Sun microsystems

Solaris Resource Manager 1.0 System Administration Guide for Solaris 2.6 (SPARC Platform Edition)

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
U.S.A.

Part No: 805-7480–10
December 1998

Copyright 1998 Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, California 94303-4900 U.S.A. All rights reserved.

ShareII Copyright 1989-1998 Softway Pty. Limited. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, SunDocs, Sun Enterprise, Sun Enterprise SyMON, Solaris, Solaris Resource Manager, AnswerBook, NFS, and docs.sun.com are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc. ShareII is a trademark or registered trademark of Softway Pty. Limited in the U.S. and other countries.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227–14(g)(2)(6/87) and FAR 52.227–19(6/87), or DFAR 252.227–7015(b)(6/95) and DFAR 227.7202–3(a).

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 1998 Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, Californie 94303-4900 Etats-Unis. Tous droits réservés.

ShareII Copyright 1989-1998 Softway Pty. Limited. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées du système Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, SunDocs, Sun Enterprise, Sun Enterprise SyMON, Solaris, Solaris Resource Manager, AnswerBook, NFS, et docs.sun.com sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc. ShareII est une marque de fabrique ou une marque déposée de Softway Pty. Limited aux Etats-Unis et dans d'autres pays.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REPONDRE A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.

Adobe PostScript

Please
Recycle

Contents

Preface

This guide is for system administrators who are responsible for configuring and administrating Solaris Resource Manager™ 1.0 software on the Solaris™ Operating Environment.

Before You Read This Book

Before using this book, you should be familiar with the information in these AnswerBook™ collections, available at `docs.sun.com`.

- *Solaris 2.6 System Administrator Collection Vol 1*
- *Solaris 2.6 System Administrator AnswerBook Vol 2*

How This Book Is Organized

This book is organized into chapters, an appendix, and a glossary.

Chapter 1 provides an introduction to Solaris Resource Manager and describes how this product can be used to allocate and control major system resources.

Chapter 2 discusses the operation of the Solaris Resource Manager software and provides usage examples.

Chapter 3 describes how to configure the Solaris Resource Manager software on your Solaris system.

Chapter 4 describes the effects of the UNIX™ boot procedure on the Solaris Resource Manager product.

Chapter 5 discusses the per-user structure introduced in Solaris Resource Manager.

Chapter 6 discusses the scheduler, which is used to control the allocation of the CPU resource.

Chapter 7 describes how to control the amount of virtual memory held by users and individual processes.

Chapter 8 describes the mechanism for collecting accrued usage values for system and user resources.

Chapter 9 provides assistance in diagnosing problems in the operation of Solaris Resource Manager.

Chapter 10 describes possible error messages and their meanings.

Appendix A provides sample scripts.

Glossary is a list of words and phrases found in this book and their definitions.

Related Books

The following resources provide installation, configuration, usage, and release information for the Solaris Resource Manager product:

- The *Solaris Resource Manager 1.0 Release Notes for Solaris 2.6 (SPARC Platform Edition)* document is included in the product box. It provides a brief product introduction to the Solaris Resource Manager software, identifies patches required to use the product, and contains information on bugs and known problems.
- The *Solaris Resource Manager 1.0 Installation Guide for Solaris 2.6 (SPARC Platform Edition)* describes how to install Solaris Resource Manager on your operating system. It is included in the product box.
- The *Solaris Resource Manager 1.0 Reference Manual for Solaris 2.6 (SPARC Platform Edition)* is the AnswerBook version of the Solaris Resource Manager man pages. These entries supplement the Solaris 2.6 base manual pages installed on your system. The administration guide will reference these pages. Online versions of the man pages, accessible using the `man` command, are also provided in the Solaris Resource Manager SUNWsrm package.

Ordering Sun Documents

The SunDocs^{SM} program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals using this program.

For a list of documents and how to order them, see the catalog section of the SunExpress™ Internet site at `http://www.sun.com/sunexpress`.

Accessing Sun Documentation Online

The docs.sun.com™ Web site enables you to access Sun technical documentation online. You can browse the `docs.sun.com` archive or search for a specific book title or subject. The URL is `http://docs.sun.com`.

The Solaris Resource Manager 1.0 Collection, which includes this document, is available on `docs.sun.com`; you can check this location to see if an updated version of this guide has been produced.

TABLE P–1 Typographic Conventions

Typeface or Symbol	**Meaning**	**Example**
`AaBbCc123`	The names of commands, files, and directories in text; on-screen computer output	Edit your `.login` file. Use `ls –a` to list all files. `machine_name% you have mail.`

`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name%` **`su`**
		`Password:`

AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type `rm` *filename*.
AaBbCc123	Book titles, new words or terms, or words to be emphasized.	Read Chapter 6 in *User's Guide*. These are called *class* options. You *must* log in first.

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–2 Shell Prompts

Shell	**Prompt**
C shell prompt	`machine_name%`
C shell superuser prompt	`machine_name#`
Bourne shell and Korn shell prompt	`$`
Bourne shell and Korn shell superuser prompt	`#`

CHAPTER **1**

Overview

The Solaris Resource Manager software ensures resource availability for users, groups, and applications. It provides the ability to allocate and control major system resources such as CPU, virtual memory, and number of processes. It also implements administrative policies that govern which resources different users can access, and more specifically, what level of consumption of those resources each user is permitted. The Solaris Resource Manager product is a key enabler for server consolidation and increased resource utilization.

The Solaris Resource Manager product is based on ShareII™ technology from Softway Pty. Limited, Australia.

Introduction to Solaris Resource Manager

Organizational Goals It Addresses

Businesses often require that IT organizations control costs and guarantee service levels for enterprise applications. Resource management makes a number of procedures available that lower overall total cost of ownership, give more accurate control over who uses the system and how they use it, and sometimes serve both goals.

By using Solaris Resource Manager software to categorize and prioritize usage, administrators can effectively utilize reserve capacity during off-peak periods, often eliminating the need for additional processing power.

By segregating workloads within the system, Solaris Resource Manager enables the system administrator to run and manage dissimilar applications on a single system, rather than dedicating an entire system—complete with peak capacity—to each application. Traditionally, the most common approach to ensuring predictable service and response time is to host one function per system. This method works, but the proliferation of systems in the data center is expensive and difficult to manage.

Data center managers want the ability to consolidate multiple applications on a single UNIX server, thus fully utilizing all available resources. At the same time, all users must receive resources commensurate with their service levels and the relative importance of their work.

When to Use Solaris Resource Manager

Solaris Resource Manager can provide effective resource control in a variety of situations including server consolidation, Internet Service Provider (ISP) web hosting, administrating sites with large or varied user populations, and establishing policies to ensure that critical applications get the response time they require.

Solaris Resource Manager is ideal for environments that are consolidating multiple applications on a single server. The cost and complexity of managing numerous machines encourages system managers to consolidate applications on larger, more scalable systems. With Solaris Resource Manager, it's easy to achieve these economies of scale.

As an example, a single Sun™ server could provide application, file, and print services for heterogeneous clients, messaging/mail service, web service, and mission-critical database applications. Since Sun Enterprise™ servers scale from 1 to 64 processors, one server could be configured for several departments to share or for an entire enterprise to use. In other server consolidation efforts, the development, prototype, and production environments are combined on a single large machine such as the Sun Enterprise 10000 or Sun Enterprise 6500, rather than being hosted on three separate servers. Still other consolidation projects combine database and application servers within a single machine, or multiple data marts. Regardless of the application type or configuration, Solaris Resource Manager helps ensure that the system's resources are allocated among all users, applications, and groups according to the defined policy. Critical applications are protected because they are guaranteed the share of the available system resources they need.

Similarly, with Solaris Resource Manager, an ISP can confidently host many (perhaps thousands) of web servers on a single machine. Solaris Resource Manager allows administrators to control the resource consumption associated with each web site, protecting each from the potential excesses of the others. Solaris Resource Manager also prevents a faulty CGI script from exhausting CPU resources, or a user application from leaking all available virtual memory. In the past, ISPs have had to assign dedicated machines to each client, at significant cost and complexity.

Solaris Resource Manager can help manage resources in any system that has a large number of users. Educational institutions are good examples of sites that serve a large and diverse user base. (In fact, Solaris Resource Manager has its roots in an early CPU resource scheduler developed at the Universities of Sydney and New South Wales.) Where there is a mix of workloads, Solaris Resource Manager can be configured to favor certain users. In large brokerage firms, traders intermittently require fast access to execute a query or perform a calculation. Other system users, however, have more consistent workloads. If the traders are granted a proportionately larger amount of processing power, Solaris Resource Manager ensures that they will have the responsiveness they need.

Main Features

Solaris Resource Manager provides the ability to administer the consumption of various important resources in the system, such as processor time, virtual memory, process count, login count, and connect time. The Solaris Resource Manager administrative model adds flexibility by permitting the delegation of administrative rights within a hierarchy, relieving the data center staff from the need to be involved in intra-group administrative transactions. In addition, Solaris Resource Manager provides mechanisms for collecting resource usage data that can be applied to capacity planning or chargeback purposes.

One of the fundamental jobs of the operating system is to arbitrate which processes will get access to the system's resources. The default Solaris timeshare scheduler tries to give every process approximately equal access to the system's resources. Limitations on access are applied to processes without physical memory resources, which are not permitted to run, and processes with pending I/O requests, which are blocked.

This scheme is the basis for most modern operating systems; it works well as long as "equal access for all" is a suitable policy for the organization. However, more sophisticated mechanisms are required to implement different policies. For example, a manufacturing department might own a large system that is usually used very lightly because of fluctuating seasonal demand. At the same time, the engineering department almost always needs more computational cycles. Although it is wasteful to underuse a large machine's resources, sharing the manufacturing system with engineering has traditionally been problematic. With simple scheduling policies, there is no way to express to the operating system that the manufacturing department's users are more important than the engineering users on the same system. If manufacturing has a critical job running that consumes 75 percent of the system's resources, the job will make suitable progress if all other jobs request 25 percent of the system or less. However, if an engineering job arrives that demands 50 percent of the system, that critical manufacturing job will likely not get what it needs to maintain adequate headway, because the system will try to accommodate both jobs on an equal basis.

Now assume that the administrators determine that manufacturing's normal processing requirements can be met with 80 percent of the machine's capabilities. Using Solaris Resource Manager, the system administrator can specify that the manufacturing department's users can have up to 85 percent of the system's processing capability if they request it, and the scheduler will apportion the remainder to any other user. A more extreme but equally valid configuration might specify that manufacturing users can have up to 100 percent of the system if necessary, effectively precluding any other group's processes from running in the event that manufacturing really needs the entire system.

Solaris Resource Manager provides a new CPU scheduling class that replaces the standard timesharing scheduler. Called the SHR scheduler class, this module implements what is called a fair share scheduler. The term is something of a misnomer, because it is the system administrator who specifies what "fair" means. In the example above, "fair" meant that manufacturing could get 100 percent of the system. The SHR scheduler is responsible for allocating resources according to the plan laid out in the administrative profile.

Solaris Resource Manager maintains a database of resource consumption and associated limits.

The SHR scheduler takes into account the administrative specification for resource guarantees. It is capable of managing resources that are renewable (such as CPU time) or fixed (such as number of logins).

Other Solaris Resource Manager modules implement restrictions on the consumption of various resources. For example, connect time and number of user logins are managed by a Pluggable Authentication Module (PAM). The PAM module consults the Solaris Resource Manager database each time a user attempts to log in. Once the system authenticates the user (generally through password matching), the user's connect time and the number of current logins are checked against the limits. The login is rejected if either of the limits is exceeded.

Relationship to Other Solaris Resource Control Features

The Solaris operating environment includes several other features that provide control over certain types of resources. Some features, such as realtime scheduling, `nice(1)`, quotas, and processor sets, are part of the basic Solaris operating system.

Bandwidth Allocator is an unbundled software package, dynamic system domains are features of the Sun Enterprise 10000 system platform, and dynamic reconfiguration is a feature of the Sun Enterprise system platform.

All of these components offer types of resource management, but they differ from Solaris Resource Manager capabilities in one way or another.

The standard Solaris operating system uses the timeshare (TS) scheduling class for most conventional work. However, it also offers realtime (RT) scheduling to users with sufficient privilege. The RT scheduling class implements a very different (and intentionally very weighted) scheduling policy to ensure that specific workloads or processes get immediate access to the processor. Solaris Resource Manager can coexist on the same system as the RT scheduling class, but it will have no control over any process running in the RT class. The Solaris Resource Manager fair share scheduler is able to manage the CPU time resources of only those processes that are not running in the RT scheduling class. For example, on a four-processor system, a single-threaded process can consume one entire processor; in fact this is precisely what happens if the requesting process is CPU-bound. If this system also runs Solaris Resource Manager, regular user processes will be competing for the three CPUs not already consumed by the realtime process. (Note that the RT processes might not use the CPU continually. When it is idle, Solaris Resource Manager will control all four processors.)

The `nice(1)` command permits users to manipulate their execution priority. Without superuser privilege, this command only permits the user to lower his priority. In some instances this is a useful feature (for example, when a user starts a low-priority batch job from his interactive login session), but it relies on the cooperation of the user. Solaris Resource Manager enforces administrative policies, even without the cooperation of the user.

Solaris file systems have quota mechanisms that enable the administrator to restrict the disk consumption of individual users. This functionality is independent of Solaris Resource Manager.

Processor sets were introduced in Solaris 2.6. This feature permits the administrator to divide multiprocessor systems into logical groups and permits users to launch processes into those groups. The advantage is that workloads running in one processor set are protected from CPU activity taking place in any other processor set. In some ways, this is similar to what Solaris Resource Manager does, but the two features operate on completely different bases. Processor sets control only CPU activity. The control is at a relatively coarse-grained hardware level, because processors may belong to exactly one processor set at a time. Especially in the case of relatively small systems, the granularity may be quite high: on a 4-processor system, the minimum resource that can be assigned is 25 percent of the system.

Solaris Resource Manager has much finer-grained control; each user is allocated a share of the system. The shares can be distributed arbitrarily on a fine granularity, and the scheduler will allocate resources accordingly. For example, if 50 shares are granted, and one user has 40 of them, that user will get 40 / 50 = 80 percent of the resource. Similarly, if 67 total shares are granted, a user with 57 shares will get 85 percent of the resource. In addition, Solaris Resource Manager can control resources other than CPU.

The Sun Enterprise 10000 has a feature called dynamic system domains, which permit the administrator to logically divide a single system rack into one or more independent systems, each running its own copy of Solaris. For example, a system with 32 CPUs on 8 system boards might be operated as 1 system with 16 CPUs, and 2 other systems with 8 CPUs each. Three copies of Solaris would be running in such a circumstance. The dynamic system domain feature also permits controlled movement of resources into and out of each of the Solaris images, thus creating a relatively coarse-grained facility for managing physical resources. (The minimum unit of inter-domain allocation is an entire system board.) Solaris Resource Manager is similar to dynamic system domains in that it provides the administrator with mechanisms to allocate resources, but it does so in very different ways. Solaris Resource Manager runs within a single instance of Solaris, and provides fine-grained administrative control to the resources in the system. Dynamic system domains divide a single piece of hardware into multiple instances of Solaris; it provides tools to manage the transfer of resources between instances of Solaris running in the same Sun Enterprise 10000 frame. Solaris Resource Manager is orthogonal to, and can be used in conjunction with, dynamic system domains. Solaris Resource Manager can be run in each instance of Solaris within a Sun Enterprise 10000 system.

The dynamic reconfiguration feature of Sun Enterprise servers enables users to dynamically add and delete system boards, which contain hardware resources such as processors, memory and IO devices. The effect of a dynamic reconfiguration operation on memory has no impact on Solaris Resource Manager memory-limit checking.

The Bandwidth Allocator is an unbundled package that works with the Solaris kernel to enforce limits on the consumption of network bandwidth. Bandwidth Allocator is a form of resource management software that applies to a different class of resources. Solaris Resource Manager and Bandwidth Allocator have different and disjoint management domains: Solaris Resource Manager operates on a per-user or per-application basis, while Bandwidth Allocator manages on a per-port, per-service, or per-protocol basis.

Differences Between Solaris Resource Manager and Similar Products

The Solaris Resource Manager is related to many other software components that may be present in the system, but it does not replace any of them. As noted, Bandwidth Allocator manages a different type of resource. And while Solaris Resource Manager can be viewed as having some system management and monitoring functions, it is not a system monitor in the sense that Sun Enterprise SyMON™ 2.0 is. Nor is Solaris Resource Manager really a capacity planning tool: it helps the administrator manage capacity, and its accounting functions construct usage records that the administrative staff might use to do trend analysis, but it does not do capacity planning in the traditional sense of the term. Solaris Resource Manager is also not a job scheduler; it controls how a process runs on its host system, rather than when or where it runs. Finally, because Solaris Resource Manager operates only on a single system, it is also not a mechanism for implementing load balancing across cluster members. Solaris Resource Manager can be used effectively to manage workloads individually on each member of a cluster, however. For example, arrangements might be made to prioritize work from a failed member of a high-availability cluster over a background workload running on standby member.

CHAPTER **2**

Normal Operations

This chapter describes Solaris Resource Manager principles of operation and key concepts. Examples are provided to reinforce the descriptions and to illustrate some of the common ways that Solaris Resource Manager is used.

Lnodes Overview

Solaris Resource Manager is built around a fundamental addition to the Solaris kernel called an lnode (limit node). lnodes correspond to UNIX UIDs, and may represent individual users, groups of users, applications, and special requirements. lnodes are indexed by UID and are used to record resource allocations policies and accrued resource usage data by processes at the user, group of users, and/or application level.

Hierarchical Structure

The Solaris Resource Manager management model organizes lnodes into a hierarchical structure called the scheduling tree. The scheduling tree is organized by UID: each lnode references the UID of the lnode's parent in the tree. Each sub-tree of the scheduling tree is called a scheduling group, and the user at the root of a scheduling group is the group's header. The root user is the group header of the entire scheduling tree. A group header can be delegated the ability to manage resource policies within the group. lnodes are initially created by parsing the UID file. An lnode administration command (`limadm(1MSRM)`) will create additional lnodes after installation of Solaris Resource Manager and assign lnodes to parents.

The scheduling tree data is stored in a flat file database, which can be modified as required.

Though UIDs used by lnodes do not have to correspond to a system account, with an entry in the system password map, it is strongly recommended that a system account is created for the UID of every lnode. For non-leaf lnodes (those with subordinate lnodes below them in the hierarchy), it may be the case that the account associated with that lnode is purely administrative and no-one ever logs in to it. However, it is equally possible that it can be the lnode of a real user who does log in and run processes attached to this non-leaf lnode.

Note that Solaris Resource Manager scheduling groups and group headers have nothing to do with the system groups defined in the `/etc/group` database. Each node of the scheduling tree, including group headers, corresponds to a real system user with a unique UID.

Hierarchical Limits

If a hierarchical limit is assigned to a group header in an lnode tree (scheduling group), then it applies to the usage of that user plus the total usage of all members of the scheduling group. This allows limits to be placed on entire groups, as well as on individual members. Resources are allocated to the group header, who may allocate them to users or groups of users that belong to the same group.

Processes

Every process is attached to an lnode. The `init` process is always attached to the root lnode. When processes are created by the `fork(2)` system call, they are attached to the same lnode as their parent. Processes may be re-attached to any lnode using a Solaris Resource Manager system call, given sufficient privilege. Privileges are set by root or by users with the correct administrative permissions enabled.

Resource Control

Solaris Resource Manager provides control of the following system resources: CPU (rate of processor) usage, virtual memory, number of processes, number of logins, and terminal connect-time.

Solaris Resource Manager keeps track of each user’s usage of each resource. For all resources except CPU usage, users may be assigned hard limits on their resource usages. A hard limit will cause resource consumption attempts to fail if the user allows the usage to reach the limit. Hard limits are directly enforced by either the kernel or whatever software is responsible for managing the respective resource. A limit value of zero indicates no limit. All limit attributes of the root lnode should be left set to zero.

Solaris Resource Manager progressively decays past usage so that only the most recent usage is significant. The system administrator sets a half-life parameter which controls the rate of decay. A long half-life favors even usage, typical of longer batch jobs, while a short half-life favors interactive users.

Generally all system resources can be divided into one of two classes: fixed (or non-renewable) resources and renewable resources. Solaris Resource Manager manages these two types of resources differently.

Fixed Resources

Fixed or non-renewable resources are those which are available in a finite quantity, such as virtual memory, number of processes, number of logins, and connect time. Fixed resources may be consumed (allocated) and relinquished (deallocated), but no other entity can use the resource before the owner deallocates it. Solaris Resource Manager employs a usage and limit model to control the amount of fixed resources used. Usage is defined as the current resource being used, and limit is the maximum level of usage that is permitted by Solaris Resource Manager.

Renewable Resources

Renewable resources are those which are in continuous supply, such as CPU time. Renewable resources may only be consumed, and, once consumed, cannot be reclaimed. At any one time, a renewable resource will have limited availability and if not used at that time will no longer be available in the future. (An analogy is sunlight. There is only a certain amount arriving from the sun at any given instant, but more will surely be coming for the next few million years.) For this reason, renewable resources can be reassigned to other users without explicit reallocation to ensure no waste. Solaris Resource Manager employs a usage, limit, and decay model to control a user’s rate of consumption of a renewable resource. Usage is defined as the total resource used, with a limit set on the ratio of usages in comparison to other fellow users. Decay refers to the period by which historical usage is discounted. The next resource quantum, for example, clock tick, will be allocated to the active lnode with the lowest decayed total usage value in relation to its allocated share. The decayed usage value is a measure of the total usage over time less some portion of historical usage determined by a half-life decay model.

The CPU resource is controlled using the Solaris Resource Manager SHR scheduler. Users are dynamically allocated CPU time in proportion to the number of shares they possess (analogous to shares in a company), and in inverse proportion to their recent usage. The important feature of the SHR scheduler is that while it manages the scheduling of individual threads (technically, in Solaris, the scheduled entity is a lightweight process (LWP)), it also portions CPU resources between users.

Each user also has a set of flags, which are boolean-like variables used to enable or disable selective system privileges, for example, login. Flags may be set individually per user, or may be inherited from a parent lnode.

The usages, limits, and flags of a user can be read by any user, but can be altered only by users who have administrative powers.

CPU Resource Management

The allocation of the renewable CPU service is controlled using a fair share scheduler. Each lnode is assigned a number of CPU shares, analogous to shares in a company. The processes associated with each lnode are allocated CPU resources in proportion to the total number of outstanding active shares, where active means that the lnode has running processes attached. Only active lnodes are considered for an allocation of the resource, as only they have active processes running and need CPU time. As a process consumes CPU ticks, the CPU usage attribute of its lnode increases. The scheduler regularly adjusts the priorities of all processes to force the relative ratios of CPU usages to converge on the relative ratios of CPU shares for all active lnodes at their respective levels. In this way, users can expect to receive at least their entitlements of CPU service in the long run, regardless of the behavior of other users. The scheduler is hierarchical, because it also ensures that groups receive their group entitlement independently of the behavior of the members. Solaris Resource Manager is a long-term scheduler; it ensures that all users and applications receive a fair share over the course of the "scheduler term." This means that when a light user starts to request the CPU, that user will receive commensurately more resource than heavy users until their comparative usages are in line with their relative "fair" share allocation. The more you use over your entitlement now, the less you will receive in the future. Additionally, Solaris Resource Manager has a decay period, set by the system administrator, that forgets about past usage. The decay model is one of half-life decay, where 50 percent of the resource has been decayed away within one half life. This ensures that steady, even users are not penalized by short-term, process-intensive users. The half-life decay period sets the responsiveness or "term" of the scheduler; the default value is 120 seconds. Shorter values tend to provide more even response across the system, at the expense of slightly less accuracy in computing and maintaining system-wide resource allocation. Regardless of administrative settings, the scheduler tries to prevent marooning (resource starvation) and ensure reasonable behavior, even in extreme situations.

The primary advantage of the Solaris Resource Manager scheduler over the standard Solaris scheduler is that it schedules users or applications rather than individual processes. Every process associated with an lnode is subject to a set of limits. For the simple case of one user running a single active process, this is the same as subjecting each process to the limits listed in the corresponding lnode. When more than one process is attached to an lnode, as when members of a group each run multiple processes, all of the processes are collectively subject to the listed limits. This means that users or applications cannot consume CPU at a greater rate than their entitlements allow, regardless of how many concurrent processes they run.The method for assigning entitlements as a number of shares is simple and understandable, and the effect of changing a user's shares is predictable.

Solaris Resource Manager will never waste CPU availability. No matter how low a user's allocation, that user will always be given all the available CPU if there are no competing users. One of the consequences of this is that users may notice performance that is less smooth than they are used to. If a user with a very low effective share is running an interactive process without any competition, it will appear to run quickly. However, as soon as another user with a greater effective share demands some CPU time, it will be given to that user in preference to the first user, so the first user will notice a marked job slow-down. Nevertheless, Solaris Resource Manager goes to some lengths to ensure that legitimate users are not marooned and unable to do any work. All processes being scheduled by Solaris Resource Manager (except those with a maximum `nice` value) will be allocated CPU regularly by the scheduler. There is also logic to prevent a new user that has just logged on from being given an arithmetically "fair," but excessively large proportion of the CPU to the detriment of existing users.

Virtual Memory (Per-User and Per-Process Limits)

Virtual memory is managed using a fixed resource model. The virtual memory limit applies to the sum of the memory sizes of all processes attached to the lnode. In addition, there is a per-process virtual memory limit that restricts the total size of the process's virtual address space size, including all code, data, stack, file mappings, and shared libraries. Both limits are hierarchical. Limiting virtual memory is useful for avoiding virtual memory starvation. For example, Solaris Resource Manager will stop an application that is leaking memory from consuming unwarranted amounts of virtual memory to the detriment of all users. Instead, such a process only starves itself or, at worse, others in its resource group.

Number of Processes

The number of processes that users may run simultaneously is controlled using a fixed resource model with hierarchical limits.

Terminals and Login Connect-Time

The system administrator and group header can set terminal login privileges, number of logins, and connect-time limits, which are enforced hierarchically by Solaris Resource Manager. As a user approaches a connect-time limit, warning messages are sent to the user's terminal. When the limit is reached, the user is notified, then forcibly logged out after a short grace period.

User Administration

The system administrator can set administrative privileges for any lnode, including assigning administrative privileges selectively to users. A user with hierarchical administrative privilege is called a sub-administrator. A sub-administrator may create, remove, and modify the lnodes of users within the sub-tree of which they are the group header.

Sub-administrators cannot normally alter their own limits or flags, and cannot circumvent their own flags or limits by altering flags or usages within their group.

The central administrator (or super-user) may alter the limits, usages and flags of any user, including itself. Ordinary users can be granted this privilege by the setting of a flag.

Usage Data Overview

The Solaris Resource Manager system maintains information (primarily current and accrued resource usage) that may be used by administrators to conduct comprehensive system resource accounting. No accounting programs are supplied as part of Solaris Resource Manager but its utility programs provide a base for the development of a customized resource accounting system.

For more information regarding setting up accounting procedures, see Chapter 8.

Examples

The examples in this section demonstrate Solaris Resource Manager functions used to control system resources and allocation, and to display information.

Server Consolidation Example

The first example illustrates these commands:

liminfo	Prints password attributes and limits information for one or more users to a terminal window
limadm	Changes limit attributes or deletes limits database entries for a list of users
srmuser	Displays or sets operation modes and system-wide Solaris Resource Manager tunable parameters
srmstat	Displays lnode activity information

Consider the case of consolidating two servers, each running a database application, onto a single machine. Simply running both applications on the single machine results in a working system; without Solaris Resource Manager, the Solaris operating system allocates resources to the applications on an equal-use basis, and does not protect one application from competing demands by the other application. However, Solaris Resource Manager provides mechanisms that keep the applications from suffering resource starvation. This is accomplished with Solaris Resource Manager by starting each database attached to lnodes referring to the databases, *db1* and *db2*. In order to do this, three new administrative placeholder users must be created, for example, *databases*, *db1*, and *db2*. These are added to the lnode database; since lnodes correspond to UNIX UIDs, these must also be added to the `passwd` file (or password map, if the system is using a name service such as NIS or NIS+). Assuming that the UIDs are added to the `passwd` file or password map, the placeholder users *db1* and *db2* are assigned to the *databases* lnode group with the command:

```
% limadm set sgroup=0 databases
% limadm set sgroup=databases db1 db2
```

which assumes that `/usr/srm/bin` is in the user's path.

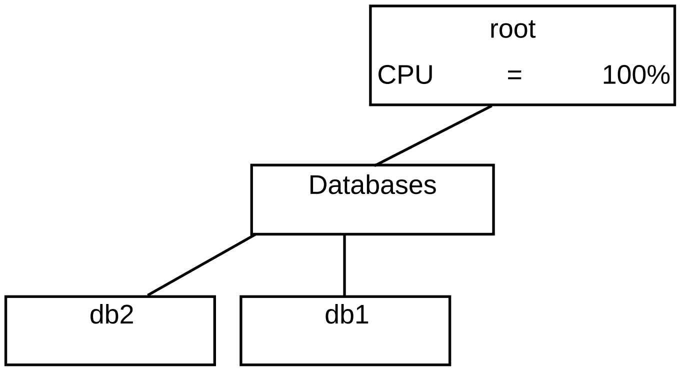

Figure 2–1 Server Consolidation

Because there are no other defined groups, the *databases* group currently has full use of the machine. Two lnodes associated with the databases are running, and the processes that run the database applications are attached to the appropriate lnodes with the `srmuser` command in the startup script for the database instances, for example:

```
% srmuser db1 /usr/bin/database1/init.db1
% srmuser db2 /usr/bin/database2/init.db2
```

When either database, *db1* or *db2*, is started up, use the `srmuser` command to ensure that the database is attached to the correct lnode and charged correctly (`srmuser` does not affect the ownership of the process to do this). To run the above command, a user must have the UNIX permissions required to run `init.db1` and the administrative permission to attach processes to the lnode *db1*. As users log on and use the databases, activities performed by the databases are accrued to the lnodes *db1* and *db2*.

By using the default allocation of one share to each lnode, the usage in the *databases* group will average out over time to ensure that the databases, *db1* and *db2*, receive equal allocation of the machine. Specifically, there is one share outstanding—to the *databases* group—and *databases* owns it. Each of the lnodes *db1* and *db2* are also granted the default allocation of one share. Within the *databases* group, there are two shares outstanding, so *db1* and *db2* get equal allocation out of *databases*' resources (in this simple example, there are no competing allocations, so *databases* has access to the entire system).

If it turns out that activity on Database1 requires 60 percent of the machine's CPU capacity and Database2 requires 20 percent of the capacity, the administrator can specify that the system provide at least this much (assuming that the application demands it) by increasing the number of `cpu.shares` allocated to *db1*:

```
% limadm set cpu.shares=3 db1
```

There are now four shares outstanding in the *databases* group; *db1* has three, and *db2* has one. This change is effected immediately upon execution of the above command. There will be a period of settling when the lnode *db1* (Database1) will actually receive more than its entitled 60 percent of the machine resource, as Solaris Resource Manager works to average the usage over the course of time. However, depending on the decay global parameter, it will not last long.

To monitor this activity at any point, use the commands `liminfo` and `srmstat`, in separate windows:

```
% liminfo -c db1
     # limit information shows all the data and
     # settings for the lnode db1.
```

See "A Typical Application Server" on page 23.

Alternatively, `srmstat` provides a regularly updating display:

```
% srmstat -ac    # srmstat shows the server activity and the
               # flag -ac sets a screen default update period
               # of 4 seconds to display the results.
```

You now have a machine running with two database applications, one receiving 75 percent of the resource and the other receiving 25 percent. Remember that the super-user (root) is the top-level group header user. Processes running as root thus have access to the entire system, if they so request. Accordingly, additional lnodes should be created for running backups, daemons, and other scripts so that the root processes cannot possibly take over the whole machine, as they might if run in the traditional manner.

Adding a Computational Batch Application User

This example introduces the following command:

srmkill	Kills all the active processes attached to an lnode

The Finance department owns the database system, but a user (Joe) from Engineering has to run a computational job and would like to use Finance's machine during off hours when the system is generally idle. The Finance department dictates that Joe's job is less important than the databases, and they agree to run his work only if it will not interfere with the system's primary job. To enforce this policy, add a new group (*batch*) to the lnode *database*, and add Joe to the new *batch* group of the server's lnode hierarchy.

```
% limadm set cpu.shares=20 databases
% limadm set cpu.shares=1 batch
% limadm set cpu.shares=1 joe
% limadm set sgroup=batch joe
```

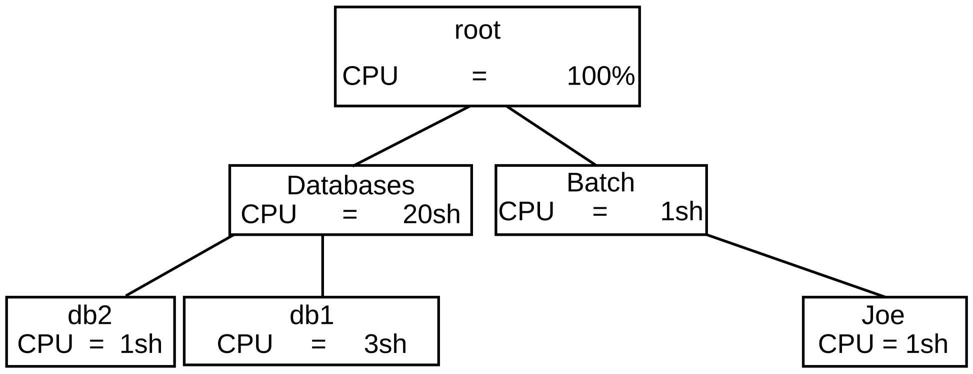

Figure 2–2 Adding a Computation Batch Application

This command sequence changes the allocation of shares so that the *databases* group has 20 shares, while the *batch* group has just one. This specifies that members of the *batch* group (only Joe) will use at most 1/21 of the machine if the *databases* group is active. The *databases* group receives 20/21, or 95.2 percent, more than the 60% + 20% = 80% previously determined to be sufficient to handle the database work. If the *databases* are not requesting their full allocation, Joe will receive more than his 4.8 percent allocation. If the *databases* are completely inactive, Joe's allocation might reach 100 percent. When the number of outstanding shares allocated to *databases* is increased from 1 to 20, there is no need to make any changes to the allocation of shares for *db1* and *db2*. Within the *databases* group, there are still four shares outstanding, allocated in the 3:1 ratio. Different levels of the scheduling tree are totally independent; what matters is the ratio of shares between peer groups.

Even with these assurances, the Finance department further wants to ensure that Joe is not even able to log on during the prime daytime hours. This can be accomplished by putting some login controls on the *batch* group. Since the controls are sensitive to time of day, this can be implemented by running a script that changes the number of logins allowed the *batch* group at the beginning and end of the day. For example, this could be implemented with `crontab` entries, such as:

```
0 6 * * * /usr/srm/bin/limadm set logins=0 batch
0 18 * * */usr/srm/bin/limadm set logins=100 batch
```

At 6:00 a.m., *batch*'s login limit is reduced to 0, and at 18:00 (6 p.m), the limit is raised to permit up to 100 logins.

An even stricter policy can be implemented by adding another line to the `crontab` entry:

```
01 6 * * * /usr/srm/bin/srmkill joe
```

This uses the `srmkill` command to kill any processes attached to the lnode Joe at 6:01 a.m. This will be unnecessary if the only resources that the job requires are those controlled by Solaris Resource Manager. This action could be useful if Joe's job could reasonably tie up other resources that would interfere with normal work. An example would be a job that holds a key database lock or dominates an I/O channel.

Joe can now log in and run his job only at night. Because Joe (and the entire *batch* group) has significantly fewer shares than the other applications, his application will run with less than 5 percent of the machine. Similarly, `nice(1)` can be used to reduce the priority of processes attached to this job, so it runs at lower priority than other jobs running with equal Solaris Resource Manager shares.

At this point, the Finance department has ensured that its database applications have sufficient access to their system and will not interfere with each other's work. They have also accommodated Joe's overnight batch processing loads, while assuring themselves that his work also will not interfere with their mission-critical processing.

Putting on a Web Front-end Process

Assume a decision has been made to put a web front-end on Database1, but limit this application to only 10 users at a time. Use the process limits function to do this.

First, create a new lnode called *ws1*. By starting the Webserver application under the *ws1*lnode, you can control the number of processes that are available to it, and hence the number of active `http` sessions.

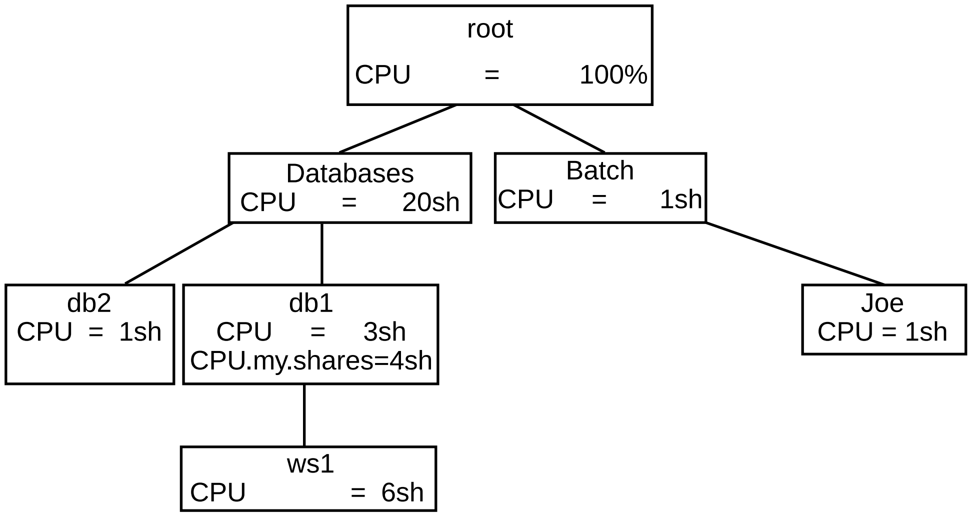

Figure 2–3 Adding a Web Front-end Process

Since Webserver is part of the Database1 application, you may want to give it a share of the *db1* lnode and allow it to compete with Database1 for resources. Allocate 60 percent of compute resources to the Webserver and 40 percent to the Database1 application itself:

```
# limadm set cpu.shares=6 ws1
# limadm set sgroup=db1 ws1
# limadm set cpu.myshares=4 db1
# srmuser ws1 /etc/bin/Webserver1/init.webserver
```

The last line starts up the Webserver and charges the application to the *ws1* lnode. Note that for Database1, the `cpu.myshares` have been allocated at 4. This sets the ratio of shares for which *db1* will compete with its child process, Webserver, at a ratio of 4:6.

Note - `cpu.shares` shows the ratio for resource allocation at the peer level in a hierarchy, while `cpu.myshares` shows the ratio for resource allocation at the parent:children level when the parent is actively running applications. Solaris Resource Manager allocates resources based on the ratio of outstanding shares of all active lnodes at their respective levels, where "respective level" includes the `my.shares` of the group parent and all children.

To control the number of processes that Webserver can run, put a process limit on the *ws1* lnode. The example uses 20 since a Webserver query will typically spawn 2 processes, so this in fact limits the number of active Webserver queries to 10:

```
# limadm set process.limit=20 ws1
```

Another application has now been added to the scheduling tree, as a leaf node under an active lnode. To distribute the CPU resource between the active parent and child, use `cpu.myshares` to allocate some portion of the available resource to the parent and some to the child. Process limits are employed to limit the number of active sessions on an lnode.

Adding More Users Who Have Special Memory Requirements

This example implements the resource control mechanisms CPU sharing, process limits, and login controls, and it addresses display tools for printing lnodes and showing active lnodes.

srmadm	Administer Solaris Resource Manager
limreport	Output information on selected users
limdaemon	Direct daemon to send messages when any limits are reached

Another user, Sally, has also asked to use the machine at night, for her application. Since her application is CPU-intensive, to ensure that Joe's application does not suffer, put a limit on Sally's usage of virtual memory, in terms of both her total usage and her "per-process" usage.

```
%  limadm set memory.limit=50M sally
%  limadm set memory.plimit=25M sally
```

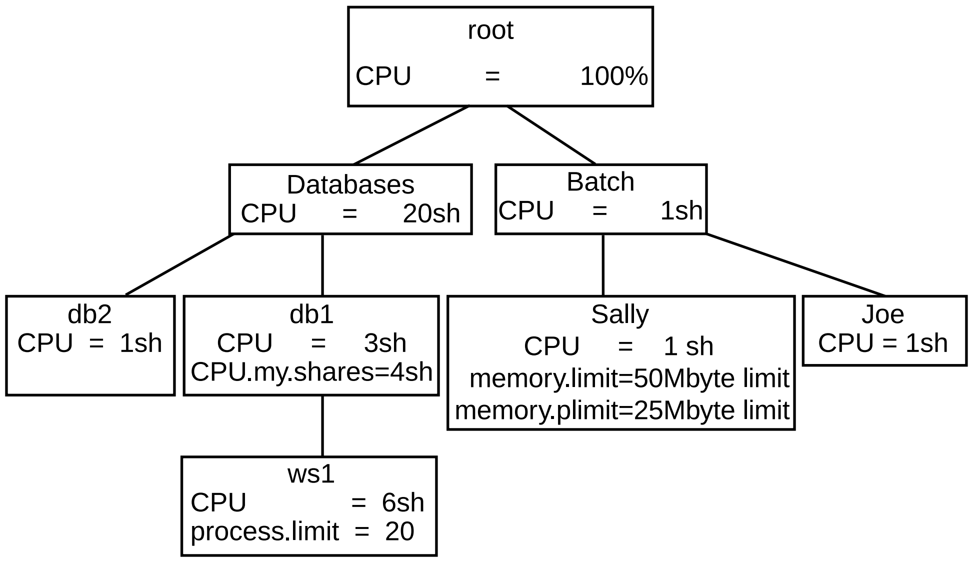

Figure 2–4 Adding More Users

If and when Sally's application tries to exceed either her total virtual memory limit or process memory limit, the `limdaemon` command will notify Sally and the system administrator, through the console, that the limit has been exceeded.

Use the `limreport(1MSRM)` command to generate a report of who is on the system and their usages to date. A typical use of `limreport` is to see who is using the machine at any time and how they fit within the hierarchy of users.

```
% limreport 'flag.real' - uid sgroup lname cpu.shares cpu.usage |sort +1n +0n
```

Note - `limreport` has several parameters, In this example, a check is made on "flag.real" (only looking for "real" lnodes/UIDs), then the dash (-) is used to indicate that the default best guess for the output format should be used, and the list "`uid sgroup lname cpu.shares cpu.usage`" indicates `limreport` should output these five parameters for each lnode with "flag.real" set to TRUE. Output is piped to a UNIX primary sort on the second column and secondary sort on the first column to provide a simple report of who is using the server.

Anyone with the correct path and permissions can check on the status of Solaris Resource Manager at any time using the command `srmadm show`. This will output a formatted report of the current operation state of Solaris Resource Manager and its main configuration parameters. This is useful to check that Solaris Resource Manager is active and all the controlling parameters are active. It also shows the values of global parameters such as the decay rate and location of the Solaris Resource Manager data store.

It is possible to run Solaris Resource Manager without limits active and without CPU scheduling active, which can be valuable at start up, for debugging and for initially configuring Solaris Resource Manager:

```
# srmadm set share=n:limits=n, -
```

Sharing a Machine Across Departments

A different development group would like to purchase an upgrade to this machine (more processors and memory) in exchange for having access to the system when it is idle. Both groups should benefit. To set this up, eestablish a new group called *development* at the same level as *databases* and *batch*. Allocate *development* 33 percent of the machine as they have added 50 percent more CPU power and memory to the original system:

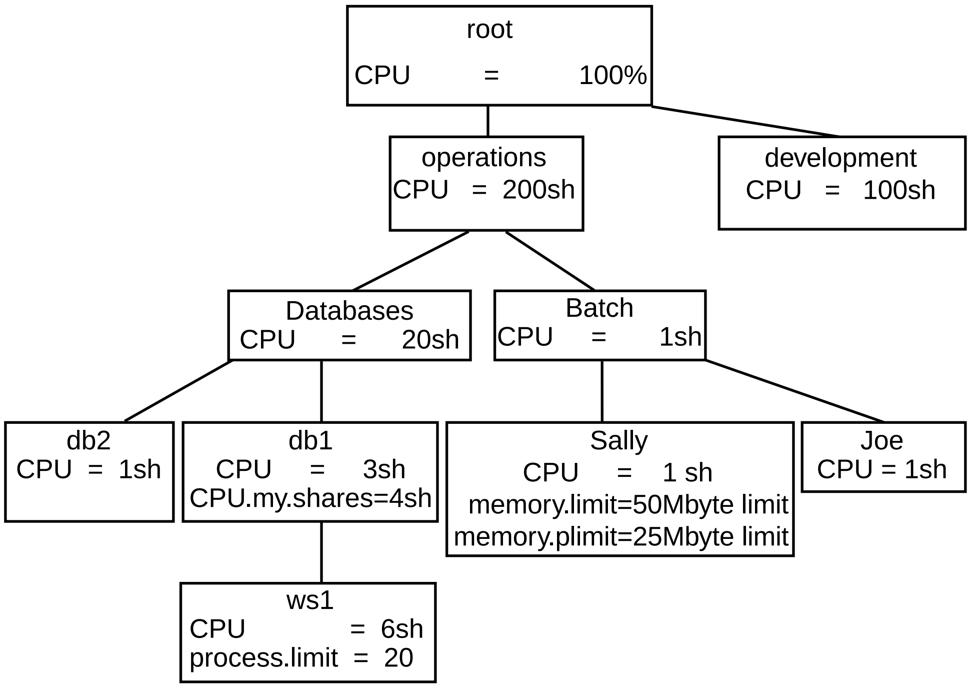

Figure 2–5 Sharing a Machine, Step 1

The Development group has hundreds of users. To avoid being involved in the distribution of their resource, use the administration flag capability of Solaris Resource Manager to enable the Development systems administrator to allocate their resources. You set up limits at the operations and development level as agreed jointly and then you both work to control your own portions of the machine.

To add the new level into the hierarchy, add the group *operations* as a new lnode and change the parent groups of *batch* and *databases* to *operations*:

```
% limadm set sgroup=operations batch databases
```

To set the administration flag:

```
% limadm set flag.admin=set operations development
```

Since under normal circumstances all servers have daemons and backup processes to be run, these should be added on a separate high-level lnode.

Note - Do not use root, since root has no limits.

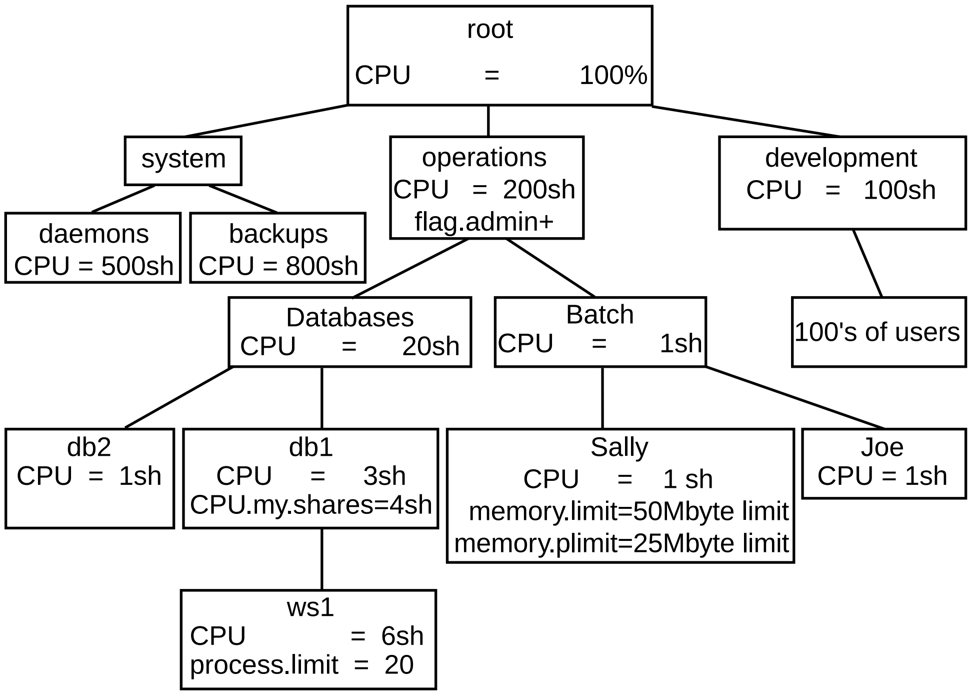

Figure 2–6 Sharing a Machine, Step 2

As seen in the examples, you can use Solaris Resource Manager to consolidate several different types of users and applications on the same machine. By the judicious use of CPU share controls, virtual memory limits, process limits, and login controls, you can ensure that these diverse applications receive only the resources that they need and require. The limits ensure that no application or user is going to adversely impact any other user's or group of users' application. Solaris Resource Manager supports some simple reporting tools to show users and systems administrators exactly what is happening at any given moment, and over the course of time. The report generation capability can be used to show the breakdown of resource utilization across applications and groups for capacity planning and billing purposes.

A Typical Application Server

This output would be displayed from a `liminfo` listing of *db1* at the end of the example in the previous section. Typing:

```
# liminfo -c db1
```

Produces:

```
# ./liminfo  db1
Login name:             db1      Uid(Real,Eff):    223 (223,223)
Sgroup (uid) :    other (98)     Gid(Real,Eff):       50 (50,50)

Shares:                 3        Myshares:                     6
Share:              60.0%        E-share:                  35.4%
Usage:              76000        Accrued usage:          6.4e+08

Mem usage:          11.06 B      Term usage:                  0s
Mem limit:              0 B      Term accrue:                 0s
Proc mem limit:         0 B      Term limit:                  0s
Mem accrue:         13.67 TB.s

Processes:              8        Current logins:               1
Process limit:          0

Last used:  Tue Jul 4 15:04:20 1998
Directory:  /usr/people/db1
Name:       Database1
Shell:      /usr/sh/

Flags:  userlimadm+
```

Figure 2–7 `liminfo` Listing

Refer to `liminfo(1SRM)` for more information on the fields described below.

The first two lines of output from the `liminfo(1SRM)` command relate to aspects of the lnode UID and its position in the lnode tree.

Login name	The login name and initial GID from the password map that corresponds to the UID of the attached lnode. Every lnode is associated with a system UID. It is strongly recommended that a system account be created for the UID of every lnode. In this instance a placeholder UID is used for the Database1 of *db1*.

	Note that the default PAM configuration under Solaris Resource Manager creates an lnode for any user who logs in without one. By default, lnodes created by `root` or by a user with the `uselimadm` flag set are created with the lnode for the user other as their parent, or if that does not exist, with the root lnode as their parent. Lnodes created by a user with the administration flag set are created with that user as their parent. The parent of an lnode can be changed with the general command for changing lnode attributes, `limadm`.
Uid	The UID of the lnode attached to the current process. Normally, this will be the same as that of the real UID of the process (the logged in user), but in some circumstances (described later) it may differ.
R,Euid and R,Egid	The real and effective UID and GID of the current process. This is the same information that is provided by the standard system `id(1M)` command. It is not strictly related to Solaris Resource Manager, but it is displayed for convenience. These fields are not displayed if `liminfo(1SRM)` is displaying information on a user other than the default (that is, it was provided with a login name or UID as an argument).
Sgroup (uid) [sgroup]	The name and UID of the parent lnode in the lnode tree hierarchy. This will be blank for the root lnode. Many Solaris Resource Manager features depend on the position of an lnode within the tree hierarchy, so it is useful for a user to trace successive parent lnodes back to the root of the tree.

After the blank line, the next two lines of the `liminfo(1SRM)` display show fields relating to CPU scheduling.

Shares [cpu.shares]	This is the number of shares of CPU entitlement allocated to this user. It is only directly comparable to other users with the same parent lnode, and to the "Myshares" value of the parent lnode itself. Administrators might normally set the shares of all users within a particular

	scheduling group to the same value (giving those users equal entitlements). This value will normally be something greater than 1, so that administrators have some leeway to decrease the shares of specific users when appropriate.
Myshares [cpu.myshares]	This value is only used if this user has child lnodes (that is, if there are other lnodes that have an `sgroup` value of this user) that are active (that is, have processes attached). Where this is the case, this value gives the relative share of CPU for processes attached to this lnode, compared with those attached to its child lnodes.
Share	The calculated percentage of the system CPU resources to which the current user is entitled. As other users log on and log off (or lnodes become active or inactive), this value will change, because only active users are included in the calculation. Recent usage by the current user is not included in this calculation.
E-Share	This is the effective share of this user (that is, the actual percentage of the system CPU resources which this user would be given in the short term if the user required it and all other active users were also demanding their share). It can be thought of as the current willingness of Solaris Resource Manager to allocate CPU resources to that lnode. This value will change over time as the user uses (or refrains from using) CPU resources. Lnodes that are active but idle (that is, with attached processes sleeping), and so have a low usage, will have a high effective share value. Correspondingly, the effective share can be very small for users with attached processes that are actively using the CPU.
Usage [cpu.usage]	The accumulated usage of system resources that are used to determine scheduling priority. Typically, this indicates recent CPU usage, though other parameters may also be taken into account. The parameter mix used can be viewed with the `srmadm(1MSRM)` command. Each increment to this value decays exponentially over time so that eventually Solaris Resource Manager will "forget" about the resource usage. The rate of this decay is

	most easily represented by its half-life, which can be seen with the `srmadm(1MSRM)` command.
Accrued usage [cpu.accrue]	This is the same resource accumulation measurement as "Usage," but it is never decayed. It is not used directly by Solaris Resource Manager but may be used by administration for accounting purposes. Unlike usage, this value represents the sum of the accrued usages for all lnodes within the group, as well as that of the current lnode.

After the second blank line, the next four lines of the `liminfo(1SRM)` display show four fields relating to virtual memory:

Mem usage [memory.usage][memory.myusage]

This is the combined memory usage of all processes attached to this lnode.

If two values are displayed, separated by a frontslash (`/`) character, then this lnode is a group header and the first value is the usage for the whole scheduling group, while the second value is that of just the current user.

Mem limit [memory.limit]

The maximum memory usage allowed for all processes attached to this lnode and its members (if any). That is, the sum of the memory usage for all processes within the group plus those attached to the group header will not be allowed to exceed this value. Note that in this instance, a "0" value indicates that there is no limit.

Proc mem limit [memory.plimit]

The per-process memory limit is the maximum memory usage allowed for any single process attached to this lnode and its members.

Mem accrue [memory.accrue]

The memory accrue value is measured in byte-seconds and is an indication of overall memory resources used over a period of time.

After the third blank line, the next four lines of the `liminfo(1SRM)`display show fields relating to the user and processes.

Processes [process.usage][process.myusage]

This is the number of processes attached to this lnode. Note that this is processes, not counting threads within a process.

If two values are displayed, separated by a frontslash (/) character, then this lnode is a group header and the first value is the usage for the whole scheduling group, while the second value is that of just the current user.

Process limit [process.limit]

The maximum total number of processes allowed attached to this lnode and its members.

Current logins [logins]

Current number of simultaneous Solaris Resource Manager login sessions for this user. When a user logs in through any of the standard system login mechanisms (including `login(1)`, `rlogin(1)`, etc.) basically anything that uses PAM for authentication and creates a `utmp(4)` entry) then this counter is incremented. When the session ends, the count is decremented.

If a user's `flag.onelogin` flag evaluates to set, the user is only permitted to have a single Solaris Resource Manager login session.

Last used [lastused]

This field shows the last time the lnode was active. This will normally be the last time the user logged out.

Directory

The user's home directory (items from the password map rather than from Solaris Resource Manager are shown for convenience).

Name

The *db1* (finger) information, which is usually the user's name (items from the password map rather than from Solaris Resource Manager are shown for convenience).

Shell

The user's initial login shell (items from the password map rather than from Solaris Resource Manager are shown for convenience).

Flags

Flags that evaluate to set or group in the lnode is displayed here. Each flag displayed is followed by suffix characters indicating the value and the way in which the flag was set (for example, whether it was explicitly from this lnode (+) or inherited (^)).

CHAPTER **3**

Configuration

Solaris Resource Manager provides a great deal of flexibility in its configuration to the central administrator (for example, the root user). This chapter describes the following configuration areas:

- Kernel boot parameters, used when the kernel is first started (see "Kernel Boot Parameters" on page 29).
- Global Solaris Resource Manager parameters supplied via the `srmadm(1MSRM)` command (see "Global Solaris Resource Manager Parameters via `srmadm`" on page 32).
- Parameters given to the `limdaemon(1MSRM)` program (see "Using `limdaemon` Options" on page 34).
- PAM subsystem and account management (see "PAM Subsystem" on page 35).

Kernel Boot Parameters

The kernel has certain Solaris Resource Manager parameters which can be set by the central administrator when the kernel is booted. Solaris reads the `/etc/system` file at boot time and uses it to configure kernel modules (see `system(4)` for details). The parameters that can be set in the SHR module (all are 32–bit integers) to override the Solaris Resource Manager default behavior are:

SRMLnodes	The number of lnodes to cache in the kernel. On Solaris systems, each kernel lnode requires about 3Kb. A value of zero (the default) means that the

kernel will determine the value. The heuristic then used is:

```
(nproc / SRMProcsPerUid ) + SRMLnodesExtra
```

where *nproc* is the maximum number of simultaneous processes allowed in the system. A minimum value of 6 overrides this calculation. The maximum specified by SRMMemoryMax will also override this calculation.

SRMProcsPerUid	The expected average number of processes used by each user. The default is 4.
SRMLnodesExtra	A bias used in the heuristic to determine the size of the in-memory lnode array. The default is 20.
SRMNhash	The number of entries in the hash table that is used to map UID values to lnodes in the kernel. On Solaris each entry is 4 bytes long. The default is zero, which means to use the same value as for the number of lnodes.
SRMMemoryMax	The reciprocal of this value is a fraction which specifies the maximum percentage of real memory to use for the Solaris Resource Manager lnode and hash tables combined. The default is 20, which means that a maximum of 5 percent of real memory will be used for Solaris Resource Manager data structures.
SRMMemWarnFreq	The minimum interval, in seconds, between "memory exceeded" notification warnings for a single lnode. The default value is 4.

For example, in the `/etc/system` file the line

```
set srmlim:SRMMemWarnFreq=10
```

will ensure that memory exceeded messages are sent no more frequently than one every 10 seconds for any single user.

There are also some parameters not in Solaris Resource Manager which affect the behavior of Solaris Resource Manager. These include:

initclass	This is the name of the scheduling class in which the `init(1M)` process is started. Under Solaris Resource Manager this should be given as the string "SHR" (including the double-quote characters). The default Solaris value is "TS." To use Solaris Resource Manager for CPU resource control, the following line would be included in the `/etc/system` file: `set initclass="SHR"` to override the default.
extraclass	This is a name of a scheduling class module to load, without necessarily using it as the default scheduling class. To use Solaris Resource Manager with just non-CPU resource control, the following line would be included in the `/etc/system` file: `set extraclass="SHR"`

To boot a system without Solaris Resource Manager loaded at all, an alternate `/etc/system` named `/etc/system.noshrload` is used. See “Booting Without Solaris Resource Manager” on page 41 for instructions on this process.

Multi-User Startup Configuration

During a normal system boot, when the system changes from single-user to multi-user mode, a Solaris Resource Manager initialization script is run to set various Solaris Resource Manager parameters. Details of what this script does are given in Chapter 4.

If the initialization script itself (`/etc/init.d/init.srm`) is modified, a copy of both the original and modified versions should be kept separately. Applying Solaris Resource Manager updates will not necessarily preserve existing initialization scripts.

Global Solaris Resource Manager Parameters via `srmadm`

The `srmadm(1MSRM)` command allows an administrator to set, modify or display the global Solaris Resource Manager parameters. Refer to the man page for complete details of all parameters.

The `srmadm(1MSRM)` command can be called any number of times to set various parameters. It is not necessary to include them all on a single invocation. This also means that `srmadm(1MSRM)` can be used to change the operational parameters of a running Solaris Resource Manager system on the fly, although some caution should be taken.

Of particular importance to administrators are the `srmadm(1MSRM)` options which enable or disable the main features of Solaris Resource Manager. These are:

`fileopen[={y|n}]`

The default database is `/var/srm/srmDB` and it can be overrided with the `–f` option. Note that closing the Solaris Resource Manager 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 CPU scheduler is disabled; Solaris Resource Manager limit enforcement ceases. When enabled, 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, the Solaris Resource Manager CPU 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, Solaris Resource Manager CPU scheduler’s usage calculations are frozen, and processes are scheduled ’round-robin’ with fixed equal priorities.

`limits[={y|n}]`

When enabled, 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, the Solaris Resource Manager CPU 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 entitlements of CPU service whenever possible, regardless of the actions of their members.

`limshare[={y|n}]`

When enabled, the Solaris Resource Manager CPU 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 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 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.

There are two optional parameters to `srmadm(1MSRM)` which are also useful to an administrator. These are:

- The `–v` parameter. This prints a formatted report of all current parameter settings on standard output. If two or three `–v` options are supplied, then the report is more and more verbose. Invoking `srmadm(1MSRM)` with no arguments is equivalent to supplying a single `–v` option.
- The `–d` parameter. This initializes the Solaris Resource Manager system structure with default values instead of reading the current kernel settings. The default values, which mainly give control over scheduling behavior, are built into `srmadm(1MSRM)`, and provide a good starting point from which to customize Solaris Resource Manager. The kernel begins with the same values preset.

The following are examples of typical `srmadm(1MSRM)` commands.

To turn on Solaris Resource Manager, enabling the CPU scheduler and resource limits:

```
# srmadm set -f /var/srm/srmDB fileopen=y:share=y:limits=y
```

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

```
# srmadm set usagedecay=300s
```

To display the current flag settings and charges:

```
% srmadm
```

To show all the default settings:

```
% srmadm show -dv
```

Disabling Solaris Resource Manager

The `srmadm(1MSRM)` command can disable Solaris Resource Manager by clearing the fileopen flag; all processes are moved onto the surrogate root lnode, other changed lnodes in the cache are flushed to disk and the lnode file is closed. This automatically forces the share and limits flags off, disabling the Solaris Resource Manager CPU scheduler and limit enforcement respectively. The share and limits flags may be turned off independently if required while leaving the lnode file open. This is preferable to closing the file, as processes can stay attached to their correct lnodes.

Note that if the Solaris Resource Manager scheduler alone is disabled in mid-operation, all this does is suspend the usage and decay algorithm. The scheduler still continues handling processes in the SHR scheduling class, but as each is assigned an updated priority, the same value is used resulting in simple "round-robin" scheduling.

Re-enabling Solaris Resource Manager by opening the file and setting the share and/or limits flags after the file has been closed will not cause existing processes to move off the root lnode. Closing the Solaris Resource Manager database during normal operation is not recommended. If this is done, the system should be rebooted in order to ensure correct attachment of processes to lnodes.

Using `limdaemon` Options

`limdaemon(1MSRM)` has several options which can be configured when it is started:

- The `-l` option will cause `limdaemon(1MSRM)` to log messages via `syslog(3)` -
- The `-m` tag and `-p` prio options are used in conjunction with `-l` to tag the messages and control message routing according to the `syslogd(1M)` configuration
- The `-s` option will cause `limdaemon(1MSRM)` to include a time-stamp on messages (except those via `syslog(3)` which already have a time-stamp)
- The `-c` option will cause `limdaemon(1MSRM)` to suppress the updating of terminal connect-time usages

- The `–d` option will cause `limdaemon(1MSRM)` to decay connect-time usages for all terminals of logged in users, with the interval between decays being the argument of the `–t` option (default 1 minute)
- The `–D`*n* option will cause `limdaemon(1MSRM)` to decay connect-time usages for all terminals all users once every *n* minutes
- The `–k` option will terminate the currently running `limdaemon(1MSRM)`
- The `–t` option can be used to set the time period (in minutes) between updates to the connect-time usage attribute in the terminal device category. The default is 1 minute
- The `–e` option can be used to suppress the logging off of users who have reached their connect-time limit. This option is implied by the use of the `–c` option
- The `–w` option sets the number of minutes before expiration of connect-time that the warning message is given. The default warning interval is 5 minutes
- The `–g` option can be used to set the grace time (in seconds). The default grace time is 30 seconds

The administrator should determine the balance needed between the additional overhead incurred for rapid updating of connect-time usage attributes, and the greater granularity that will appear with less frequent updating. See the `limdaemon(1MSRM)` man page for more information on these and other options.

As an example, the command:

```
% limdaemon -g300
```

starts the daemon and sets the grace time to five minutes. Note that it is not necessary to follow the command with a shell ’&’ character. When `limdaemon` is started, it makes itself into a daemon. That is, a child process is forked which detaches itself from the controlling terminal, placing itself in a process group of its own.

PAM Subsystem

Solaris 2.6 systems support Pluggable Authentication Modules (PAM). Whenever a user requests an operation that involves changing or setting the user’s identity (such as logging into the system, invoking an ’r’ command such as `rcp` or `rsh`, using `ftp`, or using `su`), a set of configurable modules are used to provide authentication, account management, credentials management, and session management. Solaris Resource Manager provides a module for login accounting, and to modify the behavior of `su`.

The program used to request the operation is termed the service.

The PAM system as a whole is documented in the man pages `pam.conf(4)`, `pam(3)`, `pam_unix(5)`, and `pam_srm(5SRM)`.

The Solaris Resource Manager PAM module provides account management and session management functions. Behavior of PAM can be controlled by editing the file `/etc/pam.conf`. For normal Solaris Resource Manager behavior, the Solaris Resource Manager PAM module should be configured as requisite for all login-like services for session management, and as requisite for account management for all PAM services. Usually, the Solaris Resource Manager module should be placed after all other required and requisite modules, and before any other sufficient or optional modules.

On installation, Solaris Resource Manager edits `/etc/pam.conf` to provide a suitable behavior. It inserts lines like these for each service (including `other`) that already has session or account management configured:

```
login account requisite pam_srm.so.1 nolnode=/etc/srm/nolnode
login session requisite pam_srm.so.1
other account requisite pam_srm.so.1 nolnode=/etc/srm/nolnode
```

The first line says that for service login, the module `pam_share.so.1` is to be used to provide account management functionality, that it must allow login if login is to succeed, and it is to be given the argument `nolnode=/etc/srm/nolnode`. See `pam.conf(4)` for a full explanation of the various control flags (`required, requisite, optional,` and `sufficient`).

The second line says that the login service will use the `pam_share.so.1` module for session management.

The full list of supported arguments for Solaris Resource Manager account and session management modules is found in `pam_srm(5SRM)`.

Account Management

When the Solaris Resource Manager account management PAM module gets control, it:

1. determines if Solaris Resource Manager is installed and enabled, and tells the PAM system to ignore this module if not,
2. determines if the user has an lnode, and calls an administrator-configurable 'no lnode' script if not,
3. determines whether the user has permission to use the requested service and device,

4. determines if the user has exceeded the warnings limit, and refuses permission to log in if so,
5. calls an administrator-configurable ’every login’ script.

If any of these steps fail, the remainder are not performed, and the Solaris Resource Manager account management PAM module denies use of the service. An explanatory message is passed to the user via the service where possible.

Scripts

The default ’missing lnode’ script will create an lnode for the user and send mail notifying the system administrator that it has done so. The default script is `/etc/srm/nolnode`, but this can be changed by editing the file `/etc/pam.conf` and changing the value of the nolnode option on Solaris Resource Manager account management module lines. The ’every login’ script is not usually configured. It can be configured by adding an everylogin=pathname option to any Solaris Resource Manager account management module in `/etc/pam.conf`. Scripts are invoked as the root user. Standard input, output and error are closed. If a script exits non-zero, access will be denied. All information is passed as environment variables, which are derived directly from information passed to PAM from the service:

`USER`	The login name supplied to the program. It has been authenticated by looking it up in the password map; if not present, the account management module will already have returned an error code to PAM.
`UID`	The UID of the user being authenticated. For services that change UID (such as su) this is the UID of the user invoking the service; for services that set UID (such as login) this is the target UID (i.e., that of USER).
`RHOST`	For access attempts across a network, this contains the name of the host the attempt comes from. Its value is otherwise implementation dependent.
`SERVICE`	The name of the access service, for example, `rsh`, `login`, `ftp`, etc.
`TTY`	The name of the TTY that the service is being invoked on. Some services that do not (strictly speaking) have a controlling terminal (such as ftp), will fill this variable with something vaguely sensible (for example, `ftp12345`, where *12345* is

	the process identifier (PID) of `ftpd`); others leave it empty or replace it with the service name.
`DEBUG`	If debug was specified in the `pam.conf` file, DEBUG is set to true; otherwise it is set to false. No other environment variables are set, so any script must set its own PATH variable if required.

The default 'no lnode' script creates the lnode in the default scheduling group (`other` if such a user exists in the password map, otherwise `root`) and mails the system administrator a reminder to move the new lnode into the appropriate place in the scheduling hierarchy. For a sample script, see "Default 'no lnode' Script" on page 91.

PAM Interaction With Device Groups

The Solaris Resource Manager PAM module looks up the terminal and service names in the device hierarchy, and returns a 'permission denied' to its invoker if limits are exceeded or if a device flag evaluates to '`set`'.

The device categories examined are `terminal` for the terminal name, and `services` for the kind of service requested. For example, an rlogin attempt may try to use a file in the network device group, so the flags tested for the user are (assuming all flags are set to `group`) as shown below. These flags are checked in order:

- `terminal.flag.network`
- `terminal.flag.all`
- `services.flag.rlogin`
- `services.flag.netservices`

Access will be permitted only if they all evaluate 'set'. In addition, limits will be checked for the corresponding categories (`terminal` and `services`).

Session Management

For login-like services (those that create an entry in the `utmp` file), the session management facilities of PAM will be invoked as well as the account management facilities if both are configured in `/etc/pam.conf`.

The Solaris Resource Manager product's session management handles charging for devices. It looks to see if the user has exceeded the connection time limit, or has the `onelogin` flag evaluate to set and is already logged in, and if so, prevents login.

Otherwise, it generates a message to the `limdaemon` process to inform it of the login and the configured cost for the terminal being logged in on. It then informs the kernel that the current process is a 'login header process', that the `limdaemon` process must be informed of when it dies.

The `limdaemon` process then tracks connect-time limits, and issues warnings if they are about to be exceeded.

CHAPTER **4**

Boot Procedure

During the UNIX boot procedure, various facets of Solaris Resource Manager are enabled at different points. The major steps are shown here:-

- When the kernel first starts, various parameters are loaded from the `/etc/system` file. Some of these affect Solaris Resource Manager. These are documented in detail below.
- As the kernel continues its initialization, after process 0 has been created, but before process 1 is started, Solaris Resource Manager is initialized. This involves loading the SHR module and arranging for process 1 (the `init(1M)` process) to be scheduled by Solaris Resource Manager. This is done by starting `init(1M)` in the SHR scheduling class, instead of the default scheduling class.
- Initially the `init(1M)` process, and all its children, will be attached to the surrogate root lnode.
- When the kernel is fully initialized, the system will undergo transition from single-user to one of the multi-user modes (usually run-level 2 or 3). Early in this procedure, the `/etc/init.d/init.srm` script will be run. The actions performed by this script are described in "Boot Sequence Events" on page 42 and enable normal Solaris Resource Manager operations.

Booting Without Solaris Resource Manager

If it is necessary to boot the system without Solaris Resource Manager active, this can be done easily by changing the *initclass* variable in the `/etc/system` file to refer to timesharing (TS) instead of SHR. A simple way of doing this is to use the `–a` (ask) option of the `boot(1M)` command, so that you will be prompted for a system file.

For other prompts, just press the RETURN key to accept the default values, until you are prompted for the name of the system file. At the prompt for the name of the system file, enter `etc/system.noshrload` (no leading slash) as the response. Here is an example of the procedure:

```
ok boot -a
Booting from: sd(0,0,0) -a
Enter filename [kernel/unix]:
Enter default directory for modules
 [/platform/SUNW,UltraSPARC/kernel /kernel /usr/kernel]:
SunOS Release 5.6 Version ... [UNIX(R) System V Release 4.0]
Copyright (c) 1983-1997, Sun Microsystems, Inc.
Name of system file [etc/system]: etc/system.noshrload
root filesystem type [ufs]:
Enter physical name of root device
 [/sbus@1,f8000000/esp@0,800000/sd@3,0:a]:
```

Note that `/etc/system.noshrload` is simply a backup copy of `/etc/system` made at the time Solaris Resource Manager was installed. If there have been subsequent edits to `/etc/system`, then `/etc/system.noshrload` should be maintained in parallel so that it differs only by the occurrence of the Solaris Resource Manager modification:

```
# diff /etc/system /etc/system.noshrload
< # enable srm
< set initclass="SHR"
```

Boot Sequence Events

The sequence in which events occur after the system boot while switching to multi-user mode is particularly important in Solaris Resource Manager. The following steps show a sequence which correctly establishes the Solaris Resource Manager system:

1. Configure and enable Solaris Resource Manager using the `srmadm(1MSRM)` command. At this point, the limits database will be opened and the Solaris Resource Manager scheduler will be enabled. See “Enabling Solaris Resource Manager Using `srmadm`” on page 44 for information on using `srmadm(1MSRM)` to enable Solaris Resource Manager.
2. Assign the ’lost’ and ’idle’ lnodes.

3. Start the Solaris Resource Manager daemon. See “Starting the Solaris Resource Manager Daemon” on page 45 for information on this procedure.
4. Start other system daemons on an appropriate lnode.

The default script used in steps 1 through 3 of the above process is shown in the appendix.

System Daemon Processes

Of particular importance is the attachment of daemons (system maintenance processes which normally run permanently) to an lnode other than the root lnode. Processes attached to the root lnode are scheduled specially and will always be given all the CPU resources they demand, so it is not advised to attach to the root lnode any process which is potentially CPU-intensive. Attaching daemons to their own lnode allows the central administrator to allocate them a suitable CPU share.

During the boot procedure, each new process inherits its lnode attachment from its parent process. Since the `init(1M)` process is attached to the root lnode, so are all subsequent processes. It is not until the Solaris Resource Manager initialization script is run and the lnode database is opened that processes can be attached to other lnodes, and even then this only happens when a process does an explicit `setuid(2)` system call (such as `login(1)` does) or explicitly asks Solaris Resource Manager to attach to a nominated lnode, such as the `srmuser(1SRM)` command does. Running a program with the `setuid` file mode bit set does not cause a change in lnode attachment.

The consequence of this is that all system programs started automatically during system startup will be attached to the root lnode. This is often not desirable, as any process attached to the root lnode that becomes CPU intensive will severely disrupt the execution of other processes. Therefore it is recommended that any daemon processes started as part of the boot procedure be explicitly attached to their own lnode by using the `srmuser(1SRM)` command to invoke them. This will not affect their real or effective UID.

A possible example is shown here:

```
# /usr/srm/bin/srmuser Start in.named attached to the my_daemons lnode.
```

These lines could be used to replace the existing invocation of the `named(1M)` daemon in its start-up script. This requires that a user account and lnode for "network" should be established beforehand.

Enabling Solaris Resource Manager Using `srmadm`

The `srmadm(1MSRM)` command allows the administrator to control the operating state and system-wide configuration of the Solaris Resource Manager system. This command is typically used during transition to run-level 2 or 3 from within the Solaris Resource Manager `init.d(4)` script `/etc/init.d/init.srm` to ensure that appropriate values for all parameters are set each time the system is booted, and to ensure that the Solaris Resource Manager system will be enabled prior to users having access to the system. The `srmadm(1MSRM)` command is also used to administer the global Solaris Resource Manager parameters. Refer to the `srmadm(1MSRM)` manual page for a list of the parameters that may be set using `srmadm`. The `srmadm(1MSRM)` commands issued in the Solaris Resource Manager `init.d(4)` script will:

- Open the limits database. Up until this point, any processes which are started are attached automatically to a surrogate root lnode. The surrogate root lnode is used to ensure that there is always an lnode available to connect processes to, regardless of the operational state of Solaris Resource Manager. For this reason, it is important that the limits database be opened before any non-root processes are started. When the limits database is opened, the values in the usage attributes in the surrogate root lnode are added into their counterparts in the real root lnode. A limitation of this technique is that any net decrease in usage will not be counted. This ensures that usage alterations prior to the limits database being opened are not discarded.
- Enable limit enforcement.
- Set the parameters which control the behavior of the Solaris Resource Manager SHR scheduler, for example, the usage decay rate.
- Enable the Solaris Resource Manager scheduler. Prior to this, processes in the SHR scheduling class are scheduled in a simple round-robin fashion and the CPU entitlements set within the Solaris Resource Manager system have no effect.

Refer to “Global Solaris Resource Manager Parameters via `srmadm`” on page 32 for some of the common invocations of the `srmadm(1MSRM)` command.

Starting the Solaris Resource Manager Daemon

The `limdaemon(1MSRM)` program is the Solaris Resource Manager user-mode daemon. It is normally invoked at transition to run-level 2 or 3 as the last step in the Solaris Resource Manager `init.d(4)` script. It shouldn't be confused with the `srmgr` system process (in the SYS class), initiated by the kernel. The following `ps(1)` listing shows both these processes:

```
# ps -efc | egrep 'limdaemon|srmgr'
root    4    0  SYS  60 18:42:14 ?      0:05 srmgr
root   92    1  SHR  19 18:42:32 ?      0:41 limdaemon
```

The `limdaemon` program performs the following functions:

- receives notification messages and delivers them to the terminals of destination users
- receives login or log out notification messages, maintaining an exact record of all Solaris Resource Manager login sessions currently in progress
- periodically updates the connect-time usages for all users with Solaris Resource Manager login sessions currently in progress (optional)
- detects users who have reached their connect-time limit and kills the process and logs them off (optional) after a grace interval
- logs all actions using `syslog(3)` to `syslogd(1M)`

When notified of Solaris Resource Manager login sessions, `limdaemon` monitors the terminal connect-time of all users and checks it against their connect-time limit. When their connect-time limit is nearly reached, they are sent a notification message. Once the expiration time is reached, a further grace time is allowed before all their processes are terminated and they are logged out.

The `limdaemon` program decays connect-time usages. Usage decay for the terminal device category must be performed, if connect-time limits are used. Refer to "Using `limdaemon` Options" on page 34 for details on the command-line options that can be used to control the behavior of `limdaemon`.

CHAPTER **5**

Managing Lnodes

The Solaris Resource Manager system is built around a fundamental addition to the kernel: a per-user structure called an lnode. For every unique UID defined in the password map, there should exist a corresponding lnode. (This is every unique UID returned by successive `getpwent(3C)` calls.) An lnode may exist without a corresponding password map entry, but this is not recommended. lnodes are stored on disk and automatically moved in and out of memory by the kernel. In-memory copies of lnodes that have been changed since reading them from disk are written back as part of the regular system synchronization operations, as well as on demand when the `sync(1M)` command is run, and when necessary to free space in the lnode cache for reading in further lnodes. An lnode is essentially a fixed-size place in which many kinds of per-user data may be stored and updated.

lnodes are maintained as a tree hierarchy, with the central administrator as the head of the tree, and other users as group headers of smaller groups of users within the tree. The central administrator is the super-user, or root user of the system.

Errors relating to lnodes, such as orphans and group loops, are discussed in Chapter 9.

Delegated Administration

The prime responsibility for the administration of lnodes rests with the central administrator. While Solaris Resource Manager introduces several resource controls that may be assigned and managed, it also allows certain administrative privileges to be selectively assigned to non-root users, thereby distributing the burden of user administration. Administrative privileges may be assigned to appropriate users by setting the user's *uselimadm* or *admin* flag. Users with a set *uselimadm* flag have the same administrative privilege within the `limadm(1MSRM)` program as the super-user. A group header user with a set *admin* flag is called a sub-administrator, and has privileges (as described below) over users within their scheduling group.

The central administrator controls the overall division of the system's resources by creating and assigning limits to scheduling groups who have root as their parent. Sub-administrators typically perform the same types of resource control, but limited to users within their scheduling group. The division of resources by the sub-administrator is limited to the resources that have been allocated to the group (for example., those allocated to the group header lnode). Note that a sub-administrator may assign an admin flag to any user in their scheduling group, further sub-dividing their own administrative responsibilities.

Sub-administrators may do the following:

1. Create and delete lnodes for users within their scheduling group.
2. Alter the resource limits of any user within their scheduling group.

   Note that even though a sub-administrator may set the limit of a resource to be greater than that of the limit for the group, resources consumed by group members are also considered to be consumed for group headers and limits on individual users will be enforced when an attempt is made to exceed the group header limit.
3. Alter the flags of any lnode within their scheduling group, provided that the flag does not have the `noadmin` condition. Flag assignments by sub-administrators are further constrained in that a user cannot be given a privilege that is not already held by the sub-administrator. This restriction is applied to prevent a sub-administrator from circumventing the security within Solaris Resource Manager.
4. Adjust any of their own attributes that have the `selfadmin` condition.

A sub-administrator's main tools are the `limadm(1MSRM)` and `limreport(1SRM)` commands. The `limadm` program performs operations on the limits, flags, and other Solaris Resource Manager attributes of one or more existing users. Combined with the report generator, `limreport`, these tools allow a scheduling group to be autonomously self-managed without disturbing the resource allocations or management of other, disjoint scheduling groups.

The super-user is exempt from all resource limits, always has full administrative privileges regardless of the settings of its flags, can add, delete and change user accounts and is able to change any usage, limit, or flag value of any lnode using the `limadm` program.

Security

Solaris Resource Manager has a wide effect on the administration of a Solaris system, so it is important that it be installed and maintained in a manner that ensures the system is secure.

There are a number of ways in which the system administrator can ensure that the security of the Solaris Resource Manager system is maintained. The most important, as with any Solaris system, is to ensure the privacy of the root password. Anyone who knows the root user password has unrestricted access to the system's resources, the same as the central administrator.

There are a number of special administrative privileges that may be granted to users within Solaris Resource Manager by the setting of certain system flags within their lnode. These can help increase the security of a system because they allow delegated users to carry out the tasks that are required of them without the need to give them full super-user privileges.

Some of these privileges should not be granted lightly, since they give the recipient user broad-ranging powers. The passwords of users possessing special privileges should be protected diligently, just as the password of the root user should be protected.

There are circumstances in which the central administrator can leave the system open to security breaches if not careful with the manipulation of the structure of the scheduling tree. It is important for the central administrator to understand how to correctly modify the scheduling tree, and to know how to detect potential problems in the current structure.

The *uselimadm* and *admin* Flags

The central administrator may assign administrative privileges within Solaris Resource Manager via the *uselimadm* and *admin* flags in a user's lnode. The *uselimadm* flag to the `limadm(1MSRM)` command allows a user the same administrative privilege as the central administrator. The *admin* flag, when set for a group header, gives the group header administrative privilege over the members of the group that they head, but they are not allowed to alter the contents of any lnode outside their group.

Group headers that have a set *admin* flag are called sub-administrators. There are several security precautions taken within Solaris Resource Manager to prevent misuse of the administrative privilege granted to sub-administrators: Refer to the "A Typical Application Server" on page 23 and "Lnode Maintenance Programs" on page 54 for a detailed explanation.

A sub-administrator, when deleting lnodes, should ensure that sub-trees are deleted from the bottom-most lnodes up. If you start at the top of the sub-tree you are deleting, you will lose control of the children of the lnodes deleted because they will become orphaned when their parents are removed. Once orphaned, the sub-administrator cannot alter the lnodes as they are outside the scheduling group.

Suggested Sub-administrator Lnode Structure

A problem that sub-administrators may face is that they share group limits with their group members. For example, if the group header lnode has a process limit set on it, then that limit controls the number of processes that may be used by the entire group, including the group header. Unless further limited, any user within the scheduling group can prevent the sub-administrator from being able to create new processes, simply by exceeding their own process limit. One way of preventing this is for the group administrator to set individual limits on each of the group members. However, in order to be effective, these limits may have to be overly restrictive. Also, forcing a sub-administrator to manage individual limits is at odds with the Solaris Resource Manager goal of hierarchical resource control. An alternate way of solving this problem is for the administrator to change the structure of the lnodes within his or her group. Rather than placing users directly beneath their own lnode, they should create a "control" lnode beneath their own as their only child lnode, and then make users children of the control lnode. This results in the structure shown.

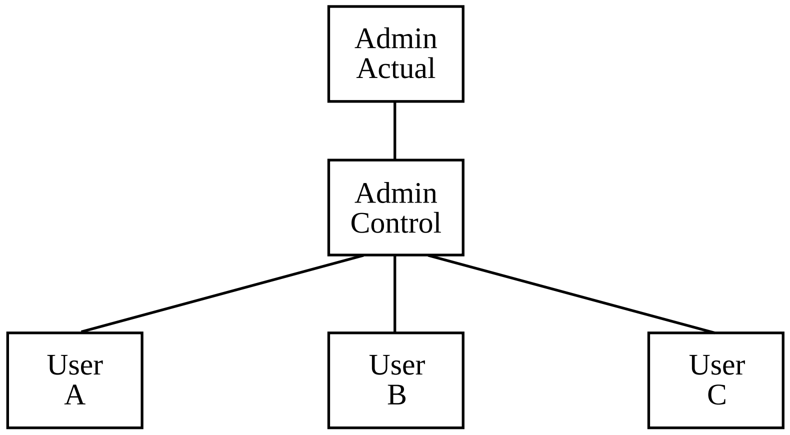

Figure 5–1 Sub-administrator Lnode Structure

Referring to the above figure, the UID of the sub-administrator’s account would correspond to that of the lnode labelled "Actual," the parent of the tree. This is the lnode that would have the *admin* flag set. A dummy account would be created for the "Control" lnode. No login need be permitted on this account. The lnodes labelled "A," "B," and "C" correspond to users under the sub-administrator’s control.

In this case, the process limit for the "Actual" lnode could be 100, while that of the "Control" lnode could be 90, with limits for individual users set to 0. This setup would ensure that even if the users A, B and C were using a total of 90 processes (all they are allowed), the sub-administrator can still create 10 processes. It is still possible in this case for users to stop each other from creating processes. The only way to prevent this is to set individual limits on those users. But in this example, those limits could be set to 40 each, still allowing flexibility while preventing a single user from completely starving the others. Also note that in this example the sub-administrator can create extra lnodes for new users as children of the "Control" lnode without having to worry about re-balancing limits.

Limits Database

The limits database is the database of user information that Solaris Resource Manager uses to perform all resource control. It contains one lnode per UID, which is accessed by using the UID as a direct index into the file. If there is an lnode for a numerically large UID, the limits database will appear to be quite large. However, where the UIDs of users in the system are not sequential, the limits database will have large gaps, or holes, and on a file system type that supports it, may be stored as a sparse file. This means that no disk blocks are actually allocated for storage of the "empty" sections of the file. *ufs* file systems support sparse files, but *tmpfs* file systems do not. See the discussion below in “Saving and Restoring the Limits Database” on page 51 for the implications of sparse files on saving and restoring the limits database.

Whenever you create a new user, you have to create a new lnode.

Creating the Limits Database

The Solaris Resource Manager start-up file (`/etc/init.d/init.srm`) will create an initial limits database when invoked for the first time or at any boot if the file is found to be missing.

The limits database typically resides in the `/var/srm` directory.

The limits database should be owned by root, group owned by root, and readable only by the owner. Write permission is not required since only kernel code with super-user credentials writes to the file.

Caution - If a user can write to the Solaris Resource Manager limits database, system security may be compromised.

Saving and Restoring the Limits Database

Because the limits database may be a sparse file care should be taken when copying the file. The file will most likely consume a lot of disk space if it is written by a utility that does not support sparse files, since the empty regions of the file will read as sequences of zeros and be written back out as real blocks instead of empty regions. This could happen if the file were being copied, backed up or restored by a utility such as `tar(1)`, `cpio(1)` or `cp(1)` though programs such as `ufsdump(1M)` and `ufsrestore(1M)` will preserve holes.

Backup and restoration of the limits database can also be done by using `limreport(1SRM)` to generate an ASCII version of the file and using `limadm(1MSRM)` to re-create the original file from that saved ASCII version. For example, the command:

```
limreport 'flag.real' - lname preserve > /var/tmp/savelnodes
```

will create `/var/tmp/savelnodes` as an ASCII representation of the lnodes for each user in the password map. Note that this will not save lnodes for which there is no corresponding password map entry. It is recommended that lnodes should exist for at most the set of all UIDs in the password map.

The command:

```
# limadm set -f - < /var/tmp/savelnodes
```

will recreate the lnodes whose data was saved. This command will not delete lnodes that were not saved, so these techniques can also be used to save and restore selections of lnodes rather than the whole limits database.

“The `limreport` and `limadm` Commands” on page 57 describes the use of the `limreport(1SRM)` and `limadm(1MSRM)` commands in more detail. It is useful for the administrator to be familiar with the use of these commands for saving and restoring lnodes, since they may need to be used when a change to the interpretation of the lnode structure (as defined by the configuration file) is made.

Note that as the contents of the limits database are changing regularly during normal system operation, it is advisable to perform backup operations while the system is quiescent, or in single-user mode. Similarly, restoring an entire limits database should only ever be done when the Solaris Resource Manager is not in use, such as when the system is in single-user mode.

Creating and Deleting Lnodes

Whenever a new user is created, a corresponding lnode should be created and its limits and privileges should be set. Until an lnode is created, the user is unable to log in. When using Solaris Resource Manager, the administrator should maintain the limits database in parallel with the normal Solaris password database. The command:

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

can be used to print a list of the UIDs and login names of any users who do not have corresponding lnodes.

In this release, lnodes are not automatically created and deleted by the system commands used to create and delete accounts. It is up to the administrator to perform these actions. However, lnodes can be automatically created on-demand when the user logs in, see "PAM Subsystem" on page 35 for more details.

Similarly, just before a user account is deleted from the password map, the corresponding lnode should be removed from the limits database by using the `limadm(1MSRM)` command.

If the UID of a user is ever changed, the contents of their lnode should be copied to a new lnode corresponding to the new number and the original lnode should be deleted. Refer to "Copying and Removing Lnodes" on page 58.

Any child lnodes should be attached either to the newly created lnode, or to some other suitable parent lnode. The command:

```
# limreport 'sgroup==X' '%u\tsgroup=Y\n' uid | limadm set -u -f -
```

can be used to find all lnodes with a scheduling group parent whose UID is X, and make them children of the lnode with a UID of Y.

The following steps illustrate changing the UID of an lnode from X to Y.

1. Save the state of the lnode in which the UID is to be changed:

   ```
   # limreport 'uid==X' - lname preserve > /var/tmp/savelnode.X
   ```

2. Change the UID of the password map entry for the user from the old value (X) to that of the new UID (Y).
3. Create an lnode for the new UID, restoring the state from that which was previously saved:

   ```
   # limadm set -f /var/tmp/savelnode.X
   ```

4. For all child lnodes of the lnode to be changed (UID X), change their scheduling group to the new lnode (UID Y):

```
# limreport 'sgroup==X' '%u\tsgroup=Y\n' uid | limadm set -u -f -
```

5. Ensure there are no processes currently attached to the old lnode. See "Creating and Deleting Lnodes" on page 52.
6. Use the `chown(2)` command to change the owner of all files owned by the original UID to that of the new UID. For example:

```
# find / -user X -print | xargs chown Y
```

7. Delete the old lnode:

```
# limadm delete X
```

Lnode Maintenance Programs

The `limadm` command is the main tool available to administrators to maintain a user's lnode. This command changes Solaris Resource Manager attribute values for a given list of user accounts. If an lnode does not exist for any of the users, then a default-filled blank one is created first. New lnodes are created with the following properties:

- *flag.real* is set;
- *cpu.shares* and *cpu.myshares* attributes are set to 1;
- The flags *uselimadm* and *admin* are set to clear;
- all other flags are set to inherit;
- all limit and usage attributes are set to zero.

The scheduling group of the new lnode is set to that of the invoker of `limadm` if that user is a sub-administrator; otherwise (if the invoker is `root` or has the *uselimadm* flag set) it is set to user other if an lnode for that user account exists, or else to the root lnode.

The `limadm` invoker needs sufficient administrative privilege to perform the specified changes. They must either be the super-user, have a set *uselimadm* flag, or be a sub-administrator and only be changing the attributes of members of their scheduling group. Restrictions apply to a sub-administrator's use of `limadm`: -

- they cannot change the value of their own attributes, except those that have the `selfadmin` condition (for example, the *cpu.myshares* attribute).
- a user's `sgroup` attribute can only be assigned to a group header who is a member of the invoker's scheduling group, or the invoker themselves.

- they cannot change the attributes of users outside their scheduling group.
- they cannot alter the value of any attribute used to store `usages`. If this restriction was absent, then sub-administrators could circumvent the group limits in their own lnode by reducing the usage of one of their children, thereby reducing the group usage.
- if they have a flag that evaluates contrary to the flag's default value, they cannot alter the value of that flag for any member of their group, except to set it to the same contrary value. This ensures that sub-administrators with explicitly denied privileges cannot grant those privileges to any users under their influence.This ensures that sub-administrators with explicitly denied privileges cannot grant those privileges to any users under their influence.

The `limadm` command allows an administrator to remove an lnode without deleting the corresponding user account in the password map. To use `limadm`, the invoker must be the super-user, have a set *uselimadm* flag, or have a set *admin* flag. If the invoker only has a set *admin* flag, then they can only delete the lnodes of users for whom they are a group header.

Units

Values within Solaris Resource Manager are represented in one of three types of units:

Scaled	The scaled unit is the default, easily readable format used to display and enter values. Scaled units help users avoid making entry errors, by reducing the number of digits that need to be entered. -
Raw (or unscaled)	Raw units are the basic unit in which a value is represented. For example, the raw units for virtual memory usage are bytes, and the raw units for virtual memory accrual are byte-seconds. These are mainly used when billing for usage, when exact quantities are required. -
Internal	Internal units are used by Solaris Resource Manager to store memory attributes in machine-dependent units rather than in bytes.

Conversions

Solaris Resource Manager programs carry out conversions to and from the internal units used to store attribute values, so that the user is always presented with scaled units or raw units. This means that, with few exceptions, the user never need be concerned with the internal units used by Solaris Resource Manager.

The terms exa, peta, tera, giga, mega, and kilo are used within Solaris Resource Manager to represent powers of two, not powers of ten. For example, a megabyte is 1,048,576 bytes, not 1,000,000 bytes. The powers of two for each term are 60 (exa), 50 (peta), 40 (tera), 30 (giga), 20 (mega), and 10 (kilo).

The programs that are the primary interface between users and the Solaris Resource Manager system are `limadm(1MSRM)`, `liminfo(1SRM)`, and `limreport(1SRM)`. The conversions and scaling that they carry out are detailed in the following sub-sections.

The `limadm` Command

When changing attribute values, `limadm` allows numbers to be suffixed by scale characters: *[EPTGMK][B][.][wdhms]*. Uppercase and lowercase are interchangeable. If the attribute has the dimension of storage (memory attributes) or of storage accrual, then a character from the first group (EPTGMK) is allowed. This multiplies by the number of bytes in one exabyte (E), petabyte (P), terabyte (T), gigabyte (G), megabyte (M) or kilobyte (K) as appropriate. The optional B character may be appended for the sake of human readability, but has no effect. If the attribute has the dimension of time (type date or time), or of storage accrual, then a character from the second group is allowed. This multiplies by the number of seconds in one week (w), day (d), hour (h), minute (m), or second (s) as appropriate. An optional period may separate the storage and time units (for example. mh, M.h and MB.h all stand for 'megabyte hours'). Where ambiguity exists in the use of the M suffix, `limadm` attempts to derive its meaning from the context. If this is not possible, it is assumed to mean mega, in preference to minutes.

These conversion characters are useful when inputting large numbers to avoid errors in the order of magnitude of the entry, but the quantity is stored in internal units regardless of the method of entry.

A special scale character `u` may also be used, by itself, but only for memory attribute values. It indicates that the number is in machine-dependent (internal) units instead of bytes.

The `liminfo` Command

The `liminfo(1SRM)` command uses the same suffixes when reporting as `limadm(1MSRM)` uses for input (see above). Normally, `liminfo` converts values into appropriate scaled formats to be printed, but the `–r` option can be used to cause `liminfo` to print values in their raw (unscaled) form. For example, memory is normally scaled to a suitable unit, such as megabytes (for example, ’102 MB’), but specifying the `–r` option causes it to be printed in bytes (for example, 106954752 bytes).

The `limreport` Command

The `limreport(1SRM)` command always reports values in their raw (unscaled) form. If scaled values are required, the conversion must be stated explicitly in the expression used to display the value. For example, to display total virtual memory usage for all users in kilobytes, rounded up to the nearest kilobyte:

```
# limreport ’flag.real’ ’%-8.8s %d KB\n’ lname ’(memory.usage+1k-1)/1k’
```

As this example demonstrates, it is permitted to use the scaling suffixes on numbers in expressions, which simplifies the conversion of raw units to scaled values.

Note that the internal units for some attributes are not the same as their ’raw’ form. Normally, this does not need to concern the user since all the Solaris Resource Manager programs carry out conversion to scaled units or raw units, but it means that, for example, select-expressions in `limreport` that specify an exact match on a number of bytes will always fail to match if a number is specified that is not an integral multiple of the relevant internal unit.

Manipulating Lnodes

The `limreport` and `limadm` Commands

The `limreport(1SRM)` and `limadm(1MSRM)` commands provide the administrator with an extremely simple way of saving and restoring the contents of lnodes for any number of users. The `limreport` command is used to select and extract the lnodes that are to be saved, and `limadm` is used to restore them. The most common uses for this combination of commands are for copying lnodes and for altering the lnode structure.

The `limreport` command provides a flexible method for selecting and displaying users’ attributes. It provides two levels of selection: selection of lnodes, and selection of attributes to be displayed for each lnode selected. The lnode selection is achieved by specification of a select-expression, which may be a single condition or a set of conditions joined by logical operators in a C-style syntax. The attribute selection is achieved by listing the attributes' symbolic names. The way in which the attributes are displayed may be specified by a format control string, similar to the C function `limreport`, with extensions to handle special Solaris Resource Manager types. If a format control string of '-' is specified, `limreport` uses default formats for each attribute displayed. Refer to `limreport(1SRM)` for further details.

The `limadm` command provides a facility to indivisibly change the contents of attributes within lnodes, given that the invoker has sufficient privilege. Change commands may be specified directly on the command line or the name of a file containing the change commands may be specified (by using the `–f` option).

`limreport` is able to generate attribute value assignments using the lim syntax (refer to the preserve identifier in the lim syntax), the output of which may be input to `limreport` using the `–f` option. This allows the administrator to use the two programs together to selectively save and restore the contents of the limits database.

Copying and Removing Lnodes

The command:

```
# limreport 'uid==X' - Y preserve | limadm set -u -f -
```

will copy an lnode from UID X to UID Y. The expression 'uid==X' provides the method for selecting the source lnode. The preserve identifier causes `limreport` to output all attribute values that are not read-only in a syntax that is suitable to pass to `limadm`. Placing the UID Y prior to the preserve identifier causes this to be the first item in the data passed to `lim`, thus providing the selection of the target lnode.

If the source lnode is no longer required, it may be removed using `limadm`.

Note - Caution should be used when using a match by UID as the `limreport` selection expression. If multiple login names share a UID, they will all be matched. In the example above, this would not matter, the same lnode data will be preserved and loaded multiple times. In the Solaris system, UID 0 has login names of both `root` and `smtp`.

CHAPTER **6**

The SHR Scheduler

The Solaris Resource Manager SHR scheduler is used to control the allocation of the CPU resource. The concept of shares allows administrators to easily control relative entitlements to CPU resources for users, groups, and applications. The concept of shares is analogous to that of shares in a company; what matters is not how many you have, but how many compared with other shareholders.

Technical Description

There are four attributes per lnode associated with the Solaris Resource Manager scheduler: *cpu.shares*, *cpu.myshares*, *cpu.usage*, and *cpu.accrue*. The output of `liminfo(1SRM)` displays these attributes, and other useful values.

Solaris Resource Manager scheduling is implemented by way of the SHR scheduling class. This includes support for the `dispadmin(1M)`, `priocntl(1)`, `nice(1)`, and `renice(1)` commands. At the system call level, SHR is compatible with the TS scheduling class.

Shares

A user's *cpu.shares* attribute is used to apportion CPU entitlement with respect to the user's parent and active peers. A user's *cpu.myshares* attribute is meaningful only if the user has child users who are active; it is used to determine the proportion of CPU entitlement with respect to them.

For example, if users A and B are the only children of parent P, and A, B and P each have one share each within group P (that is, A and B have *cpu.shares* set to 1, while P has *cpu.myshares* set to 1), then they each have a CPU entitlement of one-third of the total entitlement of the group.

The actual CPU entitlement of a user thus depends on the parent's relative entitlement. This, in turn, depends on the relative values of *cpu.shares* of the parent to the parent's peers and to the *cpu.myshares* of the grandparent, and so on up scheduling tree.

For system management reasons, processes attached to the root lnode are not subject to the shares attributes. Any process attached to the root lnode is always given almost all the CPU resources it requests.

It is important that no CPU-intensive processes be attached to the root lnode, since that would severely impact on the execution of other processes. To avoid this, the following precautions should be taken: -

- The central administrator account should have its own UID, one that is different than the super-user account. This account should be used when logging in to perform non-administrative activities. If a UID of super-user is needed to carry out administrative functions, the central administrator can use the `su(1)` command to change UIDs while still remaining attached to its own lnode.
- The `srmuser(1SRM)` command can be used in the `init.d(4)` scripts to attach any daemon processes to a non-root lnode. Any processes started in the boot script have, by default, an effective UID of root, and are attached to the root lnode. The user command allows daemons to retain an effective UID of root, while attached to their own lnode. This will avoid problems if any of the daemons become CPU-intensive.

Not all group headers in the scheduling tree need to represent actual users who run processes, and in these cases it is not necessary to allocate them a share of CPU. Such lnodes can be indicated by setting their *cpu.myshares* attribute to zero. The CPU accrue attribute in such a group header still includes all charges levied on all members of its group.

Allocated Share

The *cpu.shares* and *cpu.myshares* attributes determine each active lnode's current allocated share of CPU, as a percentage. The shares of inactive users make no difference to allocated share. If only one user is active, then that user will have 100 percent of the available CPU resource. If there are only two active users with equal shares in the same group, each will have allocated shares of 50 percent. Refer to "Calculation of Allocated Share" on page 63 for more information on how the allocated share is calculated.

Usage and Decay

The *cpu.usage* attribute increases whenever a process attached to the lnode is charged for a CPU tick. The usage attribute value exponentially decays at a rate determined by the usage decay global Solaris Resource Manager parameter. The usage decay rate (described by a half-life in seconds) is set by the `srmadm(1MSRM)` command.

Although all processes have an lnode, regardless of their current scheduling class, those outside the SHR scheduling class are never charged.

Accrued Usage

The accrued usage attribute increases by the same amount as the usage attribute, but is not decayed. It therefore represents the total accumulated usage for all processes that have been attached to the lnode and its members since the attribute was last reset.

Effective Share

An lnode's allocated share, together with its `cpu.usage` attribute, determines its current effective share. The Solaris Resource Manager scheduler adjusts the priorities of all processes attached to an lnode so that their rate of work is proportional to the lnode's effective share, and inversely proportional to the number of runnable processes attached to it.

Per-Process Share Priority (`sharepri`)

Each process attached to an lnode has internal Solaris Resource Manager-specific data maintained for it by the operating system kernel. The most important of these values for scheduling purposes is the `sharepri` value. At any time, the processes with the lowest sharepri values will be the most eligible to be scheduled for running on a CPU.

Sample Share Allocation

Scheduling Tree Structure

The following points relate to the structure of the scheduling tree, which is an area requiring special consideration by the central administrator:

- The scheduling tree is the structure used by Solaris Resource Manager to implement a hierarchy of resource and privilege control. If a sub-administrator gains control over a sub-tree of the scheduling tree that they would normally not have access to, they can gain access to additional resource usage and privileges without the approval of the central administrator. One way for this to happen is if an administrator removes an lnode, leaving an orphaned sub-tree behind. A sub-administrator may recreate the missing lnode in their own group, thus giving them control over the previously orphaned section of the tree.
- The central administrator can use the `limreport(1SRM)` command to identify orphaned sections of the scheduling tree, via the built-in orphan identifier. Any orphans found should then immediately be reattached. However, the `limreport` command may also be used by sub-administrators to find orphaned sub-trees so that they can add them to their group, as discussed above. This provides the potential for a breach of security.
- When a new lnode is created, it is mostly zero-filled, which causes most flags to have the default value of inherit. This is the desired effect for most flags, since they are used to indicate device privileges. The flags which are explicitly cleared at lnode creation time are the `uselimadm` and `admin` flags. This is to prevent new users from automatically gaining any administrative privilege.

Description of Tree

The tree shown below defines a structure consisting of several group headers and several ordinary users. The top of the tree is the root user. A group header lnode is shown with two integers, which represent the values of its shares and myshares attributes respectively. A leaf lnode is shown with a single integer, which represents the value of its shares attribute only.

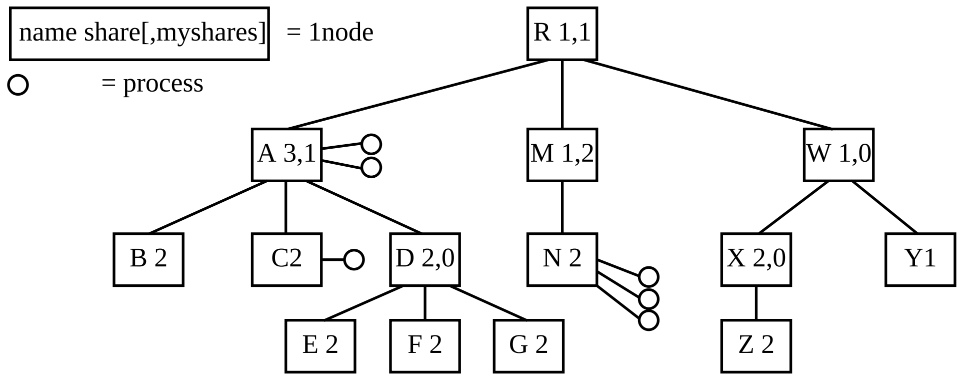

Figure 6–1 Scheduling Tree Structure

Calculation of Allocated Share

Lnodes A, C and N currently have processes attached to them. At the topmost level, the CPU would only need to be shared between A and M since there are no processes for W or any member of scheduling group W. The ratio of shares between A and M is 3:1, so the allocated share at the topmost level would be 75 percent to group A, and 25 percent to group M.

The 75 percent allocated to group A would then be shared between its active users (A and C), in the ratio of their shares within group A (that is, 1:2). Note that the myshares attribute is used when determining A's shares with respect to its children. A would therefore get one third of the group's allocated share, and C would get the remaining two thirds. The whole of the allocation for group M would go to lnode N since it is the only lnode with processes.

The overall distribution of allocated share of available CPU would therefore be 0.25 for A, 0.5 for C and 0.25 for N.

Further suppose that the A, C, and N processes are all continually demanding CPU and that the system has at most two CPUs. In this case, Solaris Resource Manager will schedule them so that the individual processes receive these percentages of total available CPU: -

- for the two A processes: 12.5 percent each;
- for the C process: 50 percent;
- for the three N processes: 8.3 percent each

The rate of progress of the individual processes is controlled so that the target for each lnode is met. On a system with more than two CPUs and only these six runnable processes, the C process will be unable to consume the 50 percent entitlement and the residue is shared in proportion between A and N.

Relationship of Solaris Resource Manager With the Solaris `nice` Facility

The `nice` facility in Solaris allows a user to reduce the priority of a process so that normal processes will not be slowed by non-urgent ones. With Solaris Resource Manager, the incentive for users to use this facility is a reduced charge rate for CPU time used at a lower priority.

Solaris Resource Manager implements this effect by allowing the central administrator to bias the sharepri decay rate for processes which are niced. The pridecay global Solaris Resource Manager parameter in the `srmadm(1MSRM)` command is used to set the decay rates for the priorities of processes with normal and maximum nice values. The rates for all intervening nice values are interpolated between them and similarly extrapolated to the minimum `nice`. For example, the priority (for example, `sharepri`) for normal processes may be decayed with a half-life of two seconds, while the priority of processes with a maximum `nice` value may be decayed with a half-life of sixty seconds.

The effect is that `nice`d processes get a smaller share of CPU than other processes on the same lnode. Under Solaris Resource Manager `nice` has little influence on execution rates for processes on different lnodes unless the queue of runnable processes exceeds the number of CPUs.

Solaris Resource Manager treats processes with a maximum `nice` value (for example, those started with a `nice –19` command) specially. Such processes will only be granted CPU ticks if no other process requests them and they would otherwise be idle.

For information on the relationship of Solaris Resource Manager to other resource control features, see “Relationship to Other Solaris Resource Control Features” on page 4.

Dynamic Reconfiguration

The dynamic reconfiguration (DR) feature of Sun Enterprise servers enables users to dynamically add and delete system boards, which contain hardware resources such as processors, memory, and IO devices. Solaris Resource Manager keeps track of the available processor resources for scheduling purposes and appropriately handles the changes, fairly redistibuting currently available processor resources among eligible users/processes.

Since Solaris Resource Manager controls only the virtual memory sizes of processes, not the physical memory used by processes and users, the effect of a DR operation on memory has no impact on Solaris Resource Manager’s memory-limit checking.

Idle Lnode

The idle lnode (`srmidle`) is the lnode assigned by the central administrator to charge for all the kernel’s idle CPU costs. At installation, the idle user (username *srmidle*) was created with a value of *41*. The idle lnode should have zero shares, to ensure that the processes attached to it are run only when no other processes are active. The idle lnode is assigned using the `srmadm(1MSRM)` command.

At boot time, the default idle lnode is the root lnode. At transition to multi-user mode, the `init.d(4)` script will set the idle lnode to that of the account *srmidle* if such an account exists. This behavior can be customized by specifying a different lnode to use in the `/etc/init.d/init.srm` script.

If the idle lnode is not root, then it must be a direct child of root.

Other Lnode

The other lnode (`srmother`) is the lnode assigned by the systems administrator as the default parent lnode for new users created after the initial install (where root is the default parent lnode). The other lnode, which is auto-created by the system at installation time, has a default value of `1` share, to ensure that lnodes attached to it will have access to the CPU.

The other lnode should have no resource limits, a CPU share of `1` or more, and no special privileges.

Lost Lnode

Under Solaris Resource Manager, the `setuid(2SRM)` system call has a side effect of attaching the calling process to a new lnode. If the change of attachment fails, typically because the new lnode does not exist, then the process is attached instead to the lost lnode (`srmlost`), which was created when you installed Solaris Resource Manager. If this attachment also fails or no lost lnode has been nominated, then the `setuid(2SRM)` function is unaffected and the process continues on its current lnode.

The `init.srm` script sets the lost lnode during the transition to multiuser mode. This behavior can be overridden by specifying an lnode to use in the `/etc/init.d/init.srm` file. To avoid security breaches, the lost lnode should have a CPU share of 1, and no special privileges. (If you alter the values, consider the requirements for this user when making the change.)

CHAPTER 7

Memory Limits, Process Memory Limits, and Process Count Limits

Solaris Resource Manager allows an administrator to control:

- The amount of virtual memory held by users or an application (this is a measure at the lnode level of all processes' virtual memory usage).
- The amount of virtual memory held by a group of users or an application (this is a measure at the group header lnode level of all processes' virtual memory usage where those processes are attached to the group header and/or to its children).
- The amount of virtual memory held by an individual process.
- The number of processes that a user/lnode can run concurrently.

Attributes for Control of Virtual Memory (Total)

There are a number of system attributes used for recording virtual memory usage, and for assigning limits to it.

The system attributes used to administer memory usage are:

`memory.myusage`	The `memory.myusage` attribute of an lnode is equal to the sum of the virtual memory usage of all processes currently attached to the lnode.

`memory.usage`	The `memory.usage attribute` of an lnode is equal to the sum of its `memory.myusage` attribute and the `memory.usage` attributes of its child lnodes.
`memory.limit`	The `memory.limit` attribute is applied to the `memory.usage` attribute. It limits the combined memory usage of all processes attached to the lnode and all member lnodes.

Attribute for Control of Process Memory (Per-Process)

The following system attribute is used to record process usage and assign limits to it.

`memory.plimit`	The `memory.plimit` attribute of an lnode is a per-process limit that is applied separately to the memory usage of each process attached to it or to any of its members.

Technical Description of Memory Limits

Whenever a process attempts to increase its memory size, it is subject to the memory limits (total and per-process) of the lnode to which it is attached. There are five ways in which a process can attempt to increase its memory size:

1. Calling an allocation routine, such as `malloc(3C)`, which results in an invocation of the `sbrk(2)` system call. If a memory limit is exceeded, the call will return an error with errno set to ENOMEM.
2. Expanding the stack and causing a stack fault which would normally cause an extra page of memory to be given to the process. If a memory limit is exceeded, the process will be sent a SIGSEGV signal.
3. Use of `mmap(2)`.
4. Using `fork(2)`. The child address space is duplicated while it is still owned by the parent process. During duplication, the new address space will not exceed the

`memory.plimit` since the parent must already be within this limit; however, the allocation is subject to the `memory.limit`.

5. Using `exec(2)`. During an exec, memory usage first decreases as the old address space is discarded. However, if the address space of the new program is larger, and causes a limit to be exceeded, the exec can fail.

Dynamic Reconfiguration and Virtual Memory Limits

The dynamic reconfiguration (DR) feature available on the Enterprise 10000 systems has limited impact on the virtual memory limits imposed on lnodes. Specifically, DR makes it possible to add or remove physical memory from the system while the system is up and running. (The system's pool of virtual memory includes all of the physical memory, plus the swap space configured into the system.) Additionally, the `swap(1M)` command can be used on any Solaris system to add (–`a`) or delete (–`d`) swap space from the system. Thus, the total amount of virtual memory can grow or shrink during operation.

This has an indirect impact on the virtual memory limits imposed by Solaris Resource Manager. Because virtual memory is managed in absolute units (rather than shares), the effective limits do not change when the system's resources change during operation. (Note that this behavior is different from the dynamic reconfiguration of processors, as discussed in "Dynamic Reconfiguration" on page 64.)

Attributes for Control of Process Count

The system attributes used to administer process usage (the number of processes) are:

`process.myusage`	The `process.myusage` attribute of an lnode is equal to the number of processes attached to the lnode.
`process.usage`	The `process.usage` attribute of an lnode is equal to the sum of its process.myusage attribute and the process.usage attributes of its child lnodes.

`process.limit`	The `process.limit` attribute is a limit which applies to the process.usage attribute of an lnode to limit the number of processes which can be attached to the lnode and all member lnodes concurrently.

Technical Description of Process Count

The `fork(2)`, `fork1(2)` and `vfork(2)` system calls are used to create new processes. If this would cause the process limit to be exceeded, the system call fails, returning an EAGAIN error. Most programs interpret EAGAIN as meaning a temporary shortage of system resources and try the fork again, perhaps after a short sleep. If the fork failure is due to a Solaris Resource Manager limit, this can lead to looping for an indefinite amount of time because EAGAIN will be returned on each retry until the limit is fixed for the affected lnode.

CHAPTER **8**

Usage Data

Solaris Resource Manager provides the administrator with a precise mechanism for collection of accrued usage values for system, application and user resources. This information, plus the functions and utilities provided with Solaris Resource Manager, can be used as a base for the development of a resource billing system.

Accrue Attributes

An lnode's accrue attributes are used to store information about the accrual of resource usage. For example, an lnode's `cpu.accrue` attribute contains the accrued CPU usage for all lnodes within the group as well as that of the current lnode. When any of an lnode's accrue attributes are updated, the change is also applied to the lnode's parent (as with changes to the usage attribute) and so on up to the root lnode, so that the accrued usage at each level in the scheduling tree is the sum of the accrued usage for the lnode and the accrued usage of its children, if any.

Billing Issues

The administrator must decide which lnodes are to be billed for resource usage. For example, administrators may only be concerned with billing entire departments, so they may only wish to bill the group headers of the topmost groups, whose accrued usage will include all the accrued usage of the lnodes at lower levels within their departments.

For administrators to be able to implement a billing system, they must determine a costing function for each resource to be billed. This may be a simple linear relationship (where unit cost is the same, regardless of the amount used), or it may be a non-linear relationship, such as a step function, or a curve where unit cost varies as the amount of usage varies.

In deciding upon a costing function for each resource, the administrator should keep in mind that the costing function will not only control the assignment of cost to accrued resource usage, but can also have an impact on the way in which a user uses the resource. For example, if the costing function for virtual memory usage causes the unit cost to increase as the amount of usage increases, there is a strong incentive for the users to keep virtual memory usage low. Therefore, it is possible for the administrator to control user behavior through the use of an appropriate costing strategy.

There is only one accrue attribute per resource. It contains the accrued usage for the resource based on the usage attribute for the resource. This means that there is no accrued usage corresponding to the `myusage` attribute. For group headers, there is no accrued usage for the user as an individual, since the accrue attribute holds the group's accrued usage. For lnodes with no children, leaf lnodes, this does not matter, since the `myusage` attribute and the usage attribute are identical. If a bill is required for the individual accrued usage for a group header, it must be calculated from the group total less the sum of the individual totals of each child in the group.

The `liminfo` Command

The default output of the `liminfo(1SRM)` command is designed for users who wish to find out their current usages, limits, privileges, etc. The `liminfo` is also of use to administrators who want to enquire on the attributes of other users. There are a number of different report formats that can be requested, including options that make the output of `liminfo` suitable for processing by a pipeline of filters. Refer to `liminfo(1SRM)` for details of the command line options and their meanings, and for a description of the fields that can be displayed.

The `limreport` Command

The `limreport(1SRM)` command allows an administrator to query any attributes, including the accrue attributes, of any user. The command provides a flexible method for the selection of information to be displayed from the chosen lnodes.

For example, the command:

```
% limreport 'cpu.accrue!=0' '%u %u %10d\n' uid  lname cpu.accrue
```

selects all lnodes with any accrued usage in the usr domain, and lists the UID and accrued usage attribute from each of the selected lnodes. To sort these values by `cpu.accrue` and list only the top ten users would be a simple matter of piping the result to a sort command:

```
% limreport 'cpu.accrue!=0' '%u %u %10d\n' uid  lname cpu.accrue | sort -2n | head
```

The `limadm` Command

The `limadm(1MSRM)` command can be used within a billing system to zero the accrue attributes after they have been billed. For example, the command:

```
# limreport 1 '%u\tcpu.accrue=0,mem.accrue=0\n' uid | limadm set -u -f
```

uses the `limreport(1SRM)` command to generate a list of commands piped to `limadm`. Every lnode is selected, and for each lnode, the accrue attribute is zeroed.

The administrator should take care in deciding when to clear the accrue attribute of an lnode. The timing will depend on the billing strategy. For example, if bills are to be produced at a group level, and then individual bills are to be produced for the group members, the accrue attributes of the group members should not be cleared until after both bills have been produced. However, if individual bills are not to be produced, the group members' accrue attributes should be cleared at the same time as the group header's, even though they may not have been individually used.

CHAPTER **9**

Troubleshooting

This section is included to provide assistance to administrators in diagnosing problems in the operation of Solaris Resource Manager.

User Cannot Log In

- There is no lnode corresponding to the UID of the user. This means that an administrator has not set up an lnode for that user's account. The problem is identified by the message to the effect that: No limits information is available. Refer to "Orphaned Lnodes" on page 80.
- The user has the `nologin` or `noattach` flag set.
- The user has the `onelogin` flag set and is already logged in at another terminal or window.
- The user has reached the connect-time usage limit. The user needs to wait for the usage to decay before logging in again, or the administrator can change either the user's terminal.usage attribute or the terminal.limit attribute to allow the user additional terminal connect-time.
- The user's lnode may exist but has been orphaned by removal of its parent lnode. See "Orphaned Lnodes" on page 80.

None of the Solaris Resource Manager limitations listed above apply to the root user.

User Not Informed of Reaching Limits

During normal operation of Solaris Resource Manager, the logged-in user receives notification messages whenever a limit is reached. Users may be unaware of the cause of the problems they are having, and the system will appear to behave mysteriously. However, the system administrator will be notified.

The delivery of notification messages is carried out by the Solaris Resource Manager daemon program, `limdaemon`. There are a number of possibilities that the administrator can investigate if notification messages are not being delivered: - -

- The console window is hidden. Where a user has logged in using a particular window and then opened additional windows that cover the login window, the user may miss a message delivered to their login window.
- The `limdaemon(1MSRM)` program is not running.
- `limdaemon` is unable to dynamically allocate additional memory to maintain its internal structures. If this happens, `limdaemon` displays a diagnostic message on the system console the first time that it fails to get sufficient memory. It continues to attempt to get memory, but fails silently after the first attempt.
- The `utmp` file is corrupt or missing. `limdaemon` relies on this file to determine the terminals where a user is logged in, so that notification messages can be sent to those terminals. If the `utmp` file is corrupt or missing, an error message is reported on the console, and the notification message delivery is suppressed.
- `limdaemon` is unable to deliver a message due to a system limitation. For example, if `limdaemon` needs to open a window on a terminal to deliver the message and is unable to then the message is dropped.-

Unable to Change User's Group

The `sgroup` attribute determines the lnode's parent in the scheduling tree. This hierarchy is used to regulate resource usage and to schedule the CPU. For this reason there are several security precautions placed on the modification of the `sgroup` attribute to avoid inadvertent errors when changing it, and to avoid circumvention of Solaris Resource Manager.

To modify the sgroup attribute, a user needs one of the following privileges:

- be the super-user;
- have a set `uselimadm` flag;

- have a set `admin` flag and be a group header for the lnode being changed.

Orphaned lnodes cannot be made parents of other lnodes. See "Orphaned Lnodes" on page 80.

Corrupted Limits Database

If a corruption occurs to the limits database, it is a major concern to the administrator. There is no single symptom which can reliably be used to determine that the limits database has been corrupted, but there are a number of indicators which should be recognized as potentially reflecting a corrupt limits database.

If corruption is proven, the administrator should revert to an uncorrupted version of the limits database and matching configuration. If the corruption is limited to a small section of the limits database, the administrator may be able to save the contents of all other lnodes and reload them into a fresh limits database using the `limreport(1SRM)` and `limadm(1MSRM)` commands.

For more information about detecting and correcting a corrupt limits database, see "Corruption of the Limits Database" on page 82.

Terminal Connect-time Not Updated

The most likely cause of this problem is that the `limdaemon(1MSRM)` program is not running. `limdaemon` periodically updates the usage and accrue attributes in the terminal device category for all currently logged in users. Typically, it would be started from the Solaris Resource Manager `init.d(4)` script.

Users Frequently Exceeding Limits

There are several likely causes for this: -

- User's administrative limit is set too low for the user's requirements.
- The usage attribute is not being decayed. The administrator is responsible for ensuring that decays are performed on the device categories for all renewable resources (including the terminal device category). Typically, this would be done by regular execution of the `limdaemon(1MSRM)` command. If a decay is not

performed for a renewable resource, the usage attribute for that resource continues to increase until its limit is reached.

- The period between executions of `limdecay` is too long. The frequency of execution of `limdaemon(1MSRM)` should be set to accommodate the granularity of the shortest decay interval.
- The decay attribute for a renewable resource is too small, or the interval attribute is too large. If the decay for a renewable resource over a given time interval is set below the typical consumption rate of that resource, the usage attribute will gradually increase until its limit is reached.

System Runs Slowly

Processes on the Root Lnode

For reasons of system management, processes attached to the root lnode are given almost all the CPU resources they demand. Therefore, if a CPU-bound process is attached to the root lnode, it will tie up a CPU, causing processes on other lnodes to slow or stop.

The following precautions can be taken to prevent this from occurring:

- The administrator should always login to an lnode of their own for normal use, rather than attaching to the root lnode. If they need to attach to the root lnode for some reason, they should be careful not to use any CPU-intensive applications, such as compilers. To use a UID of super-user without attaching to the root lnode, the administrator can use the `su(1)` command.
- The `init.d(4)` scripts should be changed to use the `srmuser(1SRM)` program to attach all daemons to lnodes of their own, so that they are not attached (by inheritance) to the root lnode.

A program that runs as `setuid-root` does not automatically attach to the root lnode. Normally, the process remains attached to the lnode of the parent that created it, and only the effective UID is changed.

CPU Resources Not Controlled by Solaris Resource Manager

Solaris Resource Manager only controls CPU use by processes in the SHR scheduling class. If excessive demands are made at higher priority by other scheduling classes, especially real-time (RT) and system (SYS), then SHR can only schedule with the residual CPU resource.

The use of the RT class conflicts with the Solaris Resource Manager software's ability to control the system. Real-time processes get complete access to the system, specifically so that they can deliver real-time response (generally on the order of a few hundred microseconds). Processes running in the SHR class by definition have lower priority than anything running in real-time, and Solaris Resource Manager has no control over RT processes. Real-time processes can easily consume *all* available resources, leaving Solaris Resource Manager nothing left to allocate to remaining processes.

One notable system service that runs *entirely* in the SYS class is the NFS server. Solaris Resource Manager cannot control the NFS daemons, since they run in SYS. The Solaris Resource Manager product's ability to allocate processor resources may be reduced on systems offering extensive NFS service.

While processes are executing kernel code (inside a system call), the usual time-slice preemption rules do not apply. Most system calls will only do a reasonable amount of work before they reach a preemption point. However if the system is under high load of the more intensive system calls this can result in reduced overall responsiveness and is outside the scope of a scheduling class to control.

If the system is short of available real memory then the resulting I/O bottleneck as the page fault rate increases and process swapping increases leading to increased kernel consumption of CPU. Large amounts of time waiting on I/O may indicate lost CPU capacity. Again, this is outside the scope of a scheduling class to control.

The SHR scheduling class is a time-sharing (TS) scheduler. It uses the same global priority range as the TS and the IA schedulers. It is not appropriate to mix use of SHR with TS and IA except for the transition in moving all processes into or out of the SHR class. System operation with a mix of processes in SHR and TS classes will result in reduced quality of scheduling behavior in both classes. For this reason, Solaris Resource Manager prevents non-root processes from moving themselves or others to the TS or IA classes. The RT class uses an alternate priority range and may be used with the SHR class in the same way as with the TS and IA classes.

If processes run by root contain code that uses the `priocntl(2)` system call directly instead of using the `setpriority(3C)` library routine to adjust process priorities then they may move the target processes into another scheduling class (typically TS). The `setpriority(3C)` library routine code accounts for the fact that the `priocntl(2)` interface to SHR is binary compatible with that for TS and so avoids the problem. The `–c` option of `ps(1)` or the `–d` option of `priocntl(1)` can be used to display the scheduling class of processes.

The same difficulty arises with root privilege processes which explicitly use `priocntl(2)` to manage the scheduling class membership of processes.

Unexpected Notification Messages

Notification messages will be received by any user affected by a limit being reached. Therefore, if a group limit is reached, the group header, and all users below them in the scheduling hierarchy, will receive a notification message.

If a user is attached to another lnode, a limit may be reached. The user will not receive a notification message, but all the other affected users will. The cause of the problem may not be apparent to the group that is affected.

Orphaned Lnodes

The definition of an orphaned lnode is one that has a nonexistent parent lnode. This is of concern to Solaris Resource Manager administrators because Solaris Resource Manager prevents processes from attaching to any lnode that is orphaned or has an orphaned ancestor in the scheduling tree.

The kernel checks changes to the `sgroup` attribute in order to prevent the creation of orphans by invalid alterations to the scheduling group parent.

The major effect of an lnode being orphaned is that it can no longer have processes attached to it. Since no process can connect to it, the lnode cannot be used for logging in. Any attempts to log in using the corresponding account will fail.

The easiest way for an administrator to detect orphaned lnodes is to use the `limreport(1SRM)` command with the built-in orphan identifier. The command

```
% limreport orphan - uid sgroup lname
```

will list the UID, scheduling group parent, and login name of users who have orphaned lnodes. The `sgroup` attribute can be used to determine which of the lnodes is at the top of an orphaned section of the tree.

The first step an administrator should take when an orphaned lnode is discovered is to find the top of the orphaned section of the scheduling tree, since this is the lnode which needs to be re-attached. If the top of the orphaned section is not correctly identified, only part of the orphaned section will be re-attached to the tree.

Once the top of the orphaned section is determined, an administrator with sufficient privilege can use `limadm(1MSRM)` to set the sgroup attribute of the topmost orphaned lnode to a valid lnode within the scheduling tree. This will cause the orphaned lnode to be re-attached to the tree as a member of the group that the valid lnode heads. `limadm` verifies that the new scheduling group parent to be applied is able to be activated, thus ensuring that the lnode being changed will no longer be orphaned.

Alternatively, the administrator may create a new user whose UID is equal to the UID in the `sgroup` attribute of the orphaned lnode. This will cause the automatic reattachment of the orphaned section of the tree.

Group Loops

When an lnode is made active, all of its parents up to the root lnode are also activated. If, in doing this, one of the lnodes is seen to have a parent which has already been encountered, the kernel has discovered a group loop.

If the limits database is corrupted, it is possible for a group loop to occur, where one of the ancestors of an lnode is also one of its children. The kernel automatically connects the loop into the scheduling tree by breaking it arbitrarily and connecting it as a group beneath the root lnode. This means that the lnode at the point where the loop is connected to the scheduling tree becomes a group header of a topmost group. It is possible that members of this group might inherit privileges or higher limits than they would otherwise have.

Causes

Group loops are prevented by `limadm(1MSRM)` when setting scheduling group parents. The only cause of a group loop is corruption to the limits database. This is a serious problem, and may cause all sorts of other difficulties in Solaris Resource Manager since the limits database is so basic to its operation.

Effects

When a group loop is discovered by the kernel, it silently attaches one of the lnodes in the loop directly beneath the root lnode. It cannot determine which is the uppermost lnode since the loop has no beginning or ending.

Correction

The problem is self-correcting in respect to the structure of the scheduling tree since the kernel attaches the lnode to the root lnode. Since the attachment is from an arbitrary point in the loop, the administrator needs to determine where the lnode should be attached and should also check the point of attachment for every other member in the loop.

The result of automatic group loop repair can be seen by listing the lnodes which are children of the root lnode. The command

```
% limreport 'sgroup==0' - uid lname
```

will list all lnodes that have the root lnode as their parent. If any lnodes are listed which should not be children of the root lnode, they are possibly the top of a group loop that has been attached beneath the root lnode.

The major concern of the administrator when a group loop is detected is that, since the cause of the group loop was corruption to the limits database, many more serious problems could arise. If the administrator suspects corruption in the limits database, it is best to carry out some validation checks against the file to determine if it has been corrupted and then take remedial action. Refer to “Crash Recovery” on page 82 for details on detecting and correcting a corrupt limits database.

Crash Recovery

There are many concerns for an administrator when a Solaris system has a failure, but there are some additional considerations when a Solaris Resource Manager system is being used. They are:

- the limits database may have been corrupted by a disk error or for some other reason.
- the operation of `limdaemon(1MSRM)` during a system failure, particularly in relation to connect-time usage charging.

The following sections discuss these in detail, and offer suggestions for handling the situation, where appropriate.

Corruption of the Limits Database

Solaris Resource Manager maintenance of the limits database is very robust and corruption is very unlikely. However, if corruption does occur, it is a major concern to the administrator, since this database is basic to the operation of Solaris Resource Manager. Any potential corruption should be investigated and, if detected, corrected.

Symptoms

There is no single symptom which can reliably be used to determine that the limits database has been corrupted, but there are a number of indicators which should be recognized as potentially reflecting a corrupt limits database:

- the detection of a group loop by the Solaris Resource Manager kernel is a positive indication that the limits database has been corrupted. Group loops are strictly prevented by Solaris Resource Manager, and the only way they can occur is for an `sgroup` attribute to be corrupted in some way. Refer to “Group Loops” on page 81 for more details.
- a user having the message ’No limits information available’ displayed when they attempt to log in, and their login being rejected. This may be caused if the corruption to the limits database causes their `flag.real` attribute to be cleared, which effectively deletes their lnode. It will affect not only the lnode which is deleted, but also any lnodes which are orphaned (Refer to the Orphaned Lnodes section for details). Note that the ’No limits information available’ message will also appear if no lnode has been created for the account, or if it has been intentionally deleted, so it is not a clear indicator that the limits database has been corrupted.
- unrealistic values suddenly appearing in usage or limits attributes. This may cause some users to suddenly hit limits.
- users suddenly complaining of a loss of privilege or unexpected privileges, caused by corruption of privilege flags.

If an administrator suspects that there is any corruption in the limits database, the most likely way to detect it is to use `limreport(1SRM)` to request a list of lnodes where attributes that should have values within a known range are outside that range. This command could also be used to list lnodes whose `flag.real` is clear. This will indicate accounts in the password map for which no lnode exists.

Treatment

If corruption is detected, the administrator should revert to an uncorrupted version of the limits database. If the corruption is limited to a small section of the limits database, the administrator may be able to save the contents of all other lnodes and reload them into a fresh limits database using the `limreport(1SRM)` and `limadm(1MSRM)` commands. This would be preferable if no recent copy of the limits database was available since the new limits database would now contain the most recent usage and accrue attributes. The procedure for saving and restoring the limits database is documented in the Managing lnodes section. For simple cases of missing lnodes it can be sufficient to just recreate them by using the `limadm` command.

Connect-Time Loss by `limdaemon`

If `limdaemon(1MSRM)` terminates for any reason, all users currently logged in cease to be charged for any connect-time usage. Furthermore, when `limdaemon(1MSRM)` is restarted, any users logged in will continue to use that terminal free of charge. This is because the daemon relies on login notifications from `login(1)` to establish a Solaris Resource Manager login session record within the internal structures it uses to calculate connect-time usages. Therefore, whenever it starts, there are no Solaris Resource Manager login sessions established until the first notification is received.

Typically this will not be a problem if the termination of `limdaemon` is due to a system crash, since the crash will cause other processes to terminate also. Login sessions would then not be able to recommence until the system is restarted.

If `limdaemon(1MSRM)` terminates for some other reason, the administrator has two choices:

1. Restart the daemon immediately, and ignore the lost charging of terminal connect-time for users who are already logged in. This could mean that a user has free use of a terminal indefinitely unless sought out and logged off.
2. Bring the system back to single-user mode then return to multi-user mode, thus ensuring that all current login sessions are terminated and users can only log in again after the daemon has been restarted.

CHAPTER **10**

Notification Messages

The Solaris Resource Manager system error messages are:

memory limit reached

The virtual memory usage (`memory.usage`) of an lnode has reached the virtual memory limit (`memory.limit`)

process limit reached

The number of processes (`process.usage`) of an lnode reached the process count limit (`process.limit`).

per-process memory limit reached

The virtual memory usage of a process reached the virtual memory limit (`memory.plimit`).

lnode attach failed in `setuid`

If the Solaris Resource Manager system is installed and enabled, then the `setuid` system call, in addition to its standard function, attaches the calling process to the lnode associated with its new real UID. If attachment fails, it is usually because there is no lnode associated with the new UID.

currently barred from logging in

The `flag.nologin` is set for the user at the time the user tries to log in.

already logged in - only one login allowed

The `flag.onelogin` is set for the user, and the user has already logged in from another terminal.

no permission to use this terminal

The `terminal.flag.all`, `terminal.flag.console`, `terminal.hardwired`, or `terminal.flag.network` is set for the user when the user tries to log in from a particular terminal.

terminal connect time limit reached

The terminal connect time (`terminal.usage`) reached the limit (`terminal.limit`).

APPENDIX **A**

Solaris Resource Manager Script Examples

Initialization Script

The following start-up script is supplied with the system as `/etc/init.d/init.srm` and is executed with an argument of start as the system is changing to run-level 2 or 3 (multi-user mode). It is also executed with an argument of stop at system shutdown.

```
#!/bin/sh
#
# Copyright (c) 1998 by Sun Microsystems, Inc.
# All rights reserved.
#
# Copyright 1995-1997 Softway Pty. Ltd.
#
# Start/stop Solaris Resource Manager v1.0
#
#ident  "@(#)init.srm 1.17 98/10/27 SMI"

#######################################################################
# Default values.

DATADIR=/var/srm
ShareDb=$DATADIR/srmDB
LimdaemonOptions=
ChargeOptionsOn="limits=y:share=y:adjgroups=y:limshare=y"
ChargeOptionsOff="limits=n:share=n:adjgroups=n:limshare=n"

LostLnode=srmlost
IdleLnode=srmidle
```

```
########################################################################

# ECHO=echo # For a verbose startup and shutdown
ECHO=:  # For a quiet startup and shutdown

SRMDIR=/usr/srm
SRMBIN=$SRMDIR/bin
SRMSBIN=$SRMDIR/sbin
ETCSRM=/etc/srm

PATH=/sbin:/usr/sbin:/bin:$PATH:$SRMSBIN:$SRMBIN
export PATH
case "$1" in
'start')
 if [ ! -x $SRMBIN/srmadm ]; then
  echo "Solaris Resource Manager *not* installed." \
   "Missing srmadm command."
  exit
 fi

 # Only bother if sched/SHR is loaded.
 if [ '$SRMBIN/srmadm' != yes ]
 then
  #
  # Usually this is because /etc/system doesn't have the usual
  # set initclass="SHR"
  # or at least a set extraclass="SHR"
  #
  echo "Solaris Resource Manager *not* loaded."
  exit
 else
  echo "Enabling Solaris Resource Manager v1.0."
  if [ '$SRMBIN/srmadm show fileopen' = yes ]; then
   echo "SRM database file already open - stopping first."
   limdaemon -k
   sleep 2
   srmadm set $ChargeOptionsOff
   sync
   srmadm set fileopen=n
   $ECHO "SRM inactive"
  fi
  $ECHO "Starting SRM..."
 fi

 # Check the limconf file.
 if [ ! -s $ETCSRM/limconf ]; then
  echo "SRM - file $ETCSRM/limconf is missing " >&2
  echo "SRM not started."
  exit 1
 fi

 if [ ! -f "$ShareDb" ]; then
  echo "SRM database '$ShareDb' not present - " \
   "creating empty database"
  if [ ! -d "$DATADIR" ]; then
   mkdir "$DATADIR"
   chmod 400 "$DATADIR"
   chown root "$DATADIR"
   chgrp root "$DATADIR"
 fi
 touch "$ShareDb" ||
 {
  echo "Failed to create '$ShareDb'" >&2
  echo "SRM not started"
  exit 1
 }
 chmod 400 "$ShareDb"
 chown root "$ShareDb"
 chgrp root "$ShareDb"
fi

CreateLnodes=0
if [ ! -s "$ShareDb" ]; then
 $ECHO "SRM Warning: Using empty database" >&2
 CreateLnodes=1
fi

$ECHO "SRM starting ... \c"

# Open Lnode file.
srmadm set -f "$ShareDb" fileopen=y
if [ $? != 0 ]; then
 echo
 echo "srmadm set -f $ShareDb failed" >&2
 echo "SRM not started"
 exit 1
fi

# Set SRM global options.
srmadm set $ChargeOptionsOn
if [ $? != 0 ]; then
 echo
 echo "srmadm set $ChargeOptionsOn failed" >&2
 echo "SRM not completely enabled"
 exit 1
fi

# Create if needed the other lnode.
liminfo other 2>/dev/null | grep "^Login name:  *other " >/dev/null 2>&1
if [ $? -ne 0 ]; then
 # If user "other" exists but has no lnode, create one.
 limadm set cpu.shares=1 other 2>/dev/null
 limadm set sgroup=root other 2>/dev/null
fi

# Create if needed, and set the lost lnode.
if [ x"$LostLnode" != x ]; then
        liminfo "$LostLnode" 2>/dev/null | \
  grep "^Login name:  *$LostLnode " >/dev/null 2>&1
 if [ $? -ne 0 ]; then
  limadm set cpu.shares=1 "$LostLnode"
  limadm set sgroup=root "$LostLnode"
 fi
 srmadm set lost="$LostLnode" ||
 $ECHO "SRM - Warning: No user '$LostLnode' for lost lnode"
fi

# Create if needed, and set the idle lnode.
if [ x"$IdleLnode" != x ]; then
```

```
        liminfo "$IdleLnode" 2>/dev/null | \
  grep "^Login name:  *$IdleLnode " >/dev/null 2>&1
 if [ $? -ne 0 ]; then
  limadm set cpu.shares=0 "$IdleLnode"
  limadm set sgroup=root "$IdleLnode"
 fi
 srmadm set idle="$IdleLnode" ||
 $ECHO "SRM - Warning: No user '$IdleLnode' for idle lnode"
fi

# If creating SRM database, set up existing users.
if [ "$CreateLnodes" -eq 1 ]; then
 echo "SRM - creating user lnodes; may take a while"
# We now want to catch any other users which were not found
# on the filesystems.  First we need to decide what the maximum
# uid value we will create an l-node entry for in the database.
# We choose less than the uid for 'nobody' so that we can try
# and minimise the apparent size of the database (which is a sparse
# file).  If the user 'nobody' does not exist then we just have
# to take our chances with using all possible uid values.
# Unfortunately all this means that there are certain circumstances
# where not all users will be taken into account.

MaxUID=`awk -F: "\\$1==\"nobody\" { print \\$3 - 100 }" /etc/passwd`
 if [ $? -eq 0 -a x"$MaxUID" != x ] ; then
  Cond="uid >= 0 && uid < $MaxUID && !flag.real"
 else
  Cond="uid >= 0 && !flag.real"
 fi
 UIDS=`limreport "$Cond" '%d\n' uid | wc -l`
 if [ $UIDS -gt 0 ]; then
  $ECHO "$UIDS other lnodes to be created " \
   "due to passwd entries"
   limreport "$Cond" 'limadm set -u sgroup=0:cpu.shares=1 %d\necho " uid %7d\r\c"\n' uid uid | sh

  echo
 fi
fi

limdaemon $LimdaemonOptions

echo "Solaris Resource Manager v1.0 Enabled."
   ;;

'stop')
 # SRM shutdown should be done as late as possible before
 # filesystems are unmounted.
 if [ -x $SRMBIN/srmadm ] && $SRMBIN/srmadm show fileopen > /dev/null
 then
  limdaemon -k
  sleep 2
  srmadm set $ChargeOptionsOff
  srmadm set fileopen=n
  sync
  $ECHO "Solaris Resource Manager Disabled"
 fi
 ;;

*)
 echo "Usage: $0 {start|stop}"
 ;;
esac
```

Default ’no lnode’ Script

This script creates the lnode in the default scheduling group (`other` if such a user exists in the password map, otherwise `root`) and mails the system administrator a reminder to move the new lnode into the appropriate place in the scheduling hierarchy.

```
#!/bin/sh
#
#ident  "@(#)nolnode.sh 1.6 98/10/28 SMI"
#
# Copyright (c) 1998 by Sun Microsystems, Inc.
# All rights reserved.
#
# Copyright 1995-1997 Softway Pty. Ltd.
#
# A script that called by the PAM module to create a lnode
#

PATH=/usr/srm/bin:/sbin:/bin export PATH
LOCALE=C export LOCALE
if [ "$DEBUG" = "true" ]
then
    exec >> /tmp/nolnodelog 2>&1
    echo
    date
    echo "Attempting to create lnode for $USER"
else
    exec > /dev/null  2>&1
fi

err=‘limadm set -u cpu.shares=1 "$UID" 2>&1‘
if [ $? -eq 0 ]
then
    cat <<-EOF | /usr/lib/sendmail root
 Subject: New lnode created for "$USER"

 Remember to change scheduling group and shares for
 "$USER".

 ‘limreport lname==’"’"$USER"’"’ ’Currently in group "%s" with %d shares\\n’ sgroupname cpu.shares‘
EOF
else
    cat <<-EOF | /usr/lib/sendmail root
 Subject: Could not create lnode for "$USER"

 after "$SERVICE" attempt on tty "$TTY", uid "$UID",
 rhost \""$RHOST"\",
 limadm said \""$err"\"
EOF
    exit 1 # deny access
fi
# permit access
exit 0
```

Glossary

There are a number of new concepts introduced in Solaris Resource Manager, and there are a number of areas which overlap conceptually with other parts of Solaris.

In order to simplify the discussion within this document and to avoid confusion the following terms are defined.

accrue	For both fixed and renewable resources there may be an accrue usage attribute which is the integral of the corresponding usage attribute over time.
active	An lnode is active if there are any processes attached to it, or attached to any of its descendants. An lnode cannot be removed while it is active.
admin user	A user whose Inotde's admin flag evalutes to set. Such a user may create, delete and modify Inodes within their scheduling group.
administrator	Anyone whose role includes maintaining the system. Solaris Resource Manager provides functionality to allow administrative authority to be delegated without needing to give super-user privileges away. See also admin user, super-user, uselimadm user and "Delegated Administration" on page 47.
allocated share	The fraction of available CPU resources that would be given to a user in the long term with a given configuration of lnode tree hierarchy, shares, and active lnodes.
ancestor	One lnode is the ancestor of another if successive references to the sgroup attribute, starting from the first lnode, eventually reference the other. That is, the latter lnode is a descendant, or a member, of the first.

attached	When a user logs in, their PAM module process attaches to the lnode which corresponds to their UID. Any processes which are subsequently spawned are attached to the same lnode by default. The lnode that a process is attached to is the one which is used to determine the process' limits, CPU entitlement, provileges, etc.
attribute	The data fields of an lnode are called attributes. All lnodes have the same internal structure, therefore all users have the same set of attributes. Attributes may be system, user, (only used by user-mode programs) or domain. The different types of attributes differ in the field-numbers allotted to them. System addributes are those which are used directly by the kernel, such as the numeric variables to control resources such as processes, memory size, and flags that control system privileges at the kernel level. User attributes can be added at any time by the administrator and existing user attributes can be modified any time provided they do not disrupt programs that use the attribute. Domain attributes are not declared in the configuration source file, because the declaration of a domain defines them implicitly.
central administrator	The central administrator is the root user (or the super-user) of the system. The root lnode is always the top of the scheduling tree. The central administrator has overall responsibility for the administration of all users and resources, but may delegate some administrative responsibility to other ordinary users by granting administrative privileges. Typically the central administrator would determine the allocation of resources to the groups that are children of the root lnode, and grant administrative privileges to the group headers of each of those groups, thus relieving much of their own administrative burden.
child	All lnodes which are directly beneath another in the scheduling tree are the children of that lnode. One lnode is a child of another if the first lnode's `sgroup` attribute is set to the UID of the second lnode. Correspondingly, the latter lnode is referred to as the parent or group header of the former.
decay	A decay is the periodic reduction of the usage of a renewable resource. For all resources except CPU usage, the decay is a fixed amount which is deemed to be subtracted from the usage attribute on a regular basis. For CPU usage, an exponential (multiplicative) decay is used.
effective share	See Chapter 6.

entitlement	This term refers to the amount of CPU time which is allocated to a particular user.
field number	The array slot(s) used by attributes specified in the configuration file.
fixed resource	A resource whose total supply is finite.
flag	There is a special type of attribute called a flag which is similar to a boolean variable, except that it may have one of four falues: `set`, `clear`, `group`, or `inherit`.Flags are used within Solaris Resource Manager to control privileges.
group loop	See “Group Loops” on page 81.
group	Within Solaris Resource Manager, this term normally refers to a scheduling group. See scheduling group.
group header	The lnode at the head of a group is referred to as a group header lnode. Solaris Resource Manager allows selected administrative privileges to be granted to group headers so that, for example, the head of a department may assume full responsibility for the creation and removal of groups and users, and allocation of resources, within the department. Group headers are not considered members of the group that they head.
idle lnode	A special lnode to which unused CPU time is accrued. This may be useful for accounting purposes. The default username is *srmidle*, which has a UID of 41.
inherit	One of the possible values that may be given to a flag attribute. When a flag with an immediate value of `inherit` is being evaluated, the same flag in the parent lnode is evaluated to determine the actual value. This process is recursive. If the flag is set to `inherit` on the root lnode, then the final value is determined from the default value. The result of evaluating a flag is always `set` or `clear`.
kernel	The core of the operating system, which supports system calls, file systems and process scheduling. Solaris Resource Manager consists of two kernel modules and a number of kernel hooks, as well as user-level (non-kernel) programs and library routines.
limit	A limit is a numeric attribute associated with a usage attribute. A user’s usage of a resource is prevented from exceeding the limits for

	that resource. There are two kinds of limits: hard and soft .A hard limit will cause resource consumption or allocation attempts to fail if they would cause the usage to exceed the limit.A soft limit typically does not directly constrain usage, but represents instead a point at which the user is informed of their usage and is encouraged to reduce it. A limit of zero is a special case, meaning no limit applies.
lnode database	The on-disk copy of all the lnodes used by Solaris Resource Manager, indexed by UID.
lnode	An lnode is a fixed-length structure used by Solaris Resource Manager to hold all per-user data required over and above the data stored in the password map. It is a structure stored on disk in the lnode file and is read and written by the kernel as required. There is at most a single lnode for each unique UID. Different accounts which have the same UID use the same lnode.
lost lnode	A special lnode used when the `setuid(2SRM)` system call cannot attach a process to the lnode corresponding to the target UID of the system call, usually because that lnode does not exist.
notification	Any message which is sent to the Solaris Resource Manager daemon, `limdaemon(1MSRM)`, is a notification message. Some notification messages have special meaning to `limdaemon`.
orphan	An lnode whose parent lnode does not exist. That is, the UID specified in the sgroup attribute of the lnode does not itself correspond to an lnode.
other lnode	If an account exists for a user named other, and an lnode exists for that account, then that lnode will be used as the default for the parent of lnodes newly created by root or *uselimadm* users using the `limadm(1MSRM)` command.
parent	The group header of an lnode in the scheduling tree is the lnode's parent.
peer	The peers of an lnode are other lnodes within its scheduling group, excluding the parent of the group.
renewable resource	A resource where more units of that resource become available over time. For example, CPU usage or connect-time.
root lnode	The lnode for UID 0. This lnode is the head of the entire tree of lnodes, making all other lnodes its members.

scheduling group	Solaris Resource Manager allows all users to be organized into a system-wide hierarchy of scheduling groups typically reflecting the structure of the organizations using the system. The term scheduling group is used in preference to simply using group to avoid confusion with the existing UNIX group concept, even though scheduling groups are used for much more than just scheduling. Solaris Resource Manager groups need bear no relation to the groups defined in the UNIX `/etc/group` file. A scheduling group at any level in the lnode hierarchy can be treated as a single user. That is, resource limits assigned to a scheduling group apply to the net usage of all groups and users within that group.
scheduling tree	The tree of lnodes, headed by the root lnode, with particular reference to the parent-child relationship between lnodes, the allocation of CPU shares and how the Solaris Resource Manager scheduler determines process run rates
shares	Shares are a way of defining the proportion of CPU entitlement that an lnode has with respect to its parent and peer lnodes. This concept of CPU shares is analogous to shares in a company; what matters is not how many you have, but how many compared with other shareholders.
Solaris Resource Manager login session	This is any login-like connection to the system of which Solaris Resource Manager is aware - this requires cooperation between Solaris Resource Manager and the various ’gateway’ programs which are responsible for authenticating users and granting them access.
sub-administrator	A sub-administrator is a group header who has administrative privilege over the members of the scheduling group that he or she heads. A group header is granted sub-administrator status by setting their admin flag. This allows them to create and delete users’ lnodes within their group, to control resource and privilege allocation within their group and to further delegate administrative responsibility to group headers within their group.
super-user	A person is referred to as a super-user if they know the root password. Processes have super-user privilege if they are running with an effective UID of 0. .
topmost group	Any group with root as its group header.

units	This term refers to the basic quantity of a given resource. Values within Solaris Resource Manager are represented as one of three types of units: scaled, raw, or internal.
usage	A user's usage of a resource is a numeric attribute which increases whenever the user consumes or is allocated some of the resource. For fixed resources, the usage is decreased whenever any of the resource is freed. For renewable resources, the usage is decreased whenever a decay is performed.
uselimadm user	A user whose lnode has the flag `uselimadm` attribute set. This gives a user the same privileges in regard to Solaris Resource Manager administration as the super-user.
user attribute	See attribute.
user-mode	The way in which code is executed by normal programs and processes on a UNIX system. The alternative, kernel-mode, is used by system calls, device drivers and SYS class scheduler. Solaris Resource Manager has some components which run in user-mode and some which run in kernel-mode.