To Do:
The system call must pass on information, the additional delay (in msec) (that has been accepted from the userland) to the scheduler such that anytime the said process is chosen by the scheduler, it adjusts the vruntime of the said process so as to delay its selection.

System Call:

- SYSCALL_DEFINE2 was used as 2 input parameters are passed to the syscall.The two parameters are the pid of the process and other is the additional delay which is to be added to the vruntime of the process whenever the process is chosen by the scheduler.
- Syscall:

```
+SYSCALL_DEFINE2(update_process, int, pid, s64, update_value){
+       struct task_struct *p = NULL, *temp_struct = NULL;
+       if(pid <= 0 || update_value <= 0){
+               return -EINVAL;
+       }
+       for_each_process(temp_struct){
+               if((long)task_pid_nr(temp_struct) == pid){
+                       p = temp_struct;
+                       break;
+               }
+       }
+       if(p == NULL){
+               return -EINVAL;
+       }
+       p->se.update_value = update_value;
+       printk("update_value changed");
+       return 0;
+}
```

- This syscall is defined in kernel/sys.c and the system call is added to the system call table for x86_64
- The system call defined also handles errors, If invalid process id is passed, then it returns -EINVAL. And errno is set accordingly.
  And if the update_value(delay) passed by the caller is not in the correct range, it returns -EINVAL and errno is set accordingly. These errors are printed using perror() in test.c (caller program)

Modifying CFS:

- We add a variable ‘update_value’ of type u64 in the struct sched_entity’, which is defined in ‘include/linux/shed.h’. This is used to store delay_time which is added to vruntime whenever the said process is selected by the scheduler.

- For other process's update_value is set to zero. This is done by initialization of the variable update_value by modifying the file 'kernel/sched/core.c' The variable is initialized in ' __sched_fork' function where other variables like vruntime is also initialized.
- In fair.c whenever the said process, is selected its vruntime is increased by the value stored in update_value. This is done in the function pick_next_task_fair() which is defined kernel/sched/fair.c.
  // by the following code

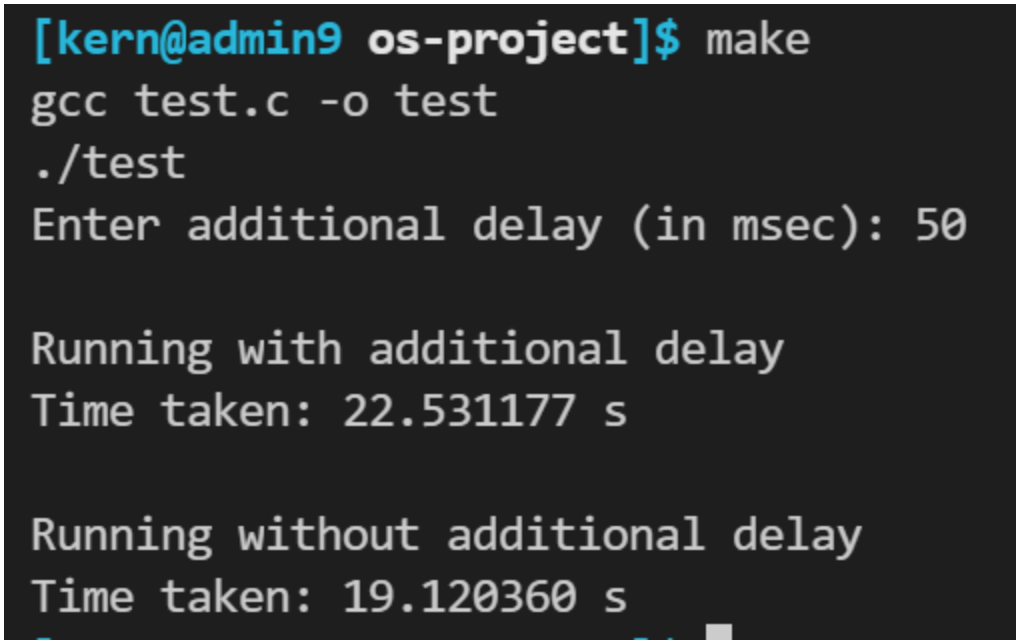
```
if(p->se.update_value > 0 ){
        p->se.vruntime = p->se.vruntime + p->se.update_value;
}
u64 timeslice_p = ktime_get_ns();
pr_info("Pid, TimeSlice: %d %llu\n", p->pid, timeslice_p);
 return p;
```

- To print the timestamp corresponding to when each process is selected and dispatched to run. ktime_get_ns() function is used which gives time in ns.Now timestamp along the pid of the process it is associated with is printed in the kernel log .this is done by pr_info and ktime_get_ns fucntions.
- The kernel log can be checked by this command sudo dmesg | tail -5
  Example:
  [7227.744798] Pid, TimeSlice: 4475 7227746131678
  [7227.745054] Pid, TimeSlice: 4476 7227746387847
  [7227.745250] Pid, TimeSlice: 4475 7227746583980
  [7227.745456] Pid, TimeSlice: 4478 7227746789584
  [7227.745729] Pid, TimeSlice: 4478 7227747062638

  format(pid(int) timeslice(ns))

Test.c

- Makefile to run test.c is provided just do 'make'.
- The program asks for additional delay value (in ms) input, from the user after which it forks the process.
- In the child process system call update_process is called with the delay provided by the user, both parent and child process perform the same task of looping.
  But we observe mostly child process which has additional delay takes more time to finish the task.The time taken by both child and the parent process to do the same task is printed.This shows the process with additional delay is scheduled less than the process which has no delay.



Diff Files:
Two diff files are provided one before kernel was compiled and one after the kernel was compiled after adding the new system call.