Homework #1: C Warmup: V$\underset{1}{\underline{o}}w\underset{2}{\underline{e}}$l C$\underset{3\ 4}{\underline{ou}}$nt

Issued: Tuesday, September 1
Due: Thursday, September 10

Purpose

This assignment allows you to learn about Unix/C development tools, simple input/output functions, arrays, and character strings. It is intended to prepare you for the similar, but more complex, assignments that follow.

Preparation

Begin by deciding what development tools you want to use. Then, practice using them. You'll likely be developing your program on one of the `onyx` nodes. You'll need to edit text files. You could use `vi`, `vim`, `gvim`, `emacs`, `eclipse`, or whatever you're familiar with from other courses. You'll use `bash`, `make`, and `gcc` to compile and execute your program. For debugging, you can choose from `gdb`, `ddd`, and `valgrind`.

Practice using these tools before attacking the assignment. For example, try them on:

```
pub/Shout/Shout.c
```

Assignment

Using the abovementioned program as a foundation, write a C program that reads text from `stdin` and writes the number of vowels in the text (i.e., a single integer) to `stdout`.

This is assignment is just a warmup: don't get carried away. Our programs will "evolve" over the course of the semester, adding features, and employing design techniques as we learn them. For now, you *must* represent the the set of vowels with this local-variable definition:

```
        const char vowels[]="aeiou";
```

Hardcoding values like this is awful, we'll avoid it later. By the way, since it's a set, order and duplication are irrelevant.

Other Requirements

- Employ good modularity, formatting, and documentation.
- Do *not* use `<strings.h>` or its cousins. Write your own functions.
- Encapsulate your vowel-counting logic in one or more functions.
- Ignore uppercase/lowercase differences. Use a function.
- Aside from type representations, impose no arbitrary limits or sizes.
- For now, use this silly makefile:

  ```
          vc: vc.c ; gcc -o vc vc.c -g -Wall
  ```
- Demonstrate that you used a debugger to fix a bug.