Atharva Walawalkar ID: 924254653
Github: AtharvaWal2002 CSC415 Operating Systems

Assignment 6 - Coding a device driver

Description

This assignment implements a Linux kernel device driver that performs Run Length Encoding (RLE) compression and decompression. The driver provides a character device interface that allows user applications to compress and decompress data using the RLE algorithm.

Run Length Encoding (RLE) Overview

Run Length Encoding is a simple form of data compression that encodes consecutive repeated characters as a count followed by the character. For example:

- "AAAA" becomes "4A" (4 occurrences of 'A')
- "AAABBB" becomes "3A3B" (3 'A's followed by 3 'B's)
- "WWWWWWWWWWWWBWWWWWWWWWWWWB" becomes "12W1B12W1B"

This makes RLE particularly effective for data with many repeated characters.

Approach:

After analyzing the requirements for a device driver that performs RLE compression/decompression, I determined the following implementation strategy:

IOCTL Command Design:
For the IOCTL command implementation, I decided to use 'r' as the unique magic number for RLE operations to avoid conflicts with other drivers while maintaining an association with RLE functionality. The command structure was planned using the _IOW macro, which takes three parameters: the magic number 'r', a command number (1), and an integer parameter for mode setting. I chose _IOW over alternatives like _IO or _IOR because our mode setting operation only requires writing from user space to kernel, without any need for bidirectional data transfer.

Data Structure Design:
The private data structure was designed to contain three elements: a character buffer for storing input/output data, a current buffer size tracker, and an operation mode flag for switching between compression and decompression states. This design ensures that each process maintains its state independently, which is essential for proper driver operation when multiple processes are accessing it simultaneously.

Memory Management Design:
After careful consideration of various factors, I have decided to select 4KB as the maximum buffer size. This decision was based on balancing typical text compression needs with kernel memory efficiency, while ensuring safe handling of user input sizes. This size provides ample space for most text operations while preventing excessive memory usage in the kernel space.

RLE Algorithm Implementation:
For the compression algorithm, I implemented a two-pass approach where the first scan counts consecutive characters and the second pass generates the compressed output. In contrast, the decompression was designed to work in a single pass for efficiency. To maintain consistency in byte storage and prevent overflow issues, I added a count limit of 255 for run lengths.

Control Interface:
The control interface was implemented using an IOCTL command for mode switching. The modes were represented by simple numeric values: 0 for compression and 1 for decompression, which is distinct from the user interface choices (1(Compress) ,2(Decompress) ,3(Exit)). To maintain system stability, validation checks were added to prevent invalid mode settings from being processed.

Testing Strategy:
The testing strategy encompassed various test cases including single character repetition (like "AAAA"), mixed patterns (such as "AAABBB"), and numeric characters as input. I developed a user interface that allows easy testing of both compression and decompression modes, ensuring all functionality could be thoroughly verified.

Installation and Usage

Prerequistes for running a driver

```
sudo apt update
sudo apt install gcc-12
sudo apt install pahole
sudo ln -sf /sys/kernel/btf/vmlinux /usr/lib/modules/$(uname -r)/build/vmlinux
```

Commands

```
# Go to Module directory
cd Module
```

```
# Compile the driver
make

# Remove any existing driver (if needed or if you have a pre existing driver loaded)
sudo rmmod rle_driver

# Load the new driver
sudo insmod rle_driver.ko

# Set permissions for the device file
sudo chmod 666 /dev/rledev

# Go to Test directory
cd ../Test

# Compile and run
make
./rle_test

# Unload the driver
sudo rmmod rle_driver

(Optional)
# Check if driver loaded successfully
sudo dmesg | tail

# Check if device file exists and has correct permissions
ls -l /dev/rledev

# Check if module is loaded
lsmod | grep rle
```

Issues and Resolutions:

ISSUE 1:

My first issue was that the Decompression was failing after first operation, this was happening due to incorrect handling of the file offset in file position. I had an offset check in read that was causing failure

```
student@student:~/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module$ sudo insmod rle_driver.ko
student@student:~/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module$ sudo chmod 666 /dev/rledev
student@student:~/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module$ cd ../Test
student@student:~/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Test$ make run
gcc -c -o Walawalkar_Atharva_HW6_main.o Walawalkar_Atharva_HW6_main.c -g -I.
gcc -o Walawalkar_Atharva_HW6_main Walawalkar_Atharva_HW6_main.o -g -I. -l pthread
./Walawalkar_Atharva_HW6_main

RLE Device Driver Test Menu
1. Compress string
2. Decompress data
3. Exit
Choose option: 1
Enter string to compress: AAAABBBB
Original: Data (8 bytes): AAAABBBB
Compressed (use this format for decompression): 4 A 4 B

RLE Device Driver Test Menu
1. Compress string
2. Decompress data
3. Exit
Choose option: 2
Enter compressed data in format: count char count char
Example: 3 A 3 B for AAABBB
Enter input: 4 A 4 B
Decompressed: Data (0 bytes):

RLE Device Driver Test Menu
1. Compress string
2. Decompress data
3. Exit
Choose option: 2
Enter compressed data in format: count char count char
Example: 3 A 3 B for AAABBB
Enter input: 3 A 3 B
Decompressed: Data (0 bytes):

RLE Device Driver Test Menu
1. Compress string
2. Decompress data
3. Exit
Choose option:
```

Error was solvcd by getting rid of the offset check this doesn't break operations as since each operation was independent in read and didnt need this check.

Got rid of the below check:

```
if (*offset > 0) {
        return 0;  // Early return if offset is non-zero
}
```

1. First read worked (offset starting at 0)
2. Offset gets incremented after first read
3. Next read sees offset > 0 and returns 0 immediately
4. Result: No data returned after first read

ISSUE 2:

Had issues with the class creation API, I was referring to an older linux driver document where the class_create function took 2 arguments as inputs (e.g. class_create(THIS_MODULE, CLASS_NAME)) but in the latest iteration of class_creation it just needs the CLASS_NAME argument to work this caused below error:

```
student@student:~/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module$ make
make -C /lib/modules/`uname -r`/build M=/home/student/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module modules
make[1]: Entering directory '/usr/src/linux-headers-6.8.0-49-generic'
warning: the compiler differs from the one used to build the kernel
  The kernel was built by: aarch64-linux-gnu-gcc-12 (Ubuntu 12.3.0-1ubuntu1~22.04) 12.3.0
  You are using:           gcc-12 (Ubuntu 12.3.0-1ubuntu1~22.04) 12.3.0
  CC [M]  /home/student/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module/rle_driver.o
In file included from /home/student/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module/rle_driver.c:15:
/home/student/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module/rle_driver.c: In function 'rle_init':
./include/linux/init.h:184:22: error: passing argument 1 of 'class_create' from incompatible pointer type [-Werror=incompatible-pointer-types]
  184 | #define THIS_MODULE (&__this_module)
      |                      ^~~~~~~~~~~~~~
      |                      |
      |                      struct module *
/home/student/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module/rle_driver.c:340:30: note: in expansion of macro 'THIS_MODULE'
  340 |     rle_class = class_create(THIS_MODULE, CLASS_NAME);
      |                              ^~~~~~~~~~~
In file included from ./include/linux/device.h:31,
                 from /home/student/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module/rle_driver.c:20:
./include/linux/device/class.h:228:54: note: expected 'const char *' but argument is of type 'struct module *'
  228 | struct class * __must_check class_create(const char *name);
      |                                           ~~~~~~~~~~~~~^~~~
/home/student/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module/rle_driver.c:340:17: error: too many arguments to function 'class_create'
  340 |     rle_class = class_create(THIS_MODULE, CLASS_NAME);
      |                 ^~~~~~~~~~~~
./include/linux/device/class.h:228:29: note: declared here
  228 | struct class * __must_check class_create(const char *name);
      |                             ^~~~~~~~~~~~
cc1: some warnings being treated as errors
make[3]: *** [scripts/Makefile.build:243: /home/student/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module/rle_driver.o] Error 1
make[2]: *** [/usr/src/linux-headers-6.8.0-49-generic/Makefile:1925: /home/student/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module] Error 2
make[1]: *** [Makefile:240: __sub-make] Error 2
make[1]: Leaving directory '/usr/src/linux-headers-6.8.0-49-generic'
make: *** [Makefile:13: all] Error 2
student@student:~/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module$
```

This issue was solved by updating the class creation call according to the newer version of the documentation and using just 1 argument.

ISSUE 3:

Added support for integer input while using decompress initially the code only supported , ascii values for decompression call.

Solved: Solved this bug by using the atoi(ascii to integer) library this handles user input accordingly in the test file

Screenshot of loading the Driver:

```
student@student:~/Documents/csc415-assignment-6-device-driver-AtharvaWal2002$ cd Module/
student@student:~/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module$ make
make -C /lib/modules/`uname -r`/build M=/home/student/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module modules
make[1]: Entering directory '/usr/src/linux-headers-6.8.0-49-generic'
warning: the compiler differs from the one used to build the kernel
  The kernel was built by: aarch64-linux-gnu-gcc-12 (Ubuntu 12.3.0-1ubuntu1~22.04) 12.3.0
  You are using:           gcc-12 (Ubuntu 12.3.0-1ubuntu1~22.04) 12.3.0
  CC [M]  /home/student/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module/rle_driver.o
  MODPOST /home/student/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module/Module.symvers
  CC [M]  /home/student/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module/rle_driver.mod.o
  LD [M]  /home/student/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module/rle_driver.ko
  BTF [M] /home/student/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module/rle_driver.ko
make[1]: Leaving directory '/usr/src/linux-headers-6.8.0-49-generic'
student@student:~/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module$ sudo insmod rle_driver.ko
student@student:~/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module$ sudo chmod 666 /dev/rledev
student@student:~/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module$ cd ../Test
student@student:~/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Test$ make
gcc -c -o Walawalkar_Atharva_HW6_main.o Walawalkar_Atharva_HW6_main.c -g -I.
gcc -o Walawalkar_Atharva_HW6_main Walawalkar_Atharva_HW6_main.o -g -I. -l pthread
student@student:~/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Test$
```

Screenshot of unloading the driver:

```
student@student:~/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Test$ sudo rmmod rle_driver
student@student:~/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Test$
```

Screenshot of Compilation:

```
student@student:~/Documents/csc415-assignment-6-device-driver-AtharvaWal2002$ cd Module
student@student:~/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module$ make
make -C /lib/modules/`uname -r`/build M=/home/student/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module modules
make[1]: Entering directory '/usr/src/linux-headers-6.8.0-49-generic'
warning: the compiler differs from the one used to build the kernel
  The kernel was built by: aarch64-linux-gnu-gcc-12 (Ubuntu 12.3.0-1ubuntu1~22.04) 12.3.0
  You are using:           gcc-12 (Ubuntu 12.3.0-1ubuntu1~22.04) 12.3.0
  CC [M]  /home/student/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module/rle_driver.o
  MODPOST /home/student/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module/Module.symvers
  CC [M]  /home/student/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module/rle_driver.mod.o
  LD [M]  /home/student/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module/rle_driver.ko
  BTF [M] /home/student/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module/rle_driver.ko
make[1]: Leaving directory '/usr/src/linux-headers-6.8.0-49-generic'
student@student:~/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module$ sudo insmod rle_driver.ko
student@student:~/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module$ sudo chmod 666 /dev/rledev
student@student:~/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Module$ cd ../Test/
student@student:~/Documents/csc415-assignment-6-device-driver-AtharvaWal2002/Test$ make run
gcc -c -o Walawalkar_Atharva_HW6_main.o Walawalkar_Atharva_HW6_main.c -g -I.
gcc -o Walawalkar_Atharva_HW6_main Walawalkar_Atharva_HW6_main.o -g -I. -l pthread
./Walawalkar_Atharva_HW6_main
```

Screenshot of execution:

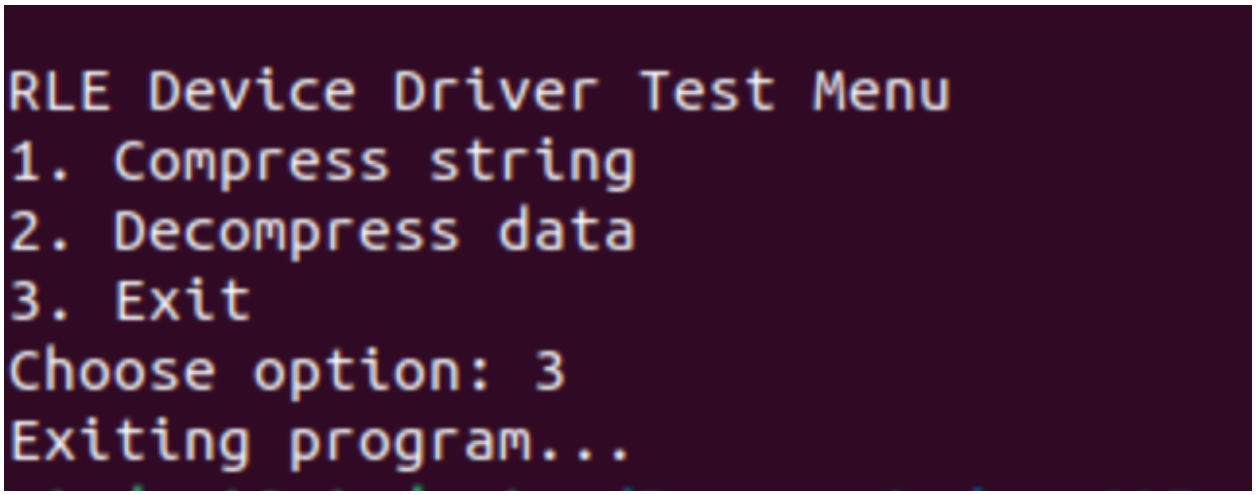
compress:

```
RLE Device Driver Test Menu
1. Compress string
2. Decompress data
3. Exit
Choose option: 1
Enter string to compress: AAAccc
Original: Data (6 bytes): AAAccc
Compressed (use this format for decompression): 3 A 3 c
```

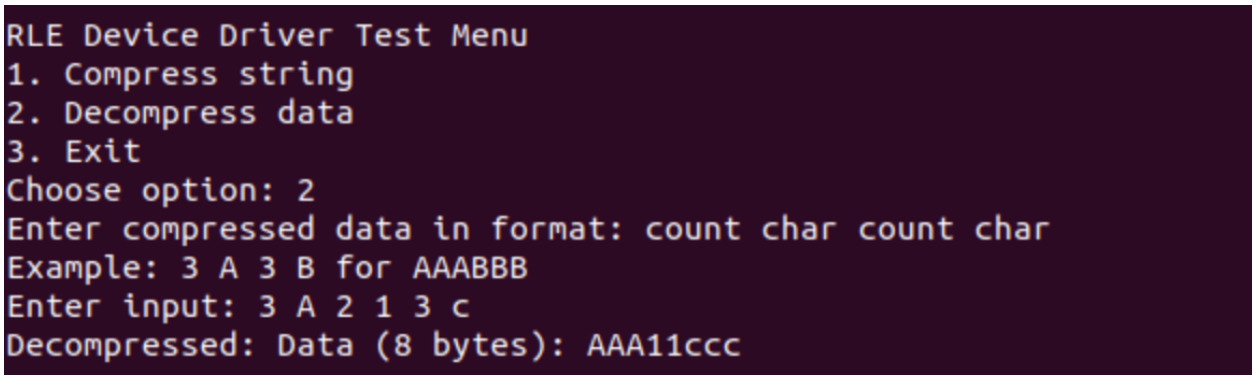
Decompress:

```
RLE Device Driver Test Menu
1. Compress string
2. Decompress data
3. Exit
Choose option: 2
Enter compressed data in format: count char count char
Example: 3 A 3 B for AAABBB
Enter input: 3 A 3 c
Decompressed: Data (6 bytes): AAAccc
```

Exit



Using alphanumeric patterns:



Invalid choice handling

```
RLE Device Driver Test Menu
1. Compress string
2. Decompress data
3. Exit
Choose option: 3 A 2 1
Exiting program...
```

```
RLE Device Driver Test Menu
1. Compress string
2. Decompress data
3. Exit
Choose option: 4
Invalid option. Please choose 1-3
```

Complete compilation + Execution: