COMP3301 Assignment 3

OpenBSD VHD Kernel Driver - Filing the System

Due: 3pm Monday in Week 13 (21st of October)
Submission: Git
Code submission is marked in your prac session in week 13
Last Updated: October 9, 2024

1 Academic Integrity

All assessments are individual. You should feel free to discuss aspects of C programming and assessment specifications with fellow students and discuss the related APIs in general terms. You should not actively help (or seek help from) other students with the actual coding of your assessment. It is cheating to look at another student's code, and it is cheating to allow your code to be seen or shared in printed or electronic form. You should note that all submitted code will be subject to automated checks for plagiarism and collusion. If we detect plagiarism or collusion (outside of the base code given to everyone), formal misconduct proceedings will be initiated against you. If you're having trouble, seek help from a teaching staff member. Do not be tempted to copy another student's code. You should read and understand the statements on student misconduct in the course profile and on the school website: https://eecs.uq.edu.au/current-students/guidelines-and-policies-students/student-conduct

1.1 Use of AI Tools

All assessment tasks evaluate students' abilities, skills and knowledge without the aid of generative Artificial Intelligence (AI) or Machine Translation (MT). Students are advised that the use of AI technologies to develop responses (e.g. code generation) is **strictly prohibited** and may constitute student misconduct under the Student Code of Conduct.

2 Background

This assignment extends the OpenBSD kernel to add support for using VHD disk images as block devices. This is similar to the existing functionality provided by the `vnd(4)` driver, which supports using files containing a disk image as a block device. The purpose of this assignment is to exercise concepts of low-level disk operations and caching in an operating system kernel environment.

From a high-level point of view, a physical disk device presents a sequence of bytes that can be written to or read from, with the ability to quickly seek an arbitrary position and read and write at that point. Note that this is a simplification that ignores that disks address and provide access to blocks of bytes, not individual bytes.

A file on most operating systems offers similar features, i.e., a sequence of bytes that can be accessed by address. Because of these similarities, it is possible for an operating system to provide a common set of operations on both files and disks (e.g., open, close, read, write, seek, etc.) and allow them to be used interchangeably. For example, you could use `tar` to write an archive to a file in a filesystem, or directly to a disk device. `dd`, `cp`, `cat`, etc can read the bytes

from a raw disk into a file or visa versa. However, operating systems generally provide extra functionality on top of disk devices such as the ability to partition disks and mount filesystems from them.

2.1 vnd(4)

The vnd(4) driver in OpenBSD provides a "disk-like interface to a file". This means the OpenBSD kernel can open a file and present it as a block device to the rest of the system, which in turn allows for the creation and use of filesystems on these disk images.

The `vnd(4)` driver currently only supports using raw disk images as backing files. There's a one-to-one mapping of data offsets for data in the end disk device and the byte offset of that data in the underlying file. This makes the implementation very simple, with the downside that the backing file has to be the same size as the disk `vnd` is presenting. If you have a 32G disk image, the file will be 32G regardless of how much data is actually stored inside a filesystem mounted on top of it. Similar functionality exists in the `loop` driver in Linux, and the `lofi` driver in Solaris and Illumos.

2.2 Virtual Hard Disk (VHD)

Virtual Hard Disk (VHD) is a file format for disk images used by Connectix's Virtual PC hypervisor, which was later acquired by Microsoft and renamed to Microsoft Virtual PC. A defining feature of VHD files is that they allocate space and extend the size of the file backing a disk image on demand. This is different to raw disk images where the backing file has to pre-allocate space for the entire disk image up front. With VHD, you can also have fixed-size images, works in a very similar way to the raw images used by `vnd`.

VHD also supports using a read-only file as a base image and making writing changes to a separate file. When used by a hypervisor, this allows for thin provisioning of virtual machines where only changes made by each virtual machine are stored rather than repeated copies of disks. Due to the popularity of Connectix/Microsoft Virtual PC, it is common for disk images to be distributed as VHD files.

The VHD format used by this assignment is documented at:
https://stluc.manta.uqcloud.net/comp3301/public/2024/comp3301-vhd-spec.pdf

The same file can be found on Blackboard. This specification differs from the official Microsoft specification in that it includes annotations for ease of understanding and contains corrections to errors in the official specification. The official VHD format is documented by Microsoft in `Virtual Hard Disk Format Spec_10_18_06.doc`.

3 Instructions

To complete the assignment, you will need to do the following:

1. Download the base code patch

```
cd ~
ftp https://stluc.manta.uqcloud.net/comp3301/public/2024/comp3301-2024-
    a3.patch
```

2. Create the `a3` branch

```
cd /usr/src
git checkout -b a3 openbsd-7.5
```

3. Apply the base code patch

```
git am < ~/comp3301-2024-a3.patch
```

4. Install the includes

```
cd /usr/src/include
doas make includes
```

5. Build the kernel

```
cd /usr/src/sys/arch/amd64/compile/GENERIC.MP
make obj
make config
make -j4
doas make install
```


6. Reboot

```
doas reboot
```


7. Build and install `vhdctl`

```
cd /usr/src/usr.sbin/vhdctl
make obj
make
doas make install
```

8. Create `vhd(4)` device node 0

```
doas cp /usr/src/etc/etc.amd64/MAKEDEV /dev
cd /dev
doas chmod 755 MAKEDEV
doas ./MAKEDEV vhd0
```

4 Specifications

You will be extending the OpenBSD kernel to add support for using `VHD` files as a backend for a `vhd(4)` virtual block device. `vhd(4)` is roughly based on `vnd(4)`. Boilerplate code for the device entry points and command line utility will be provided, but you will be implementing the handling of the file and the VHD file format within the provided kernel driver.

Only a subset of the VHD functionality listed in Microsoft's specification of the file format is required. The following functionality is required:

- Read support
- Write support
- Fixed-size images
- Dynamic images

Differencing images do not need to be supported. In addition to supporting the VHD file format, the kernel should implement the following:

- Deny attaching VHD files which are invalid, corrupted or contain features not supported by your kernel driver.
- Deny detaching VHD files when the disk is open unless the force flag is passed to the `VHDIOCDETACH` `ioctl`.
- Return the name of the VHD file the device was attached to for the `VHDIOCFNAME` `ioctl`.
- Populate a `struct stat` for the currently attached VHD file for the `VHDIOCSTAT` `ioctl`.

4.1 VHD Disk I/O

Reads and writes against the `vhd` block device should be mapped to `vn_rdrw()` against the VHD backing file. Writing to the `vhd` device must not corrupt the backing VHD disk image. **The read and write functionality contributes to the majority of the marks for this assignment.**

Caching of VHD structures (e.g., footer, dynamic disk header and block allocation table) and data blocks should be implemented for dynamic images, for improved performance. For example, if you read a sector from block `0x3301` and immediately read another sector from the same block, you should not need to read the backing VHD file twice. Same goes for mixed reads and writes, for example, read on block `0x3010` followed by a write on the same block should result in one read and one write (not 2 reads and 1 write). The number of data block cached, the cache replacement algorithm (if you intend on caching 2 or more blocks) and the implementation details of the caching is up to you to decide, as long as some form of caching is implemented.

4.2 `ioctl` interface

The following `ioctl`s should only work on the raw partition of the character device associated with each `vhd` disk. Except for `VHDIOCATTACH`, they should only work when a `vhd` disk is attached to a backing file.

`VHDIOCATTACH`

Specify the VHD file to attach as a block device, and parameters for using the disk in the kernel. The `vhd_attach` struct contains the following fields:

- `vhd_file` - The name of the VHD file to attach a `vhd` disk to.
- `vhd_readonly` - The `vhd` disk can be written to when set to 0, and if the VHD file is read-only, the `vhd` disk should fail to attach. The `vhd` disk should be read-only when set to a non-zero value, and the VHD file should be opened read-only.

`VHDIOCDETACH`

This `ioctl` requests the block device be detached, and the backing file closed. If the disk is in use, the request should fail with `EBUSY` unless the unsigned int `ioctl` argument is set to a non-zero value. A non zero value requests the forced removal of the block device and close of the backing VHD file.

`VHDIOCFNAME`

This `ioctl` requests the name of the VHD file used for the currently attached block device. The name should be the same as what was passed as the filename in the `VHDIOCATTACH` `ioctl`.

`VHDIOCSTAT`

This `ioctl` is equivalent to an `fstat(2)` system call against the backing file.

5 Provided Tools/Files

5.1 `vhdctl`

The patch includes source for a `vhdctl` utility that uses the `ioctl`s defined above to control `vhd` devices. It is similar to `vnconfig`. The source can be found after the patch is applied in `src/usr.sbin/vhdctl`.

5.2 `rawtest.img`

This is a raw disk of size 10 MiB which contains a filesystem with the following files:


```
-rw-r--r--   1 root   wheel     10 Oct   4 21:37 comp3301.txt
drwxr-xr-x   2 root   wheel    512 Oct   4 21:39 folder/
-rwxr-xr-x   1 root   wheel   6496 Oct   4 21:40 hello*
-rw-r--r--   1 root   wheel     94 Oct   4 21:40 hello.c
-rw-r--r--   1 root   wheel     13 Oct   4 21:36 hello.txt
-rw-r--r--   1 root   wheel   2739 Oct   4 21:38 test.txt
```

Download:
https://stluc.manta.uqcloud.net/comp3301/public/2024/rawtest.img.

5.3 `fixedtest.vhd`

This is a fixed-size VHD version of `rawtest.img`. Download:
https://stluc.manta.uqcloud.net/comp3301/public/2024/fixedtest.vhd.

5.4 `dynatest.vhd`

This is a dynamic VHD version of `rawtest.img`. Download:
https://stluc.manta.uqcloud.net/comp3301/public/2024/dynatest.vhd.

5.5 `vhdtool`

This tool is not available yet. It will allow you to create VHD files and convert disk images between raw and VHD. It will also allow you to perform basic consistency checks on VHD files. You do not need this tool at the current stage, it will be released later.

6 Misc. Requirements

6.1 Code Style

Your code is to be written according to OpenBSD's style guide, as per the `style(9)` man page. An automatic tool for checking for style violations is available at:
https://stluc.manta.uqcloud.net/comp3301/public/2022/cstyle.pl.

Code style marks will be calculated based on the number of style violations in the code which you have written yourself or modified - style violations in the OpenBSD source tree or in the base code will not affect code style marks. Some level of functionality is required to score marks for code style (i.e., no submission implies no style violations, however no style marks will be awarded in that case).

6.2 Reliability, Robustness, Portability and Modularity

In order to score higher marks, your code is expected to be reliable and robust, that is it should handle all errors appropriately and should not crash unexpectedly. Your code should also be portable and modular, in the sense that constants and magic numbers should not be hard coded and similar code should not be duplicated in multiple areas.

6.3 Compilation

Your code must be compile-able under the `GENERIC.MP` (generic multiprocessor) configuration for the AMD64 (aka x86-64 and x64) architecture. Code with compile-time errors will be marked manually, and may result in heavy penalties.

7 Git Submission

Submission must be made electronically by committing and pushing your changes to your course-provided Git repository on `source.eait.uq.edu.au`. Code checked into any other branch in your repository or not pushed to the `a3` branch (i.e., left on your VM) will not be marked.

As per the `source.eait.uq.edu.au` usage guidelines, you should only commit source code and makefiles. Please do not commit `cscope` outputs, core dumps or base code patch files.

Your `a3` branch should consist of:

- The OpenBSD 7.5 base commit (provided)
- The A3 base patch commit (provided)

- Commit(s) for adding the required functionality (by yourself)

Commit history and commit messages do not contribute to your grade. However, it is strongly recommended that you commit frequently and use appropriate commit messages. You are kindly reminded that this is a professional academic assignment, and therefore, inappropriate commit messages (i.e., profanity) may result in the deduction of style marks.

7.1 Marking

Your submission will be marked by course staff during an in-person demo with you at your prac session of the due week. You must attend your session in person; otherwise, your submission will not be marked. Online attendance, e.g., Zoom, is not permitted.

7.2 Demo Session

You will be asked to demonstrate your assignment as part of your regular practical session. Demonstrations will run on a marking VM (not your VM), which has nothing but the latest code that you have pushed to the `a3` branch before the submission deadline. Manual and automated tests and manual code examinations will be conducted during the demo. Changes to your code during the session will not be accepted.

It is your responsibility to ensure that your code compiles and operates correctly. Code that fails to compile or code that crashes the kernel upon boot results in your submission not being marked for functionality.

You will be asked to explain certain aspects of your submission, failure to demonstrate your understanding or to justify your design decisions may result in penalties for the functionality marks of the relevant categories. Failure to prove that you have a sufficient understanding of the code that you have presumably written or a wrong understanding of the concepts yet correct code may result in allegations of academic misconduct.

8 Recommendations

8.1 Backups

It is *highly recommended* that you use Git to regularly backup your assignment code. Beware that corruptions to your file-system can happen in the event of kernel crashes, and will require you to start from scratch if there is no backup. Regularly committing code to Git and pushing to origin ensures that your code will not be lost in the event of a file-system corruption.

It is also recommended that you take a snapshot of your virtual machine (VM) before you attempt the programming aspect of this assignment. This can be done by running `snapshot <name>` in your VM Control.

8.2 Relevant Manual Pages

The following manual pages may be useful for the understanding of existing functionalities or the implementation of the assignment. You are not required or expected to use all of these in the code which you write.

- vnd(3) - `vnode` Disk Driver
- vnode(9) - `vnodes`
- vfs(9) - Kernel Interface to File Systems
- ioctl(2) - Control Device
- MAKEDEV(8) - Create System and Device Special Files
- malloc(9) - Kernel Memory Allocator
- RB_PROTOTYPE(3) - Red-Black Tree Macros
- KASSERT(9) - Kernel Assert Routines

8.3 Testing

Sample commands and expected outputs will be posted onto the Ed discussion board as students progress through the assignment. You are expected to conduct testing yourself and create your own test cases.

9 Specification Clarifications

Specification clarifications and/or updates may be issued, they will be communicated through Blackboard and/or Ed. If needed, you are encouraged to seek for spec clarifications in the A3 Spec Clarification Megathread on Ed rather than by making individual posts or emailing the teaching staff.

All clarifications made since the initial release of this specification are coloured red in this document.