Homework: xv6 log

This assignment explores the xv6 log in two parts. First, you'll artificially create a crash which illustrates why logging is needed. Second, you'll remove one inefficiency in the xv6 logging system.

Submit your solution before the beginning of the next lecture to the submission web site.

Creating a Problem

The point of the xv6 log is to cause all the disk updates of a filesystem operation to be atomic with respect to crashes. For example, file creation involves both adding a new entry to a directory and marking the new file's inode as in-use. A crash that happened after one but before the other would leave the file system in an incorrect state after a reboot, if there were no log.

The following steps will break the logging code in a way that leaves a file partially created.

First, replace `commit()` in `log.c` with this code:

```
#include "mmu.h"
#include "proc.h"
void
commit(void)
{
  int pid = myproc()->pid;
  if (log.lh.n > 0) {
    write_log();
    write_head();
    if(pid > 1)             // AAA
      log.lh.block[0] = 0; // BBB
    install_trans();
    if(pid > 1)             // AAA
      panic("commit mimicking crash"); // CCC
    log.lh.n = 0;
    write_head();
  }
}
```

The BBB line causes the first block in the log to be written to block zero, rather than wherever it should be written. During file creation, the first block in the log is the new file's inode updated to have non-zero `type`. Line BBB causes the block with the updated inode to be written to block 0 (whence it will never be read), leaving the on-disk inode still marked unallocated. The CCC line forces a crash. The AAA lines suppress this buggy behavior for `init`, which creates files before the shell starts.

Second, replace `recover_from_log()` in `log.c` with this code:

```
static void
recover_from_log(void)
{
  read_head();
  cprintf("recovery: n=%d but ignoring\n", log.lh.n);
```

```
  // install_trans();
  log.lh.n = 0;
  // write_head();
}
```

This modification suppresses log recovery (which would repair the damage caused by your change to `commit()`).

Finally, remove the `-snapshot` option from the definition of `QEMUEXTRA` in your Makefile so that the disk image will see the changes.

Now remove `fs.img` and run xv6:

```
  % rm fs.img ; make qemu
```

Tell the xv6 shell to create a file:

```
  $ echo hi > a
```

You should see the panic from `commit()`. So far it is as if a crash occurred in a non-logging system in the middle of creating a file.

Now re-start xv6, keeping the same `fs.img`:

```
  % make qemu
```

And look at file `a`:

```
  $ cat a
```

You should see `panic: ilock: no type`. Make sure you understand what happened. Which of the file creation's modifications were written to the disk before the crash, and which were not?

Solving the Problem

Now fix `recover_from_log()`:

```
static void
recover_from_log(void)
{
  read_head();
  cprintf("recovery: n=%d\n", log.lh.n);
  install_trans();
  log.lh.n = 0;
  write_head();
}
```

Run xv6 (keeping the same `fs.img`) and read `a` again:

```
  $ cat a
```

This time there should be no crash. Make sure you understand why the file system now works.

Why was the file empty, even though you created it with `echo hi > a`?

Now remove your modifications to `commit()` (the if's and the AAA and BBB lines), so that logging works again, and remove `fs.img`.

Streamlining Commit

Suppose the file system code wants to update an inode in block 33. The file system code will call `bp=bread(block 33)` and update the buffer data. `write_log()` in `commit()` will copy the data to a block in the log on disk, for example block 3. A bit later in `commit`, `install_trans()` reads block 3 from the log (containing block 33), copies its contents into the in-memory buffer for block 33, and then writes that buffer to block 33 on the disk.

However, in `install_trans()`, it turns out that the modified block 33 is guaranteed to be still in the buffer cache, where the file system code left it. Make sure you understand why it would be a mistake for the buffer cache to evict block 33 from the buffer cache before the commit.

Since the modified block 33 is guaranteed to already be in the buffer cache, there's no need for `install_trans()` to read block 33 from the log. Your job: modify `log.c` so that, when `install_trans()` is called from `commit()`, `install_trans()` does not perform the needless read from the log.

To test your changes, create a file in xv6, restart, and make sure the file is still there.

Submit: your modified log.c