NAME

priocntl – display or set scheduling parameters of specified process(es)

SYNOPSIS

priocntl –l
priocntl –d [**–i** *idtype*] [*idlist*]
priocntl –s [**–c** *class*] [*class-specific options*] [**–i** *idtype*] [*idlist*]
priocntl –e [**–c** *class*] [*class-specific options*] *command* [*argument(s)*]

AVAILABILITY

SUNWcsu

DESCRIPTION

The **priocntl** command displays or sets scheduling parameters of the specified process(es). It can also be used to display the current configuration information for the system's process scheduler or execute a command with specified scheduling parameters.

Processes fall into distinct classes with a separate scheduling policy applied to each class. The process classes currently supported are the real-time class, time-sharing class, and the interactive class. The characteristics of these classes and the class-specific options they accept are described below in the **USAGE** section under the headings **Real-Time Class**, **Time-Sharing Class**, and **Inter-Active Class**. With appropriate permissions, the **priocntl** command can change the class and other scheduling parameters associated with a running process.

In the default configuration, a runnable real-time process runs before any other process. Therefore, inappropriate use of real-time processes can have a dramatic negative impact on system performance.

If an *idlist* is present it must appear last on the command line and the elements of the list must be separated by white space. If no *idlist* is present an *idtype* argument of **pid**, **ppid**, **pgid**, **sid**, **class**, **uid**, or **gid** specifies the process ID, parent process ID, process group ID, session ID, class, user ID, or group ID, respectively, of the **priocntl** command itself.

The command

priocntl –d [**–i** *idtype*] [*idlist*]

displays the class and class-specific scheduling parameters of the process(es) specified by *idtype* and *idlist*.

The command

priocntl –s [**–c** *class*] [*class-specific options*] [**–i** *idtype*] [*idlist*]

sets the class and class-specific parameters of the specified processes to the values given on the command line. The **–c** *class* option specifies the class to be set. (The valid *class* arguments are **RT** for real-time **TS** for time-sharing or **IA** for inter-active.)

The class-specific parameters to be set are specified by the class-specific options as explained under the appropriate heading below. If the **–c** *class* option is omitted, *idtype* and *idlist* must specify a set of processes which are all in the same class, otherwise an error results. If no class-specific options are specified the process's class-specific parameters are set to the default values for the class specified by **–c** *class* (or to the default parameter values for the process's current class if the **–c** *class* option is also omitted).

In order to change the scheduling parameters of a process using **priocntl** the real or effective user ID (respectively, groupID) of the user invoking **priocntl** must match the real or effective user ID (respectively, groupID) of the receiving process or the effective user ID of the user must be super-user. These are the minimum permission requirements enforced for all classes. An individual class may impose additional permissions requirements when setting processes to that class or when setting class-specific scheduling parameters.

When *idtype* and *idlist* specify a set of processes, **priocntl** acts on the processes in the set in an implementation-specific order. If **priocntl** encounters an error for one or more of the target processes, it may or may not continue through the set of processes, depending on the nature of the error.

If the error is related to permissions, **priocntl** prints an error message and then continue through the process set, resetting the parameters for all target processes for which the user has appropriate permissions. If **priocntl** encounters an error other than permissions, it does not continue through the process set but prints an error message and exits immediately.

A special **sys** scheduling class exists for the purpose of scheduling the execution of certain special system processes (such as the swapper process). It is not possible to change the class of any process to **sys**. In addition, any processes in the **sys** class that are included in the set of processes specified by *idtype* and *idlist* are disregarded by **priocntl**. For example, if *idtype* were **uid**, an *idlist* consisting of a zero would specify all processes with a UID of **0**, except processes in the **sys** class and (if changing the parameters using the −**s** option) the **init** process.

The **init** process (process ID **1**) is a special case. In order for the **priocntl** command to change the class or other scheduling parameters of the **init** process, *idtype* must be **pid** and *idlist* must be consist of only a **1**. The **init** process may be assigned to any class configured on the system, but the time-sharing class is almost always the appropriate choice. (Other choices may be highly undesirable; see the *System Administration Guide, Volume II* for more information.)

The command

priocntl −**e** [−**c** *class*] [*class-specific options*] *command* [*argument . . .*]

executes the specified command with the class and scheduling parameters specified on the command line (*arguments* are the arguments to the command). If the −**c** *class* option is omitted the command is run in the user's current class.

OPTIONS

−l Display a list of the classes currently configured in the system along with class-specific information about each class. The format of the class-specific information displayed is described under **USAGE**.

−d Display the scheduling parameters associated with a set of processes.

−s Set the scheduling parameters associated with a set of processes.

−e Execute a specified command with the class and scheduling parameters associated with a set of processes.

–i *idtype* This option together with the *idlist* arguments (if any), specify one or more processes to which the **priocntl** command is to apply. The interpretation of *idlist* depends on the value of *idtype*. The valid *idtype* arguments and corresponding interpretations of *idlist* are as follows:

–i pid *idlist* is a list of process IDs. The **priocntl** command applies to the specified processes.

–i ppid *idlist* is a list of parent process IDs. The **priocntl** command applies to all processes whose parent process ID is in the list.

–i pgid *idlist* is a list of process group IDs. The **priocntl** command applies to all processes in the specified process groups.

–i sid *idlist* is a list of session IDs. The **priocntl** command applies to all processes in the specified sessions.

–i class *idlist* consists of a single class name (**RT** for real-time or **TS** for time-sharing or **IA** for inter-active). The **priocntl** command applies to all processes in the specified class.

–i uid *idlist* is a list of user IDs. The **priocntl** command applies to all processes with an effective user ID equal to an ID from the list.

–i gid *idlist* is a list of group IDs. The **priocntl** command applies to all processes with an effective group ID equal to an ID from the list.

–i all The **priocntl** command applies to all existing processes. No *idlist* should be specified (if one is it is ignored). The permission restrictions described below still apply.

If the **–i** *idtype* option is omitted when using the **–d** or **–s** options the default *idtype* of **pid** is assumed.

–c *class* Specifies the *class* to be set. (The valid *class* arguments are **RT** for real-time or **TS** for time-sharing or **IA** for inter-active.) If the specified class is not already configured, it will automatically be configured.

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

–p *rtpri* Set the real-time priority of the specified process(es) to *rtpri*.

–t *tqntm* [**–r** *res*]
Set the time quantum of the specified process(es) to *tqntm*. You may optionally specify a resolution as explained below.

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

–m *tsuprilim*
Set the user priority limit of the specified process(es) to *tsuprilim*.

–p *tsupri* Set the user priority of the specified process(es) to *tsupri*.

The valid class-specific options for setting inter-active parameters are:

–m *iamode* Mark the specified process(es) as currently interactive, or not.

USAGE

Real-Time Class

The real-time class provides a fixed priority preemptive scheduling policy for those processes requiring fast and deterministic response and absolute user/application control of scheduling priorities. If the real-time class is configured in the system it should have exclusive control of the highest range of scheduling priorities on the system. This ensures that a runnable real-time process is given CPU service before any process belonging to any other class.

The real-time class has a range of real-time priority (*rtpri*) values that may be assigned to processes within the class. Real-time priorities range from 0 to x, where the value of x is configurable and can be displayed for a specific installation that has already configured a real-time scheduler, by using the command

priocntl −l

The real-time scheduling policy is a fixed priority policy. The scheduling priority of a real-time process never changes except as the result of an explicit request by the user/application to change the *rtpri* value of the process.

For processes in the real-time class, the *rtpri* value is, for all practical purposes, equivalent to the scheduling priority of the process. The *rtpri* value completely determines the scheduling priority of a real-time process relative to other processes within its class. Numerically higher *rtpri* values represent higher priorities. Since the real-time class controls the highest range of scheduling priorities in the system it is guaranteed that the runnable real-time process with the highest *rtpri* value is always selected to run before any other process in the system.

In addition to providing control over priority, **priocntl** provides for control over the length of the time quantum allotted to processes in the real-time class. The time quantum value specifies the maximum amount of time a process may run assuming that it does not complete or enter a resource or event wait state (**sleep**). Note that if another process becomes runnable at a higher priority, the currently running process may be preempted before receiving its full time quantum.

The command

priocntl −d [−i *idtype***] [***idlist***]**

displays the real-time priority and time quantum (in millisecond resolution) for each real-time process in the set specified by *idtype* and *idlist*.

Any combination of the −**p** and −**t** options may be used with **priocntl −s** or **priocntl −e** for the real-time class. If an option is omitted and the process is currently real-time, the associated parameter is unaffected. If an option is omitted when changing the class of a process to real-time from some other class, the associated parameter is set to a default value. The default value for *rtpri* is **0** and the default for time quantum is dependent on the value of *rtpri* and on the system configuration; see **rt_dptbl**(4).

When using the **–t** *tqntm* option you may optionally specify a resolution using the **–r** *res* option. (If no resolution is specified, millisecond resolution is assumed.) If *res* is specified it must be a positive integer between 1 and 1,000,000,000 inclusive and the resolution used is the reciprocal of *res* in seconds. For example, specifying **–t 10** **–r 100** would set the resolution to hundredths of a second and the resulting time quantum length would be 10/100 seconds (one tenth of a second). Although very fine (nanosecond) resolution may be specified, the time quantum length is rounded up by the system to the next integral multiple of the system clock's resolution. Requests for time quantums of zero or quantums greater than the (typically very large) implementation-specific maximum quantum result in an error.

In order to change the class of a process to real-time (from any other class) the user invoking **priocntl** must have super-user privilege. In order to change the *rtpri* value or time quantum of a real-time process the user invoking **priocntl** must either be super-user, or must currently be in the real-time class (shell running as a real-time process) with a real or effective user ID matching the real or effective user ID of the target process.

The real-time priority and time quantum are inherited across the **fork**(2) and **exec**(2) system calls.

Time-Sharing Class

The time-sharing scheduling policy provides for a fair and effective allocation of the CPU resource among processes with varying CPU consumption characteristics. The objectives of the time-sharing policy are to provide good response time to interactive processes and good throughput to CPU-bound jobs while providing a degree of user/application control over scheduling.

The time-sharing class has a range of time-sharing user priority (*tsupri*) values that may be assigned to processes within the class. User priorities range from –*x* to +*x*, where the value of *x* is configurable. The range for a specific installation can be displayed by using the command

priocntl –l

The purpose of the user priority is to provide some degree of user/application control over the scheduling of processes in the time-sharing class. Raising or lowering the *tsupri* value of a process in the time-sharing class raises or lowers the scheduling priority of the process. It is not guaranteed, however, that a time-sharing process with a higher *tsupri* value will run before one with a lower *tsupri* value. This is because the *tsupri* value is just one factor used to determine the scheduling priority of a time-sharing process. The system may dynamically adjust the internal scheduling priority of a time-sharing process based on other factors such as recent CPU usage.

In addition to the system-wide limits on user priority (displayed with **priocntl –l**), there is a per process user priority limit (*tsuprilim*), which specifies the maximum *tsupri* value that may be set for a given process.

The command

priocntl –d [–i *idtype*] [*idlist*] displays the user priority and user priority limit for each time-sharing process in the set specified by *idtype* and *idlist*.

Any time-sharing process may lower its own *tsuprilim* (or that of another process with the same user ID). Only a time-sharing process with super-user privilege may raise a *tsuprilim*. When changing the class of a process to time-sharing from some other class, super-user privilege is required in order to set the initial *tsuprilim* to a value greater than zero.

Any time-sharing process may set its own *tsupri* (or that of another process with the same user ID) to any value less than or equal to the process's *tsuprilim*. Attempts to set the *tsupri* above the *tsuprilim* (and/or set the *tsuprilim* below the *tsupri*) result in the *tsupri* being set equal to the *tsuprilim*.

Any combination of the −**m** and −**p** options may be used with **priocntl** −**s** or **priocntl** −**e** for the time-sharing class. If an option is omitted and the process is currently time-sharing the associated parameter is normally unaffected. The exception is when the −**p** option is omitted and −**m** is used to set a *tsuprilim* below the current *tsupri*. In this case the *tsupri* is set equal to the *tsuprilim* which is being set. If an option is omitted when changing the class of a process to time-sharing from some other class, the associated parameter is set to a default value. The default value for *tsuprilim* is **0** and the default for *tsupri* is to set it equal to the *tsuprilim* value which is being set.

The time-sharing user priority and user priority limit are inherited across the **fork**(2) and **exec**(2) system calls.

Inter-Active Class

The inter-active scheduling policy provides for a fair and effective allocation of the CPU resource among processes with varying CPU consumption characteristics while providing good responsiveness for user interaction. The objectives of the inter-active policy are to provide good response time to interactive processes and good throughput to CPU-bound jobs. Only the super user has access to the inter-active class, the user has no control over scheduling policys.

EXAMPLES

Real-Time Class examples follow:

example% priocntl −s −c RT −t 1 −r 10 −i *idtype idlist*

The above example sets the class of any non-real-time processes selected by *idtype* and *idlist* to real-time and sets their real-time priority to the default value of **0**. The real-time priorities of any processes currently in the real-time class are unaffected. The time quantums of all of the specified processes are set to 1/10 seconds.

example% priocntl −e −c RT −p 15 −t 20 *command*

This example executes *command* in the real-time class with a real-time priority of 15 and a time quantum of 20 milliseconds.

Time-Sharing Class examples follow:

example% priocntl −s −c TS −i *idtype idlist*

The above example sets the class of any non-time-sharing processes selected by *idtype* and *idlist* to time-sharing and sets both their user priority limit and user priority to **0**. Processes already in the time-sharing class are unaffected.

This example executes *command* with the arguments *arguments* in the time-sharing class with a user priority limit of **0** and a user priority of −**15**.

example% priocntl −**e** −**c TS** −**m 0** −**p** −**15** *command* **[***arguments***]**

SEE ALSO

nice(1), **ps**(1), **exec**(2), **fork**(2), **priocntl**(2), **rt_dptbl**(4)

DIAGNOSTICS

priocntl prints the following error messages:

Process(es) not found
None of the specified processes exists.

Specified processes from different classes
The −**s** option is being used to set parameters, the −**c** *class* option is not present, and processes from more than one class are specified.

Invalid option or argument
An unrecognized or invalid option or option argument is used.