PintOS Project 1

2025-03-21

Before you start

- Make commits as many as possible
  - Commit message will tell what you implement!
  - If you don’t commit frequently, your code will be thoroughly inspected.

- Read the manual carefully
  - https://web.stanford.edu/class/cs140/projects/pintos/pintos.pdf
  - Section 2, A.2, A.3, E
  - Background and FAQ is also important!
  - NOTE: Except for manual, you can’t refer any materials which you can get via internet (IF NOT, IT WILL BE PLARGIARISM)

Manual for the project 1

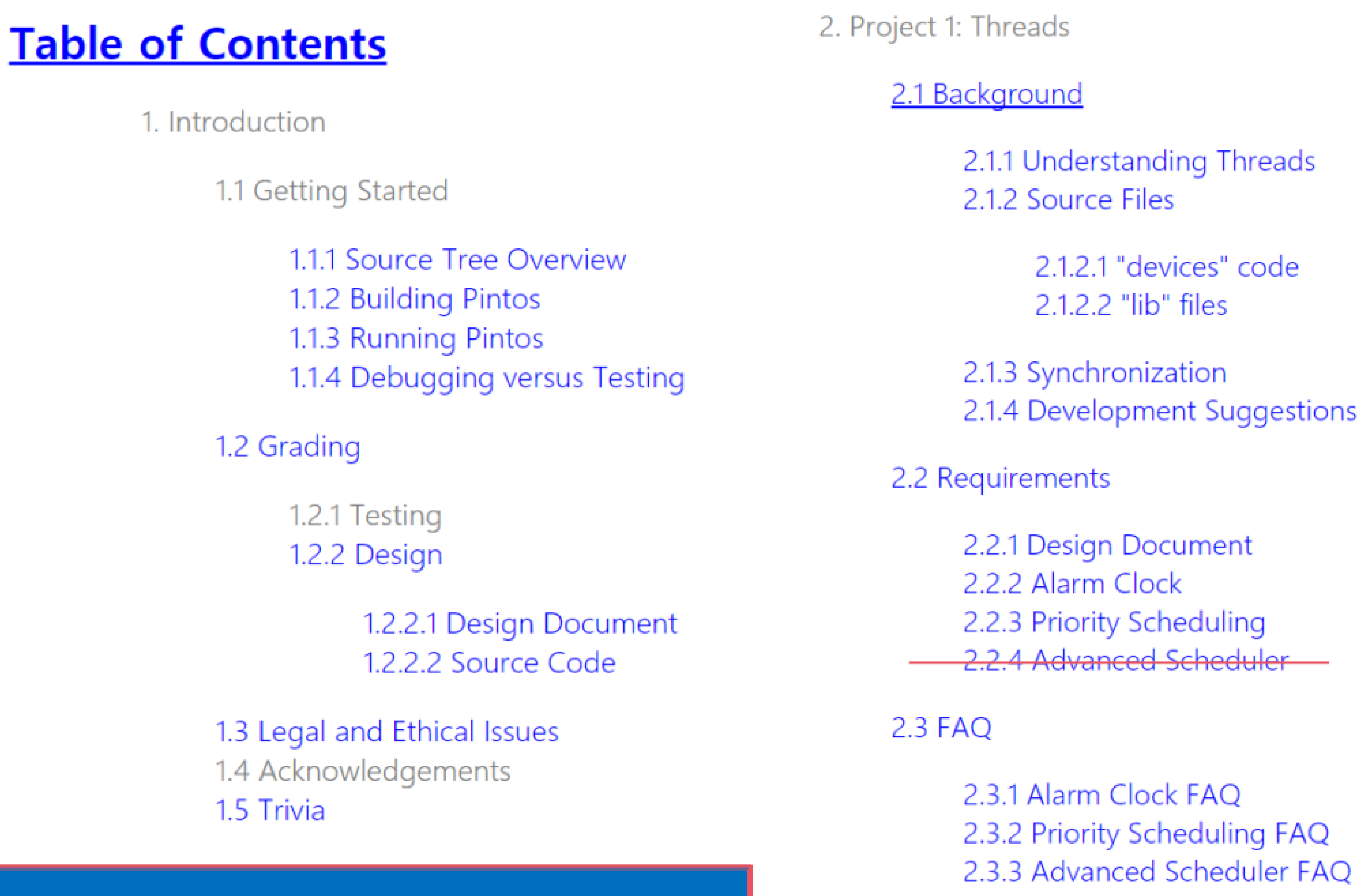


Table of Contents

Please read the manual one by one words.

Manual for the project 1

E. Debugging Tools

Please read the manual one by one words.

1. Alarm Clock

Background

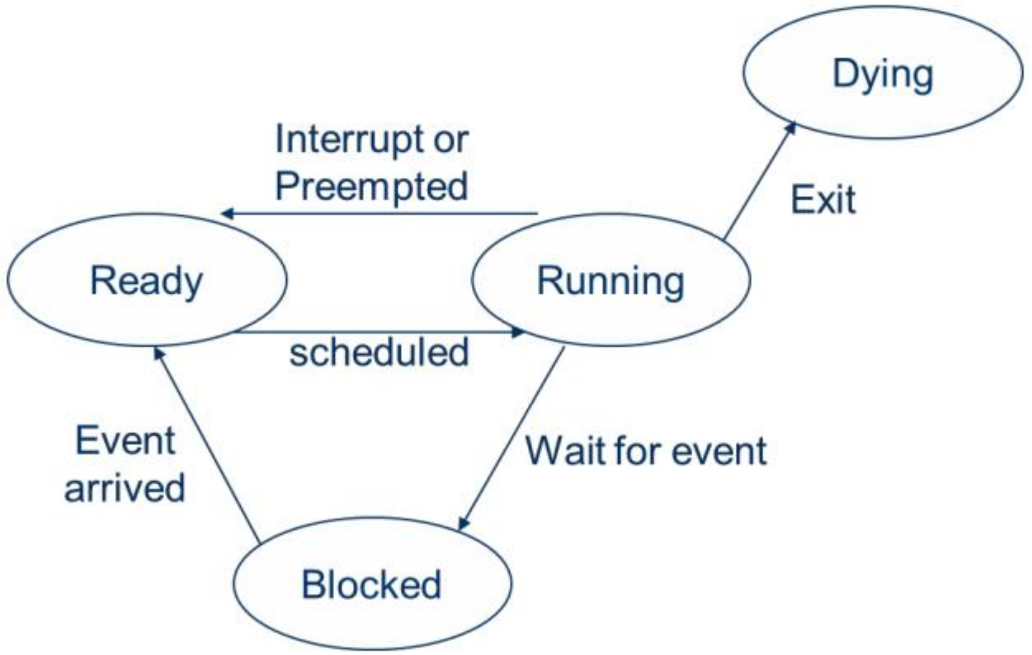
- Thread

```
struct thread {
	tid_t tid;
	enum thread_status status;
	char name[16];
	uint8_t *stack;
	int priority;
	struct list_elem allelem;

	/* Shared between thread.c and synch.c.
	struct list_elem elem;

	add more fields here as you need them.

#ifdef USERPROG
	/* Owned by userprog/process.c. */
	uint32_t *pagedir;
#endif
	/* Owned by thread.c. */
	unsigned magic;
};
```



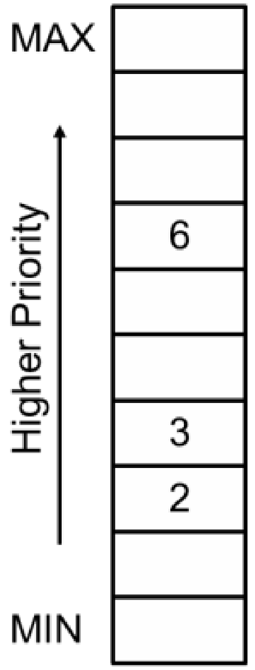
Alam clock

- To do
  - Suspend execution for approximately TICKS (timer ticks)
    - Simply calls “busy waits”
  - To avoid busy waiting
    - Put thread to sleep and wake it up when enough time has elapsed
  - Reimplement `timer_sleep( )`
    - Defined in `<src/devices/timer.c>`
  - Related file
    - `devices/timer.c , threads/thread.c , threads/thread.h`
    - `lib/kernel/list.c`
  - [HINT] You might need to know <u>how a timer tick is increased.</u>

2. Priority Scheduling

Background

- Priority scheduling
  - Each thread is given a scheduling priority
  - Chooses the thread with the highest priority in the ready queue to run next
  - Thread priorities in Pintos
    - 64 priority levels (default =31)
    - Lower number correspond to lower priorities



Priority Scheduling

- To do
  - Current PintOS system uses a round robin scheduler

  - Change this to instead schedule the highest priority thread that can run
    - Function `void thread_set_priority (int new_priority)`
      Sets the current thread's priority to new_priority. If the current thread no longer has the highest priority, yields.
    - Function `int thread_get_priority (void)`
      Returns the current thread's priority.

  - Related file
    - threads/thread.c , threads/thread.h

Priority Scheduling

- Priority donation
    - Do not do Priority donation

Grading

- You can check your progress anytime.
  - **After compiling**
  - In “threads/build” directory,
    ```
    $ cd build
    $ make grade
    $ vi grade
    ```
  - Test list that you should pass during the project #1.

Alarm clock

- ✓ alarm-single
- ✓ alarm-multiple
- ✓ alarm-simultaneous
- ✓ alarm-priority
- ✓ alarm-zero
- ✓ alarm-negative

Priority scheduling

- ✓ priority-change
- ✓ priority-preempt
- ✓ priority-fifo

Grading

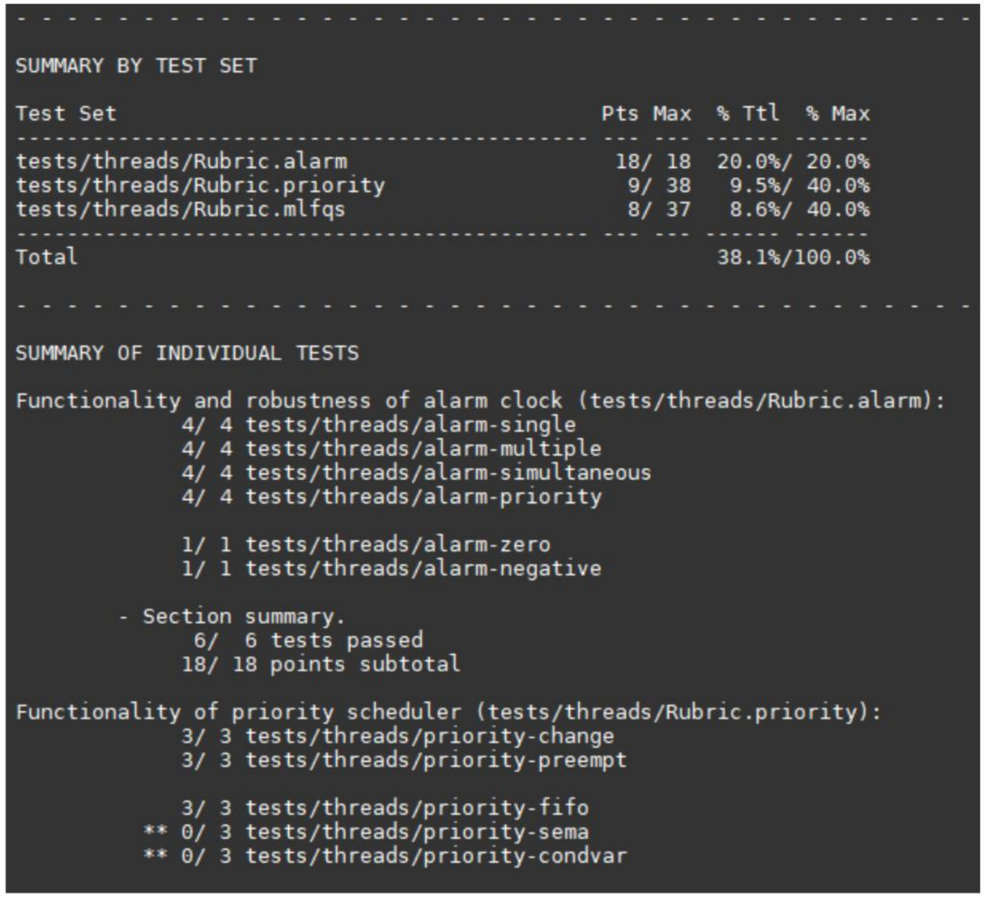
- “make grade” takes long time.
- For each test, you can use this commands
  - `$ pintos -v -k -T 60 --bochs -- -q run “test_name”`

  - Ex) if you want to check ‘alarm-multiple’ test,
    ```
    $ pintos -v -k -T 60 --bochs -- -q run alarm-multiple
    ```

- You can check test code
  - src/tests/threads/(test_name).c

Example

If *grade* is same with this, you will get 100 points for grade of project 1



```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

SUMMARY BY TEST SET

Test Set                                      Pts Max  % Ttl  % Max
--------------------------------------------- --- --- ------ ------
tests/threads/Rubric.alarm                     18/ 18  20.0%/ 20.0%
tests/threads/Rubric.priority                   9/ 38   9.5%/ 40.0%
tests/threads/Rubric.mlfqs                      8/ 37   8.6%/ 40.0%
--------------------------------------------- --- --- ------ ------
Total                                                  38.1%/100.0%

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

SUMMARY OF INDIVIDUAL TESTS

Functionality and robustness of alarm clock (tests/threads/Rubric.alarm):
             4/ 4 tests/threads/alarm-single
             4/ 4 tests/threads/alarm-multiple
             4/ 4 tests/threads/alarm-simultaneous
             4/ 4 tests/threads/alarm-priority

             1/ 1 tests/threads/alarm-zero
             1/ 1 tests/threads/alarm-negative

        - Section summary.
              6/  6 tests passed
             18/ 18 points subtotal

Functionality of priority scheduler (tests/threads/Rubric.priority):
             3/ 3 tests/threads/priority-change
             3/ 3 tests/threads/priority-preempt

             3/ 3 tests/threads/priority-fifo
          ** 0/ 3 tests/threads/priority-sema
          ** 0/ 3 tests/threads/priority-condvar
```

Submission

- Due: **April 2nd 11.59 PM**
- **Submit via GitHub**

- Push your code before the due date.
  - The Last submission before the due will be graded.

- If you have question during the project 1, please mail to TAs.
  jungss0123@unist.ac.kr
  eomjaeeun@unist.ac.kr
  ogus05@unist.ac.kr

About Errors

- If you happen to see this error [no "Pintos booting" message]
  - In pintos/src/Make.tests
  - You can find "VERBOSE = " in line 52 and simply change into "VERBOSE = 1"
- Permission error when you type make grade
  - cd ${your_pintos_directory}/tests
  - chmod +x make-grade

GDB

- Before you use the pintos-gdb, you should change the path.
  - At `pintos/src/utils/pintos-gdb`
  - `GDBMACROS=/root/pintos/src/misc/gdb-macros`