Operating Systems
Tutorial 3:Constructing Alarm clock in Pintos

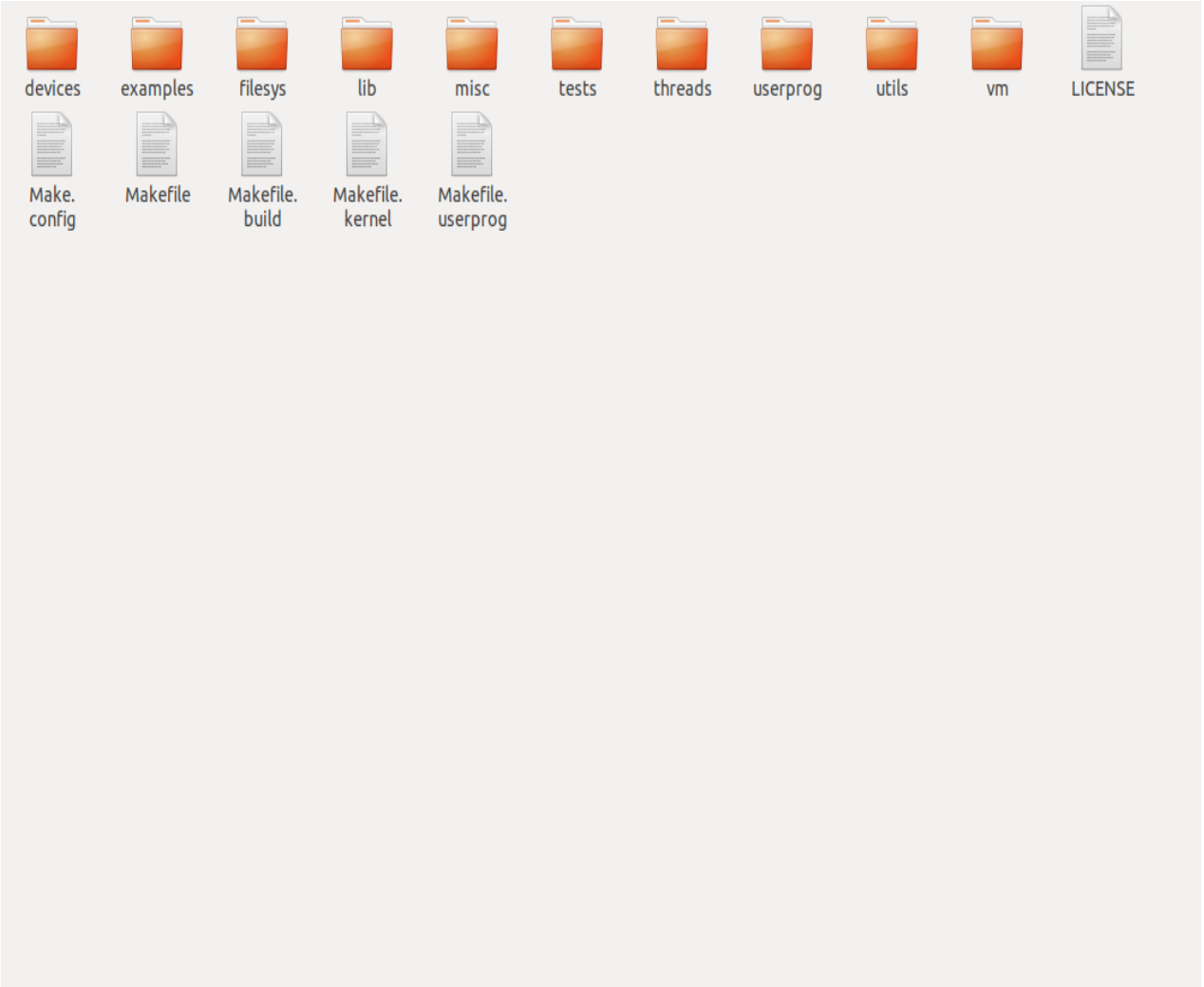
Nguyen Trung Nam (tutor)

Hanoi University of Science and Technology

Compiled with reference to other presentations

Source Tree Overview

- threads/ : source code for the base kernel, which you will modify starting in project 1.
- userprog/ : source code for the user program loader, which you will modify starting with project 2.
- vm/ : an almost empty directory, implement virtual memory here in project 3.
- filesys/ : source code for the a basic file system, you will use this file system starting with project 2,but will not modify it until project 4.



Source Tree Overview

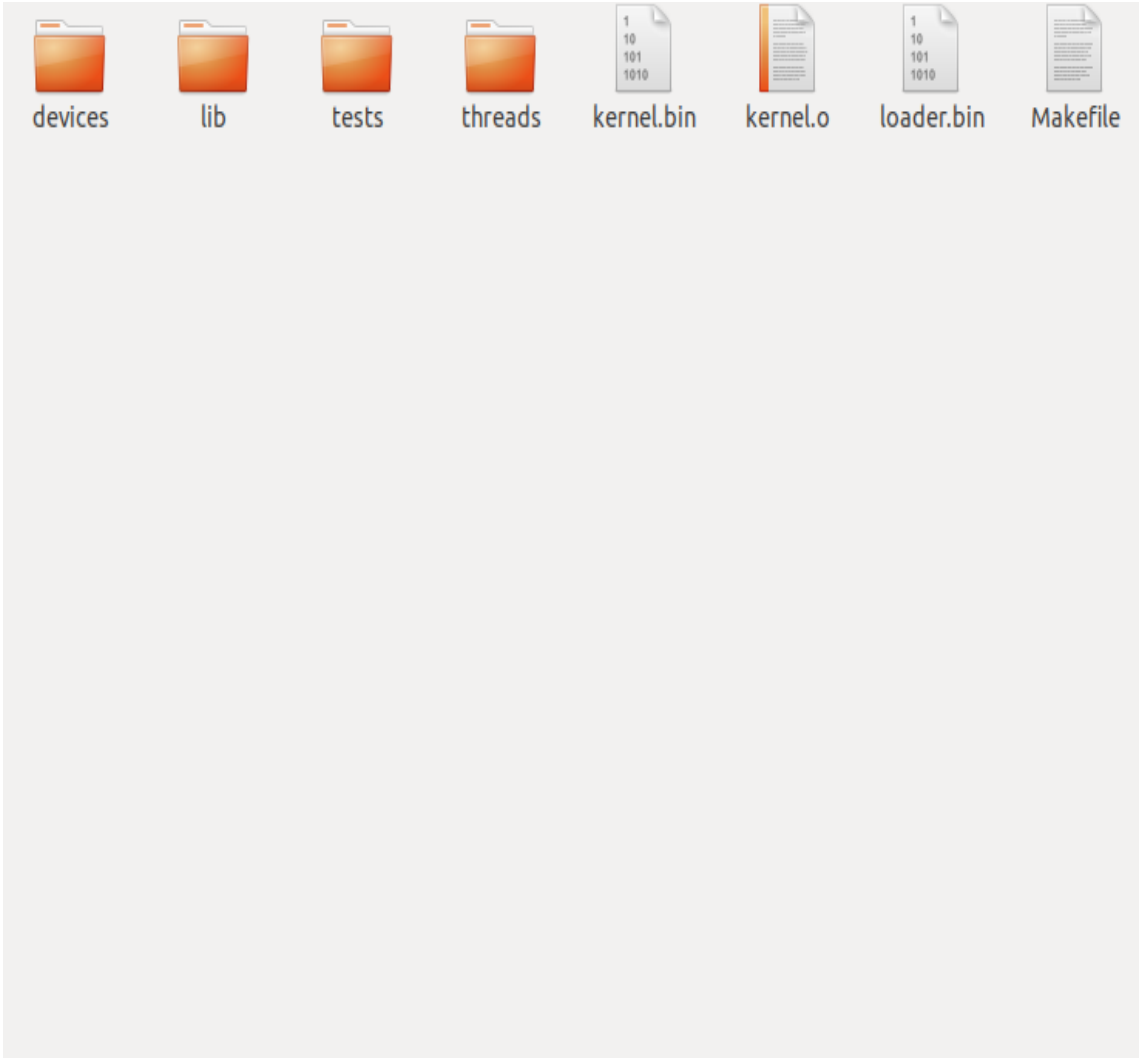
- devices/ : source code for I/O device interfacing : keyboard, timer,disk, etc. Implementation the timer in project 1.
- lib/ : an implementation of a subset of the standard C library.
- lib/kernel/ : Part of the C library that are included only in the Pintos kernel.
- lib/user/ : part of the C library that are included only in Pintos user programs.

Source Tree Overview

- test/ : test of each project,you can modify this code if it helps you test your submission.
- examples/ : example user programs for use starting with project 2.
- utils/ & misc/: these files may come in handy if you decide to try working with Pintos on your own machine. You can ignore them.

Building & Running Pintos

- Cd threads/ & typing "make" command.
- Makefile : a copy of "src/Makefile.build",describes how to build the kernel.
- kernel.o : result of linking object files complied from each individual kernel source file into a single object file.
- kernel.bin : memory image of the kernel,the exact bytes loaded into memory to run the Pintos kernel.
- loader.bin : memory image for the kernel loader,a small chunk of code written in assembly language that reads the kernel from disk into memory and starts it up.
- Running Pintos Kernel :
  pintos run <project test>



Tests Kernel Pintos

- Each Pintos project comes with a set of tests :

- Useful for debugging.

- Also what we will use to grade your code.

- Use "$make check" command to run the tests.

```
southcenter@SouthCenter:~/pintos-anon/src/threads/build$ make check
pintos -v -k -T 60 --qemu  -- -q  run alarm-single < /dev/null 2> tests/threads/alarm-single.errors > tests/threads/alarm-single.output
perl -I../.. ../../tests/threads/alarm-single.ck tests/threads/alarm-single tests/threads/alarm-single.result
pass tests/threads/alarm-single
pintos -v -k -T 60 --qemu  -- -q  run alarm-multiple < /dev/null 2> tests/threads/alarm-multiple.errors > tests/threads/alarm-multiple.output
perl -I../.. ../../tests/threads/alarm-multiple.ck tests/threads/alarm-multiple tests/threads/alarm-multiple.result
pass tests/threads/alarm-multiple
pintos -v -k -T 60 --qemu  -- -q  run alarm-simultaneous < /dev/null 2> tests/threads/alarm-simultaneous.errors > tests/threads/alarm-simultan
eous.output
perl -I../.. ../../tests/threads/alarm-simultaneous.ck tests/threads/alarm-simultaneous tests/threads/alarm-simultaneous.result
pass tests/threads/alarm-simultaneous
pintos -v -k -T 60 --qemu  -- -q  run alarm-priority < /dev/null 2> tests/threads/alarm-priority.errors > tests/threads/alarm-priority.output
perl -I../.. ../../tests/threads/alarm-priority.ck tests/threads/alarm-priority tests/threads/alarm-priority.result
FAIL tests/threads/alarm-priority
Test output failed to match any acceptable form.

Acceptable output:
  (alarm-priority) begin
  (alarm-priority) Thread priority 30 woke up.
  (alarm-priority) Thread priority 29 woke up.
  (alarm-priority) Thread priority 28 woke up.
  (alarm-priority) Thread priority 27 woke up.
  (alarm-priority) Thread priority 26 woke up.
  (alarm-priority) Thread priority 25 woke up.
  (alarm-priority) Thread priority 24 woke up.
  (alarm-priority) Thread priority 23 woke up.
  (alarm-priority) Thread priority 22 woke up.
  (alarm-priority) Thread priority 21 woke up.
  (alarm-priority) end
Differences in `diff -u' format:
  (alarm-priority) begin
- (alarm-priority) Thread priority 30 woke up.
- (alarm-priority) Thread priority 29 woke up.
- (alarm-priority) Thread priority 28 woke up.
- (alarm-priority) Thread priority 27 woke up.
- (alarm-priority) Thread priority 26 woke up.
  (alarm-priority) Thread priority 25 woke up.
```

Tests Kernel Pintos


```
southcenter@SouthCenter:~/pintos-anon/src/threads/build$ pintos run alarm-single
qemu-system-x86_64 -device isa-debug-exit -hda /tmp/Cekqv_rGYC.dsk -m 4 -net none -serial stdio
WARNING: Image format was not specified for '/tmp/Cekqv_rGYC.dsk' and probing guessed raw.
         Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
         Specify the 'raw' format explicitly to remove the restrictions.
qemu-system-x86_64: warning: TCG doesn't support requested feature: CPUID.01H:ECX.vmx [bit 5]
PiLo hda1
Loading............
Kernel command line: run alarm-single
Pintos booting with 3,968 kB RAM...
367 pages available in kernel pool.
367 pages available in user pool.
Calibrating timer...  270,336,000 loops/s.
Boot complete.
Executing 'alarm-single':
(alarm-single) begin
(alarm-single) Creating 5 threads to sleep 1 times each.
(alarm-single) Thread 0 sleeps 10 ticks each time,
(alarm-single) thread 1 sleeps 20 ticks each time, and so on.
(alarm-single) If successful, product of iteration count and
(alarm-single) sleep duration will appear in nondescending order.
(alarm-single) thread 0: duration=10, iteration=1, product=10
(alarm-single) thread 1: duration=20, iteration=1, product=20
(alarm-single) thread 2: duration=30, iteration=1, product=30
(alarm-single) thread 3: duration=40, iteration=1, product=40
(alarm-single) thread 4: duration=50, iteration=1, product=50
(alarm-single) end
Execution of 'alarm-single' complete.
Timer: 279 ticks
Thread: 250 idle ticks, 30 kernel ticks, 0 user ticks
Console: 984 characters output
Keyboard: 0 keys pressed
Powering off...
```

Debugging Tool

- printf() : easy way to debug
- Assert() : Assert(expression).
- Backtraces : insert a call debug_backtrace() ,prototyped in <debug.h>

Pintos Bootup Sequence

- src/threads/init.c -> main()

```
bss_init (); /* Clear the BSS */

argv = read_command_line ();
argv = parse_options (argv);

thread_init ();
console_init ();

printf ("Pintos booting with...");

/* Initialize memory system. */
palloc_init (user_page_limit);
malloc_init ();
paging_init ();

/* Segmentation. */
tss_init ();
gdt_init ();
```

```
/* Enable Interrupts */
intr_init ();

/* Timer Interrupt */
timer_init ();

/* Keyboard */
kbd_init ();
input_init ();
exception_init ();

/* Enable syscalls */
syscall_init ();

/* Initialize threading */
thread_start ();
serial_init_queue ();
timer_calibrate ();
```

```
/* Initialize the hard
   drive and fs */
ide_init ();
locate_block_devices ();
filesys_init (format_filesys);

printf ("Boot complete.\n");

/* Run actions specified
   on kernel command line. */
run_actions (argv);

shutdown ();
thread_exit ();
```

17