TCSS 422 — Computer Operating Systems
Winter 2015 — Homework Assignment 6

Due Date: Tuesday, Mar. 10

Guidelines

Homework should be electronically submitted to the instructor by the end of the day on the due date. A submission link is provided on the course Canvas page for this assignment.

Assignment Description

In this assignment you'll complete the implementation of a compressed ramdisk device driver for MINIX. A ramdisk is a virtual storage device that stores its contents in RAM rather than on a secondary-storage device, e.g., a hard drive. In effect, a ramdisk lets a computer system use part of RAM as if it were a hard drive. Starter files are available on the course Canvas page for this assignment.

Implementation Specifications

The provided `crd.c` source file is based on the existing MINIX memory device driver (`/usr/src/drivers/memory/memory.c`). While the MINIX memory device driver manages several devices, `crd.c` has been stripped-down to a bare minimum of code implementing only a single ramdisk. The provided code is functional, but implements an uncompressed ramdisk. Your primary task is to modify the device driver to incorporate compression.

To use and test the new ramdisk device, you will first need to create a device entry for it. All devices appear as files in the `/dev` folder and have a set of associated permissions recorded in the `/etc/system.conf` file. The name for this new device is `crd` (for *c*ompressed *r*am*d*isk). To create your device's `/dev` entry, run the following command:

```
mknod /dev/crd b 17 0
```

A file named `crd` should now be present in the `/dev` folder. Next, edit the `/etc/system.conf` file by adding the following text to the end of the file:

```
service crd
{
      uid 0;
      system ALL;
      ipc ALL;
};
```

It's now time to compile and run the device driver. For consistency with the rest of the MINIX source code, the source code for your new device driver should be placed in the directory tree containing the rest of the MINIX source code. Create the folder `/usr/src/drivers/crd` and copy the two provided files `crd.c` and `Makefile` there.

To compile the device driver, first make sure that your present working directly is `/usr/src/drivers/crd`. Next, run the command `make` to compile the device driver. The `make` program reads the provided `Makefile` for instructions on how to compile and link the source code. When complete, there will be several new files in the current folder, including the executable device driver `crd`.

Device drivers are started differently from other programs. To start the device driver run the command:

```
service up /usr/src/drivers/crd/crd -dev /dev/crd
```

If you display a list of running processes using the "`ps x`" command you can see that the device driver has been loaded and is running. The `/dev/crd` file is now effectively a file whose contents are stored in memory. You can read and write to this file as you would any other. For example,

```
echo "Hello there!" > /dev/crd
head -1 /dev/crd
```

The first command writes a string to the file, while the second command displays the first line of the file (which should be just the previously written string).

The true purpose of a ramdisk is to act like a hard drive, i.e., to store part of the file system. We can format the ramdisk and mount it to the MINIX file system just as we would any hard drive. For example,

```
mkfs.mfs /dev/crd
mount /dev/crd /mnt
```

The first command formats the ramdisk, while the second command mounts it on the `/mnt` folder. Note: if you've another device or folder already mounted on `/mnt` you'll need to pick another location. Any new files and folders created in `/mnt` will be stored in the ramdisk rather than the VM's virtual hard drive. For example,

```
echo "Hello there!" > /mnt/hi.txt
cat /dev/zero > /mnt/junk
```

These two command together fill all the space on the ramdisk by creating two files, one containing a simple string and the other containing nothing but zeros. You can still examine the raw contents of the ramdisk via `/dev/crd`. For example,

```
cat /dev/crd
```

The output of the above command will largely be unintelligable (file system data structures are not meant to be human-readable), but you can recognize some elements such as file names and file contents.

Once you begin modifying `crd.c` you'll need to shut down the device driver so that you can recompile and restart it. To shut down the device driver run the command:

```
service down crd
```

You're now ready to modify `crd.c` so that it implements a compressed ramdisk. The compression process will be transparent to users of the ramdisk, i.e., any data written to the ramdisk is automatically compressed and any data read from the ramdisk is automatically uncompressed. From the user's perspective the ramdisk behaves exactly the same as an uncompressed ramdisk.

The provided ramdisk implementation stores up to 2 MB of data broken into 4 KB "blocks." At present, data in the ramdisk is uncompressed and thus each 4 KB block of data is stored in a 4 KB array. You must modify the implementation so that each block of data is compressed, occupying (hopefully) far less than 4 KB. To compress and uncompress data you can use functions from the zlib compression library, specifically the `compress` and `uncompress` functions. Documentation for these functions can be found at http://www.zlib.net/manual.html#Utility.

In `crd.c`, the `m_block_transfer` function handles all reads and writes to/from the ramdisk. These transfers of data are between two processes, the ramdisk device driver and the process requesting the transfer. The actual copying of data is done by the operating system via the `sys_safecopyfrom` and `sys_safecopyto` functions. You will need to modify the `m_block_transfer` function to incorporate compression into the ramdisk. The source code is commented to help you understand how the code works at present, from which you should be able to design your modifications.

Deliverables

The following items should be submitted to the instructor. Failure to correctly submit assignment files will result in a 10% grade penalty.

1) The completed `crd.c` source file.

Do not include any extraneous files, such as Eclipse IDE files, object files, or subversion files.