NAME

rsh, remsh, remote_shell – remote shell

SYNOPSIS

rsh [**–n**] [**–l** *username*] *hostname command*
rsh *hostname* [**–n**] [**–l** *username*] *command*
remsh [**–n**] [**-l** *username*] *hostname command*
remsh *hostname* [**–n**] [**–l** *username*] *command*
hostname [**–n**] [**–l** *username*] *command*

AVAILABILITY

SUNWcsu

DESCRIPTION

rsh connects to the specified *hostname* and executes the specified *command*. **rsh** copies its standard input to the remote command, the standard output of the remote command to its standard output, and the standard error of the remote command to its standard error. Interrupt, quit and terminate signals are propagated to the remote command; **rsh** normally terminates when the remote command does.

If you omit *command*, instead of executing a single command, **rsh** logs you in on the remote host using **rlogin**(1). Shell metacharacters which are not quoted are interpreted on the local machine, while quoted metacharacters are interpreted on the remote machine. See **EXAMPLES**.

Hostnames are given in the *hosts* database, which may be contained in the **/etc/hosts** file, the Internet domain name database, or both. Each host has one official name (the first name in the database entry) and optionally one or more nicknames. Official hostnames or nicknames may be given as *hostname*.

If the name of the file from which **rsh** is executed is anything other than **rsh**, **rsh** takes this name as its *hostname* argument. This allows you to create a symbolic link to **rsh** in the name of a host which, when executed, will invoke a remote shell on that host. By creating a directory and populating it with symbolic links in the names of commonly used hosts, then including the directory in your shell's search path, you can run **rsh** by typing *hostname* to your shell.

If **rsh** is invoked with the basename **remsh**, **rsh** will check for the existence of the file **/usr/bin/remsh**. If this file exists, **rsh** will behave as if **remsh** is an alias for **rsh**. If **/usr/bin/remsh** does not exist, **rsh** will behave as if **remsh** is a host name.

Each remote machine may have a file named **/etc/hosts.equiv** containing a list of trusted hostnames with which it shares usernames. Users with the same username on both the local and remote machine may **rsh** from the machines listed in the remote machine's **/etc/hosts** file. Individual users may set up a similar private equivalence list with the file **.rhosts** in their home directories. Each line in this file contains two names: a *hostname* and a *username* separated by a space. The entry permits the user named *username* who is logged into *hostname* to use **rsh** to access the remote machine as the remote user. If the name of the local host is not found in the **/etc/hosts.equiv** file on the remote machine, and the local username and hostname are not found in the remote user's **.rhosts** file, then the access is denied. The hostnames listed in the **/etc/hosts.equiv** and **.rhosts** files must be the official hostnames listed in the **hosts** database; nicknames may not be used in either of these files.

rsh will not prompt for a password if access is denied on the remote machine unless the *command* argument is omitted.

OPTIONS

–**l** *username* Use *username* as the remote username instead of your local username. In the absence of this option, the remote username is the same as your local username.

–**n** Redirect the input of **rsh** to **/dev/null**. You sometimes need this option to avoid unfortunate interactions between **rsh** and the shell which invokes it. For example, if you are running **rsh** and invoke a **rsh** in the background without redirecting its input away from the terminal, it will block even if no reads are posted by the remote command. The –**n** option will prevent this.

The type of remote shell (**sh**, **rsh**, or other) is determined by the user's entry in the file **/etc/passwd** on the remote system.

EXIT CODES

Returns **0** upon successful completion, **1** otherwise.

EXAMPLES

The following command:

example% rsh lizard cat lizard.file >> example.file

appends the remote file **lizard.file** from the machine called "lizard" to the file called **example.file** on the machine called "example," while the command:

example% rsh lizard cat lizard.file ">>" lizard.file2

appends the file **lizard.file** on the machine called "lizard" to the file **another.lizard.file** which also resides on the machine called "lizard."

FILES

/etc/hosts	Internet host table
/etc/hosts.equiv	trusted remote hosts and users
/etc/passwd	system password file

SEE ALSO

on(1), **rlogin**(1), **telnet**(1), **vi**(1), **in.named**(1M), **hosts**(4), **hosts.equiv**(4)

NOTES

When a system is listed in **hosts.equiv**, its security must be as good as local security. One insecure system listed in **hosts.equiv** can compromise the security of the entire system.

You cannot run an interactive command (such as **vi**(1)); use **rlogin** if you wish to do so.

Stop signals stop the local **rsh** process only; this is arguably wrong, but currently hard to fix for reasons too complicated to explain here.

The current local environment is not passed to the remote shell.

Sometimes the –**n** option is needed for reasons that are less than obvious. For example, the command:

example% rsh somehost dd if=/dev/nrmt0 bs=20b | tar xvpBf –

will put your shell into a strange state. Evidently, what happens is that the **tar** terminates before the **rsh**. The **rsh** then tries to write into the ''broken pipe'' and, instead of terminating neatly, proceeds to compete with your shell for its standard input. Invoking **rsh** with the –**n** option avoids such incidents.

This bug occurs only when **rsh** is at the beginning of a pipeline and is not reading standard input. Do not use the –**n** if **rsh** actually needs to read standard input. For example,

example% tar cf – . | rsh sundial dd of=/dev/rmt0 obs=20b

does not produce the bug. If you were to use the –**n** in a case like this, **rsh** would incorrectly read from **/dev/null** instead of from the pipe.