.

This option can be used more than once on the same command line in order to include more than one directory.

Use of the -f option does not preclude the specification of individual files on the freeWrap command line.

-forcewrap Option

The -forcewrap option can be used to force freeWrap to act as a wrapping program even if it has been renamed. Without this command line option, freeWrap behaves like a WISH shell when it has been renamed.

-i Option

This option will replace the freeWrap icon within your wrapped application with the contents of the specified ICO file. Use the -i option to specify the file containing the icons you wish to use for your wrapped application. This option is only relevant when wrapping an application for the Windows operating system.

Example: `freewrap myprog.tcl -i myprog.ico`

In this example, *myprog.ico* must be an ICO formatted file. Each icon in this file must be stored as a BMP image. Icons in the PNG format will not be replaced. The -i option will only replace an icon if the new version is exactly the same size as the original icon in freeWrap. PNG files are compressed and, therefore, a new PNG formatted icon is unlikely to be the same size as the original freeWrap icon of the same dimensions and color depth.

When creating your ICO file, keep in mind that freeWrap contains the following versions of the freeWrap icon in a BMP format.

Size	Colors
16x16	16
16x16	256

Size	Colors
32x32	4-bit
32x32	8-bit
32x32	24-bit
48x48	4-bit
48x48	8-bit
48x48	24-bit
128x128	24-bit
256x256	24-bit

Icons, of these resolutions, found in your ICO file will be used to replace the freeWrap icons. If your ICO file doesn't contain at least these versions of your icon then only the matching icons will be replaced. This would leave a freeWrap icon that could be displayed by Windows at some time.

-o Option

The -o Option allows you to specify the name of the executable program you are creating/wrapping. The option value may be either single file name, a full file path, or a relative file path.

-p Option

Using the -p option creates a wrapped application without the freeWrap executable component. This file is called a freeWrap program package. A freeWrap program package can be run using freeWrap as a single-file shell By default, freeWrap program packages are given a file extension of *fwp*.

Example wrapping: freewrap myapp.tcl -p

Example execution: freewish myapp.fwp

-w Option

By default, freeWrap attaches the wrapped files to a copy of the freeWrap program you use to do the wrapping. The -w option allows attaching the wrapped files to a different copy of freeWrap. Since freeWrap is available for multiple operating systems, this feature is useful for assembling freeWrapped applications for other operating systems while on a single computer.

Example (assembling a Windows version while running freeWrap on Linux):

```
freewrap myprog.tcl -w ..\FW_binaries\win64\freewrap.exe
```

Example (assembling a Linux version while running freeWrap on Windows):

```
freewrap myprog.tcl -w ..\FW_binaries\linux64\freewrap
```

Remember, the argument following the -w option must be the file path to a version of the freeWrap program that can execute on the other operating system.

freeWrap as a single-file WISH interpreter

Renaming the freeWrap program to some other file name causes freeWrap to behave as an a stand-alone, single-file WISH that can be used to run any TCL/TK script or freeWrap program package. This can be done in the following manner.

Copy freewrap.exe to a new file name

Example: `copy freewrap.exe  wishrun.exe`

Use the new file as you would normally use WISH

Example: `wishrun  script_name.tcl`

freeWrap response to untrapped errors

FreeWrap will respond to untrapped script errors in the following manner:

Application Type	***Response to Untrapped Script Error***
Wrapped console applications (created with freewrapTCLSH)	Wrapped application prints resulting error message to standard error stream.
Wrapped graphical applications (created with freewrap)	Wrapped application displays message window which must be acknowledged before the user can interact with the application again.
Running a script from the command line as a single-file TCLSH interpreter. Example: freewishTCLSH myErrorScript.tcl	The associated error message will print to the standard error stream.
Running a script from the command line as a single-file WISH interpreter. Example: freewish myErrorScript.tcl	Errors that occur before TK enters its event loop will cause the TK console to open and display the error message. An example of this case can be seen in the following line of code which is not contained in a procedure and executes immediately when starting the script. *puts Hello World* Errors resulting from delayed code execution display a message window which must be acknowledged before the user can interact with the application again. Examples of this case can be seen in the following lines of code. *after 3000 {puts Hello World}*

Application Type	*Response to Untrapped Script Error*
	proc sayHello {} { puts Hello World }
Sourcing a script when interactively running as a single-file TCLSH interpreter.	The associated error message will print to the standard error stream. Example session: *% source dtest1.tcl* *can not find channel named "Hello"* *%*
Sourcing a script when interactively running as a single-file WISH interpreter.	Errors that occur before TK re-enters its event loop will print the error message to the TK console. An example of this case can be seen in the following line of code which is not contained in a procedure and executes immediately when sourcing the script. *puts Hello World* Errors resulting from delayed code execution display a message window which must be acknowledged before the user can interact with the application again. Examples of this case can be seen in the following lines of code. *after 3000 {puts Hello World}* *proc sayHello {} { puts Hello World }*

freeWrap program packages

FreeWrap normally produces an executable file when wrapping an application. However, it is also possible the create a file that only contains the wrapped files for the application. This allows you to distribute smaller packages that can later be run using freeWrap as a single-file TCLSH or WISH interpreter. A freeWrap program package can contain all the files for your application in a single compressed file.

Use the -p option when wrapping your application in order to create a freeWrap program package instead of an executable file.

Example wrapping: **freewrap myApp.tcl -f MyAppDirectory -p**

Example execution: **freewish myapp.fwp**

FreeWrap program packages are not encrypted and can be run using a copy of freeWrap 7.0 or later.

When run, the freeWrap program package is mounted at *//zipfs:/FWpkg0* in the virtual file system. In the wrapping example above, **myApp.tcl** would be found at *//zipfs:/FWpkg0/myApptcl* and the wrapped directory **MyAppDirectory** would be found at *//zipfs:/FWpkg0/myAppDirectory*. The script myApp.tcl would be immediately executed when the program package is run.

freeWrap's console window

Under freeWrap, the TCL/TK ***console*** command is available for both Windows and UNIX. The console window is the location that will receive any STDOUT or STDERR output. The console can also be used to interactively enter TCL/TK commands. Use ***console show*** to display the window and ***console hide*** to remove it.

Global variable argv0

When running freeWrap or a wrapped application, the argv0 global variable will be set to the name of the executable program. For a wrapped application, this is the name of the executable program produced by wrapping. For a standalone TCL shell, this is the name of the executable shell program.

Using the DDE and Registry packages (Windows only)

The DDE and Registry packages have been statically compiled into freeWrap. There is no need to load them with a ***package require*** command. Simply use the ***dde*** and ***registry*** commands without any preceding ***package require*** command.

Using wrapped files.

When running a wrapped application, the first file that was specified on the command line at the time of wrapping will be executed as a TCL/TK script. All other files contained in a directory specified on the freeWrap command line are available, within the virtual file system, to the executing wrapped application.

You CAN do the following with the wrapped files.

1. ***source*** them
2. ***open*** them
3. ***read*** them
4. ***close*** them
5. ***glob*** them. However, you may find that the ***zipfs list*** command does a better job of locating files in the application's virtual file system.
6. **load** them
7. Use any ***file*** commands that do not write to the files
8. Use them with the ***image create*** command
9. Specify them for -***bitmap*** widget options.

You CANNOT do the following with the wrapped files.

1. ***file delete*** them (since they exist in the application, not on disk)

Naming and referring to wrapped files

Details

All files included in a wrapped application must be referred to by their full virtual file system path within the application. However, any relative or full path specification can be used on the freeWrap command line.

Individual files specified on the freeWrap command line are placed in the *//zipfs:/app* directory of the wrapped application's virtual file system.

When wrapping a subdirectory of files the subdirectory structure is maintained within the virtual file system. For example, wrapping directory /usr/home/MyProjects/MyApp that contains subdirectories /usr/home/MyProjects/MyApp/src and /usr/home/MyProjects/MyApp/data will produce the virtual file system subdirectories //zipfs:/app/MyApp/src and //zipfs:/app/MyApp/data.

For example the following wrapping command line:

```
./freewrap MyAppProg.tcl -f MyApp etc/MyAppExtra.tcl
```

will result in the following files in the wrapped application's virtual file system.

```
//zipfs:/app/MyAppProg.tcl

//zipfs:/app/MyAppExtra.tcl

//zipfs:/app/MyApp/src

//zipfs:/app/MyApp/data
```

The easy way

At run time, your application can use the [zipfs list] command to determine exactly where the files are in the virtual directory structure. For example:

```
set filePath [zipfs list */u1.tcl]
```

You may also want to add some code to your application that enables it to find the necessary file either when developing or when wrapped. For example:

```
set filePath {}
if {[namespace exists ::freewrap]} {
    if {$::freewrap::runMode eq {wrappedExec}} {
        set filePath [zipfs list */u1.tcl]
       } {
          set filePath $::env(SrcPath)/utils/u1.tcl
        }
} {
    set filePath $::env(SrcPath)/utils/u1.tcl
  }
```

```
if {$filePath eq {}} {
    puts {Could not find file u1.tcl}
   } {
      source $filePath
      # Place additional code here.
     }
```

Wrapping and using TCL/TK extensions (packages)

TCL/TK extensions can be wrapped into your application and then loaded dynamically at run time. Alternatively, if you are willing to recompile freeWrap, TCL/TK extensions may also be statically compiled into freeWrap. FreeWrap already includes some statically compiled TCL/TK extensions.

Wrapped applications can load TCL/TK shared binary extension that have been compiled with the TEA (i.e., stubs) interface. Stubs-enabled shared libraries can be included in the wrapped application or exist as separate files.

TCL extension packages included as part of freeWrap as well as packages included in your wrapped applications are, in most cases, automatically found by freeWrap. The main exception to this rule are freeWrap program packages. These packages normally require you to add a path to **auto_path** which corresponds to the subdirectory (or a parent directory) containing the wrapped pkgIndex.tcl files you include in your application.
For example, if wrapping places your file at //zipfs:/app/MyApp/extProg/pkgIndex.tcl you should add //zipfs:/app/MyApp to **auto_path**. This can be easily done by including a command in your application similar to:

lappend auto_path //zipfs:/app/MyApp

The above command should be executed prior to the ***package require*** command that loads the TCL extension.

Extensions containing a single binary file

TCL extensions with a single binary file (other script files may be present) are also easy to wrap if:

1. The binary library does not call other binary libraries located inside the wrapped application.
2. The binary library does not call scripts located inside the wrapped application.

Your application code simply needs to add the path to the binary extension's pkgIndex.tcl file to the auto_path variable so that the usual ***package require*** or ***load*** commands will find the extension.

The **zipfs list** command can be used to make setting **auto_path** easier. For example:

lappend ::auto_path [file dirname [zipfs list */myExtension]]

Special variables and commands provided by freeWrap

The ::freewrap namespace

FreeWrap has a namespace which contains all of the freeWrap specific variables, commands and procedures. These variables, commands and procedures may be referenced using the ::freewrap:: prefix or imported into any other namespace.

Variables

The following variables are defined in the ::freewrap namespace of each wrapped application.

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>patchLevel</td><td>Revision level of the freeWrap program used to wrap the application.</td></tr>
<tr><td>progname</td><td>The proper name for the freeWrap program for the current operating system. This is normally freewrap.exe under Windows and freewrap under UNIX.</td></tr>
<tr><td>runMode</td><td>This variable indicates whether freeWrap is running as:
<table>
<tr><th>Value of variable</th><th>Meaning</th></tr>
<tr><td>interactiveShell</td><td>an interactive shell</td></tr>
<tr><td>programPackage</td><td>a wrapped application without the freeWrap executable core.</td></tr>
<tr><td>standAloneShell</td><td>a stand-alone shell running a script</td></tr>
<tr><td>wrappedExec</td><td>a wrapped executable program</td></tr>
</table>
</td></tr>
</table>

Procedures

The following procedures are defined in the ::freewrap namespace of each wrapped application. The commands names starting with shell_ are only available under the Windows operating system.

Syntax: isSameRev file_name

Description: Checks whether the specified file contains a copy of the same freeWrap revision as the currently executing program.

Returns: 0, if file does not contain a copy and 1, if file contains a copy.

Syntax: iswrapped file_name

Description: Determines whether the file named file_name is a freeWrapped application.

If file_name is a freeWrapped application this procedure returns a value of 1.

If file_name is NOT a freeWrapped application this procedure returns a value of 0.

Syntax: shell_assoc_exist extension

Description: Check whether a key exists for an extension

Example: shell_assoc_exist .txt => 1
Example: shell_assoc_exist .NEVER => 0

Syntax: shell_fileType_exist fileType

Description: Determine whether a file type exists

Example: shell_fileType_exist txtfile => 1
Example: shell_fileType_exist NEVER => 0

Syntax: shell_fileExtension_setup extension, fileType

Description: Creates a file extension and associates it with fileType.
Example: shell_fileExtension_setup .txt txtfile

Remove connection between extension and fileType
Example: shell_fileExtension_setup .txt ""

Syntax: shell_fileType_setup fileType, title

Description: Creates a file type.
Example: shell_fileType_setup txtfile "Text Document"

Syntax: shell_fileType_open fileType, openCommand

Description: Creates an open command. Sets action for double click.
Example: shell_fileType_open txtfile "C:\\WINDOWS\\NOTEPAD.EXE %1"

Syntax: shell_fileType_print fileType, printCommand

Description: Creates a print command for right mouse button menu.
Example: shell_fileType_print txtfile "C:\\WINDOWS\\NOTEPAD.EXE /p %1"}

Syntax: shell_fileType_icon fileType, icon

Description: Sets an icon for a fileType.
Example: shell_fileType_icon txtfile "C:\\WINDOWS\\SYSTEM\\shell32.dll,-152"
Example: shell_fileType_icon txtfile "C:\\mydir\\myicon.ico"

We can give a name.ico file or a dll or exe file here. If a dll or exe file is used the index for the resource inside the file must be specified

Syntax: shell_fileType_quickView fileType, quickViewCmd

Description: Sets the command to execute to perform a quick view for a fileType.
Example: shell_fileType_quickView txtfile "write.exe %1"

Syntax: shell_fileType_addAny_cmd fileType, cmdName, cmd

Adds any command you want to a fileType.
Example: shell_fileType_addAny_cmd scrfile config "%1"

Syntax: shell_fileType_setMenuName fileType, cmdName, str

Description: Change description in right mouse menu for a command associated with a fileType.
Example: shell_fileType_setMenuName txtfile print "Print file"

Syntax: shell_fileType_showExt fileType, yesOrNo

Description: Always show the extension on the fileType.
Example: shell_fileType_showExt txtfile 1

Turn off "Always show" of extension on the fileType
Example: shell_fileType_showExt txtfile 0

Syntax: shell_fileType_setCmdOrder fileType, cmds

Description: Over-ride the default ordering of commands on right mouse menu.
Example: shell_fileType_setCmdOrder txtfile {print open}

Syntax: shell_fileType_neverShowExt fileType, yesOrNo

Description: Never show extension on fileType.
Example: shell_fileType_neverShowExt txtfile 1

Turn off "Never show" of extension on the fileType.
Example: shell_fileType_neverShowExt txtfile 0

Syntax: shell_getCmds file

Description: Retrieves all the commands associated with an extension.
Example: shell_getCmds file.txt => open print

Syntax: shell_getCmd_imp file, cmd

Description: Retrieves the implementation of a command given a file extension and command.
Example: shell_getCmd_imp test.txt open => C:\\WINDOWS\\NOTEPAD.EXE %1

Commands

The following TCL commands are defined in the ::freewrap namespace of each wrapped application.

Syntax: getSpecialDir dirType

Description: Find "Start Menu", "Desktop" and similar directory locations under Windows. DirType must be one of the following strings:

ALTSTARTUP	FONTS
APPDATA	HISTORY
BITBUCKET	INTERNET
COMMON_ALTSTARTUP	INTERNET_CACH
COMMON_DESKTOPDIRECTORY	NETHOOD
COMMON_FAVORITES	NETWORK
COMMON_PROGRAMS	PERSONAL
COMMON_STARTMENU	PRINTERS
COMMON_STARTUP	PRINTHOOD
CONTROLS	PROGRAMS
COOKIES	RECENT
DESKTOP	SENDTO
DESKTOPDIRECTORY	STARTMENU
DRIVES	STARTUP
FAVORITES	TEMPLATES

Syntax: Nagle chanID [ON | OFF]

Description: Use this command to report or set the status of the Nagle algorithm for a TCP socket opened by TCL.

Argument	Explanation
chanID	The first command argument must be the name of a currently open socket returned by TCL.
state	This is the second, optional argument for the command. If this argument is not provided then the command returns the current status of the TCP socket specified by chanID. A return value of ON indicates that the Nagle algorithm is currently active for the socket. A return value of OFF indicates that the Nagle algorithm is not currently active for the socket. If the second supplied argument is ON or a TCL logical true value then the Nagle algorithm for the socket will be activated. If the second supplied argument is OFF or a TCL logical false value then the Nagle algorithm for the socket will be turned off.

If the second argument is present, the resulting status of the Nagle algorithm will be returned as the result.

Syntax: shortcut linkPath[-objectPath objectPath] [-description description] [-workingDirectory dir] [-icon path index] [-arguments args]

Description: Creates a Windows shortcut. The only required parameter is the linkPath. This means you can create a shortcut with no target, which probably isn't useful. The icon of the shortcut will default to the icon of the target item if not specified.

Argument	Explanation
linkPath	The path to the shortcut file (including the extension .lnk)
objectPath	The target of the link
description	Shortcut description
workingDirectory	Working (startup) directory for the target of the shortcut
path index (icon)	Specifies the path to a file and the index of the icon in that file to use for the shortcut
args	Arguments passed to the target of the shortcut when started.

How freeWrap encryption works

You can now specify your own encryption key when wrapping your application. Doing so will encrypt all files stored into the wrapped application's internal Virtual File System. FreeWrap uses the encryption capability of TCL's **zipfs** command to provide this feature.

Building freeWrap

FreeWrap does not include any alterations to the TCL/TK core. It is plain-vanilla TCL/TK with a few extensions such as Tktable thrown in.

Dependencies

Building the freeWrap program itself requires the following additional libraries not provided with the freeWrap distribution:

1. TCL (Tool Command Language) static library
2. TK (Tool Kit for TCL) static library
3. Zlib compression static library
4. Tktable extension
5. TCLmtls extension
6. tcllib and tklib
7. TWAPI (TCL Windows API extension) - for Windows builds only

Build environment

Current distribution

The freeWrap 7.03 executable program distributions were built under the following environments:

64-bit Linux

Built on OpenSUSE 15.2 64-bit.

Should run on any recent 64-bit Linux distribution.

64-bit Windows

Built on Windows 11 using the MSYS2 platform with the mingw-w64 tool chain targeting 64-bit Windows. The following MSYS2 installer was retrieved from www.msys2.org:

msys2-x86_64-20250830.exe

MSYS2 requires 64-bit Windows 7 or newer. After running the MSYS2 installer, an MSYS2 UCRT64 terminal was opened and the following commands were executed in order to install MAKE and a few other basic development utilities.

```
pacman -S base-devel
```

The following command was next run to install mingw-w64 support to generate 64-bit programs.

```
pacman -S mingw-w64-x86_64-toolchain
```

The following command should be run to install additional C headers required for TWAPI compilation.

```
pacman -S mingw-w64-x86_64-libtommath
```

Ensure you add the minGW bin path to the PATH variable by modifying the .bash_profile file.

FreeWrap 7.03 Build Process

All necessary source code, other than for freeWrap, should be copied to a location with a common top directory. For example:

/Dev/fwbuild/win64/dyncall-1.4

/Dev/fwbuild/win64/tclmtls-1.1.0

/Dev/fwbuild/win64/tcl9.0

/Dev/fwbuild/win64/tk9.0

/Dev/fwbuild/win64/tcllib-2.0

/Dev/fwbuild/win64/tklib-0.8

/Dev/fwbuild/win64/Tktable-2.12.2

/Dev/fwbuild/win64/twapi-5.2

/Dev/fwbuild/win64/zlib-1.3.2

Under Linux freeWrap should be compiled using the normal GNU make and gcc programs.

Under Windows freeWrap is built using MSYS2 and minGW. The MSYS2 and minGW environment allows freeWrap to be built with almost the same procedures under both Windows and Linux. For Windows, perform all package builds from an MSYS2 console.

The freeWrap distribution includes a separate Makefile for each target operating system. Extract the freeWrap source code archive files to a directory of your choosing. Building freeWrap for Linux is done from the *linux64* subdirectory of the extracted source code archive. Building freeWrap for Windows is done from the *win64* subdirectory of the extracted archive. Makefiles appropriate for the associated operating system can be found in these two directories.

Use the following steps to build under either Linux or Windows.

The following procedural steps have been written with the assumption that the resulting compilation products will be installed under the /usr/local subdirectory.

1. Compile static library for TCL

Ensure that the Info-ZIP program cannot be found in any of the subdirectories specified in the PATH environment variable. If Info-ZIP is present then the TCL configuration script will generate a Makefile that uses Info-ZIP. If Info-ZIP is not present then TCL will be built using the *minizip* program. These freeWrap build instructions are written for the case where the *minizip* program is used.

A static TCL library should be built by using the configure script found in the operating system specific directory of the TCL distribution (i.e., ***unix*** directory for Linux, ***win*** directory for Windows). Change to this directory and issue the following command line.

```
./configure --prefix=/usr/local --exec_prefix=/usr/local --disable-shared
```

Next, use the normal

```
make
```

command to perform the compilation.

Unfortunately, under Linux, the normal **make** process compiles TCL's sqlite source code for use with the TCL stubs interface. We need to correct this since we want to statically link sqlite into freeWrap. After running **make** once, perform the following to make this correction:

a. From the normal TCL build directory (unix), edit the Makefile found under pkgs/sqlite3.53.0

b. Modify the Makefile by removing the following options from the definition of DEFS:

   -DUSE_TCL_STUBS=1
   -DUSE_TCLOO_STUBS=1

c. Delete the pkgs/sqlite3.53.0/tclsqlite3.o file.

d. Run **make** again from the operating system specific directory of the TCL distribution (unix).

Use the normal

```
make install
```

command to complete the installation of TCL

2. Compile static library for TK

A static TK library should be built by using the configure script found in the

operating system specific directory of the TK distribution (i.e., ***unix*** directory for Linux, ***win*** directory for Windows). Change to this directory and issue the following command line.

```
./configure --prefix=/usr/local --exec_prefix=/usr/local --disable-shared
```

You will need to copy the minizip.exe program from the TCL compilation directory into the TK compilation directory with a command similar to:

```
cp ../../tcl9.0/unix/minizip .
```

Next, use the normal

```
make
```

(When asked whether to overwrite libtk*.zip answer [a]ppend)

```
make install
```

commands to perform the compilation and installation.

3. **Build zlib compression library**

Obtain the source code package for version 1.3.2 of the **zlib** general purpose data compression library.

Source code for this library is available from https://zlib.net

Under Linux

Run the following commands from the top of the source tree:

```
./configure -static
make
make install
```

Under Windows

Build the **zlib** libraries using the Makefile.gcc file under the win32 directory of the **zlib** source code tree.

You will need to add the following lines at the top of the win32/Makefile.gcc file after the definition of IMPLIB:

```
SHAREDLIB =
INCLUDE_PATH=/usr/local/include
LIBRARY_PATH=/usr/local/lib
BINARY_PATH=/usr/local/bin
```

Run the following commands from the top of the source tree. Ignore any error about libz.dll.a

```
make -f win32/Makefile.gcc
make -f win32/Makefile.gcc install
```

4. Build the Tktable TK extension

Obtain **Tktable 2.12.2** from
https://chiselapp.com/user/bohagan/repository/TkTable/uv/Tktable2.12.2.tar.gz.

Extract the contents of the Tktable file.

From the top of the Tktable source tree, run the configure script with the --disable-shared option.

Under Linux

```
./configure --disable-shared --enable-64bit --prefix=/usr/local
--libdir=/usr/local/lib64 -with-tcl=/usr/local/lib64
-with-tk=/usr/local/lib64
```

Under Windows

```
./configure --disable-shared --enable-64bit --prefix=/usr/local
--libdir=/usr/local/lib -with-tcl=/usr/local/lib -with-tk=/usr/local/lib
```

This will create a Tktable Makefile that will produce a static library which the freeWrap Makefile will use.

However, before using the Tktable Makefile you must look for and remove all instances of the following text from the definition of DEFS in Makefile.

```
-DUSE_TCL_STUBS=1
-DUSE_TCLOO_STUBS=1
-DUSE_TK_STUBS=1
```

Under Windows, before using the Tktable Makefile, we must correct the TCLSH_PROG parameter in the Makefile to so that it points to the TCLSH we built:

```
TCLSH_PROG = /usr/local/bin/tclsh90s.exe
```

After making these changes to the Makefile you can then do the normal:

```
make
```

```
make install
```

5. Build the TCLmtls extension

The TCLmtls extension provides all the functionality of the TLS extension. Unlike the TLS extension, which requires the installation of OpenSSL libraries, this extension uses the mbedTLS libraries. This allows us to statically link the TCLmtls extension without any additional dependency on another shared library.

The TCLmtls extension is hosted in a **git** repository. Therefore, you must install git on the operating system platform you will be using to compile freeWrap. Under Windows, *Git for Windows* was used. For Linux, the Linux distribution's standard software repositories were used to install the **git** application.

a. Issue the following commands in a console window to retrieve then compile TCLmtls. Before running the commands, change the console's current directory to the top of your project's source code directory tree.

```
git clone --branch v1.1.0 --depth 1 https://github.com/chpock/tclmtls.git

cd tclmtls

git submodule update --init --recursive

mkdir build

cd build

../configure --disable-shared --disable-stubs
make

make install
```

b. Unfortunately, there are three libraries in this build that the **`make install`** command above does not install. Under Linux, you need to copy the following files to the /usr/local/lib64/mtls1.1.0 directory. For Windows, copy the files to /usr/local/lib/mtls1.1.0.

build/mbedtls/library/libmbedcrypto.a
build/mbedtls/library/libmbedtls.a
build/mbedtls/library/libmbedx509.a

6. Build the TWAPI TCL extension

If building freeWrap for Windows and you want to include the TWAPI extension then:

(a) Retrieve the TWAPI version **5.2** source code from https://github.com/apnadkarni/twapi/tags.

Extract the TWAPI source code package to the top TWAPI source code directory.

(b) Create a directory named **build** immediately under the top of the TWAPI source code

tree.

(c) Open an MSYS2 console and change to the new **build** directory.

(d) On Windows enter the following commands into the MSYS2 console.

```
../configure --enable-threads --disable-shared --enable-64bit --
exec=/usr/local --exec-prefix=/usr/local
```

(e) Modify build/Makefile by adding **-I$(includedir)** at the end of the definition of INCLUDES.

(f) Next, issue the following commands in the MSYS2 console.

```
make clean
make
make install
```

(g) Retrieve the dyncall source code from http://www.dyncall.org.

(h) Open an MSYS2 console and change to the top of the *dyncall* source code directory tree.

(i) Enter the following commands into the MSYS2 console.

```
make -f Makefile.embedded CC=gcc
make -f Makefile.embedded install DESTDIR=/usr/local
```

where /usr/local is the path to where the generated *dyncall* library, include, and man page files should be copied.

7. Install the tcllib package

After copying the tcllib package to the common source code directory structure, install the tcllib-2.0 library by running the following two commands from the top of the tcllib package directory. Ignore any errors about critcl.

```
./configure --exec=/usr/local --exec-prefix=/usr/local
make install
```

Ignore any error messages pertaining to critcl.

8. Install the tklib package

After copying the tklib package to the common source code directory structure, install the tklib-0.8 library by running the following two commands from the top of the tklib package directory.

```
./configure --exec=/usr/local --exec-prefix=/usr/local
make install
```

9. Compile freeWrap

Use the Makefile that comes with the freeWrap source code distribution to build freeWrap.

(a) Background Information

The freeWrap Makefile will use the *main.c* and other source code files provided with the freeWrap distribution.

The freeWrap Makefile will use the *src/main.c* file provided with the freeWrap source code distribution to build any version of freeWrap. This file has been written to use TCL's Virtual File System and perform initialization normally done by the standard TCL/TK distribution.

The *main.c* source code is also the location in which any statically linked TCL/TK extensions should be initialized. The file currently contains source code for initializing initializing the registry, dde, sqlite3, Tktable, and TWAPI, and extensions.

The src/Windows/TWAPI_pkgIndex.tcl file is a customized copy of the pkgIndex.tcl file included with the TWAPI binary distributions. This is included into freeWrap during the building of freeWrap with the Makefile. TWAPI_pkgIndex.tcl allows for proper loading and initialization of the individual TWAPI modules.

The *src\Windows\ico.tcl* file is a copy from the **ico** package of tklib corrected to support 256x256 pixel windows icons. This is included into freeWrap during the building of freeWrap with the Makefile.

The *src/freelib.c* and *src/freewrap_main.tcl* files implement some TCL commands added by freeWrap.

The *src/freewrap.tcl* script is the portion of freeWrap that controls the wrapping process.

Files found in the *src/themes* directory provide a number of ttk themes included in freeWrap.

Some necessary files for building under the Windows operating system are included in the *src/Windows* directory. These files are automatically used by the freeWrap Makefile.

(b) Modify the Makefile to reflect the file paths for your computer system.

i. The path defined by INSTALL_BASE within the Makefile must point to the installation directory for the version of TCL/TK from which you are building freeWrap. The Makefile will copy the TCL/TK scripts and libraries it needs from this location.

   The LIB_TCL variable in the freeWrap Makefile must point to the static libraries created by the TCL compilation. These files are usually stored someplace like /usr/local/lib/libtcl9.0.a when "make install" is run as part of the TCL build process.

ii. The path defined by the BUILD_DIR must point to the top level directory for the

TCL and TK source code and all the other packages that must be built in order to make freeWrap.

(c) **Run *make* to build freeWrap.**

After modifying the freeWrap Makefile, issue the following command from the operating system specific directory (linux64 for linux, win64 for Windows), of the freeWrap build tree, to build the TCL/TK version of freeWrap.

```
make
```

This will create the programs ***freewrap*** and ***freewish***. The former program can be used to wrap your TCL/TK applications. The latter is exactly the same file as the ***freewrap*** file except that it has been renamed and, therefore, will behave as a stand-alone wish application.

To build the TCL-only version of freeWrap issue the following command.

```
make FW_EXT=TCLSH
```

This will create the programs ***freewrapTCLSH*** and ***freewishTCLSH***. The former program can be used to wrap your TCL applications. The latter is exactly the same file as the ***freewrapTCLSH*** file except that it has been renamed and, therefore, it will behave as a stand-alone tclsh application.

Virtual File System

Introduction

The freeWrap program is a TCL script that has been attached to a single-file version of the WISH shell program. The necessary files used by WISH are contained in a ZIP archive attached to the end of the WISH application.

FreeWrap was created using the **zipfs** capabilities of TCL 9. TCL's **zipfs** command provides the ability to mount the contents of a ZIP archive file as a virtual file system. When mounted, all of the contents of the ZIP archive appear to be files contained within the directory on which the ZIP file is mounted.

See the description of the **zipfs** command in the TCL 9 documentation for information on how to mount a ZIP file.

Limitations

Files within mounted archives can be written to and new files or directories cannot be created within the mounted directory. Further, modifications to files are limited to the mounted archive in memory and are not persisted to disk.

Searching for a file in the virtual file system can be done using the **zipfs** command. The **[zipfs list]** command does have a *-glob* option that can be used. TCL's **glob** command can also be used to find virtual file system files. However, you may find that the [**zipfs list]** command does a better job of locating files in the application's virtual file system.