NAME

kibitz − allow two people to interact with one shell

SYNOPSIS

kibitz [*kibitz-args*] *user* [*program program-args...*]
kibitz [*kibitz-args*] *user@host* [*program program-args...*]

INTRODUCTION

kibitz allows two (or more) people to interact with one shell (or any arbitrary program). Uses include:

- A novice user can ask an expert user for help. Using **kibitz**, the expert can see what the user is doing, and offer advice or show how to do it right.
- By running **kibitz** and then starting a full-screen editor, people may carry out a conversation, retaining the ability to scroll backwards, save the entire conversation, or even edit it while in progress.
- People can team up on games, document editing, or other cooperative tasks where each person has strengths and weaknesses that complement one another.

USAGE

To start **kibitz**, user1 runs kibitz with the argument of the user to kibitz. For example:

kibitz user2

kibitz starts a new shell (or another program, if given on the command line), while prompting user2 to run **kibitz**. If user2 runs**kibitz** as directed, the k eystrokes of both users become the input of the shell. Similarly, both users receive the output from the shell.

To terminate **kibitz** it suffices to terminate the shell itself. For example, if either user types ^D (and the shell accepts this to be EOF), the shell terminates followed by **kibitz**.

Normally, all characters are passed uninterpreted. However, if the escape character (described when **kibitz** starts) is issued, the user may talk directly to the **kibitz** interpreter. Any **Expect**(1) or **Tcl**(3) commands may be given. Also, job control may be used while in the interpreter, to, for example, suspend or restart **kibitz**.

Various processes can provide various effects. For example, you can emulate a two-way write(1) session with the command:

kibitz user2 sleep 1000000

ARGUMENTS

kibitz takes arguments, these should also be separated by whitespace.

The **−noproc** flag runs **kibitz** with no process underneath. Characters are passed to the other **kibitz**. This is particularly useful for connecting multiple interactive processes together. In this mode, characters are not echoed back to the typist.

−noescape disables the escape character.

−escape *char* sets the escape character. The default escape character is ^].

−silent turns off informational messages describing what kibitz is doing to initiate a connection.

−tty *ttyname* defines the tty to which the invitation should be sent.

If you start **kibitz** to user2 on a remote computer, **kibitz** performs a **rlogin** to the remote computer with your current username. The flag **−proxy** *username* causes **rlogin** to use *username* for the remote login (e.g.

if your account on the remote computer has a different username). If the **-proxy** flag is not given, **kibitz** tries to determine your current username by (in that order) inspecting the environment variables USER and LOGNAME, then by using the commands **whoami** and **logname**.

The arguments **-noescape** and **-escape** can also be given by user2 when prompted to run **kibitz**.

MORE THAN TWO USERS

The current implementation of kibitz explicitly understands only two users, however, it is nonetheless possible to have a three (or more) -way kibitz, by kibitzing another **kibitz**. For example, the following command runs **kibitz** with the current user, user2, and user3:

```
% kibitz user2 kibitz user3
```

Additional users may be added by simply appending more "kibitz user" commands.

The **xkibitz** script is similar to **kibitz** but supports the ability to add additional users (and drop them) dynamically.

CAVEATS

kibitz assumes the 2nd user has the same terminal type and size as the 1st user. If this assumption is incorrect, graphical programs may display oddly.

kibitz handles character graphics, but cannot handle bitmapped graphics. Thus,

```
% xterm -e kibitz    will work
% kibitz xterm       will not work
```

However, you can get the effect of the latter command by using **xkibitz** (see SEE ALSO below). **kibitz** uses the same permissions as used by rlogin, rsh, etc. Thus, you can only **kibitz** to users at hosts for which you can rlogin. Similarly, **kibitz** will prompt for a password on the remote host if rlogin would.

If you **kibitz** to users at remote hosts, **kibitz** needs to distinguish your prompt from other things that may precede it during login. (Ideally, the end of it is preferred but any part should suffice.) If you have an unusual prompt, set the environment variable EXPECT_PROMPT to an egrep(1)-style regular expression. Brackets should be preceded with one backslash in ranges, and three backslashes for literal brackets. The default prompt r.e. is "($|%|#) ".

kibitz requires the **kibitz** program on both hosts. **kibitz** requires **expect**(1).

By comparison, the **xkibitz** script uses the X authorization mechanism for inter-host communication so it does not need to login, recognize your prompt, or require kibitz on the remote host. It does however need permission to access the other X servers.

BUGS

An early version of Sun’s tmpfs had a bug in it that causes **kibitz** to blow up. If **kibitz** reports "error flushing ...: Is a directory" ask Sun for patch #100174.

If your Expect is not compiled with multiple-process support (i.e., you do not have a working select or poll), you will not be able to run kibitz.

ENVIRONMENT

The environment variable SHELL is used to determine the shell to start, if no other program is given on the command line.

If the environment variable EXPECT_PROMPT exists, it is taken as a regular expression which matches the end of your login prompt (but does not otherwise occur while logging in). See also CAVEATS above.

If the environment variables USER or LOGNAME are defined, they are used to determine the current user name for a **kibitz** to a remote computer. See description of the **-proxy** option in ARGUMENTS above.

SEE ALSO

Tcl(3), **libexpect**(3), **xkibitz**(1)
"Exploring Expect: A Tcl-Based Toolkit for Automating Interactive Programs" by Don Libes, O'Reilly and Associates, January 1995.
"Kibitz − Connecting Multiple Interactive Programs Together", by Don Libes, Software − Practice & Experience, John Wiley & Sons, West Sussex, England, Vol. 23, No. 5, May, 1993.

AUTHOR

Don Libes, National Institute of Standards and Technology