Solaris Resource Manager 1.3 Reference Manual

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 816–7750–10
DECEMBER 2002

020930@4660

Contents

Preface

Overview

A man page is intended to answer concisely the question “What does it do?” The man pages in general comprise a reference manual. They are not intended to be a tutorial.

The man pages in this reference manual are used with the Solaris Resource Manager product. The entries specify "SRM" after the section numbers, which distinguishes them from other man pages. A few of these man pages have the same base names as existing SunOS man pages. The Solaris Resource Manager entries are addendums that provide product-specific operation information that supplements the base man pages.

The following contains a brief description of each section in the man pages and the information it references:

- Section 1 describes, in alphabetical order, commands available with the operating system.
- Section 1M describes, in alphabetical order, commands that are used chiefly for system maintenance and administration purposes.
- Section 2 describes all of the system calls. Most of these calls have one or more error returns. An error condition is indicated by an otherwise impossible returned value.
- Section 3 describes functions found in various libraries, other than those functions that directly invoke UNIX system primitives, which are described in Section 2 of this volume.
- Section 4 outlines the formats of various files. The C structure declarations for the file formats are given where applicable.
- Section 5 contains miscellaneous documentation such as character set tables.
- Section 6 contains available games and demos.

- Section 7 describes various special files that refer to specific hardware peripherals, and device drivers. STREAMS software drivers, modules and the STREAMS-generic set of system calls are also described.
- Section 9 provides reference information needed to write device drivers in the kernel operating systems environment. It describes two device driver interface specifications: the Device Driver Interface (DDI) and the Driver/Kernel Interface (DKI).
- Section 9E describes the DDI/DKI, DDI-only, and DKI-only entry-point routines a developer may include in a device driver.
- Section 9F describes the kernel functions available for use by device drivers.
- Section 9S describes the data structures used by drivers to share information between the driver and the kernel.

Below is a generic format for man pages. The man pages of each manual section generally follow this order, but include only needed headings. For example, if there are no bugs to report, there is no BUGS section. See the `intro` pages for more information and detail about each section, and `man`(1) for more information about man pages in general.

NAME — This section gives the names of the commands or functions documented, followed by a brief description of what they do.

SYNOPSIS — This section shows the syntax of commands or functions. When a command or file does not exist in the standard path, its full pathname is shown. Options and arguments are alphabetized, with single letter arguments first, and options with arguments next, unless a different argument order is required.

The following special characters are used in this section:

[ ]	The option or argument enclosed in these brackets is optional. If the brackets are omitted, the argument must be specified.
. . .	Ellipses. Several values may be provided for the previous argument, or the previous argument can be specified multiple times, for example, ' `"filename..."`.
|	Separator. Only one of the arguments separated by this character can be specified at time.

	{ } Braces. The options and/or arguments enclosed within braces are interdependent, such that everything enclosed must be treated as a unit.
PROTOCOL	This section occurs only in subsection 3R to indicate the protocol description file.
DESCRIPTION	This section defines the functionality and behavior of the service. Thus it describes concisely what the command does. It does not discuss OPTIONS or cite EXAMPLES.. Interactive commands, subcommands, requests, macros, functions and such, are described under USAGE.
IOCTL	This section appears on pages in Section 7 only. Only the device class which supplies appropriate parameters to the ioctl (2) system call is called `ioctl` and generates its own heading. `ioctl` calls for a specific device are listed alphabetically (on the man page for that specific device). `ioctl` calls are used for a particular class of devices all of which have an `io` ending, such as `mtio`(7I)
OPTIONS	This lists the command options with a concise summary of what each option does. The options are listed literally and in the order they appear in the SYNOPSIS section. Possible arguments to options are discussed under the option, and where appropriate, default values are supplied.
OPERANDS	This section lists the command operands and describes how they affect the actions of the command.
OUTPUT	This section describes the output - standard output, standard error, or output files - generated by the command.
RETURN VALUES	If the man page documents functions that return values, this section lists these values and describes the conditions under which they are returned. If a function can return only constant values, such as 0 or –1, these values are listed in tagged paragraphs. Otherwise, a single paragraph describes the return values of each function. Functions declared void do not return values, so they are not discussed in RETURN VALUES.
ERRORS	On failure, most functions place an error code in the global variable `errno` indicating why they

	failed. This section lists alphabetically all error codes a function can generate and describes the conditions that cause each error. When more than one condition can cause the same error, each condition is described in a separate paragraph under the error code.
USAGE	This section is provided as a guidance on use. This section lists special rules, features and commands that require in-depth explanations. The subsections listed below are used to explain built-in functionality: Commands Modifiers Variables Expressions Input Grammar
EXAMPLES	This section provides examples of usage or of how to use a command or function. Wherever possible a complete example including command line entry and machine response is shown. Whenever an example is given, the prompt is shown as `example%` or if the user must be superuser, `example#`. Examples are followed by explanations, variable substitution rules, or returned values. Most examples illustrate concepts from the SYNOPSIS, DESCRIPTION, OPTIONS and USAGE sections.
ENVIRONMENT VARIABLES	This section lists any environment variables that the command or function affects, followed by a brief description of the effect.
EXIT STATUS	This section lists the values the command returns to the calling program or shell and the conditions that cause these values to be returned. Usually, zero is returned for successful completion and values other than zero for various error conditions.
FILES	This section lists all filenames referred to by the man page, files of interest, and files created or required by commands. Each is followed by a descriptive summary or explanation.
ATTRIBUTES	This section lists characteristics of commands, utilities, and device drivers by defining the attribute type and its corresponding value. See `attributes`(5) for more information.

SEE ALSO	This section lists references to other man pages, in-house documentation and outside publications.
DIAGNOSTICS	This section lists diagnostic messages with a brief explanation of the condition causing the error.
WARNINGS	This section lists warnings about special conditions which could seriously affect your working conditions. This is not a list of diagnostics.
NOTES	This section lists additional information that does not belong anywhere else on the page. It takes the form of an aside to the user, covering points of special interest. Critical information is never covered here.
BUGS	This section describes known bugs and wherever possible, suggests workarounds.

Solaris Resource Manager User Commands

NAME

liminfo – display password attributes and limits information for one or more users

SYNOPSIS

```
liminfo [ -a | -g ] [ -c | -C sec ] [ -l | -u ] [ -nrvx ]
     [ -lname | -uid ... ]
```

DESCRIPTION

With no arguments, `liminfo` outputs a multiline report of the invoker's lnode and password map information. An lnode exists for every *user* under Solaris Resource Manager; it contains a collection of various types of information on the user. `liminfo` is designed for the convenience of users who want to view their current usages, limits, privileges, and so on.

If a single user argument is given, then a report is output for that user. If more than one user argument is given, then several reports are generated, separated by newline characters. Each user argument is interpreted as a login name, unless the name is not found in the password map; in this case, if it is numeric, it is interpreted as a UID.

OPTIONS

The following options are supported:

`-a` Reports on all currently active lnodes. An active lnode is one to which one or more processes are currently attached, or one that has active child limits entries.

`-g` Reports on all currently active limits entries excluding group headers that are active only because they have active members.

`-C` *sec* Continuous output. Outputs all the reports, then a single formfeed character, and then sleeps for the given number of seconds. This entire process is repeated until the command is killed.

`-c` Continuous output. Up-to-date reports are output as they are for the `-C` option until the process is killed.

`-n` Output each report on a single line, with tab characters replacing newlines. Each report is terminated with a newline character.

`-r` Output all values in raw form. Some values, such as memory usage, are normally output in a scaled format, for example, 102 Mbytes. This option causes all such values to be output in basic units such as bytes, seconds, or byte-seconds, for example, 106954752 bytes. Note that this can result in long numbers that may throw off the columnar layout of the formatted output.

`-v` Output in a verbose format. The names and values of all password fields and limits entry attributes are listed.

`-x` Output all values and structure field names of all limits entry attributes in hexadecimal. This form of output is intended for diagnostic support only.

`-l` Interpret all user arguments as login names only. This option is mutually exclusive with `-u`.

`-u` Interpret all user arguments as numeric UIDs only. This option also allows the use of UIDs that are not listed in the password map. This option is mutually exclusive with `-l`.

OUTPUT

This is a brief explanation of the fields in the default report. All password fields are determined by finding the password fields of a UID that match the lnode's UID. Fields that correspond directly to lnode attributes are indicated by [*attribute-name*].

`Login name:`
The login name, from the password map.

`Uid:`
The UID of the lnode.

`Gid:`
The initial GID, from the password map.

`R,Euid: R,Egid:`
The real and effective UIDs and GIDs of the `liminfo` process. These fields only appear when no *user* arguments are given to `liminfo`, in which case the lnode is the one to which the process is attached.

`Sgroup (uid):`
The login name and UID of the scheduling group parent of the lnode [`sgroup`].

`Shares:`
The number of CPU shares with respect to the lnode's scheduling group parent and peers [`cpu.shares`].

`Myshares:`
The number of CPU shares with respect to the lnode's scheduling group children [`cpu.myshares`].

`Share:`
The calculated *allocated share*, which is the lnode's own fraction of the total shares of all currently *active* lnodes, expressed as a percentage. This percentage figure is the CPU share to which this lnode is entitled.

`E-share:`
The calculated *effective share*, which is the lnode's own fraction of the total usage of all currently *active* lnodes, expressed as a percentage. This percentage figure is the CPU share this lnode is *actually* getting.

`Usage:`
The current decayed CPU usage of the lnode [`cpu.usage`].

`Accrued usage:`
The accrued CPU usage (that is, never decayed) of the lnode [`cpu.accrue`].

`Term usage:`
The current decayed terminal connection time of the lnode.

`Term accrued usage:`
The accrued terminal connection time (that is, never decayed) of the lnode.

`Term limit:`
The terminal connection time limit.

`Mem usage:`
The total virtual memory *usage*/*myusage* of all processes currently attached to the lnode [`memory.usage`] [`memory.myusage`].

`Mem limit:`
The hard virtual memory limit [`memory.limit`].

`Proc mem limit:`
The hard per-process virtual memory limit [`memory.plimit`].

`Mem accrue:`
The accrued virtual memory usage [`memory.accrue`].

`Last used:`
The date/time at which the lnode was last active [`lastused`].

`Directory:`
The home directory, from the password map.

`Name:`
The comment, or `gecos` field, from the password map.

`Shell:`
The initial shell, from the password map.

`Processes:`
The number of processes attached to the limits entry/group [`process.usage`] [`process.myusage`].

`Process limit:`
The limit on the number of processes [`process.limit`].

`Current logins:`
The number of login header processes currently attached to the limits entry [`logins`].

`Flags:`
A list of all flags with names of the form `flag.`*name*, that evaluate to `set` [`flag.uselimadm`] [`flag.nologin`] [`flag.onelogin`] [`flag.asynckill`], and others. Only the *name* part of each flag is output, suffixed with one or more of the following symbols:

`+`	The flag evaluates to `set`
`-`	The flag evaluates to `clear`
`*`	The flag evaluates to *group*
`^`	The value is inherited

EXAMPLES

To display the current resource usages and limits:

```
liminfo
```

To have a dynamically updating report of current resource usages and limits:

```
liminfo -c
```

To display all the resource usages and limits of `user1`:

```
liminfo -v user1
```

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWsrmb

SEE ALSO

`limadm`(1MSRM), `srm`(5SRM)

Solaris Resource Manager 1.3 System Administration Guide

NOTES

This man page is applicable to SunOS 5.6, SunOS 5.7, and SunOS 5.8.

NAME

limreport – output information on selected users using printf formats

SYNOPSIS

limreport [-F] *select-expression format* [*identifier*...]

DESCRIPTION

limreport is a simple but powerful report generator. limreport scans sequentially through the password map, evaluating *select-expression* for each user. If the result is non-zero, a non-empty string, or a set flag, then formatted data is written to standard output by interpreting the remaining arguments in the style of printf (see printf(3C)).

The command:

```
limreport 'uid>=100&&cpu.accrue>1.5e10' '%s\t%-.0f\n' lname cpu.accrue
```

lists the login names and accrued CPU usages of all users whose UIDs are 100 or more and whose accrued usage exceeds a given amount.

limreport reads information from the password map and from users' lnodes. If limreport is invoked such that no lnode data is required, then no lnodes are read. This means that even if no lnodes are available, limreport may still be used successfully on password map data only.

OPTIONS

The command-line option is:

-F Use local lnode database file instead of password map. This option is useful when the system has a large list of users in the /etc/passwd file (and in corresponding network name services such as NIS, NIS+ or LDAP). When used, this option forces limreport to use the local lnode database file (/var/srm/srmDB) to get information about created lnodes.

This option has the following limitations:

- The value of the level and orphan attributes will always be 0.
- The value of the sgroupname attribute will always be an empty string.
- It is possible for the CPU and memory accruals to not be up-to-date.

EXPRESSIONS

limreport uses C-style expressions with five special types and a few additional operators. There is a string type, a flag type similar to a C enum type, the integer and long numeric types, and the floating-point type double. Operators have C-like precedence, with the exception that relational operators have lower precedence than bitwise operators.

flag operands are accepted by relational operators only. C-style type conversion is performed on numeric operands of arithmetic and relational operators to raise them to equivalent types.

The elements of an expression are:

* /	Multiply/divide. Numeric operands only.
%	Integer modulo. Integral numeric operands only.
+ -	Add/subtract (or unary negate). Numeric operands only.

`&` `|` `^`	Bitwise and/or/exclusive-or.
`~`	Bitwise complement (single operand), or regular expression matching (two *string* operands). With two operands, evaluates to `1` if the string on the left side contains a match for the regular expression string on the right side. Evaluates to `0` otherwise. Regular expressions are in the style of `ed`(1).
`:`	Is a member of scheduling group. Evaluates to `1` if the user whose numeric UID or string login name on the left side is a member of the scheduling group whose numeric UID or string login name is on the right side. Evaluates to `0` otherwise. (Note that users are not considered to belong to the group of which they are the header.)
`==` `!=`	Equal/not equal to. Evaluates to `1` if the left side and the right side evaluate to identical/different numeric, string, or flag values. Evaluates to `0` otherwise.
`>` `<`	Greater/less than. Evaluates to `1` if the left side evaluates to a numeric (or string) value which is greater/less than the value of the right side. Evaluates to `0` otherwise.
`>=` `<=`	Greater/less than or equal to. Evaluates to `1` if the left side evaluates to a numeric (or string) value which is greater/less than or equal to the value of the right side. Evaluates to `0` otherwise.
`&&` `||`	Logical and/or. Evaluates to `1` if the left side and/or the right side evaluates to a non-zero numeric value, a non-null string value, or a set flag value. Evaluates to `0` otherwise.
`!`	Unary logical not. Evaluates to `1` if the right side evaluates to a zero numeric value, a null string value, or a non-set flag value. Evaluates to `0` otherwise.
`(expr)`	A sub-expression. May be used to override default operator precedence.
`"..."`	A constant string of characters.
`{...}`	A date and time, equivalent to an integer constant giving the number of seconds since an epoch. The format is {[[[[*year*/]*month*/]*day*]*hour*:]*min*[.*sec*]}. *Year* is a four-digit integer (for example, `1989`, `2001`). *Month* may be a one- or two-digit number from 1 to 12, or (deprecated) a three-letter, mixed-case English month name (for example, `Jan`, `sep`, `NOV`). *Day* is a one- or two-digit day-of-the-month from 1 to 28, 29, 30, or 31, depending on the number of days in a given month. *Hour* is a one- or two-digit number from 0 to 23. *Min* is a one- or two-digit number from 0 to 59. *Sec* is a two-digit number from 00 to 59. The separators `/`, `:`, and space are optional, and in their absence the numeric values *month*, *day*, *hour*, and *min* must each be exactly two

digits long. Any missing elements of the date and time format imply that the current value be taken from the system time.

Examples of valid date and time values are {19910101 12:00.01} one second past midday on the first day of the year 1991, {17:00} five o'clock in the afternoon today, {01271200} noon on the 27th of January in the current year.

identifier[?] — Any one of the identifiers described below. Evaluates to either *integer*, *long*, *double*, *string*, or *flag*, depending on its type. Identifiers of 16-bit size are converted to type int32. For flag attributes, the optional ? suffix alters the attribute evaluation method (see below). For non-flag attributes, the ? has no effect.

number — A numeric constant, interpreted as a decimal integer unless prefixed by 0 or 0x in which case it is taken as an octal or hexadecimal integer respectively, or unless it contains a decimal point or exponent part ({eE}_integer), in which case it is taken as a decimal floating-point number, a double. An integer constant has type int32 unless its magnitude is too large to represent in 32 bits, in which case it has type int64. An optional suffix of l or L may be appended to force a type of int64.

A number may optionally be suffixed by one or two *scale* characters: number[eptgmkb][wdhms]. The first character multiplies by the number of bytes in one exabyte, petabyte, terabyte, gigabyte, megabyte, kilobyte, or byte. The second character multiplies by the number of seconds in a week, day, hour, minute, or second. If only one character is present and its meaning is ambiguous (for example, m), then its meaning is deduced from the context. The b (byte) scale character serves no function other than in numbers of the form 100bm, where it forces the m to mean minutes, not megabytes.

Note that the expression will probably have to be enclosed in single quotes to stop the shell from interpreting symbols such as & as shell syntax and to preserve string constant quotation marks.

IDENTIFIERS

An identifier is either the name of an attribute, as defined in the Solaris Resource Manager configuration file, or a built-in name. Most built-in identifiers represent a field extracted from the user's password map entry; however, there are also some useful, miscellaneous ones. See the srm(5SRM) man page for a list of all standard attributes and their types.

There is no numeric constraint on the uses to which an identifier may be put. Identifiers that represent memory usage and limit attributes evaluate to a number of bytes. Identifiers that are dates and times evaluate to a number of seconds. Identifiers representing accrued memory usage attributes evaluate to a number of byte-seconds.

Flag identifiers evaluate to the flag's full value; if a flag has an immediate value of *inherit*, then the value is (recursively) evaluated from the parent lnode. If a flag identifier is suffixed with a `?` character, then the immediate value of the flag is used instead, so the value *inherit* is possible.

The built-in identifiers and their types/values are:

`lname`	string; login name.
`uid`	integer; user UID.
`gid`	integer; initial group GID.
`pword`	string; encrypted password.
`dirpath`	string; initial (home) directory.
`shellpath`	string; initial shell.
`comment`	string; the comment, or `gecos` field.
`gecos`	string; a synonym for `comment`.
`sgroupname`	string; the login name of the user's scheduling group parent.
`now`	integer; the current time.
`level`	integer; the depth of the lnode in the scheduling tree; root is 0. For orphan lnodes, this is the depth within the disjoint tree.
`orphan`	integer; is non-zero if the lnode is an orphan.
`preserve`	string; a list of attribute value assignments using the syntax of `limadm`(1MSRM). Read-only attributes are omitted from the list. The command: `limreport 'flag.real' - lname preserve` will generate output which, if passed to `limadm` using the `-f` option, will completely reconstruct the state of all users' lnodes at the time of execution of the `limreport` command.
`myuid`	integer; the UID of the lnode to which `limreport` is attached.
`mylname`	string; the login name corresponding to `myuid`.
`clear`	flag; a constant.
`set`	flag; a constant.
`group`	flag; a constant.
`inherit`	flag; a constant.

In order to see all identifiers available to you, both built-in and user-defined, invoke `limreport` with the `-?` option or `limadm` with the `-a` option.

Note that the built-in identifiers `sgroupname`, `preserve`, `mylname`, `orphan`, and `level` default to the null string or -1 (for `orphan` and `level`) when the selected lnode does not exist. An error message will also be written to standard error.

FORMATS

The *format* argument is interpreted as the format string of a `printf`(3C) function. All modifiers following the `%` metacharacter are permitted except `*`. If the *format* argument is empty or contains only a dash (`-`), then the default format is used. The default format outputs the arguments in their simplest forms (as unsigned integers or floating-point numbers), separated by tab characters and terminated by a newline.

To facilitate the output of dates, which are represented as number of seconds since midnight on the 1st of January, 1970, the modifier `%t` may be used. This will cause the corresponding numeric *integer* argument to be interpreted as a time and output as a string in the style of `ctime`(3C). Modifiers in `%t` have the same effect as for `%s`.

To facilitate the output of time intervals, which are represented as a number of seconds, the modifier `%i` may be used. This will cause the corresponding numeric *integer* argument to be interpreted as a time interval and output as a string in the form '[Nw] [Nd] [Nh] [Nm] [Ns]'. Modifiers in `%i` have the same effect as for `%s`. If a maximum width is specified (a number following a decimal point), then trailing units are truncated and a + appended if the full length would exceed that many characters.

To facilitate the output of flag values, the modifier `%v` may be used. This will cause the corresponding *flag* argument to be converted to one of the strings `clear`, `set`, `group`, or `inherit`, and output as a string. Modifiers in `%v` have the same effect as for `%s`.

EXAMPLES

EXAMPLE 1 To produce a list of all login names and UIDs except UID zero, one pair per line:

```
limreport 'uid >= 1' '%s,%u\n' lname uid
```

EXAMPLE 2 To produce a report on all users currently logged in:

```
limreport 'flag.real && logins!=0' '%-8.8s %5d %3d %10d\n' lname cpu.myshares process.myusage memor
```

EXAMPLE 3 To list the UIDs of all users with no lnode:

```
limreport \!flag.real - uid
```

Note – `flag.real` can be used in expressions to check only existing lnodes. This will eliminate unwanted messages pertaining to lnodes that do not exist.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE

Architecture	SPARC
Availability	SUNWsrmb

SEE ALSO

`ed`(1), `ctime`(3C), `printf`(3C), `liminfo`(1SRM), `limadm`(1MSRM), `srm`(5SRM)

Solaris Resource Manager 1.3 System Administration Guide

DIAGNOSTICS

All usage, diagnostic, and error messages are written to standard error.

If any error messages are output, `limreport` will return a non-zero exit status, indicating the number of such messages.

Invoking `limreport` with no arguments produces a summary of usage, including a list of all valid identifiers.

Malformed format strings, malformed expressions, and unknown identifiers cause `limreport` to produce informative error messages and exit. If the number of *expression* arguments does not match the number of `%` modifiers in the format string, an error is produced and `limreport` exits.

If, during operation, `limreport` is unable to read a user's lnode, then a diagnostic message is written and operation continues with the next user in the password map.

NOTES

This man page is applicable to SunOS 5.6, SunOS 5.7, and SunOS 5.8.

NAME

rcapstat – report resource cap enforcement daemon statistics

SYNOPSIS

rcapstat [-g] [*interval* [*count*]]

DESCRIPTION

The `rcapstat` command reports on the lnodes and projects capped by `rcapd`(1MSRM). Each report contains current statistics that pertain to the lnode or project, and paging statistics. Paging refers to the act of relocating portions of memory, called pages, to or from physical memory. `rcapd` pages out the most infrequently used pages.

The paging statistics in the first report issued show the activity since the daemon was started. Subsequent reports reflect the activity since the last report was issued.

Reports are issued every *interval* seconds up to the quantity specified by *count*, or forever if *count* is not specified.

OPTIONS

The following command-line option is supported:

`-g` Report the global memory utilization cap set by `rcapadm`(1MSRM), and report current memory utilization as a percentage of all physical memory installed on the system.

OUTPUT

The following list defines the column headings in the `rcapstat` report and provides information about how to interpret the report.

`id` The ID of the capped lnode or project.

`lnode/project` The collection ID type, which is `project` or `lnode`.

`nproc` The number of processes in the collection.

`vm` The total virtual memory size of the collection of processes, including all mapped files and devices, in kilobytes (K), megabytes (M), or gigabytes (G).

`rss` The total resident set size (RSS) of the collection of processes, in kilobytes (K), megabytes (M), or gigabytes (G).

`cap` The RSS cap defined for a project or an lnode. See `rcapd`(1MSRM) for information about how to specify memory caps.

`at` The total amount of memory that `rcapd` attempted to page out since the last `rcapstat` sample.

Paging refers to the act of relocating portions of memory, called pages, to or from physical memory. `rcapd` pages out the most infrequently used pages.

`avgat` The average amount of memory that `rcapd` attempted to page out during each sample cycle that occurred

since the last `rcapstat` sample. The rate at which `rcapd` samples collection RSS can be set with `rcapadm`(1MSRM).

`pg` — An estimate of the total amount of memory that `rcapd` successfully paged out since the last `rcapstat` sample.

`avgpg` — An estimate of the average amount of memory that `rcapd` successfully paged out during each sample cycle that occurred since the last `rcapstat` sample. The rate at which `rcapd` samples process RSS sizes can be set with `rcapadm`.

OPERANDS

interval — Specifies the sampling interval in seconds. The default interval is 5 seconds.

count — Specifies the number of times that the statistics are repeated. By default, `rcapstat` reports statistics until a termination signal is received or until the `rcapd` process exits.

EXAMPLES

EXAMPLE 1 Using `rcapstat` to Report Cap and Lnode Information

Caps are defined for two lnodes associated with two users, `user1` has a cap of 50 megabytes and `user2` has a cap of 10 megabytes.

The following command produces reports at 5-second sampling intervals. A report will be issued five times, once after each sample.

```
# rcapstat 5 5
    id lnode   nproc    vm   rss   cap    at avgat    pg avgpg
112270 user1      24  123M   35M   50M   50M    0K 3312K    0K
 78194 user2       1 2368K 1856K   10M    0K    0K    0K    0K
    id lnode   nproc    vm   rss   cap    at avgat    pg avgpg
112270 user1      24  123M   35M   50M    0K    0K    0K    0K
 78194 user2       1 2368K 1856K   10M    0K    0K    0K    0K
    id lnode   nproc    vm   rss   cap    at avgat    pg avgpg
112270 user1      24  123M   35M   50M    0K    0K    0K    0K
 78194 user2       1 2368K 1928K   10M    0K    0K    0K    0K
    id lnode   nproc    vm   rss   cap    at avgat    pg avgpg
112270 user1      24  123M   35M   50M    0K    0K    0K    0K
 78194 user2       1 2368K 1928K   10M    0K    0K    0K    0K
    id lnode   nproc    vm   rss   cap    at avgat    pg avgpg
112270 user1      24  123M   35M   50M    0K    0K    0K    0K
 78194 user2       1 2368K 1928K   10M    0K    0K    0K    0K
```

The first three lines of output constitute the first report, which contains the cap and lnode information for the two lnodes and paging statistics since `rcapd` was started. The `at` and `pg` columns are a number greater than zero for `user1` and zero for `user2`, which indicates that at some time in the daemon's history, `user1` exceeded its cap but `user2` did not.

The subsequent reports show no significant activity.

EXAMPLE 2 Using `rcapstat` to Monitor the RSS of a Project

```
example% rcapstat 5 5
    id project   nproc    vm   rss   cap    at avgat    pg avgpg
376565 user1        57  209M   46M   10M  440M  220M 5528K 2764K
376565 user1        57  209M   44M   10M  394M  131M 4912K 1637K
376565 user1        56  207M   43M   10M  440M  147M 6048K 2016K
376565 user1        56  207M   42M   10M  522M  174M 4368K 1456K
376565 user1        56  207M   44M   10M  482M  161M 3376K 1125K
```

The project `user1` has an RSS in excess of its physical memory cap. The nonzero values in the `pg` column indicate that `rcapd` is consistently paging out memory as it attempts to meet the cap by lowering the physical memory utilization of the project's processes. However, `rcapd` is unsuccessful, as indicated by the varying `rss` values that do not show a corresponding decrease. This means that the application's resident memory is being actively used, forcing `rcapd` to affect the working set. Under this condition, the system will continue to experience high page fault rates, and associated I/O, until the working set size is reduced, the cap is raised, or the application changes its memory access pattern. Note that a page fault occurs when either a new page must be created, or the system must copy in a page from the swap device.

EXAMPLE 3 Determining the Working Set Size of a Project

This example is a continuation of Example 1, and it uses the same project.

```
example% rcapstat 5 5
    id project    nproc    vm   rss   cap    at avgat    pg avgpg
376565 user1         56  207M   44M   10M  381M  191M   15M 7924K
376565 user1         56  207M   46M   10M  479M  160M 2696K  898K
376565 user1         56  207M   46M   10M  424M  141M 7280K 2426K
376565 user1         56  207M   43M   10M  401M  201M 4808K 2404K
376565 user1         56  207M   43M   10M  456M  152M 4800K 1600K
376565 user1         56  207M   44M   10M  486M  162M 4064K 1354K
376565 user1         56  207M   52M  100M  191M   95M 1944K  972K
376565 user1         56  207M   55M  100M    0K    0K    0K    0K
376565 user1         56  207M   56M  100M    0K    0K    0K    0K
376565 user1         56  207M   56M  100M    0K    0K    0K    0K
376565 user1         56  207M   56M  100M    0K    0K    0K    0K
```

By inhibiting cap enforcement, either by raising the cap of a project or by changing the minimum cap enforcement memory pressure value (see `rcapadm`(1MSRM)), the resident set can become the working set. The `rss` column might stabilize to show the project working set size, as shown in the previous example. This is the minimum cap value that will allow the project's processes to operate without perpetually incurring page faults.

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`1` An error occurred.

2 Invalid command-line options were specified.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWrcapu

SEE ALSO

`rcapadm`(1MSRM), `rcapd`(1MSRM)

Solaris Resource Manager 1.3 System Administration Guide

NOTES

If the interval specified to `rcapstat` is shorter than the reporting interval specified to `rcapd`(with `rcapadm`(1MSRM)), the output for some intervals can be zero. This is because `rcapd` does not update statistics more frequently than the interval specified with `rcapadm`, and this interval is independent of (and less precise than) the sampling interval used by `rcapstat`.

The command described in this man page is only available on SunOS 5.8.

NAME

srmstat – report system usage rate statistics

SYNOPSIS

srmstat [-a | u] [-r] [-k *sec*] [-c | -C *n*] [-s *sec*]

DESCRIPTION

`srmstat` outputs statistics of actual CPU usage rates by active topmost scheduling groups, which should show the share of the system being allocated to each group of users. In particular, it is possible to compare the actual working rate with the intended and effective share.

The `srmstat` table headings include Long-Term Target Usage (LTU) and Short-Term Target Usage (STU). For a number of reasons, there are often gaps between a user's current usage and target usage. (Target usage is the usage Solaris Resource Manager determines that a user is entitled to based on the user's shares and the number of active users.) The STU is the amount Solaris Resource Manager would allocate in the immediate future in order to lessen the gap between a user's current usage and LTU.

Since the statistics indicate actual CPU usage by default, if the *srmidle* user has been defined, any unused CPU capacity is shown as having been used by that user; otherwise it appears in the *System* line.

If `srmstat` is invoked without arguments, it will output the table once, then exit.

OPTIONS

The command-line options are:

`-a` The table will show all users and groups instead of just scheduling groups.

`-c` Continuous mode. Until the process is killed, up-to-date tables are output, each terminated by a form-feed character `\f` and separated by a sleep interval.

`-C` *n* Continuous mode. Until the repeat count, *n*, is reached or the process is killed, up-to-date tables are output, each terminated by a form-feed character `\f` and separated by a sleep interval. The integer *n* causes `srmstat` to exit after *n* tables have been output.

`-k` *sec* Set the *half-life* for decaying the displayed rate to *sec* seconds (the default is 4).

`-r` The statistics indicate rate of accrual of usage costs (scheduling rates) rather than actual CPU usage. Scheduling rate differs from CPU usage because the cost of CPU capacity given to processes running at different `nice` levels is weighted according to the `nice`; and because the cost function may include factors other than CPU usage. Any unused CPU capacity is costed at the current per-tick charge and included in the *System* line; if the `srmidle` user has been defined, this user is included in the display to show any processing attached to that lnode.

`-s` *sec* In continuous mode, sleep for *sec* seconds between tables (the default is 4).

`-u` The table will show only users instead of scheduling groups.

The 'No.' column of the table shows the number of actual users contained in a group.

EXAMPLES To have a dynamically updating report of CPU usage by groups, run:

```
srmstat -c
```

ATTRIBUTES See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWsrmb

SEE ALSO `srm`(5SRM)

Solaris Resource Manager 1.3 System Administration Guide

NOTES This man page is applicable to SunOS 5.6, SunOS 5.7, and SunOS 5.8.

NAME

srmuser – run programs attached to a user's lnode

SYNOPSIS

srmuser [-efps] [-l | -u] *username* [*command* [*args*...]]

DESCRIPTION

`srmuser` attaches to the lnode of the given user, then `exec`s the shell named in the SHELL environment variable, or `/bin/sh` if the environment variable is not defined or is null. If present, the *command* argument with its optional following *args* are passed as a single argument to the shell preceded by a `-c` argument (see `sh`(1)). This effectively causes the command's CPU, memory, and process usages to be accrued to and limited by the specified lnode, and all flag permission checks to be evaluated under that lnode.

If the Solaris Resource Manager software is not enabled, then the lnode attachment is silently omitted, and the shell is forked anyway unless the `-e` option is given. If the Solaris Resource Manager software is enabled, but the lnode attachment fails, then an error message is written and the shell is not forked, unless the `-f` flag is given.

The *username* argument is interpreted as a login name, unless the name is not found in the password map; in this case, if it is numeric, then it is interpreted as a UID.

The main purpose of `srmuser` is to allow processing that would be CPU intensive if run on the root lnode to be run on some other lnode. The shell and command will accrue their usages to the attached lnode and be scheduled according to its share. However, the shell and command retain the process credentials of their invoker, so this continues to affect their ability to create files or to exceed the limits of the attached lnode.

OPTIONS

The command–line options are:

`-e` Fail if Solaris Resource Manager is not enabled. This flag is overridden by `-f`.

`-f` Force `srmuser` to fork a shell even if it cannot attach to the specified lnode. The shell remains attached to the original lnode.

`-p` Preserve lnode attachment even in a case where the invoked command performs a `setuid` operation. This behavior is also inherited by the children of the original process.

`-s` Silent; don't write any error messages.

The mutually exclusive options `-l` and `-u` determine the interpretation of the *username* argument:

`-l` Interpret *username* argument as a login name only.

`-u` Interpret *username* argument as a numeric UID only. This option also allows use of UIDs that are not listed in the password map.

PERMISSIONS

`srmuser` permits attachment to the specified lnode in any of these cases:

- It is currently attached to the root lnode.

- It is already attached to the specified lnode.
- Its real UID is the same as the UID of the specified lnode.
- It is currently attached to an lnode with a set `uselimadm` flag.
- It is currently attached to an lnode with a set `admin` flag (a sub-administrator), and the specified lnode is a member of the attached lnode's scheduling group.

DIAGNOSTICS

The return values and causes are:

0	Success.
1	Incorrect usage; illegal option or missing arguments.
2	Unknown user; cannot find login name or UID in password map.
3	Permission check failed.
4	Cannot attach to lnode; Solaris Resource Manager not enabled.
5	Unable to fork shell; out of memory or processes, or `exec` failed.

Note – The return values 1 through 5 can be ambiguous. If the `srmuser` command itself fails, it will return a value between 1 and 5. If it succeeds, the return value of the last command run in the invoked shell is the code returned. If `srmuser` calls a binary, the return value will be that of the binary. If `srmuser` calls a script with a number of commands, the return value will be the result of the last command executed. Thus, if the return value of the invoked shell is 1 through 5, it will be unclear whether it is the `srmuser` command or the shell that failed.

FILES

/bin/sh	The default shell.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWsrmb

SEE ALSO

`sh`(1), `liminfo`(1SRM), `srm`(5SRM)

Solaris Resource Manager 1.3 System Administration Guide

NOTES

This man page is applicable to SunOS 5.6, SunOS 5.7, and SunOS 5.8.

Solaris Resource Manager Administration Commands

NAME

dispadmin – process scheduler administration

SYNOPSIS

dispadmin -l

dispadmin -c SHR -g [-r *res*]

dispadmin -c SHR -s *file*

ADDENDUM

When Solaris Resource Manager is installed, it includes the Solaris Resource Manager CPU scheduler, which has a class name of "SHR." The `dispadmin` command can be used to identify the presence of the SHR class and to get or set the time slice quantum used by the SHR scheduler.

The SHR scheduler uses the same range of global scheduling priorities as the TS scheduling class (see `ts_dptbl`(4)). However, the priority assignments of processes are dynamically managed by the SHR scheduler to achieve CPU run-rate sharing for users, without the use of a dispatcher table. Consequently, there is no separate loadable module to provide this table.

The remaining parameter, the time slice quantum, may be examined and modified on a running system. The time slice quantum is the length of the time quantum allocated to processes in ticks (Hz). A CPU-bound process will usually run for the duration of the quantum before being preempted in favor of any higher priority process. The default time slice quantum of 11 ticks is a reasonable compromise between interactive responsiveness and system overheads due to process context switching. Larger values could be used on systems where the workload is CPU-bound and non-interactive. Smaller values may improve interactive responsiveness at the cost of increased context switching.

EXAMPLES

The `-l` option displays configured scheduler classes, typically:

```
CONFIGURED CLASSES

SYS                 (System Class)

SHR                 ((SHR) SRM Scheduler)

. . .
```

The following command retrieves the current quantum for the SHR class from kernel memory and writes it to the standard output. The time quantum value is expressed in hundredths of a second.

```
dispadmin -c SHR -g -r 100
```

The output might be:

```
#
 # (SHR) SRM Scheduler Configuration
 #
```

```
 Resolution=100  # Resolution

Quantum=11      # Global time quantum for all processes
```

If this were saved in a file, `shr.config`, for example, the values could be edited and the quantum used by the kernel overwritten with the command:

```
dispadmin -c SHR -s shr.config
```

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWsrmb

SEE ALSO

`priocntl`(1), `ts_dptbl`(4), `priocntl`(1MSRM), `srm`(5SRM)

Solaris Resource Manager 1.3 System Administration Guide

NOTES

This man page is applicable to SunOS 5.6, SunOS 5.7, and SunOS 5.8.

NAME

limadm – change limit attributes or delete lnodes for a list of users

SYNOPSIS

limadm set [-n] [-l | -u] *attribute*[+ | - | =]*value*[:...] *user* ...

limadm -a

limadm delete [-r] [-l | -u] *user* ...

limadm set [-n] [-l | -u] -f *filename*

DESCRIPTION

limadm can either change limit values in the limits database for a given list of users with the set subcommand or delete an entry from the limits database with the delete subcommand. Either subcommand requires that the caller have the appropriate privilege as described in Permissions. The -a option displays the list of changeable limit attributes.

For example, to change a limit attribute, the command:

```
limadm set
sgroup=sysprog:cpu.shares=50:flag.uselimadm=set:memory.limit=5.5m
user1
```

sets the lnode for user1 to have a scheduling group parent of sysprog, 50 shares, full administrative privilege, and a memory limit of 5.5 Mbytes.

limadm applies all the changes, separated by colons, in the given order to each user in turn. Each change is of the form: the name of an *attribute*, followed by an operation character indicating addition, subtraction or assignment, (+|-|=), followed by a *value*, whose type depends on the attribute. Attributes that are readonly cannot be changed.

By default, getpwent(3C) interprets each *user* argument as a login name unless the name is not found in the password map, in which case, if it is numeric, it is interpreted as a UID.

Attributes: Lnode Data Fields

Only the attributes listed in srm(5SRM) can be changed.

The following services.flag attributes are used to enable/disable access to various network and/or local services for users of a Solaris Resource Manager enabled host.

services.flag.su	If clear, user not allowed to execute su command
services.flag.xterm	Does not exist in SRM1.1 or SRM1.2; has no effect in SRM1.0
services.flag.rsh	Enable/disable rsh access to this host
services.flag.ftp	Enable/disable ftp access to this host
services.flag.rlogin	Enable/disable rlogin access to this host
services.flag.rexec	Enable/disable rexec access to this host

`services.flag.localservices`	Enables/disable all local services, such as `flag.su`
`services.flag.netservices`	Enable/disable all network services, such as `rsh, ftp, rlogin, rexec`

The following `terminal.flag` attributes are used to control terminal access to a Solaris Resource Manager enabled host.

`terminal.flag.network`	Allow/disable access from network
`terminal.flag.hardwired`	Allow/disable access from hardwired devices, such as a modem
`terminal.flag.console`	Allow/disable access from the console
`terminal.flag.all`	Set/clear all `terminal.flag` flags

The following flag attributes are used in Solaris Resource Manager to control processes.

`flag.asynckill`	If set, Solaris Resource Manager kills all attached processes when an lnode becomes inactive (for example, when a user logs out). This can be used if users are not allowed to have background processes running after they log out.
`flag.asyncnice`	If set, Solaris Resource Manager lowers the priorities of all processes attached to an lnode when it becomes inactive, improving the runtime of other processes.
`flag.noattach`	If clear, lnode use is allowed. If set, lnode use is denied (processes cannot be attached to this lnode).

The following flag attributes are used to control the number of concurrent logins of a user and/or a scheduling group. (Note that it is also possible to set no restrictions, and permit an unlimited number of logins.)

`flag.nologin=set`	No login permitted
`flag.onelogin=set`	One login permitted

When subtracting from an attribute limit, the result is silently prevented from falling to zero or below. This prevents the value from wrapping around, and also prevents the value from unintentionally becoming zero (a zero limit value has a special meaning: no limit). No numeric limits are applied to addition or assignment.

The following limit attributes are currently not supported:

```
terminal.interval
services.limit
services.accrue
services.usage
services.decay
services.interval
services.lastdecay
```

File Format

The `-f` option causes `limadm` to read a list of users and changes from a named file, or standard input.

Each line of the file starts with a reference to a single *user* (a login name or UID number), followed by a tab, followed by a list of changes identical to what may be entered on the command line. `limadm` reads each line in turn, interprets the *user* name or UID subject to the `-l` and `-u` options, then compiles and performs the list of changes on that *user*.

Permissions

The invoker must have sufficient administrative privilege to perform the given changes to the given users. Two levels of privilege exist:

1. The invoker is the superuser, or has a set `uselimadm` flag. Any attribute in any lnode that the kernel permits to be changed can be changed. Automatically created lnodes are given a default `sgroup` of `srmother`, unless no such login name exists, in which case the default is `root`.
2. The invoker has only a set `admin` flag with the `uselimadm` flag clear. The following restrictions apply:
   - Only the lnodes of users who are members of the invoker's own scheduling group can be altered.
   - Only a member of the invoker's scheduling group can be assigned within the invoker's scheduling group. Automatically created lnodes are given the `sgroup` of the invoker by default.
   - Flag assignment: if the value of the invoker's flag is contrary to the flag's default value, then the invoker may not set the value of that flag to be otherwise, for any user. For example, the invoker's `uselimadm` flag is `clear`, so the invoker may only set other users' `uselimadm` flags to either `clear` or `inherit`.

Flag Assignment

The flag assignment definitions are:

`limadm set flag=set user`	Sets the flag for user. Note that `s` can be used as an abbreviation for `set`.
`limadm set flag=clear user`	Clears the flag for user. Note that `c` can be used as an abbreviation for `clear`.
`limadm set flag=inherit user`	Inherits the flag from the user's parent. Note that `i` can be used as an abbreviation for `inherit`.

`limadm set flag=group user`	Copy flag's value from the user's group flag value. Note that `g` can be used as an abbreviation for `group`.

OPTIONS

The command-line options are:

`set`	Change limit attributes of an lnode. If the lnode does not exist, one will be created automatically.
`-a`	Write a list of all changeable attributes to standard output.
`-n`	Suppress automatic creation of lnodes; write warnings about non-existent lnodes instead.
`-f`	Read from file. The following argument is the name of a file containing a list of users and changes, one per line. If the argument that follows is a dash (`-`), values are read from standard input. See the File Format section.
`delete`	Delete the user's lnode.
`-r`	Force deletion, even if the lnode has children. (The child lnodes will become orphans.)

The mutually exclusive options `-l` and `-u` determine the interpretation of the *user* arguments:

`-l`	Interpret *user* argument as a login name only (the default).
`-u`	Interpret *user* arguments as numeric UIDs only. This option also allows the use of UIDs that are not listed in the password map.

The `-l` and `-u` options also affect the interpretation of values assigned to attributes with the type *uid*.

EXAMPLES

EXAMPLE 1 To assign sub-administrator privileges to a list of users:

```
limadm set -u flag.admin=set UID1 UID2 UID3
```

EXAMPLE 2 To delete a list of user entries from the limits database:

```
limadm delete user1 user2... userN
```

EXAMPLE 3 To display the list of changeable attributes:

```
limadm -a
```

EXAMPLE 4 To disable all network services or limit access for user "John":

First, set all flags (for example, `rsh`) to `group`, so that they will inherit their values from the `services.flag.netservices` flag. (By default, all of these flags are set to `inherit`.) For example, to set `services.flag.ftp`, type:

`limadm set services.flag.ftp=group` *john*

Then, to disable network services for John, type:

`limadm set services.flag.netservices=clear` *john*

To give John access to `ftp` only, type:

`limadm set services.flag.ftp=set` *john*

When a flag is in `set` state, it does not inherit its value from the parent (`limadm set services.flag.netservices` in the above example).

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWsrmb

SEE ALSO

`ctime`(3C), `printf`(3C), `srm`(5SRM)

Solaris Resource Manager 1.3 System Administration Guide

DIAGNOSTICS

`limadm` complains about unknown attribute names and values that overflow their attribute type. On the command line, if a valid attribute name is followed by a missing or malformed operator or value, then an error message is written, accompanied by a brief synopsis of the values that may be assigned to the attribute, and `limadm` exits. If the `-f` flag is in force, then no synopsis is given, and `limadm` skips to the next line of the file.

If an unknown login name or UID number is encountered, `limadm` writes a diagnostic message and skips to the next argument or to the next line in the input file.

WARNINGS

Subtracting from a limit attribute is not an atomic operation and may result in a negative or zero value, if other events are also changing the attribute.

Addition is not checked for overflow, so adding to a numeric attribute may cause the value to wrap around silently.

The ambiguous scale character `m` is assumed to mean ‘megabytes’ (not minutes) if it is the only character following a number with units of storage accrual.

Decimal points can be used with `integer` types. However, numbers containing a decimal point and starting with a zero are interpreted as decimal numbers, not as octal floating-point numbers.

NOTES

This man page is applicable to SunOS 5.6, SunOS 5.7, and SunOS 5.8.

NAME

limdaemon – Solaris Resource Manager notification and connect-time daemon

SYNOPSIS

limdaemon [-ce] [-d] [-D *n*] [-g *n*] [-t *n*] [-w *warn*] [-m *tag*] [-p *fac.lvl*]

limdaemon -k

DESCRIPTION

`limdaemon` is the Solaris Resource Manager user-mode daemon, invoked soon after boot from the script `/etc/init.d/init.srm` (see `init.d`(4)), typically just before `srmadm` is invoked to configure and enable Solaris Resource Manager. It performs the following functions:

- Receives notification messages, translates them into human-readable text, and delivers them to the terminals of the destination users
- Receives any login/logout messages, maintaining an exact record of all login sessions currently in progress
- Periodically updates the connect-time usages in the *terminal* device category for all users with login sessions currently in progress
- Detects users who have reached their connect-time limit and logs them out
- Logs actions through `syslog` and allows messages to be tagged with a priority and prepended with a tag

OPTIONS

`-c`	Suppress the update of terminal connect-time usage. This implies `-e`.
`-d`	Decay connect-time usages for all terminals of logged-in users, using the argument of the `-t` option (default 1 minute) as the interval between decays.
`-D` *n*	Decay connect-time usages for all terminals of all users once every *n* minutes.
`-e`	Suppress enforcement of terminal connect-time limits.
`-g` *n*	Allow a grace time of *n* seconds (default 30) after a user's terminal connect-time limit is reached before forcefully logging them out.
`-k`	Terminate the currently running `limdaemon`. This is done by sending it a SIGTERM signal.
`-m` *tag*	Attach a user-supplied *tag* to every line of text added to the system log.
`-p` *fac.lvl*	Enter the message to the system log with the specified priority. The priority can be specified as a numeric or as a *facility.level* pair. The default priority is *user.notice*. Error messages generated by `limdaemon` itself are logged with priority *daemon.error*.

`-t` *n*	Update the terminal connect-time usage and accrue attributes in all lnodes of logged-in users once every *n* minutes (default 1). Zero is not allowed.
`-w` *warn*	Give users a warning message, *warn* minutes (default 5) prior to an impending logout due to expiration of connect-time.

STARTUP

When `limdaemon` is started, it immediately forks and the parent exits. It is not necessary to follow its invocation with a shell `&` character. The process ID is written in ASCII to the `/var/run/.limdaemon.lock` file for Solaris Resource Manager 1.2 and Solaris Resource Manager 1.3. The process ID is written in ASCII to the `/tmp/.limdaemon.lock` file for Solaris Resource Manager 1.0 and Solaris Resource Manager 1.1. The child process detaches itself from the controlling terminal and places itself in a process group of its own, cutting itself off from keyboard-generated interrupts. When `limdaemon` opens and writes a notification message to `syslog`, it deliberately prevents the device from becoming its controlling terminal.

`limdaemon` writes audit information to `syslog`. The `-p` and `-m` options allow message priorities as well as specified tags to be added to the audit trail.

Message priorities can be specified as a numeric or as a *facility.level* pair. The *facility* identifies the part of the system generating the message, and the *level* indicates the level of priority. The default priority is *user.notice*. Every line added to the log can be prepended with a user-supplied tag.

NOTIFICATION

A notification message contains two UIDs: the user who caused the message, and that of the lnode whose limit was reached. `limdaemon` delivers the message to the terminals of all logged-in members and the group header of the scheduling group headed by the limiting lnode. Messages that are declared with the *useronly* condition are delivered only to the terminals of the user who causes the message.

CONNECT-TIME

Login operations cause `limdaemon` to receive a special `login` message, containing a process ID and the cost of the device on which the `login` was established. The process performing the `login` operation is marked by Solaris Resource Manager as a *login header* process. If that process dies, moves to a different lnode, or initiates another login operation, a matching `logout` message is received.

`limdaemon` maintains in memory an exact record of all current login header processes and their costs, which it uses to update users' connect-time usages. If `limdaemon` terminates for any reason, this information is lost.

`limdaemon` wakes up once every 60 seconds (or as specified by the `-t` option) to accumulate the connect-time usage of each user currently logged in, in proportion to the sum of the costs of all devices to which the user is connected. If the `-c` option is not given, the accumulated usage of each user is added into each lnode.

`limdaemon` also decays connect-time usages for all device categories if the `-d` option is specified.

`limdaemon` also decays connect-time usages for all terminals if the `-d` option or the `-D` option is specified. The `-d` option will cause only logged-in users to have usages decayed. These decays occur whenever `limdaemon` accumulates connect-time usage (specified by the `-t` option). The `-D` option will decay usages for all lnodes, which can be expensive for systems with many users.

Unless the `-e` option is given, when users reach their connect-time limits, `limdaemon` sends them a notification message asking them to log out. After a grace time of 30 seconds, or as specified by the `-g` option, `limdaemon` sends a SIGTERM signal to all processes attached to the user's lnode; then, after a further delay of 15 seconds, a SIGKILL signal is sent.

`limdaemon` is informed of all terminal attribute changes that may affect its internal information whenever it receives a special *lnode change* message, sent by another user-mode program. It is the responsibility of all user programs that change attribute values to send the correct *lnode change* message if an attribute requires one. By convention, all such attributes are declared with the *daemon* condition in the configuration file. Each *lnode change* message contains the UID of the changed lnode. `limdaemon` re-fetches the given lnode and re-reads the values of all attributes that concern it. This means that a single message will serve to notify `limdaemon` of more than one attribute change.

SIGNALS

`limdaemon` traps and ignores most signals, logging them in `syslog`. If `limdaemon` receives a SIGHUP signal, it re-reads the configuration file. If the re-read fails, `limdaemon` continues execution, but will fail to deliver messages and update connect-time usages until receiving another SIGHUP signal and successfully re-reading the configuration file. If `limdaemon` receives a SIGUSR1 signal, it dumps a list of all currently known login header processes to `syslog`. If `limdaemon` receives a SIGTERM signal, it shuts down in an orderly fashion.

FILES

/dev/console	System console device.
/etc/passwd	The password file.
/var/adm/utmp	The user log file.

The `.limdaemon.lock` file is an ASCII string that corresponds to the process ID of the currently running `limdaemon`. It can be created in the `/var/run/.limdaemon.lock` file for Solaris Resource Manager 1.2 and Solaris Resource Manager 1.3, and in the `/tmp/.limdaemon` lock file for Solaris Resource Manager 1.0 and Solaris Resource Manager 1.1.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWsrmb

SEE ALSO `logger`(1), `syslogd`(1M), `kill`(2), `syslog`(3C), `init.d`(4), `limadm`(1MSRM), `srm`(5SRM)

Solaris Resource Manager 1.3 System Administration Guide

DIAGNOSTICS All error messages are written to the system log file or to the system console.

NOTES This man page is applicable to SunOS 5.6, SunOS 5.7, and SunOS 5.8.

NAME

priocntl – Solaris Resource Manager addendum to priocntl

ADDENDUM

The Solaris Resource Manager CPU scheduler (SHR) implements a per-process user priority and user priority limit. The external behavior and range of these parameters is compatible with the TS class. An approximately linear correspondence is maintained between the user priority and `nice` value (see `nice`(1) of processes in the SHR class. If the user priority range is -x to +x, then -x corresponds to maximum `nice` (19), a default user priority of 0 corresponds to default `nice` (0), and +x corresponds to `nice` -20. A change to either user priority or `nice` value causes the corresponding change to the other parameter.

The valid SHR class-specific options for setting time-sharing parameters are:

`-m` *shruprilim*	Set the user priority limit of the specified processes to *shruprilim*.
`-p` *shrupri*	Set the user priority of the specified processes to *shrupri*, and set their `nice` values correspondingly.

The valid range for the parameters can be displayed by using the command:

```
priocntl -l
```

The command:

```
priocntl -d
```

displays the user priority and user priority limit of selected processes.

A process in the SHR class that has a higher `nice` value gets a lower rate of CPU service than SHR class processes with lower `nice` values on the same lnode. Moreover, SHR class processes with higher `nice` values are charged at a lesser rate than SHR class processes with lower `nice` values.

The maximum `nice` value is treated specially. Processes in the SHR class with this value of `nice` are given CPU service only when no other processes are runnable.

Any SHR class process may set its own *shrupri*, or that of any other SHR class process with the same UID to any value less than or equal to the *shruprilim* of the target process. Only a process with superuser privileges may raise the *shruprilim* of a SHR class process.

Once a process belongs to the SHR scheduling class, only the superuser can move the process to another class.

Only the superuser can move processes into the SHR class. Normally the `init` process is started in the SHR scheduling class at system boot, and subsequent processes inherit membership of the SHR class.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWsrmb

SEE ALSO `nice`(1), `priocntl`(1), `srm`(5SRM)

Solaris Resource Manager 1.3 System Administration Guide

NOTES This man page is applicable to SunOS 5.6, SunOS 5.7, and SunOS 5.8.

NAME

rcapadm – configure resource capping daemon

SYNOPSIS

rcapadm

rcapadm [[-n] -E | -D] [-m lnode | project]
[-i *interval=value*, . . . , *interval=value*] [-c *percent*]

DESCRIPTION

`rcapadm` allows the superuser to configure various attributes of the resource capping daemon. If used without arguments, `rcapadm` displays the current status of the resource capping daemon if it has been configured.

OPTIONS

`-n` — Do not affect the running state of the resource capping daemon when enabling or disabling it.

`-E` — Enable the resource capping daemon so that it will be started each time the system is booted. Also start the resource capping daemon now, if the `-n` option is not specified and it is not currently running.

`-D` — Disable the resource capping daemon so that it will not be started when the system is booted. Also stop the resource capping daemon now, if the `-n` option is not specified and it is currently running.

`-m lnode | project` — Choose `rcapd` process collection mode. Two modes of operation are supported: `lnode` and `project`. In `lnode` mode, caps are defined through the lnode limit attribute `rss.limit`. In `project` mode, caps are defined through the project attribute `rcap.max-rss`, which is configured in the `project`(4) database. See `rcapd`(1MSRM)for more information.

`-i` *interval=value,...,interval=value* — Set intervals for various periodic operations performed by `rcapd`. All intervals are specified in seconds. You can set the following intervals:

- `scan` — The rate at which `rcapd` scans for new processes. By default, the scan rate is every 15 seconds. The minimum value is 1 second.
- `sample` — The rate of process resident set size sampling. By default, the

sample rate is every 5 seconds. The minimum value is 1 second.

`report` The rate at which various paging statistics are updated by `rcapd`. These statistics can be viewed by using `rcapstat`(1SRM). By default, the reporting rate is every 5 seconds. When the rate is set to `0`, statistics will not be updated.

Note – Paging refers to the act of relocating portions of memory, called pages, to or from physical memory. `rcapd` pages out the most infrequently used pages.

`config` The reconfiguration rate. At each reconfiguration event, `rcapd` checks its configuration file for updates, and scans lnode (limits) databases or project databases for new lnode or project caps, or new collections. By default, the reconfiguration rate is every 60 seconds. When the rate is set to `0`, no periodic reconfiguration occurs, although the running daemon can still be reconfigured by sending it `SIGHUP`.

`-c` *percent* Set the minimum physical memory utilization for memory cap enforcement. Caps will not be enforced until the physical memory available to processes is low. The *percent* value should be in the range 0 to 100. The minimum (and default) value is `0`, which means that memory caps are always enforced.

EXAMPLES

EXAMPLE 1 Configuring the Resource Capping Daemon in `lnode` Mode With Strict Caps

```
# rcapadm -m lnode -E -i scan=15,sample=5,report=5,config=60
```

EXAMPLE 2 Setting Resident Set Size Cap for the Lnode *oracle*

/usr/srm/sbin/limadm set rss.limit=*10G oracle*

EXAMPLE 3 Setting Resident Set Size Cap for the Project *oracle* in the Project Database

oracle:100::oracle,root::rcap.max-rss=10737418240

EXIT STATUS

The following exit values are returned:

`0` Successful completion. The modifications to the current configuration were valid and made successfully.

`1` An error occurred. A fatal error occurred either in obtaining or modifying the resource capping configuration.

`2` Invalid command-line options were specified.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWrcapu

SEE ALSO

`project`(4), `liminfo`(1SRM), `rcapstat`(1SRM), `limadm`(1MSRM), `rcapd`(1MSRM)

Solaris Resource Manager 1.3 System Administration Guide

NOTES

The command described in this man page is only available on SunOS 5.8.

NAME

rcapd – resource cap enforcement daemon

SYNOPSIS

rcapd [-d]

DESCRIPTION

The rcapd daemon enforces resource caps on collections of processes. Per-lnode or per-project physical memory caps are supported. For information about projects, see project(4).

Collections of processes can be defined as members of particular lnodes or as members of particular projects. When the resident set size (RSS) of a collection of processes exceeds its cap, the daemon takes action and reduces the RSS of the collection.

The virtual memory system divides physical memory into segments known as pages. To read data from a file into memory, the virtual memory system reads in one page at a time, or *pages in* a file. To reduce resource consumption, the daemon can page out, or relocate, infrequently used pages to an area outside of physical memory.

The way in which a cap is specified is determined by rcapd's configured operating mode.

If rcapd is in lnode mode, caps are defined for lnodes that have positive values for the following limit attribute:

rss.limit — The amount of physical memory, in specified units, that is available for the lnode's member processes

Use limadm(1MSRM) to set the rss.limit attribute for lnodes.

If rcapd is in project mode, caps are defined for projects that have positive values for the following attribute:

rcap.max-rss — The total amount of physical memory, in bytes, that is available to the project's member processes

See project(4) for a description of project attributes.

The rcapd daemon can be configured by using rcapadm(1MSRM). The rcapd daemon can be monitored with rcapstat(1SRM). Configuration changes can be incorporated into rcapd by sending it SIGHUP (see kill(1)), or according to the configuration interval (see rcapadm(1MSRM)).

OPTIONS

The following option is supported:

-d — Enable debug mode. Messages are displayed on the user's invoking terminal.

RETURN VALUES

The following exit values are returned:

0 Successful completion.

1 An error occurred.

2 Invalid command-line options were specified.

FILES

/etc/project Project database.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWrcapu (32-bit)
	SUNWrcapx (64-bit)

SEE ALSO

`project`(4), `liminfo`(1SRM), `rcapstat`(1SRM), `limadm`(1MSRM), `rcapadm`(1MSRM)

Solaris Resource Manager 1.3 System Administration Guide

NOTES

If killed, `rcapd` can leave processes in a stopped state. Use `SIGTERM` to cause `rcapd` to terminate properly.

A collection's RSS can exceed its cap for some time before the cap is enforced, even if sufficient pageable memory is available. This period of time can be reduced by decreasing the RSS sampling interval with `rcapadm`.

Only one instance of `rcapd` can run at any given time.

The command described in this man page is only available on SunOS 5.8.

NAME

srmadm – Solaris Resource Manager administration

SYNOPSIS

```
srmadm

srmadm show [-d | i] mode

srmadm show [-v | -V level] [-d | -i]

srmadm set [-v | -V level] -d

srmadm set [-v | -V level] [-d] [-f filename] mode[={y | n}] [:…]

srmadm set [-v | -V level] [-d] parameter[=value] [:…]
```

DESCRIPTION

The `srmadm` command displays or sets operation modes and system-wide tunable parameters of Solaris Resource Manager.

Invoked without options, `srmadm` reports whether Solaris Resource Manager is installed on the system.

OPTIONS

`show [-d | -i] mode`

Displays the state of the specified operation mode of Solaris Resource Manager. `srmadm` displays `yes` on the standard output and returns 0 if the specified mode is enabled, otherwise it displays `no` and returns 1.

The following operating modes are recognized:

`fileopen`	Solaris Resource Manager currently has a limits database open.
`share`	Solaris Resource Manager SHR scheduler is currently enabled.
`limits`	Solaris Resource Manager virtual memory and process limiting functions are currently enabled.
`adjgroups`	The hierarchical group balancing feature of the Solaris Resource Manager SHR scheduler is currently enabled.
`limshare`	The priority ceiling feature of the Solaris Resource Manager SHR scheduler is currently enabled.

When used with the `-d` option, the default value of the mode will be displayed.

When used with the `-i` option, the initial value of the mode will be displayed. The initial value is the one preset on reboot before Solaris Resource Manager is brought online.

`show [-d | i]`

Outputs a formatted report of the current operational state of Solaris Resource Manager and its main configuration parameters.

When used with the `-d` option, the default values of the modes and parameters will be displayed.

When used with the `-i` option, the default initial values of the modes and parameters will be displayed. The initial values are those preset on reboot before Solaris Resource Manager is brought online.

`set -d`

`set [-d] [-f` *filename*`] mode[={y | n}] [: ...]`

`set [-d] parameter[=`*value*`][: ... ]`

Enables or disables the specified operation mode(s) or assigns a value to the specified Solaris Resource Manager parameter.

When used with the `-d` option and *without* modes or parameters, *all* parameters and modes will be set to their default values.

When used with the `-d` option and *with* modes or parameters, *all* parameters and modes will be set to the default values except for the specified modes or parameters which will be set to the specified values.

When enabling the `fileopen` mode by setting it to "y," an optional *filename* for the limits database can be provided to Solaris Resource Manager using the `-f` option. If no *filename* is provided, the default limits database will be used. The default file name is `/var/srm/srmDB`. An empty database can be created by using the `touch` command.

Using the `set` option on a parameter or mode and without any value will set the specified parameter or mode to the default value.

The setting of both modes and parameters can be combined in a single invocation of the command.

Note that the invoker must be the superuser to change values.

`-v`

`-V [ level ]`

When using with the `show` option and no `mode` argument, you may also use the `-v` or `-V` option to specify a verbosity level to be used for outputting the current settings of Solaris Resource Manager.

The highest level of verbosity is 3. If a level higher than 1 is supplied, then additional statistics and settings of the Solaris Resource Manager system are output. Invoking `srmadm` with no `-v` is equivalent to supplying a `-v` or `-V` 1. When used with the `show` option and no mode argument is supplied, `-v` is equivalent to `-V` 1.

When using with the `set` option, the `-v` or `-V` options cause the operational state on completion of the command to be displayed (at the requested level of verbosity), provided the command is successful.

OPERATION MODES

`fileopen[={y | n}]`

The default database is `/var/srm/srmDB` and it can be overridden with the `-f` option.

Note that closing the Solaris Resource Manager limits database file in mid–operation should be regarded as an emergency action. It has several undesirable consequences: all processes will continue running on the surrogate root lnode, which may give them more privilege than normal; the SHR scheduler is disabled; Solaris Resource Manager limit enforcement ceases.

When disabled (`fileopen=n`), the Solaris Resource Manager currently has no limits database open, and its cache contains only the surrogate root lnode to which all processes are attached.

`share=[{y | n}]`
When enabled (`share=y`), the SHR scheduler is used and CPU scheduling takes place according to Solaris Resource Manager's dynamic usage and decay algorithm. This mode cannot be set unless the `fileopen` mode is enabled. When disabled, the SHR scheduler's usage calculations are frozen, and processes are scheduled round-robin with fixed equal priorities.

`limits=[{y | n}]`
When enabled (`limits=y`), Solaris Resource Manager enforces the virtual memory and process limits. This mode cannot be set unless the `fileopen` mode is enabled. When disabled, Solaris Resource Manager will keep usage attributes up to date, but will not enforce limits.

`adjgroups=[{y | n}]`
When enabled (`adjgroups=y`), the SHR scheduler's global group effective share adjustment is used. The enabled state is recommended in most circumstances.

Every run interval, the normalized usages of all limits entries are recalculated. If the `adjgroups` scheduling mode is enabled, then extra processing of normalized usages is performed as follows. The scheduler makes a pass over the scheduling tree, comparing each group's recently received effective share with its entitlement. Groups that have received less than their group entitlement are biased to receive a greater effective share in the next run interval. This has the effect of ensuring that groups receive their entitlement of CPU service whenever possible, regardless of the actions of their members.

`limshare=[{y | n}]`
When enabled (`limshare=y`), the SHR scheduler applies its priority ceiling feature to limit all users' effective shares to prevent extremely low-usage users from briefly acquiring almost 100 percent of CPU. The enabled state is recommended.

The rate of CPU service for a user is roughly inversely proportional to the user's usage. If users have not been active for a very long time, then their usage decays to near-zero. When such a user logs in (or the lnode becomes active in any way), then for the duration of the next run interval, the user's processes could have such high priority that they monopolize the CPU.

Enabling the `limshare` scheduling flag causes the scheduler to estimate the effective share that an lnode will receive before the next run interval. If the result exceeds the user's assigned entitlement by a given factor (see `maxushare`), then the user's normalized usage is readjusted to prevent this.

PARAMETERS

`idle[={uid | lname}]`
Causes `srmadm` to nominate the Solaris Resource Manager *idle lnode* with the UID of the given user as the argument. The default user name is `srmidle` with UID 41.

All Solaris Resource Manager idle CPU costs are charged to this lnode. The `srmidle` lnode must have zero shares to ensure that the processes attached to it are run only when no other processes can be run.

`lost[={uid | lname}]`
Causes `srmadm` to nominate the Solaris Resource Manager *lost lnode* as that belonging to the user with the given UID or login name. It is not allowable for the *lost lnode* to be set to root's lnode, and using a login name of *root* as the value for this parameter is an error. Using a UID of *0* as the value for this parameter results in Solaris Resource Manager having no *lost lnode*. The default user name is `srmlost` with UID 42.

Using the default value substitution feature with this parameter when the user `srmlost` does not exist results in Solaris Resource Manager having no *lost lnode*.

The lost lnode is used by the `setuid`(2) system call. Under Solaris Resource Manager, `setuid` has the additional function of attaching the calling process to a new lnode. If this fails, or there is no lost lnode, then the process remains attached to its original lnode.

`delta[=seconds]`
The run interval for the SHR scheduler. This is the time interval that elapses between recalculations of the normalized usages of all active users. The normalized usage affects the priorities of a user's processes, so larger values of delta effectively reduce the short-term responsiveness of the scheduler.

`maxusage[=float]`
The upper bound for CPU usages used in the priority calculation. Users with usages larger than this will use this value for their priority calculation. This prevents users with high CPU usages from skewing the priorities of other users.

`usagedecay[={seconds | hours{h}}]`
The decay rate for users' usages, expressed as a half-life in seconds. The optional suffix character `h` may be used to specify the value in hours.

`pridecay[=seconds:seconds]`
The decay rate for the priorities of processes with normal and maximum `nice` values respectively, expressed as half-lives. The rates for other `nice` values are interpolated between these two and extrapolated down to minimum `nice`. The second value must be greater than the first.

`maxushare[=float]`
If the `limshare` scheduling mode is enabled, this limits the maximum effective share an individual user can have to float times their allocated share. It must not be less than 1.0, and the default is 2.0.

EXAMPLES

To check whether Solaris Resource Manager is installed:

srmadm

To check if a limits database is opened:

srmadm show fileopen

To display the default setting for `fileopen`:

srmadm show -d fileopen

To enable Solaris Resource Manager, which requires opening a limits database, enabling the SHR scheduler, and enforcing the source limits:

srmadm set -f /var/srm/myDB share=y:limits=y:fileopen=y

A number of `srmadm` commands, including this one, are performed by the Solaris Resource Manager `/etc/rc2.d/S10srm` script during the transition to system `init` level 2.

To display the current flag settings and charges:

srmadm show

To display the default flag settings and charges:

srmadm show -d

To set all parameters and modes to the default values:

srmadm set -d

To set the usage decay rate to have a half-life of 5 minutes:

srmadm set usagedecay=300

To set all parameters and modes to the default values except the usage decay rate, which is set to 5 minutes:

srmadm set -d usagedecay=300

To set the scheduler run rate and usage decay rate to their default values:

srmadm set delta:usagedecay

To show all the default settings:

srmadm show -d -V 3

FILES

/etc/init.d/init.srm	The Solaris Resource Manager `init` script.
/etc/rc1.d/K52srm	Link to the Solaris Resource Manager `init` script used at entry to system `init` level 1.

/etc/rc2.d/S10srm	Link to the Solaris Resource Manager `init` script used at entry to system `init` level 2.
/var/srm/srmDB	The Solaris Resource Manager limits database.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWsrmb

SEE ALSO

`touch`(1), `setuid`(2), `limadm`(1MSRM), `srm`(5SRM)

Solaris Resource Manager 1.3 System Administration Guide

NOTES

This man page is applicable to SunOS 5.6, SunOS 5.7, and SunOS 5.8.

NAME

srmkill – send a signal to processes attached to a limits database entry

SYNOPSIS

srmkill [-s *signame* | *-signum*] [-l | -u] *user...*

DESCRIPTION

The srmkill command sends a signal to all processes that are currently attached to the lnode specified by the *user* argument for which the caller has permission.

If no signal is specified, then by default SIGTERM is sent.

The *user* argument is interpreted as a login name, unless no such name is found in the password map, in which case, if it is numeric, it is interpreted as a UID.

OPTIONS

The mutually exclusive options -l and -u determine the interpretation of the *user* argument:

-l	Interpret *user* argument as a login name only.
-u	Interpret *user* argument as a numeric UID only. This option also allows the use of UIDs that are not listed in the password map.
-s *signame*	Specify the signal to send, using one of the symbolic names defined in the <sys/signal.h> description. Values of *signame* will be recognized in a case-independent fashion, without the SIG prefix.
-signum	Specify the signal to send, where *signum* is a numeric value as defined in the <sys/signal.h> description.

EXAMPLES

Any of these commands:

```
srmkill -l user1
```

```
srmkill -s TERM user1
```

```
srmkill -s term user1
```

sends a SIGTERM signal to the processes currently attached to the user1 lnode.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWsrmb

SEE ALSO

kill(1), limadm(1MSRM), srm(5SRM)

Solaris Resource Manager 1.3 System Administration Guide

NOTES

This man page is applicable to SunOS 5.6, SunOS 5.7, and SunOS 5.8.

Solaris Resource Manager System Calls

NAME

brk, sbrk – change the amount of space allocated for the calling process's data segment

ADDENDUM

If the Solaris Resource Manager software is installed and enabled, then `brk` and `sbrk` may fail due to the reaching of a memory limit. Errno is set to ENOMEM on failure.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWsrmr

SEE ALSO

`brk`(2), `srm`(5SRM)

Solaris Resource Manager 1.3 System Administration Guide

NOTES

This man page is applicable to SunOS 5.6, SunOS 5.7, and SunOS 5.8.

NAME

nice – change priority of a process

ADDENDUM

`nice` allows a process in the Solaris Resource Manager CPU scheduling class (SHR) to change the relative rate at which it uses CPU access time. A process that has a higher `nice` value gets a lower rate of CPU service than processes with lower `nice` values on the same lnode.

As a special case, processes at the maximum `nice` value are only given CPU service when no other processes are waiting for CPU access time.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWsrmr

SEE ALSO

`nice`(2), `priocntl`(2), `limadm`(1MSRM), `srm`(5SRM)

Solaris Resource Manager 1.3 System Administration Guide

NOTES

This man page is applicable to SunOS 5.6, SunOS 5.7, and SunOS 5.8.

NAME

setuid, setegid, seteuid, setgid – set user and group IDs

ADDENDUM

If the Solaris Resource Manager software is installed and enabled, then the `setuid`(2SRM) system call, in addition to its standard function, also attaches the calling process to the lnode associated with its new *real UID*. The attachment is only performed if `setuid` is successful and the calling process's real UID is changed; otherwise the calling process remains attached to its original lnode.

If no lnode with the new real UID exists, then the calling process is instead attached to the system's `srmlost` lnode. The `srmlost` lnode is nominated at system startup and is typically an otherwise unused lnode, reserved specifically for this purpose. To avoid security breaches, the `srmlost` lnode usually has moderate resource limits, a small CPU share, and no special privileges.

If no `srmlost` lnode is ever nominated, or attachment to it fails, then the calling process remains attached to its original lnode.

The standard semantics of `setuid` are not affected by this new functionality: none of the above mentioned failures will cause `setuid` to fail or return an error. Instead, whenever `setuid` fails to attach a process to the specified lnode, a message is logged on the console to alert the system administrator.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWsrmr

SEE ALSO

`setuid`(2), `srm`(5SRM)

Solaris Resource Manager 1.3 System Administration Guide

WARNINGS

In the event of failure to attach to the specified lnode or the `srmlost` lnode, the process remains attached to its original lnode. This is a potential security breach, because the original lnode may possess privileges that should have been denied by the new lnode, and resource usage following the `setuid` will still be charged to the original lnode.

It is the responsibility of the system administrator to ensure that lnodes exist for all users.

NOTES

This man page is applicable to SunOS 5.6, SunOS 5.7, and SunOS 5.8.

Solaris Resource Manager Headers, Tables, and Macros

NAME

pam_srm – account and session management PAM module for Solaris Resource Manager

SYNOPSIS

`/usr/lib/security/pam_srm.so.1`

DESCRIPTION

The Solaris Resource Manager service module for PAM, `/usr/lib/security/pam_srm.so.1`, provides functionality for account management and session management.

`/usr/lib/security/pam_srm.so.1` is designed to be stacked on top of any other PAM service modules, for all login gateways. It is usually configured as *requisite*.

Entry Points

This service module provides the following entry points:

`pam_sm_acct_mgmt()`

Informs the user of any resource-limit-exceeded warnings, and returns PAM_PERM_DENIED if the number of such warnings is greater than warn limit. It is assumed the user has already been authenticated.

The following options may be passed to the Solaris Resource Manager account management module:

`debug`	Cause extra `syslog` debugging information at `LOG_DEBUG` level.
`nowarn`	Turn off warning messages.
`everylogin=`*scriptfile*	As the last phase in account management, run the script file (as superuser, or for `su`, as the invoking user). If the script file exits non-zero, access will be denied.
`nolnode=`*scriptfile*	If the account requested has no lnode, run the script file (as superuser, or for `su`, as the invoking user).

Both scripts are invoked without arguments and have only the following environment variables set:

`USER`
The name of the user who will be logged in if access is permitted.

`UID`
The UID of the user who will be logged in, or for `su`, the UID of the user invoking `su`.

`DEBUG`
Either `true` or `false`, depending on the value of the `debug` option.

`SERVICE`
The name of the service that invoked `pam`.

`pam_sm_open_session()`

Attaches to the lnode as a login session. If there is already a login session for the user, and the user's `onelogin` flag evaluates to `set`, `pam_sm_open_session()`

returns PAM_SESSION_ERR. The `debug` and `nowarn` options are accepted by `pam_open_session()` with meanings as above.

`pam_sm_close_session()`
Informs Solaris Resource Manager that the login session has completed.

Login Control

Currently, Solaris Resource Manager can control the number of concurrent logins of a user and/or a scheduling group by setting the following flags of the limit lnode using the `limadm`(1MSRM) command:

`flag.nologin=set` No login permitted

`flag.onelogin=set` One login permitted

It is also possible to set no restrictions, and permit an unlimited number of logins.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWsrmb, SUNWsrmr

SEE ALSO

`pam`(3PAM), `pam_authenticate`(3PAM), `pam_setcred`(3PAM), `syslog`(3C), `libpam`(4), `pam.conf`(4), `limadm`(1MSRM), `srm`(5SRM)

Solaris Resource Manager 1.3 System Administration Guide

WARNINGS

If the `pam_conv()` function provided by the `login` gateway does not support PAM_TEXT_INFO and PAM_ERROR_MSG message styles, warnings and informational messages provided by `pam_share` will not be visible to the user.

NOTES

This man page is applicable to SunOS 5.6, SunOS 5.7, and SunOS 5.8.

NAME

srm – Solaris Resource Management user administration and resource control

INTRODUCTION

Solaris Resource Manager is an optional enhancement of the Solaris operating environment to provide:

- Explicit allocation of system resources (CPU, virtual memory, terminal connect-time and number of connections, and process count) to users
- Accumulated accounting information on the resource usage of each user
- Fully hierarchical grouping of users, allowing the same degree of control and accounting at higher organizational levels
- Facilities for decentralized administration of users by sub-administrators

GETTING STARTED

Type the command:

```
liminfo
```

If Solaris Resource Manager is operating, then the output is a display of your assorted resource and accounting information, which is an overview of most of the functionality of Solaris Resource Manager.

The default directory for Solaris Resource Manager man pages is `/usr/srm/man`. The default directory for SunOS man pages is `/usr/man`.

Lnode

Solaris Resource Manager maintains additional non-volatile information for each user. The kernel is modified to represent each user internally with a new structure called a limit node (*lnode*). Each user's lnode is indexed by the user's UID number. The lnode contains all extra per-user information required by Solaris Resource Manager. Most of the information displayed by `liminfo` is taken directly from the lnode.

There are some special lnodes:

`root`	Always exists; it is the root of the scheduling tree and is not subject to limits.
`srmlost`	See `srmadm`(1MSRM).
`srmidle`	See `srmadm`(1MSRM).
`srmother`	See `limadm`(1MSRM).

Attributes: Lnode Data Fields

The data fields in an lnode are called *attributes*, which are referenced by name. Each attribute has one of the *types*:

integer	
long	
time	`integer`; a time interval in seconds
date	`integer`; a date/time in seconds relative to system epoch, 1-Jan-1970

uid	`integer`; a UID
double	
flag	`enum { inherit, set, clear, group }`

All attributes with integer or long types are currently treated as unsigned.

A *flag* is similar to a boolean: it evaluates to either `set` or `clear`. A third value, *group*, is used exclusively by device flags such as `terminal.flag.`*devicename*. The special value *inherit* is explained below in the section on Hierarchical Control.

Scheduling Tree

Lnodes are arranged in a strict, system-wide hierarchy called the *scheduling tree*, with the *root* lnode at its root. The first layer of lnodes below root refer to root as their *parent*, and are described as *children* of root. This relationship repeats with each level of the hierarchy. Each subtree of the scheduling tree is called a *scheduling group*. The lnode at the root of each subtree is called the group's *header*. All lnodes in a scheduling group except the header are known as *members* of the header's scheduling group.

The administrators of a system may use scheduling groups to represent the organizations, departments, and projects that use the system. Note that scheduling groups have nothing to do with the *file groups* defined in the group database.

All internal as well as leaf lnodes represent users, so all group headers require unique UIDs. Group header users can optionally be granted limited administrative power over the members of their scheduling group. There is no special property that distinguishes a user from a group header; a group header is simply a user who is the scheduling tree parent of one or more other users.

The structure of the scheduling tree is defined by the following attribute:

`sgroup`	*Uid*; assigned by the administrator. The UID of the lnode's parent in the scheduling tree. The value of this attribute is meaningless for the root lnode but evaluates to zero.

RESOURCE CONTROL

Solaris Resource Manager can be customized by the system administrator to control all kinds of resources. Control for the following system resources is built into Solaris Resource Manager:

- CPU (rate of process execution)
- Virtual memory size
- Process count
- Terminal connect-time/number of connections

Resource Models

Every user's resource allocations are controlled by *usage* and *limit* attributes in the lnode. A usage has a value that is increased as a resource is consumed, and decreased as the resource is released. A limit has a value which the usage is not permitted to exceed. A limit of zero is commonly used to represent no limit. Accounting information is kept in *accrue* attributes, which have non-decreasing values that measure the consumption of a resource over time. A variety of privileges are controlled by *flag* attributes.

Hierarchical Control

Resources are *hierarchically* controlled, that is, an entire group can be allocated resources as though it were a single user. This is achieved by making an lnode's resource limit apply to the total usage of the scheduling group of which the lnode is header.

The *usage* attribute of a hierarchically controlled resource is the sum of the user's own resource usage plus the *usage* attributes of the child lnodes. There is often an accompanying *myusage* attribute, which is equal to the user's own usage. The *limit* attribute applies to *usage*, not to *myusage*.

Privileges are hierarchically controlled using the special flag value *inherit*. Whenever a flag's value is tested, if it is found to be *inherit*, then the value is taken from the lnode's scheduling tree parent. If that is also *inherit*, then the search continues up through the scheduling tree until a real value is found, or the root lnode is reached. If the root lnode's flag is *inherit*, then a configurable system-wide default value is used.

Semantics

Every process in the system is attached to an lnode and is subject to the kernel limits and privileges of that lnode. When a process forks, the child is attached to the same lnode as its parent. The `init` process and all system processes are usually left attached to the root lnode. A process gets attached to a different lnode only in certain cases when it calls the `setuid`(2) system call, for which it must have superuser permission.

An *active* lnode is one that has one or more processes attached to it, or has one or more active member lnodes. That is, when a process attaches to an lnode, that lnode and all of its parents in the scheduling tree become active.

If Solaris Resource Manager is installed in the kernel, but no lnode database exists, all processes are attached to a surrogate root lnode. This is replaced by the real root lnode when the lnode file is opened.

CPU

The Solaris Resource Manager CPU scheduler, SHR, differs from the Solaris time-sharing scheduler (TS) in that it schedules users against each other, rather than LWPs, making it impossible for any user to acquire more CPU service just by running more processes concurrently.

When Solaris Resource Manager is enabled, a scheduling class module, SHR which is a functional replacement of the TS class, is loaded. The `init` process is then usually started in the SHR class, hence all LWPs of processes started by `init` begin in the SHR class. LWPs can still be moved into the RT class for real-time scheduling; system kernel processes remain in the SYS class. Only LWPs in the SHR class are subjected to Solaris Resource Manager scheduling.

Regardless of their scheduling class, all LWPs of a process are always attached to the same lnode. Like ownership of an address space, or credentials, lnode attachment is a process property that affects all the LWPs of the process.

The relevant attributes are:

`cpu.shares`	integer; assigned by the administrator. The number of shares given to the whole group. This defines what fraction of the parent group's entitlement is allocated to this group, as a ratio with the summed `cpu.shares` of all active peer lnodes and `cpu.myshares` of the parent lnode.
`cpu.myshares`	integer; assigned by the administrator. The number of shares given to the group header user. This defines what fraction of the group's entitlement is allocated to the group header user, as a ratio with the summed `cpu.shares` of all active children lnodes. It is meaningless for leaf lnodes.
`cpu.usage`	double; accumulated and decayed by the kernel. The weighted sum of charges for recent CPU service.
`cpu.accrue`	double; accumulated by the kernel. The weighted sum of charges for CPU service.
`lastused`	date; set by the kernel. The most recent time at which the `cpu.usage` attribute was updated.

CPU entitlements are chosen by the administrators, who assign each lnode a number of *shares* and *myshares*. These are analogous to shares in a company: the absolute quantity is not important; they are meaningful only in comparison with other lnode's shares. Administrators are free to choose whatever numbers they want for these two attributes, as long as they are in proportion to the desired CPU entitlements. Furthermore, the choice of numbers at any level of the scheduling tree is completely independent of the choice of numbers at any other level or branch.

For example, consider a single level of one branch of a scheduling tree, with user A1 as the header, and users B1, B2, and B3 as children:

A1	`cpu.shares=250,`	`cpu.myshares=15`
B1	`cpu.shares=10,`	`cpu.myshares=100`
B2	`cpu.shares=20,`	`cpu.myshares=3000`
B3	`cpu.shares=5,`	`cpu.myshares=2`

The total number of shares at this level = *header's myshares* + *children's shares* = 15 + (10 + 20 + 5) = 50. Suppose that the *group* A1 has a CPU entitlement of 60 percent. The *user* A1 therefore has a CPU entitlement of (15 / 50) x 60% = 18%, the group B1 has a CPU entitlement of (10 / 50) x 60% = 12%, the group B2 (20 / 50) x 60% = 24%, and the group B3 (5 / 50) x 60% = 6%. These entitlements might be further subdivided below B1, B2, and B3, in a similar fashion.

As shown in the example, the *shares* attributes of an lnode, in relation to other lnodes, define the *CPU* entitlement of the lnode. Over the longer term, provided they use it, Solaris Resource Manager will ensure that lnodes receive CPU in proportion to their entitlements.

By ignoring inactive lnodes, a related value is directly calculated by the scheduler as a fraction between 0 and 1, and formally called the *allocated share*. All allocated shares are recomputed whenever any lnode becomes active or inactive, or whenever a *myshares* or *shares* attribute is changed.

As processes execute, charges are accumulated in the `cpu.usage` attribute of the lnodes to which they are attached. The kernel periodically decays the CPU usage in every lnode by multiplying it with a decay factor which is less than 1, so that more recent CPU usage has greater weight when taken into account for scheduling. The scheduler continually adjusts the priority of all processes to make each lnode's relative CPU usage converge on its allocated share. This negative feedback mechanism allows direct control of the proportion of CPU rate of service granted to each user and group.

At any instant, it is unlikely that users will be receiving exactly their allocated shares worth of CPU rate, due to uneven demand. The *effective share* of an lnode is the rate of CPU that it must receive in order to restore the balance of CPU usages with allocated shares. A user's effective share is a rough measure of what fraction of CPU will be given to the user at the time, should it be required.

The `nice` command has an appropriate effect on processes: a higher `nice` value means that a process will run more slowly. Under Solaris Resource Manager, however, such a process will accumulate charges at a discounted rate, so the user also benefits by using `nice`. The maximum `nice` value is treated specially: the process is a *background* process and it is scheduled only when there is spare CPU capacity not demanded by non-background processes. The `priocntl` command can also be used to set or display a process's `nice` value. See `nice`(1) and `priocntl`(1MSRM).

Memory

The relevant attributes are:

`memory.myusage`	long; read-only; computed by the kernel. The total virtual memory space occupied by all processes attached to the lnode, measured in bytes to 1 Kbyte resolution.
`memory.usage`	long; read-only; computed by the kernel. The total virtual memory space occupied by all processes attached to the lnode and all its member lnodes, measured in bytes to 1 Kbyte resolution. This is computed as `memory.myusage` of the lnode plus the sum of the `memory.usage` attributes of all child lnodes.

`memory.limit`	long; assigned by the administrator. The maximum allowed value, in bytes, of the `memory.usage` attribute. If zero, then there is no limit, unless limited by inheritance.
`memory.plimit`	long; assigned by the administrator. The maximum virtual memory space, in bytes, that may be occupied by any individual process attached to the lnode. If zero, then there is no limit, unless limited by inheritance.
`memory.accrue`	long; accumulated by the kernel. The accurate, continuous sum over time of the value of the `memory.usage` attribute, with dimension byte-seconds.

Memory is allocated by stack page faults, by the `mmap`(2) system call, and by a few other system calls. If a memory limit is reached, these will fail. A failed stack fault will cause the process to terminate unless the terminating signal is caught on an alternate stack. A failed system call will appear to programs as though there is no more virtual memory (swap space). Some programs will accept this fact and continue normal operation, possibly outputting a warning. Some programs will fail outright, possibly with a helpful diagnostic message. Solaris Resource Manager will write a warning message to the terminals of all affected local users (see `limdaemon`(1MSRM)).

Memory accounting and limits apply to processes irrespective of their scheduling class.

Connect-Time

The relevant attributes are :

`terminal.usage`	long (time interval); increased and decreased by `limdaemon`(1MSRM). The number of seconds of connect-time currently charged to the group. This may not equate to the real-time duration of connections if device costs other than 1.0 have been configured.
`terminal.decay`	long (time interval); assigned by the administrator. The amount subtracted from the `terminal.usage` attribute for every decay point.
`terminal.interval`	time; assigned by the administrator. The time interval between decay points.
`terminal.lastdecay`	date; assigned by the administrator and updated by `limdaemon`(1MSRM). The time at which the most recently performed decay point occurred.

`terminal.limit`	long (time interval); assigned by the administrator. The maximum allowed value of the `terminal.usage` attribute. If zero, then there is no limit, unless limited by inheritance.
`terminal.accrue`	long (time interval); increased by `limdaemon`(1MSRM). The total number of seconds of connect-time used by the group.
`terminal.flag.`*devicename*	flag; assigned by the administrator. These flags need not exist for all devices. A user may log in on a given device only if its corresponding flag exists and evaluates to `set`, or if no such flag exists.

Logins are recognized by Solaris Resource Manager through a special PAM module, `pam_srm`(5SRM).

At login time, the connect-time limits of the user and all scheduling groups to which the user belongs are checked. If the user's terminal usage exceeds its limit, then the user is informed and login is denied. Otherwise, the name and cost of the device are output and the user is allowed to log in. The superuser is exempt from these checks.

While logged in, if any users or scheduling groups come within 5 real minutes of reaching a connect-time limit, a warning message is written to the terminals of all such users or members of the scheduling groups. When the limit is reached, a message is written requesting immediate logout. A short time later (default grace period: 30 seconds), if any of the requested users are still logged in, then their associated processes are sent a SIGTERM signal, and soon after (15 seconds), a SIGKILL signal.

The update of terminal usage and enforcement of terminal limits is performed on a periodic basis by the Solaris Resource Manager daemon process, `limdaemon`(1MSRM). If the daemon is not running, then usages are not increased and limits are not enforced, except for those limits checked at login time. Connect-time accounting and limits apply to processes irrespective of their scheduling class.

Warning Messages

Whenever users reach a limit or approach a connect-time limit, a warning message is written to their terminals by the `limdaemon`(1MSRM) process. Messages are written directly to a user's terminal by looking up the `utmp` entries for the device name, and using the `write`(2) system call. The warnings are in terse, human-readable text, and typically have the form:

`User` *username* "resource limit reached" by *username* .

The first *username* is the lnode in which the limit was reached. The message is delivered to all users in the scheduling group headed by this lnode because all of these users are affected. The second *username* is the user whose action caused the limit to be reached.

Privileges

Privileges are usually controlled using *flag* attributes. The following attributes are predefined:

`flag.uselimadm`	flag; assigned by the administrator. If set, the user is a sub-administrator who can freely add, modify, and remove lnodes, alter any attribute in any lnode, and attach a process to any lnode. This flag has no effect on the superuser, who always has these powers. See `flag.admin` below, and SUB-ADMINISTRATORS AND GROUP ADMINISTRATORS.
`flag.admin`	flag; assigned by the administrator. If set, the user is a *group administrator*. This flag grants privileges over users in the scheduling subtree of which the lnode is group header. This flag has no effect on the superuser, who always has these powers. See SUB-ADMINISTRATORS AND GROUP ADMINISTRATORS.
`flag.nologin`	flag; assigned by the administrator. If set, the user cannot log in and is denied connection by remote shell or execution daemons. If both the *nologin* and *onelogin* flags are set, then *nologin* takes precedence.
`flag.onelogin`	flag; assigned by the administrator. If set, the user can have at most one login connection. Connections by remote shell or execution daemons are counted as logins for this purpose. If both the *nologin* and *onelogin* flags are set, then *nologin* takes precedence.
`logins`	integer; read-only, computed by the Solaris Resource Manager kernel. The current number of logins. These represent login or remote connections recognized by Solaris Resource Manager. This attribute does not strictly control a privilege, but is described here because it is related to the *onelogin* and *asynckill* flags.
`flag.asynckill`	flag; assigned by the administrator. If set, then all processes attached to the lnode are killed whenever the value in the `logins` attribute drops to zero.
`flag.asyncnice`	flag; assigned by the administrator. If set, then all processes attached to the lnode are set to lowest `nice` when the `logins` attribute drops to zero.

limreport

`limreport`(1SRM) is a simple but powerful report generator available to all users. It scans sequentially through the password map, selecting users according to a specified *selection* expression. For each selected user, a report is output. Its format is specified in the style of `printf`(3C). Expressions for selection and for insertion into the report can refer to any lnode attribute and to password map fields.

liminfo

`liminfo`(1SRM) outputs a report on the contents of the invoker's lnode, or gives a sequence of reports on the lnodes of a list of users. There are five report formats available. Three of the reports are designed for easy reading by users, the fourth is designed specifically for use by filters, and the fifth is for debugging.

SUB-ADMINISTRATORS AND GROUP ADMINISTRATORS

A user with a set `uselimadmin` flag (see Privileges) is a sub-administrator. Sub-administrators have the same powers as a central administrator. Group administrators have only a set `admin` flag

The tools listed in the next section, LNODE MANAGEMENT, are available with full function to the superuser and with restrictions to group administrators.

LNODE MANAGEMENT

These commands provide for management of lnodes.

limadm

`limadm`(1MSRM) is the main tool for altering the attributes of an lnode. Alterations are expressed as a list of numeric or symbolic assignments, additions, or subtractions to named attributes. Superusers and `uselimadm` users can use `limadm` to alter any writable attribute in any lnode. Sub-administrators are prevented from altering the assigned limits and privileges of lnodes other than those of their own scheduling group members. Any valid assignment to an attribute of a non-existing lnode creates the lnode. This command should be used to create an lnode just after creating the first reference to the corresponding UID in the password map.

`limadm`(1MSRM) can also be used to delete an inactive leaf lnode. Sub-administrators are prevented from deleting lnodes outside their scheduling group members. This command should be used to delete an lnode just prior to deleting the last reference to the corresponding UID from the password map.

limdaemon

`limdaemon`(1MSRM) is started automatically in the Solaris Resource Manager startup script at boot time. It decays the *usage* attributes of terminals in all lnodes, or only in the lnodes of a given list of users. The decay of usage values is regulated using the *decay*, *interval*, and *lastdecay* attributes in each lnode.

srmuser

`srmuser`(1SRM) attaches a shell to a named lnode, and optionally executes a given command. This can be useful when performing a costly operation on behalf of another user, because the user rather than the administrator is charged for the CPU and memory used. Note that, unlike `su`(1M), this does not alter the real and effective UIDs.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWsrmb, SUNWsrmr

SEE ALSO

`nice`(1), `su`(1M), `mmap`(2), `setuid`(2), `write`(2), `printf`(3C), `liminfo`(1SRM), `limreport`(1SRM), `srmstat`(1SRM), `srmuser`(1SRM), `dispadmin`(1MSRM), `limadm`(1MSRM), `limdaemon`(1MSRM), `priocntl`(1MSRM), `srmadm`(1MSRM), `srmkill`(1MSRM), `brk`(2SRM), `nice`(2SRM), `setuid`(2SRM), `pam_srm`(5SRM)

Solaris Resource Manager 1.3 System Administration Guide

WARNINGS

The default state of a newly created lnode has most attributes zeroed; only `cpu.shares` and `cpu.myshares` are set to a minimum value, which is *1*, while the `uselimadm` and `admin` flags are set to `clear`, and all other flags are set to `inherit`. Therefore, administrative privilege over other users is denied by default, and it is only by deliberate action that users will be given limits to encounter. Thus, it is the responsibility of sub-administrators to consider fully the implications of their decisions before acting.

The Solaris Resource Manager system is powerful and its effects can be widely felt, so misuse may have large and unpleasant consequences. Users of Solaris Resource Manager must formulate clear and strong policies on system administration.

The current Solaris Resource Manager message notification mechanism only sends the message to local users.

Any locale-specific translation of messages delivered to local users is in accordance with the locale of the `limdaemon` process, which may differ from that of the users.

NOTES

This man page is applicable to SunOS 5.6, SunOS 5.7, and SunOS 5.8.