Aidan Smith

CSE 130

Design for Assignment 0

1.0 Introduction

This document describes all data, functional and behavioral requirements for dog.

1.1 Goals and objectives

Create an executable named dog that works very similarly to cat.

1.2 Statement of scope

dog will take in file names and operate much like cat. It needs to be able to read from files and write to the command line using the executable.

2.0 Design

A description of the design of dog.

2.1 dog.c

dog.c is made up of two functions and a main to read in arguments. These are expanded on in 3.0 Functions.

2.2 Makefile

Makefile is used to run dog cleanly, its usage is described in the file.

3.0 Functions

A description of functions used in dog.

3.1 static int openFile(char* argv)

openFile() reads in the current input and gives it the proper file number if it is a file, or makes it the standard input if it is a "-". An error message is given if there is a failure of opening the file.

3.2 static void writeFile(int fileNum, char* argv)

writeFile() reads from the file and writes from the file into the terminal. If "-" is given as an argument or no arguments are given, then the user types into the terminal. The current input is taken in as an argument, argv, to create error messages if there is a failure at reading or writing from the file.

3.3 int main(int argc, char* argv)

main is the main function of dog. The program starts here, giving its inputs and determining what functions to call.

4.0 User interface

This is described in README.md.

5.0 Restrictions and differences

dog is meant to be a very similar copy to cat. The main restriction in comparison to cat is that no flags are supported with dog. Another difference to cat is that dog reads files from right to left, as compared to cat which reads them from left to right.