ECE531: Advanced Operating Systems – Final Project

Fall 2023

Due: Friday, 15 December 2023 5PM (Last day of Finals)

Overview:

- Design a project that extends our custom operating system.

Guidelines:

- You may work either alone or in group of two or three. If you work in a group your end project will have higher expectations.
- You probably will want to use one of the homeworks as a starting point.
- It is required that you write some code for this project.
  - It is fine if you use code from outside sources (as long as you note this) but it is expected that there will be a non-trivial amount of code you wrote yourself.
  - One exception is if your project involves taking an existing piece of code (say an existing USB driver for example) and adapting it so that it will run on the 531 OS. In this case the work might involve a lot of small changes plus a lot of build system work. This is probably fine but be sure this is addressed when you pick the topic.

Part 1: Topic Selection (due ~~31 October~~ 3 November 2023) (5pts)

Each group should send a brief e-mail describing your project topic and listing group members.

Part 2: Progress Report (due 16 November 2023) (10pts)

This is primarily to make sure your project is on track to be finished in time; if things are not going well the topic can be adjusted.

Give a brief status update, as well as provide some related work for your project. Do a literature search and find examples of other people who have done similar research. It is OK if you find a lot of related work, or even if someone has done the exact same thing before. This is a class project (not a PhD thesis) so reproducing something that has been done before is perfectly fine.

Since this is a grad course it would be great if you could find academic papers as sources, but it's more likely you will just find personal websites or blogs. That's fine. Sometimes you might find an interesting paper via Google, but you will not be able to find a free copy online (just a site asking you to pay money). If it's a journal like the IEEE or ACM you can log into the UMaine library and get copies of the articles for free. If you're on an on-campus internet connection (ending in .maine.edu) you can go to the UMaine library homepage, search for IEEE or ACM there, and it will give you a link to click through to get to those sites (IEEE explore or similar) and you will then be able to find and download those papers.

Send this report by e-mail. Only one submission is needed per group.

What to include in the update:

1. The list of group members.
2. A brief (one or two sentence) description of the project.
3. The related work
   (a) Have at least two items of related work for each group member
   (b) Cite the works properly
   (c) An example is included below.
4. Give a brief status on the work. Do you need help with anything? Do you need to change your topic?
5. Finally: please say which day you'd like to present (Tuesday or Thursday). There's a small amount of extra credit for going on the earlier day. Note that this is mostly to help me plan the schedule, if you need to change the date for whatever reason as it gets closer I can usually accommodate that

A short example of roughly what I expect for the related work:

Our project is a chiptune music player running on top of our OS.

Related Work:
Our research involves extending our OS enough that it can drive a chiptune music player. This involves implementing i2c support for driving some displays, and SPI support for talking to some custom music player hardware. Weaver[1] wrote a Raspberry Pi based chiptune player, but unlike our project it runs on top of Linux. Mallow and Snap[2] also wrote a music player, but it runs on STM32L hardware without an operating system and has to emulate the chiptune chip in software.

[1] V. Weaver. "An AY-3-8910 Chiptune Player for Raspberry Pi." Proc. of the 4th Conference on Unlikely Pi Uses, p 10-18, May 2013.

[2] M. Mallow and G. Snap. "Playing Chiptunes on ECE271 Hardware." Journal of Embedded Programming, p11-19, Vol 1 Issue 15, June 2015.

Part 3: In-Class Presentation 5 and 7 December 2023 (40pts)

- You will have 10 minutes to present. Plan for 8 minutes of presenting plus 2 minutes for questions. (Note: these times are tentative until we finalize the numbers of groups.)
- You may present slides using the projector if you want, but that's not strictly necessary.
- You should cover the following things (but feel free to include more):
  - Intro/Overview: what you did and why
  - Brief Related Work: any similar projects, how your project differs (it's OK if it doesn't)
  - Hardware: describe any hardware used in the project
    If it's interfaced via built-in BCM2835 hardware briefly describe that interface too.

- Software: describe the software you wrote and how it ties into an operating system (is it a device driver? is it userspace and talks via syscalls? Does it bypass the OS? is it a proof-of-concept outside the OS?)
- Challenges: Describe any challenges you encountered
- Future Work: if you had more time, what else would you do
- Demo: show off what you did, if possible

Part 4: Project Writeup, Officially due 15 December 2023 (45pts)

This will be a short paper (at least 6 pages, but you can include pictures, diagrams, etc.). Please use the IEEE Conference paper format if possible `https://www.ieee.org/conferences/publishing/templates.html`

Your paper should contain the following sections:

1. **Abstract**: 100 words or so brief intro to what you did.
2. **Introduction**: What your project is and what the goal was.
3. **Related work**: List and properly cite related projects (this should be based on what you submitted as part of the status update). Not only cite them, but for each have at least a 1-line summary of what the work did and how it compares to your project (and it's fine if they did essentially the same thing).
4. **Setup**:
   (a) Hardware Interface (if applicable): Describe any low-level hardware involved in your project, either part of the Pi itself or anything you attached. Be sure to list any websites / data sheets you used, and briefly describe how the interface works.
   (b) Software Implementation: Briefly describe your code. Describe the operating system interface needed to use the code (is it a device driver, is it userspace accessed by system call, userspace doing direct access, is it a proof of concept running fully in userspace, etc).
5. **Results**:
   (a) Did the project work? What did you accomplish? It's OK if the project didn't work, in that case just describe what you attempted and what went wrong.
6. **Future Work**: List any improvements you might make if you had more time and resources to work on the project.
7. **Conclusion**:
   (a) Summarize what you did and what you accomplished.
   (b) If you worked in a group: List who worked on what part.
8. **Appendix / Code**:

(a) I would like a copy of any relevant source code if possible. (this can be submitted as a separate file, does not have to be included in the report).

(b) Reminder: unless agreed to otherwise during topic selection, it is expected that some non-trivial amount of code was written by your group as part of the project

You can e-mail your final report to me. pdf or word document is fine, the code should be attached too.

Potential Project Ideas:

These are just suggestions, feel free to come up with anything else that might be relevant.
Note, the difficulty level is my best guess, things might be way harder or easier than I estimate.

- Easy
  - Some sort of video game that runs in your OS (this could involve improving the kernel as well as writing userspace library code)
  - Read from the Pi's temperature sensor, do something useful with the results (this might be harder on Pi4)
  - Read from the Pi's random number generator, do something useful with the results (this might be harder on Pi4)
- Medium
  - Drive an i2c display
  - Drive some SPI hardware
  - Drive some 1-wire hardware
  - Optimize the memcpy() and other string routines to be as fast as possible and benchmark them (in assembly language?)
  - Read values from a PS/2 mouse (I have an adapter if you want to try this)
  - Create an interface for the hardware performance counters
- Hard
  - Play sound output through the PWM/sound interface
  - Write a FAT filesystem driver
  - Write a driver for some other filesystem
  - Write a driver for the SD card interface
  - Write a more advanced scheduler
- Really Hard
  - Port our operating system to work in 64-bit mode
  - Implement full virtual memory
  - Get the ethernet card working
  - Get USB working

  - Read from the Pi-cam or other webcam
  - Generate 3D graphics using the GPU
  - Experiment with replacing the boot firmware

Past Projects:

- FAT filesystem
- Temperature and RNG
- PWM Audio
- PS/2 Mouse input
- PS/2 Keyboard input
- i2c driver
- REPL (programming language) shell
- Graphical Framebuffer Game
- Virtual Memory / Memory Protection