Name ts_dptbl – time-sharing 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).

Processes in the time-sharing class which are running in user mode (or in kernel mode before going to sleep) are scheduled according to the parameters in a time-sharing dispatcher parameter table (`ts_dptbl`). Processes in the inter-active scheduling class are also scheduled according to the parameters in the time-sharing dispatcher parameter table. (Time-sharing processes and inter-active processes running in kernel mode after sleeping are run within a special range of priorities reserved for such processes and are not affected by the parameters in the `ts_dptbl` until they return to user mode.) The `ts_dptbl` consists of an array (`config_ts_dptbl[]`) of parameter structures (`struct tsdpent_t`), one for each of the *n* priority levels used by time-sharing processes and inter-active processes in user mode. The structures are accessed via a pointer, (`ts_dptbl`), to the array. The properties of a given priority level *i* are specified by the *i*th parameter structure in this array (`ts_dptbl[` *i*`]`).

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

`ts_globpri` The global scheduling priority associated with this priority level. The mapping between time-sharing priority levels and global scheduling priorities is determined at boot time by the system configuration. `ts_globpri` is the only member of the `ts_dptbl` which cannot be changed with `dispadmin(1M)`.

`ts_quantum` The length of the time quantum allocated to processes at this level in ticks (`hz`).

In the high resolution clock mode (`hires_tick` set to `1`), the value of `hz` is set to `1000`. Increase quantums to maintain the same absolute time quantums.

`ts_tqexp` Priority level of the new queue on which to place a process running at the current level if it exceeds its time quantum. Normally this field links to a lower priority time-sharing level that has a larger quantum.

`ts_slpret` Priority level of the new queue on which to place a process, that was previously in user mode at this level, when it returns to user mode after sleeping. Normally this field links to a higher priority level that has a smaller quantum.

`ts_maxwait` A per process counter, `ts_dispwait` is initialized to zero each time a time-sharing or inter-active process is placed back on the dispatcher queue after its time quantum has expired or when it is awakened (`ts_dispwait` is not reset to zero when a process is preempted by a higher priority process). This counter is incremented once per second for each process on a dispatcher or sleep queue. If a process' `ts_dispwait` value exceeds the `ts_maxwait` value for its level, the process' priority is changed to that indicated by `ts_lwait`. The purpose of this field is to prevent starvation.

`ts_lwait` Move a process to this new priority level if `ts_dispwait` is greater than `ts_maxwait`.

An administrator can affect the behavior of the time-sharing portion of the scheduler by reconfiguring the `ts_dptbl`. Since processes in the time-sharing and inter-active scheduling classes share the same dispatch parameter table (`ts_dptbl`), changes to this table will affect both scheduling classes. There are two methods available for doing this: reconfigure with a loadable module at boot-time or by using `dispadmin(1M)` at run-time.

ts_dptbl Loadable Module

The `ts_dptbl` can be reconfigured with a loadable module which contains a new time sharing dispatch table. The module containing the dispatch table is separate from the TS loadable module which contains the rest of the time-sharing and inter-active software. This is the only method that can be used to change the number of time-sharing priority levels or the set of global scheduling priorities used by the time-sharing and inter-active classes. The relevant procedure and source code is described in the `REPLACING THE TS_DPTBL LOADABLE MODULE` section.

dispadmin Configuration File

With the exception of `ts_globpri` all of the members of the `ts_dptbl` can be examined and modified on a running system using the `dispadmin(1M)` command. Invoking `dispadmin` for the time-sharing or inter-active class allows the administrator to retrieve the current `ts_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 `ts_quantum` 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 parameter values for each of the time-sharing priority levels. The first line specifies the parameters for time-sharing level 0, the second line specifies the parameters for time-sharing level 1, etc. There must be exactly one line for each configured time-sharing priority level.

Examples

EXAMPLE 1 A Sample From a Configuration File

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

```
# Time-Sharing Dispatcher Configuration File RES=1000


# ts_quantum  ts_tqexp  ts_slpret  ts_maxwait  ts_lwait  PRIORITY
#                                                        LEVEL
500           0         10         5           10        # 0
500           0         11         5           11        # 1
500           1         12         5           12        # 2
500           1         13         5           13        # 3
500           2         14         5           14        # 4
500           2         15         5           15        # 5
450           3         16         5           16        # 6
450           3         17         5           17        # 7
.             .         .          .           .         . .
.             .         .          .           .         . .
.             .         .          .           .         . .
50            48        59         5           59        # 58
50            49        59         5           59        # 59
```

EXAMPLE 2 Replacing The ts_dptbl Loadable Module

In order to change the size of the time sharing 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.

EXAMPLE 2 Replacing The ts_dptbl Loadable Module *(Continued)*

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

   ```
   cc -c -0 -D_KERNEL
   ts_dptbl.c
   ld -r -o TS_DPTBL ts_dptbl.o
   ```

3. Copy the current dispatch table in `/kernel/sched` to `TS_DPTBL.bak`.
4. Replace the current `TS_DPTBL` in `/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 two variables affected are `ts_maxupri` and `ts_maxkmdpri`. The syntax for setting these is as follows:

   ```
   set TS:ts_maxupri=(value for max time-sharing user priority)
   set TS:ts_maxkmdpri=(number of kernel mode priorities - 1)
   ```

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

Great care should be used in replacing the dispatch table using this method. If you do not get it right, panics may result, thus making the system unusable.

The following is an example of a `ts_dptbl.c` file used for building the new `ts_dptbl`.

```
/* BEGIN ts_dptbl.c */
#include <sys/proc.h>
#include <sys/priocntl.h>
#include <sys/class.h>
#include <sys/disp.h>
#include <sys/ts.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, "Time sharing dispatch table"
};
static struct modlinkage modlinkage = {
    MODREV_1, &modlmisc, 0
};
_init()
```

EXAMPLE 2 Replacing The ts_dptbl Loadable Module *(Continued)*

```
{
    return (mod_install(&modlinkage));
}
_info(modinfop)
    struct modinfo *modinfop;
{
    return (mod_info(&modlinkage, modinfop));
}
/*
 * array of global priorities used by ts procs sleeping or
 * running in kernel mode after sleep. Must have at least
 * 40 values.
 */
pri_t config_ts_kmdpris[] = {
         60,61,62,63,64,65,66,67,68,69,
         70,71,72,73,74,75,76,77,78,79,
         80,81,82,83,84,85,86,87,88,89,
         90,91,92,93,94,95,96,97,98,99,
};
tsdpent_t    config_ts_dptbl[] = {

/*  glbpri  qntm  tqexp  slprt  mxwt  lwt  */

    0,      100,  0,     10,    5,    10,
    1,      100,  0,     11,    5,    11,
    2,      100,  1,     12,    5,    12,
    3,      100,  1,     13,    5,    13,
    4,      100,  2,     14,    5,    14
    5,      100,  2,     15,    5,    15,
    6,      100,  3,     16,    5,    16,
    7,      100,  3,     17,    5,    17,
    8,      100,  4,     18,    5,    18,
    9,      100,  4,     19,    5,    19,
    10,     80,   5,     20,    5,    20,
    11,     80,   5,     21,    5,    21,
    12,     80,   6,     22,    5,    22,
    13,     80,   6,     23,    5,    23,
    14,     80,   7,     24,    5,    24,
    15,     80,   7,     25,    5,    25,
    16,     80,   8,     26,    5,    26,
    17,     80,   8,     27,    5,    27,
    18,     80,   9,     28,    5,    28,
    19,     80,   9,     29,    5,    29,
    20,     60,   10,    30,    5,    30,
    21,     60,   11,    31,    5,    31,
```

EXAMPLE 2 Replacing The ts_dptbl Loadable Module *(Continued)*

```
    22,     60,   12,    32,    5,    33,
    24,     60,   14,    34,    5,    34,
    25,     60,   15,    35,    5,    35,
    26,     60,   16,    36,    5,    36,
    27,     60,   17,    37,    5,    37,
    28,     60,   18,    38,    5,    38,
    29,     60,   19,    39,    5,    39,
    30,     40,   20,    40,    5,    40,
    31,     40,   21,    41,    5,    41,
    32,     40,   22,    42,    5,    42,
    33,     40,   23,    43,    5,    43,
    34,     40,   24,    44,    5,    44,
    35,     40,   25,    45,    5,    45,
    36,     40,   26,    46,    5,    46,
    37,     40,   27,    47,    5,    47,
    38,     40,   28,    48,    5,    48,
    39,     40,   29,    49,    5,    49,
    40,     20,   30,    50,    5,    50,
    41,     20,   31,    50,    5,    50,
    42,     20,   32,    51,    5,    51,
    43,     20,   33,    51,    5,    51,
    44,     20,   34,    52,    5,    52,
    45,     20,   35,    52,    5,    52,
    46,     20,   36,    53,    5,    53,
    47,     20    37,    53,    5,    53,
    48,     20,   38,    54,    5,    54,
    49,     20,   39,    54,    5,    54,
    50,     10,   40,    55,    5,    55,
    51,     10,   41,    55,    5,    55,
    52,     10,   42,    56,    5,    56,
    53,     10,   43,    56,    5,    56,
    54,     10,   44,    57,    5,    57,
    55,     10,   45,    57,    5,    57,
    56,     10,   46,    58,    5,    58,
    57,     10,   47,    58,    5,    58,
    58,     10,   48,    59,    5,    59,
    59,     10,   49,    59,    5,    59,

};

short config_ts_maxumdpri = sizeof (config_ts_dptbl)/16 - 1;
/*
 * Return the address of config_ts_dptbl
 */
tsdpent_t *
```

EXAMPLE 2 Replacing The ts_dptbl Loadable Module *(Continued)*

```
ts_getdptbl()
{
      return (config_ts_dptbl);
}

/*
 * Return the address of config_ts_kmdpris
 */
 int *
 ts_getkmdpris()
{
      return (config_ts_kmdpris);
}

/*
 * Return the address of ts_maxumdpri
 */
short
ts_getmaxumdpri()
{
       return (config_ts_maxumdpri);
}

/* END ts_dptbl.c */
```

See Also priocntl(1), dispadmin(1M), priocntl(2), system(4)

System Administration Guide: Basic Administration

Programming Interfaces Guide

Notes `dispadmin` does some limited sanity checking on the values supplied in the configuration file. The sanity checking is intended to ensure that the new `ts_dptbl` values do not cause the system to panic. The sanity checking does not attempt to analyze the effect that the new values will have on the performance of the system. Unusual `ts_dptbl` configurations may have a dramatic negative impact on the performance of the system.

No sanity checking is done on the `ts_dptbl` values specified in the `TS_DPTBL` loadable module. Specifying an inconsistent or nonsensical `ts_dptbl` configuration through the `TS_DPTBL` loadable module could cause serious performance problems and/or cause the system to panic.