Virtual Memory and RomFS

ECE531: Advanced Operating Systems – Homework 7

Fall 2025

Due: Friday, 21 November 2025, 5:00pm

This homework involves virtual memory and caches.

1. **Download the homework code template**

   - Download the code from:
     `https://web.eece.maine.edu/~vweaver/classes/ece531/ece531_hw7_code.tar.gz`
   - Uncompress the code. On Linux or Mac you can just
     `tar -xzvf ece531_hw7_code.tar.gz`

2. **Investigate virtual memory support (2pts)**

   - The provided code implements some very simple ARM1176 virtual memory support. It sets up a large, 1:1 "sections" mapping, so the virtual and physical addresses are the same, with 1MB page granularity (4096 page table entries)
   - Look at the code in `./kernel/memory/arm1176-mmu.c` for all of the details. It's a lot of low-level setting up of ARM special registers.
   - The code as provided does not provide memory protection support. Try running the provided command `cause_error kernelread`
     which will try to read operating system memory. Then try running `cause_error kernelwrite` which will try to overwrite your operating system with garbage.
   - Now, edit the file `./kernel/memory/memory.c` and uncomment the `enable_mmu(1);` line that is commented out. Rebuild, reboot. At bootup you should get some extra boot messages about the MMU.
   - Now try running `cause_error kernelwrite` again. Also try `cause_error kernelread` and `cause_error kernelexec`
   - **Question: What happened when you ran the test without virtual memory? Did enabling virtual memory change the behavior?**

3. **Virtual Memory Questions (3pts)**

   Answer these questions in the README file:

   (a) Name one feature found in Linux/UNIX that is made possible (or easier) because of virtual memory.

   (b) What important piece of logic on a modern processor hides a lot of the overhead of page table lookups?

   (c) What is it called when a memory access happens, the CPU walks the page tables, but the virtual memory page requested is not found in the page table? Whose job is it to handle this now that the CPU has given up?

(d) The operating system has found the page requested by the CPU, but it tries to allocate space for it in physical memory but physical memory is full. How can more room be freed up in physical memory?

4. **Investigate the romfs filesystem**

   - The filesystem we are using is called ROMFS. You can read the documentation on it here: `https://www.kernel.org/doc/Documentation/filesystems/romfs.txt`
   - We have no disk support, so the romfs is loaded as a ramdisk at boot time (it is included as a binary blob during the kernel build). All block reads/writes are turned into RAM accesses with the proper offset. The code for this is simple, and can be found in: `kernel/drivers/ramdisk/ramdisk.c`
   - The romfs filesystem support is in `kernel/fs/romfs/romfs.c`
     The important functions are:
     - `romfs_get_inode()` – given a filename, finds the inode for it
     - `romfs_stat()` – returns metadata from a file
     - `romfs_mount()` – mounts the filesystem
     - `romfs_statfs()` – the statfs system call, reports disk usage for tools like "df"
     - `romfs_getdents()` – returns the directory entry info

5. **Modify values returned by the filesystem (3pts)**

   (a) When you run `ls -l` (lowercase L) it lists the files including some of the metadata. You'll notice that romfs does not store file times, so the romfs code has to make up an offset from the UNIX epoch (1 Jan 1970). Currently it is showing a date in 1978.

   Modify the `romfs_read_inode()` call so that it shows a different modify time when doing `ls -l`. Change the value being set in inode->ctime, inode->mtime, and inode->atime.

   (b) You can run `df` to see how much disk space is available. For our system the ramdisk is small and for romfs it is always full as it would be silly to have empty space on a read-only filesystem.
   Modify the `romfs_statfs()` routine so that it lies and reports the ramdisk size as being 1GB and 50% full. (hint, look at the blocks and bfree values).

6. **Device / Filesystem Questions (2pts)**

   Answer these questions in the README file:

   (a) What is the purpose of a filesystem?

   (b) Would the device driver for a SD-card be a character or block device. Explain why.

7. **Submit your work**

   - Run `make submit` in your code directory and it should make a file called `hw7_submit.tar.gz`. E-mail that file to me as well as the document with the answers to the questions.