NYU Tandon School of Engineering

CS-GY 6233, Fall 2023

Homework #1 (due Sep 24th, 11:55PM)

Academic Honesty

Aside from the narrow exception for collaboration on homework, all work submitted in this course must be your own. Cheating and plagiarism will not be tolerated. If you have any questions about a specific case, please ask me. We will be checking for this!

NYU Tandon's Policy on Academic Misconduct:
http://engineering.nyu.edu/academics/code-of-conduct/academic-misconduct

Homework Notes

- Read the assignment carefully, including what files to include.
- Don't assume limitations unless they are explicitly stated.
- Treat provided examples as just that, not as an exhaustive list of cases that should work.
- When in doubt regarding what needs to be done, ask. Another option is to test it in the real UNIX operating system. Does it behave the same way?
- Test your solutions, make sure they work.

Rubric

Since we had some issues before on homework 1. Here are some of the things we know we will test, but these are not the only things we will test. Therefore, make sure to test your program thoroughly and thoughtfully.

- -50: `uniq` does not work
- -10: `uniq` does not handle long lines (more than 512 characters)
- -10: Debug `printf` left in code
- -10: `cat example.txt | uniq` does not work
- -10: `uniq -c example.txt` does not work
- -10: `uniq -d example.txt` does not work
- -10: `uniq -i example.txt` does not work

Part 1: Implementing the `uniq` command (50 points)

For this assignment you will be implementing the commonly used command line program in Linux called "`uniq`" but you will be doing this yourself in xv6 to get an idea of how command line programs are written. `uniq` is a Unix utility which, when fed a text file, outputs the file with adjacent identical lines collapsed to one. If a filename is provided on the command line (i.e., `uniq` FILE) then `uniq` should open it, read, filter out, print without repeated lines in this file, and then close it. If no filename is provided, `uniq` reads from standard input.

For the examples in this assignment you will be running the `uniq` command on the README.md file included in xv6. Before you begin coding please delete all the text in README.md file and paste the following lines of text in its place. This is the file and subsequent text you will be running the `uniq` program on.

Text to copy and paste into xv6 README.md file:

No. 1

No. 2

No. 2

No. 2

No. 3

No. 4

No. 5

No. 6

No. 6

No. 2

no. 2

Now that you have set up your test file you may begin coding by creating a `uniq.c` file and coding your program there.

Here's an example of the basic usage of `uniq`:

```
$ cat README.md

No. 1

No. 2

No. 2

No. 2

No. 3

No. 4

No. 5

No. 6

No. 6

No. 2

no. 2
```

```
$ uniq README.md

No. 1

No. 2

No. 3

No. 4

No. 5

No. 6

No. 2

no. 2
```

You should also be able to invoke it without a file, and have it read from standard input. For example, you can use a pipe to direct the output of another xv6 command into `uniq`:

```
$ cat README.md | uniq
```

```
No. 1

No. 2

No. 3

No. 4

No. 5

No. 6

No. 2

no. 2
```

Hints

1. Many aspects of this are similar to the `wc.c` program: both can read from standard input if no arguments are passed or read from a file if one is given on the command line. Reading its code will help you if you get stuck.
2. Still confused with `uniq's` behavior? Use man `uniq` for help.
3. The following link should help give you an idea on how to parse command line arguments and fetch them from the main function: https://dev-notes.eu/2018/03/parse-command-line-arguments-in-c/

Part 2: Extending `uniq` (50 points)

The traditional UNIX `uniq` utility can do lots of things, such as:

- `-c`: count and group prefix lines by the number of occurrences
- `-d`: only print duplicate lines
- `-i`: ignore differences in case when comparing

Here, we are going to implement these three behaviors in your version of uniq. The expected output of these commands should be:

```
$ uniq -c README.md
1 No. 1
3 No. 2
1 No. 3
1 No. 4
1 No. 5
2 No. 6
1 No. 2
1 no. 2

$ uniq -d README.md
No. 2
No. 6

$ uniq -i README.md
No. 1
No. 2
No. 3
```

```
No. 4

No. 5

No. 6

No. 2
```

```
$ uniq -c -i README.md

1 No. 1

3 No. 2

1 No. 3

1 No. 4

1 No. 5

2 No. 6

2 No. 2
```

Notice that “No. 2” should be the same as “no. 2” if `uniq` is not case-sensitive. Also, `-c` and `-d` won’t appear at the same time.