OS ASSIGNMENT 2

README:

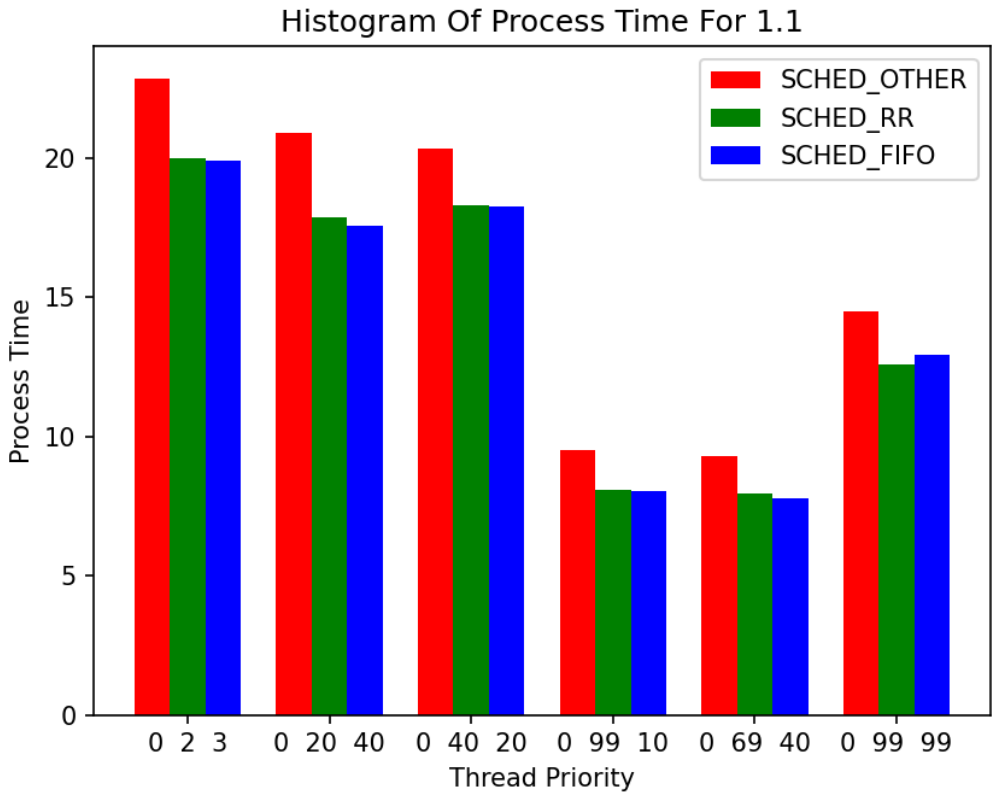
For 1.1:

I have first made 3 function countsA countsB countsC which counts from 1 to 2^23. Then i have created 3 thread function(countA…) in which i have stated the timer and the ran countsA and rest of the functions in it. I have created a main function in which I am going to create 3 threads and join them. but before that i have set the priority of the threads as well as set the policy of the threads also as given in the question. After running the program I print the time.

Explaining the output as the SCHED_OTHER has the nice value the lowest i.e 0 and we can set any other priority values for SCHED_FIFO and SCHED_RR . Sometime it could be due to the no of cores allocated to the vm which causes disparity. Thereafter the SCHED_OTHER take the longest time than the SCHED_FIFO or SCHED_RR which is according to CFS scheduler.

Data for same is in a txt file named: datafor1.1.txt and the graph implemented in graphfor1.1.py

The histogram is as follows:

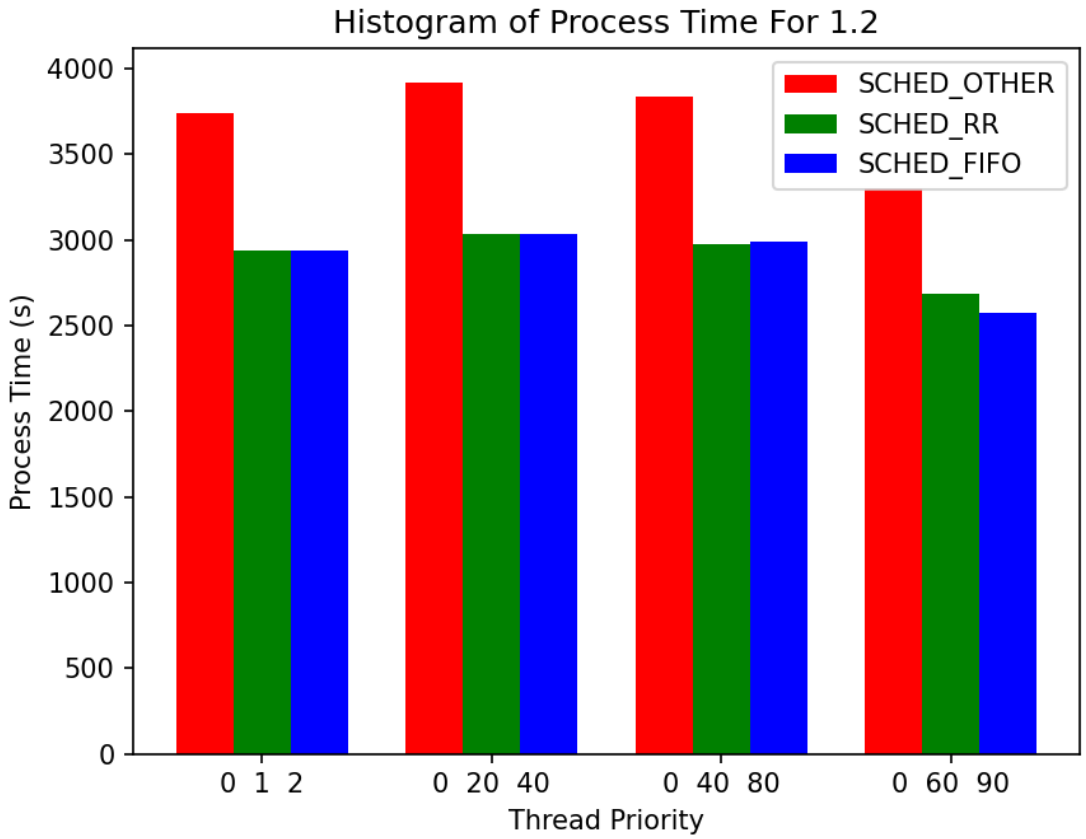


For 1.2

In the question it is said that i have to create 3 process and we have compile the kernel. We have to use fork and the child processes should use execl to run different bash script.For this i have done everything in the main function itself and created 3 process id and then forked them and then used execp to run the bash script. i am using a nested if else to create child processes. Then i am getting the pid and seeing if it is 0 (means it has forked with error) then using it to set the scheduling policy as well as the priority and then running the bash script. I am starting the time just after we have created a child process and then i am ending the time after the child process has finished by comparing the process ids. for the wait status of the process i have created a for loop in which i have check the state and the get the end time and save it. after that we have all the end time and all child process has been completed I print the time taken by 3 process to execute the bash script.

data for the same is in a txt file named: datafor1.2.txt the graph implemented in graphfor1.2.py

The histogram is as follows:



README:

For 2:

In this question we had to make a systemcall to copy a 2d matrix.

For this to compile a system call i have first i have gone to arch/x86/entry/syscalls/syscall_64.tbl. and added a systemcall at 451 named kernel_2d_memcpy. In the table he first column is the system call number.The second column says that this system call is common to both 32-bit and 64-bit CPUs. The third column is the name of the system call, and the fourth is the name of the function implementing it. after adding this i have gone to the root directory and made a directory named kenel_2d_memcpy_syscall in which has 2 files ker_2d_memcpy_syscall.c and a make file to compile the system call. you can see the changes in the diff.patch file that i have submitted. then after that i have gone to root and make changes in the make file of the kernel compilation in that line 1103 we have to add the name of the directory in which we have made the system call. then we have just run make -j$(nproc) and make modules_install and copy the bzimage to the boot directory and then run make install. after that we have to reboot the system and then we can use the system call.

For checking the systemcall i have made a test file which shows that the systemcall works fine and it is copying the matrix correctly by printing what the matrix before and what has it changed to.