Assignment #1

Due: April 9, 2012 11:59pm

Overview

This assignment serves two purposes. Its first purpose is as an introduction to programming in C (a warm-up of sorts). Its second purpose is as an introduction to working in the Unix environment.

For this assignment you will implement a simplified version of the `uniq` program. This program serves to eliminate duplicate lines read from standard input. This basic functionality is modified by command-line switches.

`my_uniq`

Write a program named `my_uniq` (the name is changed to avoid unintentional clashes with the standard utility) that reads from *standard input* and writes to *standard output*. `my_uniq` reads lines of input and compares adjacent lines. `my_uniq` will output one copy of each line read, but will not write copies of repeated adjacent lines (duplicate adjacent lines are reduced to one line).

Your program must also support the following command-line switches

- `c` – precedes each output line with a count (and a colon) of the number of repeated adjacent copies of the line that occurred in the input
- `d` – suppresses printing of lines that are not repeated (i.e., only print duplicate lines)
- `u` – suppresses printing of lines that are repeated (i.e., only print unique lines)

The command-line options may be specified separately (each prefixed with a `-`) or together (one `-` preceding multiple options without spaces).

Consider the following example

```
% cat bob
abc
abc
123
abc
% my_uniq < bob
abc
123
abc
% my_uniq -c < bob
2: abc
1: 123
1: abc
% my_uniq -d < bob
abc
% my_uniq -u < bob
123
abc
% my_uniq -uc < bob
1: 123
1: abc
```

You can read the `man` page for `uniq` for more details as well as use the program on the unix[1–4] machines (the output differs slightly for the count option). If both the `-d` and `-u` switches are given, then report an error.

Note: Your program *must* support lines of any length. This will require that your program support "growing" a string while reading input. You should use the function developed for Lab 2.

Functions of Interest

You might find some of the following functions to be useful.

`fgets, strcmp, strdup, malloc, free`

Running Time Requirement

Your program must run in a reasonable amount of time on a relatively large file. You may test on the dictionary file on the unix[1–4] machines (`/usr/share/dict/linux.words`). Your program should take less than a second to process this entire file when using the -d switch to prevent printing (since every line is uniq).

Submit

Submit:

- All source code.
- A Makefile that will build your program.

Submit your assignment, using `handin` on unix[1–4], to the `357_hw1` directory under the `akeen` account. If you are using a late day, then submit to the respective directory based on how late your submission is (be sure that you have late days to use).

- On-time: `handin akeen 357_hw1 <files>`
- Using One Late Day: `handin akeen 357_hw1_day1 <files>`
- Using Two Late Days: `handin akeen 357_hw1_day2 <files>`

Grading

Feature	Percentage
Basic Functionality	70
-c	10
-d	5
-u	5
Error Handling	10