NAME

rt_dptbl – real-time dispatcher parameter table

DESCRIPTION

The process scheduler (or dispatcher) is the portion of the kernel that controls allocation of the CPU to processes. The scheduler supports the notion of scheduling classes where each class defines a scheduling policy, used to schedule processes within that class. Associated with each scheduling class is a set of priority queues on which ready to run processes are linked. These priority queues are mapped by the system configuration into a set of global scheduling priorities which are available to processes within the class. (The dispatcher always selects for execution the process with the highest global scheduling priority in the system.) The priority queues associated with a given class are viewed by that class as a contiguous set of priority levels numbered from 0 (lowest priority) to *n* (highest priority—a configuration dependent value). The set of global scheduling priorities that the queues for a given class are mapped into might not start at zero and might not be contiguous (depending on the configuration).

The real-time class maintains an in-core table, with an entry for each priority level, giving the properties of that level. This table is called the real-time dispatcher parameter table (**rt_dptbl**). The **rt_dptbl** consists of an array (**config_rt_dptbl[]**) of parameter structures (**struct rtdpent_t**), one for each of the *n* priority levels. The structure are accessed via a pointer, (**rt_dptbl**), to the array. The properties of a given priority level *i* are specified by the *i*th parameter structure in this array (**rt_dptbl[***i***]**).

A parameter structure consists of the following members. These are also described in the **/usr/include/sys/rt.h** header file.

rt_globpri The global scheduling priority associated with this priority level. The **rt_globpri** values cannot be changed with **dispadmin**(1M).

rt_quantum The length of the time quantum allocated to processes at this level in ticks (Hz). The time quantum value is only a default or starting value for processes at a particular level as the time quantum of a real-time process can be changed by the user with the **priocntl** command or the **priocntl** system call.

An administrator can affect the behavior of the real-time portion of the scheduler by reconfiguring the **rt_dptbl**. There are two methods available for doing this: reconfigure with a loadable module at boot-time or by using **dispadmin**(1M) at run-time.

RT_DPTBL LOADABLE MODULE

The **rt_dptbl** can be reconfigured with a loadable module which contains a new real time dispatch table. The module containing the dispatch table is separate from the RT loadable module which contains the rest of the real time software. This is the only method that can be used to change the number of real time priority levels or the set of global scheduling priorities used by the real time class. The relevant procedure and source code is described in the **REPLACING THE RT_DPTBL LOADABLE MODULE** section.

DISPADMIN CONFIGURATION FILE

The **rt_quantum** values in the **rt_dptbl** can be examined and modified on a running system using the **dispadmin**(1M) command. Invoking **dispadmin** for the real-time class allows the administrator to retrieve the current **rt_dptbl** configuration from the kernel's in-core table, or overwrite the in-core table with values from a configuration file. The configuration file used for input to **dispadmin** must conform to the specific format described below.

Blank lines are ignored and any part of a line to the right of a # symbol is treated as a comment. The first non-blank, non-comment line must indicate the resolution to be used for interpreting the time quantum values. The resolution is specified as

RES=*res*

where *res* is a positive integer between 1 and 1,000,000,000 inclusive and the resolution used is the reciprocal of *res* in seconds. (For example, **RES=1000** specifies millisecond resolution.) Although very fine (nanosecond) resolution may be specified, the time quantum lengths are rounded up to the next integral multiple of the system clock's resolution.

The remaining lines in the file are used to specify the **rt_quantum** values for each of the real-time priority levels. The first line specifies the quantum for real-time level 0, the second line specifies the quantum for real-time level 1, etc. There must be exactly one line for each configured real-time priority level. Each **rt_quantum** entry must be either a positive integer specifying the desired time quantum (in the resolution given by *res*), or the value -2 indicating an infinite time quantum for that level.

EXAMPLES

The following excerpt from a **dispadmin** configuration file illustrates the format. Note that for each line specifying a time quantum there is a comment indicating the corresponding priority level. These level numbers indicate priority within the real-time class, and the mapping between these real-time priorities and the corresponding global scheduling priorities is determined by the configuration specified in the **RT_DPTBL** loadable module. The level numbers are strictly for the convenience of the administrator reading the file and, as with any comment, they are ignored by **dispadmin** on input. **dispadmin** assumes that the lines in the file are ordered by consecutive, increasing priority level (from 0 to the maximum configured real-time priority). The level numbers in the comments should normally agree with this ordering; if for some reason they don't, however, **dispadmin** is unaffected.

```
# Real-Time Dispatcher Configuration File
RES=1000
#   TIME QUANTUM     PRIORITY
    # (rt_quantum)      LEVEL
          100              #  0
          100              #  1
          100              #  2
          100              #  3
          100              #  4
          100              #  5
           90              #  6
           90              #  7
            .              .  .
            .              .  .
            .              .  .
           10              #  58
           10              #  59
```

REPLACING THE RT_DPTBL LOADABLE MODULE

In order to change the size of the real time dispatch table, the loadable module which contains the dispatch table information will have to be built. It is recommended that you save the existing module before using the following procedure.

1. Place the dispatch table code shown below in a file called **rt_dptbl.c** An example of an **rt_dptbl.c** file follows.
2. Compile the code using the given compilation and link lines supplied.

   cc –c –0 –D_KERNEL rt_dptbl.c
   ld –r –o RT_DPTBL rt_dptbl.o

3. Copy the current dispatch table in **/usr/kernel/sched** to **RT_DPTBL.bak**.
4. Replace the current **RT_DPTBL** in **/usr/kernel/sched**.
5. You will have to make changes in the **/etc/system** file to reflect the changes to the sizes of the tables. See **system**(4). The **rt_maxpri** variable may need changing. The syntax for setting this is:

   set RT:rt_maxpri=(class-specific value for maximum real-time priority)

6. Reboot the system to use the new dispatch table.

NOTE: Great care should be used in replacing the dispatch table using this method. If you don't get it right, the system may not behave properly. The following is an example of a **rt_dptbl.c** file used for building the new **rt_dptbl**.

```
        /* BEGIN rt_dptbl.c */

#include <sys/proc.h>
#include <sys/priocntl.h>
#include <sys/class.h>
#include <sys/disp.h>
#include <sys/rt.h>
#include <sys/rtpriocntl.h>

/*
 * This is the loadable module wrapper.
 */
#include <sys/modctl.h>

extern struct mod_ops mod_miscops;

/*
 * Module linkage information for the kernel.
 */

static struct modlmisc modlmisc = {
        &mod_miscops, "realtime dispatch table"
};

static struct modlinkage modlinkage = {
        MODREV_1, &modlmisc, 0
};

_init()
{
        return (mod_install(&modlinkage));
}

_info (struct modinfo *modinfop)
{
        return (mod_info(&modlinkage, modinfop));
}


rtdpent_t   config_rt_dptbl[] = {
/*  prilevel    Time quantum */
    100,            100,
    101,            100,
    102,            100,
    103,            100,
```

104,	**100,**
105,	**100,**
106,	**100,**
107,	**100,**
108,	**100,**
109,	**100,**
110,	**80,**
111,	**80,**
112,	**80,**
113,	**80,**
114,	**80,**
115,	**80,**
116,	**80,**
117,	**80,**
118,	**80,**
119,	**80,**
120,	**60,**
121,	**60,**
122,	**60,**
123,	**60,**
124,	**60,**
125,	**60,**
126,	**60,**
127,	**60,**
128,	**60,**
129,	**60,**
130,	**40,**
131,	**40,**
132,	**40,**
133,	**40,**
134,	**40,**
135,	**40,**
136,	**40,**
137,	**40,**
138,	**40,**
139,	**40,**
140,	**20,**
141,	**20,**
142,	**20,**
143,	**20,**
144,	**20,**
145,	**20,**
146,	**20,**
147,	**20,**
148,	**20,**

```
    149,          20,
    150,          10,
    151,          10,
    152,          10,
    153,          10,
    154,          10,
    155,          10,
    156,          10,
    157,          10,
    158,          10,
    159,          10,
};

/*
 * Return the address of config_rt_dptbl
 */
rtdpent_t *
rt_getdptbl()
{
        return (config_rt_dptbl);
}
```

FILES **<sys/rt.h>**

SEE ALSO **priocntl**(1), **dispadmin**(1M), **priocntl**(2), **system**(4)

System Administration Guide, Volume II
System Interfaces Guide