Q1 Write-Up

Linux Scheduler

Working

The following changes were made for this implementation:
In the sched.h file present in include/linux/sched.h, I introduced a new u64 type as delaytime in the sched_entity struct, representing the delay that we need to pass. In the core.c file present in kernel/sched/core.c, in the __sched_fork() function, I added a new attribute to the task_struct p as p->se.delaytime and initialized it to 0. After that in the scheduler file fair.c present in kernel/sched/fair.c, in the update_curr() function, after the vruntime is updated by the calc_delta_fair() amount, I further updated the vruntime by the delaytime we initialized by doing
curr->vruntime += curr->delaytime; Hence we modified the CFS scheduler so that the vruntime is updated according to the delay passed.

To define the system call I modified the sys.c file present in kernel/sys.c, and used the SYSCALL_DEFINE2 macro that has parameters as the PID of the process whose vruntime should be changed and the delta amount time (in milli-seconds) by which the selection should be delayed.
To get the task_struct of the pid, first I used the find_get_pid() function which returns a pid struct and used pid_task() to get the task struct of this pid. After that, I have updated the delaytime of our task_struct variable 'p' by increasing the delaytime by the delta (multiplied by 10^6 to get in nanoseconds) using the statement p->se.delaytime += delta*(1000000); Now, whenever the vruntime is calculated for the process, it will also get updated by this delta delay (as we defined in fair.c).
Finally, I modified the syscall_64.tbl file present in arch/x86/entry/syscalls and added the new syscall name as 'change_vruntime' and number as 449.