Administering Resource Management in Oracle® Solaris 11.2

Part No: E37631
July 2014

Contents

Using This Documentation 11

1 Introduction to Resource Management 13
- Resource Management Overview 13
  - Resource Classifications 14
  - Resource Management Control Mechanisms 15
  - Resource Management Configuration 16
  - Interaction With Non-Global Zones 16
- When to Use Resource Management 16
  - Server Consolidation 17
  - Supporting a Large or Varied User Population 17
- Setting Up Resource Management Task Map 18

2 About Projects and Tasks 21
- Project and Task Facilities 21
- Project Identifiers 22
  - Determining a User's Default Project 22
  - Setting User Attributes With the `useradd` and `usermod` Commands 23
  - `project` Database 23
  - PAM Subsystem 24
  - Naming Services Configuration 24
  - Local `/etc/project` File Format 24
  - Project Configuration for NIS 26
  - Project Configuration for LDAP 27
- Task Identifiers 27
- Commands Used With Projects and Tasks 28

3 Administering Projects and Tasks 31
- Administering Projects and Tasks Task Map 31
- Example Commands and Command Options 32

Command Options Used With Projects and Tasks 32
Using `cron` and `su` With Projects and Tasks 34
Administering Projects 35
▼ How to Define a Project and View the Current Project 35
▼ How to Delete a Project From the `/etc/project` File 37
How to Validate the Contents of the `/etc/project` File 38
How to Obtain Project Membership Information 38
▼ How to Create a New Task 39
▼ How to Move a Running Process Into a New Task 39
Editing and Validating Project Attributes 40
▼ How to Add Attributes and Attribute Values to Projects 40
▼ How to Remove Attribute Values From Projects 41
▼ How to Remove a Resource Control Attribute From a Project 41
▼ How to Substitute Attributes and Attribute Values for Projects 41
▼ How to Remove the Existing Values for a Resource Control Attribute 42

4 About Extended Accounting 43
Introduction to Extended Accounting 43
How Extended Accounting Works 44
Extensible Format 45
`exacct` Records and Format 45
Using Extended Accounting on an Oracle Solaris System with Zones Installed 46
Extended Accounting Configuration 46
Starting and Persistently Enabling Extended Accounting 46
Records 47
Commands Used With Extended Accounting 47
Perl Interface to `libexacct` 48

5 Administering Extended Accounting Tasks 51
Administering the Extended Accounting Facility Task Map 51
Using Extended Accounting Functionality 51
▼ How to Activate Extended Accounting for Flows, Processes, Tasks, and Network Components 52
How to Display Extended Accounting Status 53
How to View Available Accounting Resources 53
▼ How to Deactivate Process, Task, Flow, and Network Management Accounting 54
Using the Perl Interface to `libexacct` 55

How to Recursively Print the Contents of an `exacct` Object ... 55
How to Create a New Group Record and Write It to a File ... 56
How to Print the Contents of an `exacct` File ... 57
Example Output From `Sun::Solaris::Exacct::Object->dump()` ... 58

6 About Resource Controls ... 59
Resource Controls Concepts ... 59
Resource Limits and Resource Controls ... 60
Interprocess Communication and Resource Controls ... 60
Resource Control Constraint Mechanisms ... 60
Project Attribute Mechanisms ... 61
Configuring Resource Controls and Attributes ... 61
Available Resource Controls ... 62
Zone-Wide Resource Controls ... 65
Units Support ... 66
Resource Control Values and Privilege Levels ... 67
Global and Local Actions on Resource Control Values ... 68
Resource Control Flags and Properties ... 70
Resource Control Enforcement ... 71
Global Monitoring of Resource Control Events ... 72
Applying Resource Controls ... 72
Temporarily Updating Resource Control Values on a Running System ... 73
Updating Logging Status ... 73
Updating Resource Controls ... 73
Commands Used With Resource Controls ... 73

7 Administering Resource Controls Tasks ... 75
Administering Resource Controls Task Map ... 75
Setting Resource Controls ... 76
▼ How to Set the Maximum Number of LWPs for Each Task in a Project ... 76
▼ How to Set Multiple Controls on a Project ... 77
Using the `prctl` Command ... 78
▼ How to Use the `prctl` Command to Display Default Resource Control Values ... 79
▼ How to Use the `prctl` Command to Display Information for a Given Resource Control ... 81
▼ How to Use `prctl` to Temporarily Change a Value ... 81
▼ How to Use `prctl` to Lower a Resource Control Value ... 82

▼ How to Use `prctl` to Display, Replace, and Verify the Value of a Control on a Project ... 82
Using `rctladm` ... 83
How to Use `rctladm` ... 83
Using `ipcs` ... 84
How to Use `ipcs` ... 84
Capacity Warnings ... 84
▼ How to Determine Whether a Web Server Is Allocated Enough CPU Capacity ... 85

8 About Fair Share Scheduler ... 87
Introduction to the Scheduler ... 87
CPU Share Definition ... 88
CPU Shares and Process State ... 88
CPU Share Versus Utilization ... 89
CPU Share Examples ... 89
Example 1: Two CPU-Bound Processes in Each Project ... 90
Example 2: No Competition Between Projects ... 90
Example 3: One Project Unable to Run ... 91
FSS Configuration ... 91
Projects and Users ... 91
CPU Shares Configuration ... 92
FSS and Processor Sets ... 93
FSS and Processor Sets Examples ... 93
Combining FSS With Other Scheduling Classes ... 95
Setting the Scheduling Class for the System ... 96
Scheduling Class on a System with Zones Installed ... 96
Commands Used With FSS ... 96

9 Administering the Fair Share Scheduler Tasks ... 97
Administering the Fair Share Scheduler Task Map ... 97
Monitoring the FSS ... 98
▼ How to Monitor System CPU Usage by Projects ... 98
▼ How to Monitor CPU Usage by Projects in Processor Sets ... 98
Configuring the FSS ... 99
Listing the Scheduler Classes on the System ... 99
▼ How to Make FSS the Default Scheduler Class ... 99
▼ How to Manually Move Processes From the TS Class Into the FSS Class ... 100

▼ How to Manually Move Processes From All User Classes Into the FSS Class 100
▼ How to Manually Move a Project's Processes Into the FSS Class 101
How to Tune Scheduler Parameters 101

10 About Physical Memory Control Using the Resource Capping Daemon 103
Introduction to the Resource Capping Daemon 103
How Resource Capping Works 104
Attribute to Limit Physical Memory Usage for Projects 104
`rcapd` Configuration 105
Using the Resource Capping Daemon on a System With Zones Installed 105
Memory Cap Enforcement Threshold 106
Determining Cap Values 106
`rcapd` Operation Intervals 107
Monitoring Resource Utilization With `rcapstat` 109
Commands Used With `rcapd` 110

11 Administering the Resource Capping Daemon Tasks 111
Setting the Resident Set Size Cap 111
▼ How to Add an `rcap.max-rss` Attribute for a Project 111
▼ How to Use the `projmod` Command to Add an `rcap.max-rss` Attribute for a Project 112
Configuring and Using the Resource Capping Daemon Task Map 112
Administering the Resource Capping Daemon With `rcapadm` 113
▼ How to Set the Memory Cap Enforcement Threshold 113
▼ How to Set Operation Intervals 113
▼ How to Enable Resource Capping 114
▼ How to Disable Resource Capping 114
▼ How to Specify a Temporary Resource Cap for a Zone 115
Producing Reports With `rcapstat` 115
Reporting Cap and Project Information 116
Monitoring the RSS of a Project 116
Determining the Working Set Size of a Project 117
Reporting Memory Utilization and the Memory Cap Enforcement Threshold 118

12 About Resource Pools 119
Introduction to Resource Pools 120

Introduction to Dynamic Resource Pools 121
About Enabling and Disabling Resource Pools and Dynamic Resource Pools 121
Resource Pools Used in Zones 121
When to Use Pools 122
Resource Pools Framework 123
    /etc/pooladm.conf Contents 124
    Pools Properties 124
Implementing Pools on a System 125
project.pool Attribute 125
SPARC: Dynamic Reconfiguration Operations and Resource Pools 125
Creating Pools Configurations 126
Specific Assignment of CPUs, Cores, and Sockets 127
Directly Manipulating the Dynamic Configuration 127
poold Overview 127
Managing Dynamic Resource Pools 128
Configuration Constraints and Objectives 128
    Configuration Constraints 128
    Configuration Objectives 129
    poold Properties 132
poold Functionality That Can Be Configured 133
    poold Monitoring Interval 133
    poold Logging Information 133
    Logging Location 136
    Log Management With logadm 136
How Dynamic Resource Allocation Works 136
    About Available Resources 136
    Determining Available Resources 136
    Identifying a Resource Shortage 137
    Determining Resource Utilization 137
    Identifying Control Violations 138
    Determining Appropriate Remedial Action 138
Using poolstat to Monitor the Pools Facility and Resource Utilization 139
    poolstat Output 139
    Tuning poolstat Operation Intervals 140
Commands Used With the Resource Pools Facility 140

13 Creating and Administering Resource Pools Tasks 143
Administering Resource Pools Task Map 143

Enabling and Disabling the Pools Facility 144
▼ How to Enable the Resource Pools Service Using svcadm 145
▼ How to Disable the Resource Pools Service Using svcadm 145
▼ How to Enable the Dynamic Resource Pools Service Using svcadm 145
▼ How to Disable the Dynamic Resource Pools Service Using svcadm 148
▼ How to Enable Resource Pools Using pooladm 148
▼ How to Disable Resource Pools Using pooladm 148
Specific CPU Assignment 148
Configuring Pools 149
▼ How to Create a Static Configuration 149
▼ How to Modify a Configuration 150
▼ How to Associate a Pool With a Scheduling Class 152
▼ How to Set Configuration Constraints 154
▼ How to Define Configuration Objectives 155
▼ How to Set the poold Logging Level 157
▼ How to Use Command Files With poolcfg 157
Transferring Resources 158
▼ How to Move CPUs Between Processor Sets 158
Activating and Removing Pool Configurations 159
▼ How to Activate a Pools Configuration 159
▼ How to Validate a Configuration Before Committing the Configuration 159
▼ How to Remove a Pools Configuration 160
Setting Pool Attributes and Binding to a Pool 160
▼ How to Bind Processes to a Pool 160
▼ How to Bind Tasks or Projects to a Pool 161
▼ How to Set the project.pool Attribute for a Project 161
▼ How to Use project Attributes to Bind a Process to a Different Pool 161
Using poolstat to Report Statistics for Pool-Related Resources 162
Displaying Default poolstat Output 162
Producing Multiple Reports at Specific Intervals 163
Reporting Resource Set Statistics 163

14 Resource Management Configuration Example 165
Configuration to Be Consolidated 165
Consolidation Configuration 166
Creating the Configuration 166
Viewing the Configuration 167

Index 173

Using This Documentation

- **Overview** – Describes how to write applications that partition and manage system resources such as processor sets and scheduling classes.
- **Audience** – Programmers with experience with operating system interfaces.
- **Required knowledge** – Knowledge of C and of Solaris (or other Unix-like) system interfaces.

Product Documentation Library

Late-breaking information and known issues for this product are included in the documentation library at http://www.oracle.com/pls/topic/lookup?ctx=E36784.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Feedback

Provide feedback about this documentation at http://www.oracle.com/goto/docfeedback.

CHAPTER 1

Introduction to Resource Management

Oracle Solaris resource management functionality enables you to control how applications use available system resources. You can do the following:

- Allocate computing resources, such as processor time
- Monitor how the allocations are being used, then adjust the allocations as necessary
- Generate extended accounting information for analysis, billing, and capacity planning

This chapter covers the following topics.

- "Resource Management Overview" on page 13
- "When to Use Resource Management" on page 16
- "Setting Up Resource Management Task Map" on page 18

Resource Management Overview

Modern computing environments have to provide a flexible response to the varying workloads that are generated by different applications on a system. A *workload* is an aggregation of all processes of an application or group of applications. If resource management features are not used, the Oracle Solaris operating system responds to workload demands by adapting to new application requests dynamically. This default response generally means that all activity on the system is given equal access to resources. Resource management features enable you to treat workloads individually. You can do the following:

- Restrict access to a specific resource
- Offer resources to workloads on a preferential basis
- Isolate workloads from each another

The ability to minimize cross-workload performance compromises, along with the facilities that monitor resource usage and utilization, is referred to as *resource management*. Resource management is implemented through a collection of algorithms. The algorithms handle the series of capability requests that an application presents in the course of its execution.

Resource management facilities permit you to modify the default behavior of the operating system with respect to different workloads. *Behavior* primarily refers to the set of decisions that are made by operating system algorithms when an application presents one or more resource requests to the system. You can use resource management facilities to do the following:

- Deny resources or prefer one application over another for a larger set of allocations than otherwise permitted
- Treat certain allocations collectively instead of through isolated mechanisms

The implementation of a system configuration that uses the resource management facilities can serve several purposes. You can do the following:

- Prevent an application from consuming resources indiscriminately
- Change an application's priority based on external events
- Balance resource guarantees to a set of applications against the goal of maximizing system utilization

When planning a resource-managed configuration, key requirements include the following:

- Identifying the competing workloads on the system
- Distinguishing those workloads that are not in conflict from those workloads with performance requirements that compromise the primary workloads

After you identify cooperating and conflicting workloads, you can create a resource configuration that presents the least compromise to the service goals of the business, within the limitations of the system's capabilities.

Effective resource management is enabled in the Oracle Solaris system by offering control mechanisms, notification mechanisms, and monitoring mechanisms. Many of these capabilities are provided through enhancements to existing mechanisms such as the `proc`(4) file system, processor sets, and scheduling classes. Other capabilities are specific to resource management. These capabilities are described in subsequent chapters.

Resource Classifications

A resource is any aspect of the computing system that can be manipulated with the intent to change application behavior. Thus, a resource is a capability that an application implicitly or explicitly requests. If the capability is denied or constrained, the execution of a robustly written application proceeds more slowly.

Classification of resources, as opposed to identification of resources, can be made along a number of axes. The axes could be implicitly requested as opposed to explicitly requested, time-based, such as CPU time, compared to time-independent, such as assigned CPU shares, and so forth.

Generally, scheduler-based resource management is applied to resources that the application can implicitly request. For example, to continue execution, an application implicitly requests additional CPU time. To write data to a network socket, an application implicitly requests bandwidth. Constraints can be placed on the aggregate total use of an implicitly requested resource.

Additional interfaces can be presented so that bandwidth or CPU service levels can be explicitly negotiated. Resources that are explicitly requested, such as a request for an additional thread, can be managed by constraint.

Resource Management Control Mechanisms

The three types of control mechanisms that are available in the Oracle Solaris operating system are constraints, scheduling, and partitioning.

Constraint Mechanisms

Constraints allow the administrator or application developer to set bounds on the consumption of specific resources for a workload. With known bounds, modeling resource consumption scenarios becomes a simpler process. Bounds can also be used to control ill-behaved applications that would otherwise compromise system performance or availability through unregulated resource requests.

Constraints do present complications for the application. The relationship between the application and the system can be modified to the point that the application is no longer able to function. One approach that can mitigate this risk is to gradually narrow the constraints on applications with unknown resource behavior. The resource controls discussed in Chapter 6, “About Resource Controls” provide a constraint mechanism. Newer applications can be written to be aware of their resource constraints, but not all application writers will choose to do this.

Scheduling Mechanisms

Scheduling refers to making a sequence of allocation decisions at specific intervals. The decision that is made is based on a predictable algorithm. An application that does not need its current allocation leaves the resource available for another application's use. Scheduling-based resource management enables full utilization of an undercommitted configuration, while providing controlled allocations in a critically committed or overcommitted scenario. The underlying algorithm defines how the term “controlled” is interpreted. In some instances, the scheduling algorithm might guarantee that all applications have some access to the resource. The fair share scheduler (FSS) described in Chapter 8, “About Fair Share Scheduler” manages application access to CPU resources in a controlled way.

Partitioning Mechanisms

Partitioning is used to bind a workload to a subset of the system's available resources. This binding guarantees that a known amount of resources is always available to the workload. The resource pools functionality that is described in Chapter 12, "About Resource Pools" enables you to limit workloads to specific subsets of the machine.

Configurations that use partitioning can avoid system-wide overcommitment. However, in avoiding this overcommitment, the ability to achieve high utilizations can be reduced. A reserved group of resources, such as processors, is not available for use by another workload when the workload bound to them is idle.

Resource Management Configuration

Portions of the resource management configuration can be placed in a network name service. This capability allows the administrator to apply resource management constraints across a collection of machines, rather than on an exclusively per-machine basis. Related work can share a common identifier, and the aggregate usage of that work can be tabulated from accounting data.

Resource management configuration and workload-oriented identifiers are described more fully in Chapter 2, "About Projects and Tasks". The extended accounting facility that links these identifiers with application resource usage is described in Chapter 4, "About Extended Accounting".

Interaction With Non-Global Zones

Resource management features can be used with zones to further refine the application environment. Interactions between these features and zones are described in applicable sections in this guide.

When to Use Resource Management

Use resource management to ensure that your applications have the required response times.

Resource management can also increase resource utilization. By categorizing and prioritizing usage, you can effectively use reserve capacity during off-peak periods, often eliminating the need for additional processing power. You can also ensure that resources are not wasted because of load variability.

Server Consolidation

Resource management is ideal for environments that consolidate a number of applications on a single server.

The cost and complexity of managing numerous machines encourages the consolidation of several applications on larger, more scalable servers. Instead of running each workload on a separate system, with full access to that system's resources, you can use resource management software to segregate workloads within the system. Resource management enables you to lower overall total cost of ownership by running and controlling several dissimilar applications on a single Oracle Solaris system.

If you are providing Internet and application services, you can use resource management to do the following:

- Host multiple web servers on a single machine. You can control the resource consumption for each web site and you can protect each site from the potential excesses of other sites.
- Prevent a faulty common gateway interface (CGI) script from exhausting CPU resources.
- Stop an incorrectly behaving application from leaking all available virtual memory.
- Ensure that one customer's applications are not affected by another customer's applications that run at the same site.
- Provide differentiated levels or classes of service on the same machine.
- Obtain accounting information for billing purposes.

Supporting a Large or Varied User Population

Use resource management features in any system that has a large, diverse user base, such as an educational institution. If you have a mix of workloads, the software can be configured to give priority to specific projects.

For example, in large brokerage firms, traders intermittently require fast access to execute a query or to perform a calculation. Other system users, however, have more consistent workloads. If you allocate a proportionately larger amount of processing power to the traders' projects, the traders have the responsiveness that they need.

Resource management is also ideal for supporting thin-client systems. These platforms provide stateless consoles with frame buffers and input devices, such as smart cards. The actual computation is done on a shared server, resulting in a timesharing type of environment. Use resource management features to isolate the users on the server. Then, a user who generates excess load does not monopolize hardware resources and significantly impact others who use the system.

Setting Up Resource Management Task Map

The following task map provides a high-level overview of the steps to set up resource management on your system.

Task	Description	For Instructions
Identify the workloads on your system and categorize each workload by project.	Create project entries in either the `/etc/project` file, in the NIS map, or in the LDAP directory service.	"`project` Database" on page 23
Prioritize the workloads on your system.	Determine which applications are critical. These workloads might require preferential access to resources.	Refer to your business service goals.
Monitor real-time activity on your system.	Use performance tools to view the current resource consumption of workloads that are running on your system. You can then evaluate whether you must restrict access to a given resource or isolate particular workloads from other workloads.	`cpustat`(1M), `iostat`(1M), `mpstat`(1M), `prstat`(1M), `sar`(1), and `vmstat`(1M) man pages
Make temporary modifications to the workloads that are running on your system.	To determine which values can be altered, refer to the resource controls that are available in the Oracle Solaris system. You can update the values from the command line while the task or process is running.	"Available Resource Controls" on page 62, "Global and Local Actions on Resource Control Values" on page 68, "Temporarily Updating Resource Control Values on a Running System" on page 73 and `rctladm`(1M) and `prctl`(1) man pages.
Set resource controls and project attributes for every project entry in the `project` database or naming service project database.	Each project entry in the `/etc/project` file or the naming service project database can contain one or more resource controls or attributes. Resource controls constrain tasks and processes attached to that project. For each threshold value that is placed on a resource control, you can associate one or more actions to be taken when that value is reached. You can set resource controls by using the command-line interface.	"`project` Database" on page 23, "Local `/etc/project` File Format" on page 24, "Available Resource Controls" on page 62, "Global and Local Actions on Resource Control Values" on page 68, and Chapter 8, "About Fair Share Scheduler"
Place an upper bound on the resource consumption of physical memory by collections of processes attached to a project.	The resource cap enforcement daemon will enforce the physical memory resource cap defined for the project's `rcap.max-rss` attribute in the `/etc/project` file.	"`project` Database" on page 23 and Chapter 10, "About Physical Memory Control Using the Resource Capping Daemon"
Create resource pool configurations.	Resource pools provide a way to partition system resources, such as processors, and maintain those partitions across reboots. You can add one `project.pool` attribute to each entry in the `/etc/project` file.	"`project` Database" on page 23 and Chapter 12, "About Resource Pools"

Task	Description	For Instructions
Make the fair share scheduler (FSS) your default system scheduler.	Ensure that all user processes in either a single CPU system or a processor set belong to the same scheduling class.	"Configuring the FSS" on page 99 and `dispadmin`(1M) man page
Activate the extended accounting facility to monitor and record resource consumption on a task or process basis.	Use extended accounting data to assess current resource controls and to plan capacity requirements for future workloads. Aggregate usage on a system-wide basis can be tracked. To obtain complete usage statistics for related workloads that span more than one system, the project name can be shared across several machines.	"How to Activate Extended Accounting for Flows, Processes, Tasks, and Network Components" on page 52 and `acctadm`(1M) man page
(Optional) If you need to make additional adjustments to your configuration, you can continue to alter the values from the command line. You can alter the values while the task or process is running.	Modifications to existing tasks can be applied on a temporary basis without restarting the project. Tune the values until you are satisfied with the performance. Then, update the current values in the `/etc/project` file or in the naming service project database.	"Temporarily Updating Resource Control Values on a Running System" on page 73 and `rctladm`(1M) and `prctl`(1) man pages
(Optional) Capture extended accounting data.	Write extended accounting records for active processes and active tasks. The files that are produced can be used for planning, chargeback, and billing purposes. There is also a Practical Extraction and Report Language (Perl) interface to `libexacct` that enables you to develop customized reporting and extraction scripts.	`wracct`(1M) man page and "Perl Interface to `libexacct`" on page 48

♦ ♦ ♦ CHAPTER 2

About Projects and Tasks

This chapter discusses the *project* and *task* facilities of Oracle Solaris resource management. Projects and tasks are used to label workloads and separate them from one another.

The following topics are covered in this chapter:

- "Project and Task Facilities" on page 21
- "Project Identifiers" on page 22
- "Task Identifiers" on page 27
- "Commands Used With Projects and Tasks" on page 28

To use the projects and tasks facilities, see Chapter 3, "Administering Projects and Tasks".

Project and Task Facilities

To optimize workload response, you must first be able to identify the workloads that are running on the system you are analyzing. This information can be difficult to obtain by using either a purely process-oriented or a user-oriented method alone. In the Oracle Solaris system, you have two additional facilities that can be used to separate and identify workloads: the project and the task. The *project* provides a network-wide administrative identifier for related work. The *task* collects a group of processes into a manageable entity that represents a workload component.

The controls specified in the `project` name service database are set on the process, task, and project. Since process and task controls are inherited across `fork` and `settaskid` system calls, all processes and tasks that are created within the project inherit these controls. For information on these system calls, see the `fork`(2) and `settaskid`(2) man pages.

Based on their project or task membership, running processes can be manipulated with standard Oracle Solaris commands. The extended accounting facility can report on both process usage and task usage, and tag each record with the governing project identifier. This process enables offline workload analysis to be correlated with online monitoring. The project identifier can be shared across multiple machines through the `project` name service database. Thus, the resource consumption of related workloads that run on (or span) multiple machines can ultimately be analyzed across all of the machines.

Project Identifiers

The project identifier is an administrative identifier that is used to identify related work. The project identifier can be thought of as a workload tag equivalent to the user and group identifiers. A user or group can belong to one or more projects. These projects can be used to represent the workloads in which the user (or group of users) is allowed to participate. This membership can then be the basis of chargeback that is based on, for example, usage or initial resource allocations. Although a user must be assigned to a default project, the processes that the user launches can be associated with any of the projects of which that user is a member.

Determining a User's Default Project

To log in to the system, a user must be assigned a default project. A user is automatically a member of that default project, even if the user is not in the user or group list specified in that project.

Because each process on the system possesses project membership, an algorithm to assign a default project to the login or other initial process is necessary. The algorithm is documented in the man page `getprojent`(3C). The system follows ordered steps to determine the default project. If no default project is found, the user's login, or request to start a process, is denied.

The system sequentially follows these steps to determine a user's default project:

1. If the user has an entry with a `project` attribute defined in the `/etc/user_attr` extended user attributes database, then the value of the `project` attribute is the default project. See the `user_attr`(4) man page.
2. If a project with the name `user.`*user-id* is present in the `project` database, then that project is the default project. See the `project`(4) man page for more information.
3. If a project with the name `group.`*group-name* is present in the `project` database, where *group-name* is the name of the default group for the user, as specified in the `passwd` file, then that project is the default project. For information on the `passwd` file, see the `passwd`(4) man page.
4. If the special project `default` is present in the `project` database, then that project is the default project.

This logic is provided by the `getdefaultproj` library function. See the `getprojent`(3PROJECT) man page for more information.

Setting User Attributes With the `useradd` and `usermod` Commands

You can use the following commands with the `-K` option and a *key*=*value* pair to set user attributes in local files:

`useradd`	Set default project for user
`usermod`	Modify user information

Local files can include the following:

- `/etc/group`
- `/etc/passwd`
- `/etc/project`
- `/etc/shadow`
- `/etc/user_attr`

If a network naming service such as NIS is being used to supplement the local file with additional entries, these commands cannot change information supplied by the network name service. However, the commands do verify the following against the external *naming service database*:

- Uniqueness of the user name (or role)
- Uniqueness of the user ID
- Existence of any group names specified

For more information, see the `useradd`(1M), `usermod`(1M), and `user_attr`(4) man pages.

`project` Database

You can store project data in a local file, in the Domain Name System (DNS), in a Network Information Service (NIS) project map, or in a Lightweight Directory Access Protocol (LDAP) directory service. The `/etc/project` file or naming service is used at login and by all requests for account management by the pluggable authentication module (PAM) to bind a user to a default project.

Note - Updates to entries in the project database, whether to the /etc/project file or to a representation of the database in a network naming service, are not applied to currently active projects. The updates are applied to new tasks that join the project when either the login or the newtask command is used. For more information, see the login(1) and newtask(1) man pages.

PAM Subsystem

Operations that change or set identity include logging in to the system, invoking an rcp or rsh command, using ftp, or using su. When an operation involves changing or setting an identity, a set of configurable modules is used to provide authentication, account management, credentials management, and session management.

For an overview of PAM, see Chapter 1, "Using Pluggable Authentication Modules," in "Managing Kerberos and Other Authentication Services in Oracle Solaris 11.2 ".

Naming Services Configuration

Resource management supports naming service project databases. The location where the project database is stored is defined in the /etc/nsswitch.conf file. By default, files is listed first, but the sources can be listed in any order.

```
project: files [nis] [ldap]
```

If more than one source for project information is listed, the nsswitch.conf file directs the routine to start searching for the information in the first source listed, and then search subsequent sources.

For more information about the /etc/nsswitch.conf file, see Chapter 2, "About the Name Service Switch," in "Working With Oracle Solaris 11.2 Directory and Naming Services: DNS and NIS " and nsswitch.conf(4).

Local /etc/project File Format

If you select files as your project database source in the nsswitch.conf file, the login process searches the /etc/project file for project information. See the projects(1) and project(4) man pages for more information.

The `project` file contains a one-line entry of the following form for each project recognized by the system:

```
projname:projid:comment:user-list:group-list:attributes
```

The fields are defined as follows:

projname	The name of the project. The name must be a string that consists of alphanumeric characters, underline (_) characters, hyphens (-), and periods (.). The period, which is reserved for projects with special meaning to the operating system, can only be used in the names of default projects for users. *projname* cannot contain colons (:) or newline characters.
projid	The project's unique numerical ID (PROJID) within the system. The maximum value of the *projid* field is `UID_MAX` (`2147483647`).
comment	A description of the project.
user-list	A comma-separated list of users who are allowed in the project. Wildcards can be used in this field. An asterisk (*) allows all users to join the project. An exclamation point followed by an asterisk (!*) excludes all users from the project. An exclamation mark (!) followed by a user name excludes the specified user from the project.
group-list	A comma-separated list of groups of users who are allowed in the project. Wildcards can be used in this field. An asterisk (*) allows all groups to join the project. An exclamation point followed by an asterisk (!*) excludes all groups from the project. An exclamation mark (!) followed by a group name excludes the specified group from the project.
attributes	A semicolon-separated list of name-value pairs, such as resource controls (see Chapter 6, "About Resource Controls"). *name* is an arbitrary string that specifies the object-related attribute, and *value* is the optional value for that attribute. `name[=`*value*`]` In the name-value pair, names are restricted to letters, digits, underscores, and periods. A period is conventionally used as a separator between the categories and subcategories of the resource control (rctl). The first character of an attribute name must be a letter. The name is case sensitive. Values can be structured by using commas and parentheses to establish precedence.

A semicolon is used to separate name-value pairs. A semicolon cannot be used in a value definition. A colon is used to separate project fields. A colon cannot be used in a value definition.

Note - Routines that read this file halt if they encounter a malformed entry. Any projects that are specified after the incorrect entry are not assigned.

This example shows the default `/etc/project` file:

```
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
```

This example shows the default `/etc/project` file with project entries added at the end:

```
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
user.ml:2424:Lyle Personal:::
booksite:4113:Book Auction Project:ml,mp,jtd,kjh::
```

You can also add resource controls and attributes to the `/etc/project` file:

- To add resource controls for a project, see "Setting Resource Controls" on page 76.
- To define a physical memory resource cap for a project using the resource capping daemon described in `rcapd`(1M), see "Attribute to Limit Physical Memory Usage for Projects" on page 104.
- To add a `project.pool` attribute to a project's entry, see "Creating the Configuration" on page 166.

Project Configuration for NIS

If you are using NIS, you can specify in the `/etc/nsswitch.conf` file to search the NIS project maps for projects:

```
project: nis files
```

The NIS maps, either `project.byname` or `project.bynumber`, have the same form as the `/etc/project` file:

```
projname:projid:comment:user-list:group-list:attributes
```

For more information, see Chapter 5, "About the Network Information Service," in "Working With Oracle Solaris 11.2 Directory and Naming Services: DNS and NIS ".

Project Configuration for LDAP

If you are using LDAP, you can specify in the `/etc/nsswitch.conf` file to search the LDAP `project` database for projects:

```
project: ldap files
```

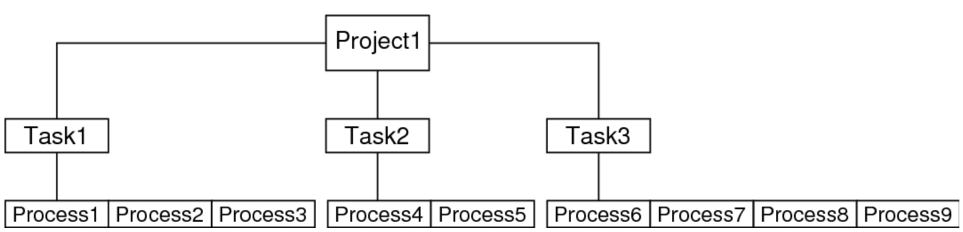
For more information about LDAP, see Chapter 1, "Introduction to the LDAP Naming Service," in "Working With Oracle Solaris 11.2 Directory and Naming Services: LDAP ". For more information about the schema for project entries in an LDAP database, see "Oracle Solaris Schemas" in "Working With Oracle Solaris 11.2 Directory and Naming Services: LDAP ".

Task Identifiers

Each successful login into a project creates a new *task* that contains the login process. The task is a process collective that represents a set of work over time. A task can also be viewed as a *workload component*. Each task is automatically assigned a task ID.

Each process is a member of one task, and each task is associated with one project.

FIGURE 2-1 Project and Task Tree



All operations on process groups, such as signal delivery, are also supported on tasks. You can also bind a task to a *processor set* and set a scheduling priority and class for a task, which modifies all current and subsequent processes in the task.

A task is created whenever a project is joined. The following actions, commands, and functions create tasks:

- `login`

- `cron`
- `newtask`
- `setproject`
- `su`

You can create a finalized task by using one of the following methods. All further attempts to create new tasks will fail.

- You can use the `newtask` command with the `-F` option.
- You can set the `task.final` attribute on a project in the `project` naming service database. All tasks created in that project by `setproject` have the `TASK_FINAL` flag.

For more information, see the `login`(1), `newtask`(1), `cron`(1M), `su`(1M), and `setproject`(3PROJECT) man pages.

The extended accounting facility can provide accounting data for processes. The data is aggregated at the task level.

Commands Used With Projects and Tasks

The commands that are shown in the following table provide the primary administrative interface to the project and task facilities.

Man Page Reference	Description
`projects`(1)	Displays project memberships for users. Lists projects from `project` database. Prints information on given projects. If no project names are supplied, information is displayed for all projects. Use the `projects` command with the `-l` option to print verbose output.
`newtask`(1)	Executes the user's default shell or specified command, placing the execution command in a new task that is owned by the specified project. `newtask` can also be used to change the task and the project binding for a running process. Use with the `-F` option to create a finalized task.
`projadd`(1M)	Adds a new project entry to the `/etc/project` file. The `projadd` command creates a project entry only on the local system. `projadd` cannot change information that is supplied by the network naming service. Can be used to edit project files other than the default file, `/etc/project`. Provides syntax checking for `project` file. Validates and edits project attributes. Supports scaled values.
`projmod`(1M)	Modifies information for a project on the local system. `projmod` cannot change information that is supplied by the network naming service. However, the command does verify the uniqueness of the project name and project ID against the external naming service.

Man Page Reference	Description
	Can be used to edit project files other than the default file, `/etc/project`. Provides syntax checking for `project` file. Validates and edits project attributes. Can be used to add a new attribute, add values to an attribute, or remove an attribute. Supports scaled values. Can be used with the `-A` option to apply the resource control values found in the project database to the active project. Existing values that do not match the values defined in the `project` file are removed.
`projdel`(1M)	Deletes a project from the local system. `projdel` cannot change information that is supplied by the network naming service.
`useradd`(1M)	Adds default project definitions to the local files. Use with the `-K` *key*=*value* option to add or replace user attributes.
`userdel`(1M)	Deletes a user's account from the local file.
`usermod`(1M)	Modifies a user's login information on the system. Use with the `-K` *key*=*value* option to add or replace user attributes.

♦ ♦ ♦ CHAPTER 3

Administering Projects and Tasks

This chapter describes how to use the project and task facilities of Oracle Solaris resource management.

The following topics are covered.

- "Example Commands and Command Options" on page 32
- "Administering Projects" on page 35

For an overview of the projects and tasks facilities, see Chapter 2, "About Projects and Tasks".

Note - If you are using these facilities on an Oracle Solaris system with zones installed, only processes in the same zone are visible through system call interfaces that take process IDs when these commands are run in a non-global zone.

Administering Projects and Tasks Task Map

Task	Description	For Instructions
View examples of commands and options used with projects and tasks.	Display task and project IDs, display various statistics for processes and projects that are currently running on your system.	"Example Commands and Command Options" on page 32
Define a project.	Add a project entry to the `/etc/project` file and alter values for that entry.	"How to Define a Project and View the Current Project" on page 35
Delete a project.	Remove a project entry from the `/etc/project` file.	"How to Delete a Project From the `/etc/project` File" on page 37
Validate the `project` file or project database.	Check the syntax of the `/etc/project` file or verify the uniqueness of the project name and project ID against the external naming service.	"How to Validate the Contents of the `/etc/project` File" on page 38

Task	Description	For Instructions
Obtain project membership information.	Display the current project membership of the invoking process.	"How to Obtain Project Membership Information" on page 38
Create a new task.	Create a new task in a particular project by using the `newtask` command.	"How to Create a New Task" on page 39
Associate a running process with a different task and project.	Associate a process number with a new task ID in a specified project.	"How to Move a Running Process Into a New Task" on page 39
Add and work with project attributes.	Use the project database administration commands to add, edit, validate, and remove project attributes.	"Editing and Validating Project Attributes" on page 40

Example Commands and Command Options

This section provides examples of commands and options used with projects and tasks.

Command Options Used With Projects and Tasks

`ps` Command

Use the `ps` command with the `-o` option to display task and project IDs. For example, to view the project ID, type the following:

```
# ps -o user,pid,uid,projid
USER PID   UID  PROJID
jtd  89430 124  4113
```

`id` Command

Use the `id` command with the `-p` option to print the current project ID in addition to the user and group IDs. If the `user` operand is provided, the project associated with that user's normal login is printed:

```
#  id -p
uid=124(jtd) gid=10(staff) projid=4113(booksite)
```

`pgrep` and `pkill` Commands

To match only processes with a project ID in a specific list, use the `pgrep` and `pkill` commands with the `-J` option:

```
# pgrep -J projidlist
# pkill -J projidlist
```

To match only processes with a task ID in a specific list, use the `pgrep` and `pkill` commands with the `-T` option:

```
# pgrep -T taskidlist
# pkill -T taskidlist
```

`prstat` Command

To display various statistics for processes and projects that are currently running on your system, use the `prstat` command with the `-J` option:

```
% prstat -J
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 12905 root     4472K 3640K cpu0    59    0   0:00:01 0.4% prstat/1
   829 root       43M   33M sleep   59    0   0:36:23 0.1% Xorg/1
   890 gdm        88M   26M sleep   59    0   0:22:22 0.0% gdm-simple-gree/1
   686 root     3584K 2756K sleep   59    0   0:00:34 0.0% automountd/4
     5 root        0K    0K sleep   99  -20   0:02:43 0.0% zpool-rpool/138
  9869 root       44M   17M sleep   59    0   0:02:06 0.0% poold/9
   804 root     7104K 5968K sleep   59    0   0:01:28 0.0% intrd/1
   445 root     7204K 4680K sleep   59    0   0:00:38 0.0% nscd/33
   881 gdm      7140K 5912K sleep   59    0   0:00:06 0.0% gconfd-2/1
   164 root     2572K 1648K sleep   59    0   0:00:00 0.0% pfexecd/3
   886 gdm      7092K 4920K sleep   59    0   0:00:00 0.0% bonobo-activati/2
    45 netcfg   2252K 1308K sleep   59    0   0:00:00 0.0% netcfgd/2
   142 daemon   7736K 5224K sleep   59    0   0:00:00 0.0% kcfd/3
    43 root     3036K 2020K sleep   59    0   0:00:00 0.0% dlmgmtd/5
   405 root     6824K 5400K sleep   59    0   0:00:18 0.0% hald/5
PROJID    NPROC  SWAP   RSS MEMORY      TIME  CPU PROJECT
     1        4 4728K   19M   0.9%   0:00:01 0.4% user.root
     0      111  278M  344M    17%   1:15:02 0.1% system
    10        2 1884K 9132K   0.4%   0:00:00 0.0% group.staff
     3        3 1668K 6680K   0.3%   0:00:00 0.0% default

Total: 120 processes, 733 lwps, load averages: 0.01, 0.00, 0.00
```

To display various statistics for processes and tasks that are currently running on your system, use the `prstat` command with the `-T` option:

```
% prstat -T
       PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 12907 root     4488K 3588K cpu0    59    0   0:00:00 0.3% prstat/1
   829 root       43M   33M sleep   59    0   0:36:24 0.1% Xorg/1
```

```
   890 gdm        88M   26M sleep   59    0   0:22:22 0.0% gdm-simple-gree/1
  9869 root       44M   17M sleep   59    0   0:02:06 0.0% poold/9
     5 root        0K    0K sleep   99  -20   0:02:43 0.0% zpool-rpool/138
   445 root     7204K 4680K sleep   59    0   0:00:38 0.0% nscd/33
   881 gdm      7140K 5912K sleep   59    0   0:00:06 0.0% gconfd-2/1
   164 root     2572K 1648K sleep   59    0   0:00:00 0.0% pfexecd/3
   886 gdm      7092K 4920K sleep   59    0   0:00:00 0.0% bonobo-activati/2
    45 netcfg   2252K 1308K sleep   59    0   0:00:00 0.0% netcfgd/2
   142 daemon   7736K 5224K sleep   59    0   0:00:00 0.0% kcfd/3
    43 root     3036K 2020K sleep   59    0   0:00:00 0.0% dlmgmtd/5
   405 root     6824K 5400K sleep   59    0   0:00:18 0.0% hald/5
   311 root     3488K 2512K sleep   59    0   0:00:00 0.0% picld/4
   409 root     4356K 2768K sleep   59    0   0:00:00 0.0% hald-addon-cpuf/1
TASKID    NPROC  SWAP   RSS MEMORY      TIME  CPU PROJECT
  1401        2 2540K 8120K   0.4%   0:00:00 0.3% user.root
    94       15   84M  162M   7.9%   0:59:37 0.1% system
   561        1   37M   24M   1.2%   0:02:06 0.0% system
     0        2    0K    0K   0.0%   0:02:47 0.0% system
    46        1 4224K 5524K   0.3%   0:00:38 0.0% system
Total: 120 processes, 733 lwps, load averages: 0.01, 0.00, 0.00
```

Note - The `-J` and `-T` options cannot be used together.

Using `cron` and `su` With Projects and Tasks

`cron` Command

The `cron` command issues a `settaskid` to ensure that each `cron`, `at`, and `batch` job executes in a separate task, with the appropriate default project for the submitting user. The `at` and `batch` commands also capture the current project ID, which ensures that the project ID is restored when running an `at` job.

`su` Command

The `su` command joins the target user's default project by creating a new task, as part of simulating a login.

To switch the user's default project by using the `su` command, type the following:

\# **su -** *user*

Administering Projects

How to Define a Project and View the Current Project

This example shows how to use the `projadd` command to add a project entry and the `projmod` command to alter that entry.

1. **Become root or assume an equivalent role.**

2. **View the default `/etc/project` file on your system by using `projects -l`.**

```
# projects -l
system
        projid : 0
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
user.root
        projid : 1
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
noproject
        projid : 2
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
default
        projid : 3
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
group.staff
        projid : 10
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
```

3. **Add a project with the name *booksite*. Assign the project to a user who is named *mark* with project ID number *4113*.**

```
# projadd -U mark -p 4113 booksite
```

4. **View the `/etc/project` file again.**

```
# projects -l
system
        projid : 0
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
user.root
        projid : 1
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
noproject
        projid : 2
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
default
        projid : 3
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
group.staff
        projid : 10
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
booksite
        projid : 4113
        comment: ""
        users  : mark
        groups : (none)
        attribs:
```

5. **Add a comment that describes the project in the comment field.**

```
# projmod -c `Book Auction Project' booksite
```

6. **View the changes in the `/etc/project` file.**

```
# projects -l
system
        projid : 0
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
user.root
        projid : 1
        comment: ""
```

```
        users  : (none)
        groups : (none)
        attribs:
noproject
        projid : 2
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
default
        projid : 3
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
group.staff
        projid : 10
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
booksite
        projid : 4113
        comment: "Book Auction Project"
        users  : mark
        groups : (none)
        attribs:
```

See Also To bind projects, tasks, and processes to a pool, see "Setting Pool Attributes and Binding to a Pool" on page 160.

How to Delete a Project From the `/etc/project` File

This example shows how to use the `projdel` command to delete a project.

1. **Become root or assume an equivalent role.**
2. **Remove the project *booksite* by using the `projdel` command.**

   ```
   # projdel booksite
   ```

3. **Display the `/etc/project` file.**

```
# projects -l
system
        projid : 0
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
user.root
```

```
        projid : 1
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
noproject
        projid : 2
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
default
        projid : 3
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
group.staff
        projid : 10
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
```

4. **Log in as user *mark* and type `projects` to view the projects that are assigned to this user.**

```
# su - mark
# projects
default
```

How to Validate the Contents of the `/etc/project` File

If no editing options are given, the `projmod` command validates the contents of the `project` file.

To validate a NIS map, type the following:

```
# ypcat project | projmod -f —
```

To check the syntax of the `/etc/project` file, type the following:

```
# projmod -n
```

How to Obtain Project Membership Information

Use the `id` command with the `-p` flag to display the current project membership of the invoking process.

```
$ id -p
uid=100(mark) gid=1(other) projid=3(default)
```

How to Create a New Task

1. **Log in as a member of the destination project,** ***booksite*** **in this example.**
2. **Create a new task in the** ***booksite*** **project by using the `newtask` command with the `-v` (verbose) option to obtain the system task ID.**

   ```
   machine% newtask -v -p booksite
   16
   ```

   The execution of `newtask` creates a new task in the specified project, and places the user's default shell in this task.
3. **View the current project membership of the invoking process.**

   ```
   machine% id -p
   uid=100(mark) gid=1(other) projid=4113(booksite)
   ```

   The process is now a member of the new project.

How to Move a Running Process Into a New Task

This example shows how to associate a running process with a different task and new project. To perform this action, you must be the root user, have the required rights profile, or be the owner of the process and be a member of the new project.

1. **Become root or assume an equivalent role.**

   Note - If you are the owner of the process or a member of the new project, you can skip this step.

   ---
2. **Obtain the process ID of the** ***book_catalog*** **process.**

   ```
   # pgrep book_catalog
    8100
   ```
3. **Associate process** ***8100*** **with a new task ID in the** ***booksite*** **project.**

   ```
   # newtask -v -p booksite -c 8100
    17
   ```

The -c option specifies that newtask operate on the existing named process.

4. **Confirm the task to process ID mapping.**

```
# pgrep -T 17
 8100
```

Editing and Validating Project Attributes

You can use the projadd and projmod project database administration commands to edit project attributes.

The -K option specifies a replacement list of attributes. Attributes are delimited by semicolons (;). If the -K option is used with the -a option, the attribute or attribute value is added. If the -K option is used with the -r option, the attribute or attribute value is removed. If the -K option is used with the -s option, the attribute or attribute value is substituted.

▼ How to Add Attributes and Attribute Values to Projects

Use the projmod command with the -a and -K options to add values to a project attribute. If the attribute does not exist, it is created.

1. **Become root or assume an equivalent role.**

2. **Add a task.max-lwps resource control attribute with no values in the project *myproject*. A task entering the project has only the system value for the attribute.**

```
# projmod -a -K task.max-lwps myproject
```

3. **You can then add a value to task.max-lwps in the project *myproject*. The value consists of a privilege level, a threshold value, and an action associated with reaching the threshold.**

```
# projmod -a -K "task.max-lwps=(priv,100,deny)" myproject
```

4. **Because resource controls can have multiple values, you can add another value to the existing list of values by using the same options.**

```
# projmod -a -K "task.max-lwps=(priv,1000,signal=KILL)" myproject
```

The multiple values are separated by commas. The task.max-lwps entry now reads:

```
task.max-lwps=(priv,100,deny),(priv,1000,signal=KILL)
```

How to Remove Attribute Values From Projects

This procedure uses the values:

```
task.max-lwps=(priv,100,deny),(priv,1000,signal=KILL)
```

1. **Become root or assume an equivalent role.**
2. **To remove an attribute value from the resource control `task.max-lwps` in the project *myproject*, use the `projmod` command with the `-r` and `-K` options.**

```
# projmod -r -K "task.max-lwps=(priv,100,deny)" myproject
```

If `task.max-lwps` has multiple values, such as:

```
task.max-lwps=(priv,100,deny),(priv,1000,signal=KILL)
```

The first matching value would be removed. The result would then be:

```
task.max-lwps=(priv,1000,signal=KILL)
```

How to Remove a Resource Control Attribute From a Project

To remove the resource control `task.max-lwps` in the project *myproject*, use the `projmod` command with the `-r` and `-K` options.

1. **Become root or assume an equivalent role.**
2. **Remove the attribute `task.max-lwps` and all of its values from the project *myproject*:**

```
# projmod -r -K task.max-lwps myproject
```

How to Substitute Attributes and Attribute Values for Projects

To substitute a different value for the attribute `task.max-lwps` in the project *myproject*, use the `projmod` command with the `-s` and `-K` options. If the attribute does not exist, it is created.

1. **Become root or assume an equivalent role.**
2. **Replace the current `task.max-lwps` values with the new values shown:**

```
# projmod -s -K "task.max-lwps=(priv,100,none),(priv,120,deny)" myproject
```

The result would be:

```
task.max-lwps=(priv,100,none),(priv,120,deny)
```

How to Remove the Existing Values for a Resource Control Attribute

1. **Become root or assume an equivalent role.**
2. **To remove the current values for `task.max-lwps` from the project *myproject*, type:**

```
# projmod -s -K task.max-lwps myproject
```

CHAPTER 4

About Extended Accounting

By using the project and task facilities that are described in Chapter 2, "About Projects and Tasks" to label and separate workloads, you can monitor resource consumption by each workload. You can use the *extended accounting* subsystem to capture a detailed set of resource consumption statistics on both processes and tasks.

The following topics are covered in this chapter.

- "Introduction to Extended Accounting" on page 43
- "How Extended Accounting Works" on page 44
- "Extended Accounting Configuration" on page 46
- "Commands Used With Extended Accounting" on page 47
- "Perl Interface to `libexacct`" on page 48

To begin using extended accounting, skip to "How to Activate Extended Accounting for Flows, Processes, Tasks, and Network Components" on page 52.

Introduction to Extended Accounting

The extended accounting subsystem labels usage records with the project for which the work was done. You can also use extended accounting, in conjunction with the Internet Protocol Quality of Service (IPQoS) flow accounting module, to capture network flow information on a system.

Before you can apply resource management mechanisms, you must first be able to characterize the resource consumption demands that various workloads place on a system. The extended accounting facility in the Oracle Solaris operating system provides a flexible way to record system and network resource consumption for the following:

- Tasks.
- Processes.
- Selectors provided by the IPQoS `flowacct` module. For more information, see `ipqos`(7IPP).

- Network management. See `dladm`(1M) and `flowadm`(1M).

Unlike online monitoring tools, which enable you to measure system usage in real time, extended accounting enables you to examine historical usage. You can then make assessments of capacity requirements for future workloads.

With extended accounting data available, you can develop or purchase software for resource chargeback, workload monitoring, or capacity planning.

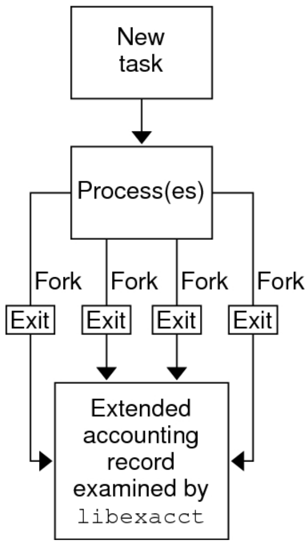
How Extended Accounting Works

The extended accounting facility in the Oracle Solaris operating system uses a versioned, extensible file format to contain accounting data. Files that use this data format can be accessed or be created by using the API provided in the included library, `libexacct` (see `libexacct`(3LIB)). These files can then be analyzed on any platform with extended accounting enabled, and their data can be used for capacity planning and chargeback.

If extended accounting is active, statistics are gathered that can be examined by the `libexacct` API. `libexacct` allows examination of the `exacct` files either forward or backward. The API supports third-party files that are generated by `libexacct` as well as those files that are created by the kernel. There is a Practical Extraction and Report Language (Perl) interface to `libexacct` that enables you to develop customized reporting and extraction scripts. See "Perl Interface to `libexacct`" on page 48.

For example, with extended accounting enabled, the task tracks the aggregate resource usage of its member processes. A task accounting record is written at task completion. Interim records on running processes and tasks can also be written. For more information on tasks, see Chapter 2, "About Projects and Tasks".

FIGURE 4-1 Task Tracking With Extended Accounting Activated



Extensible Format

The extended accounting format is substantially more extensible than the legacy system accounting software format. Extended accounting permits accounting metrics to be added and removed from the system between releases, and even during system operation.

Note - Both extended accounting and legacy system accounting software can be active on your system at the same time.

`exacct` Records and Format

Routines that allow `exacct` records to be created serve two purposes.

- To enable third-party `exacct` files to be created.
- To enable the creation of tagging records to be embedded in the kernel accounting file by using the `putacct` system call (see `getacct(2)`).

Note - The `putacct` system call is also available from the Perl interface.

The format permits different forms of accounting records to be captured without requiring that every change be an explicit version change. Well-written applications that consume accounting data must ignore records they do not understand.

The `libexacct` library converts and produces files in the `exacct` format. This library is the *only* supported interface to `exacct` format files.

Note - The `getacct`, `putacct`, and `wracct` system calls do not apply to flows. The kernel creates flow records and writes them to the file when IPQoS flow accounting is configured.

Using Extended Accounting on an Oracle Solaris System with Zones Installed

The extended accounting subsystem collects and reports information for the entire system (including non-global zones) when run in the global zone. The global administrator or a user granted appropriate authorizations through the `zonecfg` utility can also determine resource consumption on a per-zone basis. See Chapter 2, "Non-Global Zone Configuration Overview," in "Introduction to Oracle Solaris Zones " for more information.

Extended Accounting Configuration

The directory `/var/adm/exacct` is the standard location for placing extended accounting data. You can use the `acctadm` command to specify a different location for the process and task accounting-data files. See `acctadm`(1M) for more information.

Starting and Persistently Enabling Extended Accounting

The `acctadm` command described in `acctadm`(1M) starts extended accounting through the Oracle Solaris service management facility (SMF) service described in `smf`(5).

The extended accounting configuration is stored in the SMF repository. The configuration is restored at boot by a service instance, one for each accounting type. Each of the extended accounting types is represented by a separate instance of the SMF service:

`svc:/system/extended-accounting:flow`	Flow accounting
`svc:/system/extended-accounting:process`	Process accounting
`svc:/system/extended-accounting:task`	Task accounting
`svc:/system/extended-accounting:net`	Network accounting

Enabling extended accounting by using `acctadm`(1M) causes the corresponding service instance to be enabled if not currently enabled, so that the extended accounting configuration will be restored at the next boot. Similarly, if the configuration results in accounting being disabled for a service, the service instance will be disabled. The instances are enabled or disabled by `acctadm` as needed.

To permanently activate extended accounting for a resource, run:

`#` **`acctadm -e`** *resource_list*

resource_list is a comma-separated list of resources or resource groups.

Records

The `acctadm` command appends new records to an existing file in `/var/adm/exacct`.

Commands Used With Extended Accounting

Command Reference	Description
`acctadm`(1M)	Modifies various attributes of the extended accounting facility, stops and starts extended accounting, and is used to select accounting attributes to track for processes, tasks, flows and network.
`wracct`(1M)	Writes extended accounting records for active processes and active tasks.
`lastcomm`(1)	Displays previously invoked commands. `lastcomm` can consume either standard accounting-process data or extended-accounting process data.

For information on commands that are associated with tasks and projects, see "Example Commands and Command Options" on page 32. For information on IPQoS flow accounting, see the `ipqosconf`(1M) man page.

Perl Interface to `libexacct`

The Perl interface allows you to create Perl scripts that can read the accounting files produced by the `exacct` framework. You can also create Perl scripts that write `exacct` files.

The interface is functionally equivalent to the underlying C API. When possible, the data obtained from the underlying C API is presented as Perl data types. This interface allows easier access to the data, and removes the need for buffer pack and unpack operations. Moreover, all memory management is performed by the Perl library.

The various project, task, and `exacct`-related functions are separated into groups. Each group of functions is located in a separate Perl module. Each module begins with Oracle Solaris standard `Sun::Solaris::` Perl package prefix. All of the classes provided by the Perl `exacct` library are found under the `Sun::Solaris::Exacct` module.

The underlying `libexacct`(3LIB) library provides operations on `exacct` format files, catalog tags, and `exacct` objects. `exacct` objects are subdivided into two types:

- Items, which are single-data values (scalars)
- Groups, which are lists of Items

The following table summarizes each of the modules.

Module (should not contain spaces)	Description	For More Information
`Sun::Solaris::Project`	This module provides functions to access the project manipulation functions `getprojid`(2), `endprojent`(3PROJECT), `fgetprojent`(3PROJECT), `getdefaultproj`(3PROJECT), `getprojbyid`(3PROJECT), `getprojbyname`(3PROJECT), `getprojent`(3PROJECT), `getprojidbyname`(3PROJECT), `inproj`(3PROJECT), `project_walk`(3PROJECT), `setproject`(3PROJECT), and `setprojent`(3PROJECT).	`Project`(3PERL)

Module (should not contain spaces)	Description	For More Information
Sun::Solaris::Task	This module provides functions to access the task manipulation functions gettaskid(2) and settaskid(2).	Task(3PERL)
Sun::Solaris::Exacct	This module is the top-level exacct module. This module provides functions to access the exacct-related system calls getacct(2), putacct(2), and wracct(2). This module also provides functions to access the libexacct(3LIB) library function ea_error(3EXACCT). Constants for all of the exacct EO_*, EW_*, EXR_*, P_*, and TASK_* macros are also provided in this module.	Exacct(3PERL)
Sun::Solaris::Exacct:: Catalog	This module provides object-oriented methods to access the bitfields in an exacct catalog tag. This module also provides access to the constants for the EXC_*, EXD_*, and EXD_* macros.	Exacct::Catalog(3PERL)
Sun::Solaris::Exacct:: File	This module provides object-oriented methods to access the libexacct accounting file functions ea_open(3EXACCT), ea_close(3EXACCT), ea_get_creator(3EXACCT), ea_get_hostname(3EXACCT), ea_next_object(3EXACCT), ea_previous_object(3EXACCT), and ea_write_object(3EXACCT).	Exacct::File(3PERL)
Sun::Solaris::Exacct:: Object	This module provides object-oriented methods to access an individual exacct accounting file object. An exacct object is represented as an opaque reference blessed into the appropriate Sun::Solaris::Exacct::Object subclass. This module is further subdivided into the object types Item and Group. At this level, there are methods to access the ea_match_object_catalog(3EXACCT) and ea_attach_to_object(3EXACCT) functions.	Exacct::Object(3PERL)
Sun::Solaris::Exacct:: Object::Item	This module provides object-oriented methods to access an individual exacct accounting file Item. Objects of this type inherit from Sun::Solaris::Exacct::Object.	Exacct::Object::Item(3PERL)
Sun::Solaris::Exacct:: Object::Group	This module provides object-oriented methods to access an individual exacct accounting file Group. Objects of this type inherit from Sun::Solaris::Exacct::Object. These objects provide access to the ea_attach_to_group(3EXACCT) function. The Items contained within the Group are presented as a Perl array.	Exacct::Object::Group(3PERL)

Module (should not contain spaces)	Description	For More Information
Sun::Solaris::Kstat	This module provides a Perl tied hash interface to the kstat facility. A usage example for this module can be found in /bin/kstat, which is written in Perl.	Kstat(3PERL)

For examples that show how to use the modules described in the previous table, see "Using the Perl Interface to libexacct" on page 55.

♦ ♦ ♦ CHAPTER 5

Administering Extended Accounting Tasks

This chapter describes how to administer the extended accounting subsystem.

For an overview of the extending accounting subsystem, see Chapter 4, "About Extended Accounting".

Administering the Extended Accounting Facility Task Map

Task	Description	For Instructions
Activate the extended accounting facility.	Use extended accounting to monitor resource consumption by each project running on your system. You can use the *extended accounting* subsystem to capture historical data for tasks, processes, and flows.	"How to Activate Extended Accounting for Flows, Processes, Tasks, and Network Components" on page 52
Display extended accounting status.	Determine the status of the extended accounting facility.	"How to Display Extended Accounting Status" on page 53
View available accounting resources.	View the accounting resources available on your system.	"How to View Available Accounting Resources" on page 53
Deactivate the flow, process, task, and net accounting instances.	Turn off the extended accounting functionality.	"How to Deactivate Process, Task, Flow, and Network Management Accounting" on page 54
Use the Perl interface to the extended accounting facility.	Use the Perl interface to develop customized reporting and extraction scripts.	"Using the Perl Interface to `libexacct`" on page 55

Using Extended Accounting Functionality

Users can manage extended accounting (start accounting, stop accounting, and change accounting configuration parameters) if they have the appropriate rights profile for the accounting type to be managed:

- Extended Accounting Flow Management
- Process Management
- Task Management
- Network Management

How to Activate Extended Accounting for Flows, Processes, Tasks, and Network Components

To activate the extended accounting facility for tasks, processes, flows, and network components, use the `acctadm` command. The optional final parameter to `acctadm` indicates whether the command should act on the flow, process, system task, or network accounting components of the extended accounting facility.

Note - Roles contain authorizations and privileged commands. For information on how to create the role and assign the role to a user through the role-based access control (RBAC) feature of Oracle Solaris, see "Securing Users and Processes in Oracle Solaris 11.2 ".

---

1. **Become root or assume an equivalent role.**
2. **Activate extended accounting for processes.**

```
# acctadm -e extended -f /var/adm/exacct/proc process
```

3. **Activate extended accounting for tasks.**

```
# acctadm -e extended,mstate -f /var/adm/exacct/task task
```

4. **Activate extended accounting for flows.**

```
# acctadm -e extended -f /var/adm/exacct/flow flow
```

5. **Activate extended accounting for network.**

```
# acctadm -e extended -f /var/adm/exacct/net net
```

Run `acctadm` on links and flows administered by the `dladm` and `flowadm` commands.

See Also See `acctadm`(1M) for more information.

How to Display Extended Accounting Status

Type `acctadm` without arguments to display the current status of the extended accounting facility.

```
machine% acctadm
                 Task accounting: active
            Task accounting file: /var/adm/exacct/task
          Tracked task resources: extended
        Untracked task resources: none
              Process accounting: active
         Process accounting file: /var/adm/exacct/proc
       Tracked process resources: extended
     Untracked process resources: host
                 Flow accounting: active
            Flow accounting file: /var/adm/exacct/flow
          Tracked flow resources: extended
        Untracked flow resources: none
```

In the previous example, system task accounting is active in extended mode and `mstate` mode. Process and flow accounting are active in extended mode.

Note - In the context of extended accounting, microstate (`mstate`) refers to the extended data, associated with microstate process transitions, that is available in the process usage file (see `proc(4)`). This data provides substantially more detail about the activities of the process than basic or extended records.

How to View Available Accounting Resources

Available resources can vary from system to system, and from platform to platform. Use the `acctadm` command with the `-r` option to view the accounting resource groups available on your system.

```
machine% acctadm -r
process:
extended pid,uid,gid,cpu,time,command,tty,projid,taskid,ancpid,wait-status,zone,flag,
memory,mstate     displays as one line
basic    pid,uid,gid,cpu,time,command,tty,flag
task:
extended taskid,projid,cpu,time,host,mstate,anctaskid,zone
basic    taskid,projid,cpu,time
flow:
extended
saddr,daddr,sport,dport,proto,dsfield,nbytes,npkts,action,ctime,lseen,projid,uid
basic    saddr,daddr,sport,dport,proto,nbytes,npkts,action
net:
```

```
extended name,devname,edest,vlan_tpid,vlan_tci,sap,cpuid, \
priority,bwlimit,curtime,ibytes,obytes,ipkts,opks,ierrpkts \
oerrpkts,saddr,daddr,sport,dport,protocol,dsfield
basic    name,devname,edest,vlan_tpid,vlan_tci,sap,cpuid, \
priority,bwlimit,curtime,ibytes,obytes,ipkts,opks,ierrpkts \
oerrpkts
```

▼ How to Deactivate Process, Task, Flow, and Network Management Accounting

To deactivate process, task, flow, and network accounting, turn off each of them individually by using the `acctadm` command with the `-x` option.

1. **Become root or assume an equivalent role.**

2. **Turn off process accounting.**

```
# acctadm -x process
```

3. **Turn off task accounting.**

```
# acctadm -x task
```

4. **Turn off flow accounting.**

```
# acctadm -x flow
```

5. **Turn off network management accounting.**

```
# acctadm -x net
```

6. **Verify that task accounting, process accounting, flow and network accounting have been turned off.**

```
 # acctadm
            Task accounting: inactive
       Task accounting file: none
     Tracked task resources: none
   Untracked task resources: extended
         Process accounting: inactive
    Process accounting file: none
  Tracked process resources: none
Untracked process resources: extended
            Flow accounting: inactive
       Flow accounting file: none
     Tracked flow resources: none
   Untracked flow resources: extended
            Net accounting: inactive
       Net accounting file: none
     Tracked Net resources: none
```

```
Untracked Net resources: extended
```

Using the Perl Interface to `libexacct`

How to Recursively Print the Contents of an `exacct` Object

Use the following code to recursively print the contents of an `exacct` object. Note that this capability is provided by the library as the `Sun::Solaris::Exacct::Object::dump()` function. This capability is also available through the `ea_dump_object()` convenience function.

```
sub dump_object
     {
             my ($obj, $indent) = @_;
             my $istr = '  ' x $indent;

             #
             # Retrieve the catalog tag.  Because we are
             # doing this in an array context, the
             # catalog tag will be returned as a (type, catalog, id)
             # triplet, where each member of the triplet will behave as
             # an integer or a string, depending on context.
             # If instead this next line provided a scalar context, e.g.
             #    my $cat  = $obj->catalog()->value();
             # then $cat would be set to the integer value of the
             # catalog tag.
             #
             my @cat = $obj->catalog()->value();

             #
             # If the object is a plain item
             #
             if ($obj->type() == &EO_ITEM) {
                     #
                     # Note: The '%s' formats provide s string context, so
                     # the components of the catalog tag will be displayed
                     # as the symbolic values. If we changed the '%s'
                     # formats to '%d', the numeric value of the components
                     # would be displayed.
                     #
                     printf("%sITEM\n%s  Catalog = %s|%s|%s\n",
                        $istr, $istr, @cat);
                     $indent++;

                     #
                     # Retrieve the value of the item.  If the item contains
                     # in turn a nested exacct object (i.e., an item or
                     # group),then the value method will return a reference
                     # to the appropriate sort of perl object
                     # (Exacct::Object::Item or Exacct::Object::Group).
```

```
                # We could of course figure out that the item contained
                # a nested item orgroup by examining the catalog tag in
                # @cat and looking for a type of EXT_EXACCT_OBJECT or
                # EXT_GROUP.
                #
                my $val = $obj->value();
                if (ref($val)) {
                        # If it is a nested object, recurse to dump it.
                        dump_object($val, $indent);
                } else {
                        # Otherwise it is just a 'plain' value, so
                        # display it.
                        printf("%s  Value = %s\n", $istr, $val);
                }

        #
        # Otherwise we know we are dealing with a group.  Groups
        # represent contents as a perl list or array (depending on
        # context), so we can process the contents of the group
        # with a 'foreach' loop, which provides a list context.
        # In a list context the value method returns the content
        # of the group as a perl list, which is the quickest
        # mechanism, but doesn't allow the group to be modified.
        # If we wanted to modify the contents of the group we could
        # do so like this:
        #    my $grp = $obj->value();   # Returns an array reference
        #    $grp->[0] = $newitem;
        # but accessing the group elements this way is much slower.
        #
        } else {
                printf("%sGROUP\n%s  Catalog = %s|%s|%s\n",
                    $istr, $istr, @cat);
                $indent++;
                # 'foreach' provides a list context.
                foreach my $val ($obj->value()) {
                        dump_object($val, $indent);
                }
                printf("%sENDGROUP\n", $istr);
        }
}
```

How to Create a New Group Record and Write It to a File

Use this script to create a new group record and write it to a file named `/tmp/exacct`.

```
#!/usr/bin/perl

use strict;
use warnings;
use Sun::Solaris::Exacct qw(:EXACCT_ALL);
# Prototype list of catalog tags and values.
     my @items = (
             [ &EXT_STRING | &EXC_DEFAULT | &EXD_CREATOR      => "me"       ],
```

```
        [ &EXT_UINT32 | &EXC_DEFAULT | &EXD_PROC_PID     => $$         ],
        [ &EXT_UINT32 | &EXC_DEFAULT | &EXD_PROC_UID     => $<         ],
        [ &EXT_UINT32 | &EXC_DEFAULT | &EXD_PROC_GID     => $(         ],
        [ &EXT_STRING | &EXC_DEFAULT | &EXD_PROC_COMMAND => "/bin/rec" ],
);

# Create a new group catalog object.
my $cat = ea_new_catalog(&EXT_GROUP | &EXC_DEFAULT | &EXD_NONE)

# Create a new Group object and retrieve its data array.
my $group = ea_new_group($cat);
my $ary = $group->value();

# Push the new Items onto the Group array.
foreach my $v (@items) {
        push(@$ary, ea_new_item(ea_new_catalog($v->[0]), $v->[1]));
}

# Open the exacct file, write the record & close.
my $f = ea_new_file('/tmp/exacct', &O_RDWR | &O_CREAT | &O_TRUNC)
   || die("create /tmp/exacct failed: ", ea_error_str(), "\n");
$f->write($group);
$f = undef;
```

How to Print the Contents of an exacct File

Use the following Perl script to print the contents of an exacct file.

```
#!/usr/bin/perl

     use strict;
     use warnings;
     use Sun::Solaris::Exacct qw(:EXACCT_ALL);

     die("Usage is dumpexacct <exacct file>\n") unless (@ARGV == 1);

     # Open the exacct file and display the header information.
     my $ef = ea_new_file($ARGV[0], &O_RDONLY) || die(error_str());
     printf("Creator:  %s\n", $ef->creator());
     printf("Hostname: %s\n\n", $ef->hostname());

     # Dump the file contents
     while (my $obj = $ef->get()) {
             ea_dump_object($obj);
     }

     # Report any errors
     if (ea_error() != EXR_OK && ea_error() != EXR_EOF)  {
             printf("\nERROR: %s\n", ea_error_str());
             exit(1);
     }
     exit(0);
```

Example Output From `Sun::Solaris::Exacct::Object->dump()`

Here is example output produced by running `Sun::Solaris::Exacct::Object->dump()` on the file created in "How to Create a New Group Record and Write It to a File" on page 56.

```
Creator:  root
Hostname: localhost
GROUP
       Catalog = EXT_GROUP|EXC_DEFAULT|EXD_NONE
       ITEM
         Catalog = EXT_STRING|EXC_DEFAULT|EXD_CREATOR
         Value = me
       ITEM
         Catalog = EXT_UINT32|EXC_DEFAULT|EXD_PROC_PID
         Value = 845523
       ITEM
         Catalog = EXT_UINT32|EXC_DEFAULT|EXD_PROC_UID
         Value = 37845
       ITEM
         Catalog = EXT_UINT32|EXC_DEFAULT|EXD_PROC_GID
         Value = 10
       ITEM
         Catalog = EXT_STRING|EXC_DEFAULT|EXD_PROC_COMMAND
         Value = /bin/rec
ENDGROUP
```

♦ ♦ ♦ CHAPTER 6

About Resource Controls

After you determine the resource consumption of workloads on your system as described in Chapter 4, "About Extended Accounting", you can place boundaries on resource usage. Boundaries prevent workloads from over-consuming resources. The *resource controls* facility is the constraint mechanism that is used for this purpose.

This chapter covers the following topics.

- "Resource Controls Concepts" on page 59
- "Configuring Resource Controls and Attributes" on page 61
- "Applying Resource Controls" on page 72
- "Temporarily Updating Resource Control Values on a Running System" on page 73
- "Commands Used With Resource Controls" on page 73

For information about how to administer resource controls, see Chapter 7, "Administering Resource Controls Tasks".

Resource Controls Concepts

In the Oracle Solaris operating system, the concept of a per-process resource limit has been extended to the task and project entities described in Chapter 2, "About Projects and Tasks". These enhancements are provided by the resource controls (rctls) facility. In addition, allocations that were set through the `/etc/system` tunables are now automatic or configured through the resource controls mechanism as well.

A resource control is identified by the prefix `zone`, `project`, `task`, or `process`. Resource controls can be observed on a system-wide basis. It is possible to update resource control values on a running system.

For a list of the standard resource controls that are available in this release, see "Available Resource Controls" on page 62. See "Available Resource Controls" on page 62 for information on available zone-wide resource controls.

Resource Limits and Resource Controls

UNIX systems have traditionally provided a resource limit facility (*rlimit*). The rlimit facility allows administrators to set one or more numerical limits on the amount of resources a process can consume. These limits include per-process CPU time used, per-process core file size, and per-process maximum heap size. *Heap size* is the amount of scratch memory that is allocated for the process data segment.

The resource controls facility provides compatibility interfaces for the resource limits facility. Existing applications that use resource limits continue to run unchanged. These applications can be observed in the same way as applications that are modified to take advantage of the resource controls facility.

Interprocess Communication and Resource Controls

Processes can communicate with each other by using one of several types of interprocess communication (IPC). IPC allows information transfer or synchronization to occur between processes. The resource controls facility provides resource controls that define the behavior of the kernel's IPC facilities. These resource controls replace the `/etc/system` tunables.

Obsolete parameters that are used to initialize the default resource control values might be included in the `/etc/system` file on this Oracle Solaris system. However, using the obsolete parameters is not recommended.

To observe which IPC objects are contributing to a project's usage, use the `ipcs` command with the `-J` option. See "How to Use `ipcs`" on page 84 to view an example display. For more information about the `ipcs` command, see `ipcs(1)`.

For information about Oracle Solaris system tuning, see the "Oracle Solaris 11.2 Tunable Parameters Reference Manual ".

Resource Control Constraint Mechanisms

Resource controls provide a mechanism for the constraint of system resources. Processes, tasks, projects, and zones can be prevented from consuming amounts of specified system resources. This mechanism leads to a more manageable system by preventing over-consumption of resources.

Constraint mechanisms can be used to support capacity-planning processes. An encountered constraint can provide information about application resource needs without necessarily denying the resource to the application.

Project Attribute Mechanisms

Resource controls can also serve as a simple attribute mechanism for resource management facilities. For example, the number of CPU shares made available to a project in the fair share scheduler (FSS) scheduling class is defined by the `project.cpu-shares` resource control. Because the project is assigned a fixed number of shares by the control, the various actions associated with exceeding a control are not relevant. In this context, the current value for the `project.cpu-shares` control is considered an attribute on the specified project.

Another type of project attribute is used to regulate the resource consumption of physical memory by collections of processes attached to a project. These attributes have the prefix `rcap`, for example, `rcap.max-rss`. Like a resource control, this type of attribute is configured in the `project` database. However, while resource controls are synchronously enforced by the kernel, resource caps are asynchronously enforced at the user level by the resource cap enforcement daemon, `rcapd`. For information on `rcapd`, see Chapter 10, "About Physical Memory Control Using the Resource Capping Daemon" and `rcapd`(1M).

The `project.pool` attribute is used to specify a pool binding for a project. For more information on resource pools, see Chapter 12, "About Resource Pools".

Configuring Resource Controls and Attributes

The resource controls facility is configured through the `project` database. See Chapter 2, "About Projects and Tasks". Resource controls and other attributes are set in the final field of the `project` database entry. The values associated with each resource control are enclosed in parentheses, and appear as plain text separated by commas. The values in parentheses constitute an "action clause." Each action clause is composed of a privilege level, a threshold value, and an action that is associated with the particular threshold. Each resource control can have multiple action clauses, which are also separated by commas. The following entry defines a per-task lightweight process limit and a per-process maximum CPU time limit on a project entity. The `process.max-cpu-time` would send a process a SIGTERM after the process ran for 1 hour, and a SIGKILL if the process continued to run for a total of 1 hour and 1 minute. See Table 6-3.

```
development:101:Developers:::task.max-lwps=(privileged,10,deny);
  process.max-cpu-time=(basic,3600,signal=TERM),(priv,3660,signal=KILL)
```
typed as one line

Note - On systems that have zones enabled, zone-wide resource controls are specified in the zone configuration using a slightly different format. See "Setting Zone-Wide Resource Controls" in "Introduction to Oracle Solaris Zones " for more information.

The `rctladm` command allows you to make runtime interrogations of and modifications to the resource controls facility, with *global scope*. The `prctl` command allows you to make runtime interrogations of and modifications to the resource controls facility, with *local scope*.

For more information, see "Global and Local Actions on Resource Control Values" on page 68, `rctladm`(1M) and `prctl`(1).

Note - On a system with zones installed, you cannot use `rctladm` in a non-global zone to modify settings. You can use `rctladm` in a non-global zone to view the global logging state of each resource control.

Available Resource Controls

A list of the standard resource controls that are available in this release is shown in the following table.

The table describes the resource that is constrained by each control. The table also identifies the default units that are used by the `project` database for that resource. The default units are of two types:

- Quantities represent a limited amount.
- Indexes represent a maximum valid identifier.

Thus, `project.cpu-shares` specifies the number of shares to which the project is entitled. `process.max-file-descriptor` specifies the highest file number that can be assigned to a process by the `open`(2) system call.

TABLE 6-1 Standard Project, Task, and Process Resource Controls

Control Name	Description	Default Unit
`project.cpu-cap`	Absolute limit on the amount of CPU resources that can be consumed by a project. A value of `100` means 100% of one CPU as the `project.cpu-cap` setting. A value of `125` is 125%, because 100% corresponds to one full CPU on the system when using CPU caps.	Quantity (number of CPUs)
`project.cpu-shares`	Number of CPU shares granted to this project for use with the fair share scheduler (see `FSS`(7)).	Quantity (shares)
`project.max-crypto-memory`	Total amount of kernel memory that can be used by `libpkcs11` for hardware crypto acceleration. Allocations for kernel buffers	Size (bytes)

Control Name	Description	Default Unit
	and session-related structures are charged against this resource control.	
project.max-locked-memory	Total amount of physical locked memory allowed. If priv_proc_lock_memory is assigned to a user, consider setting this resource control as well to prevent that user from locking all memory. Note that this resource control replaced project.max-device-locked-memory, which has been removed.	Size (bytes)
project.max-msg-ids	Maximum number of message queue IDs allowed for this project.	Quantity (message queue IDs)
project.max-port-ids	Maximum allowable number of event ports.	Quantity (number of event ports)
project.max-processes	Maximum number of process table slots simultaneously available to this project. Note that because both normal processes and zombie processes take up process table slots, the max-processes control thus protects against zombies exhausting the process table. Because zombie processes do not have any LWPs by definition, the max-lwps control cannot protect against this possibility.	Quantity (process table slots)
project.max-sem-ids	Maximum number of semaphore IDs allowed for this project.	Quantity (semaphore IDs)
project.max-shm-ids	Maximum number of shared memory IDs allowed for this project.	Quantity (shared memory IDs)
project.max-shm-memory	Total amount of System V shared memory allowed for this project.	Size (bytes)
project.max-lwps	Maximum number of LWPs simultaneously available to this project.	Quantity (LWPs)
project.max-tasks	Maximum number of tasks allowable in this project.	Quantity (number of tasks)
project.max-contracts	Maximum number of contracts allowed in this project.	Quantity (contracts)
task.max-cpu-time	Maximum CPU time that is available to this task's processes.	Time (seconds)
task.max-lwps	Maximum number of LWPs simultaneously available to this task's processes.	Quantity (LWPs)

Control Name	Description	Default Unit
task.max-processes	Maximum number of process table slots simultaneously available to this task's processes.	Quantity (process table slots)
`process.max-cpu-time`	Maximum CPU time that is available to this process.	Time (seconds)
`process.max-file-descriptor`	Maximum file descriptor index available to this process.	Index (maximum file descriptor)
`process.max-file-size`	Maximum file offset available for writing by this process.	Size (bytes)
`process.max-core-size`	Maximum size of a core file created by this process.	Size (bytes)
`process.max-data-size`	Maximum heap memory available to this process.	Size (bytes)
`process.max-stack-size`	Maximum stack memory segment available to this process.	Size (bytes)
`process.max-address-space`	Maximum amount of address space, as summed over segment sizes, that is available to this process.	Size (bytes)
`process.max-port-events`	Maximum allowable number of events per event port.	Quantity (number of events)
`process.max-sem-nsems`	Maximum number of semaphores allowed per semaphore set.	Quantity (semaphores per set)
`process.max-sem-ops`	Maximum number of semaphore operations allowed per `semop` call (value copied from the resource control at `semget()` time).	Quantity (number of operations)
`process.max-msg-qbytes`	Maximum number of bytes of messages on a message queue (value copied from the resource control at `msgget()` time).	Size (bytes)
`process.max-msg-messages`	Maximum number of messages on a message queue (value copied from the resource control at `msgget()` time).	Quantity (number of messages)

You can display the default values for resource controls on a system that does not have any resource controls set or changed. Such a system contains no non-default entries in `/etc/system` or the `project` database. To display values, use the `prctl` command.

Zone-Wide Resource Controls

Zone-wide resource controls limit the total resource usage of all process entities within a zone. Zone-wide resource controls can also be set using global property names as described in "Zone Components" in "Introduction to Oracle Solaris Zones ".

TABLE 6-2 Zones Resource Controls

Control Name	Description	Default Unit
`zone.cpu-cap`	Absolute limit on the amount of CPU resources that can be consumed by a non-global zone. A value of `100` means 100% of one CPU as the `project.cpu-cap` setting. A value of `125` is 125%, because 100% corresponds to one full CPU on the system when using CPU caps.	Quantity (number of CPUs)
`zone.cpu-shares`	Number of fair share scheduler (FSS) CPU shares for this zone	Quantity (shares)
`zone.max-lofi`	Maximum number of `lofi` devices that can be created by a zone. The value limits each zone's usage of the minor node namespace.	Quantity (number of `lofi` devices)
`zone.max-locked-memory`	Total amount of physical locked memory available to a zone. When `priv_proc_lock_memory` is assigned to a zone, consider setting this resource control as well to prevent that zone from locking all memory.	Size (bytes)
`zone.max-lwps`	Maximum number of LWPs simultaneously available to this zone	Quantity (LWPs)
`zone.max-msg-ids`	Maximum number of message queue IDs allowed for this zone	Quantity (message queue IDs)
zone.max-processes	Maximum number of process table slots simultaneously available to this zone. Because both normal processes and zombie processes take up process table slots, the `max-processes` control thus protects against zombies exhausting the process table. Because zombie processes do not have any LWPs by definition, the `max-lwps` control cannot protect against this possibility.	Quantity (process table slots)

Control Name	Description	Default Unit
`zone.max-sem-ids`	Maximum number of semaphore IDs allowed for this zone	Quantity (semaphore IDs)
`zone.max-shm-ids`	Maximum number of shared memory IDs allowed for this zone	Quantity (shared memory IDs)
`zone.max-shm-memory`	Total amount of System V shared memory allowed for this zone	Size (bytes)
`zone.max-swap`	Total amount of swap that can be consumed by user process address space mappings and `tmpfs` mounts for this zone.	Size (bytes)

For information on configuring zone-wide resource controls, see "Configuring Resource Controls and Attributes" on page 61 and "Setting Zone-Wide Resource Controls" in "Introduction to Oracle Solaris Zones ".

Note that it is possible to apply a zone-wide resource control to the global zone.

Units Support

Global flags that identify resource control types are defined for all resource controls. The flags are used by the system to communicate basic type information to applications such as the `prctl` command. Applications use the information to determine the following:

- The unit strings that are appropriate for each resource control
- The correct scale to use when interpreting scaled values

The following global flags are available:

Global Flag	Resource Control Type String	Modifier	Scale
RCTL_GLOBAL_BYTES	bytes	B	1
		KB	2^{10}
		MB	2^{20}
		GB	2^{30}
		TB	2^{40}
		PB	2^{50}
		EB	2^{60}
RCTL_GLOBAL_SECONDS	seconds	s	1
		Ks	10^3

Global Flag	Resource Control Type String	Modifier	Scale
		Ms	10^6
		Gs	10^9
		Ts	10^{12}
		Ps	10^{15}
		Es	10^{18}
RCTL_GLOBAL_COUNT	count	none	1
		K	10^3
		M	10^6
		G	10^9
		T	10^{12}
		P	10^{15}
		E	10^{18}

Scaled values can be used with resource controls. The following example shows a scaled threshold value:

```
task.max-lwps=(priv,1K,deny)
```

Note - Unit modifiers are accepted by the `prctl`, `projadd`, and `projmod` commands. You cannot use unit modifiers in the `project` database itself.

Resource Control Values and Privilege Levels

A threshold value on a resource control constitutes an enforcement point where local actions can be triggered or global actions, such as logging, can occur.

Each threshold value on a resource control must be associated with a privilege level. The privilege level must be one of the following three types.

- Basic, which can be modified by the owner of the calling process
- Privileged, which can be modified only by privileged (root) callers
- System, which is fixed for the duration of the operating system instance

A resource control is guaranteed to have one system value, which is defined by the system, or resource provider. The system value represents how much of the resource the current implementation of the operating system is capable of providing.

Any number of privileged values can be defined, and only one basic value is allowed. Operations that are performed without specifying a privilege value are assigned a basic privilege by default.

The privilege level for a resource control value is defined in the privilege field of the resource control block as RCTL_BASIC, RCTL_PRIVILEGED, or RCTL_SYSTEM. See `setrctl`(2) for more information. You can use the `prctl` command to modify values that are associated with basic and privileged levels.

Global and Local Actions on Resource Control Values

There are two categories of actions on resource control values: global and local.

Global Actions on Resource Control Values

Global actions apply to resource control values for every resource control on the system. You can use the `rctladm` command described in the `rctladm`(1M) man page to perform the following actions:

- Display the global state of active system resource controls
- Set global logging actions

You can disable or enable the global logging action on resource controls. You can set the `syslog` action to a specific degree by assigning a severity level, `syslog=`*level*. The possible settings for *level* are as follows:

- `debug`
- `info`
- `notice`
- `warning`
- `err`
- `crit`
- `alert`
- `emerg`

By default, there is no global logging of resource control violations. The level `n/a` indicates resource controls on which no global action can be configured.

Local Actions on Resource Control Values

Local actions are taken on a process that attempts to exceed the control value. For each threshold value that is placed on a resource control, you can associate one or more actions. There are three types of local actions: `none`, `deny`, and `signal=`. These three actions are used as follows:

`none` — No action is taken on resource requests for an amount that is greater than the threshold. This action is useful for monitoring resource usage without affecting the progress of applications. You can also enable a global message that displays when the resource control is exceeded, although the process exceeding the threshhold is not affected.

`deny` — You can deny resource requests for an amount that is greater than the threshold. For example, a `task.max-lwps` resource control with action `deny` causes a `fork` system call to fail if the new process would exceed the control value. See the `fork`(2) man page.

`signal=` — You can enable a global signal message action when the resource control is exceeded. A signal is sent to the process when the threshold value is exceeded. Additional signals are not sent if the process consumes additional resources. Available signals are listed in Table 6-3.

Not all of the actions can be applied to every resource control. For example, a process cannot exceed the number of CPU shares assigned to the project of which it is a member. Therefore, a deny action is not allowed on the `project.cpu-shares` resource control.

Due to implementation restrictions, the global properties of each control can restrict the range of available actions that can be set on the threshold value. (See the `rctladm`(1M) man page.) A list of available signal actions is presented in the following table. For additional information about signals, see the `signal`(3HEAD) man page.

TABLE 6-3 Signals Available to Resource Control Values

Signal	Description	Notes
SIGABRT	Terminate the process.	
SIGHUP	Send a hangup signal. Occurs when carrier drops on an open line. Signal sent to the process group that controls the terminal.	
SIGTERM	Terminate the process. Termination signal sent by software.	
SIGKILL	Terminate the process and kill the program.	
SIGSTOP	Stop the process. Job control signal.	

Signal	Description	Notes
SIGXRES	Resource control limit exceeded. Generated by resource control facility.	
SIGXFSZ	Terminate the process. File size limit exceeded.	Available only to resource controls with the RCTL_GLOBAL_FILE_SIZE property (`process.max-file-size`). See `rctlblk_set_value`(3C) for more information.
SIGXCPU	Terminate the process. CPU time limit exceeded.	Available only to resource controls with the RCTL_GLOBAL_CPUTIME property (`process.max-cpu-time`). See `rctlblk_set_value`(3C) for more information.

Resource Control Flags and Properties

Each resource control on the system has a certain set of associated properties. This set of properties is defined as a set of flags, which are associated with all controlled instances of that resource. Global flags cannot be modified, but the flags can be retrieved by using either `rctladm` or the `getrctl` system call.

Local flags define the default behavior and configuration for a specific threshold value of that resource control on a specific process or process collective. The local flags for one threshold value do not affect the behavior of other defined threshold values for the same resource control. However, the global flags affect the behavior for every value associated with a particular control. Local flags can be modified, within the constraints supplied by their corresponding global flags, by the `prctl` command or the `setrctl` system call. See `setrctl`(2).

For the complete list of local flags, global flags, and their definitions, see `rctlblk_set_value`(3C).

To determine system behavior when a threshold value for a particular resource control is reached, use `rctladm` to display the global flags for the resource control . For example, to display the values for `process.max-cpu-time`, type the following:

```
$ rctladm process.max-cpu-time
 process.max-cpu-time  syslog=off  [ lowerable no-deny cpu-time inf seconds ]
```

The global flags indicate the following.

`lowerable` — Superuser privileges are not required to lower the privileged values for this control.

`no-deny` — Even when threshold values are exceeded, access to the resource is never denied.

`cpu-time`	SIGXCPU is available to be sent when threshold values of this resource are reached.
`seconds`	The time value for the resource control.
`no-basic`	Resource control values with the privilege type `basic` cannot be set. Only privileged resource control values are allowed.
`no-signal`	A local signal action cannot be set on resource control values.
`no-syslog`	The global `syslog` message action may not be set for this resource control.
`deny`	Always deny request for resource when threshold values are exceeded.
`count`	A count (integer) value for the resource control.
`bytes`	Unit of size for the resource control.

Use the `prctl` command to display local values and actions for the resource control.

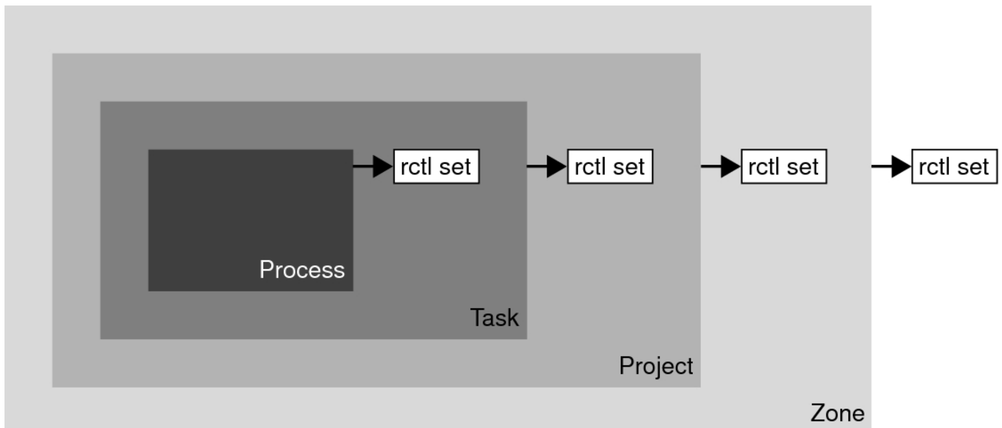
```
$ prctl -n process.max-cpu-time $$
 process 353939: -ksh
 NAME    PRIVILEGE    VALUE    FLAG   ACTION              RECIPIENT
 process.max-cpu-time
         privileged   18.4Es    inf   signal=XCPU                 -
         system       18.4Es    inf   none
```

The `max` (RCTL_LOCAL_MAXIMAL) flag is set for both threshold values, and the `inf` (RCTL_GLOBAL_INFINITE) flag is defined for this resource control. An `inf` value has an infinite quantity. The value is never enforced. Hence, as configured, both threshold quantities represent infinite values that are never exceeded.

Resource Control Enforcement

More than one resource control can exist on a resource. A resource control can exist at each containment level in the process model. If resource controls are active on the same resource at different container levels, the smallest container's control is enforced first. Thus, action is taken on `process.max-cpu-time` before `task.max-cpu-time` if both controls are encountered simultaneously.

FIGURE 6-1 Process Collectives, Container Relationships, and Their Resource Control Sets



Global Monitoring of Resource Control Events

Often, the resource consumption of processes is unknown. To get more information, try using the global resource control actions that are available with the `rctladm` command. Use `rctladm` to establish a `syslog` action on a resource control. Then, if any entity managed by that resource control encounters a threshold value, a system message is logged at the configured logging level. See Chapter 7, "Administering Resource Controls Tasks" and the `rctladm`(1M) man page for more information.

Applying Resource Controls

Each resource control listed in Table 6-1 can be assigned to a project at login or when `newtask`, `su`, or the other project-aware launchers `at`, `batch`, or `cron` are invoked. Each command that is initiated is launched in a separate task with the invoking user's default project. See the man pages `login`(1), `newtask`(1), `at`(1), `cron`(1M), and `su`(1M) for more information.

Updates to entries in the `project` database, whether to the `/etc/project` file or to a representation of the database in a network name service, are not applied to currently active projects. The updates are applied when a new task joins the project through login or `newtask`.

Temporarily Updating Resource Control Values on a Running System

Values changed in the `project` database only become effective for new tasks that are started in a project. However, you can use the `rctladm` and `prctl` commands to update resource controls on a running system.

Updating Logging Status

The `rctladm` command affects the global logging state of each resource control on a system-wide basis. This command can be used to view the global state and to set up the level of `syslog` logging when controls are exceeded.

Updating Resource Controls

You can view and temporarily alter resource control values and actions on a per-process, per-task, or per-project basis by using the `prctl` command. A project, task, or process ID is given as input, and the command operates on the resource control at the level where the control is defined.

Any modifications to values and actions take effect immediately. However, these modifications apply to the current process, task, or project only. The changes are not recorded in the `project` database. If the system is restarted, the modifications are lost. Permanent changes to resource controls must be made in the `project` database.

All resource control settings that can be modified in the `project` database can also be modified with the `prctl` command. Both basic and privileged values can be added or be deleted. Their actions can also be modified. By default, the basic type is assumed for all set operations, but processes and users with root privileges can also modify privileged resource controls. System resource controls cannot be altered.

Commands Used With Resource Controls

The commands that are used with resource controls are shown in the following table.

Command Reference	Description
`ipcs`(1)	Allows you to observe which IPC objects are contributing to a project's usage

Command Reference	Description
prctl(1)	Allows you to make runtime interrogations of and modifications to the resource controls facility, with local scope
rctladm(1M)	Allows you to make runtime interrogations of and modifications to the resource controls facility, with global scope

The resource_controls(5) man page describes resource controls available through the project database, including units and scaling factors.

♦ ♦ ♦ CHAPTER 7

Administering Resource Controls Tasks

This chapter describes how to administer the resource controls facility.

For an overview of the resource controls facility, see Chapter 6, "About Resource Controls".

Administering Resource Controls Task Map

Task	Description	For Instructions
Set resource controls.	Set resource controls for a project in the /etc/project file.	"Setting Resource Controls" on page 76
Get or revise the resource control values for active processes, tasks, or projects, with local scope.	Make runtime interrogations of and modifications to the resource controls associated with an active process, task, or project on the system.	"Using the prctl Command" on page 78
On a running system, view or update the global state of resource controls.	View the global logging state of each resource control on a system-wide basis. Also set up the level of syslog logging when controls are exceeded.	"Using rctladm" on page 83
Report status of active interprocess communication (IPC) facilities.	Display information about active interprocess communication (IPC) facilities. Observe which IPC objects are contributing to a project's usage.	"Using ipcs" on page 84
Determine whether a web server is allocated sufficient CPU capacity.	Set a global action on a resource control. This action enables you to receive notice of any entity that has a resource control value that is set too low.	"How to Determine Whether a Web Server Is Allocated Enough CPU Capacity" on page 85

Setting Resource Controls

How to Set the Maximum Number of LWPs for Each Task in a Project

This procedure adds a project named `x-files` to the `/etc/project` file and sets a maximum number of LWPs for a task created in the project.

1. **Become root or assume an equivalent role.**
2. **Use the `projadd` command with the `-K` option to create a project called `x-files`. Set the maximum number of LWPs for each task created in the project to `3`.**

   ```
   # projadd -K 'task.max-lwps=(privileged,3,deny)' x-files
   ```

3. **View the entry in the `/etc/project` file by using one of the following methods:**
   - Type:

     ```
     # projects -l
     system
             projid : 0
             comment: ""
             users  : (none)
             groups : (none)
             attribs:
     .
     .
     .
     x-files
             projid : 100
             comment: ""
             users  : (none)
             groups : (none)
             attribs: task.max-lwps=(privileged,3,deny)
     ```

   - Type:

     ```
     # cat /etc/project
     system:0:System:::
     .
     .
     .
     x-files:100::::task.max-lwps=(privileged,3,deny)
     ```

Example 7-1 Sample Session

After implementing the steps in this procedure, when the root user creates a new task in project `x-files` by joining the project with `newtask`, the user will not be able to create more than three LWPs while running in this task. This is shown in the following annotated sample session.

```
# newtask -p x-files csh

# prctl -n task.max-lwps $$
process: 111107: csh
NAME    PRIVILEGE    VALUE    FLAG   ACTION            RECIPIENT
task.max-lwps
        usage            3
        privileged       3       -   deny                       -
        system       2.15G     max   deny                       -
# id -p
uid=0(root) gid=1(other) projid=100(x-files)

# ps -o project,taskid -p $$
 PROJECT TASKID
 x-files    73

# csh       /* creates second LWP */

# csh       /* creates third LWP */

# csh       /* cannot create more LWPs */
Vfork failed
#
```

How to Set Multiple Controls on a Project

The `/etc/project` file can contain settings for multiple resource controls for each project as well as multiple threshold values for each control. Threshold values are defined in action clauses, which are comma-separated for multiple values.

1. **Become root or assume an equivalent role.**
2. **Use the `projmod` command with the `-s` and `-K` options to set resource controls on project `x-files`:**

```
# projmod -s -K 'task.max-lwps=(basic,10,none),(privileged,500,deny);
process.max-file-descriptor=(basic,128,deny)' x-files      one line in file
```

The following controls are set:

- A `basic` control with no action on the maximum LWPs per task.

- A privileged deny control on the maximum LWPs per task. This control causes any LWP creation that exceeds the maximum to fail, as shown in the previous example "How to Set the Maximum Number of LWPs for Each Task in a Project" on page 76.
- A limit on the maximum file descriptors per process at the basic level, which forces the failure of any open call that exceeds the maximum.

3. **View the entry in the file by using one of the following methods:**

- Type:

```
# projects -l
.
.
.
x-files
        projid : 100
        comment: ""
        users  : (none)
        groups : (none)
        attribs: process.max-file-descriptor=(basic,128,deny)
                 task.max-lwps=(basic,10,none),(privileged,500,deny)      one line in file
```

- Type:

```
# cat /etc/project
.
.
.
x-files:100::::process.max-file-descriptor=(basic,128,deny);
task.max-lwps=(basic,10,none),(privileged,500,deny)      one line in file
```

Using the prctl Command

Use the prctl command to make runtime interrogations of and modifications to the resource controls associated with an active process, task, or project on the system. See the prctl(1) man page for more information.

How to Use the `prctl` Command to Display Default Resource Control Values

This procedure must be used on a system on which no resource controls have been set or changed. There can be only non-default entries in the `/etc/system` file or in the `project` database.

- **Use the `prctl` command on any process, such as the current shell that is running.**

```
# prctl $$
process: 3320: bash
NAME    PRIVILEGE       VALUE    FLAG   ACTION                       RECIPIENT
process.max-port-events
        privileged      65.5K       -   deny                                 -
        system          2.15G     max   deny                                 -
process.max-msg-messages
        privileged      8.19K       -   deny                                 -
        system          4.29G     max   deny                                 -
process.max-msg-qbytes
        privileged      64.0KB      -   deny                                 -
        system          16.0EB    max   deny                                 -
process.max-sem-ops
        privileged        512       -   deny                                 -
        system          2.15G     max   deny                                 -
process.max-sem-nsems
        privileged        512       -   deny                                 -
        system          32.8K     max   deny                                 -
process.max-address-space
        privileged      16.0EB    max   deny                                 -
        system          16.0EB    max   deny                                 -
process.max-file-descriptor
        basic             256       -   deny                              3320
        privileged      65.5K       -   deny                                 -
        system          2.15G     max   deny                                 -
process.max-core-size
        privileged      8.00EB    max   deny                                 -
        system          8.00EB    max   deny                                 -
process.max-stack-size
        basic           10.0MB      -   deny                              3320
        privileged      32.0TB      -   deny                                 -
        system          32.0TB    max   deny                                 -
process.max-data-size
        privileged      16.0EB    max   deny                                 -
        system          16.0EB    max   deny                                 -
process.max-file-size
        privileged      8.00EB    max   deny,signal=XFSZ                     -
        system          8.00EB    max   deny                                 -
process.max-cpu-time
        privileged      18.4Es    inf   signal=XCPU                          -
        system          18.4Es    inf   none                                 -
task.max-cpu-time
        usage              0s
        system          18.4Es    inf   none                                 -
```

```
task.max-processes
        usage               2
        system          2.15G     max   deny                                  -
task.max-lwps
        usage               3
        system          2.15G     max   deny                                  -
project.max-contracts
        privileged      10.0K       -   deny                                  -
        system          2.15G     max   deny                                  -
project.max-locked-memory
        usage               0B
        system          16.0EB    max   deny                                  -
project.max-port-ids
        privileged      8.19K       -   deny                                  -
        system          65.5K     max   deny                                  -
project.max-shm-memory
        privileged       510MB      -   deny                                  -
        system          16.0EB    max   deny                                  -
project.max-shm-ids
        privileged        128       -   deny                                  -
        system          16.8M     max   deny                                  -
project.max-msg-ids
        privileged        128       -   deny                                  -
        system          16.8M     max   deny                                  -
project.max-sem-ids
        privileged        128       -   deny                                  -
        system          16.8M     max   deny                                  -
project.max-crypto-memory
        usage               0B
        privileged       510MB      -   deny                                  -
        system          16.0EB    max   deny                                  -
project.max-tasks
        usage               2
        system          2.15G     max   deny                                  -
project.max-processes
        usage               4
        system          2.15G     max   deny                                  -
project.max-lwps
        usage              11
        system          2.15G     max   deny                                  -
project.cpu-cap
        usage               0
        system          4.29G     inf   deny                                  -
project.cpu-shares
        usage               1
        privileged          1       -   none                                  -
        system          65.5K     max   none                                  -
zone.max-lofi
        usage               0
        system          18.4E     max   deny                                  -
zone.max-swap
        usage            180MB
        system          16.0EB    max   deny                                  -
zone.max-locked-memory
        usage               0B
        system          16.0EB    max   deny                                  -
zone.max-shm-memory
        system          16.0EB    max   deny                                  -
```

```
zone.max-shm-ids
        system          16.8M     max   deny                                 -
zone.max-sem-ids
        system          16.8M     max   deny                                 -
zone.max-msg-ids
        system          16.8M     max   deny                                 -
zone.max-processes
        usage             73
        system          2.15G     max   deny                                 -
zone.max-lwps
        usage            384
        system          2.15G     max   deny                                 -
zone.cpu-cap
        usage              0
        system          4.29G     inf   deny                                 -
zone.cpu-shares
        usage              1
        privileged         1        -   none                                 -
        system          65.5K     max   none
```

How to Use the `prctl` Command to Display Information for a Given Resource Control

- **Display the maximum file descriptor for the current shell that is running.**

```
# prctl -n process.max-file-descriptor $$
process: 110453: -sh
NAME    PRIVILEGE       VALUE    FLAG   ACTION       RECIPIENT
process.max-file-descriptor
        basic             256       -   deny            11731
        privileged      65.5K       -   deny                -
        system          2.15G     max   deny
```

How to Use `prctl` to Temporarily Change a Value

This example procedure uses the `prctl` command to temporarily add a new privileged value to deny the use of more than three LWPs per project for the `x-files` project. The result is comparable to the result in "How to Set the Maximum Number of LWPs for Each Task in a Project" on page 76.

1. **Become root or assume an equivalent role.**
2. **Use `newtask` to join the `x-files` project.**

```
# newtask -p x-files
```

3. **Use the `id` command with the `-p` option to verify that the correct project has been joined.**

```
# id -p
uid=0(root) gid=1(other) projid=101(x-files)
```

4. **Add a new privileged value for `project.max-lwps` that limits the number of LWPs to three.**

```
# prctl -n project.max-lwps -t privileged -v 3 -e deny -i project x-files
```

5. **Verify the result.**

```
# prctl -n project.max-lwps -i project x-files
process: 111108: csh
NAME    PRIVILEGE    VALUE    FLAG   ACTION            RECIPIENT
project.max-lwps
        usage          203
        privileged    1000       -   deny                      -
        system       2.15G     max   deny                      -
```

How to Use `prctl` to Lower a Resource Control Value

1. **Become root or assume an equivalent role.**

2. **Use the `prctl` command with the `-r` option to change the lowest value of the `process.max-file-descriptor` resource control.**

```
# prctl -n process.max-file-descriptor -r -v 128 $$
```

How to Use `prctl` to Display, Replace, and Verify the Value of a Control on a Project

1. **Become root or assume an equivalent role.**

2. **Display the value of `project.cpu-shares` in the project *group.staff*.**

```
# prctl -n project.cpu-shares -i project group.staff
project: 2: group.staff
NAME    PRIVILEGE       VALUE    FLAG   ACTION     RECIPIENT
project.cpu-shares
        usage               1
        privileged          1       -   none               -
        system          65.5K     max   none
```

3. **Replace the current `project.cpu-shares` value `1` with the value `10`.**

```
# prctl -n project.cpu-shares -v 10 -r -i project group.staff
```

4. **Display the value of `project.cpu-shares` in the project `group.staff`.**

```
# prctl -n project.cpu-shares -i project group.staff
project: 2: group.staff
NAME    PRIVILEGE       VALUE    FLAG   ACTION     RECIPIENT
project.cpu-shares
        usage               1
        privileged          1       -   none                             -
        system          65.5K     max   none
```

Using rctladm

How to Use rctladm

Use the `rctladm` command to make runtime interrogations of and modifications to the global state of the resource controls facility. See the rctladm(1M) man page for more information.

For example, you can use `rctladm` with the `-e` option to enable the global `syslog` attribute of a resource control. When the control is exceeded, notification is logged at the specified `syslog` level. To enable the global `syslog` attribute of `process.max-file-descriptor`, type the following:

```
# rctladm -e syslog process.max-file-descriptor
```

When used without arguments, the `rctladm` command displays the global flags, including the global type flag, for each resource control.

```
# rctladm
process.max-port-events     syslog=off  [ deny count ]
process.max-msg-messages    syslog=off  [ deny count ]
process.max-msg-qbytes      syslog=off  [ deny bytes ]
process.max-sem-ops         syslog=off  [ deny count ]
process.max-sem-nsems       syslog=off  [ deny count ]
process.max-address-space   syslog=off  [ lowerable deny no-signal bytes ]
process.max-file-descriptor syslog=off  [ lowerable deny count ]
process.max-core-size       syslog=off  [ lowerable deny no-signal bytes ]
process.max-stack-size      syslog=off  [ lowerable deny no-signal bytes ]
.
.
.
```

Using `ipcs`

How to Use `ipcs`

Use the `ipcs` utility to display information about active interprocess communication (IPC) facilities. See the `ipcs`(1) man page for more information.

You can use `ipcs` with the `-J` option to see which project's limit an IPC object is allocated against.

```
# ipcs -J
    IPC status from <running system> as of Wed Mar 26 18:53:15 PDT 2003
T         ID      KEY        MODE       OWNER    GROUP    PROJECT
Message Queues:
Shared Memory:
m       3600      0       --rw-rw-rw-   uname    staff    x-files
m        201      0       --rw-rw-rw-   uname    staff    x-files
m       1802      0       --rw-rw-rw-   uname    staff    x-files
m        503      0       --rw-rw-rw-   uname    staff    x-files
m        304      0       --rw-rw-rw-   uname    staff    x-files
m        605      0       --rw-rw-rw-   uname    staff    x-files
m          6      0       --rw-rw-rw-   uname    staff    x-files
m        107      0       --rw-rw-rw-   uname    staff    x-files
Semaphores:
s          0      0       --rw-rw-rw-   uname    staff    x-files
```

Capacity Warnings

A global action on a resource control enables you to receive notice of any entity that is tripping over a resource control value that is set too low.

For example, assume you want to determine whether a web server possesses sufficient CPUs for its typical workload. You could analyze `sar` data for idle CPU time and load average. You could also examine extended accounting data to determine the number of simultaneous processes that are running for the web server process.

However, an easier approach is to place the web server in a task. You can then set a global action, using `syslog`, to notify you whenever a task exceeds a scheduled number of LWPs appropriate for the machine's capabilities.

See the `sar`(1) man page for more information.

How to Determine Whether a Web Server Is Allocated Enough CPU Capacity

1. **Use the `prctl` command to place a privileged (root-owned) resource control on the tasks that contain an `httpd` process. Limit each task's total number of LWPs to 40, and disable all local actions.**

   ```
   # prctl -n task.max-lwps -v 40 -t privileged -d all `pgrep httpd`
   ```

2. **Enable a system log global action on the `task.max-lwps` resource control.**

   ```
   # rctladm -e syslog task.max-lwps
   ```

3. **Observe whether the workload trips the resource control.**

   If it does, you will see `/var/adm/messages` such as:

   ```
   Jan  8 10:15:15 testmachine unix: [ID 859581 kern.notice]
   NOTICE: privileged rctl task.max-lwps exceeded by task 19
   ```

CHAPTER 8

About Fair Share Scheduler

The analysis of workload data can indicate that a particular workload or group of workloads is monopolizing CPU resources. If these workloads are not violating resource constraints on CPU usage, you can modify the allocation policy for CPU time on the system. The fair share scheduling class described in this chapter enables you to allocate CPU time based on shares instead of the priority scheme of the timesharing (TS) scheduling class.

This chapter covers the following topics.

- "Introduction to the Scheduler" on page 87
- "CPU Share Definition" on page 88
- "CPU Shares and Process State" on page 88
- "CPU Share Versus Utilization" on page 89
- "CPU Share Examples" on page 89
- "FSS Configuration" on page 91
- "FSS and Processor Sets" on page 93
- "Combining FSS With Other Scheduling Classes" on page 95
- "Setting the Scheduling Class for the System" on page 96
- "Scheduling Class on a System with Zones Installed" on page 96
- "Commands Used With FSS" on page 96

To begin using the fair share scheduler, see Chapter 9, "Administering the Fair Share Scheduler Tasks".

Introduction to the Scheduler

A fundamental job of the operating system is to arbitrate which processes get access to the system's resources. The process scheduler, which is also called the dispatcher, is the portion of the kernel that controls allocation of the CPU to processes. The scheduler supports the concept of scheduling classes. Each class defines a scheduling policy that is used to schedule processes within the class. The default scheduler in the Oracle Solaris operating system, the TS scheduler, tries to give every process relatively equal access to the available CPUs. However, you might want to specify that certain processes be given more resources than others.

You can use the *fair share scheduler* (FSS) to control the allocation of available CPU resources among workloads, based on their importance. This importance is expressed by the number of *shares* of CPU resources that you assign to each workload.

You give each project CPU shares to control the project's entitlement to CPU resources. The FSS guarantees a fair dispersion of CPU resources among projects that is based on allocated shares, independent of the number of processes that are attached to a project. The FSS achieves fairness by reducing a project's entitlement for heavy CPU usage and increasing its entitlement for light usage, in accordance with other projects.

The FSS consists of a kernel scheduling class module and class-specific versions of the `dispadmin`(1M) and `priocntl`(1) commands. Project shares used by the FSS are specified through the `project.cpu-shares` property in the `project`(4) database.

Note - If you are using the `project.cpu-shares` resource control on an Oracle Solaris system with zones installed, see "Setting Zone-Wide Resource Controls" in "Introduction to Oracle Solaris Zones " and "Using the Fair Share Scheduler on an Oracle Solaris System With Zones Installed" in "Creating and Using Oracle Solaris Zones ".

CPU Share Definition

The term "share" is used to define a portion of the system's CPU resources that is allocated to a project. If you assign a greater number of CPU shares to a project, relative to other projects, the project receives more CPU resources from the fair share scheduler.

CPU shares are not equivalent to percentages of CPU resources. Shares are used to define the relative importance of workloads in relation to other workloads. When you assign CPU shares to a project, your primary concern is not the number of shares the project has. Knowing how many shares the project has in comparison with other projects is more important. You must also take into account how many of those other projects will be competing with it for CPU resources.

Note - Processes in projects with zero shares always run at the lowest system priority (0). These processes only run when projects with nonzero shares are not using CPU resources.

CPU Shares and Process State

In the Oracle Solaris system, a project workload usually consists of more than one process. From the fair share scheduler perspective, each project workload can be in either an *idle* state or an *active* state. A project is considered idle if none of its processes are using any CPU resources. This usually means that such processes are either *sleeping* (waiting for I/O completion) or stopped. A project is considered active if at least one of its processes is using CPU resources. The sum of shares of all active projects is used in calculating the portion of CPU resources to be assigned to projects.

When more projects become active, each project's CPU allocation is reduced, but the proportion between the allocations of different projects does not change.

CPU Share Versus Utilization

Share allocation is not the same as utilization. A project that is allocated 50 percent of the CPU resources might average only a 20 percent CPU use. Moreover, shares serve to limit CPU usage only when there is competition from other projects. Regardless of how low a project's allocation is, it always receives 100 percent of the processing power if it is running alone on the system. Available CPU cycles are never wasted. They are distributed between projects.

The allocation of a small share to a busy workload might slow its performance. However, the workload is not prevented from completing its work if the system is not overloaded.

CPU Share Examples

Assume you have a system with two CPUs running two parallel CPU-bound workloads called *A* and *B*, respectively. Each workload is running as a separate project. The projects have been configured so that project *A* is assigned S_A shares, and project *B* is assigned S_B shares.

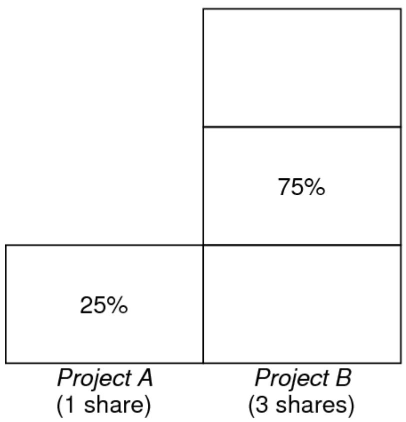
On average, under the traditional TS scheduler, each of the workloads that is running on the system would be given the same amount of CPU resources. Each workload would get 50 percent of the system's capacity.

When run under the control of the FSS scheduler with S_A=S_B, these projects are also given approximately the same amounts of CPU resources. However, if the projects are given different numbers of shares, their CPU resource allocations are different.

The next three examples illustrate how shares work in different configurations. These examples show that shares are only mathematically accurate for representing the usage if demand meets or exceeds available resources.

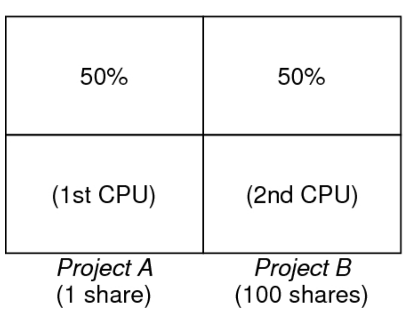
Example 1: Two CPU-Bound Processes in Each Project

If *A* and *B* each have two CPU-bound processes, and $S_A = 1$ and $S_B = 3$, then the total number of shares is 1 + 3 = 4. In this configuration, given sufficient CPU demand, projects *A* and *B* are allocated 25 percent and 75 percent of CPU resources, respectively.



Example 2: No Competition Between Projects

If *A* and *B* have only *one* CPU-bound process each, and $S_A = 1$ and $S_B = 100$, then the total number of shares is 101. Each project cannot use more than one CPU because each project has only one running process. Because no competition exists between projects for CPU resources in this configuration, projects *A* and *B* are each allocated 50 percent of all CPU resources. In this configuration, CPU share values are irrelevant. The projects' allocations would be the same (50/50), even if both projects were assigned zero shares.



Example 3: One Project Unable to Run

If *A* and *B* have two CPU-bound processes each, and project *A* is given 1 share and project *B* is given 0 shares, then project *B* is not allocated any CPU resources and project *A* is allocated all CPU resources. Processes in *B* always run at system priority 0, so they will never be able to run because processes in project *A* always have higher priorities.

FSS Configuration

Projects and Users

Projects are the workload containers in the FSS scheduler. Groups of users who are assigned to a project are treated as single controllable blocks. Note that you can create a project with its own number of shares for an individual user.

Users can be members of multiple projects that have different numbers of shares assigned. By moving processes from one project to another project, processes can be assigned CPU resources in varying amounts.

For more information on the `project`(4) database and name services, see “`project` Database” on page 23.

CPU Shares Configuration

The configuration of CPU shares is managed by the name service as a property of the `project` database.

When the first task (or process) that is associated with a project is created through the `setproject`(3PROJECT) library function, the number of CPU shares defined as resource control `project.cpu-shares` in the `project` database is passed to the kernel. A project that does not have the `project.cpu-shares` resource control defined is assigned one share.

In the following example, this entry in the `/etc/project` file sets the number of shares for project *x-files* to *5*:

```
x-files:100::::project.cpu-shares=(privileged,5,none)
```

If you alter the number of CPU shares allocated to a project in the database when processes are already running, the number of shares for that project will not be modified at that point. The project must be restarted for the change to become effective.

If you want to temporarily change the number of shares assigned to a project without altering the project's attributes in the `project` database, use the `prctl` command. For example, to change the value of project *x-files*'s `project.cpu-shares` resource control to *3* while processes associated with that project are running, type the following:

```
# prctl -r -n project.cpu-shares -v 3 -i project x-files
```

See the `prctl`(1) man page for more information.

`-r`	Replaces the current value for the named resource control.
`-n` *name*	Specifies the name of the resource control.
`-v` *val*	Specifies the value for the resource control.
`-i` *idtype*	Specifies the ID type of the next argument.
x-files	Specifies the object of the change. In this instance, project *x-files* is the object.

Project `system` with project ID 0 includes all system daemons that are started by the boot-time initialization scripts. `system` can be viewed as a project with an unlimited number of shares. This means that `system` is always scheduled first, regardless of how many shares have been given to other projects. If you do not want the `system` project to have unlimited shares, you can specify a number of shares for this project in the `project` database.

As stated previously, processes that belong to projects with zero shares are always given zero system priority. Projects with one or more shares are running with priorities one and higher.

Thus, projects with zero shares are only scheduled when CPU resources are available that are not requested by a nonzero share project.

The maximum number of shares that can be assigned to one project is 65535.

FSS and Processor Sets

The FSS can be used in conjunction with processor sets to provide more fine-grained controls over allocations of CPU resources among projects that run on each processor set than would be available with processor sets alone. The FSS scheduler treats processor sets as entirely independent partitions, with each processor set controlled independently with respect to CPU allocations.

The CPU allocations of projects running in one processor set are not affected by the CPU shares or activity of projects running in another processor set because the projects are not competing for the same resources. Projects only compete with each other if they are running within the same processor set.

The number of shares allocated to a project is system wide. Regardless of which processor set it is running on, each portion of a project is given the same amount of shares.

When processor sets are used, project CPU allocations are calculated for active projects that run within each processor set.

Project partitions that run on different processor sets might have different CPU allocations. The CPU allocation for each project partition in a processor set depends only on the allocations of other projects that run on the same processor set.

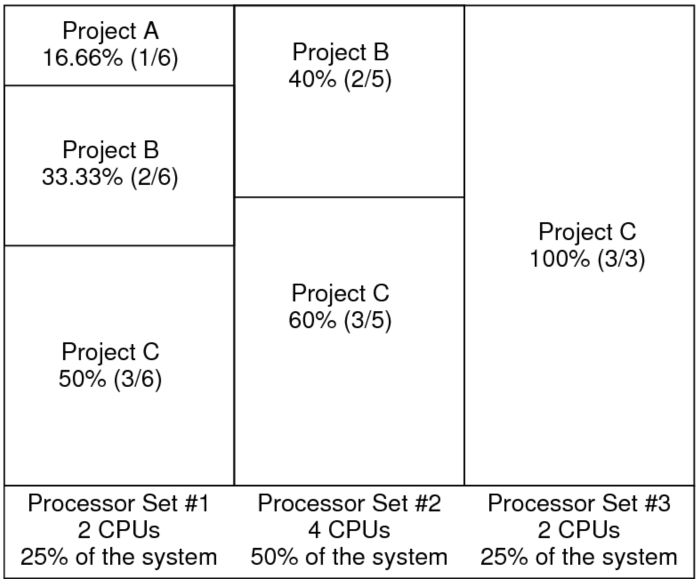
The performance and availability of applications that run within the boundaries of their processor sets are not affected by the introduction of new processor sets. The applications are also not affected by changes that are made to the share allocations of projects that run on other processor sets.

Empty processor sets (sets without processors in them) or processor sets without processes bound to them do not have any impact on the FSS scheduler behavior.

FSS and Processor Sets Examples

Assume that a server with eight CPUs is running several CPU-bound applications in projects *A*, *B*, and *C*. Project *A* is allocated one share, project *B* is allocated two shares, and project *C* is allocated three shares.

Project *A* is running only on processor set 1. Project *B* is running on processor sets 1 and 2. Project *C* is running on processor sets 1, 2, and 3. Assume that each project has enough processes to utilize all available CPU power. Thus, there is always competition for CPU resources on each processor set.



The total system-wide project CPU allocations on such a system are shown in the following table.

Project	**Allocation**
Project A	4% = $(1/6 \text{ X } 2/8)_{pset1}$
Project B	28% = $(2/6 \text{ X } 2/8)_{pset1}$+ $(2/5 * 4/8)_{pset2}$
Project C	67% = $(3/6 \text{ X } 2/8)_{pset1}$+ $(3/5 \text{ X } 4/8)_{pset2}$+ $(3/3 \text{ X } 2/8)_{pset3}$

These percentages do not match the corresponding amounts of CPU shares that are given to projects. However, within each processor set, the per-project CPU allocation ratios are proportional to their respective shares.

On the same system *without* processor sets, the distribution of CPU resources would be different, as shown in the following table.

Project	**Allocation**
Project A	16.66% = (1/6)

Project	Allocation
Project B	33.33% = (2/6)
Project C	50% = (3/6)

Combining FSS With Other Scheduling Classes

By default, the FSS scheduling class uses the same range of priorities (0 to 59) as the timesharing (TS), interactive (IA), and fixed priority (FX) scheduling classes. Therefore, you should avoid having processes from these scheduling classes share *the same* processor set. A mix of processes in the FSS, TS, IA, and FX classes could result in unexpected scheduling behavior.

With the use of processor sets, you can mix TS, IA, and FX with FSS in one system. However, all the processes that run on each processor set must be in *one* scheduling class, so they do not compete for the same CPUs. The FX scheduler in particular should not be used in conjunction with the FSS scheduling class unless processor sets are used. This action prevents applications in the FX class from using priorities high enough to starve applications in the FSS class.

You can mix processes in the TS and IA classes in the same processor set, or on the same system without processor sets.

The Oracle Solaris system also offers a real-time (RT) scheduler to users with root privileges. By default, the RT scheduling class uses system priorities in a different range (usually from 100 to 159) than FSS. Because RT and FSS are using *disjoint*, or non-overlapping, ranges of priorities, FSS can coexist with the RT scheduling class within the same processor set. However, the FSS scheduling class does not have any control over processes that run in the RT class.

For example, on a four-processor system, a single-threaded RT process can consume one entire processor if the process is CPU bound. If the system also runs FSS, regular user processes compete for the three remaining CPUs that are not being used by the RT process. Note that the RT process might not use the CPU continuously. When the RT process is idle, FSS utilizes all four processors.

You can type the following command to find out which scheduling classes the processor sets are running in and ensure that each processor set is configured to run either TS, IA, FX, or FSS processes.

```
$ ps -ef -o pset,class | grep -v CLS | sort | uniq
1 FSS
1 SYS
2 TS
2 RT
3 FX
```

Setting the Scheduling Class for the System

To set the default scheduling class for the system, see "How to Make FSS the Default Scheduler Class" on page 99, "Using the Fair Share Scheduler on an Oracle Solaris System With Zones Installed" in "Creating and Using Oracle Solaris Zones ", and dispadmin(1M). To move running processes into a different scheduling class, see "Configuring the FSS" on page 99 and priocntl(1).

Scheduling Class on a System with Zones Installed

Non-global zones use the default scheduling class for the system. If the system is updated with a new default scheduling class setting, non-global zones obtain the new setting when booted or rebooted.

The preferred way to use FSS in this case is to set FSS to be the system default scheduling class with the `dispadmin` command. All zones then benefit from getting a fair share of the system CPU resources. See "Using the Fair Share Scheduler on an Oracle Solaris System With Zones Installed" in "Creating and Using Oracle Solaris Zones " for more information on scheduling class when zones are in use.

For information about moving running processes into a different scheduling class without changing the default scheduling class and rebooting, see the priocntl(1) man page.

Commands Used With FSS

The commands that are shown in the following table provide the primary administrative interface to the fair share scheduler.

Command Reference	Description
priocntl(1)	Displays or sets scheduling parameters of specified processes, moves running processes into a different scheduling class.
ps(1)	Lists information about running processes, identifies in which scheduling classes processor sets are running.
dispadmin(1M)	Lists the available schedulers on the system. Sets the default scheduler for the system. Also used to examine and tune the FSS scheduler's time quantum value.
FSS(7)	Describes the fair share scheduler (FSS).

♦ ♦ ♦ CHAPTER 9

Administering the Fair Share Scheduler Tasks

This chapter describes how to use the fair share scheduler (FSS).

For an overview of the FSS, see Chapter 8, “About Fair Share Scheduler”. For information on scheduling class when zones are in use, see “Fair Share Scheduler on a System With Zones Installed” in “Creating and Using Oracle Solaris Zones ”.

Administering the Fair Share Scheduler Task Map

Task	Description	For Information
Monitor CPU usage.	Monitor the CPU usage of projects, and projects in processor sets.	“Monitoring the FSS” on page 98
Set the default scheduler class.	Make a scheduler such as the FSS the default scheduler for the system.	“How to Make FSS the Default Scheduler Class” on page 99
Move running processes from one scheduler class to a different scheduling class, such as the FSS class.	Manually move processes from one scheduling class to another scheduling class without changing the default scheduling class and rebooting.	“How to Manually Move Processes From the TS Class Into the FSS Class” on page 100
Move all running processes from all scheduling classes to a different scheduling class, such as the FSS class.	Manually move processes in all scheduling classes to another scheduling class without changing the default scheduling class and rebooting.	“How to Manually Move Processes From All User Classes Into the FSS Class” on page 100
Move a project's processes into a different scheduling class, such as the FSS class.	Manually move a project's processes from their current scheduling class to a different scheduling class.	“How to Manually Move a Project's Processes Into the FSS Class” on page 101
Examine and tune FSS parameters.	Tune the scheduler's time quantum value. *Time quantum* is the amount of time that a thread is allowed to run before it must relinquish the processor.	“How to Tune Scheduler Parameters” on page 101

Monitoring the FSS

You can use the `prstat` command described in the `prstat`(1M) man page to monitor CPU usage by active projects.

You can use the extended accounting data for tasks to obtain per-project statistics on the amount of CPU resources that are consumed over longer periods. See Chapter 4, "About Extended Accounting" for more information.

▼ How to Monitor System CPU Usage by Projects

● **To monitor the CPU usage of projects that run on the system, use the `prstat` command with the `-J` option.**

```
# prstat -J
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
  5107 root     4556K 3268K cpu0    59    0   0:00:00 0.0% prstat/1
  4570 root       83M   47M sleep   59    0   0:00:25 0.0% java/13
  5105 bobbyc   3280K 2364K sleep   59    0   0:00:00 0.0% su/1
  5106 root     3328K 2580K sleep   59    0   0:00:00 0.0% bash/1
     5 root        0K    0K sleep   99  -20   0:00:14 0.0% zpool-rpool/138
   333 daemon   7196K 2896K sleep   59    0   0:00:07 0.0% rcapd/1
    51 netcfg   4436K 3460K sleep   59    0   0:00:01 0.0% netcfgd/5
  2685 root     3328K 2664K sleep   59    0   0:00:00 0.0% bash/1
   101 netadm   4164K 2824K sleep   59    0   0:00:01 0.0% ipmgmtd/6
   139 root     6940K 3016K sleep   59    0   0:00:00 0.0% sysevent d/18
  5082 bobbyc   2236K 1700K sleep   59    0   0:00:00 0.0% csh/1
    45 root       15M 7360K sleep   59    0   0:00:01 0.0% dlmgmtd/7
    12 root       23M   22M sleep   59    0   0:00:45 0.0% svc.configd/22
    10 root       15M   13M sleep   59    0   0:00:05 0.0% svc.startd/19
   337 netadm   6768K 5620K sleep   59    0   0:00:01 0.0% nwamd/9
PROJID    NPROC  SWAP   RSS MEMORY      TIME  CPU PROJECT
     1        6   25M   18M   0.9%   0:00:00 0.0% user.root
     0       73  479M  284M    14%   0:02:31 0.0% system
     3        4   28M   24M   1.1%   0:00:26 0.0% default
    10        2   14M 7288K   0.3%   0:00:00 0.0% group.staff

Total: 85 processes, 553 lwps, load averages: 0.00, 0.00, 0.00
```

▼ How to Monitor CPU Usage by Projects in Processor Sets

● **To monitor the CPU usage of projects on a list of processor sets, type:**

% **prstat -J -C** *pset-list*

where *pset-list* is a list of processor set IDs that are separated by commas.

Configuring the FSS

The same commands that you use with other scheduling classes in the Oracle Solaris system can be used with FSS. You can set the scheduler class, configure the scheduler's tunable parameters, and configure the properties of individual processes.

Note that you can use `svcadm restart` to restart the scheduler service. See `svcadm`(1M) for more information.

Listing the Scheduler Classes on the System

To display the scheduler classes on the system, use the `dispadmin` command with the `-l` option.

```
$ dispadmin -l
CONFIGURED CLASSES
==================

SYS     (System Class)
TS      (Time Sharing)
SDC     (System Duty-Cycle Class)
FSS     (Fair Share)
FX      (Fixed Priority)
IA      (Interactive)
```

▼ How to Make FSS the Default Scheduler Class

The FSS must be the default scheduler on your system to have CPU shares assignment take effect.

Using a combination of the `priocntl` and `dispadmin` commands ensures that the FSS becomes the default scheduler immediately and also after reboot.

1. **Become root or assume an equivalent role.**
2. **Set the default scheduler for the system to be the FSS.**

   ```
   # dispadmin -d FSS
   ```

   This change takes effect on the next reboot. After reboot, every process on the system runs in the FSS scheduling class.

3. **Make this configuration take effect immediately, without rebooting.**

```
# priocntl -s -c FSS -i all
```

How to Manually Move Processes From the TS Class Into the FSS Class

You can manually move processes from one scheduling class to another scheduling class without changing the default scheduling class and rebooting. This procedure shows how to manually move processes from the TS scheduling class into the FSS scheduling class.

1. **Become root or assume an equivalent role.**
2. **Move the `init` process (pid 1) into the FSS scheduling class.**

```
# priocntl -s -c FSS -i pid 1
```

3. **Move all processes from the TS scheduling class into the FSS scheduling class.**

```
# priocntl -s -c FSS -i class TS
```

Note - All processes again run in the TS scheduling class after reboot.

How to Manually Move Processes From All User Classes Into the FSS Class

You might be using a default class other than TS. For example, your system might be running a window environment that uses the IA class by default. You can manually move all processes into the FSS scheduling class without changing the default scheduling class and rebooting.

1. **Become root or assume an equivalent role.**
2. **Move the `init` process (pid 1) into the FSS scheduling class.**

```
# priocntl -s -c FSS -i pid 1
```

3. **Move all processes from their current scheduling classes into the FSS scheduling class.**

```
# priocntl -s -c FSS -i all
```

Note - All processes again run in the default scheduling class after reboot.

How to Manually Move a Project's Processes Into the FSS Class

You can manually move a project's processes from their current scheduling class to the FSS scheduling class.

1. **Become root or assume an equivalent role.**
2. **Move processes that run in project ID *10* to the FSS scheduling class.**

```
# priocntl -s -c FSS -i projid 10
```

The project's processes again run in the default scheduling class after reboot.

How to Tune Scheduler Parameters

You can use the `dispadmin` command to display or change process scheduler parameters while the system is running. For example, you can use `dispadmin` to examine and tune the FSS scheduler's time quantum value. *Time quantum* is the amount of time that a thread is allowed to run before it must relinquish the processor.

To display the current time quantum for the FSS scheduler while the system is running, type:

```
$ dispadmin -c FSS -g
#
# Fair Share Scheduler Configuration
#
RES=1000
#
# Time Quantum
#
QUANTUM=110
```

When you use the `-g` option, you can also use the `-r` option to specify the resolution that is used for printing time quantum values. If no resolution is specified, time quantum values are displayed in milliseconds by default.

```
$ dispadmin -c FSS -g -r 100
#
# Fair Share Scheduler Configuration
#
```

```
RES=100
#
# Time Quantum
#
QUANTUM=11
```

To set scheduling parameters for the FSS scheduling class, use `dispadmin -s`. The values in *file* must be in the format output by the `-g` option. These values overwrite the current values in the kernel. Type the following:

```
$ dispadmin -c FSS -s file
```

♦ ♦ ♦ CHAPTER 10

About Physical Memory Control Using the Resource Capping Daemon

The resource capping daemon `rcapd` enables you to regulate physical memory consumption by processes running in projects that have resource caps defined. If you are running zones on your system, you can use `rcapd` from the global zone to regulate physical memory consumption in non-global zones. See “Physical Memory Control and the capped-memory Resource” in “Introduction to Oracle Solaris Zones ”.

The following topics are covered in this chapter.

- “Introduction to the Resource Capping Daemon” on page 103
- “How Resource Capping Works” on page 104
- “Attribute to Limit Physical Memory Usage for Projects” on page 104
- “`rcapd` Configuration” on page 105
- “Monitoring Resource Utilization With `rcapstat`” on page 109
- “Commands Used With `rcapd`” on page 110

For procedures using the `rcapd` utility, see Chapter 11, “Administering the Resource Capping Daemon Tasks”.

Introduction to the Resource Capping Daemon

A resource *cap* is an upper bound placed on the consumption of a resource, such as physical memory. Per-project physical memory caps are supported.

The resource capping daemon and its associated utilities provide mechanisms for physical memory resource cap enforcement and administration.

Like the resource control, the resource cap can be defined by using attributes of project entries in the `project` database. However, while resource controls are synchronously enforced by the kernel, resource caps are asynchronously enforced at the user level by the resource capping daemon. With asynchronous enforcement, a small delay occurs as a result of the sampling interval used by the daemon.

For information about `rcapd`, see the `rcapd`(1M) man page. For information about projects and the `project` database, see Chapter 2, “About Projects and Tasks” and the `project`(4) man page. For information about resource controls, see Chapter 6, “About Resource Controls”.

How Resource Capping Works

The daemon repeatedly samples the resource utilization of projects that have physical memory caps. The sampling interval used by the daemon is specified by the administrator. See “Determining Sample Intervals” on page 108 for additional information. When the system's physical memory utilization exceeds the threshold for cap enforcement, and other conditions are met, the daemon takes action to reduce the resource consumption of projects with memory caps to levels at or below the caps.

The virtual memory system divides physical memory into segments known as pages. Pages are the fundamental unit of physical memory in the Oracle Solaris memory management subsystem. To read data from a file into memory, the virtual memory system reads in one page at a time, or *pages in* a file. To reduce resource consumption, the daemon can *page out*, or relocate, infrequently used pages to a swap device, which is an area outside of physical memory.

The daemon manages physical memory by regulating the size of a project workload's resident set relative to the size of its working set. The resident set is the set of pages that are resident in physical memory. The working set is the set of pages that the workload actively uses during its processing cycle. The working set changes over time, depending on the process's mode of operation and the type of data being processed. Ideally, every workload has access to enough physical memory to enable its working set to remain resident. However, the working set can also include the use of secondary disk storage to hold the memory that does not fit in physical memory.

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

Attribute to Limit Physical Memory Usage for Projects

To define a physical memory resource cap for a project, establish a resident set size (RSS) cap by adding this attribute to the `project` database entry:

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

For example, the following line in the `/etc/project` file sets an RSS cap of 10 gigabytes for a project named `db`.

```
db:100::db,root::rcap.max-rss=10737418240
```

Note - The system might round the specified cap value to a page size.

You can also use the projmod command to set the rcap.max-rss attribute in the /etc/project file.

For more information, see Setting the Resident Set Size Cap.

rcapd Configuration

You use the rcapadm command to configure the resource capping daemon. You can perform the following actions:

- Set the threshold value for cap enforcement
- Set intervals for the operations performed by rcapd
- Enable or disable resource capping
- Display the current status of the configured resource capping daemon

To configure the daemon, you must be the root user or have the required administrative rights.

Configuration changes can be incorporated into rcapd according to the configuration interval (see "rcapd Operation Intervals" on page 107) or on demand by sending a SIGHUP (see the kill(1) man page).

If used without arguments, rcapadm displays the current status of the resource capping daemon if it has been configured.

The following subsections discuss cap enforcement, cap values, and rcapd operation intervals.

Using the Resource Capping Daemon on a System With Zones Installed

You can control resident set size (RSS) usage of a zone by setting the capped-memory resource when you configure the zone. For more information, see "Physical Memory Control and the capped-memory Resource" in "Introduction to Oracle Solaris Zones ". To use the capped-memory resource, the resource-cap package must be installed in the global zone. You can run rcapd *within* a zone, including the global zone, to enforce memory caps on projects in that zone.

You can set a temporary cap for the maximum amount of memory that can be consumed by a specified zone, until the next reboot. See "How to Specify a Temporary Resource Cap for a Zone" on page 115.

If you are using rcapd on a zone to regulate physical memory consumption by processes running in projects that have resource caps defined, you must configure the daemon in those zones.

When choosing memory caps for applications in different zones, you generally do not have to consider that the applications reside in different zones. The exception is per-zone services. Per-zone services consume memory. This memory consumption must be considered when determining the amount of physical memory for a system, as well as memory caps.

Memory Cap Enforcement Threshold

The *memory cap enforcement threshold* is the percentage of physical memory utilization on the system that triggers cap enforcement. When the system exceeds this utilization, caps are enforced. The physical memory used by applications and the kernel is included in this percentage. The percentage of utilization determines the way in which memory caps are enforced.

To enforce caps, memory can be paged out from project workloads.

- Memory can be paged out to reduce the size of the portion of memory that is over its cap for a given workload.
- Memory can be paged out to reduce the proportion of physical memory used that is over the memory cap enforcement threshold on the system.

A workload is permitted to use physical memory up to its cap. A workload can use additional memory as long as the system's memory utilization stays below the memory cap enforcement threshold.

To set the value for cap enforcement, see "How to Set the Memory Cap Enforcement Threshold" on page 113.

Determining Cap Values

If a project cap is set too low, there might not be enough memory for the workload to proceed effectively under normal conditions. The paging that occurs because the workload requires more memory has a negative effect on system performance.

Projects that have caps set too high can consume available physical memory before their caps are exceeded. In this case, physical memory is effectively managed by the kernel and not by rcapd.

In determining caps on projects, consider these factors.

Impact on I/O system
: The daemon can attempt to reduce a project workload's physical memory usage whenever the sampled usage exceeds the project's cap. During

cap enforcement, the swap devices and other devices that contain files that the workload has mapped are used. The performance of the swap devices is a critical factor in determining the performance of a workload that routinely exceeds its cap. The execution of the workload is similar to running it on a machine with the same amount of physical memory as the workload's cap.

Impact on CPU usage

The daemon's CPU usage varies with the number of processes in the project workloads it is capping and the sizes of the workloads' address spaces.

A small portion of the daemon's CPU time is spent sampling the usage of each workload. Adding processes to workloads increases the time spent sampling usage.

Another portion of the daemon's CPU time is spent enforcing caps when they are exceeded. The time spent is proportional to the amount of virtual memory involved. CPU time spent increases or decreases in response to corresponding changes in the total size of a workload's address space. This information is reported in the `vm` column of `rcapstat` output. See "Monitoring Resource Utilization With rcapstat" on page 109 and the `rcapstat(1)` man page for more information.

Reporting on shared memory

The `rcapd` daemon reports the RSS of pages of memory that are shared with other processes or mapped multiple times within the same process as a reasonably accurate estimate. If processes in different projects share the same memory, then that memory will be counted towards the RSS total for all projects sharing the memory.

The estimate is usable with workloads such as databases, which utilize shared memory extensively. For database workloads, you can also sample a project's regular usage to determine a suitable initial cap value by using output from the `-J` or `-Z` options of the `prstat` command. For more information, see the `prstat(1M)` man page.

rcapd Operation Intervals

You can tune the intervals for the periodic operations performed by `rcapd`.

All intervals are specified in seconds. The `rcapd` operations and their default interval values are described in the following table.

Operation	Default Interval Value in Seconds	Description
`scan`	15	Number of seconds between scans for processes that have joined or left

Operation	Default Interval Value in Seconds	Description
		a project workload. Minimum value is 1 second.
sample	5	Number of seconds between samplings of resident set size and subsequent cap enforcements. Minimum value is 1 second.
report	5	Number of seconds between updates to paging statistics. If set to 0, statistics are not updated, and output from rcapstat is not current.
config	60	Number of seconds between reconfigurations. In a reconfiguration event, rcapadm reads the configuration file for updates, and scans the project database for new or revised project caps. Sending a SIGHUP to rcapd causes an immediate reconfiguration.

To tune intervals, see "How to Set Operation Intervals" on page 113.

Determining rcapd Scan Intervals

The scan interval controls how often rcapd looks for new processes. On systems with many processes running, the scan through the list takes more time, so it might be preferable to lengthen the interval in order to reduce the overall CPU time spent. However, the scan interval also represents the minimum amount of time that a process must exist to be attributed to a capped workload. If there are workloads that run many short-lived processes, rcapd might not attribute the processes to a workload if the scan interval is lengthened.

Determining Sample Intervals

The sample interval configured with rcapadm is the shortest amount of time rcapd waits between sampling a workload's usage and enforcing the cap if it is exceeded. If you reduce this interval, rcapd will, under most conditions, enforce caps more frequently, possibly resulting in increased I/O due to paging. However, a shorter sample interval can also lessen the impact that a sudden increase in a particular workload's physical memory usage might have on other workloads. The window between samplings, in which the workload can consume memory unhindered and possibly take memory from other capped workloads, is narrowed.

If the sample interval specified to rcapstat is shorter than the interval specified to rcapd with rcapadm, the output for some intervals can be zero. This situation occurs because rcapd does not update statistics more frequently than the interval specified with `rcapadm`. The interval specified with `rcapadm` is independent of the sampling interval used by `rcapstat`.

Monitoring Resource Utilization With `rcapstat`

Use `rcapstat` to monitor the resource utilization of capped projects. To view an example `rcapstat` report, see "Producing Reports With rcapstat" on page 115.

You can set the sampling interval for the report and specify the number of times that statistics are repeated.

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

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

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

The following table defines the column headings in an `rcapstat` report.

`rcapstat` Column Headings	Description
`id`	The project ID of the capped project.
`project`	The project name.
`nproc`	The number of processes in the project.
`vm`	The total amount of virtual memory size used by processes in the project, including all mapped files and devices, in kilobytes (K), megabytes (M), or gigabytes (G).
`rss`	The estimated amount of the total resident set size (RSS) of the processes in the project, in kilobytes (K), megabytes (M), or gigabytes (G), not accounting for pages that are shared.
`cap`	The RSS cap defined for the project. See "Attribute to Limit Physical Memory Usage for Projects" on page 104 or the `rcapd`(1M) man page for information about how to specify memory caps.
`at`	The total amount of memory that `rcapd` attempted to page out since the last `rcapstat` sample.

rcapstat Column Headings	Description
avgat	The average amount of memory that rcapd attempted to page out during each sample cycle that occurred since the last rcapstat sample. The rate at which rcapd samples collection RSS can be set with rcapadm. See "rcapd Operation Intervals" on page 107.
pg	The total amount of memory that rcapd successfully paged out since the last rcapstat sample.
avgpg	An estimate of the average amount of memory that rcapd successfully paged out during each sample cycle that occurred since the last rcapstat sample. The rate at which rcapd samples process RSS sizes can be set with rcapadm. See "rcapd Operation Intervals" on page 107.

Commands Used With rcapd

Command Reference	Description
rcapstat(1)	Monitors the resource utilization of capped projects.
rcapadm(1M)	Configures the resource capping daemon, displays the current status of the resource capping daemon if it has been configured, and enables or disables resource capping. Also used to set a temporary memory cap.
rcapd(1M)	The resource capping daemon.

◆ ◆ ◆ CHAPTER 11

Administering the Resource Capping Daemon Tasks

This chapter contains procedures for configuring and using the resource capping daemon `rcapd`.

For an overview of `rcapd`, see Chapter 10, "About Physical Memory Control Using the Resource Capping Daemon".

Setting the Resident Set Size Cap

Define a physical memory resource resident set size (RSS) cap for a project by adding an `rcap.max-rss` attribute to the `project` database entry.

▼ How to Add an `rcap.max-rss` Attribute for a Project

1. **Become root or assume an equivalent role.**
2. **Add this attribute to the `/etc/project` file:**

   `rcap.max-rss=`*value*

Example 11-1 RSS Project Cap

The following line in the `/etc/project` file sets an RSS cap of 10 gigabytes for a project named `db`.

```
db:100::db,root::rcap.max-rss=10737418240
```

Note that the system might round the specified cap value to a page size.

▼ How to Use the `projmod` Command to Add an `rcap.max-rss` Attribute for a Project

1. **Become root or assume an equivalent role.**

2. **Set an `rcap.max-rss` attribute of 10 gigabytes in the `/etc/project` file, in this case for a project named `db`.**

```
# projmod -a -K rcap.max-rss=10GB db
```

The `/etc/project` file then contains the line:

```
db:100::db,root::rcap.max-rss=10737418240
```

Configuring and Using the Resource Capping Daemon Task Map

Task	Description	For Instructions
Set the memory cap enforcement threshold.	Configure a cap that will be enforced when the physical memory available to processes is low.	"How to Set the Memory Cap Enforcement Threshold" on page 113
Set the operation interval.	The interval is applied to the periodic operations performed by the resource capping daemon.	"How to Set Operation Intervals" on page 113
Enable resource capping.	Activate resource capping on your system.	"How to Enable Resource Capping" on page 114
Disable resource capping.	Deactivate resource capping on your system.	"How to Disable Resource Capping" on page 114
Report cap and project information.	View example commands for producing reports.	"Reporting Cap and Project Information" on page 116
Monitor a project's resident set size.	Produce a report on the resident set size of a project.	"Monitoring the RSS of a Project" on page 116
Determine a project's working set size.	Produce a report on the working set size of a project.	"Determining the Working Set Size of a Project" on page 117
Report on memory utilization and memory caps.	Print a memory utilization and cap enforcement line at the end of the report for each interval.	"Reporting Memory Utilization and the Memory Cap Enforcement Threshold" on page 118

Administering the Resource Capping Daemon With `rcapadm`

This section contains procedures for configuring the resource capping daemon with `rcapadm`. See “`rcapd` Configuration” on page 105 and the `rcapadm`(1M) man page for more information. Using the `rcapadm` to specify a temporary resource cap for a zone is also covered.

If used without arguments, `rcapadm` displays the current status of the resource capping daemon if it has been configured.

▼ How to Set the Memory Cap Enforcement Threshold

Caps can be configured so that they will not be enforced until the physical memory available to processes is low. See “Memory Cap Enforcement Threshold” on page 106 for more information.

The minimum (and default) value is `0`, which means that memory caps are always enforced. To set a different minimum, follow this procedure.

1. **Become root or assume an equivalent role.**

2. **Use the `-c` option of `rcapadm` to set a different physical memory utilization value for memory cap enforcement.**

   ```
   # rcapadm -c percent
   ```

   percent is in the range 0 to 100. Higher values are less restrictive. A higher value means capped project workloads can execute without having caps enforced until the system's memory utilization exceeds this threshold.

See Also To display the current physical memory utilization and the cap enforcement threshold, see “Reporting Memory Utilization and the Memory Cap Enforcement Threshold” on page 118.

▼ How to Set Operation Intervals

“`rcapd` Operation Intervals” on page 107 contains information about the intervals for the periodic operations performed by `rcapd`. To set operation intervals using `rcapadm`, follow this procedure.

1. **Become root or assume an equivalent role.**

2. **Use the `-i` option to set interval values.**

   `# rcapadm -i` *interval=value*`,...,`*interval=value*

   Note - All interval values are specified in seconds.

How to Enable Resource Capping

There are three ways to enable resource capping on your system. Enabling resource capping also sets the `/etc/rcap.conf` file with default values.

1. **Become root or assume an equivalent role.**
2. **Enable the resource capping daemon in one of the following ways:**
   - Turn on resource capping using the `svcadm` command.

     ```
     # svcadm enable rcap
     ```
   - Enable the resource capping daemon so that it will be started now and also be started each time the system is booted:

     ```
     # rcapadm -E
     ```
   - Enable the resource capping daemon at boot without starting it now by also specifying the `-n` option:

     ```
     # rcapadm -n -E
     ```

How to Disable Resource Capping

There are three ways to disable resource capping on your system.

1. **Become root or assume an equivalent role.**
2. **Disable the resource capping daemon in one of the following ways:**
   - **Turn off resource capping using the `svcadm` command.**

     ```
     # svcadm disable rcap
     ```
   - **To disable the resource capping daemon so that it will be stopped now and not be started when the system is booted, type:**

```
# rcapadm -D
```

- **To disable the resource capping daemon without stopping it, also specify the `-n` option:**

```
# rcapadm -n -D
```

Tip - Use `rcapadm -D` to safely disable `rcapd`. If the daemon is killed (see the `kill`(1) man page), processes might be left in a stopped state and need to be manually restarted. To resume a process running, use the `prun` command. See the `prun`(1) man page for more information.

How to Specify a Temporary Resource Cap for a Zone

This procedure is used to allocate the maximum amount of memory that can be consumed by a specified zone. This value lasts only until the next reboot. To set a persistent cap, use the `zonecfg` command.

1. **Become root or assume an equivalent role.**
2. **Set a maximum memory value of 512 megabytes for the zone `my-zone`.**

```
# rcapadm -z testzone -m 512M
```

Producing Reports With `rcapstat`

Use `rcapstat` to report resource capping statistics. "Monitoring Resource Utilization With `rcapstat`" on page 109 explains how to use the `rcapstat` command to generate reports. That section also describes the column headings in the report. The `rcapstat`(1) man page also contains this information.

The following subsections use examples to illustrate how to produce reports for specific purposes.

Reporting Cap and Project Information

In this example, caps are defined for two projects associated with two users. `user1` has a cap of 50 megabytes, and `user2` has a cap of 10 megabytes.

The following command produces five reports at 5-second sampling intervals.

```
user1machine% rcapstat 5 5
    id project  nproc     vm     rss    cap    at avgat    pg avgpg
112270   user1     24   123M    35M    50M   50M    0K 3312K    0K
 78194   user2      1  2368K  1856K    10M    0K    0K    0K    0K
    id project  nproc     vm     rss    cap    at avgat    pg avgpg
112270   user1     24   123M    35M    50M    0K    0K    0K    0K
 78194   user2      1  2368K  1856K    10M    0K    0K    0K    0K
    id project  nproc     vm     rss    cap    at avgat    pg avgpg
112270   user1     24   123M    35M    50M    0K    0K    0K    0K
 78194   user2      1  2368K  1928K    10M    0K    0K    0K    0K
    id project  nproc     vm     rss    cap    at avgat    pg avgpg
112270   user1     24   123M    35M    50M    0K    0K    0K    0K
 78194   user2      1  2368K  1928K    10M    0K    0K    0K    0K
    id project  nproc     vm     rss    cap    at avgat    pg avgpg
112270   user1     24   123M    35M    50M    0K    0K    0K    0K
 78194   user2      1  2368K  1928K    10M    0K    0K    0K    0K
```

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

The subsequent reports show no significant activity.

Monitoring the RSS of a Project

The following example uses project `user1`, which has an RSS in excess of its RSS cap.

The following command produces five reports at 5-second sampling intervals.

```
user1machine% rcapstat 5 5

    id project  nproc    vm    rss   cap    at avgat     pg  avgpg
376565   user1      3 6249M 6144M 6144M  690M  220M  5528K  2764K
376565   user1      3 6249M 6144M 6144M    0M  131M  4912K  1637K
376565   user1      3 6249M 6171M 6144M   27M  147M  6048K  2016K
376565   user1      3 6249M 6146M 6144M 4872M  174M  4368K  1456K
376565   user1      3 6249M 6156M 6144M   12M  161M  3376K  1125K
```

The `user1` project has three processes that are actively using physical memory. The positive values in the `pg` column indicate that `rcapd` is consistently paging out memory as it attempts to meet the cap by lowering the physical memory utilization of the project's processes. However, `rcapd` does not succeed in keeping the RSS below the cap value. This is indicated by the varying `rss` values that do not show a corresponding decrease. As soon as memory is paged out, the workload uses it again and the RSS count goes back up. This means that all of the project's resident memory is being actively used and the working set size (*WSS*) is greater than the cap. Thus, `rcapd` is forced to page out some of the working set to meet the cap. Under this condition, the system will continue to experience high page fault rates, and associated I/O, until one of the following occurs:

- The WSS becomes smaller.
- The cap is raised.
- The application changes its memory access pattern.

In this situation, shortening the sample interval might reduce the discrepancy between the RSS value and the cap value by causing `rcapd` to sample the workload and enforce caps more frequently.

Note - A page fault occurs when either a new page must be created or the system must copy in a page from a swap device.

Determining the Working Set Size of a Project

The following example is a continuation of the previous example, and it uses the same project.

The previous example shows that the `user1` project is using more physical memory than its cap allows. This example shows how much memory the project workload requires.

```
user1machine% rcapstat 5 5
    id project  nproc    vm   rss   cap    at avgat     pg  avgpg
376565   user1      3 6249M 6144M 6144M  690M    0K   689M     0K
376565   user1      3 6249M 6144M 6144M    0K    0K     0K     0K
376565   user1      3 6249M 6171M 6144M   27M    0K    27M     0K
376565   user1      3 6249M 6146M 6144M 4872K    0K  4816K     0K
376565   user1      3 6249M 6156M 6144M   12M    0K    12M     0K
376565   user1      3 6249M 6150M 6144M 5848K    0K  5816K     0K
376565   user1      3 6249M 6155M 6144M   11M    0K    11M     0K
376565   user1      3 6249M 6150M   10G   32K    0K    32K     0K
376565   user1      3 6249M 6214M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
```

Halfway through the cycle, the cap on the `user1` project was increased from 6 gigabytes to 10 gigabytes. This increase stops cap enforcement and allows the resident set size to grow, limited only by other processes and the amount of memory in the machine. The rss column might stabilize to reflect the project working set size (WSS), 6247M in this example. This is the minimum cap value that allows the project's processes to operate without continuously incurring page faults.

While the cap on user1 is 6 gigabytes, in every 5–second sample interval the RSS decreases and I/O increases as rcapd pages out some of the workload's memory. Shortly after a page out completes, the workload, needing those pages, pages them back in as it continues running. This cycle repeats until the cap is raised to 10 gigabytes, approximately halfway through the example. The RSS then stabilizes at 6.1 gigabytes. Since the workload's RSS is now below the cap, no more paging occurs. The I/O associated with paging stops as well. Thus, the project required 6.1 gigabytes to perform the work it was doing at the time it was being observed.

Also see the vmstat(1M) and iostat(1M) man pages.

Reporting Memory Utilization and the Memory Cap Enforcement Threshold

You can use the -g option of rcapstat to report the following:

- Current physical memory utilization as a percentage of physical memory installed on the system
- System memory cap enforcement threshold set by rcapadm

The -g option causes a memory utilization and cap enforcement line to be printed at the end of the report for each interval.

```
# rcapstat -g
    id project   nproc    vm   rss   cap    at avgat   pg  avgpg
376565     rcap       0    0K    0K   10G    0K    0K   0K     0K
physical memory utilization: 55%   cap enforcement threshold: 0%
    id project   nproc    vm   rss   cap    at avgat   pg  avgpg
376565     rcap       0    0K    0K   10G    0K    0K   0K     0K
physical memory utilization: 55%   cap enforcement threshold: 0%
```

CHAPTER 12

About Resource Pools

This chapter discusses the following technologies:

- Resource pools, which are used for partitioning machine resources
- Dynamic resource pools (DRPs), which dynamically adjust each resource pool's resource allocation to meet established system goals

Resource pools and dynamic resource pools are services in the Oracle Solaris service management facility (SMF). Each of these services is enabled separately.

The following topics are covered in this chapter:

- "Introduction to Resource Pools" on page 120
- "Introduction to Dynamic Resource Pools" on page 121
- "About Enabling and Disabling Resource Pools and Dynamic Resource Pools" on page 121
- "Resource Pools Used in Zones" on page 121
- "When to Use Pools" on page 122
- "Resource Pools Framework" on page 123
- "Implementing Pools on a System" on page 125
- "`project.pool` Attribute" on page 125
- "Dynamic Reconfiguration Operations and Resource Pools" on page 125
- "Creating Pools Configurations" on page 126
- "Directly Manipulating the Dynamic Configuration" on page 127
- "`poold` Overview" on page 127
- "Managing Dynamic Resource Pools" on page 128
- "Configuration Constraints and Objectives" on page 128
- "`poold` Functionality That Can Be Configured" on page 133
- "How Dynamic Resource Allocation Works" on page 136
- "Using `poolstat` to Monitor the Pools Facility and Resource Utilization" on page 139
- "Commands Used With the Resource Pools Facility" on page 140

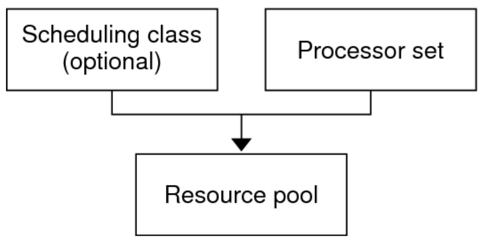
For procedures using this functionality, see Chapter 13, "Creating and Administering Resource Pools Tasks".

Introduction to Resource Pools

Resource pools enable you to separate workloads so that workload consumption of certain resources does not overlap. This resource reservation helps to achieve predictable performance on systems with mixed workloads.

Resource pools provide a persistent configuration mechanism for processor set (pset) configuration and, optionally, scheduling class assignment.

FIGURE 12-1 Resource Pool Framework



A pool can be thought of as a specific binding of the various resource sets that are available on your system. You can create pools that represent different kinds of possible resource combinations:

```
pool1: pset_default
pool2: pset1
pool3: pset1, pool.scheduler="FSS"
```

By grouping multiple partitions, pools provide a handle to associate with labeled workloads. Each project entry in the `/etc/project` file can have a single pool associated with that entry, which is specified using the `project.pool` attribute.

When pools are enabled, a *default pool* and a *default processor set* form the base configuration. Additional user-defined pools and processor sets can be created and added to the configuration. A CPU can only belong to one processor set. User-defined pools and processor sets can be destroyed. The default pool and the default processor set cannot be destroyed.

The default pool has the `pool.default` property set to `true`. The default processor set has the `pset.default` property set to `true`. Thus, both the default pool and the default processor set can be identified even if their names have been changed.

The user-defined pools mechanism is primarily for use on large machines of more than four CPUs. However, small machines can still benefit from this functionality. On small machines, you can create pools that share noncritical resource partitions. The pools are separated only on the basis of critical resources.

Introduction to Dynamic Resource Pools

Dynamic resource pools provide a mechanism for dynamically adjusting each pool's resource allocation in response to system events and application load changes. DRPs simplify and reduce the number of decisions required from an administrator. Adjustments are automatically made to preserve the system performance goals specified by an administrator. The changes made to the configuration are logged. These capabilities are primarily enacted through the resource controller `poold`, a system daemon that should always be active when dynamic resource allocation is required. Periodically, `poold` examines the load on the system and determines whether intervention is required to enable the system to maintain optimal performance with respect to resource consumption. The `poold` configuration is held in the `libpool` configuration. For more information on `poold`, see the `poold`(1M) man page.

About Enabling and Disabling Resource Pools and Dynamic Resource Pools

To enable and disable resource pools and dynamic resource pools, see “Enabling and Disabling the Pools Facility” on page 144.

Resource Pools Used in Zones

As an alternative to associating a zone with a configured resource pool on your system, you can use the `zonecfg` command to create a temporary pool that is in effect while the zone is running. See “Creating and Using Oracle Solaris Zones ” for more information.

On a system that has zones enabled, a non-global zone can be associated with one resource pool, although the pool need not be exclusively assigned to a particular zone. Moreover, you cannot bind individual processes in non-global zones to a different pool by using the `poolbind` command from the global zone. To associate a non-global zone with a pool, see “Creating and Using Oracle Solaris Zones ”.

Note that if you set a scheduling class for a pool and you associate a non-global zone with that pool, the zone uses that scheduling class by default.

If you are using dynamic resource pools, the scope of an executing instance of `poold` is limited to the global zone.

The `poolstat` utility run in a non-global zone displays only information about the pool associated with the zone. The `pooladm` command run without arguments in a non-global zone displays only information about the pool associated with the zone.

For information about resource pool commands, see "Commands Used With the Resource Pools Facility" on page 140.

When to Use Pools

Resource pools offer a versatile mechanism that can be applied to many administrative scenarios.

Batch compute server
: Use pools functionality to split a server into two pools. One pool is used for login sessions and interactive work by timesharing users. The other pool is used for jobs that are submitted through the batch system.

Application or database server
: Partition the resources for interactive applications in accordance with the applications' requirements.

Turning on applications in phases
: Set user expectations.

  You might initially deploy a machine that is running only a fraction of the services that the machine is ultimately expected to deliver. User difficulties can occur if reservation-based resource management mechanisms are not established when the machine comes online.

  For example, the fair share scheduler optimizes CPU utilization. The response times for a machine that is running only one application can be misleadingly fast. Users will not see these response times with multiple applications loaded. By using separate pools for each application, you can place a ceiling on the number of CPUs available to each application before you deploy all applications.

Complex timesharing server
: Partition a server that supports large user populations. Server partitioning provides an isolation mechanism that leads to a more predictable per-user response.

  By dividing users into groups that bind to separate pools, and using the fair share scheduling (FSS) facility, you can tune CPU allocations to favor sets of users that have priority. This assignment can be based on user role, accounting chargeback, and so forth.

Workloads that change seasonally
: Use resource pools to adjust to changing demand.

  Your site might experience predictable shifts in workload demand over long periods of time, such as monthly, quarterly, or annual cycles. If your site experiences these shifts, you can alternate between multiple pools

	configurations by invoking `pooladm` from a `cron` job. (See "Resource Pools Framework" on page 123.)
Real-time applications	Create a real-time pool by using the RT scheduler and designated processor resources.
System utilization	Enforce system goals that you establish. Use the automated pools daemon functionality to identify available resources and then monitor workloads to detect when your specified objectives are no longer being satisfied. The daemon can take corrective action if possible, or the condition can be logged.

Resource Pools Framework

The `/etc/pooladm.conf` configuration file describes the static pools configuration. A static configuration represents the way in which an administrator would like a system to be configured with respect to resource pools functionality. An alternate file name can be specified.

When the service management facility (SMF) or the `pooladm -e` command is used to enable the resource pools framework, then, if an `/etc/pooladm.conf` file exists, the configuration contained in the file is applied to the system.

The kernel holds information about the disposition of resources within the resource pools framework. This is known as the dynamic configuration, and it represents the resource pools functionality for a particular system at a point in time. The dynamic configuration can be viewed by using the `pooladm` command. Note that the order in which properties are displayed for pools and resource sets can vary. Modifications to the dynamic configuration are made in the following ways:

- Indirectly, by applying a static configuration file
- Directly, by using the `poolcfg` command with the `-d` option

More than one static pools configuration file can exist, for activation at different times. You can alternate between multiple pools configurations by invoking `pooladm` from a `cron` job. See the `cron`(1M) man page for more information on the `cron` utility.

By default, the resource pools framework is not active. Resource pools must be enabled to create or modify the dynamic configuration. Static configuration files can be manipulated with the `poolcfg` or `libpool` commands even if the resource pools framework is disabled. Static configuration files cannot be created if the pools facility is not active. For more information on the configuration file, see "Creating Pools Configurations" on page 126.

The commands used with resource pools and the `poold` system daemon are described in the following man pages:

- pooladm(1M)
- poolbind(1M)
- poolcfg(1M)
- poold(1M)
- poolstat(1M)
- libpool(3LIB)

/etc/pooladm.conf Contents

All resource pool configurations, including the dynamic configuration, can contain the following elements.

system	Properties affecting the total behavior of the system
pool	A resource pool definition
pset	A processor set definition
cpu	A processor definition

All of these elements have properties that can be manipulated to alter the state and behavior of the resource pools framework. For example, the pool property `pool.importance` indicates the relative importance of a given pool. This property is used for possible resource dispute resolution. For more information, see libpool(3LIB).

Pools Properties

The pools facility supports named, typed properties that can be placed on a pool, resource, or component. Administrators can store additional properties on the various pool elements. A property namespace similar to the project attribute is used.

For example, the following comment indicates that a given pset is associated with a particular `Datatree` database.

```
Datatree,pset.dbname=warehouse
```

For additional information about property types, see “poold Properties” on page 132.

Note - A number of special properties are reserved for internal use and cannot be set or removed. See the libpool(3LIB) man page for more information.

Implementing Pools on a System

User-defined pools can be implemented on a system by using one of these methods.

- When the Oracle Solaris software boots, an `init` script checks to see if the `/etc/pooladm.conf` file exists. If this file is found and pools are enabled, then `pooladm` is invoked to make this configuration the active pools configuration. The system creates a dynamic configuration to reflect the organization that is requested in `/etc/pooladm.conf`, and the machine's resources are partitioned accordingly.
- When the Oracle Solaris system is running, a pools configuration can either be activated if it is not already present, or modified by using the `pooladm` command. By default, the `pooladm` command operates on `/etc/pooladm.conf`. However, you can optionally specify an alternate location and file name, and use that file to update the pools configuration.

For information about enabling and disabling resource pools, see "Enabling and Disabling the Pools Facility" on page 144. The pools facility cannot be disabled when there are user-defined pools or resources in use.

To configure resource pools, you must have root privileges or have the required rights profile.

The `poold` resource controller is started with the dynamic resource pools facility.

`project.pool` Attribute

The `project.pool` attribute can be added to a project entry in the `/etc/project` file to associate a single pool with that entry. New work that is started on a project is bound to the appropriate pool. See Chapter 2, "About Projects and Tasks" for more information.

For example, you can use the `projmod` command to set the `project.pool` attribute for the project *sales* in the `/etc/project` file:

```
# projmod -a -K project.pool=mypool sales
```

SPARC: Dynamic Reconfiguration Operations and Resource Pools

Dynamic Reconfiguration (DR) enables you to reconfigure hardware while the system is running. A DR operation can increase, reduce, or have no effect on a given type of resource. Because DR can affect available resource amounts, the pools facility must be included in these operations. When a DR operation is initiated, the pools framework acts to validate the configuration.

If the DR operation can proceed without causing the current pools configuration to become invalid, then the private configuration file is updated. An invalid configuration is one that cannot be supported by the available resources.

If the DR operation would cause the pools configuration to be invalid, then the operation fails and you are notified by a message to the message log. If you want to force the configuration to completion, you must use the DR force option. The pools configuration is then modified to comply with the new resource configuration. For information on the DR process and the force option, see the dynamic reconfiguration user guide for your Sun hardware.

If you are using dynamic resource pools, note that it is possible for a partition to move out of `poold` control while the daemon is active. For more information, see "Identifying a Resource Shortage" on page 137.

Creating Pools Configurations

The configuration file contains a description of the pools to be created on the system. The file describes the elements that can be manipulated.

- system
- pool
- pset
- cpu

See `poolcfg`(1M) for more information on elements that be manipulated.

When pools are enabled, you can create a structured `/etc/pooladm.conf` file in two ways.

- You can use the `pooladm` command with the `-s` option to discover the resources on the current system and place the results in a configuration file.

  This method is preferred. All active resources and components on the system that are capable of being manipulated by the pools facility are recorded. The resources include existing processor set configurations. You can then modify the configuration to rename the processor sets or to create additional pools if necessary.
- You can use the `poolcfg` command with the `-c` option and the `discover` or `create system` *name* subcommands to create a new pools configuration.

  These options are maintained for backward compatibility with previous releases.

Use `poolcfg` or `libpool` to modify the `/etc/pooladm.conf` file. Do not directly edit this file.

Specific Assignment of CPUs, Cores, and Sockets

Use the subcommands `assign` and `unassign` assign specific CPUs, cores, and sockets.

The `assign`and `unassign` subcommands are applicable to both the persistent and runtime configurations of the pools. Using `assign` and setting `pset.min` and `pset.max` directly are are mutually exclusive. Each method overwrites the configuration set by the other. CPUs configured to psets using the `pset.min` and `pset.max` properties are considered *allocated* rather than assigned. `assign` and `unassign` add and remove specific CPUs to or from a pset. The first `assign` will clear any pset configuration set up by a previous allocation. Use `unassign` only after a successful `assign`. `unassign` cannot be used to manipulate CPUs from allocated psets.

Also see "dedicated-cpu Resource" in "Introduction to Oracle Solaris Zones ".

Directly Manipulating the Dynamic Configuration

It is possible to directly manipulate CPU resource types in the dynamic configuration by using the `poolcfg` command with the `-d` option. There are two methods used to transfer resources.

- You can make a general request to transfer any available identified resources between sets.
- You can transfer resources with specific IDs to a target set. Note that the system IDs associated with resources can change when the resource configuration is altered or after a system reboot.

For an example, see "Transferring Resources" on page 158.

If DRP is in use, note that the resource transfer might trigger action from `poold`. See "`poold` Overview" on page 127 for more information.

`poold` Overview

The pools resource controller, `poold`, uses system targets and observable statistics to preserve the system performance goals that you specify. This system daemon should always be active when dynamic resource allocation is required.

The `poold` resource controller identifies available resources and then monitors workloads to determine when the system usage objectives are no longer being met. `poold` then considers alternative configurations in terms of the objectives, and remedial action is taken. If possible, the resources are reconfigured so that objectives can be met. If this action is not possible, the daemon logs that user-specified objectives can no longer be achieved. Following a reconfiguration, the daemon resumes monitoring workload objectives.

poold maintains a decision history that it can examine. The decision history is used to eliminate reconfigurations that historically did not show improvements.

Note that a reconfiguration can also be triggered asynchronously if the workload objectives are changed or if the resources available to the system are modified.

Managing Dynamic Resource Pools

The DRP service is managed by the service management facility (SMF) under the service identifier svc:/system/pools/dynamic.

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using the svcadm command. The service's status can be queried using the svcs command. See the svcs(1) andsvcadm(1M) man pages for more information.

The SMF interface is the preferred method for controlling DRP, but for backward compatibility, the following methods can also be used.

- If dynamic resource allocation is not required, poold can be stopped with the SIGQUIT or the SIGTERM signal. Either of these signals causes poold to terminate gracefully.
- Although poold will automatically detect changes in the resource or pools configuration, you can also force a reconfiguration to occur by using the SIGHUP signal.

Configuration Constraints and Objectives

When making changes to a configuration, poold acts on directions that you provide. You specify these directions as a series of constraints and objectives. poold uses your specifications to determine the relative value of different configuration possibilities in relation to the existing configuration. poold then changes the resource assignments of the current configuration to generate new candidate configurations.

Configuration Constraints

Constraints affect the range of possible configurations by eliminating some of the potential changes that could be made to a configuration. The following constraints, which are specified in the libpool configuration, are available.

- The minimum and maximum CPU allocations
- Pinned components that are not available to be moved from a set
- The importance factor of the pool

See the `libpool`(3LIB) man page and "Pools Properties" on page 124 for more information about pools properties.

See "How to Set Configuration Constraints" on page 154 for usage instructions.

`pset.min` Property and `pset.max` Property Constraints

These two properties place limits on the number of processors that can be allocated to a processor set, both minimum and maximum. See Table 12-1 for more details about these properties.

Within these constraints, a resource partition's resources are available to be allocated to other resource partitions in the same Oracle Solaris instance. Access to the resource is obtained by binding to a pool that is associated with the resource set. Binding is performed at login or manually by an administrator who has the `PRIV_SYS_RES_CONFIG` privilege.

`cpu.pinned` Property Constraint

The `cpu-pinned` property indicates that a particular CPU should not be moved by DRP from the processor set in which it is located. You can set this `libpool` property to maximize cache utilization for a particular application that is executing within a processor set.

See Table 12-1 for more details about this property.

`pool.importance` Property Constraint

The `pool.importance` property describes the relative importance of a pool as defined by the administrator.

Configuration Objectives

Objectives are specified similarly to constraints. The full set of objectives is documented in Table 12-1.

There are two categories of objectives.

Workload dependent
: A workload-dependent objective is an objective that will vary according to the nature of the workload running on the system. An example is the `utilization` objective. The utilization figure for a resource set will vary according to the nature of the workload that is active in the set.

Workload independent — A workload-independent objective is an objective that does not vary according to the nature of the workload running on the system. An example is the CPU `locality` objective. The evaluated measure of locality for a resource set does not vary with the nature of the workload that is active in the set.

You can define three types of objectives.

Name	Valid Elements	Operators	Values
`wt-load`	`system`	N/A	N/A
`locality`	`pset`	N/A	`loose` | `tight` | `none`
`utilization`	`pset`	`< > ~`	`0–100%`

Objectives are stored in property strings in the `libpool` configuration. The property names are as follows:

- `system.poold.objectives`
- `pset.poold.objectives`

Objectives have the following syntax:

- `objectives = objective [; objective]*`
- `objective = [n:] keyword [op] [value]`

All objectives take an optional importance prefix. The importance acts as a multiplier for the objective and thus increases the significance of its contribution to the objective function evaluation. The range is from 0 to INT64_MAX (`9223372036854775807`). If not specified, the default importance value is `1`.

Some element types support more than one type of objective. An example is `pset`. You can specify multiple objective types for these elements. You can also specify multiple utilization objectives on a single `pset` element.

See "How to Define Configuration Objectives" on page 155 for usage examples.

`wt-load` Objective

The `wt-load` objective favors configurations that match resource allocations to resource utilizations. A resource set that uses more resources will be given more resources when this objective is active. `wt-load` means *weighted load*.

Use this objective when you are satisfied with the constraints you have established using the minimum and maximum properties, and you would like the daemon to manipulate resources freely within those constraints.

The `locality` Objective

The `locality` objective influences the impact that locality, as measured by locality group (`lgroup`) data, has upon the selected configuration. An alternate definition for locality is latency. An `lgroup` describes CPU and memory resources. The `lgroup` is used by the Oracle Solaris system to determine the distance between resources, using time as the measurement..

This objective can take one of the following three values:

`tight` — If set, configurations that maximize resource locality are favored.

`loose` — If set, configurations that minimize resource locality are favored.

`none` — If set, the favorableness of a configuration is not influenced by resource locality. This is the default value for the `locality` objective.

In general, the `locality` objective should be set to `tight`. However, to maximize memory bandwidth or to minimize the impact of DR operations on a resource set, you could set this objective to `loose` or keep it at the default setting of `none`.

`utilization` Objective

The `utilization` objective favors configurations that allocate resources to partitions that are not meeting the specified utilization objective.

This objective is specified by using operators and values. The operators are as follows:

`<` — The “less than” operator indicates that the specified value represents a maximum target value.

`>` — The “greater than” operator indicates that the specified value represents a minimum target value.

`~` — The “about” operator indicates that the specified value is a target value about which some fluctuation is acceptable.

A pset can only have one utilization objective set for each type of operator.

- If the `~` operator is set, then the `<` and `>` operators cannot be set.

- If the < and > operators are set, then the ~ operator cannot be set. Note that the settings of the < operator and the > operator cannot contradict each other.

You can set both a < and a > operator together to create a range. The values will be validated to make sure that they do not overlap.

Configuration Objectives Example

In the following example, `poold` is to assess these objectives for the pset:

- The `utilization` should be kept between 30 percent and 80 percent.
- The `locality` should be maximized for the processor set.
- The objectives should take the default importance of `1`.

EXAMPLE 12-1 `poold` Objectives Example

```
pset.poold.objectives "utilization > 30; utilization < 80; locality tight"
```

See "How to Define Configuration Objectives" on page 155 for additional usage examples.

`poold` Properties

There are four categories of properties:

- Configuration
- Constraint
- Objective
- Objective Parameter

TABLE 12-1 Defined Property Names

Property Name	Type	Category	Description
`system.poold.log-level`	string	Configuration	Logging level
`system.poold.log-location`	string	Configuration	Logging location
`system.poold.monitor-interval`	uint64	Configuration	Monitoring sample interval
`system.poold.history-file`	string	Configuration	Decision history location
`pset.max`	uint64	Constraint	Maximum number of CPUs for this processor set

Property Name	Type	Category	Description
pset.min	uint64	Constraint	Minimum number of CPUs for this processor set
cpu.pinned	bool	Constraint	CPUs pinned to this processor set
system.poold.objectives	string	Objective	Formatted string following poold's objective expression syntax
pset.poold.objectives	string	Objective	Formatted string following poold's expression syntax
pool.importance	int64	Objective parameter	User-assigned importance

poold Functionality That Can Be Configured

You can configure these aspects of the daemon's behavior.

- Monitoring interval
- Logging level
- Logging location

These options are specified in the pools configuration. You can also control the logging level from the command line by invoking `poold`.

poold Monitoring Interval

Use the property name `system.poold.monitor-interval` to specify a value in milliseconds.

poold Logging Information

Three categories of information are provided through logging. These categories are identified in the logs:

- Configuration
- Monitoring
- Optimization

Use the property name `system.poold.log-level` to specify the logging parameter. If this property is not specified, the default logging level is `NOTICE`. The parameter levels are hierarchical. Setting a log level of DEBUG will cause poold to log all defined messages. The INFO level provides a useful balance of information for most administrators.

At the command line, you can use the poold command with the -l option and a parameter to specify the level of logging information generated.

The following parameters are available:

- ALERT
- CRIT
- ERR
- WARNING
- NOTICE
- INFO
- DEBUG

The parameter levels map directly onto their syslog equivalents. See "Logging Location" on page 136 for more information about using syslog.

For more information about how to configure poold logging, see "How to Set the poold Logging Level" on page 157.

Configuration Information Logging

The following types of messages can be generated:

ALERT	Problems accessing the libpool configuration, or some other fundamental, unanticipated failure of the libpool facility. Causes the daemon to exit and requires immediate administrative attention.
CRIT	Problems due to unanticipated failures. Causes the daemon to exit and requires immediate administrative attention.
ERR	Problems with the user-specified parameters that control operation, such as unresolvable, conflicting utilization objectives for a resource set. Requires administrative intervention to correct the objectives. poold attempts to take remedial action by ignoring conflicting objectives, but some errors will cause the daemon to exit.
WARNING	Warnings related to the setting of configuration parameters that, while technically correct, might not be suitable for the given execution environment. An example is marking all CPU resources as pinned, which means that poold cannot move CPU resources between processor sets.

`DEBUG`	Messages containing the detailed information that is needed when debugging configuration processing. This information is not generally used by administrators.

Monitoring Information Logging

The following types of messages can be generated:

`CRIT`	Problems due to unanticipated monitoring failures. Causes the daemon to exit and requires immediate administrative attention.
`ERR`	Problems due to unanticipated monitoring error. Could require administrative intervention to correct.
`NOTICE`	Messages about resource control region transitions.
`INFO`	Messages about resource utilization statistics.
`DEBUG`	Messages containing the detailed information that is needed when debugging monitoring processing. This information is not generally used by administrators.

Optimization Information Logging

The following types of messages can be generated:

`WARNING`	Messages could be displayed regarding problems making optimal decisions. Examples could include resource sets that are too narrowly constrained by their minimum and maximum values or by the number of pinned components. Messages could be displayed about problems performing an optimal reallocation due to unforseen limitations. Examples could include removing the last processor from a processor set which contains a bound resource consumer.
`NOTICE`	Messages about usable configurations or configurations that will not be implemented due to overriding decision histories could be displayed.
`INFO`	Messages about alternate configurations considered could be displayed.
`DEBUG`	Messages containing the detailed information that is needed when debugging optimization processing. This information is not generally used by administrators.

Logging Location

The `system.poold.log-location` property is used to specify the location for `poold` logged output. You can specify a location of `SYSLOG` for `poold` output (see `syslog`(3C)).

If this property is not specified, the default location for `poold` logged output is `/var/log/pool/poold`.

When `poold` is invoked from the command line, this property is not used. Log entries are written to `stderr` on the invoking terminal.

Log Management With `logadm`

If `poold` is active, the `logadm.conf` file includes an entry to manage the default file `/var/log/pool/poold`. The entry is:

```
/var/log/pool/poold -N -s 512k
```

See the `logadm`(1M) and the `logadm.conf`(4) man pages.

How Dynamic Resource Allocation Works

This section explains the process and the factors that `poold` uses to dynamically allocate resources.

About Available Resources

Available resources are considered to be all of the resources that are available for use within the scope of the `poold` process. The scope of control is at most a single Oracle Solaris instance.

On a system that has zones enabled, the scope of an executing instance of `poold` is limited to the global zone.

Determining Available Resources

Resource pools encompass all of the system resources that are available for consumption by applications.

For a single executing Oracle Solaris instance, a resource of a single type, such as a CPU, must be allocated to a single partition. There can be one or more partitions for each type of resource. Each partition contains a unique set of resources.

For example, a machine with four CPUs and two processor sets can have the following setup:

pset 0: 0 1

pset 1: 2 3

where 0, 1, 2 and 3 after the colon represent CPU IDs. Note that the two processor sets account for all four CPUs.

The same machine cannot have the following setup:

pset 0: 0 1

pset 1: 1 2 3

It cannot have this setup because CPU 1 can appear in only one pset at a time.

Resources cannot be accessed from any partition other than the partition to which they belong.

To discover the available resources, `poold` interrogates the active pools configuration to find partitions. All resources within all partitions are summed to determine the total amount of available resources for each type of resource that is controlled.

This quantity of resources is the basic figure that `poold` uses in its operations. However, there are constraints upon this figure that limit the flexibility that `poold` has to make allocations. For information about available constraints, see "Configuration Constraints" on page 128.

Identifying a Resource Shortage

The control scope for `poold` is defined as the set of available resources for which `poold` has primary responsibility for effective partitioning and management. However, other mechanisms that are allowed to manipulate resources within this control scope can still affect a configuration. If a partition should move out of control while `poold` is active, `poold` tries to restore control through the judicious manipulation of available resources. If `poold` cannot locate additional resources within its scope, then the daemon logs information about the resource shortage.

Determining Resource Utilization

`poold` typically spends the greatest amount of time observing the usage of the resources within its scope of control. This monitoring is performed to verify that workload-dependent objectives are being met.

For example, for processor sets, all measurements are made across all of the processors in a set. The resource utilization shows the proportion of time that the resource is in use over the sample interval. Resource utilization is displayed as a percentage from 0 to 100.

Identifying Control Violations

The directives described in "Configuration Constraints and Objectives" on page 128 are used to detect the approaching failure of a system to meet its objectives. These objectives are directly related to workload.

A partition that is not meeting user-configured objectives is a control violation. The two types of control violations are synchronous and asynchronous.

- A synchronous violation of an objective is detected by the daemon in the course of its workload monitoring.
- An asynchronous violation of an objective occurs independently of monitoring action by the daemon.

The following events cause asynchronous objective violations:

- Resources are added to or removed from a control scope.
- The control scope is reconfigured.
- The `poold` resource controller is restarted.

The contributions of objectives that are not related to workload are assumed to remain constant between evaluations of the objective function. Objectives that are not related to workload are only reassessed when a reevaluation is triggered through one of the asynchronous violations.

Determining Appropriate Remedial Action

When the resource controller determines that a resource consumer is short of resources, the initial response is that increasing the resources will improve performance.

Alternative configurations that meet the objectives specified in the configuration for the scope of control are examined and evaluated.

This process is refined over time as the results of shifting resources are monitored and each resource partition is evaluated for responsiveness. The decision history is consulted to eliminate reconfigurations that did not show improvements in attaining the objective function in the past. Other information, such as process names and quantities, are used to further evaluate the relevance of the historical data.

If the daemon cannot take corrective action, the condition is logged. For more information, see "`poold` Logging Information" on page 133.

Using `poolstat` to Monitor the Pools Facility and Resource Utilization

The `poolstat` utility is used to monitor resource utilization when pools are enabled on your system. This utility iteratively examines all of the active pools on a system and reports statistics based on the selected output mode. The `poolstat` statistics enable you to determine which resource partitions are heavily utilized. You can analyze these statistics to make decisions about resource reallocation when the system is under pressure for resources.

The `poolstat` utility includes options that can be used to examine specific pools and report resource set-specific statistics.

If zones are implemented on your system and you use `poolstat` in a non-global zone, information about the resources associated with the zone's pool is displayed.

For more information about the `poolstat` utility, see the `poolstat`(1M) man page. For `poolstat` task and usage information, see "Using `poolstat` to Report Statistics for Pool-Related Resources" on page 162.

`poolstat` Output

In default output format, `poolstat` outputs a heading line and then displays a line for each pool. A pool line begins with the pool ID and the name of the pool, followed by a column of statistical data for the processor set attached to the pool. Resource sets attached to more than one pool are listed multiple times, once for each pool.

The column headings are as follows:

`id` Pool ID.

`pool` Pool name.

`rid` Resource set ID.

`rset` Resource set name.

`type` Resource set type.

`min` Minimum resource set size.

`max` Maximum resource set size.

`size` Current resource set size.

`used`	Measure of how much of the resource set is currently used. This usage is calculated as the percentage of utilization of the resource set multiplied by the size of the resource set. If a resource set has been reconfigured during the last sampling interval, this value might be not reported. An unreported value appears as a hyphen (`-`).
`load`	Absolute representation of the load that is put on the resource set. For more information about this property, see the `libpool`(3LIB) man page.

You can specify the following in `poolstat` output:

- The order of the columns
- The headings that appear

Tuning `poolstat` Operation Intervals

You can customize the operations performed by `poolstat`. You can set the sampling interval for the report and specify the number of times that statistics are repeated:

interval	Tune the intervals for the periodic operations performed by `poolstat`. All intervals are specified in seconds.
count	Specify the number of times that the statistics are repeated. By default, `poolstat` reports statistics only once.

If *interval* and *count* are not specified, statistics are reported once. If *interval* is specified and *count* is not specified, then statistics are reported indefinitely.

Commands Used With the Resource Pools Facility

The commands described in the following table provide the primary administrative interface to the pools facility. For information on using these commands on a system that has zones enabled, see "Resource Pools Used in Zones" on page 121.

Man Page Reference	Description
`pooladm`(1M)	Enables or disables the pools facility on your system. Activates a particular configuration or removes the current configuration and returns associated resources to their default status. If run without options, `pooladm` prints out the current dynamic pools configuration.

Man Page Reference	Description
poolbind(1M)	Enables the manual binding of projects, tasks, and processes to a resource pool.
poolcfg(1M)	Provides configuration operations on pools and sets. Configurations created using this tool are instantiated on a target host by using pooladm. If run with the info subcommand argument to the -c option, poolcfg displays information about the static configuration at /etc/pooladm.conf. If a file name argument is added, this command displays information about the static configuration held in the named file. For example, poolcfg -c info /tmp/newconfig displays information about the static configuration contained in the file /tmp/newconfig.
poold(1M)	The pools system daemon. The daemon uses system targets and observable statistics to preserve the system performance goals specified by the administrator. If unable to take corrective action when goals are not being met, poold logs the condition.
poolstat(1M)	Displays statistics for pool-related resources. Simplifies performance analysis and provides information that supports system administrators in resource partitioning and repartitioning tasks. Options are provided for examining specified pools and reporting resource set-specific statistics.

A library API is provided by libpool (see the libpool(3LIB) man page). The library can be used by programs to manipulate pool configurations.

♦ ♦ ♦ CHAPTER 13

Creating and Administering Resource Pools Tasks

This chapter describes how to set up and administer resource pools on your system.

For background information about resource pools, see Chapter 12, "About Resource Pools".

Administering Resource Pools Task Map

Task	Description	For Instructions
Enable or disable resource pools.	Activate or disable resource pools on your system.	"Enabling and Disabling the Pools Facility" on page 144
Enable or disable dynamic resource pools.	Activate or disable dynamic resource pools facilities on your system.	"Enabling and Disabling the Pools Facility" on page 144
Create a static resource pools configuration.	Create a static configuration file that matches the current dynamic configuration. For more information, see "Resource Pools Framework" on page 123.	"How to Create a Static Configuration" on page 149
Modify a resource pools configuration.	Revise a pools configuration on your system, for example, by creating additional pools.	"How to Modify a Configuration" on page 150
Associate a resource pool with a scheduling class.	Associate a pool with a scheduling class so that all processes bound to the pool use the specified scheduler.	"How to Associate a Pool With a Scheduling Class" on page 152
Set configuration constraints and define configuration objectives.	Specify objectives for `poold` to consider when taking corrective action. For more information on configuration objectives, see "`poold` Overview" on page 127.	"How to Set Configuration Constraints" on page 154 and "How to Define Configuration Objectives" on page 155
Set the logging level.	Specify the level of logging information that `poold` generates.	"How to Set the `poold` Logging Level" on page 157

Task	Description	For Instructions
Use a text file with the `poolcfg` command.	The `poolcfg` command can take input from a text file.	"How to Use Command Files With `poolcfg`" on page 157
Transfer resources in the kernel.	Transfer resources in the kernel. For example, transfer resources with specific IDs to a target set.	"Transferring Resources" on page 158
Activate a pools configuration.	Activate the configuration in the default configuration file.	"How to Activate a Pools Configuration" on page 159
Validate a pools configuration before you commit the configuration.	Validate a pools configuration to test what will happen when the validation occurs.	"How to Validate a Configuration Before Committing the Configuration" on page 159
Remove a pools configuration from your system.	All associated resources, such as processor sets, are returned to their default status.	"How to Remove a Pools Configuration" on page 160
Bind processes to a pool.	Manually associate a running process on your system with a resource pool.	"How to Bind Processes to a Pool" on page 160
Bind tasks or projects to a pool.	Associate tasks or projects with a resource pool.	"How to Bind Tasks or Projects to a Pool" on page 161
Bind new processes to a resource pool.	To automatically bind new processes in a project to a given pool, add an attribute to each entry in the `project` database.	"How to Set the `project.pool` Attribute for a Project" on page 161
Use `project` attributes to bind a process to a different pool.	Modify the pool binding for new processes that are started.	"How to Use `project` Attributes to Bind a Process to a Different Pool" on page 161
Use the `poolstat` utility to produce reports.	Produce multiple reports at specifed intervals.	"Producing Multiple Reports at Specific Intervals" on page 163
Report resource set statistics.	Use the `poolstat` utility to report statistics for a pset resource set.	"Reporting Resource Set Statistics" on page 163

Enabling and Disabling the Pools Facility

You can enable and disable the resource pools and dynamic resource pools services on your system by using the `svcadm` command described in the `svcadm`(1M) man page.

You can also use the `pooladm` command described in the `pooladm`(1M) man page to perform the following tasks:

- Enable the pools facility so that pools can be manipulated
- Disable the pools facility so that pools cannot be manipulated

---

Note - When a system is upgraded, if the resource pools framework is enabled and an `/etc/pooladm.conf` file exists, the pools service is enabled and the configuration contained in the file is applied to the system.

How to Enable the Resource Pools Service Using `svcadm`

1. **Become root or assume an equivalent role.**
2. **Enable the resource pools service.**

```
# svcadm enable system/pools:default
```

How to Disable the Resource Pools Service Using `svcadm`

1. **Become root or assume an equivalent role.**
2. **Disable the resource pools service.**

```
# svcadm disable system/pools:default
```

How to Enable the Dynamic Resource Pools Service Using `svcadm`

1. **Become root or assume an equivalent role.**
2. **Enable the dynamic resource pools service.**

```
# svcadm enable system/pools/dynamic:default
```

Example 13-1 Dependency of the Dynamic Resource Pools Service on the Resource Pools Service

This example shows that you must first enable resource pools if you want to run DRP.

There is a dependency between resource pools and dynamic resource pools. DRP is now a dependent service of resource pools. DRP can be independently enabled and disabled apart from resource pools.

The following display shows that both resource pools and dynamic resource pools are currently disabled:

```
# svcs "*pool*"
STATE          STIME    FMRI
disabled       2011     svc:/system/pools:default
disabled       2011     svc:/system/pools/dynamic:default
```

Enable dynamic resource pools:

```
# svcadm enable svc:/system/pools/dynamic:default
# svcs -a | grep pool
STATE          STIME    FMRI
disabled       2011     svc:/system/pools:default
offline        2011     svc:/system/pools/dynamic:default
```

Note that the DRP service is still offline.

Use the `-x` option of the `svcs` command to determine why the DRP service is offline:

```
# svcs -x "*pool*"
svc:/system/pools:default (resource pools framework)
 State: disabled since Sat Feb 12 02:36:15 2011
Reason: Disabled by an administrator.
   See: http://support.oracle.com/msg/SMF-8000-05
   See: libpool(3LIB)
   See: pooladm(1M)
   See: poolbind(1M)
   See: poolcfg(1M)
   See: poolstat(1M)
Impact: This service is not running.

svc:/system/pools/dynamic:default (dynamic resource pools)
 State: disabled since Sat Feb 12 02:36:16 2011
Reason: Disabled by an administrator.
   See: http://support.oracle.com/msg/SMF-8000-05
   See: poold(1M)
Impact: This service is not running.
```

Enable the resource pools service so that the DRP service can run:

```
# svcadm enable svc:/system/pools:default
```

When the `svcs "*pool*"` command is used, the system displays:

```
# svcs "*pool*"
STATE          STIME    FMRI
online         2011     svc:/system/pools/dynamic:default
online         2011     svc:/system/pools:default
```

Example 13-2 Effect on Dynamic Resource Pools When the Resource Pools Service Is Disabled

If both services are online and you disable the resource pools service:

```
# svcadm disable svc:/system/pools:default
```

When the `svcs "*pool*"` command is used, the system displays:

```
# svcs "*pool*"
STATE          STIME    FMRI
disabled       2011     svc:/system/pools:default
online         2011     svc:/system/pools/dynamic:default
```

However, the DRP service eventually moves to `offline` because the resource pools service has been disabled:

```
# svcs "*pool*"
STATE          STIME    FMRI
disabled       2011     svc:/system/pools:default
offline        2011     svc:/system/pools/dynamic:default
```

Determine why the DRP service is offline:

```
# svcs -x "*pool*"
svc:/system/pools:default (resource pools framework)
 State: disabled since Sat Feb 12 02:36:15 2011
Reason: Disabled by an administrator.
   See: http://support.oracle.com/msg/SMF-8000-05
   See: libpool(3LIB)
   See: pooladm(1M)
   See: poolbind(1M)
   See: poolcfg(1M)
   See: poolstat(1M)
Impact: 1 dependent service is not running.  (Use -v for list.)

svc:/system/pools/dynamic:default (dynamic resource pools)
 State: offline since Sat Feb 12 02:36:15 2011
Reason: Service svc:/system/pools:default is disabled.
   See: http://support.oracle.com/msg/SMF-8000-GE
   See: poold(1M)
   See: /var/svc/log/system-pools-dynamic:default.log
Impact: This service is not running.
```

Resource pools must be started for DRP to work. For example, resource pools could be started by using the `pooladm` command with the `-e` option:

```
# pooladm -e
```

Then the `svcs "*pool*"` command displays:

```
# svcs "*pool*"
STATE          STIME    FMRI
online         2011     svc:/system/pools:default
online         2011     svc:/system/pools/dynamic:default
```

How to Disable the Dynamic Resource Pools Service Using `svcadm`

1. **Become root or assume an equivalent role.**
2. **Disable the dynamic resource pools service.**

```
# svcadm disable system/pools/dynamic:default
```

How to Enable Resource Pools Using `pooladm`

1. **Become root or assume an equivalent role.**
2. **Enable the pools facility.**

```
# pooladm -e
```

How to Disable Resource Pools Using `pooladm`

1. **Become root or assume an equivalent role.**
2. **Disable the pools facility.**

```
# pooladm -d
```

Specific CPU Assignment

You can `assign` and `unassign` CPUs, cores, and sockets.

`cpus=`	List of cpus assigned to zone.
`cores=`	List of cores assigned to zone.
`sockets=`	List of sockets assigned to zone.

EXAMPLE 13-3 Assign Cores

Assign cores to pset *new*.

```
# poolcfg -dc 'assign to pset new (core 0 ; core 1)'
```

EXAMPLE 13-4 Update the Running Pool to Match the Persistent Static Configuration

Use the `zonecfg` command to assign cores. Use the `pooladm` command with the `-c` option to make the running pools match the static configuration.

```
# poolcfg -c 'assign to pset new (core 0 ; core 1)'
# pooladm -c
```

Configuring Pools

How to Create a Static Configuration

Use the `-s` option to `/usr/sbin/pooladm` to create a static configuration file that matches the current dynamic configuration, preserving changes across reboots. Unless a different file name is specified, the default location `/etc/pooladm.conf` is used.

Commit your configuration using the `pooladm` command with the `-c` option. Then, use the `pooladm` command with the `-s` option to update the static configuration to match the state of the dynamic configuration.

Note - The later functionality `pooladm -s` is preferred over the earlier functionality `poolcfg -c discover` for creating a new configuration that matches the dynamic configuration.

Before You Begin Enable pools on your system.

1. **Become root or assume an equivalent role.**
2. **Update the static configuration file to match the current dynamic configuration.**

   ```
   # pooladm -s
   ```

3. **View the contents of the configuration file in readable form.**

   Note that the configuration contains default elements created by the system.

   ```
   # poolcfg -c info
   system tester
           string  system.comment
           int     system.version 1
           boolean system.bind-default true
           int     system.poold.pid 177916

           pool pool_default
                   int     pool.sys_id 0
   ```

```
                boolean pool.active true
                boolean pool.default true
                int     pool.importance 1
                string  pool.comment
                pset    pset_default

        pset pset_default
                int     pset.sys_id -1
                boolean pset.default true
                uint    pset.min 1
                uint    pset.max 65536
                string  pset.units population
                uint    pset.load 10
                uint    pset.size 4
                string  pset.comment
                boolean testnullchanged true

                cpu
                        int     cpu.sys_id 3
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 2
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 1
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 0
                        string  cpu.comment
                        string  cpu.status on-line
```

4. **Commit the configuration at `/etc/pooladm.conf`.**

```
# pooladm -c
```

5. **(Optional) To copy the dynamic configuration to a static configuration file called `/tmp/backup`, type the following:**

```
# pooladm -s /tmp/backup
```

How to Modify a Configuration

To enhance your configuration, create a processor set named `pset_batch` and a pool named `pool_batch`. Then join the pool and the processor set with an association.

Note that you must quote subcommand arguments that contain white space.

1. **Become root or assume an equivalent role.**
2. **Create processor set `pset_batch`.**

   ```
   # poolcfg -c 'create pset pset_batch (uint pset.min = 2; uint pset.max = 10)'
   ```
3. **Create pool `pool_batch`.**

   ```
   # poolcfg -c 'create pool pool_batch'
   ```
4. **Join the pool and the processor set with an association.**

   ```
   # poolcfg -c 'associate pool pool_batch (pset pset_batch)'
   ```
5. **Display the edited configuration.**

   ```
   # poolcfg -c info
   system tester
           string  system.comment kernel state
           int     system.version 1
           boolean system.bind-default true
           int     system.poold.pid 177916

           pool pool_default
                   int     pool.sys_id 0
                   boolean pool.active true
                   boolean pool.default true
                   int     pool.importance 1
                   string  pool.comment
                   pset    pset_default

           pset pset_default
                   int     pset.sys_id -1
                   boolean pset.default true
                   uint    pset.min 1
                   uint    pset.max 65536
                   string  pset.units population
                   uint    pset.load 10
                   uint    pset.size 4
                   string  pset.comment
                   boolean testnullchanged true

                   cpu
                           int     cpu.sys_id 3
                           string  cpu.comment
                           string  cpu.status on-line

                   cpu
                           int     cpu.sys_id 2
                           string  cpu.comment
                           string  cpu.status on-line

                   cpu
                           int     cpu.sys_id 1
                           string  cpu.comment
   ```

```
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 0
                string  cpu.comment
                string  cpu.status on-line

pool pool_batch
        boolean pool.default false
        boolean pool.active true
        int pool.importance 1
        string pool.comment
        pset pset_batch

pset pset_batch
        int pset.sys_id -2
        string pset.units population
        boolean pset.default true
        uint pset.max 10
        uint pset.min 2
        string pset.comment
        boolean pset.escapable false
        uint pset.load 0
        uint pset.size 0

        cpu
                int     cpu.sys_id 5
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 4
                string  cpu.comment
                string  cpu.status on-line
```

6. **Commit the configuration at `/etc/pooladm.conf`.**

```
# pooladm -c
```

7. **(Optional) To copy the dynamic configuration to a static configuration file named `/tmp/backup`, type the following:**

```
# pooladm -s /tmp/backup
```

▼ How to Associate a Pool With a Scheduling Class

You can associate a pool with a scheduling class so that all processes bound to the pool use this scheduler. To do this, set the `pool.scheduler` property to the name of the scheduler. This example associates the pool `pool_batch` with the fair share scheduler (FSS).

1. **Become root or assume an equivalent role.**

2. **Modify pool `pool_batch` to be associated with the FSS.**

```
# poolcfg -c 'modify pool pool_batch (string pool.scheduler="FSS")'
```

3. **Display the edited configuration.**

```
# poolcfg -c info
system tester
        string  system.comment
        int     system.version 1
        boolean system.bind-default true
        int     system.poold.pid 177916

        pool pool_default
                int     pool.sys_id 0
                boolean pool.active true
                boolean pool.default true
                int     pool.importance 1
                string  pool.comment
                pset    pset_default

        pset pset_default
                int     pset.sys_id -1
                boolean pset.default true
                uint    pset.min 1
                uint    pset.max 65536
                string  pset.units population
                uint    pset.load 10
                uint    pset.size 4
                string  pset.comment
                boolean testnullchanged true

                cpu
                        int     cpu.sys_id 3
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 2
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 1
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 0
                        string  cpu.comment
                        string  cpu.status on-line

        pool pool_batch
                boolean pool.default false
                boolean pool.active true
                int pool.importance 1
                string pool.comment
```

```
                string pool.scheduler FSS
                pset batch

        pset pset_batch
                int pset.sys_id -2
                string pset.units population
                boolean pset.default true
                uint pset.max 10
                uint pset.min 2
                string pset.comment
                boolean pset.escapable false
                uint pset.load 0
                uint pset.size 0

                cpu
                        int     cpu.sys_id 5
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 4
                        string  cpu.comment
                        string  cpu.status on-line
```

4. **Commit the configuration at `/etc/pooladm.conf`:**

```
# pooladm -c
```

5. **(Optional) To copy the dynamic configuration to a static configuration file called `/tmp/backup`, type the following:**

```
# pooladm -s /tmp/backup
```

▼ How to Set Configuration Constraints

Constraints affect the range of possible configurations by eliminating some of the potential changes that could be made to a configuration. This procedure shows how to set the `cpu.pinned` property.

In the following examples, `cpuid` is an integer.

1. **Become root or assume an equivalent role.**

2. **Modify the `cpu.pinned` property in the static or dynamic configuration:**

   - **Modify the boot-time (static) configuration:**

   ```
   # poolcfg -c 'modify cpu <cpuid> (boolean cpu.pinned = true)'
   ```

- **Modify the running (dynamic) configuration without modifying the boot-time configuration:**

  ```
  # poolcfg -dc 'modify cpu <cpuid> (boolean cpu.pinned = true)'
  ```

How to Define Configuration Objectives

You can specify objectives for `poold` to consider when taking corrective action.

In the following procedure, the `wt-load` objective is being set so that `poold` tries to match resource allocation to resource utilization. The `locality` objective is disabled to assist in achieving this configuration goal.

1. **Become root or assume an equivalent role.**
2. **Modify system `tester` to favor the `wt-load` objective.**

   ```
   # poolcfg -c 'modify system tester (string system.poold.objectives="wt-load")'
   ```

3. **Disable the `locality` objective for the default processor set.**

   ```
   # poolcfg -c 'modify pset pset_default (string pset.poold.objectives="locality
    none")'     one line
   ```

4. **Disable the `locality` objective for the `pset_batch` processor set.**

   ```
   # poolcfg -c 'modify pset pset_batch (string pset.poold.objectives="locality
    none")'     one line
   ```

5. **Display the edited configuration.**

   ```
   # poolcfg -c info
   system tester
           string  system.comment
           int     system.version 1
           boolean system.bind-default true
           int     system.poold.pid 177916
           string  system.poold.objectives wt-load

           pool pool_default
                   int     pool.sys_id 0
                   boolean pool.active true
                   boolean pool.default true
                   int     pool.importance 1
                   string  pool.comment
                   pset    pset_default

           pset pset_default
   ```

```
                int     pset.sys_id -1
                boolean pset.default true
                uint    pset.min 1
                uint    pset.max 65536
                string  pset.units population
                uint    pset.load 10
                uint    pset.size 4
                string  pset.comment
                boolean testnullchanged true
                string  pset.poold.objectives locality none

                cpu
                        int     cpu.sys_id 3
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 2
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 1
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 0
                        string  cpu.comment
                        string  cpu.status on-line

        pool pool_batch
                boolean pool.default false
                boolean pool.active true
                int pool.importance 1
                string pool.comment
                string pool.scheduler FSS
                pset batch

        pset pset_batch
                int pset.sys_id -2
                string pset.units population
                boolean pset.default true
                uint pset.max 10
                uint pset.min 2
                string pset.comment
                boolean pset.escapable false
                uint pset.load 0
                uint pset.size 0
                string  pset.poold.objectives locality none

                cpu
                        int     cpu.sys_id 5
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 4
```

```
string   cpu.comment
string   cpu.status on-line
```

6. **Commit the configuration at `/etc/pooladm.conf`.**

```
# pooladm -c
```

7. **(Optional) To copy the dynamic configuration to a static configuration file called `/tmp/backup`, type the following:**

```
# pooladm -s /tmp/backup
```

How to Set the `poold` Logging Level

To specify the level of logging information that `poold` generates, set the `system.poold.log-level` property in the `poold` configuration. The `poold` configuration is held in the `libpool` configuration. For information, see “poold Logging Information” on page 133 and the `poolcfg`(1M) and `libpool`(3LIB) man pages.

You can also use the `poold` command at the command line to specify the level of logging information that `poold` generates.

1. **Become root or assume an equivalent role.**

2. **Set the logging level by using the `poold` command with the `-l` option and a parameter, for example, `INFO`.**

```
# /usr/lib/pool/poold -l INFO
```

For information about available parameters, see “poold Logging Information” on page 133. The default logging level is `NOTICE`.

How to Use Command Files With `poolcfg`

The `poolcfg` command with the `-f` option can take input from a text file that contains `poolcfg` subcommand arguments to the `-c` option. This method is appropriate when you want a set of operations to be performed. When processing multiple commands, the configuration is only updated if all of the commands succeed. For large or complex configurations, this technique can be more useful than per-subcommand invocations.

Note that in command files, the # character acts as a comment mark for the rest of the line.

1. **Create the input file `poolcmds.txt`.**

```
$ cat > poolcmds.txt
create system tester
create pset pset_batch (uint pset.min = 2; uint pset.max = 10)
create pool pool_batch
associate pool pool_batch (pset pset_batch)
```

2. **Become root or assume an equivalent role.**
3. **Execute the command:**

```
# /usr/sbin/poolcfg -f poolcmds.txt
```

Transferring Resources

Use the `transfer` subcommand argument to the `-c` option of `poolcfg` with the `-d` option to transfer resources in the kernel. The `-d` option specifies that the command operate directly on the kernel and not take input from a file.

The following procedure moves two CPUs from processor set `pset1` to processor set `pset2` in the kernel.

▼ How to Move CPUs Between Processor Sets

1. **Become root or assume an equivalent role.**
2. **Move two CPUs from `pset1` to `pset2`.**

   The `from` and `to` subclauses can be used in any order. Only one `to` and `from` subclause is supported per command.

```
# poolcfg -dc 'transfer 2 from pset pset1 to pset2'
```

Example 13-5 Alternative Method to Move CPUs Between Procesor Sets

If specific known IDs of a resource type are to be transferred, an alternative syntax is provided. For example, the following command assigns two CPUs with IDs `0` and `2` to the `pset_large` processor set:

```
# poolcfg -dc 'transfer to pset pset_large (cpu 0; cpu 2)'
```

Troubleshooting

If a transfer fails because there are not enough resources to match the request or because the specified IDs cannot be located, the system displays an error message.

Activating and Removing Pool Configurations

Use the `pooladm` command to make a particular pool configuration active or to remove the currently active pool configuration. See the `pooladm`(1M) man page for more information about this command.

▼ How to Activate a Pools Configuration

To activate the configuration in the default configuration file, `/etc/pooladm.conf`, invoke `pooladm` with the `-c` option, "commit configuration."

1. **Become root or assume an equivalent role.**
2. **Commit the configuration at `/etc/pooladm.conf`.**

```
# pooladm -c
```

3. **(Optional) Copy the dynamic configuration to a static configuration file, for example, `/tmp/backup`.**

```
# pooladm -s /tmp/backup
```

▼ How to Validate a Configuration Before Committing the Configuration

You can use the `-n` option with the `-c` option to test what will happen when the validation occurs. The configuration will not actually be committed.

The following command attempts to validate the configuration contained at `/home/admin/newconfig`. Any error conditions encountered are displayed, but the configuration itself is not modified.

1. **Become root or assume an equivalent role.**
2. **Test the validity of the configuration before committing it.**

```
# pooladm -n -c /home/admin/newconfig
```

▼ How to Remove a Pools Configuration

To remove the current active configuration and return all associated resources, such as processor sets, to their default status, use the -x option for "remove configuration."

1. **Become root or assume an equivalent role.**
2. **Remove the current active configuration.**

```
# pooladm -x
```

The -x option to pooladm removes all user-defined elements from the dynamic configuration. All resources revert to their default states, and all pool bindings are replaced with a binding to the default pool.

Mixing Scheduling Classes Within a Processor Set

You can safely mix processes in the TS and IA classes in the same processor set. Mixing other scheduling classes within one processor set can lead to unpredictable results. If the use of pooladm -x results in mixed scheduling classes within one processor set, use the priocntl command to move running processes into a different scheduling class. See "How to Manually Move Processes From the TS Class Into the FSS Class" on page 100. Also see the priocntl(1) man page.

Setting Pool Attributes and Binding to a Pool

You can set a project.pool attribute to associate a resource pool with a project.

You can bind a running process to a pool in two ways:

- You can use the poolbind command described in poolbind(1M) command to bind a specific process to a named resource pool.
- You can use the project.pool attribute in the project database to identify the pool binding for a new login session or a task that is launched through the newtask command. See the newtask(1), projmod(1M), and project(4) man pages.

▼ How to Bind Processes to a Pool

The following procedure uses poolbind with the -p option to manually bind a process (in this case, the current shell) to a pool named ohare.

1. **Become root or assume an equivalent role.**
2. **Manually bind a process to a pool:**

```
# poolbind -p ohare $$
```

3. **Verify the pool binding for the process by using `poolbind` with the `-q` option.**

```
$ poolbind -q $$
155509 ohare
```

The system displays the process ID and the pool binding.

How to Bind Tasks or Projects to a Pool

To bind tasks or projects to a pool, use the `poolbind` command with the `-i` option. The following example binds all processes in the `airmiles` project to the `laguardia` pool.

1. **Become root or assume an equivalent role.**
2. **Bind all processes in the `airmiles` project to the `laguardia` pool.**

```
# poolbind -i project -p laguardia airmiles
```

How to Set the `project.pool` Attribute for a Project

You can set the `project.pool` attribute to bind a project's processes to a resource pool.

1. **Become root or assume an equivalent role.**
2. **Add a `project.pool` attribute to each entry in the `project` database.**

```
# projmod -a -K project.pool=poolname project
```

How to Use `project` Attributes to Bind a Process to a Different Pool

Assume you have a configuration with two pools that are named `studio` and `backstage`. The `/etc/project` file has the following contents:

```
user.paul:1024::::project.pool=studio
```

```
user.george:1024::::project.pool=studio
user.ringo:1024::::project.pool=backstage
passes:1027::paul::project.pool=backstage
```

With this configuration, processes that are started by user `paul` are bound by default to the `studio` pool.

User `paul` can modify the pool binding for processes he starts. `paul` can use `newtask` to bind work to the `backstage` pool as well, by launching in the `passes` project.

1. **Launch a process in the `passes` project.**

```
$ newtask -l -p passes
```

2. **Use the `poolbind` command with the `-q` option to verify the pool binding for the process. Also use a double dollar sign (`$$`) to pass the process number of the parent shell to the command.**

```
$ poolbind -q $$
6384   pool backstage
```

The system displays the process ID and the pool binding.

Using `poolstat` to Report Statistics for Pool-Related Resources

The `poolstat` command is used to display statistics for pool-related resources. See "Using poolstat to Monitor the Pools Facility and Resource Utilization" on page 139 and the `poolstat`(1M) man page for more information.

The following subsections use examples to illustrate how to produce reports for specific purposes.

Displaying Default `poolstat` Output

Typing `poolstat` without arguments outputs a header line and a line of information for each pool. The information line shows the pool ID, the name of the pool, and resource statistics for the processor set attached to the pool.

```
machine% poolstat
                              pset
       id pool            size used load
        0 pool_default       4  3.6  6.2
        1 pool_sales         4  3.3  8.4
```

Producing Multiple Reports at Specific Intervals

The following command produces three reports at 5-second sampling intervals.

```
machine% poolstat 5 3
                               pset
 id pool                 size used load
 46 pool_sales              2  1.2  8.3
  0 pool_default            2  0.4  5.2
                              pset
 id pool                 size used load
 46 pool_sales              2  1.4  8.4
  0 pool_default            2  1.9  2.0
                              pset
 id pool                 size used load
 46 pool_sales              2  1.1  8.0
  0 pool_default            2  0.3  5.0
```

Reporting Resource Set Statistics

The following example uses the `poolstat` command with the `-r` option to report statistics for the processor set resource set. Note that the resource set `pset_default` is attached to more than one pool, so this processor set is listed once for each pool membership.

```
machine% poolstat -r pset
      id pool          type rid rset          min  max size used load
       0 pool_default  pset  -1 pset_default    1  65K    2  1.2  8.3
       6 pool_sales    pset   1 pset_sales      1  65K    2  1.2  8.3
       2 pool_other    pset  -1 pset_default    1  10K    2  0.4  5.2
```

♦ ♦ ♦ CHAPTER 14

Resource Management Configuration Example

This chapter reviews the resource management framework and describes a hypothetical server consolidation project.

The following topics are covered in this chapter:

- "Configuration to Be Consolidated" on page 165
- "Consolidation Configuration" on page 166
- "Creating the Configuration" on page 166
- "Viewing the Configuration" on page 167

Configuration to Be Consolidated

In this example, five applications are being consolidated onto a single system. The target applications have resource requirements that vary, different user populations, and different architectures. Currently, each application exists on a dedicated server that is designed to meet the requirements of the application. The applications and their characteristics are identified in the following table.

Application Description	Characteristics
Application server	Exhibits negative scalability beyond 2 CPUs
Database instance for application server	Heavy transaction processing
Application server in test and development environment	GUI-based, with untested code execution
Transaction processing server	Primary concern is response time
Standalone database instance	Processes a large number of transactions and serves multiple time zones

Consolidation Configuration

The following configuration is used to consolidate the applications onto a single system that has the resource pools and the dynamic resource pools facilities enabled.

- The application server has a two–CPU processor set.
- The database instance for the application server and the standalone database instance are consolidated onto a single processor set of at least four CPUs. The standalone database instance is guaranteed 75 percent of that resource.
- The test and development application server requires the IA scheduling class to ensure UI responsiveness. Memory limitations are imposed to lessen the effects of bad code builds.
- The transaction processing server is assigned a dedicated processor set of at least two CPUs, to minimize response latency.

This configuration covers known applications that are executing and consuming processor cycles in each resource set. Thus, constraints can be established that allow the processor resource to be transferred to sets where the resource is required.

- The `wt-load` objective is set to allow resource sets that are highly utilized to receive greater resource allocations than sets that have low utilization.
- The `locality` objective is set to `tight`, which is used to maximize processor locality.

An additional constraint to prevent utilization from exceeding 80 percent of any resource set is also applied. This constraint ensures that applications get access to the resources they require. Moreover, for the transaction processor set, the objective of maintaining utilization below 80 percent is twice as important as any other objectives that are specified. This importance will be defined in the configuration.

Creating the Configuration

Edit the `/etc/project` database file. Add entries to implement the required resource controls and to map users to resource pools, then view the file.

```
# cat /etc/project
.
.
.
user.app_server:2001:Production Application Server:::project.pool=appserver_pool
user.app_db:2002:App Server DB:::project.pool=db_pool;project.cpu-shares=(privileged,1,deny)
development:2003:Test and development::staff:project.pool=dev_pool;
process.max-address-space=(privileged,536870912,deny)      keep with previous line
user.tp_engine:2004:Transaction Engine:::project.pool=tp_pool
user.geo_db:2005:EDI DB:::project.pool=db_pool;project.cpu-shares=(privileged,3,deny)
.
.
.
```

Note - The development team has to execute tasks in the development project because access for this project is based on a user's group ID (GID).

Create an input file named `pool.host`, which will be used to configure the required resource pools. View the file.

```
# cat pool.host
create system host
create pset dev_pset (uint pset.min = 0; uint pset.max = 2)
create pset tp_pset (uint pset.min = 2; uint pset.max=8)
create pset db_pset (uint pset.min = 4; uint pset.max = 6)
create pset app_pset (uint pset.min = 1; uint pset.max = 2)
create pool dev_pool (string pool.scheduler="IA")
create pool appserver_pool (string pool.scheduler="TS")
create pool db_pool (string pool.scheduler="FSS")
create pool tp_pool (string pool.scheduler="TS")
associate pool dev_pool (pset dev_pset)
associate pool appserver_pool (pset app_pset)
associate pool db_pool (pset db_pset)
associate pool tp_pool (pset tp_pset)
modify system tester (string system.poold.objectives="wt-load")
modify pset dev_pset (string pset.poold.objectives="locality tight; utilization < 80")
modify pset tp_pset (string pset.poold.objectives="locality tight; 2: utilization < 80")
modify pset db_pset (string pset.poold.objectives="locality tight;utilization < 80")
modify pset app_pset (string pset.poold.objectives="locality tight; utilization < 80")
```

Update the configuration using the `pool.host` input file.

```
# poolcfg -f pool.host
```

Make the configuration active.

```
# pooladm -c
```

The framework is now functional on the system.

Enable DRP.

```
# svcadm enable pools/dynamic:default
```

Viewing the Configuration

To view the framework configuration, which also contains default elements created by the system, type:

```
# pooladm
system host
        string  system.comment
        int     system.version 1
        boolean system.bind-default true
        int     system.poold.pid 177916
```

```
string  system.poold.objectives wt-load

pool dev_pool
        int     pool.sys_id 125
        boolean pool.default false
        boolean pool.active true
        int     pool.importance 1
        string  pool.comment
        string  pool.scheduler IA
        pset    dev_pset

pool appserver_pool
        int     pool.sys_id 124
        boolean pool.default false
        boolean pool.active true
        int     pool.importance 1
        string  pool.comment
        string  pool.scheduler TS
        pset    app_pset

pool db_pool
        int     pool.sys_id 123
        boolean pool.default false
        boolean pool.active true
        int     pool.importance 1
        string  pool.comment
        string  pool.scheduler FSS
        pset    db_pset

pool tp_pool
        int     pool.sys_id 122
        boolean pool.default false
        boolean pool.active true
        int     pool.importance 1
        string  pool.comment
        string  pool.scheduler TS
        pset    tp_pset

pool pool_default
        int     pool.sys_id 0
        boolean pool.default true
        boolean pool.active true
        int     pool.importance 1
        string  pool.comment
        string  pool.scheduler TS
        pset    pset_default

pset dev_pset
        int     pset.sys_id 4
        string  pset.units population
        boolean pset.default false
        uint    pset.min 0
        uint    pset.max 2
        string  pset.comment
        boolean pset.escapable false
        uint    pset.load 0
        uint    pset.size 0
        string  pset.poold.objectives locality tight; utilization < 80
```

```
pset tp_pset
        int     pset.sys_id 3
        string  pset.units population
        boolean pset.default false
        uint    pset.min 2
        uint    pset.max 8
        string  pset.comment
        boolean pset.escapable false
        uint    pset.load 0
        uint    pset.size 0
        string  pset.poold.objectives locality tight; 2: utilization < 80

        cpu
                int     cpu.sys_id 1
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 2
                string  cpu.comment
                string  cpu.status on-line

pset db_pset
        int     pset.sys_id 2
        string  pset.units population
        boolean pset.default false
        uint    pset.min 4
        uint    pset.max 6
        string  pset.comment
        boolean pset.escapable false
        uint    pset.load 0
        uint    pset.size 0
        string  pset.poold.objectives locality tight; utilization < 80

        cpu
                int     cpu.sys_id 3
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 4
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 5
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 6
                string  cpu.comment
                string  cpu.status on-line
pset app_pset
        int     pset.sys_id 1
        string  pset.units population
        boolean pset.default false
```

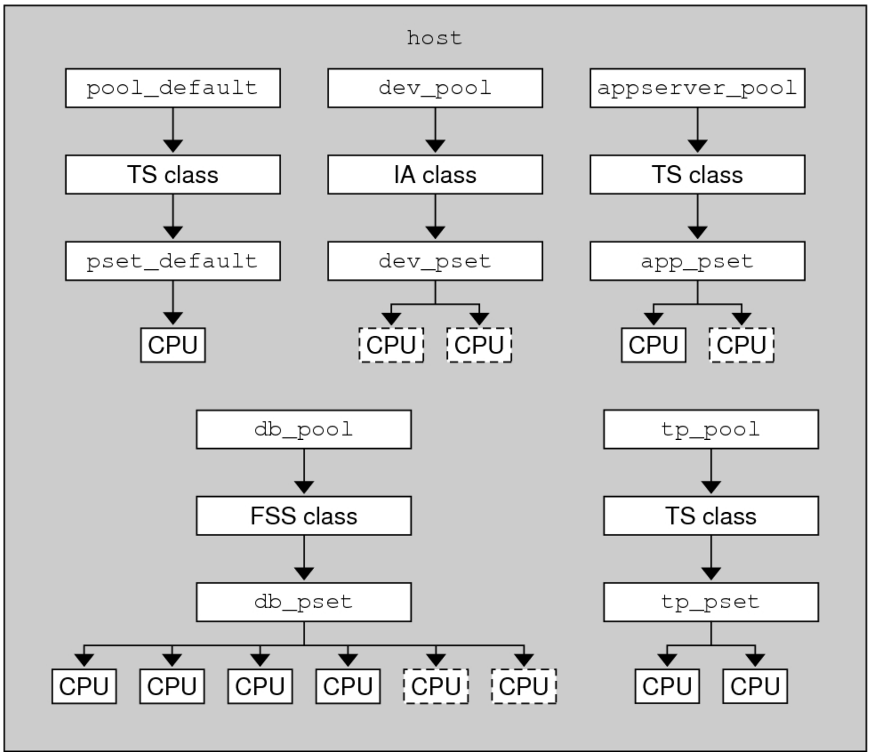
```
                uint    pset.min 1
                uint    pset.max 2
                string  pset.comment
                boolean pset.escapable false
                uint    pset.load 0
                uint    pset.size 0
                string  pset.poold.objectives locality tight; utilization < 80
                cpu
                        int     cpu.sys_id 7
                        string  cpu.comment
                        string  cpu.status on-line

        pset pset_default
                int     pset.sys_id -1
                string  pset.units population
                boolean pset.default true
                uint    pset.min 1
                uint    pset.max 4294967295
                string  pset.comment
                boolean pset.escapable false
                uint    pset.load 0
                uint    pset.size 0

                cpu
                        int     cpu.sys_id 0
                        string  cpu.comment
                        string  cpu.status on-line
```

A graphic representation of the framework follows.

FIGURE 14-1 Server Consolidation Configuration



Note - In the pool `db_pool`, the standalone database instance is guaranteed 75 percent of the CPU resource.

Index

Numbers and Symbols

/etc/project
  entry format, 24
  file, 23
/etc/user_attr file, 22
/var/adm/exacct directory, 46

A

acctadm command, 53
activating extended accounting, 51
administering resource pools, 140
attribute
  project.pool, 125

B

binding to resource pool, 160

C

changing resource controls temporarily, 73
commands
  extended accounting, 48
  fair share scheduler (FSS), 96
  projects and tasks, 28
  resource controls, 73
configuration
  rcapd, 105
configuring resource controls, 61
CPU share configuration, 92
creating resource pools, 126

D

default processor set, 120
default project, 22
default resource pool, 120
disabling dynamic resource pools, 144
disabling resource capping, 115
disabling resource pools, 144
displaying extended accounting status, 53
DRP, 121
dynamic pools configuration, 123
dynamic resource pools
  disabling, 144
  enabling, 144

E

enabling dynamic resource pools, 144
enabling resource capping, 114
enabling resource pools, 144
entry format
  /etc/project file, 24
exacct file, 44
extended accounting
  activating, 51
  chargeback, 44
  commands, 48
  file format, 44
  overview, 43
  SMF, 46
  status, displaying, 53

F

fair share scheduler (FSS), 88
  and processor sets, 93
  configuration, 99

project.cpu-shares, 88
share definition, 88
FSS *See* fair share scheduler (FSS)

I
implementing resource pools, 125
interprocess communication (IPC) *See* resource controls

L
libexacct library, 44

M
memory cap enforcement threshold, 106

P
PAM (pluggable authentication module)
identity management, 24
Perl interface, 48
pluggable authentication module *See* PAM
poold
asynchronous control violation, 138
configurable components, 133
constraints, 128
control scope, 137
cpu-pinned property, 129
description, 127
dynamic resource allocation, 121
logging information, 133
objectives, 129
synchronous control violation, 138
pools, 120
poolstat
description, 139
output format, 139
usage examples, 162
privilege levels, threshold values, 67
project
active state, 88
definition, 22
idle state, 88
with zero shares, 88
project 0, 92
project database, 23
project system *See* project 0
project.cpu-shares, 92
project.pool attribute, 125
putacct system call, 45

R
rcap.max-rss attribute, 104
rcapadm command, 105
rcapd
configuration, 105
sample intervals, 108
scan intervals, 108
rcapd daemon, 103
rcapstat command, 109
rctls, 59 *See* resource controls
removing resource pools, 160
resource cap, 103
resource capping
disabling, 115
enabling, 114
resource capping daemon, 103
resource controls
changing temporarily, 73
configuring, 61
definition, 59
global actions, 68
inf value, 71
interprocess communication (IPC), 60
list of, 62
local actions, 61, 69
overview, 59
temporarily updating, 73
threshold values, 61, 68, 69
resource limits, 60
resource management
constraints, 15
definition, 13
partitioning, 16
scheduling, 15
tasks, 27

resource pools, 120
  `/etc/pooladm.conf`, 123
  activating configuration, 159
  administering, 140
  binding to, 160
  configuration elements, 124
  creating, 126
  disabling, 144
  dynamic reconfiguration, 125
  enabling, 144
  implementing, 125
  properties, 124
  removing , 160
  removing configuration, 160
  static pools configuration, 123
rlimits *See* resource limits

S

scheduling classes, 95
server consolidation, 17
setting resource pool attributes, 160

T

tasks
  resource management, 27
temporarily updating resource controls, 73
threshold values, resource controls, 67