18-342 Group 32
Yuanshuo Cui (ycui)
Yujing Zhang (yujingz)
Liyu Ma (liyum)

Documentation for lab 4

In this lab, we upgraded existed Gravel kernel to Gravel2 kernel. The original Gravel kernel has our own SWI and IRQ handler, and has system call includes read, write, exit, time and sleep.

- **Kernel**

1. Kernel entrance (main.c init_asm.S install_handler.c interrupt_setup.c)
   In this part, we will install the SWI and IRQ handler. As well as set up the interrupt controller, mutex and system timer. After those initializations, the program will set user stack and jump to indicate user program. In lab4, we put an infinite loop in crt0.S, so there is no need to recover original IRQ handler and SWI handler. Also we do not need to forward all arguments to user program.

2. SWI handler (C_SWI_handler.c s_handler.S)
   For SWI handler, we add support for task_create, mutex_create, mutex_lock, mutex_unlock and event_wait system call. The procedure to handle these system calls is similar to previous system calls.

3. IRQ handler (C_IRQ_handler.c)
   The IRQ handler now has a new functionality that it also updates the devices.

- **System calls**

1. I/O system calls (io.c)
   We port the original read and write call into single file. Besides that, there is no change for I/O related system calls.

2. Time system calls (time.c)
   We port the original time and sleep call into single file. Besides that, there is no change for time related system calls.

3. Process system calls (proc.c)
   The task_create function will check for error, initialize running queue and devices. After that, we will do ubtest on all tasks to check if tasks are schedulable. If everything is fine, it will initialize tcb of new task and idle task, and then put them to running queue. Finally does a nosave context switch.

   The event_wait function will check for error and wait for a certain device number.

- **Lock (mutex.c)**

  The mutex_init function will set all mutex to initial status.

  The mutex_create function will try to enable a new mutex if there is still available mutex slot.

  The mutex_lock function will check for wrong mutex number and availability. If the mutex is hold by other task, the program will find the end of mutex sleep queue to put current task. When the mutex is not held by any other process, it will hold this mutex immediately. And set the priority of current task to the highest.

  The mutex_unlock function will check for wrong mutex number and availability. If the mutex is held by the current task, the program will release this mutex and update current task's tcb immediately. Also let the first task in sleep queue to hold the mutex.

- **Scheduler**

1. Context switch (ctx_switch.c ctx_switch_asm.S)

   The dispatch_init function will switch IDLE task.

   The dispatch_save function will only switch to the highest priority task including the current task and tasks in the run queue. If current task is not with the highest priority, it will add current task to run queue, swap current task and the highest priority task in run queue. And call full context switch in the .S file at last.

   The dispatch_nosave function is pretty similar but without save the current task's context.

   The dispatch_sleep function will always switch to the highest priority task in the run queue, and sleep the current task (not put into run queue).

2. Running queue (run_queue.c)

   The runqueue_init function will clear run list and run bit.

   The runqueue_add and runqueue_remove function will add tcb the running queue or clear run queue for certain priority.

3. Task schedule (sched.c)
   This part will either setup tcb for given task or IDLE task and then add it to running queue.

4. Simulate device (device.c)
   The dev_init function will setup next match time and sleep queue for each device.

   The dev_wait function will fetch sleep queue of current device, then try to put cur tcb to the

end of the sleep queue.

The dev_update function will find device that has meet the next_match value. For all these devices, program will move all tasks in sleep queue to running queue.

5. UB Test (ub_test.c)
   UB test will sort all task based on priority, and then follow the formula for ubtest to find out whether all the tasks can be scheduled.