NAME

multixterm − drive multiple xterms separately or together

SYNOPSIS

multixterm [*args*]

DESCRIPTION

Multixterm creates multiple xterms that can be driven together or separately.

In its simplest form, multixterm is run with no arguments and commands are interactively entered in the first entry field. Press return (or click the "new xterm" button) to create a new xterm running that command.

Keystrokes in the "stdin window" are redirected to all xterms started by multixterm. xterms may be driven separately simply by focusing on them.

The stdin window must have the focus for keystrokes to be sent to the xterms. When it has the focus, the color changes to aquamarine. As characters are entered, the color changes to green for a second. This provides feedback since characters are not echoed in the stdin window.

Typing in the stdin window while holding down the alt or meta keys sends an escape character before the typed characters. This provides support for programs such as emacs.

ARGUMENTS

-xa The optional −xa argument indicates arguments to pass to xterm.

-xc The optional −xc argument indicates a command to be run in each named xterm (see −xn). With no −xc argument, the command is the current shell.

-xd The optional −xd argument indicates a directory to search for files that will appear in the Files menu. By default, the directory is: ˜/lib/multixterm

-xf The optional −xf argument indicates a file to be read at startup. See FILES below for more info.

-xn The optional −xn argument indicates a name for each xterm. This name will also be substituted for any %n in the command argument (see −xc).

-xv The optional −xv flag puts multixterm into a verbose mode where it will describe some of the things it is doing internally. The verbose output is not intended to be understandable to anyone but the author.

Less common options may be changed by the startup file (see FILES below).

All the usual X and wish flags are supported (i.e., −display, −name). There are so many of them that to avoid colliding and make them easy to remember, all the multixterm flags begin with −x.

If any arguments do not match the flags above, the remainder of the command line is made available for user processing. By default, the remainder is used as a list of xterm names in the style of −xn. The default behavior may be changed using the .multixtermrc file (see DOT FILE below).

EXAMPLE COMMAND LINE ARGUMENTS

The following command line starts up two xterms using ssh to the hosts bud and dexter.

```
multixterm −xc "ssh %n" bud dexter
```

FILES

Command files may be used to drive or initialize multixterm. The File menu may be used to invoke other files. If files exist in the command file directory (see −xd above), they will appear in the File menu. Files may also be loaded by using File->Open. Any filename is acceptable but the File->Open browser defaults to files with a .mxt suffix.

Files are written in Tcl and may change any variables or invoke any procedures. The primary variables of interest are 'xtermCmd' which identifies the command (see −xc) and 'xtermNames' which is a list of names (see −xn). The procedure xtermStartAll, starts xterms for each name in the list. Other variables and procedures may be discovered by examining multixterm itself.

EXAMPLE FILE

The following file does the same thing as the earlier example command line:

```
# start two xterms connected to bud and dexter
set xtermCmd "ssh %n"
set xtermNames {bud dexter}
xtermStartAll
```

DOT FILE

At startup, multixterm reads ~/.multixtermrc if present. This is similar to the command files (see FILES above) except that .multixtermrc may not call xtermStartAll. Instead it is called implicitly, similar to the way that it is implicit in the command line use of −xn.

The following example .multixtermrc file makes every xterm run ssh to the hosts named on the command line.

```
set xtermCmd "ssh %n"
```

Then multixterm could be called simply:

```
multixterm bud dexter
```

If any command-line argument does not match a multixterm flag, the remainder of the command line is made available to .multixtermrc in the argv variable. If argv is non-empty when .multixtermrc returns, it is assigned to xtermNames unless xtermNames is non-empty in which case, the content of argv is ignored.

Commands from multixterm are evaluated early in the initialization of multixterm. Anything that must be done late in the initialization (such as adding additional bindings to the user interface) may be done by putting the commands inside a procedure called "initLate".

MENUS

Except as otherwise noted, the menus are self-explanatory. Some of the menus have dashed lines as the first entry. Clicking on the dashed lines will "tear off" the menus.

USAGE SUGGESTION − ALIASES AND COMMAND FILES

Aliases may be used to store lengthy command-line invocations. Command files can be also be used to store such invocations as well as providing a convenient way to share configurations.

Tcl is a general-purpose language. Thus multixterm command files can be extremely flexible, such as loading hostnames from other programs or files that may change from day-to-day. In addition, command files can be used for other purposes. For example, command files may be used to prepared common canned interaction sequences. For example, the command to send the same string to all xterms is:

```
xtermSend "a particularly long string"
```

The File menu (torn-off) makes canned sequences particularly convenient. Interactions could also be bound to a mouse button, keystroke, or added to a menu via the .multixtermrc file.

The following .multixtermrc causes tiny xterms to tile across and down the screen. (You may have to adjust the parameters for your screen.) This can be very helpful when dealing with large numbers of xterms.

```
set yPos 0
set xPos 0

trace variable xtermArgs r traceArgs

proc traceArgs {args} {
   global xPos yPos
   set ::xtermArgs "-geometry 80x12+$xPos+$yPos −font 6x10"
   if {$xPos} {
      set xPos 0
      incr yPos 145
      if {$yPos > 800} {set yPos 0}
   } else {
      set xPos 500
   }
}
```

The xtermArgs variable in the code above is the variable corresponding to the −xa argument.

xterms can be also be created directly. The following command file creates three xterms overlapped horizontally:

```
set xPos 0
foreach name {bud dexter hotdog} {
   set ::xtermArgs "-geometry 80x12+$xPos+0 −font 6x10"
   set ::xtermNames $name
   xtermStartAll
   incr xPos 300
}
```

USAGE SUGGESTION − SELECTING HOSTS BY NICKNAME

The following .multixtermrc shows an example of changing the default handling of the arguments from hostnames to a filename containing hostnames:

```
set xtermNames [exec cat $argv]
```

The following is a variation, retrieving the host names from the yp database:

```
    set xtermNames [exec ypcat $argv]
```

The following hardcodes two sets of hosts, so that you can call multixterm with either "cluster1" or "cluster2":

```
    switch $argv {
  cluster1 {
     set xtermNames "bud dexter"
  }
  cluster2 {
     set xtermNames "frank hotdog weiner"
  }
}
```

COMPARE/CONTRAST

It is worth comparing multixterm to xkibitz. Multixterm connects a separate process to each xterm. xkibitz connects the same process to each xterm.

LIMITATIONS

Multixterm provides no way to remotely control scrollbars, resize, and most other window system related functions.

Because xterm has no mechanism for propagating size information to external processes, particularly for character graphic applications (e.g., vi, emacs), you may have to manually ensure that the spawned process behind each xterm has the correct size. For example, if you create or set the xterm to a size, you may have to send an explicit stty command with the correct size to the spawned process(es). Alternatively, you can add the correct size argument when an xterm is created (i.e., "-geometry 80x20").

Multixterm can only control new xterms that multixterm itself has started.

As a convenience, the File menu shows a limited number of files. To show all the files, use File->Open.

FILES

$DOTDIR/.multixtermrc initial command file
~/.multixtermrc fallback command file
~/lib/multixterm/ default command file directory

BUGS

If multixterm is killed using an uncatchable kill, the xterms are not killed. This appears to be a bug in xterm itself.

Send/expect sequences can be done in multixterm command files. However, due to the richness of the possibilities, to document it properly would take more time than the author has at present.

REQUIREMENTS

Requires Expect 5.36.0 or later.
Requires Tk 8.3.3 or later.

VERSION

This man page describes version 1.8 of multixterm.

The latest version of multixterm is available from http://expect.nist.gov/example/multixterm . If your version of Expect and Tk are too old (see REQUIREMENTS above), download a new version of Expect from http://expect.nist.gov

DATE

April 30, 2002

AUTHOR

Don Libes <don@libes.com>

LICENSE

Multixterm is in the public domain; however the author would appreciate acknowledgement if multixterm or parts of it or ideas from it are used.