The Pintos Instructional Operating System Kernel

Ben Pfaff
blp@nicira.com
Nicira Networks

Anthony Romano
ajromano@stanford.edu
Stanford University

Godmar Back
gback@cs.vt.edu
Virginia Tech

Overview

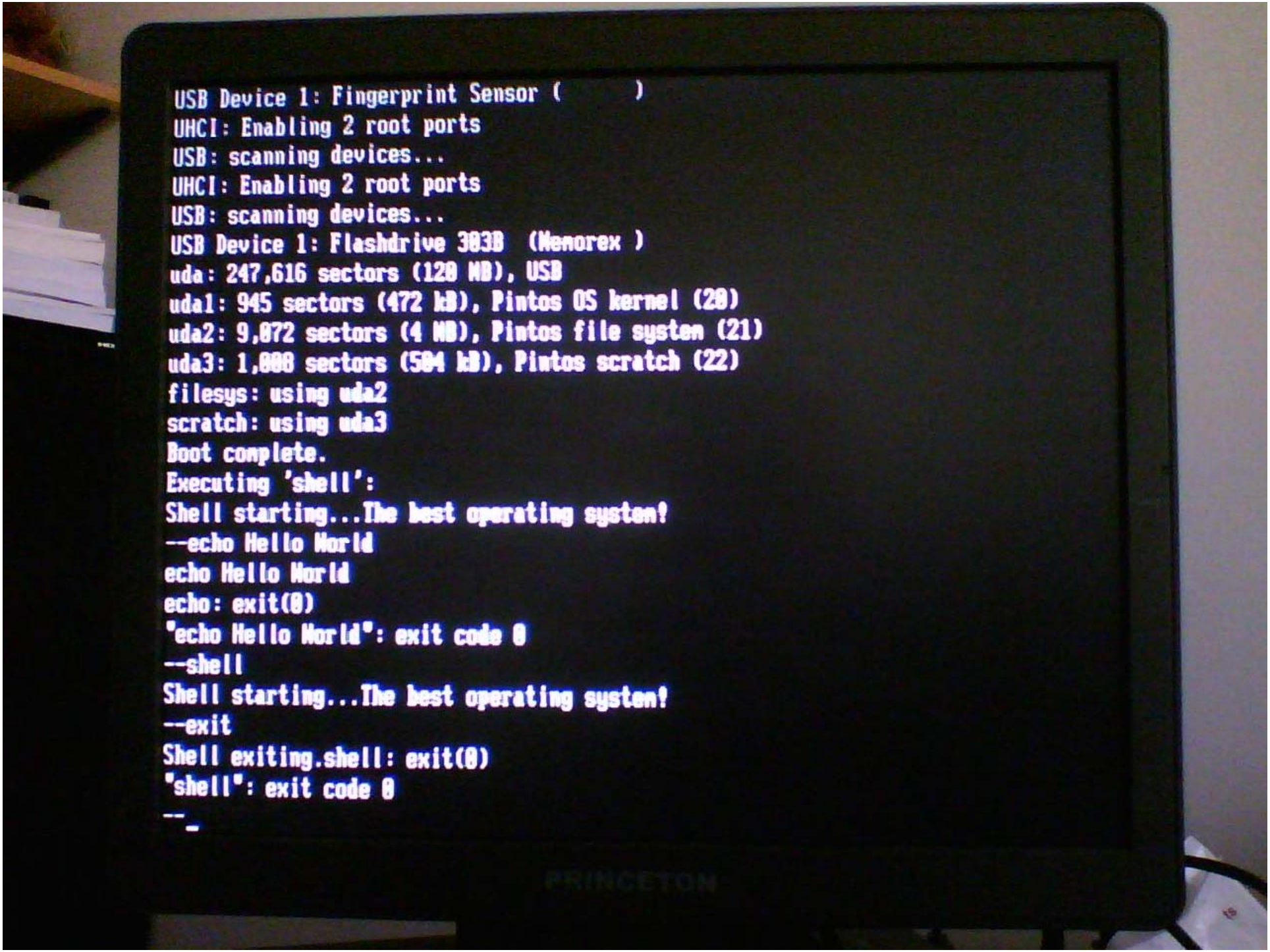
- Tool paper
- Series of 4 projects that provide backbone of lab component that accompanies OS course
- Suitable for Junior/Senior/1st Grad students
- Used by several institutions
  - Stanford (4 years+), Virginia Tech (3 years), University of San Francisco, William and Mary, University of Salzburg, Linköping Universitet, KAIST, Seoul National University, POSTECH

Teaching OS

- Internal Perspective
  - Teaches how an OS works from the inside, specifically, the kernel
  - Places students in the perspective of OS designer, rather than OS user
- Concrete Approach
  - Design and create realistic artifacts
  - Internalize abstractions by seeing concrete incarnation

Pintos Features

- Small enough so entire code can be read and understood by students
  - Unlike Linux or Windows
- Runs and debugs in simulated environment
  - 100% reproducible, like single-threaded user code
- Runs and debugs in emulated environment
  - facilitates non-intrusive analysis tools
- Runs on real hardware
  - boots on student's PC/laptop


USB Device 1: Fingerprint Sensor ( )
UHCI: Enabling 2 root ports
USB: scanning devices...
UHCI: Enabling 2 root ports
USB: scanning devices...
USB Device 1: Flashdrive 303B (Memorex)
uda: 247,616 sectors (120 MB), USB
uda1: 945 sectors (472 kB), Pintos OS kernel (20)
uda2: 9,072 sectors (4 MB), Pintos file system (21)
uda3: 1,008 sectors (504 kB), Pintos scratch (22)
filesys: using uda2
scratch: using uda3
Boot complete.
Executing 'shell':
Shell starting...The best operating system!
--echo Hello World
echo Hello World
echo: exit(0)
"echo Hello World": exit code 0
--shell
Shell starting...The best operating system!
--exit
Shell exiting.shell: exit(0)
"shell": exit code 0
--_

Project Principles (1)

- Read Before You Code
  - Provide well-documented code that serves as example of what we expect from students
  - Between 0-600 lines per project

Project Principles (2)

- Maximize Creative Freedom
  - Specify requirements
  - Don’t prescribe solution approaches

Project Principles (3)

- Practice Test-driven Development
  - All tests are public, reading tests makes requirements concrete
  - Student can add their own tests

Project	Functionality	Robustness	Regression
1	27		
2	41	35	
3	20	14	75
4	39	7	75

Project Principles (4)

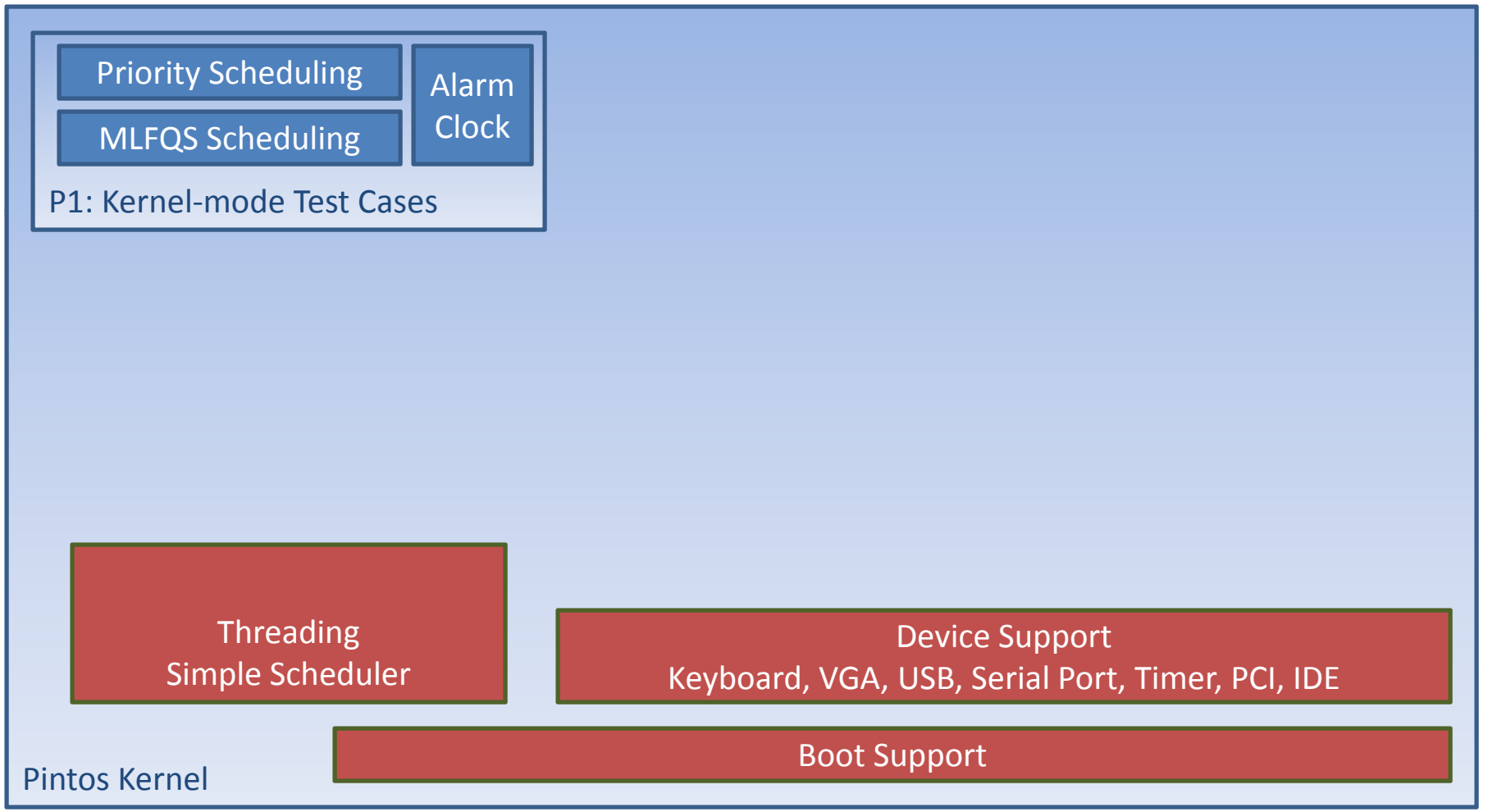
- Justify Your Design
  - Provide structured questionnaires that students use to describe and justify their design rationale

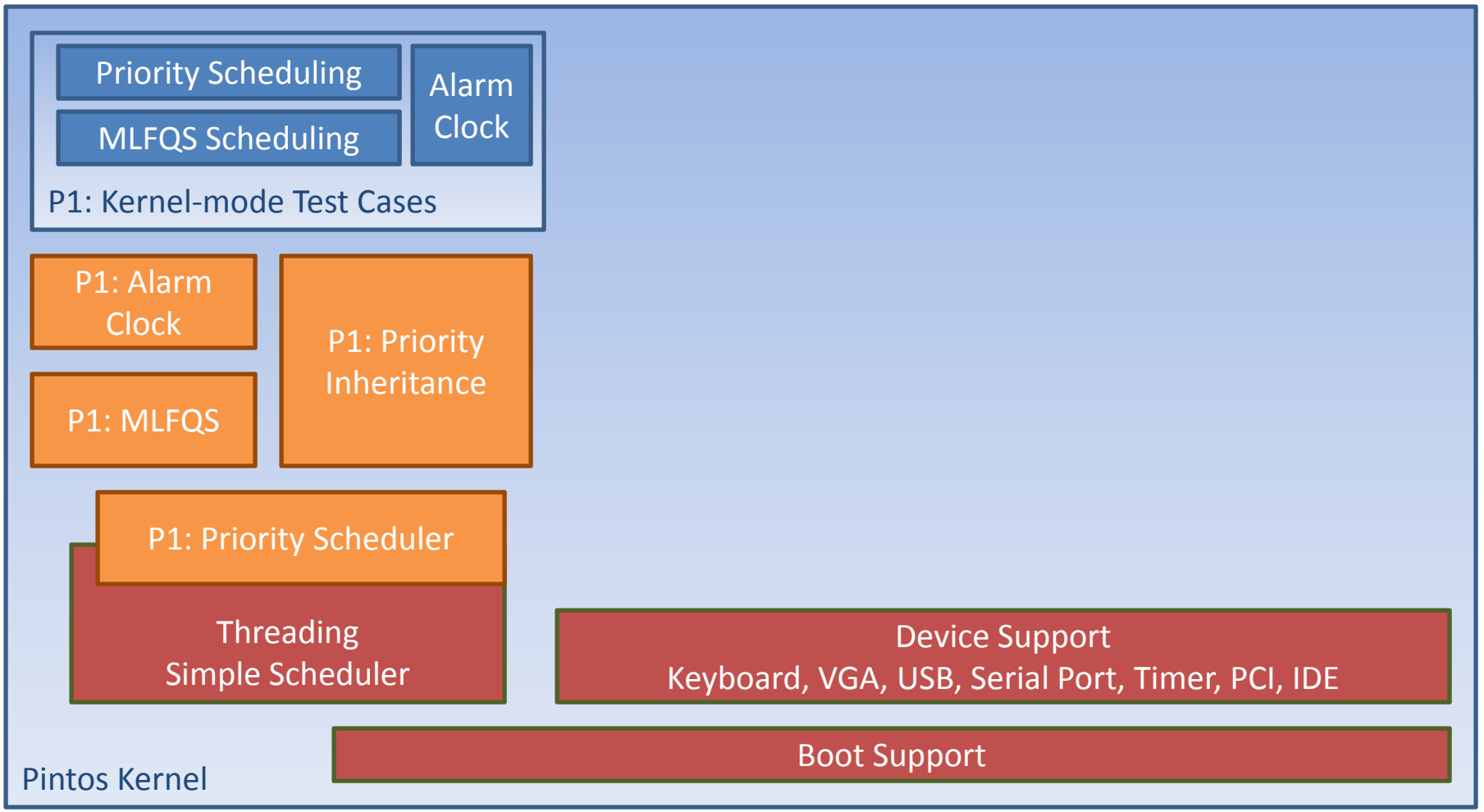
Project Principles (5)

- Work In Teams
  - 2-4 students
  - Allows for brainstorming and mutual support
  - Mimics industrial setting, e.g., use of shared source code repository and versioning
  - Design questionnaires still submitted individually

Pintos Project Themes

1. Threads
2. User Programs
3. Virtual Memory
4. File Systems


Priority Scheduling
MLFQS Scheduling
Alarm Clock
P1: Kernel-mode Test Cases
Threading
Simple Scheduler
Device Support
Keyboard, VGA, USB, Serial Port, Timer, PCI, IDE
Boot Support
Pintos Kernel
Support Code
Public Tests


Priority Scheduling
MLFQS Scheduling
Alarm Clock
P1: Kernel-mode Test Cases
P1: Alarm Clock
P1: MLFQS
P1: Priority Inheritance
P1: Priority Scheduler
Threading
Simple Scheduler
Device Support
Keyboard, VGA, USB, Serial Port, Timer, PCI, IDE
Boot Support
Pintos Kernel
Support Code
Students Create
Public Tests

Example of Project 1 Test

```
void
test_priority_change (void)
{
  msg ("Creating a high-priority thread 2.");
  thread_create ("thread 2", PRI_DEFAULT + 1, changing_thread, NULL);
  msg ("Thread 2 should have just lowered its priority.");
  thread_set_priority (PRI_DEFAULT - 2);
  msg ("Thread 2 should have just exited.");
}

static void
changing_thread (void *aux UNUSED)
{
  msg ("Thread 2 now lowering priority.");
  thread_set_priority (PRI_DEFAULT - 1);
  msg ("Thread 2 exiting.");
}
```

Expected output:

Creating a high-priority thread 2.
Thread 2 now lowering priority.
Thread 2 should have just lowered its priority.
Thread 2 exiting.
Thread 2 should have just exited.

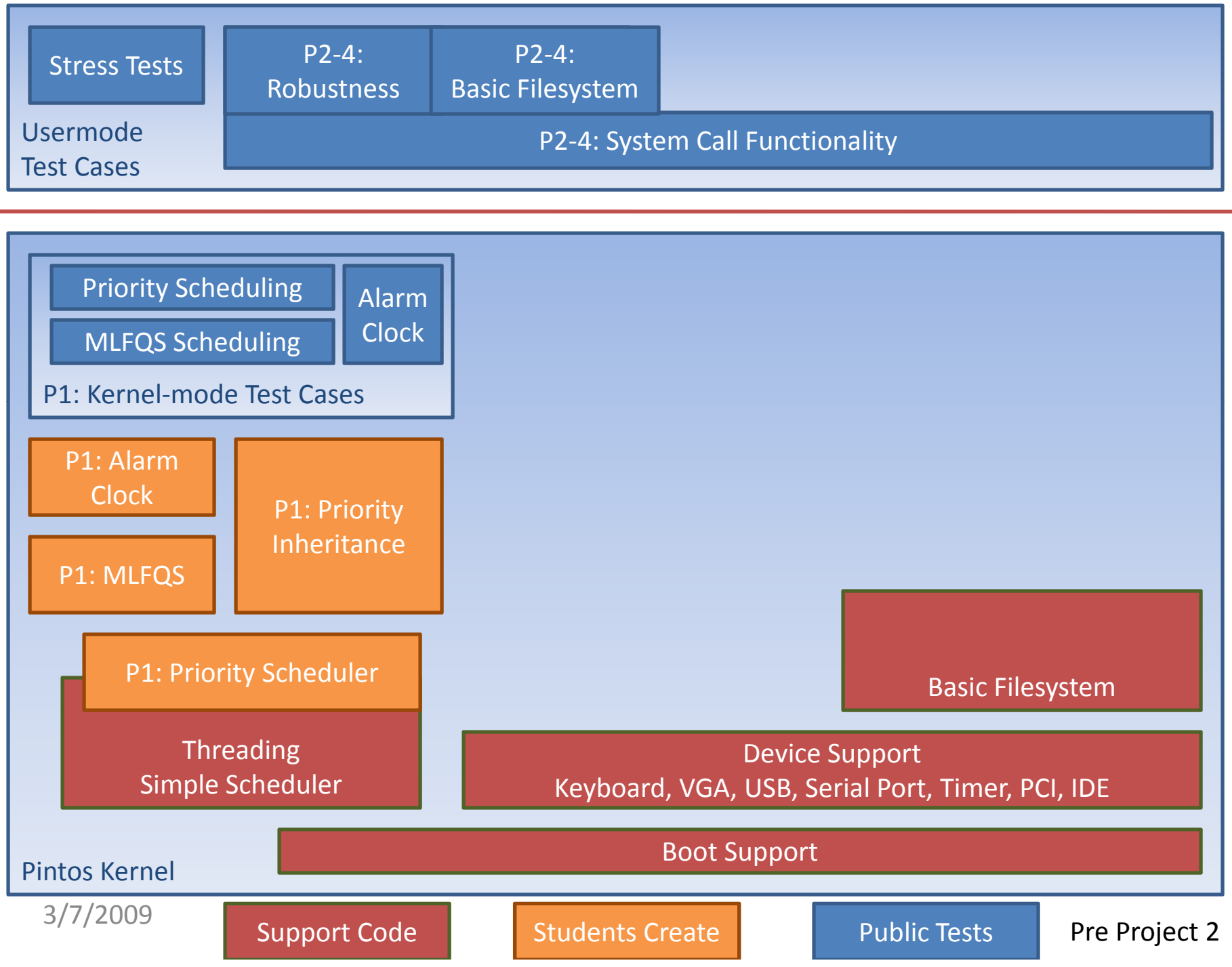
make grade (1)

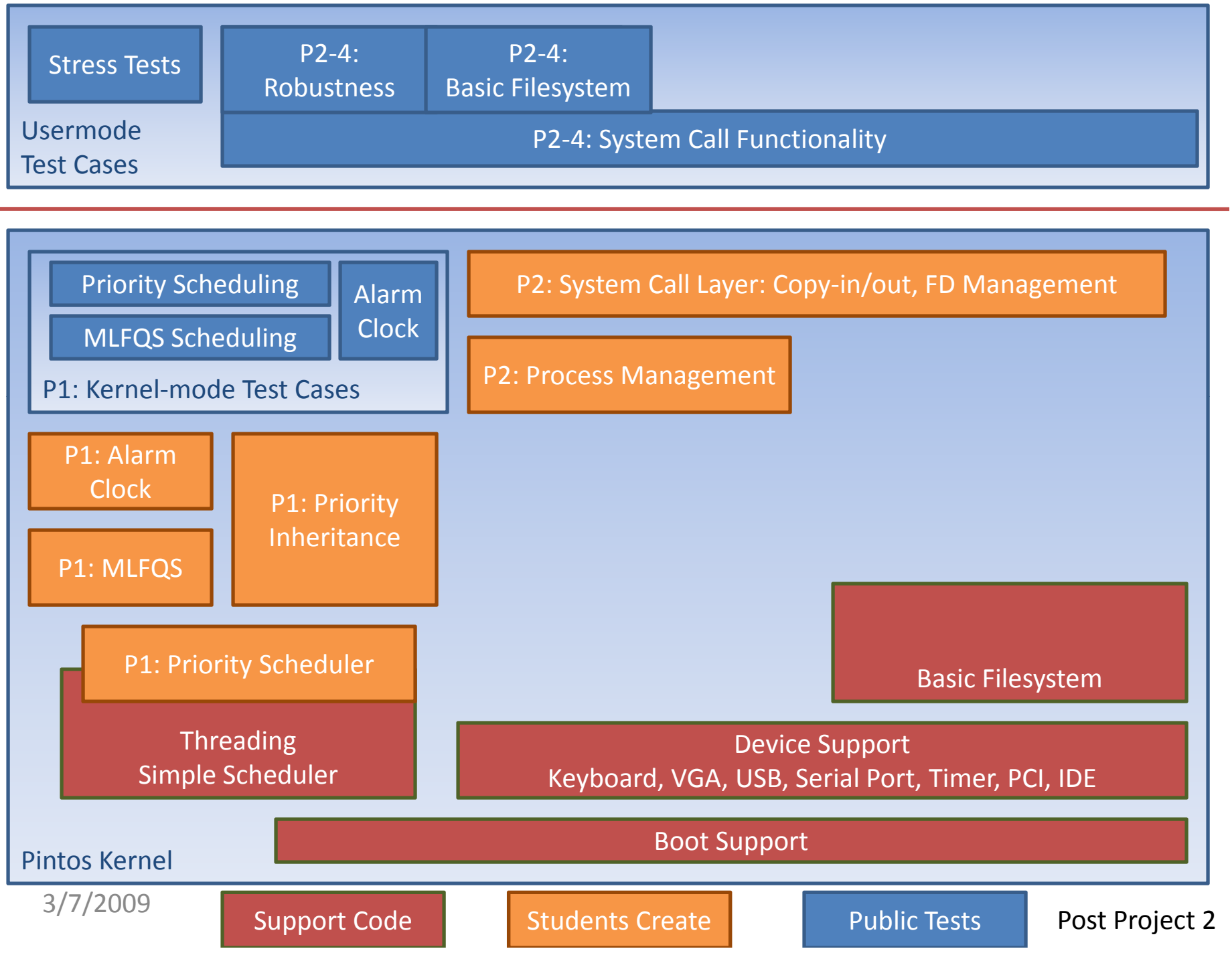
```
TOTAL TESTING SCORE: 100.0%
ALL TESTED PASSED -- PERFECT SCORE

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

SUMMARY BY TEST SET

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

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

Pintos include fully automated grading scripts, students see score before submission

make grade (2)

```
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
```


Stress Tests
P2-4: Robustness
P2-4: Basic Filesystem
Usermode Test Cases
P2-4: System Call Functionality
Priority Scheduling
Alarm Clock
MLFQS Scheduling
P1: Kernel-mode Test Cases
P1: Alarm Clock
P1: Priority Inheritance
P1: MLFQS
P1: Priority Scheduler
Basic Filesystem
Threading
Simple Scheduler
Device Support
Keyboard, VGA, USB, Serial Port, Timer, PCI, IDE
Boot Support
Pintos Kernel



Support Code | Students Create | Public Tests

Project 2 Functionality Test

```
/* This program echoes its command-line arguments */
int
main (int argc, char *argv[])
{
  int i;
  msg ("begin");
  msg ("argc = %d", argc);
  for (i = 0; i <= argc; i++)
    if (argv[i] != NULL)
      msg ("argv[%d] = '%s'", i, argv[i]);
    else
      msg ("argv[%d] = null", i);
  msg ("end");
  return 0;
}
```

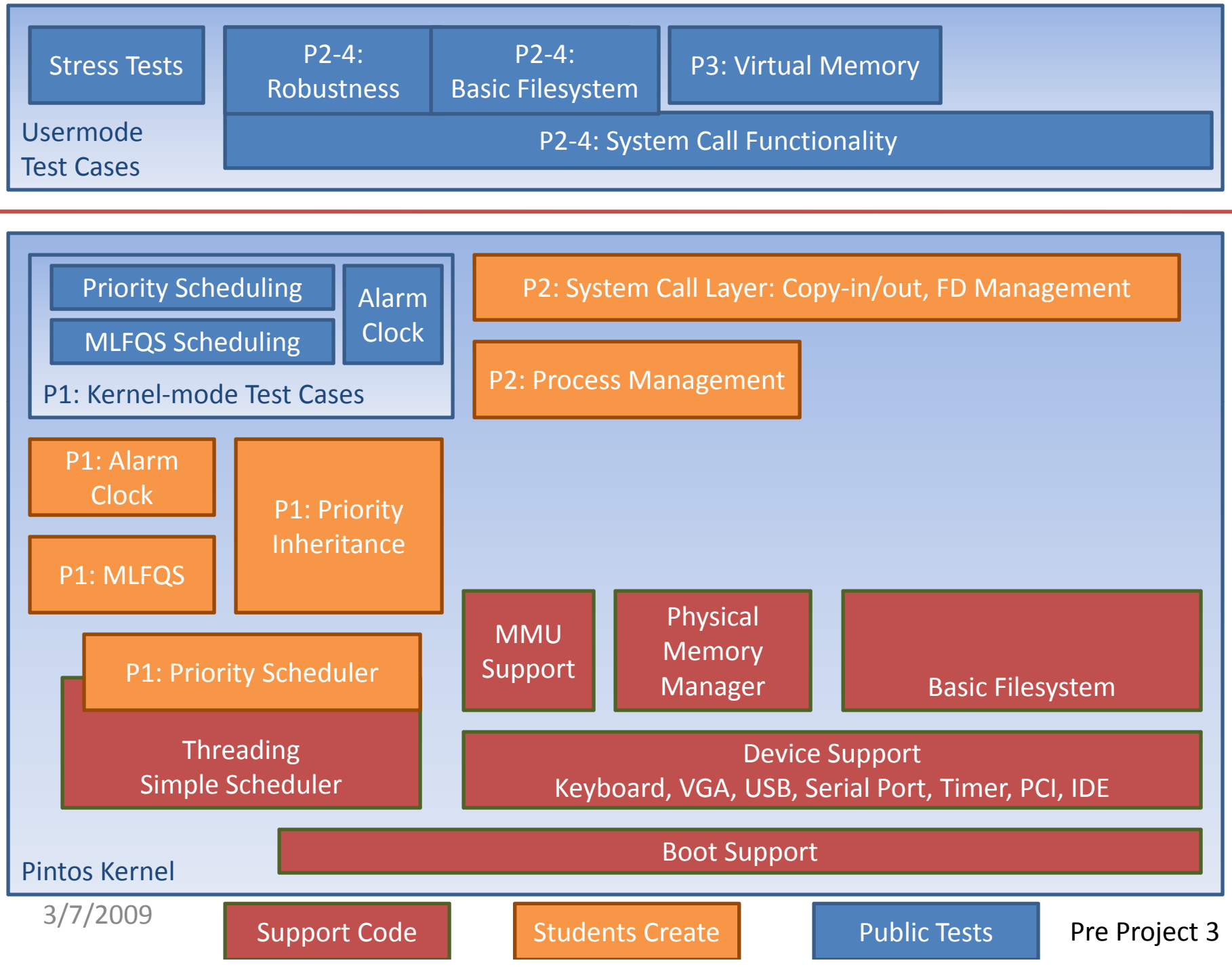
Expected output for 'args 1 2'

begin
argc=3
argv[0] = 'args'
argv[1] = '1'
argv[2] = '2'
argv[3] = null
end

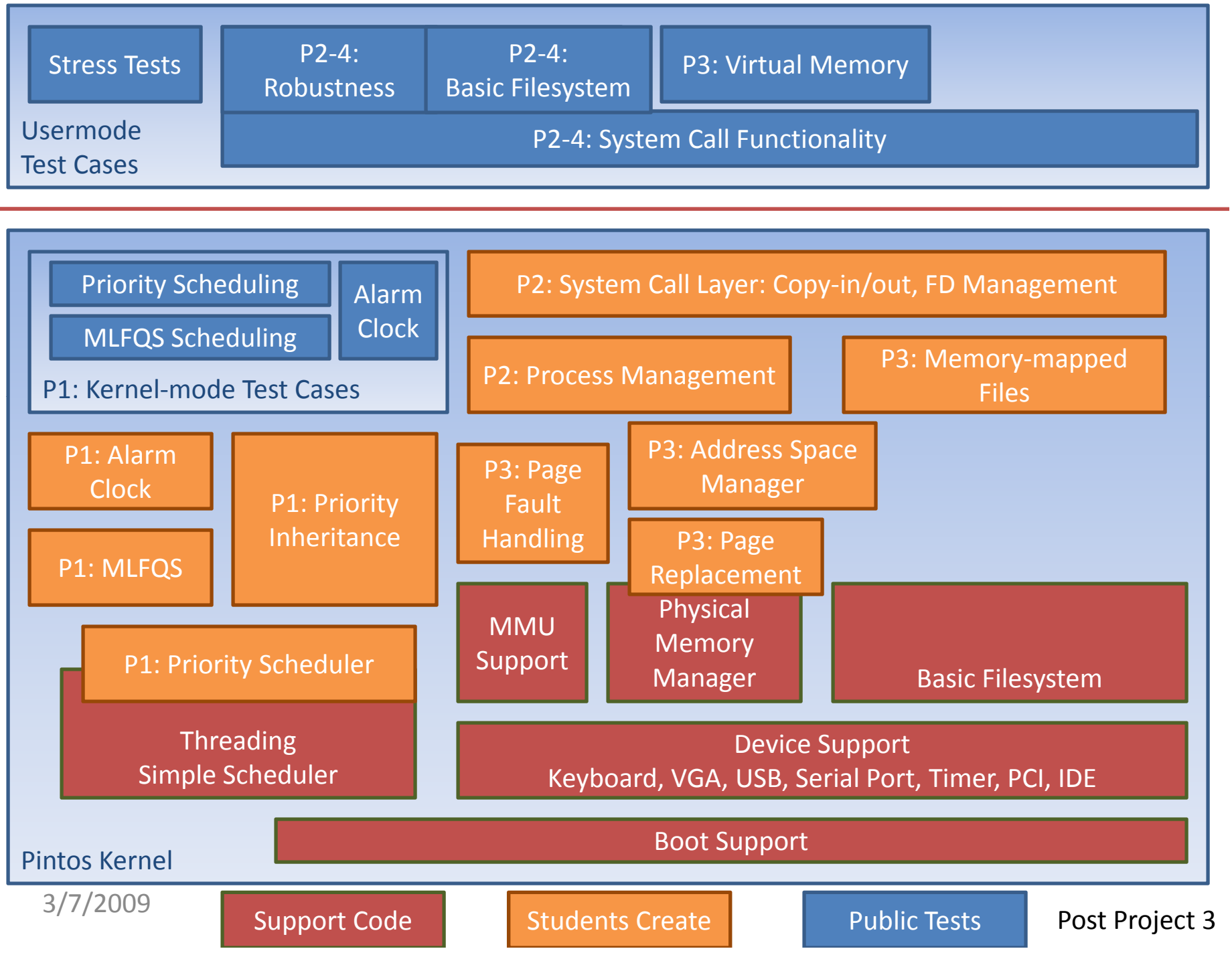
Project 2 Robustness Test

```
/* This program attempts to read memory at an address that is not mapped.
   This should terminate the process with a -1 exit code. */

#include "tests/lib.h"
#include "tests/main.h"

void
test_main (void)
{
  msg ("Congratulations - you have successfully dereferenced NULL: %d",
       *(int *)NULL);
  fail ("should have exited with -1");
}
```

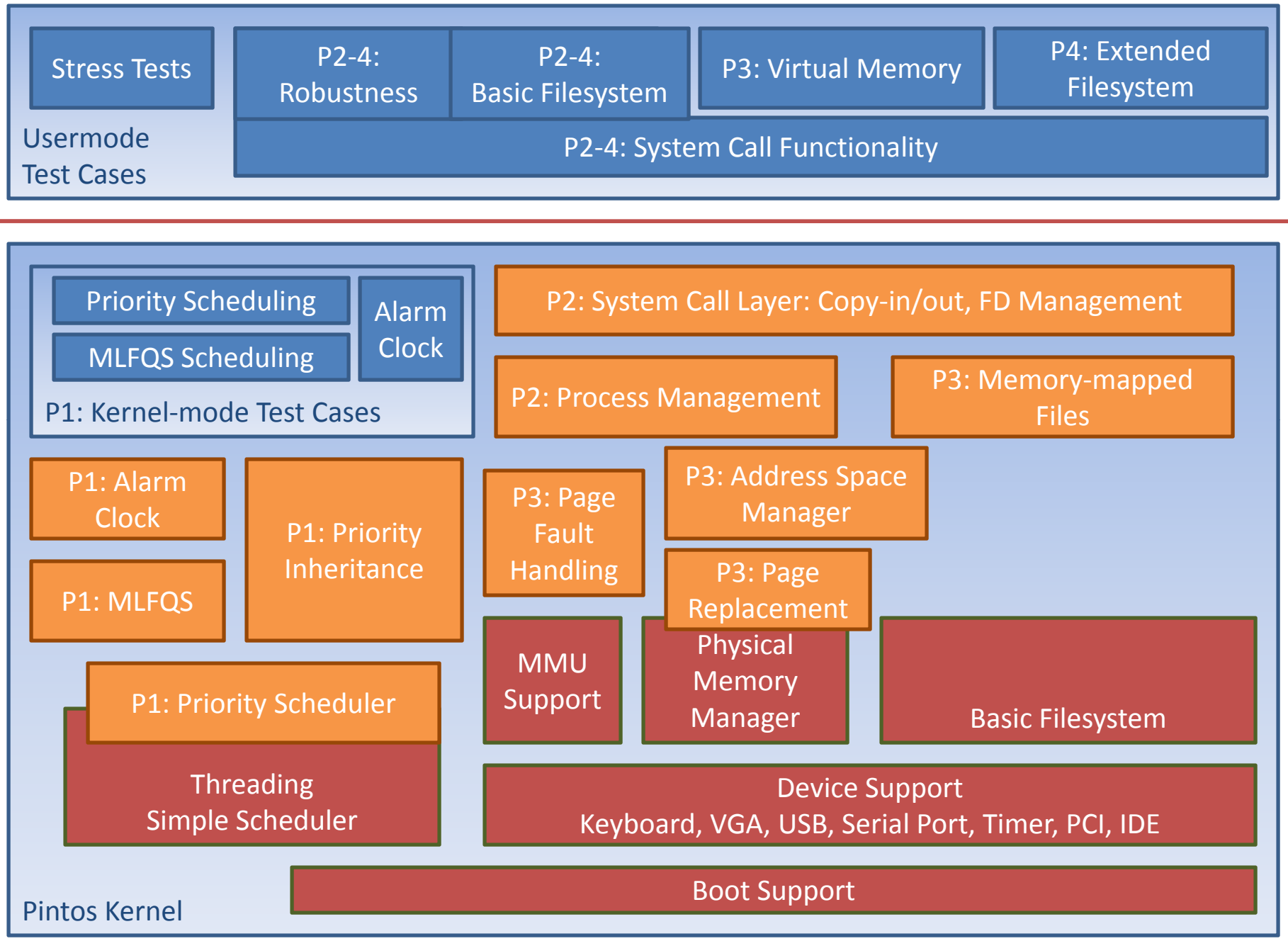
Expected output:
bad-read: exit(-1)

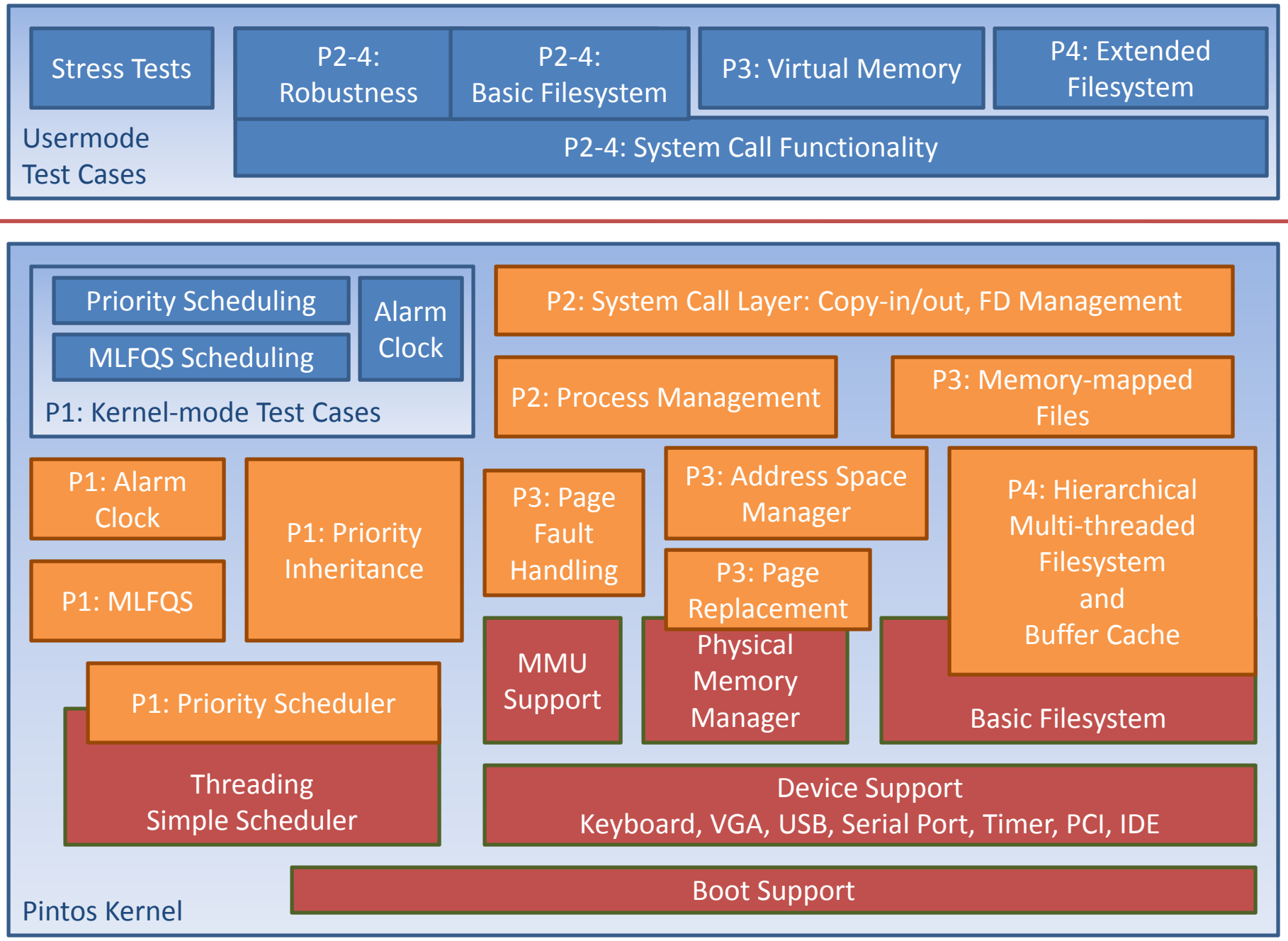


Support Code | Students Create | Public Tests



Support Code | Students Create | Public Tests

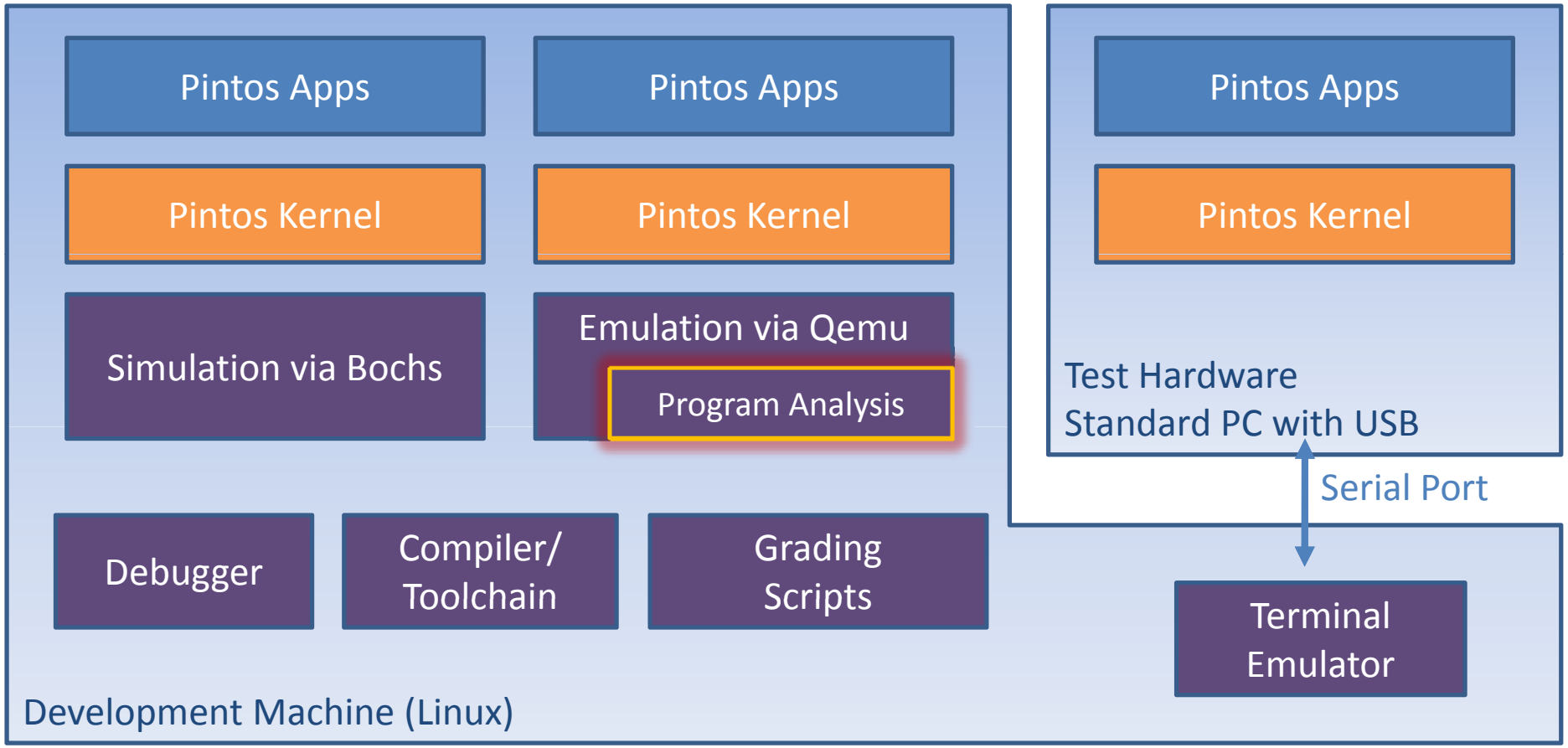

Stress Tests
P2-4: Robustness
P2-4: Basic Filesystem
P3: Virtual Memory
P4: Extended Filesystem
Usermode Test Cases
P2-4: System Call Functionality
Priority Scheduling
MLFQS Scheduling
Alarm Clock
P1: Kernel-mode Test Cases
P2: System Call Layer: Copy-in/out, FD Management
P2: Process Management
P3: Memory-mapped Files
P1: Alarm Clock
P1: Priority Inheritance
P3: Page Fault Handling
P3: Address Space Manager
P1: MLFQS
P3: Page Replacement
MMU Support
Physical Memory Manager
Basic Filesystem
P1: Priority Scheduler
Threading Simple Scheduler
Device Support Keyboard, VGA, USB, Serial Port, Timer, PCI, IDE
Boot Support
Pintos Kernel



Support Code | Students Create | Public Tests

Post Project 4

Program Analysis



Race Detection Example

```
*** Race #1 ***
- Fault Point -
  IP: c002da7d
  Function: list_begin
  Memory address at which race occurred: c003afc4
  Memory base of object in which race occurred: c003afc0
  This race affects global variable: open_inodes
  Lockset:

- Threads involved in race -

  * Backtrace (thread #1) *
     list_remove (c002d565)(lib/kernel/list.c:260)
     inode_close (c0032c1f)(filesys/inode.c:177)
     file_close (c0032224)(filesys/file.c:52)
     syscall_handler (c003175c)(userprog/syscall.c:288)
     intr_handler (c0021f47)(threads/interrupt.c:377)
     ??? (c0022107)(../../threads/intr-stubs.S:38)

  * Lockset (thread #1) *
```

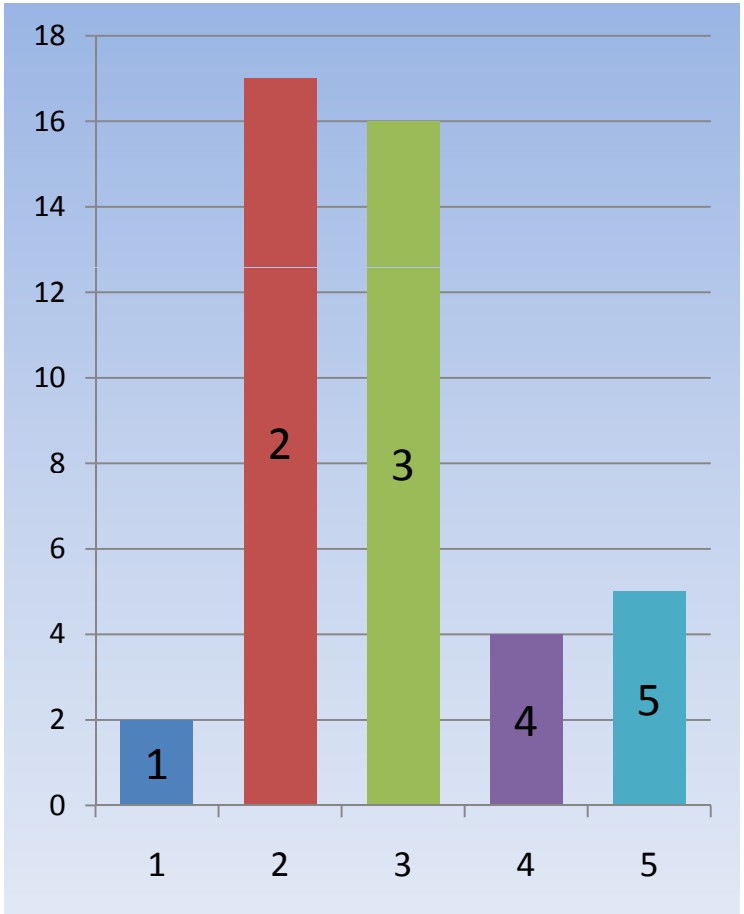
In this example, students forgot to protect the list of open inodes, which is accessed concurrently by an exiting process (left backtrace) trying to close its files and a starting process (right backtrace) trying to open and read its executable

```
  * Backtrace (thread #2) *
     list_begin (c002da7d)(lib/kernel/list.c:74)
     inode_open (c0032c83)(filesys/inode.c:118)
     dir_open_root (c00327c0)(filesys/directory.c:57)
     filesys_open (c0031b27)(filesys/filesys.c:69)
     start_process (c002f7fb)(userprog/process.c:358)
     kernel_thread (c002170f)(threads/thread.c:538)

  * Lockset (thread #2) *
```

Evaluation (Fall 2008)

- How confident are you in your ability to understand the output of the race condition checker?

1. Not at all confident, the output was very confusing. (2/44)
2. I sort of understood what it was trying to tell me, but my understanding was vague. (17/44)
3. After careful analysis of the output, I understood the causes leading to the displayed race and was able to fix it. (16/44)
4. Once I learned the general format of the output, I quickly found the underlying race condition that was flagged. (4/44)
5. No answer (5/44)



- Based on survey given during final exam
- In addition, more than 50% of students reported that the race condition checker helped them find actual bugs that made them pass project tests!

Setting Up Pintos

- Requires simple Linux server
  - 1 quad core machine can support 8-10 students easily
  - All work can be done using remote ssh access, or an IDE can be used
  - No root user access required
  - Uses mostly host tools (gcc, binutils) and packages (bochs, qemu)
- Includes texinfo manual (HTML, 129-page PDF)
  - Documentation separates generic and institution-specific parts in separate files, e.g.

    Stanford: @set coursenumber CS140

    Virginia Tech: @set coursenumber CS 3204

Placement in Curriculum

- Cannot be a first course in C
- Should probably be 4th or 5th programming course
- Can be a first or second course in OS
- Pintos projects can stretch over 10-15 weeks
- Satisfies a “deep design” requirement

Related Work

- Systems that provide internal kernel perspective
- Simulated architecture only:
  - Nachos, ToyOS, OS/161, Yalnix
- Emulated:
  - GeekOS, JOS
- Real hardware:
  - GeekOS, Xinu, PortOS, JOS, Minix, Windows CRK, adapted versions of Linux

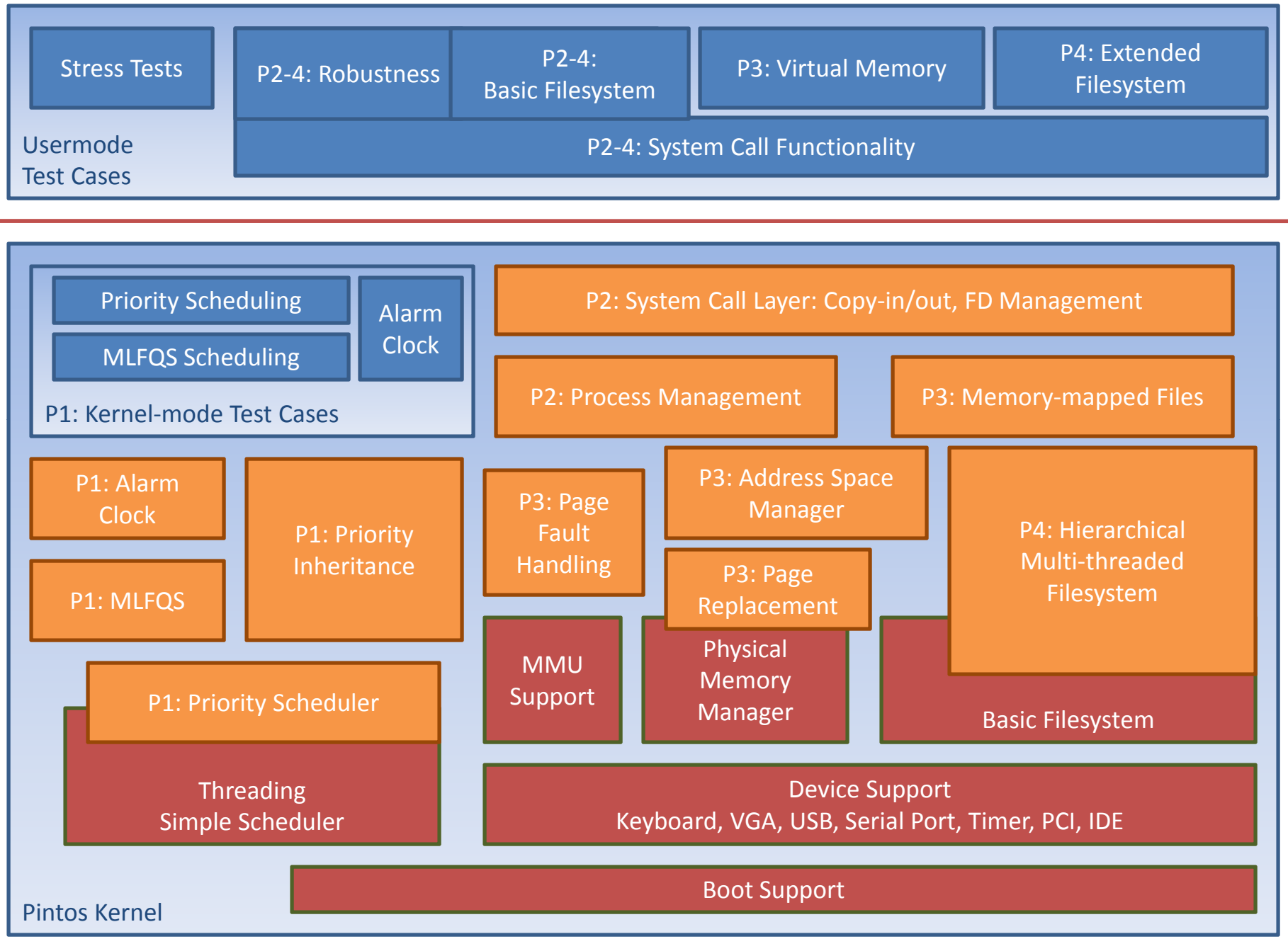
Future Work

- Educational:
  - Introduce modular assignment structure to allow instructor to tailor assignments with reduced or varied scope
  - Integrate assessment tools
  - Integrate static analysis tools
  - Integrate performance measures
- Technological:
  - Introduce multi-core/multi-processor support

Thank You!

- Ben Pfaff
- Anthony Romano
- Godmar Back
- Many Instructors, TA's, and students who have contributed with tests and suggestions

- URL: www.pintos-os.org
- Mailing list: pintos-os@googlegroups.com



Support Code | Students Create | Public Tests

Post Project 1