face 英文 訳文

NAME

wish --- Simple windowing shell

SYNOPSIS

wish ?-encoding name? ?fileName arg ...?

OPTIONS

-encoding name
: Specifies the encoding of the text stored in fileName. This option is only recognized prior to the fileName argument. fileName に格納されているテキストのエンコーディングを指定します。 このオプションは fileName 引数の前に認識されます。

-colormap new
: Specifies that the window should have a new private colormap instead of using the default colormap for the screen. ウィンドウが画面のデフォルトのカラーマップを使用するのではなく、 新しいプライベートカラーマップを持つように指定します。

-display display
: Display (and screen) on which to display window. ウィンドウを表示するディスプレイ（および画面）。

-geometry geometry
: Initial geometry to use for window. If this option is specified, its value is stored in the geometry global variable of the application's Tcl interpreter. ウィンドウに使用する初期ジオメトリ。 このオプションが指定されると、その値はアプリケーションの Tcl インタープリタの geometry グローバル変数に格納されます。

-name name
: Use name as the title to be displayed in the window, and as the name of the interpreter for send commands. name をウィンドウに表示されるタイトルとして使用し、 send コマンドのインタープリタ名として使用します。

-sync
: Execute all X server commands synchronously, so that errors are reported immediately. This will result in much slower execution, but it is useful for debugging. すべての X サーバーコマンドを同期的に実行し、 エラーが即座に報告されるようにします。 これにより実行速度は非常に遅くなりますが、デバッグには役立ちます。

-use id
: Specifies that the main window for the application is to be embedded in the window whose identifier is id, instead of being created as an independent toplevel window. Id must be specified in the same way as the value for the -use option for toplevel widgets (i.e. it has a form like that returned by the winfo id command). Note that on some platforms this will only work correctly if id refers to a Tk frame or toplevel that has its -container option enabled. アプリケーションのメインウィンドウが、 独立したトップレベルウィンドウとして作成される代わりに、 id で指定されたウィンドウに埋め込まれるように指定します。 Id はトップレベルウィジェットの -use オプションの 値と同じ方法で指定する必要があります （つまり、 winfo id コマンドに よって返される形式のようになります）。 一部のプラットフォームでは、id が -container オプションが有効になっている Tk の frame または toplevel を参照している 場合にのみ、これが正しく機能することに注意してください。

-visual visual
: Specifies the visual to use for the window. Visual may have any of the forms supported by the Tk_GetVisual procedure. ウィンドウに使用するビジュアルを指定します。 ビジュアルは Tk_GetVisual 手順でサポートされている形式のいずれかを持つことができます。

--
: Pass all remaining arguments through to the script's argv variable without interpreting them. This provides a mechanism for passing arguments such as -name to a script instead of having wish interpret them. 残りのすべての引数を解釈せずにスクリプトの argv 変数に渡します。 これにより、wish が解釈する代わりに、 -name などの引数をスクリプトに渡す メカニズムが提供されます。

DESCRIPTION

Wish is a simple program consisting of the Tcl command language, the Tk toolkit, and a main program that reads commands from standard input or from a file. wish は Tcl コマンド言語、Tk ツールキット、 標準入力やファイルからコマンドを読むメインプログラムからなる シンプルなプログラムである. It creates a main window and then processes Tcl commands. wish はメインウ

インドウを生成し Tcl コマンドを処理する. If wish is invoked with arguments, then the first few arguments, ?-encoding name? ?fileName?, specify the name of a script file, and, optionally, the encoding of the text data stored in that script file. wish が引数ありで起動されたとき、最初の数個の引数 ?-encoding name? ?filename? は スクリプトファイルの名前と、オプションでスクリプトファイルに 格納されたテキストデータのエンコーディングを指定している. A value for fileName is recognized if the appropriate argument does not start with "-". 妥当な引数が "-" で始まらないなら、ファイル名の値と見なされる.

If there are no arguments, or the arguments do not specify a fileName, then wish reads Tcl commands interactively from standard input. 引数がない場合、または引数がファイル名を指定していない場合、 wish は標準入力から対話的に Tcl コマンドを読み取る。 It will continue processing commands until all windows have been deleted or until end-of-file is reached on standard input. すべてのウィンドウが削除されるか、標準入力で EOF（ファイルの終わり）に 達するまでコマンドの処理を続ける。 If there exists a file .wishrc in the home directory of the user, wish evaluates the file as a Tcl script just before reading the first command from standard input. ユーザーのホームディレクトリに .wishrc というファイルが 存在する場合、wish は標準入力から最初のコマンドを読み取る前に そのファイルを Tcl スクリプトとして評価する。

If arguments to wish do specify a fileName, then fileName is treated as the name of a script file. wish の引数がファイル名を指定している場合、 そのファイル名はスクリプトファイルの名前として扱われる。 Wish will evaluate the script in fileName (which presumably creates a user interface), then it will respond to events until all windows have been deleted. Wish はファイル名のスクリプトを評価し（おそらくこれにより ユーザーインターフェイスが作成される）、その後すべてのウィンドウが 削除されるまでイベントに応答する。 Commands will not be read from standard input. コマンドは標準入力からは読み取られない。 There is no automatic evaluation of .wishrc when the name of a script file is presented on the wish command line, but the script file can always source it if desired. スクリプトファイルの名前が wish コマンドラインに提示された場合、 .wishrc の自動評価は行われないが、 必要に応じてスクリプトファイルからそれを source することはできる。

Note that on Windows, the wishversion.exe program varies from the tclshversion.exe program in an additional important way: it does not connect to a standard Windows console and is instead a windowed program. Windows では、wishversion.exe プログラムが tclshversion.exe プログラムと 異なる追加の重要な点があることに注意： それは標準の Windows コンソールに接続せず、 代わりにウィンドウ化されたプログラムであるという点である。 Because of this, it additionally provides access to its own console command. このため、独自の console コマンドへのアクセスも提供する。

OPTION PROCESSING

Wish automatically processes all of the command-line options described in the OPTIONS summary above. Wishは、上記の OPTIONS 概要に記載されているすべてのコマンドラインオプションを 自動的に処理します。 Any other command-line arguments besides these are passed through to the application using the argc and argv variables described later. これら以外のコマンドライン引数は、後で説明する argc および argv 変数を使用して アプリケーションに渡されます。

APPLICATION NAME AND CLASS

The name of the application, which is used for purposes such as send commands, is taken from the -name option, if it is specified; otherwise it is taken from fileName, if it is specified, or from the command name by which wish was invoked. アプリケーションの名前は、 send コマンドなどの目的に使用され、 指定されていれば -name オプションから取得されます。 それ以外の場合は、指定されていれば fileName から、または wish が呼び出されたコマンド名から取得されます。 In the last two cases, if the name contains a "/" character, then only the characters after the last slash are used as the application name. 最後の二つの場合、名前に「/」文字が含まれていると、 最後のスラッシュ以降の文字だけがアプリケーション名として使用されます。

The class of the application, which is used for purposes such as specifying options with a RESOURCE_MANAGER property or .Xdefaults file, is the same as its name except that the first letter is capitalized. アプリケーションのクラスは、RESOURCE_MANAGER プロパティや .Xdefaults ファイルでオプションを指定するなどの目的に 使用され、最初の文字が大文字になることを除いて、その名前と同じです。

VARIABLES

Wish sets the following Tcl variables: Wish は以下の Tcl 変数を設定します:

argc

: Contains a count of the number of arg arguments (0 if none), not including the options described above. arg 引数の数をカウントした値を含みます (引数がない場合は 0)。 上記で説明したオプションは含みません。

argv

Contains a Tcl list whose elements are the arg arguments that follow a -- option or do not match any of the options described in OPTIONS above, in order, or an empty string if there are no such arguments. -- オプションの後に続く arg 引数、または上記の OPTIONS で 説明されたオプションに一致しない引数を順番に含む Tcl リスト、 またはそのような引数がない場合は空の文字列を含みます。

argv0

Contains fileName if it was specified. Otherwise, contains the name by which wish was invoked. fileName が指定されていればその値を含みます。 それ以外の場合は、wish が呼び出された名前を含みます。

geometry

If the -geometry option is specified, wish copies its value into this variable. -geometry オプションが指定されている場合、wish はその値を この変数にコピーします。 If the variable still exists after fileName has been evaluated, wish uses the value of the variable in a wm geometry command to set the main window's geometry. fileName が評価された後もこの変数が存在する場合、 wish は wm geometry コマンドでメインウィンドウのジオメトリを設定するためにこの変数の値を使用します。

tcl_interactive

Contains 1 if wish is reading commands interactively (fileName was not specified and standard input is a terminal-like device), 0 otherwise. wish が対話的にコマンドを読み込んでいる場合 (fileName が指定されておらず、標準入力が端末のような デバイスである場合) は 1 を、そうでない場合は 0 を含みます。

SCRIPT FILES

If you create a Tcl script in a file whose first line is ファイルの最初の行に次の行があるファイルに Tcl スクリプトを 作成する場合

```
#!/usr/local/bin/wish
```

then you can invoke the script file directly from your shell if you mark it as executable. ファイルを実行可能としてマークすると、 シェルからスクリプトファイルを直接呼び出すことができます。 This assumes that wish has been installed in the default location in /usr/local/bin; if it is installed somewhere else then you will have to modify the above line to match. これは、wish が /usr/local/bin にデフォルトで インストールされていることを前提としています。 別の場所にインストールされている場合は、上記の行を修正する必要が あります。 Many UNIX systems do not allow the #! line to exceed about 30 characters in length, so be sure that the wish executable can be accessed with a short file name. 多くの UNIX システムでは、#! 行が約30文字を 超えることを許可していないため、wish 実行ファイルに 短いファイル名でアクセスできることを確認してください。

An even better approach is to start your script files with the following three lines: さらに良い方法は、スクリプトファイルを次の3行で始めることです:

```
#!/bin/sh
# the next line restarts using wish \
exec wish "$0" ${1+"$@"}
```

This approach has three advantages over the approach in the previous paragraph. この方法には、前の段落の方法に比べて3つの利点があります。 First, the location of the wish binary does not have to be hard-wired into the script: it can be anywhere in your shell search path. 第一に、wish バイナリの場所をスクリプトに固定する 必要がありません。シェル検索パスのどこにでも配置できます。 Second, it gets around the 30-character file name limit in the previous approach. 第二に、前の方法の30文字のファイル名制限を回避します。 Third, this approach will work even if wish is itself a shell script (this is done on some systems in order to handle multiple architectures or operating systems: the wish script selects one of several binaries to run). 第三に、この方法は、wish 自体がシェルスクリプトで あっても機能します (これは、複数のアーキテクチャや OS を 処理するためにいくつかのシステムで行われます: wish スクリプトはいくつかのバイナリの1つを 選択して実行します)。 The three lines cause both sh and wish to process the script, but the exec is only executed by sh. この3行により、sh と wish の両方が スクリプトを処理しますが、 exec は sh のみが実行します。 sh processes the script first; it treats the second line as a comment and executes the third line. sh は最初にスクリプトを処理します。 二行目をコメントとして扱い、三行目を実行します。 The exec statement cause the shell to stop processing and instead to start up wish to reprocess the entire script. exec 文はシェルの処理を停止させ、 代わりに wish を起動してスクリプト全体を再処理させます。 When wish starts up, it treats all three lines as comments, since the backslash at the end of the second line causes the third line to be treated as part of the comment on the second line. wish が起動すると、二行目の末尾のバックスラッシュにより 三行目が二行目のコメントの一部として扱われるため、これらの三行は すべてコメントとして扱われます。

The end of a script file may be marked either by the physical end of the medium, or by the character, "\032" ("\u001a", control-Z). スクリプトファイルの終わりは、メディアの物理的な終わり、 または "\032" ("\u001a", control-Z) という文字でマークされることが あります。 If this character is present in the file, the wish application will read text up to but not including the character. この文字がファイルに含まれている場合、 wish アプリケーションはこの文字までのテ

キストを 読み込みますが、この文字自体は含まれません。 An application that requires this character in the file may encode it as "\032", "\x1a", or "\u001a"; or may generate it by use of commands such as format or binary . ファイルにこの文字を必要とするアプリケーションは "\032"、"\x1a"、または "\u001a" としてエンコードするか、 format や binary などのコマンドを 使用して生成することができます。

PROMPTS

When wish is invoked interactively it normally prompts for each command with "%". wish が対話的に呼び出された場合、 通常は各コマンドのプロンプトとして "%" を表示します。 You can change the prompt by setting the variables *tcl_prompt1* and *tcl_prompt2*. プロンプトを変更するには、 変数 *tcl_prompt1* と *tcl_prompt2* を設定します。 If variable *tcl_prompt1* exists then it must consist of a Tcl script to output a prompt; instead of outputting a prompt wish will evaluate the script in *tcl_prompt1*. 変数 *tcl_prompt1* が存在する場合、プロンプトを出力する ための Tcl スクリプトでなければなりません。wish はプロンプトを 出力する代わりに *tcl_prompt1* のスクリプトを評価します。 The variable *tcl_prompt2* is used in a similar way when a newline is typed but the current command is not yet complete; if *tcl_prompt2* is not set then no prompt is output for incomplete commands. 変数 *tcl_prompt2* は、 改行が入力されたが現在のコマンドがまだ完了していない場合に 同様に使用されます。*tcl_prompt2* が設定されていない場合、 不完全なコマンドに対してプロンプトは出力されません。

SEE ALSO

tclsh toplevel Tk_Main Tk_MainLoop Tk_MainWindow