

Platform **LSF**[®] Cluster Management Tools

Version 6.1

December 2004

Comments to: doc@platform.com

This guide provides reference information for several of the utilities included with the Platform **LSF**[®] software (“LSF”). These utilities run outside of the LSF Batch System and are useful for performing many of the distributed cluster management tasks.

- Contents**
- ◆ “[ch](#)” on page 164
 - ◆ “[lsadmin](#)” on page 173
 - ◆ “[lsgrun](#)” on page 197
 - ◆ “[lslogin](#)” on page 214
 - ◆ “[lsmon](#)” on page 221
 - ◆ “[lsrcp](#)” on page 228
 - ◆ “[lsrun](#)” on page 234
 - ◆ “[lstcsh](#)” on page 237

Copyright © 1994-2004 Platform Computing Corporation

All rights reserved.

We'd like to hear from you You can help us make this document better by telling us what you think of the content, organization, and usefulness of the information. If you find an error, or just want to make a suggestion for improving this document, please address your comments to doc@platform.com.

Your comments should pertain only to Platform documentation. For product support, contact support@platform.com.

Although the information in this document has been carefully reviewed, Platform Computing Corporation ("Platform") does not warrant it to be free of errors or omissions. Platform reserves the right to make corrections, updates, revisions or changes to the information in this document.

UNLESS OTHERWISE EXPRESSLY STATED BY PLATFORM, THE PROGRAM DESCRIBED IN THIS DOCUMENT IS PROVIDED "AS IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL PLATFORM COMPUTING BE LIABLE TO ANYONE FOR SPECIAL, COLLATERAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING WITHOUT LIMITATION ANY LOST PROFITS, DATA, OR SAVINGS, ARISING OUT OF THE USE OF OR INABILITY TO USE THIS PROGRAM.

Document redistribution policy This document is protected by copyright and you may not redistribute or translate it into another language, in part or in whole.

Internal redistribution You may only redistribute this document internally within your organization (for example, on an intranet) provided that you continue to check the Platform Web site for updates and update your version of the documentation. You may not make it available to your organization over the Internet.

Trademarks ® LSF is a registered trademark of Platform Computing Corporation in the United States and in other jurisdictions.

™ ACCELERATING INTELLIGENCE, THE BOTTOM LINE IN DISTRIBUTED COMPUTING, PLATFORM COMPUTING, CLUSTERWARE, PLATFORM ACTIVECLUSTER, IT INTELLIGENCE, SITEASSURE, PLATFORM SYMPHONY, PLATFORM JOBSCHEDULER, PLATFORM INTELLIGENCE, PLATFORM INFRASTRUCTURE INSIGHT, PLATFORM WORKLOAD INSIGHT, and the PLATFORM and LSF logos are trademarks of Platform Computing Corporation in the United States and in other jurisdictions.

UNIX is a registered trademark of The Open Group in the United States and in other jurisdictions.

Microsoft is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.

® Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

Other products or services mentioned in this document are identified by the trademarks or service marks of their respective owners.

ch

changes the host on which subsequent commands are to be executed

SYNOPSIS

```
ch [-s] [-t] [host_name]
```

```
ch [-h | -v]
```

DESCRIPTION

Changes the host on which subsequent commands are to be executed.

By default, if no arguments are specified, changes the current host to the home host, the host from which the `ch` command was issued.

By default, executes commands on the home host.

By default, shell mode support is not enabled.

By default, does not display execution time of tasks.

The `ch` command allows you to quickly change to a designated host with the same execution environment. A simple shell is started that delivers all subsequent commands (except built-in commands) to the designated host for execution.

When the simple shell starts, it is in the current working directory and has the same command execution environment as that of the parent shell. Every remotely dispatched command is executed with the same environment as that on the home host. The syntax of the `ch` command is similar to that of the Bourne shell. However, there are some important differences.

The ampersand (&) following a command line (representing a background job in the Bourne shell) is ignored by `ch`. You can submit background jobs in `ch` with the built-in `post` command and bring them into the foreground with the built-in `contact` command (see below for details).

`ch` recognizes a `~` (tilde) as a special path name. If a `~` (tilde) is followed by a space, tab, new line or `/` (slash) character, then the `~` character is translated into the user's home directory. Otherwise, the `~` is translated as the home directory of the user name given by the string following the `~` character. Pipelines, lists of commands and redirection of standard input/output are all handled by invoking `/bin/sh`.

The following sequence of commands illustrates the behavior of the `ch` command. For example, the user is currently on `hostA`:

```
% ch hostB
hostB> ch hostC
hostC> ch
hostA> ... ..
```

OPTIONS

-s

Starts remote tasks with shell mode support. Shell mode support is required for running interactive shells or applications which redefine the `CTRL-C` and `CTRL-Z` keys (for example, `jove`).

-t

Turns on the timing option. The amount of time each subsequent command takes to execute is displayed.

host_name

Executes subsequent commands on the specified host.

-h

Prints command usage to `stderr` and exits.

-v

Prints LSF release version to `stderr` and exits.

USAGE

The `ch` command interprets the following built-in commands:

cd [*directory_name*]

Changes the current working directory to the specified directory. If a directory is not specified, changes to the user's home directory by default.

ch [*host_name*]

Changes the current working host to the specified host. If a host is not specified, changes to the home host by default.

post [*command* [*argument* ...]]

Posts the specified command for execution in the background on the current working host. `ch` assigns a unique task ID to this command and displays this ID, then continues to interact with the user. However, the output of background jobs may disturb the screen. You can post multiple commands on one host or on different hosts. When a previously posted command is completed, `ch` reports its status to the standard error. If a command is not specified, `ch` displays all currently running background commands.

contact *task_ID*

Brings a previously posted background command into the foreground. *task_ID* is the ID returned by the `post` command. Standard input is now passed to this foreground command. You cannot put an active foreground job into the background. A command that has been brought into the foreground with the `contact` command cannot be put back into the background.

exit

Exits `ch` if there are no posted commands running. Typing an EOF character (usually CTRL-D but may be set otherwise, see `stty(1)`) forces `ch` to exit; uncompleted posted commands are killed.

SEE ALSO

`lshrun(1)`, `rsh(1)`, `stty(1)`

LIMITATIONS

Currently, the `ch` command does not support `script`, `history`, nor `alias`.

The `ch` prompt is always the *current working host:current working directory* followed by a `>` (right angle bracket) character. If the `ch` session is invoked by a shell that supports job control (such as `tcsh` or `ksh`), `CTRL-Z` suspends the whole `ch` session. The exit status of a command line is printed to `stderr` if the status is non-zero.

lsadmin

administrative tool for LSF

SYNOPSIS

```
lsadmin subcommand
```

```
lsadmin [-h | -V]
```

SUBCOMMAND LIST

```
ckconfig [-v]
reconfig [-f] [-v]
limstartup [-f] [host_name ... | all]
limshutdown [-f] [host_name ... | all]
limrestart [-v] [-f] [host_name ... | all]
limlock [-l time_seconds]
limunlock
resstartup [-f] [host_name ... | all]
resshutdown [-f] [host_name ... | all]
resrestart [-f] [host_name ... | all]
reslogon [host_name ... | all] [-c cpu_time]
reslogoff [host_name ... | all]
limdebug [-c "class_name ..."] [-l debug_level] [-f logfile_name] [-o]
    [host_name ..."]
resdebug [-c "class_name" ] [-l debug_level] [-f logfile_name] [-o] [host_name ..."]
limtime [-l timing_level] [-f logfile_name] [-o] [host_name ..."]
restime [-l timing_level] [-f logfile_name] [-o] [host_name ..."]
help [subcommand ...] | ? [subcommand ...]
quit
-h
-v
```

DESCRIPTION

This command can only be used by LSF administrators.

lsadmin is a tool that executes privileged commands to control LIM and RES operations in the LSF cluster.

If no subcommands are supplied for lsadmin, lsadmin prompts for subcommands from the standard input.

For subcommands for which multiple host names or host groups can be specified, do not enclose the multiple names in quotation marks.

Obsolete commands

The command `lslockhost(8)` is superseded by `lsadmin limlock`, and `lsunlockhost(8)` is superseded by `lsadmin limunlock`.

OPTIONS

subcommand

Executes the specified subcommand. See Usage section.

-h

Prints command usage to `stderr` and exits.

-v

Prints LSF release version to `stderr` and exits.

USAGE

ckconfig [-v]

Checks LSF configuration files.

-v

Displays detailed messages about configuration file checking.

reconfig [-f] [-v]

Restarts LIMs on all hosts in the cluster. You should use `reconfig` after changing configuration files. The configuration files are checked before all LIMs in the cluster are restarted. If the configuration files are not correct, reconfiguration will not be initiated.

If `LSF_MASTER_LIST` is specified in `lsf.conf`, you are prompted to confirm the reconfiguration for only the master candidate hosts.

-f

Disables user interaction and forces LIM to restart on all hosts in the cluster if no fatal errors are found. This option is useful in batch mode.

-v

Displays detailed messages about configuration file checking.

limstartup [-f] [*host_name* ... | **all**]

Starts LIM on the local host if no arguments are specified.

Starts LIMs on the specified hosts or on all hosts in the cluster if the word `all` is the only argument provided. You are prompted to confirm LIM startup.

Only `root` and users listed in the parameter `LSF_STARTUP_USERS` in `lsf.sudoers(5)` can use the `all` and `-f` options to start LIM as `root`.

These users must also be able to use `rsh` or `ssh` on all LSF hosts without having to type in passwords. If permission to start up LIMs as `root` is not configured, `limstartup` will start up LIMs as yourself after your confirmation.

The shell command specified by `LSF_RSH` in `lsf.conf` is used before `rsh` is tried.

-f

Disables interaction and does not ask for confirmation for starting LIMs.

limshutdown [-f] [*host_name* ... | **all**]

Shuts down LIM on the local host if no arguments are supplied.

Shuts down LIMs on the specified hosts or on all hosts in the cluster if the word `all` is specified. You are prompted to confirm LIM shutdown.

-f

Disables interaction and does not ask for confirmation for shutting down LIMs.

limrestart [-v] [-f] [*host_name* ... | **all**]

Restarts LIM on the local host if no arguments are supplied.

Restarts LIMs on the specified hosts or on all hosts in the cluster if the word `all` is specified. You are prompted to confirm LIM restart.

`limrestart` should be used with care. Do not make any modifications until all the LIMs have completed the startup process. If you execute `limrestart host_name...` to restart some of the LIMs after changing the configuration files, but other LIMs are still running the old configuration, confusion will arise among these LIMs. To avoid this situation, use `reconfig` instead of `limrestart`.

-v

Displays detailed messages about configuration file checking.

-f

Disables user interaction and forces LIM to restart if no fatal errors are found. This option is useful in batch mode. `limrestart -f all` is the same as `reconfig -f`.

limlock [-l *time_seconds*]

Locks LIM on the local host until it is explicitly unlocked if no time is specified. When a host is locked, LIM's load status becomes `lockU`. No job will be sent to a locked host by LSF.

-l *time_seconds*

The host is locked for the specified time in seconds.

This is useful if a machine is running an exclusive job requiring all the available CPU time and/or memory.

limunlock

Unlocks LIM on the local host.

resstartup [-f] [*host_name* ... | **all**]

Starts RES on the local host if no arguments are specified.

Starts RESs on the specified hosts or on all hosts in the cluster if the word `all` is specified. You are prompted to confirm RES startup.

Only `root` and users defined by the `LSF_STARTUP_USERS` parameter in `lsf.sudoers(5)` can use the `all` and `-f` options to start RES as `root`.

These users must be able to use `rsh` or `ssh` on all LSF hosts without having to type in passwords. For `root` installation to work properly, `lsadmin` must be installed as a `setuid` to `root` program.

The shell command specified by `LSF_RSH` in `lsf.conf` is used before `rsh` is tried.

-f

Disables interaction and does not ask for confirmation for starting RESs.

resshutdown [-f] [*host_name* ... | **all**]

Shuts down RES on the local host if no arguments are specified.

Shuts down RESs on the specified hosts or on all hosts in the cluster if the word **all** is specified. You are prompted to confirm RES shutdown.

If RES is running, it will keep running until all remote tasks exit.

-f

Disables interaction and does not ask for confirmation for shutting down RESs.

resrestart [-f] [*host_name* ... | **all**]

Restarts RES on the local host if no arguments are specified.

Restarts RESs on the specified hosts or on all hosts in the cluster if the word **all** is specified. You are prompted to confirm RES restart.

If RES is running, it will keep running until all remote tasks exit. While waiting for remote tasks to exit, another RES is restarted to serve the new queries.

-f

Disables interaction and does not ask for confirmation for restarting RESs.

reslogon [*host_name* ... | **all**] [-c *cpu_time*]

Logs all tasks executed by RES on the local host if no arguments are specified.

Logs tasks executed by RESs on the specified hosts or on all hosts in the cluster if **all** is specified.

RES will write the task's resource usage information into the log file `lsf.acct.host_name`. The location of the log file is determined by `LSF_RES_ACCTDIR` defined in `lsf.conf`. If `LSF_RES_ACCTDIR` is not defined, or RES cannot access it, the log file will be created in `/tmp` instead.

-c *cpu_time*

Logs only tasks that use more than the specified amount of CPU time. The amount of CPU time is specified by *cpu_time* in milliseconds.

reslogoff [*host_name* ... | **all**]

Turns off RES task logging on the specified hosts or on all hosts in the cluster if **all** is specified.

If no arguments are specified, turns off RES task logging on the local host.

limdebug [-c "*class_name* ..."] [-l *debug_level*] [-f *logfile_name*] [-o "*host_name* ..."]

Sets the message log level for LIM to include additional information in log files. You must be `root` or the LSF administrator to use this command.

If the command is used without any options, the following default values are used:

class_name=0 (no additional classes are logged)

debug_level=0 (LOG_DEBUG level in parameter `LSF_LOG_MASK`)

logfile_name=current LSF system log file in the directory specified by LSF_LOGDIR in the format *daemon_name.log.host_name*

host_name= local host (host from which command was submitted)

In MultiCluster, debug levels can only be set for hosts within the same cluster. For example, you could not set debug or timing levels from a host in `clusterA` for a host in `clusterB`. You need to be on a host in `clusterB` to set up debug or timing levels for `clusterB` hosts.

-c "*class_name* ..."

Specify software classes for which debug messages are to be logged. If a list of classes is specified, they must be enclosed in quotation marks and separated by spaces.

Possible classes:

LC_AFS - Log AFS messages

LC_AUTH - Log authentication messages

LC_CHKPNT - log checkpointing messages

LC_COMM - Log communication messages

LC_DCE - Log messages pertaining to DCE support

LC_EXEC - Log significant steps for job execution

LC_FILE - Log file transfer messages

LC_HANG - Mark where a program might hang

LC_LICENCE - Log license management messages

LC_MULTI - Log messages pertaining to MultiCluster

LC_PIM - Log PIM messages

LC_SIGNAL - Log messages pertaining to signals

LC_TRACE - Log significant program walk steps

LC_XDR - Log everything transferred by XDR

Default: 0 (no additional classes are logged)

Note: Classes are also listed in `lsf.h`.

-l *debug_level*

Specify level of detail in debug messages. The higher the number, the more detail that is logged. Higher levels include all lower levels.

Possible values:

0 - LOG_DEBUG level in parameter LSF_LOG_MASK in `lsf.conf`.

1 - LOG_DEBUG1 level for extended logging. A higher level includes lower logging levels. For example, LOG_DEBUG3 includes LOG_DEBUG2, LOG_DEBUG1, and LOG_DEBUG levels.

2 - LOG_DEBUG2 level for extended logging. A higher level includes lower logging levels. For example, LOG_DEBUG3 includes LOG_DEBUG2, LOG_DEBUG1, and LOG_DEBUG levels.

3 - LOG_DEBUG3 level for extended logging. A higher level includes lower logging levels. For example, LOG_DEBUG3 includes LOG_DEBUG2, LOG_DEBUG1, and LOG_DEBUG levels.

Default: 0 (LOG_DEBUG level in parameter LSF_LOG_MASK)

-f logfile_name

Specify the name of the file into which debugging messages are to be logged. A file name with or without a full path may be specified.

If a file name without a path is specified, the file will be saved in the directory indicated by the parameter LSF_LOGDIR in `lsf.conf`.

The name of the file that will be created will have the following format:

logfile_name.daemon_name.log.host_name

On UNIX, if the specified path is not valid, the log file is created in the `/tmp` directory.

On Windows, no log file is created.

If LSF_LOGDIR is not defined, daemons log to the `syslog` facility.

Default: current LSF system log file in the directory specified by LSF_LOGDIR in the format *daemon_name.log.host_name*.

-o

Turns off temporary debug settings and reset them to the daemon starting state. The message log level is reset back to the value of LSF_LOG_MASK and classes are reset to the value of LSF_DEBUG_RES, LSF_DEBUG_LIM.

Log file is reset back to the default log file.

"host_name ..."

Sets debug settings on the specified host or hosts.

Default: local host (host from which command was submitted)

resdebug [-c "class_name"] [-l debug_level] [-f logfile_name] [-o] ["host_name ..."]

Sets the message log level for RES to include additional information in log files. You must be the LSF administrator to use this command, not root.

See description of `limdebug` for an explanation of options.

limtime [-l timing_level] [-f logfile_name] [-o] ["host_name ..."]

Sets timing level for LIM to include additional timing information in log files. You must be root or the LSF administrator to use this command.

If the command is used without any options, the following default values are used:

timing_level=no timing information is recorded

logfile_name=current LSF system log file in the directory specified by LSF_LOGDIR in the format *daemon_name.log.host_name*

host_name=local host (host from which command was submitted)

In MultiCluster, timing levels can only be set for hosts within the same cluster. For example, you could not set debug or timing levels from a host in `clusterA` for a host in `clusterB`. You need to be on a host in `clusterB` to set up debug or timing levels for `clusterB` hosts.

-l *timing_level*

Specifies detail of timing information that is included in log files. Timing messages indicate the execution time of functions in the software and are logged in milliseconds.

Valid values: 1 | 2 | 3 | 4 | 5

The higher the number, the more functions in the software that are timed and whose execution time is logged. The lower numbers include more common software functions. Higher levels include all lower levels.

Default: undefined (no timing information is logged)

-f *logfile_name*

Specify the name of the file into which timing messages are to be logged. A file name with or without a full path may be specified.

If a file name without a path is specified, the file will be saved in the directory indicated by the parameter `LSF_LOGDIR` in `lsf.conf`.

The name of the file that will be created will have the following format:

logfile_name.daemon_name.log.host_name

On UNIX, if the specified path is not valid, the log file is created in the `/tmp` directory.

On Windows, no log file is created.

If `LSF_LOGDIR` is not defined, daemons log to the syslog facility.

Note: Both timing and debug messages are logged in the same files.

Default: current LSF system log file in the directory specified by `LSF_LOGDIR` in the format *daemon_name.log.host_name*.

-o

Turns off temporary timing settings and resets them to the daemon starting state. The timing level is reset back to the value of the parameter for the corresponding daemon (`LSF_TIME_LIM`, `LSF_TIME_RES`).

Log file is reset back to the default log file.

"host_name ..."

Sets the timing level on the specified host or hosts.

Default: local host (host from which command was submitted)

restime [-l *timing_level*] [-f *logfile_name*] [-o] ["*host_name ...*"]

Sets timing level for RES to include additional timing information in log files. You must be the LSF administrator can use this command, not root.

See description of `limitime` for an explanation of options.

help [*subcommand ...*] | **?** [*subcommand ...*]

Displays the syntax and functionality of the specified commands. The commands must be explicit to lsadmin.

From the command prompt, you may use help or ?.

quit

Exits the lsadmin session.

SEE ALSO

ls_limcontrol(3), ls_rescontrol(3), ls_readconfenv(3),
ls_gethostinfo(3), ls_connect(3), ls_initrex(3), lsf.conf(5),
lsf.sudoers(5), lsf.acct(5), bmggroup(1), busers(1),
lsreconfig(8), lslockhost(8), lsunlockhost(8)

lsgrun

executes a task on a set of hosts

SYNOPSIS

```
lsgrun [-i] [-p | -P | -S] [-v] -f host_file | -m host_name ... | -n num_processors
      [-R "res_req"] [command [argument ...]]
```

```
lsgrun [-h | -V]
```

DESCRIPTION

Executes a task on the specified hosts. `lsgrun` is useful for fast global operations such as starting daemons, replicating files to or from local disks, looking for processes running on all hosts, checking who is logged in on each host, and so on. The hosts can be specified using a host file, a list of host names or by letting the system select the hosts.

DEFAULT BEHAVIOR

By default:

- ◆ `lsgrun` is not interactive.
- ◆ The specified task will be executed sequentially on hosts with full pseudo `tty` support.
- ◆ `lsgrun` does not create a pseudo-terminal.
- ◆ LSF uses as many processors as available to run the specified task.
- ◆ The resource requirement for host selection is `r15s:pg`.
- ◆ The prompt `Command>` is displayed to allow users to type in a command (task) terminated by a `CTRL-D` or `EOF`. The command is then executed on the specified hosts.

OPTIONS

-i

Interactive operation mode. You are asked whether the task will be executed on all hosts. If you answer `y`, the task is started on all specified hosts; otherwise, you are asked to specify hosts interactively.

-p

Parallel run mode. Executes the task on all hosts simultaneously and without pseudo `tty` support.

If this option is specified and the `-P` option is specified, the `-P` option is ignored.

This option is useful for fast start-up of tasks. However, any output from remote tasks will arrive at the terminal in arbitrary order, depending on task execution speeds on individual hosts.

-P

Creates a pseudo-terminal on UNIX hosts. This is necessary to run programs requiring a pseudo-terminal (for example, `vi`).

This option is not supported on Windows.

-S

Creates a pseudo-terminal with shell mode support on UNIX hosts.

Shell mode support is required for running interactive shells or applications which redefine the CTRL-C and CTRL-Z keys (such as `jove`).

This option is not supported on Windows.

-v

Verbose mode. Displays the name of the host or hosts running the task.

-f *host_file*

Either **-f *host_file***, **-m *host_name*** or **-n *num_processors*** is required.

Executes the task on all hosts listed in the *host_file*.

Specify a file that contains a list of host names. Host names must be separated by white space characters (for example, SPACE, TAB, and NEWLINE).

This option is exclusive of options **-n**, **-R**, and **-m**.

-m *host_name ...*

Either **-f *host_file***, **-m *host_name*** or **-n *num_processors*** is required.

Executes the task on all specified hosts.

Specify hosts on which to execute the task. If multiple host names are specified, the host names must be enclosed by " or ' and separated by white space.

This option is exclusive of options **-n**, **-R**, and **-f**.

-n *num_processors*

Either **-f *host_file***, **-m *host_name*** or **-n *num_processors*** is required.

Executes the task on hosts with the required number of available processors.

One host may be used to start several tasks if the host is multiprocessor. This option can be used together with option **-R** to select desired hosts.

This option is exclusive of options **-m** and **-f**.

-R "*res_req*"

Executes the task on hosts with the required resource requirements.

Specify the resource requirement expression for host selection. The resource requirement will be used to choose from all hosts with the same host type as the local host, unless a "type == value" exists in *res_req* to specify otherwise.

This option can be used together with option **-n** to choose a specified number of processors to run the task.

Exclusive resources need to be explicitly specified within the resource requirement string. For example, you defined a resource called `bigmem` in `lsf.shared` and defined it as an exclusive resource for `hostE` in `lsf.cluster.mycluster`. Use the following command submit a task to run on `hostE`:

```
% lsgrun -R "bigmem" myjob
```

or

```
% lsgrun -R "defined(bigmem)" myjob
```

If the `-m` option is specified with a single host name, the `-R` option is ignored.

command [*argument* ...]

Specify the command to execute. This must be the last argument on the command line.

-h

Prints command usage to `stderr` and exits.

-v

Prints LSF release version to `stderr` and exits.

SEE ALSO

`lsfintro(1)`, `lsrun(1)`, `lsplace(1)`

DIAGNOSTICS

Exit status is 0 if all commands are executed correctly.

Otherwise, the exit status is the first non-zero status returned by a remotely executed task. `lsgrun` will execute the task on all hosts even if some have non-zero exit status.

Exit status is -10 if a problem is detected in LSF.

lslogin

remotely logs in to a lightly loaded host

SYNOPSIS

```
lslogin [-v] [-m "host_name ..." | -m "cluster_name ..."] [-R "res_req"]
      [rlogin_options]
```

```
lslogin [-h | -V]
```

DESCRIPTION

Remotely logs in to a lightly loaded host.

By default, `lslogin` selects the least loaded host, with few users logged in, and remotely logs in to that host using the UNIX `rlogin` command.

In a MultiCluster environment, the default is to select the least loaded host in the local cluster.

OPTIONS

-v

Displays the name of the host to which `lslogin` remotely logs you in.

-m "*host_name ...*" | -m "*cluster_name ...*"

Remotely logs in to the specified host.

With MultiCluster job forwarding, when a cluster name is specified, remotely logs in to the least loaded host in the specified cluster, if the remote cluster accepts interactive jobs from the local cluster (see `lsf.cluster(5)`).

-R "*res_req*"

Remotely logs in to a host that meets the specified resource requirement. The resource requirement expression restricts the set of candidate hosts and determines the host selection policy.

For a complete explanation of resource requirement expressions, see `lsfintro(1)`. To find out what resources are configured in your system, use `lsinfo(1)` and `lshosts(1)`.

rlogin_options

Specify remote login options passed to the `rlogin` command.

If remote execution fails, `lslogin` will log in locally only if the local host also satisfies required resources; otherwise, log in will fail.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

EXAMPLE

```
% lslogin -R "select[it>1 && bsd]"
```

Remotely logs in to a host that has been idle for at least 1 minute, that runs BSD UNIX, and is lightly loaded both in CPU resources and the number of users logged in.

SEE ALSO

`lsfintro(1)`, `ls_placereq(3)`, `rlogin(1)`

DIAGNOSTICS

Because `lslogin` passes all unrecognized arguments to `rlogin`, incorrect options usually cause the `rlogin` usage message to be displayed rather than the `lslogin` usage message.

lsmon

displays load information for LSF hosts and periodically updates the display

SYNOPSIS

```
lsmon [-N | -E] [-n num_hosts] [-R res_req] [-I index_list] [-i interval]
      [-L file_name] [host_name ...]
lsmon [-h | -V]
```

DESCRIPTION

lsmon is a full-screen LSF monitoring utility that displays and updates load information for hosts in a cluster.

By default, displays load information for all hosts in the cluster, up to the number of lines that will fit on-screen.

By default, displays raw load indices.

By default, load information is sorted according to CPU and paging load.

By default, load information is updated every 10 seconds.

OPTIONS

-N

Displays normalized CPU run queue length load indices (see `lsfintro(1)`).

-E

Displays effective CPU run queue length load indices (see `lsfintro(1)`). Options `-N` and `-E` are mutually exclusive.

-n *num_hosts*

Displays only load information for the requested number of hosts. Information for up to *num_hosts* hosts that best satisfy resource requirements is displayed.

-R *res_req*

Displays only load information for hosts that satisfy the specified resource requirements. See `lsinfo(1)` for a list of built-in resource names.

Load information for the hosts is sorted according to load on the specified resources.

If *res_req* contains special resource names, only load information for hosts that provide these resources is displayed (see `lshosts(1)` to find out what resources are available on each host).

If one or more host names are specified, only load information for the hosts that satisfy the resource requirements is displayed.

-I *index_list*

Displays only load information for the specified load indices. Load index names must be separated by a colon (for example, `r1m:pg:ut`).

If the index list *index_list* is too long to fit in the screen of the user who invoked the command, the output is truncated. For example, if the invoker's screen is 80 characters wide, then up to 10 load indices are displayed.

- i** *interval*
Sets how often load information is updated on-screen, in seconds.
- L** *file_name*
Saves load information in the specified file while it is displayed on-screen.
If you do not want load information to be displayed on your screen at the same time, use **lsmon -L file_name < /dev/null**. The format of the file is described in `lim.acct(5)`.
- host_name ...*
Displays only load information for the specified hosts.
- h**
Prints command usage to `stderr` and exits.
- v**
Prints LSF release version to `stderr` and exits.

USAGE

You can use the following commands while `lsmon` is running:

[**^L** | **i** | **n** | **N** | **E** | **R** | **q**]

- ^L**
Refreshes the screen.
- i**
Prompts you to input a new update interval.
- n**
Prompts you to input a new number of hosts to display.
- N**
Toggles between displaying raw CPU run queue length load indices and normalized CPU run queue length load indices.
- E**
Toggles between displaying raw CPU run queue length load indices and effective CPU run queue length load indices.
- R**
Prompts you to input new resource requirements.
- q**
Quits `lsmon`.

OUTPUT

The following fields are displayed by default.

HOST_NAME

Name of specified hosts for which load information is displayed, or if resource requirements were specified, name of hosts that satisfied the specified resource requirement and for which load information is displayed.

status

Status of the host. A minus sign (-) may precede the status, indicating that the Remote Execution Server (RES) on the host is not running.

Possible statuses are:

ok

The host is in normal load sharing state and can accept remote jobs.

busy

The host is overloaded because some load indices exceed configured thresholds. Load index values that caused the host to be busy are preceded by an asterisk (*). Built-in load indices include `r15s`, `r1m`, `r15m`, `ut`, `pg`, `io`, `ls`, `it`, `swp`, `mem` and `tmp` (see below). External load indices are configured in the file `lsf.cluster.cluster_name` (see `lsf.cluster(5)`).

lockW

The host is locked by its run window. Run windows for a host are specified in the configuration file (see `lsf.conf(5)`) and can be displayed by `lshosts`. A locked host will not accept load shared jobs from other hosts.

lockU

The host is locked by the LSF administrator or `root`.

unavail

The host is down or the Load Information Manager (LIM) on the host is not running.

unlicensed

The host does not have a valid LSF license.

r15s

The 15-second exponentially averaged CPU run queue length.

r1m

The 1-minute exponentially averaged CPU run queue length.

r15m

The 15-minute exponentially averaged CPU run queue length.

ut

The CPU utilization exponentially averaged over the last minute, between 0 and 1.

pg

The memory paging rate exponentially averaged over the last minute, in pages per second.

ls

The number of current login users.

it

On UNIX, the idle time of the host (keyboard not touched on all logged in sessions), in minutes.

On Windows, the `it` index is based on the time a screen saver has been active on a particular host.

tmp

The amount of free space in `/tmp`, in megabytes.

swp

The amount of currently available swap space, in megabytes.

mem

The amount of currently available memory, in megabytes.

SEE ALSO

`lsfintro(1)`, `lshosts(1)`, `lsinfo(1)`, `lsload(1)`, `lslockhost(8)`,
`lim.acct(5)`, `ls_load(3)`

DIAGNOSTICS

Specifying an incorrect resource requirement string while `lsmon` is running (via the `R` option) causes `lsmon` to exit with an appropriate error message.

`lsmon` exits if it does not receive a reply from LIM within the update interval.

lsrcp

remotely copies files using LSF

SYNOPSIS

```
lsrcp [-a] source_file target_file
```

```
lsrcp [-h | -v]
```

DESCRIPTION

Remotely copies files using LSF.

`lsrcp` is an LSF-enabled remote copy program that transfers a single file between hosts in an LSF cluster. `lsrcp` uses RES on an LSF host to transfer files. If LSF is not installed on a host or if RES is not running then `lsrcp` uses `rcp` to copy the file.

To use `lsrcp`, you must have read access to the file being copied.

Both the source and target file must be owned by the user who issues the command.

`lsrcp` uses `rcp` to copy a source file to a target file owned by another user. See `rcp(1)` and LIMITATIONS below for details.

OPTIONS

-a

Appends *source_file* to *target_file*.

source_file target_file

Specify an existing file on a local or remote host that you want to copy, and a file to which you want to copy the source file.

File format is as follows:

```
[[user_name@][host_name]:][path/]file_name
```

user_name

Login name to be used for accessing files on the remote host. If *user_name* is not specified, the name of the user who issued the command is used.

host_name

Name of the remote host on which the file resides. If *host_name* is not specified, the local host, the host from which the command was issued is used.

path

Absolute path name or a path name relative to the login directory of the user. Shell file name expansion is not supported on either the local or remote hosts. Only single files can be copied from one host to another.

Use "\" to transfer files from a Windows host to another Windows host. For example:

```
c:\share>lsrcp file1 hostA:c:\temp\file2
```

Use "/" to transfer files from a UNIX host to a UNIX host. For example:

```
% lsrcp file1 hostD:/home/usr2/test/file2
```

Always use "/" to transfer files from a UNIX host to a Windows host, or from a Windows host to a UNIX host. This is because the operating system interprets "\" and `lsrcp` will open the wrong files.

For example, to transfer a file from UNIX to a Windows host:

```
% lsrcp file1 hostA:/c:/temp/file2
```

For example, to transfer a file from Windows to a UNIX host:

```
c:\share>lsrcp file1 hostD:/home/usr2/test/file2
```

file_name

Name of source file. File name expansion is not supported.

-h

Prints command usage to `stderr` and exits.

-v

Prints LSF release version to `stderr` and exits.

EXAMPLES

```
% lsrcp myfile @hostC:/home/usr/dir1/otherfile
```

Copies file `myfile` from the local host to file `otherfile` on `hostC`.

```
% lsrcp user1@hostA:/home/myfile user1@hostB:otherfile
```

Copies the file `myfile` from `hostA` to file `otherfile` on `hostB`.

```
% lsrcp -a user1@hostD:/home/myfile /dir1/otherfile
```

Appends the file `myfile` on `hostD` to the file `otherfile` on the local host.

```
% lsrcp /tmp/myfile user1@hostF:~/otherfile
```

Copies the file `myfile` from the local host to file `otherfile` on `hostF` in `user1`'s home directory.

DIAGNOSTICS

`lsrcp` attempts to copy *source_file* to *target_file* using `RES`. If `RES` is down or fails to copy the *source_file*, `lsrcp` will use either `rsh` or the shell command specified by `LSF_RSH` in `lsf.conf` when the `-a` option is specified. When `-a` is not specified, `lsrcp` will use `rcp`.

LIMITATIONS

File transfer using `lsrcp` is not supported in the following contexts:

- ◆ If LSF account mapping is used; `lsrcp` fails when running under a different user account
- ◆ On LSF client hosts. LSF client hosts do not run `RES`, so `lsrcp` cannot contact `RES` on the submission host
- ◆ Third party copies. `lsrcp` does not support third party copies, when neither source nor target file are on the local host. In such a case `rcp` or `rsh` (or the shell command specified by `LSF_RSH` in `lsf.conf`) will be used. If the *target_file* exists, `lsrcp` preserves the modes; otherwise, `lsrcp` uses the *source_file* modes modified with the `umask` (see `umask(2)`) of the source host.

You can do the following:

- ◆ `rcp` on UNIX. If `lsrnp` cannot contact RES on the submission host, it attempts to use `rcp` to copy the file. You must set up the `/etc/hosts.equiv` or `HOME/.rhosts` file in order to use `rcp`. See the `rcp(1)`, `rsh(1)`, `ssh(1)` manual pages for more information on using the `rcp`, `rsh`, and `ssh` commands.
- ◆ You can replace `lsrnp` with your own file transfer mechanism as long as it supports the same syntax as `lsrnp`. This might be done to take advantage of a faster interconnection network, or to overcome limitations with the existing `lsrnp`. `sbatchd` looks for the `lsrnp` executable in the `LSF_BINDIR` directory as specified in `cschrc.lsf`, `profile.lsf`, or `lsf.conf`.

SEE ALSO

`rsh(1)`, `rcp(1)`, `lsfintro(1)`, `res(8)`

lsrun

runs an interactive task through LSF

SYNOPSIS

```
lsrun [-l] [-L] [-P] [-S] [-v] [-m "host_name ..." | -m "cluster_name ..."]
      [-R "res_req"] command [argument ...]
```

```
lsrun [-h | -V]
```

DESCRIPTION

Submits a task to LSF for execution.

With MultiCluster job forwarding model, the default is to run the task on a host in the local cluster.

By default, `lsrun` first tries to obtain resource requirement information from the remote task list to find an eligible host. (See `lselectible(1)` and `ls_task(3)`.) Otherwise, `lsrun` runs the task on a host that is of the same host type (or architecture) as the submission host. If several hosts of the same architecture are available, the host with the lowest CPU and memory load is selected.

By default, if execution fails and the local host satisfies resource requirements, LSF runs the task locally.

By default, `lsrun` does not create a pseudo-terminal when running the task.

OPTIONS

-l

If execution on another host fails, runs the task locally.

-L

Forces `lsrun` to go through RES to execute a task. By default, `lsrun` will not use RES if the task is going to run on the current host.

-P

Creates a pseudo-terminal when starting the task on UNIX hosts. This is necessary in order to run programs that require a pseudo-terminal (for example, `vi`).

This option is not supported on Windows.

-S

Creates a pseudo-terminal with shell mode support when starting the task on a UNIX host. Shell mode support is required for running interactive shells or applications which redefine the CTRL-C and CTRL-Z keys (for example, `jove`).

This option is not supported on Windows.

-v

Displays the name of the host running the task.

-m "*host_name ...*" | **-m** "*cluster_name ...*"

The execution host must be one of the specified hosts. If a single host is specified, all resource requirements are ignored.

If multiple hosts are specified and you do not use the `-R` option, the execution host must satisfy the resource requirements in the remote task list (see `lsrtasks(1)`). If none of the specified hosts satisfy the resource requirements, the task will not run.

With MultiCluster job forwarding model, the execution host can be a host in one of the specified clusters, if the remote cluster accepts tasks from the local cluster. (See `RemoteClusters` section in `lsf.cluster(5)`.)

`-R "res_req"`

Runs the task on a host that meets the specified resource requirement. The size of the resource requirement string is limited to 512 bytes. For a complete explanation of resource requirement expressions, see `lsfintro(1)`. To find out what resources are configured in your system, use `lsinfo(1)` and `lshosts(1)`.

LSF supports ordering of resource requirements on all load indices, including external load indices, either static or dynamic.

Exclusive resources need to be explicitly specified within the resource requirement string. For example, you defined a resource called `bigmem` in `lsf.shared` and defined it as an exclusive resource for `hostE` in `lsf.cluster.mycluster`. Use the following command submit a task to run on `hostE`:

```
% lsrun -R "bigmem" myjob
```

or

```
% lsrun -R "defined(bigmem)" myjob
```

If the `-m` option is specified with a single host name, the `-R` option is ignored.

`-h`

Prints command usage to `stderr` and exits.

`-v`

Prints LSF release version to `stderr` and exits.

USAGE

You can use `lsrun` together with other utility commands such as `lsplace(1)`, `lsload(1)`, `lsloadadj(1)`, and `lselectible(1)` to write load sharing applications in the form of UNIX shell scripts.

`lsrun` supports interactive job control. Suspending `lsrun` suspends both the task and `lsrun`, and continuing `lsrun` continues the task.

The `-n` option of `rsh(1)` can be simulated by redirecting input from `/dev/null`. For example:

```
lsrun cat </dev/null &
```

SEE ALSO

`rsh(1)`, `lsfintro(1)`, `ls_rexecv(3)`, `lsplace(1)`, `lselectible(1)`, `lsload(1)`, `lshosts(1)`, `lsrtasks(1)`, `lsf.cluster(5)`

DIAGNOSTICS

`lsrun` exits with status `-10` and prints an error message to `stderr` if a problem is detected in LSF and the task is not run.

The exit status is -1 and an error message is printed to `stderr` if a system call fails or incorrect arguments are specified.

Otherwise, the exit status is the exit status of the task.

lscsh

load sharing `tcsh` for LSF

SYNOPSIS

```
lscsh [tcsh_options] [-L] [argument ...]
```

DESCRIPTION

`lscsh` is an enhanced version of `tcsh`. `lscsh` behaves exactly like `tcsh`, except that it includes a load sharing capability with transparent remote job execution for LSF. By default, a `lscsh` script is executed as a normal `tcsh` script with load sharing disabled.

If a command line is considered eligible for remote execution, LSF selects a suitable host— typically a powerful and/or lightly loaded host that can execute the command line correctly—and sends the command line to that host.

You can restrict who can use `@` for host redirection in `lscsh` with the parameter `LSF_SHELL_AT_USERS` in `lsf.conf`.

Remote Hosts

`lscsh` provides a high degree of network transparency. Command lines executed on remote hosts behave the same as they do on the local host. The remote execution environment is designed to mirror the local one as closely as possible by using the same values for environment variables, terminal setup, current working directory, file creation mask, and so on. Each modification to the local set of environment variables is automatically reflected on remote hosts.

Shell variables, nice values, and resource limits are not automatically propagated to remote hosts.

Job Control

Job control in `lscsh` is exactly the same as in `tcsh` except for remote background jobs. `lscsh` numbers background jobs separately for each of the hosts that are used to execute them. The output of the built-in command `job` lists background jobs together with their execution hosts.

To bring a remote background job to the foreground, the host name must be specified together with an at sign (`@`), as in the following example:

```
fg %2 @hostA
```

Similarly, the host name must be specified when killing a remote job. For example:

```
kill %2 @hostA
```

OPTIONS

tcsh_options

`lscsh` accepts all the options used by `tcsh`. See `tcsh(1)` for the meaning of specific options.

-L

Executes a script with load sharing enabled.

There are three ways to run a `lstcsh` script with load sharing enabled:

- Execute the script with the `-L` option
- Use the built-in command `source` to execute the script
- Insert `"#!/local/bin/lstcsh -L"` as the first line of the script (assuming you install `lstcsh` in `/local/bin`).

Using `@` or `lsmode` (see below) in a script will not enable load sharing if the script has not been executed using one of these three ways.

USAGE

In addition to the built-in commands in `tcsh`, `lstcsh` provides the following built-in commands:

```
lsmode [on | off] [local | remote] [@] [v | -v] [e | -e] [t | -t] [connect
[host_name ...]] [lsrtasks [lsrtasks_options]] [lsltasks [lsltasks_options]] [jobs]
```

on | **off**

Turns load sharing on or off. When off, you can send a command line to a remote host only if forced eligibility is specified with `@`.

local | **remote**

Sets operation mode of `lstcsh`.

The default is `local`.

local

Local operation mode. This is the default mode.

In this mode, a command line is eligible for remote execution only if all the specified tasks are present in the remote task list in the user's tasks file `$HOME/.lsftask`, or if `@` is specified on the command line to force specified tasks to be eligible for remote execution.

Tasks in the local task list must be executed locally.

The local mode of operation is conservative, and can fail to take advantage of the performance benefits and load balancing advantages of LSF.

The way `lstcsh` handles tasks that are not present in the remote task list nor in the local task list, depends on the mode of operation of `lstcsh` (local or remote).

remote

Remote operation mode.

In this mode, a command line is considered eligible for remote execution only if none of the specified tasks are present in the local task list in the user's tasks file `$HOME/.lsftask`.

Tasks in the remote list can be executed remotely.

The remote mode of operation is aggressive, and promotes extensive use of LSF.

The way `lstcsh` handles tasks that are not present in the remote task list nor in the local task list, depends on the mode of operation of `lstcsh` (local or remote).

@

Specify @ to explicitly specify the eligibility of a command for remote execution.

The @ may be anywhere in the command line except in the first position (which is used to set the value of shell variables).

There are several ways to use @:

@

Specify @ followed by nothing to indicate the command line is eligible for remote execution.

@ *host_name*

Specify @ followed by a host name to force the command line to be executed on that host.

Host names and the reserved word `local` following @ can all be abbreviated as long as they do not cause ambiguity.

@ `local`

Specify @ followed by the reserved word `local` to force the command line to be executed on the local host.

@ */res_req*

Specify @ followed by / and a resource requirement string to indicate the command is eligible for remote execution, and that the specified resource requirements must be used instead of those in the remote task list.

When specifying resource requirements following the @ it is necessary to use / only if the first requirement characters specified are also the first characters of a host name.

e | -e

Turns eligibility verbose mode on (e) or off (-e).

If eligibility verbose mode is on, `lscsh` shows whether the command is eligible for remote execution, and displays the resource requirement used if the command is eligible.

The default is off.

v | -v

Turns task placement verbose mode on (v) or off (-v). If verbose mode is on, `lscsh` displays the name of the host on which the command is run if the command is not run on the local host.

The default is on.

t | -t

Turns wall clock timing on (t) or off (-t).

If timing is on, the actual response time of the command is displayed. This is the total elapsed time in seconds from the time you submit the command to the time the prompt comes back.

This time includes all remote execution overhead. The `csch` time built-in does not include the remote execution overhead.

This is an impartial way of comparing the response time of jobs submitted locally or remotely, because all the load sharing overhead is included in the displayed elapsed time.

The default is off.

connect [*host_name ...*]

Establishes connections with specified remote hosts. If no hosts are specified, lists all the remote hosts to which an `lstcsh` connection has been established.

A plus sign (+) with a remote host indicates that a server-shell has also been started on it.

lsrtasks [+ *task_name[/res_req ...]* | - *task_name[/res_req ...]*]

Displays or update a user's remote task list in the user's task list `$HOME/.lsftask`.

This command has the same function as the external command `lsrtasks`, except that the modified remote task list takes effect immediately for the current `lstcsh` session.

See `lsrtasks(1)` for more details.

lsltasks [+ *task_name ...* | - *task_name ...*]

Displays or update a user's local task list in the user's task list `$HOME/.lsftask`.

This command has the same function as the external command `lsltasks`, except that the modified local task list takes effect immediately for the current `lstcsh` session.

See `lsltasks(1)` for more details.

jobs

Lists background jobs together with the execution hosts. This break of transparency is intentional in order to provide you with more control over your background jobs.

FILES

There are three optional configuration files for `lstcsh`:

`.shrc`

`.hostrc`

`.lsftask`

The `.shrc` and `.hostrc` files are used by `lstcsh` alone, whereas `.lsftask` is used by LSF to determine general task eligibility.

~/shrc

Use this file when you want an execution environment on remote hosts that is different from that on the local host. This file is sourced automatically on a remote host when a connection is established. For example, if the remote host is of different type, you may need to run a version of the executable for that particular host type, therefore it may be necessary to set a different path on the remote host.

~/hostrc

Use this file to indicate a list of host names to which the user wants to be connected (asynchronously in the background) at `lstcsh` startup time. This saves the time spent in establishing the connections dynamically during execution of shell commands. Once a connection is set up, you can execute further remote commands on those connected hosts with very little overhead.

~/lsftask

Use this file to specify lists of remote and local tasks that you want to be added to the respective system default lists. Each line of this file is of the form *task_name/res_req*, where *task_name* is the name of a task, and *res_req* is a string specifying the resource requirements of the task. If *res_req* is not specified, the command is executed on machines of the same type as the local host.

SEE ALSO

`csh(1)`, `tcsh(1)`, `lsrtasks(1)`, `lsltasks(1)`, `lselectible(1)`, `lsinfo(1)`, `lsload(1)`

LIMITATIONS

Type-ahead for the next command is discarded when a job is executing in the foreground on a remote host.

It is not possible to provide input data to load sharing shell scripts (that is, shell scripts whose content is load shared).

The `lstcsh` is fully compatible with `tcsh` 6.03 7-bit mode. Any feature that is not included in `tcsh` 6.03 is not supported.