NAME

rget, rput − network pipe

SYNOPSIS

rget [**−lcio**] [**−h** *host*] *key* [*command* [*arg* ...]]
rput [**−lcio**] [**−h** *host*] *key* [*command* [*arg* ...]]

DESCRIPTION

Rput and **rget** set up a TCP/IP channel to connect two processes together. They can looked upon as a remote pipe. Consider the well known method of copying a directory tree with **tar**:

    (cd src && tar cf − .) | (cd dst && tar xfp −)

If the directory tree is to be copied to another machine then one can use the following command on the source machine:

    cd src && rput foo tar cf − .

And on the destination machine:

cd dst && rget −h *source-machine* foo tar xfp −

The *key* is either a port number in C style decimal, octal or hex, or a random string that is hashed to a port number. **Rput** uses this port number to open a TCP socket that **rget** using the same *key* can connect to. It is customary to start **rput** first, although **rget** will retry for 2 minutes trying to connect to the remote **rput.**

After the connection is established either utility will execute *command* with the given arguments with the TCP channel as either standard output (rput) or standard input (rget). **Rput** and **rget** do not stay around for the command to finish, they simply overlay themselves with the command. If no command is given then they will themselves copy standard input into the TCP channel (rput), or output from the TCP channel to standard output (rget). So these two commands have the same effect:

    rput foo tar cf − .
    tar cf − . | rput foo

The second form has two processes copying data instead of just **tar** directly writing its output into the TCP channel. There is a better way to waste processor cycles, namely to save bandwidth:

    cd src && tar cf − . | rput foo compress

cd dst && rget −h *source-machine* foo uncompress | tar xfp −

Rput and **rget** can be very useful in the windowed environments we use these days. The **rput** can be typed into the window that has a shell running on one machine, and the **rget** is then typed into the window that has a shell running on another machine. This is easier than one of the two well known forms that use **rsh**:

    cd src && tar cf − . | rsh dest-machine "cd dst && tar xfp −"

    cd dst && rsh source-machine "cd src && tar cf − ." | tar xfp −

Especially since these forms require that one must be able to use **rsh** without a password, which may not always be the case.

The *key* can be any string of characters of any length. If its a number then it is used directly as the port number. Otherwise the characters binary values are multiplied together, bit 15 is set and the result is truncated to 16 bits to make it a port number in the anonymous port space (32768 − 65535). The port may be in-use on the source machine, but there is a small chance of this happening, and if so simply choose another key. (So if you use **rput** and **rget** in an unattended script then you should reserve a port number, otherwise a connection can't be guaranteed.)

OPTIONS

−lcio These flags allow one to reverse the default connect/listen or input/output direction of **rput** and **rget**. Reversing the connection may be necessary if one of the two systems filters out connections to unknown ports. For example:

rput −c −h *destination-machine* foo tar cf − .

rget −l foo tar xfp −

The **−io** options can be used to choose which of standard input or output should be tied to the socket. It's even possible to tie both input and output to the socket with **−io,** but only when executing a command. This is probably the only use for these options, because one usually chooses the direction with the mnemonic put/get names.

−h *host* The name of the remote host that a connection must be made to. It must be used with the program that is doing the connect, usually **rget**. This option is currently mandatory. The author is planning to increase ease of use by letting the programs find each other with UDP broadcasts or multicasts.

SEE ALSO

rsh(1).

DIAGNOSTICS

rput: Address in use
If the port computed out of *key* is already in use.

AUTHOR

Kees J. Bot <kjb@cs.vu.nl>