

**Bahir dar university,**

**Bahir dar instute of technology(BIT)**

**faculty of computing**

**Department of software engineering**

**Operating system and system programming**

**individual assignment 2**

**Title : I/O priority system call**

**Name Id**

**Jemberu kassie…………………1306925**

**Submmited to: Mr wendimu B**

**submission date :17/11/14 E.C**

**Acknowledgement**

*First I would like to express my special thanks of gratitude to my instructor, Wendmu B, who gave me the golden opportunity to do this wonderful assignment on the topic of system call , which also helped me in doing a lot of related task about how to system call implement in C language and It helps me to know about so many new things . Secondly i would also like to thank my class and dormitory friends who helped me a lot in finalizing this assignment within the limited time frame. Thank you again!!*

Introduction

This document is about system call ,specifically about the I/O scheduling on disk and it describe how this system call works ,the functionality of each parameter and also what is and what are scheduling class. I/O scheduling is the method that computer operating systems use to decide in which order I/O operations will be submitted to storage area. I/O scheduling is sometimes called disk scheduling.

And also this document describe the implementation of this system call.

I/O scheduling system call

Intioprio-set(intwhich,intwho,intprio)

In this system call we have two functions ,ioprio-set() and ioprio-get() that responsible for get and set scheduling class and priority of a process.

What / Why / How, this system call?

This system call is the linux system that used when the anopration of input output on the disk.

Why it need?

Because it's crucial to give priorityfor a particular process to get disk at a time, so this system call is important.

How it work?

**The ioprio\_get() and ioprio\_set() system calls get and set the I/O scheduling class and priority of one or more threads.**

The whichandwho arguments identify the thread(s) on which the

system calls operate. The which argument determines how who is interpreted, and it has one of the following values:

1. OPRIO\_WHO\_PROCESS

In this case who is a process ID or thread ID and identifying a single

process or thread. If who is 0, then operate on the calling thread or process.

2.IOPRIO\_WHO\_PGRP

In this case who is a process group ID identifying all the members of a

process group. If who is 0, then operate on the process

group of which the caller is a member.

3.IOPRIO\_WHO\_USER

In this case who is a user ID identifying all of the processes thathave a matching real UID.

The ioprioargument given to ioprio\_set() is a bit mask thatspecifies both the scheduling class and the priority to beassigned to the target process(es).

If <which>is specified as *IOPRIO\_WHO\_PGRP* or *IOPRIO\_WHO\_USER* when calling ioprio\_get(), and more than one process matches who, the returned priority will be the highest one found among all .

#### NoteIf the processes which have the same group id that specified by whoargument and if they are in different schedule class, first ioprio argument specify the schedule class which has higher priority than the other and if there are processes that are in that high priority schedule class, again ioprio argument specify a process with higher priority.

What is scheduling class

I/O nice, which is priority value ranges from -20 to 19 where -20 is highest priority,0 default and 19 is lowest priority, levels similar to those of CPU scheduling. These nice levels are grouped into three scheduling classes, each one containing one or more priority levels:

1. IOPRIO\_CLASS\_RT

This is the real-time I/O class. This scheduling class is

given higher priority than any other class: processes from

this class are given first access to the disk every time.

Thus, this I/O class needs to be used with some care: one

I/O real-time process can starve the entire system.

Within the real-time class, there are 8 levels of class

data (priority) that determine exactly how much time this

process needs the disk for on each service. The highest

real-time priority level is 0; the lowest is 7.

2.IOPRIO\_CLASS\_BE

This is the best-effort scheduling class, which is thedefault for any process that hasn't set a specific I/Opriority. The class data (priority) determines how much I/O bandwidth the process will get. Best-effort priority levels are analogous to CPU nice values. The priority level determines a priority relative to other processes in the best-effort scheduling class. Priority levels range from 0 (highest) to 7lowest).

2.IOPRIO\_CLASS\_IDLE

This is the idle scheduling class. Processes running at this level get I/O time only when no one else needs thedisk. The idle class has no class data. Attention is required when assigning this priority class to a process since it may become starved if higher priority processeare constantly accessing the disk.

int ioprio\_set(int which, int who, int ioprio)

**Flages**

CLONE\_IO – child process shares I/O context with parent

which – flag specifying target of who

who – id determined by which flag

ioprio – bit mask specifying scheduling class and priority to assign to who process

Returns zero on success.

### which

* IOPRIO\_WHO\_PROCESS – who is process or thread id, or 0 to use calling thread
* IOPRIO\_WHO\_PGRP – who – is a process id identifying all members of a process group, or 0 to operate on process group where calling process is member
* IOPRIO\_WHO\_USER – who is UID identifying all processes that have a matching real UID

### ioprio\_get

Get I/O scheduling class and priority.

int ioprio\_get(int which, int who)

* which – flag specifying target of who
* who – id determined by which flag

Return ioprio value of process with highest I/O priority of matching processes.

Implementation

/\*this sample source code tht describe how to implement

I/O priority in linux or in c program

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <getopt.h>

#include <unistd.h>

#include <sys/ptrace.h>

#include <asm/unistd.h>

externintsys\_ioprio\_set(int, int, int);

externintsys\_ioprio\_get(int, int);

static inline intioprio\_set(int which, int who, intioprio)

{

returnsyscall(\_\_NR\_ioprio\_set, which, who, ioprio);

}

static inline intioprio\_get(int which, int who)

{

returnsyscall(\_\_NR\_ioprio\_get, which, who);

}

enum {

IOPRIO\_CLASS\_NONE,

IOPRIO\_CLASS\_RT,

IOPRIO\_CLASS\_BE,

IOPRIO\_CLASS\_IDLE,

};

enum {

IOPRIO\_WHO\_PROCESS = 1,

IOPRIO\_WHO\_PGRP,

IOPRIO\_WHO\_USER,

};

#define IOPRIO\_CLASS\_SHIFT 13

const char \*to\_prio[] = { "none", "realtime", "best-effort", "idle", };

int main(intargc, char \*argv[])

{

intioprio = 4, set = 0, ioprio\_class = IOPRIO\_CLASS\_BE;

int c, pid = 0;

while ((c = getopt(argc, argv, "+n:c:p:")) != EOF) {

switch (c) {

case 'n':

ioprio = strtol(optarg, NULL, 10);

set = 1;

break;

case 'c':

ioprio\_class = strtol(optarg, NULL, 10);

set = 1;

break;

case 'p':

pid = strtol(optarg, NULL, 10);

break;

}

}

switch (ioprio\_class) {

case IOPRIO\_CLASS\_NONE:

ioprio\_class = IOPRIO\_CLASS\_BE;

break;

case IOPRIO\_CLASS\_RT:

case IOPRIO\_CLASS\_BE:

break;

case IOPRIO\_CLASS\_IDLE:

ioprio = 7;

break;

default:

printf("bad prio class %d\n", ioprio\_class);

return 1;

}

if (!set) {

if (!pid&&argv[optind])

pid = strtol(argv[optind], NULL, 10);

ioprio = ioprio\_get(IOPRIO\_WHO\_PROCESS, pid);

printf("pid=%d, %d\n", pid, ioprio);

if (ioprio == -1)

perror("ioprio\_get");

else {

ioprio\_class = ioprio>> IOPRIO\_CLASS\_SHIFT;

ioprio = ioprio& 0xff;

printf("%s: prio %d\n", to\_prio[ioprio\_class], ioprio);

}

} else {

if (ioprio\_set(IOPRIO\_WHO\_PROCESS, pid, ioprio | ioprio\_class<< IOPRIO\_CLASS\_SHIFT) == -1) {

perror("ioprio\_set");

return 1;

}

if (argv[optind])

execvp(argv[optind], &argv[optind]);

}

return 0;

}

The output of the above code is:

pid=0, 4

none: prio

it means the program return the value when the ioprio parameter in ioprio-set() function that specify the scheduling class and return priority number, which is 4, in to process that has pid (process id) hold by who parameter, which is 0 therefore this process first get disk space or cpu time .

**References**

["BFQ I/O Scheduler Lands Along With New Kyber Scheduler - Phoronix"](https://www.phoronix.com/scan.php?page=news_item&px=Linux-4.12-BFQ-Kyber). *[www.phoronix.com](http://www.phoronix.com)*

**Linux/UNIX system programming training courses**