Modifying CFS Scheduler

Samiksha Modi
Roll No: 2019331

Description of Code and How it Works

I defined my system call in rtnice.c present in the directory rtnice in the extracted folder of the kernel v5.9.1

The function is defined in the following way : SYSCALL_DEFINEX(), where X is the number of parameters that are passed to the system call. For me it was SYSCALL_DEFINE2(rtnice, int,mypid, int, mytime). Here x=2 since I have 2 parameters ie: pid (mypid) and soft realtime guarantee (mytime). The first parameter in the definition is compulsory and is the name of the system call implemented. I then multiply mytime with 10^11 to get a significantly large value for soft_rtnice. If mytime is a negative value, I return the error no and handle the error.

To get struct pid I use find_get_pid(). If it returns NULL then no process with the given input pid exists, so I return the error no and handle the error.
If it exists, then I use pid_task() to get the tast_struct of the given pid. The task_struct contains sched_entity (se) of the process in which I assign the soft realtime guarantee called new.

To add the data variable soft_rtnice in the task_struct,I created a new data variable in the file sched.h, and initialized it to 0 in core.c .

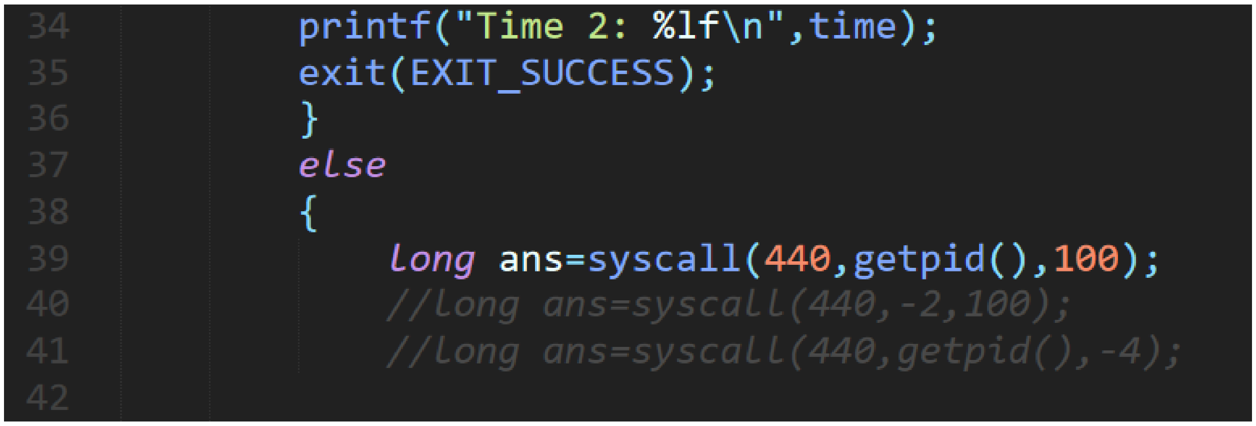
To guarantee that the process with the given pid runs for the given time, we need to first modify the scheduler in such a way that the scheduler always picks a task with lower soft_rtnice, and if two processes have the same soft_rtnice, then the priority should be given to the task with lower vruntime. I did this in entity_before() present in fair.c

I also deduct the amount of time the program has run from the soft_rtnice. For this I decrement the delta_exec value from the soft_rtnice, if the value of delta_exec is greater than the amount of soft_rtnice, then I set the soft_rtnice to 0.

User Inputs

- PID (datatpye- int)
- Soft realtime guarantee (datatype-int)

The inputs have been hardcoded into test.c
The system can be tested by changing the arguments we pass in syscall.



```
            printf("Time 2: %lf\n",time);
            exit(EXIT_SUCCESS);
            }
            else
            {
                long ans=syscall(440,getpid(),100);
                //long ans=syscall(440,-2,100);
                //long ans=syscall(440,getpid(),-4);

```

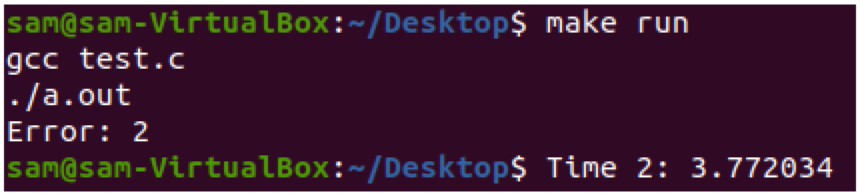
The arguments in syscall are syscall number (440), pid, and soft real-time guarantee.

Expected output

I am printing the execution time taken by both the tasks. I fork() my process and in one process I print the execution time with soft realtime guarantees. And in the other process I print the execution time without the soft realtime guarantees. Values of both the execution time will be different. Incase of an invalid input, it will show the appropriate error.

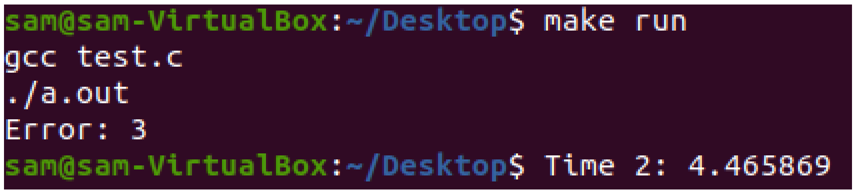
Errors Handled

1. Invalid pid input
   It will show Error: 2. It means that no process with the given pid exists.



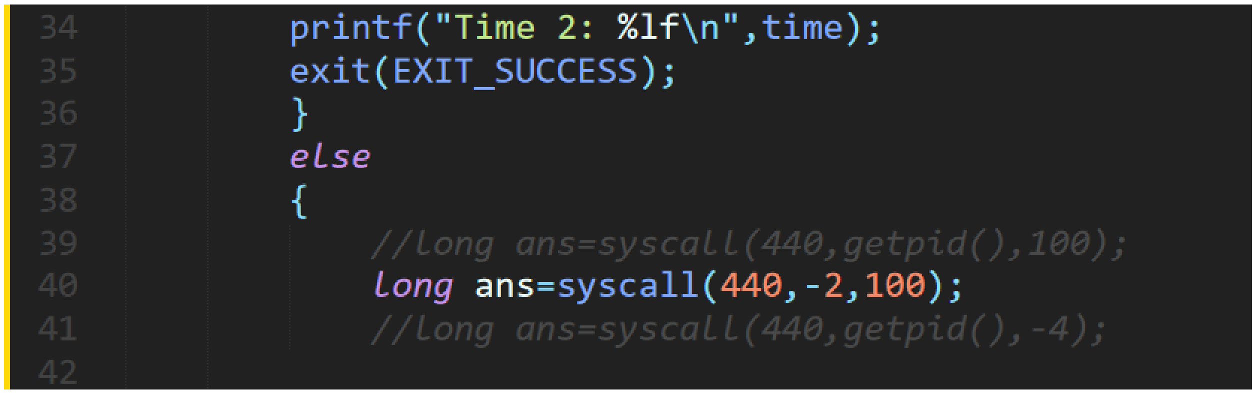
```
sam@sam-VirtualBox:~/Desktop$ make run
gcc test.c
./a.out
Error: 2
sam@sam-VirtualBox:~/Desktop$ Time 2: 3.772034
```

2. Invalid time input
   It will show Error: 3. If the user inputs a negative value for the soft realtime guarantee, it gives an error.



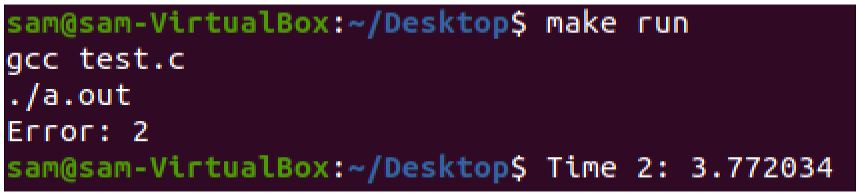
```
sam@sam-VirtualBox:~/Desktop$ make run
gcc test.c
./a.out
Error: 3
sam@sam-VirtualBox:~/Desktop$ Time 2: 4.465869
```

Sample 1 (Invalid pid input)



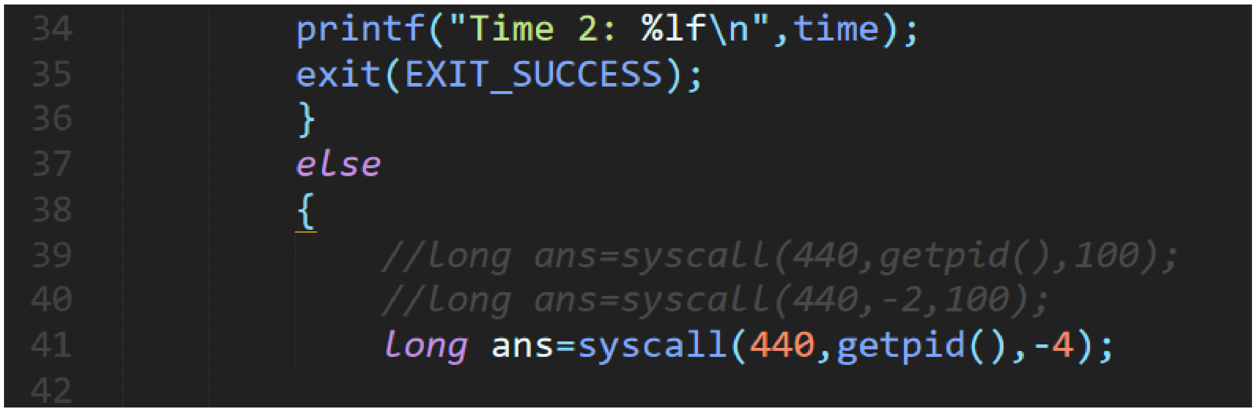
```c
        printf("Time 2: %lf\n",time);
        exit(EXIT_SUCCESS);
        }
        else
        {
            //long ans=syscall(440,getpid(),100);
            long ans=syscall(440,-2,100);
            //long ans=syscall(440,getpid(),-4);

```



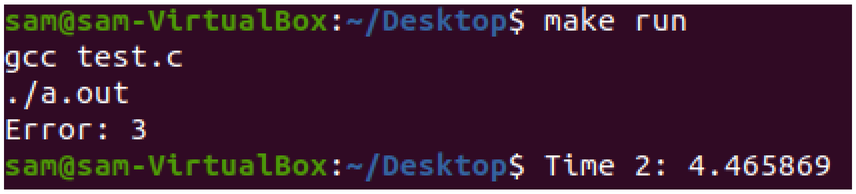
```
sam@sam-VirtualBox:~/Desktop$ make run
gcc test.c
./a.out
Error: 2
sam@sam-VirtualBox:~/Desktop$ Time 2: 3.772034
```

Sample 2 (Invalid time input)



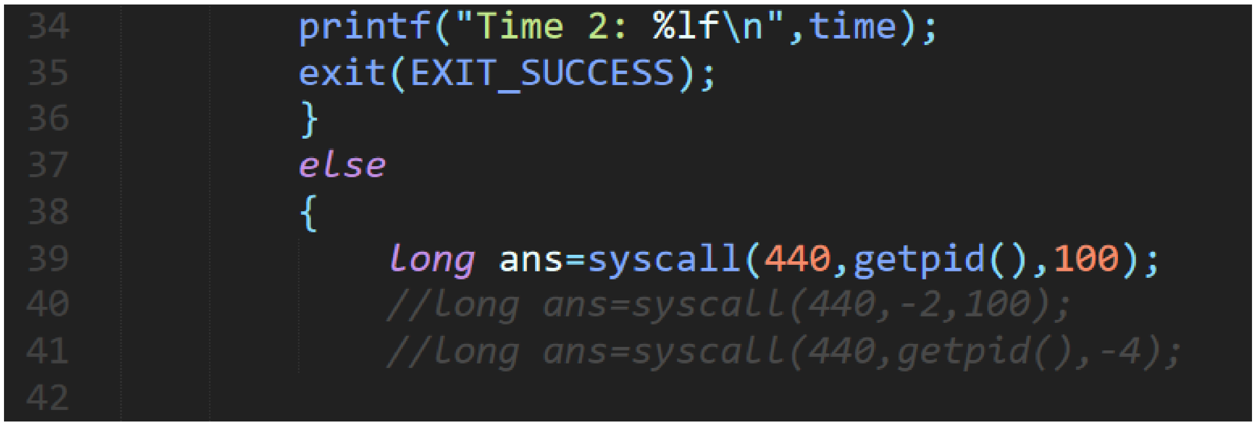
```c
        printf("Time 2: %lf\n",time);
        exit(EXIT_SUCCESS);
        }
        else
        {
            //long ans=syscall(440,getpid(),100);
            //long ans=syscall(440,-2,100);
            long ans=syscall(440,getpid(),-4);

```



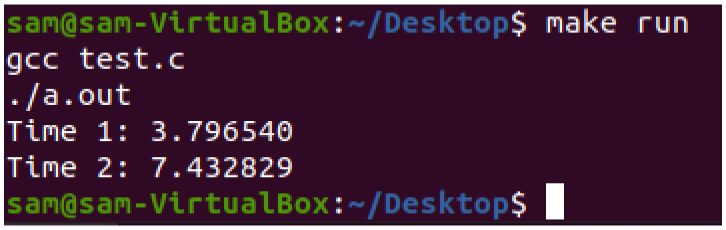
```
sam@sam-VirtualBox:~/Desktop$ make run
gcc test.c
./a.out
Error: 3
sam@sam-VirtualBox:~/Desktop$ Time 2: 4.465869
```

Sample 3 (Correct user inputs)



```
        printf("Time 2: %lf\n",time);
        exit(EXIT_SUCCESS);
        }
        else
        {
            long ans=syscall(440,getpid(),100);
            //long ans=syscall(440,-2,100);
            //long ans=syscall(440,getpid(),-4);

```



```
sam@sam-VirtualBox:~/Desktop$ make run
gcc test.c
./a.out
Time 1: 3.796540
Time 2: 7.432829
sam@sam-VirtualBox:~/Desktop$
```