Changing Process Priority in Linux

ELECTGON
www.electgon.com
ma_ext@gmx.net

20.06.2018

Contents

1 Introduction

In real time systems, it might be necessary to run an application with high priority other than rest of applications. Or at least you may prefer to control priority of all the running applications. Importance of this control appears more if you are using real times like Linux with tens of processes that you don't have control over. It is possible to build your real time application in Linux with high priority using some known methods and that is what is going to be discussed in the following lines.

The rule is: In Linux, available values for priorities are 1 (highest priority) to 139 (lowest priority). 1 to 99 are given for real time processes. 100 to 139 are given for non real time (conventional processes) [1].

For conventional processes, in some of its running time its priority is constant. In this case we can say it has static priority. By time, this process has to be allocated in CPU (so its priority should be higher than running process) or de-allocated (so its priority should be less than a waiting process). That means conventional processes have static priority and dynamic priority.

For real time processes, priority is not changing. That is because real time process should have always the right to run in the CPU unless there is other process with higher priority is coming (this is not exactly true as you can change the priority or preempt the process from the CPU under some circumstances that are not declared here).

Now, If we know how to change these priority numbers, we will be able then to change running application priority.

To change these priorities, there are mainly two mechanisms, first one is to change the priority value directly. Second one is to change the priority value indirectly using what is called niceness.

Niceness is another metric for process priority. In Linux, it can take values form -20 (highest priority) to 19 (lowest priority). Some context states the relation between Priority and Niceness as:

$Priority = 20 + Niceness$

So for the range -20 to 19 it is mapped to 100 to 139 which is allowed priority numbers for conventional processes. For some extend, it is right to say that changing priorities by changing niceness is allowed only for conventional processes (non real time). In order to change priorities of real time process, you have to use direct methods.

2 Realtime Process - Direct Change

2.1 Using sched.h Library

For that purpose we can use Linux library called sched.h to change priority value directly. This library has some functions that can be used to change priority of the running process.

First, functions used in these libraries are using opposite scale for definition of highest priority and lowest priority. In Linux 1 is (highest priority), 139 (lowest priority). But in this library 99 (highest priority), 1 is (lowest priority). It means also that these functions

can change only real time processes priorities. If you want to manipulate for conventional process, you are allowed only only to set its priority to 0.

So to avoid any confusion while working with this library:

- This library defines 0 priority for conventional processes (non real time processes).
- This library defines Priority 1 for lowest priority real time process.
- This library defines Priority 99 for highest priority real time process.

Second, this library has one structure called sched_param. In the current implementation, the structure contains only one field, sched_priority.

```
struct sched_param {
               int sched_priority;
          };
```

So in order to change the priority, we have to define a variable of that type. This variable will be used within functions defined in this library.

Third, There is what is called scheduling policy. Currently, Linux supports the following ”conventional” (i.e., non-real-time) scheduling policies

SCHED_OTHER the standard round-robin time-sharing policy;

SCHED_BATCH for ”batch” style execution of processes; and

SCHED_IDLE for running very low priority background jobs.

For ”real-time” policies the following are supported:

SCHED_FIFO a first-in, first-out policy; and

SCHED_RR a round-robin policy.

Now we can discuss important functions that can be used in manipulating process priority. As mentioned before, Linux detects priorities of running processes but giving it priority number. For previously mentioned scheduling policies, we need to know what is the priority number for each of these policies. For that purpose, we can use functions

```
int sched_get_priority_max(int policy);
int sched_get_priority_min(int policy);
```

To change priority of a running process, the following function can be used

```
 int sched_setscheduler(pid_t pid, int policy, const struct sched_param *param)
    ;
```

where pid is the running process thread id, policy is the scheduling policy, param is variable of type structure sched_param. To get the priority of the current running process, we use value 0 in the field of pid.

To get priority of a running process you can use the following function

```
int sched_getscheduler(pid_t pid);
```

The following examples shows how to use these functions

```
#include<stdio.h>
#include<time.h>
#include <sched.h>

typedef struct sched_param SC_param;
SC_param schedparam;

void main(void)  {
 int x=0;
 int prio=0;
 int set_ret;

 printf("min priority for FIFO is %d\n", sched_get_priority_min(SCHED_FIFO));
 printf("max priority for FIFO is %d\n", sched_get_priority_max(SCHED_FIFO));

 printf("min priority for RR is %d\n", sched_get_priority_min(SCHED_RR));
 printf("max priority for RR is %d\n", sched_get_priority_max(SCHED_RR));

 printf("min priority for OTHER is %d\n", sched_get_priority_min(SCHED_OTHER));
 printf("max priority for OTHER is %d\n", sched_get_priority_max(SCHED_OTHER));

  while (1)  {
   printf("Counting %d Process Priority is %d set_ret is%d\n", x, prio, set_ret
       );
   x=x+1;
   schedparam.sched_priority = x;
   sleep(1);
   prio=sched_getscheduler(0);
   set_ret=sched_setscheduler(0, SCHED_FIFO, &schedparam);
   }
 }
```

2.2 Using Shell Command

There is a shell command that can be used to change real time processes. It is chrt. This command can be used as follows

The default behavior is to run a new command:

```
chrt priority command [arguments]
```

where priority is the value that you want to run your command with. Here also priority takes value from 1 to 99. As an example:

```
$chrt 20 my_code
```

this will run my_code with the real time priority 20. You can also retrieve the real-time attributes of an existing process:

```
$chrt -p pid
```

Or set new priority for an existing process using:

```
$chrt -r -p priority pid
```

3 Conventional Process- Indirect Change

3.1 using resource.h library

Another and similar function that can be used also, is found in resource.h library. It has functions like getpriority and setpriority. These functions can change only Niceness values, which means it can change conventional processes priorities. The following functions can be used.

```
int getpriority(int which, id_t who);
int setpriority(int which, id_t who, int prio);
```

The value "which" is one of **PRIO_PROCESS**, **PRIO_PGRP**, or **PRIO_USER**, and "who" is interpreted relative to "which" (a process identifier for **PRIO_PROCESS**, process group identifier for **PRIO_PGRP**, and a user ID for **PRIO_USER**). A zero value for "who" denotes (respectively) the calling process, the process group of the calling process, or the real user ID of the calling process. The "prio" argument is the value of the needed priority in the range -20 to 19. The default priority is 0. The following examples shows how to use these functions.

```
#include<stdio.h>
#include<time.h>
#include<sys/resource.h>
void main(void)  {
 int prio=0;
 int set_ret;

 prio=getpriority(PRIO_PROCESS, 0);
 set_ret=setpriority(PRIO_PROCESS, 0, 5);
}
```

3.2 Using Shell Command

There is shell command called nice which can be used to change the priority of the process. You can use it as follows

```
$nice -NI [COMMAND]
```

where NI is the Niceness value that you want to assign. Niceness value can take values from -20 to 19. COMMAND is the function or task that you will start running from shell . This means nice will run COMMAND with Niceness value.

You can also change Niceness of running process if you know its PID using the following command.

```
$nice -NI PID
```

If you provided any value out of the range (-20 to 19) it will be rounded to its limit. i.e. the following will assign Niceness value -20 to the process of PID 2423.

```
$nice --35 2423
```

4 Track Process Priority

There is light application that can be used to see details about running processes and its memory consumption along with its priorities. It is called htop. If it is not installed in your linux machine you can get it by typing

```
$sudo apt-get install htop
```

Bibliography

[1] Understanding the Linux Kernel, (O'Reilly), 2002.

[2] Linux Manual Page: http://man7.org