About project demonstration

Hi all,

During the demonstration, TAs will ask project-related questions to verify that you know your implementation. Understanding how the system works is necessary for you to implement the project. Lack of evidence showing that you understand the project would result in score deduction of up to 100% points.

However, if you fail your first show, you can make up. You can show again during next lab session and office hour, or answer questions in a separate report. If the second show or report provide enough evidence, you only lose 10% for the first show failure.

Potential questions for project 2:

1) Show how the sleep function works. You need to specify how the timerHandler is called or how virtual timer wakes up.

2) The difference between virtual timer and physical timer.

3) IRQ responding sequence. Show the corresponding codes for vector table, first level IRQ handler, physical timer handler. Show how the system serves timer event, and how virtual timer scheme works.

4) Show how the CPU saves and restore registers when switching between USER mode and IRQ mode.

5) Show how the scheduler knows whether there's some READY task in the task queue.

If you fail the first show, you can either show again or answer these questions in a separate report.

Potential questions for project 3:

1) Show how your codes guide the CPU to execute the 2nd level handler, when a software interrupt happens.

2) Show where and how to invoke a software interrupt. Show where and how to extract the SVC number in your program.

3) Show how you pass parameters from pi_libc functions to 1st level handler; show what parameters you pass from 1st level handler to 2nd level handler; Explain why 2nd level handler can get all incoming parameters. (Hint: this has to do with the way how arm-none-eabi-gcc compile your C functions; you may need to check assembly codes generated in kernel7.lst)

4) For exit() function, show how you make the CPU switch to scheduler task when it returns from 1st level handler. (Hint: for exit() and sleep(), after handling interrupt, the CPU won't return to user task where interrupt happens; instead, it should return to scheduler task)

5) For sleep() function, show where and how you save user task context (the CPU registers) before triggering a software interrupt. (Hint: You need to save context before triggering a software, and you need to store this context into usertask->regs at some time. So next time when scheduler yields to this user task, it restarts where interrupt happens.)

If you fail the first show, you can either show again or answer these questions in a separate report. For project 3, you lose 15% for the first show failure.

Potential questions for project 4:

1) Explain the relationship between "content on SD card" and "FAT fs"; show how FAT & Partition System read file system metadata info from SD card. (hint: you need to understand operations when mounting a file system, track the code
"Kernel.vfsInitialize()" in main.cpp)

2) Find the code where fatfs.cpp writes something to SD card. Explain the involved process. (hint: FAT write do some checkings, locate the cluster and sector, then calls SD card driver in emmc.cpp ...)

3) For vfsOpen, show how to handle different situations. (hint: for whether file is already opened in inode_list, there're two cases; for permit, two cases; for whether file exists on SD card, two cases. So there can be at most 2*2*2 situations, although some of them are Don't Care cases. Make sure your codes handle them properly.)

4) For vfsRead and vfsWrite, explain how to check "size" against "file_zise", "file_offset", "MAXIMUM_BUFFER_LEN".

5) Show your implementation of the hash algorithm. Explain how it handles collision. Explain how it accelerates your VFS.

If you fail the first show, you can either show again or answer these questions in a separate report. For project 4, you lose 15% for the first show failure.

Potential questions for project 5:

1) Explain the differences among process, thread, CPU core;

2) Explain what happens after fork(); How to tell between parent process and child process

3) Which fields in task_struct are different, for parent process and child process? (Hint: read Docs and links carefully. You need to know how 15 registers function in ARM32.)

4) What aspects do we need to check when a process exit()? (Hint: It may have child or parent process)

5) What aspects do we need to check when a process wait()? (Hint: It may have child or parent process)

If you fail the first show, you can either show again or answer these questions in a separate report. For project 5, you lose 15% for the first show failure.

Questions for project 6:

1) List one advantage & one disadvantage for Shared Memory;
List one advantage & one disadvantage for Message Queue;

2) Tell the behavior differences between blocking receive and non-blocking receive; List at least one advantageous working scenario for each.

3) The implementation of message queue in project 6 is not efficient. Think out one method to improve efficiency and briefly explain it. (In less than 80 words)

There's no demo for project 6. You need to answer all 3 questions in your report.