Assignment 3

Documentation

Modifying scheduler policy and adding a system call for Linux v5.9 written in C

Part 1 : Compiling the source code

The linux 5.9 source code is downloaded and it's extracted.

- cd to that directory
- do 'cp /boot/config-$(uname -r) .config' to copy the kernel's .config file then followed by 'make menuconfig'
- use 'sudo make -j 2 && sudo make modules_install install -j 2 .The kernel and modules are compiled.
- Then do ‘reboot’.

Part 2: Modifying the scheduler policy for soft realtime requirements

- Every time the scheduler is called, we check if the real-time guarantees of processes with soft realtime requirements are being met or not. We gave higher priority to a processes soft realtime requirement compared to the vruntime that is normally considered.
- Implementation:
    - cd to the linux-5.9 dir.
    - First in the file include/linux/sched.h, add a field for the soft realtime requirement, rt_nice, in the struct sched_entity.
    - Then in the file kernel/sched/core.c, in the function static void __sched_fork, assign the default value of 0 to this rt_nice field.
    - Made changes in two functions in kernel/sched/fair.c:
        - entity_before - Added the code to compare the rt_nice of the two input sched_entities a and b. Process with higher soft realtime input are given higher priority. In case both of them are 0 (none are allotted a rt_nice), we return their comparison according to their vruntimes.
        - update_curr - If any sched_entity has a rt_nice value > 0, update the rt_nice value by subtracting the delta_exec value, otherwise do the same with vruntime.

Part 3: Writing the rt_nice system call

- In the linux-5.9 dir, make a new dir softRq/ and add the system call files there.
  - Make a new file rt_nice.c :
    - rt_nice(int pid, int soft_rq) takes an int argument pid and then an int argument soft_rq. First the value of the pid is checked. If it is <=0, -1 is returned and errno set to EINVAL. Then the value of the soft_rq is checked and if <0, then syscall returns -1 and errno is set to EINVAL. A task_struct pointer task is created. Then for each task, using for_each_process() function it's pid is checked, and if it matched the pid given as input, then the soft requirement field of its sched_entity (rt_nice) is set to the soft_rq input given. If no process with the input pid is found, then the errno is set to ESRCH, and it returns -1.
    - Thus the syscall returns 0 on successful execution, and returns -1 when unsuccessful, with the appropriate errno set.
  - A header file for it, rtNice.h:
    - Add the declaration for the syscall rt_nice as: asmlinkage long sys_rt_nice(int pid, int soft_rq).
  - A Makefile for compiling the above.
    - obj-y := rt_nice.o
- This new directory has to be added to the Linux-5.9's Makefile by changing the line 'core -y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/' to 'core -y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ softRq/'.
- The new system call has to be added,in the arch/x86/syscalls/syscall_64.tbl. I added it at 440. Thus the syscall number for rt_nice is 440.
- Then, the function was included in the file include/linux/syscalls.h by adding the line: asmlinkage long sys_rt_nice(int pid, int soft_rq) before the last #endif.
- It was recompiled again with commands by 'sudo make -j 2 && sudo make modules_install install -j 2'.
- The system is restarted.

Part 4: Inputs to the rt_nice syscall

- The first input to the syscall rt_nice is the pid of the process whose soft requirement is to be changed. The value of pid should be an integer >=1.
- The second input to the syscall is the integer soft_rq, that is the soft requirement to be given to the process whose pid is given as first input.

Part 5: Expected Output

- If the system call was executed correctly, then it would return 0, otherwise -1 with the appropriate errno set. The value returned by the system call and the errors(if any) are printed on the terminal.
- In case of successful execution, the rt_nice field of sched_entity of the process is set to the soft realtime requirement given as input.

Part 6: Errors Handled

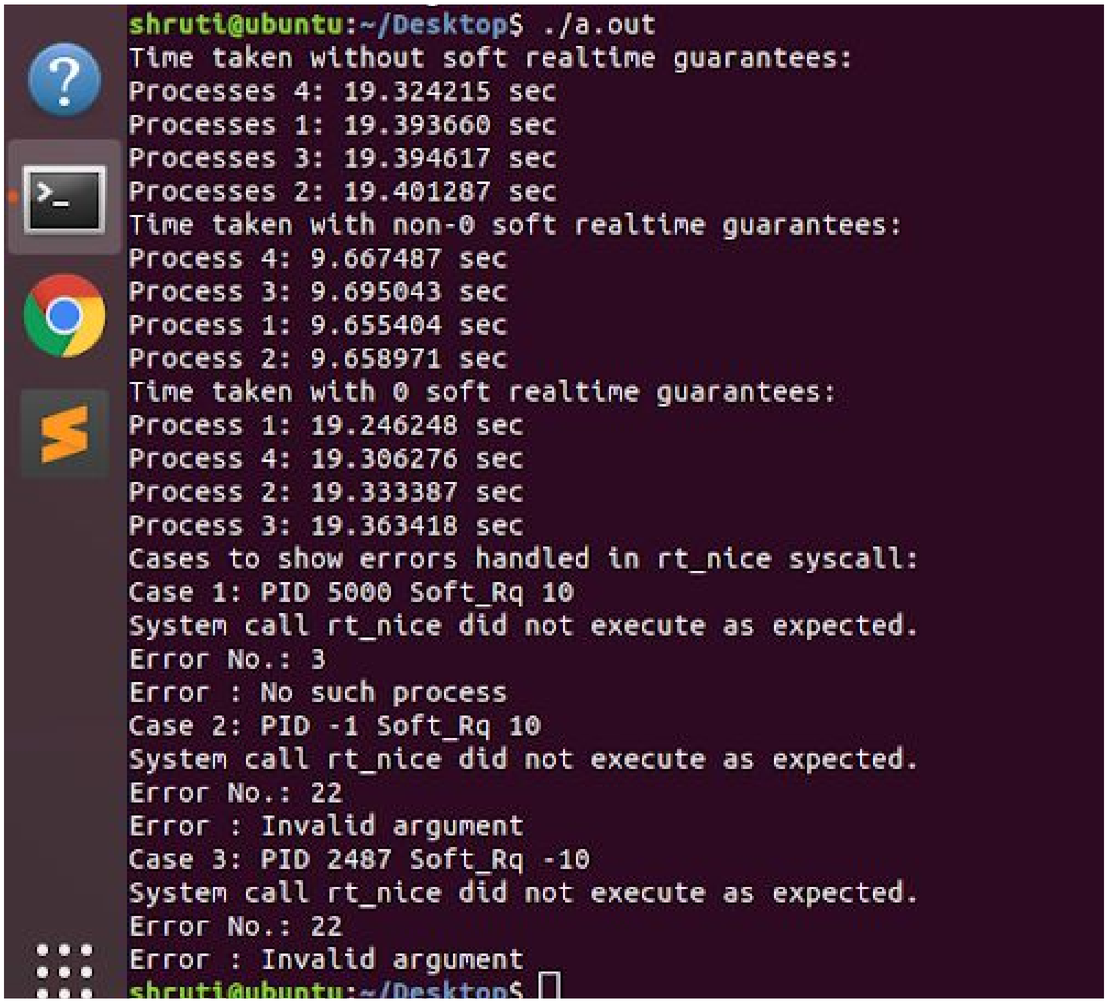
- If the user enters pid <=0, the function returns EINVAL, that is the errno: 22 and the error message is: Invalid argument.
- If the user enters a soft requirement <0, the function returns EINVAL, that is the errno: 22 and the error message is: Invalid argument.
- If the user enters a pid for which no process exists, then the syscall returns ESRCH, that is the errno: 3 and the error message: No such process.

Part 7: Testing it with test.c sample code

- The file test.c is the sample test code. It should be run with the makefile in the folder.
- First, to show the execution time taken by a process without calling the rt_nice syscall, a for loop with 4 iterations is run to fork a child process 4 times, and for each of them, the timer is started, then a time-consuming for loop is executed, and then the timer is stopped. The time taken by each of the 4 processes is printed.
- Then to show execution time taken by a process with calling the rt_nice syscall, a for loop with 4 iterations is run to fork a child process 4 times, and for each of them, first the syscall rt_nice is called to set the soft_requirement to a non-zero value, and then the timer is started, then a time-consuming for loop is called, and then the timer is stopped. The time taken by each of the 4 processes is printed.
- Again the same piece of code is repeated, but this time the soft requirement given to all the 4 child processes is 0.
- Lastly, a for loop with 3 iterations is run, to illustrate the three errors handled.
- We notice that the processes for which non-zero soft realtime requirements were provided completed execution in a shorter time. And the processes with 0 soft requirements completed execution in almost the same time as the ones for which no rt_nice syscall was made.

- The system call rt_nice is invoked by using the syscall() function, with the first argument 440 (since the system call rt_nice was added at 440 number in the kernel), and then the two arguments pid and soft_rq.
- If it is executed successfully, the syscall returns 0 and we can check the log by writing 'dmesg|tail' on the terminal. The rt_nice field of sched_entity of the process is set to soft_rq.
- If it was not executed successfully, then -1 is returned and the errno along with the error message is printed using perror().

The output of test.c looks as follows:



```
shruti@ubuntu:~/Desktop$ ./a.out
Time taken without soft realtime guarantees:
Processes 4: 19.324215 sec
Processes 1: 19.393660 sec
Processes 3: 19.394617 sec
Processes 2: 19.401287 sec
Time taken with non-0 soft realtime guarantees:
Process 4: 9.667487 sec
Process 3: 9.695043 sec
Process 1: 9.655404 sec
Process 2: 9.658971 sec
Time taken with 0 soft realtime guarantees:
Process 1: 19.246248 sec
Process 4: 19.306276 sec
Process 2: 19.333387 sec
Process 3: 19.363418 sec
Cases to show errors handled in rt_nice syscall:
Case 1: PID 5000 Soft_Rq 10
System call rt_nice did not execute as expected.
Error No.: 3
Error : No such process
Case 2: PID -1 Soft_Rq 10
System call rt_nice did not execute as expected.
Error No.: 22
Error : Invalid argument
Case 3: PID 2487 Soft_Rq -10
System call rt_nice did not execute as expected.
Error No.: 22
Error : Invalid argument
shruti@ubuntu:~/Desktop$
```