NAME

wafe – A frontend for Xt and the Athena widget applications

SYNOPSIS

wafe [**–c***character*] [**–e** *command*] [**–f** *filename*] [**–p** *program*] [**–D***DBUG-Option*] [**–i**] [**–v**] [**–n**] [**–d**]

DESCRIPTION

Wafe stands for *widget (Athena) front end*. Wafe can be used as an interpretative scripting language or as tool to build graphical interfaces for programs in arbitrary programming languages. The only prerequisite for an application program which want to use Wafe as a frontend is that it must be able to write line buffered to its output (or at least to flush its output buffer).

When Wafe is used as a frontend two separate processes are started – one of them the application, the other responsible for the user interface. The application program (backend) communicates with the frontend (Wafe) via standard using UNIX interprocess communication facilities (streams or sockets). The application process can use the graphical functionalities of *Wafe* by writing the defined commands to its output channel (stdout) and reading the information sent back the front end via its standard input channel (stdin). Since *Wafe* was developed using the extensible *tcl* interpreter language, an application program can dynamically submit requests to the front end to build up the graphical user interface; the application can also down-load application specific procedures into the front end, which can be executed without interaction with the backend application program.

OPTIONS

The following are Wafe's own command line options, they won't be passed to the application process.

–c*character* This option simply allows the user to specify the prompt character, which is not used if *–n* is specified. The default prompt character is "–".

–f *filename* Run Wafe in *file mode*. The argument after *–f* must be the name of a file containing valid Wafe commands. The file will be executed by the tcl source command. Note that in this mode no subprocess is started.

–e *command* Evaluate given command immediately after startup (can be used together with –p)

–p *program* Wafe calls *program* as client process (alternative to link magic below)

–D*DBUG option* This option can be combined with any mode. It passes the specified option to Fred Fish's DBUG library.

–i Switches to *input mode*, which means that Wafe reads from stdin (which it normally doesn't) and forwards the commands to the client-application. Be aware that programs using *input mode* can't be executed in the background! This mode can be combined with both *file mode* and the two process standard mode.

–v Prints the current version number of Wafe and exits immediately.

–n Turn of the prompt character mechanism. This causes Wafe to regard every message it receives from the application as a Wafe command.

–d Enter *direct mode*; this is a special mode, which we use mostly for devolping purposes. In direct mode, you can start Wafe itself without a link, just by typing

```
wafe --d --n
```

Commands are now read from stdin, output is sent to stdout. Additionally, the option *–n* is recommended so that it is unnecessary to prefix every command with the prompt character.

STARTING A WAFE APPLICATION

The way, Wafe communicates with the application process is controlled by various command line arguments. If Wafe is used as a frontend (none of the options *–d*, *–f* or *–v* is specified), an application program is started as a subprocess of Wafe. The name of the application program is obtained either from the *–p myprog* option or from the name of the calling program: If the Wafe application is called *myprog* and a link from Wafe to the name *x*myprog is made (note the leading x), executing *x*myprog starts Wafe with myprog as a subprocess.

Example:

```
ln -s /bin/X11/wafe xmyprog
xmyprog
```

If your local Unix variant does not support symbolic links, a hard link may be used instead of the symbolic link in the example above. Wafe searches all directories in the path for the specified program and spawns the child-process *myprog*.

The application *myprog* looses it's standard input and output streams, it's diagnostic output will be merged with Wafe's stderr. Wafe communicates with the application by reading it's stdout and writing to stdin. Commands to Wafe must be prefixed by a so called prompt character, defaulting to "%", otherwise they are just echoed to Wafe's standard output. The prompt character can be changed with the *–c* command line option. Furthermore, if you want Wafe to accept commands without prompt character, use the *–n* command line option (This means that the application completely looses its output stream).

For the full set of Wafe commands refer to the Wafe documentation, which is included in the distribution.

The application class is derivated from the *–p* option or from the link's name by capitalizing the "x" and the first letter of the executable; any dots will be substituted by dashes. This is the name, under which *X* searches the application's defaults file. So, in our example, the class is *XMyprog* and the corresponding app-defaults file will be searched. The application instance name is the link's name with any dots substituted by dashes.

Command line arguments are passed to the client provided that they do not begin with the characters "–", which is reserved for Wafe's arguments. Thus the command line

```
xmyprog -r filename --n -display friendly.host:0
```

will cause Wafe to spawn the process *myprog* as follows:

```
   myprog -r filename
```

The "–n" option is one of Wafe's command line arguments, which are described above. The arguments "-display friendly.host:0" are processed as standard X toolkit options.

EARLY BEGINNINGS

Save the following lines in a file named "hello":

```
   #!/usr/bin/X11/wafe --f
   command c topLevel label "Wafe new World" callback {echo Goodbye;
quit}
   realize
```

Assuming your system supports the #! notation, you might invoke the little demo program (after setting it executable) by typing "hello". Otherwise you might issue the command "wafe –f hello".

SEE ALSO

Tcl(3) dbug(3) wafeapp(n), X(1), xwafemail(n), xwafenews(n), xwafeftp(n), xwafeora(n), xwafetel(n), xdirtree(n), xdirtree(n), xprojektor(n), xbm(n), xwafemc(n), xruptimes(n), xnetstats(n), xiostats(n), xvmstats(n), xwafeping(n), xwafecf(n)

COPYRIGHT

```
    Wirtschaftsuniversitaet Wien,
    Abteilung fuer Wirtschaftsinformatik
    Augasse 2-6,
    A-1090 Vienna, Austria
    neumann@wu-wien.ac.at, nusser@wu-wien.ac.at
```

AUTHORS

Gustaf Neumann, Stefan Nusser