Bo Liu (bliu62)

1 Introduction

This Dog.c program emulate the functionality of cat

1.1 goals and objectives

Resemble the functionality of cat, print the content of text or binary file, print the input from the user.

1.2 Statement of scope

This program can take none or any arguments : filename or - option. For input are filenames Dog.c prints the content of the file in reverse order. For none or - argument, it get the input From the user, print what user types on the command line.

2 Data design

2.1 Internal data

For the header of main function such that *int main (int argc, char *argv[]),* argc indicates the number of argument received by program; argc==1 if there are no argument
Char argv[] give the access to the individual file, such that ./dog 1.txt 2.txt, argv[2] point to the 1.txt file

Int n, count, fd get value from the read or open to indicate the success of the open or read file

2.2 Global data

Create char array : char buffer [ ] to store the input from content of file or users.
Create a global int variable: MAX_LEN to set the length of buffer to be 3278

3 Architectural and component- level design

3.1 **System structure**

The program may take zero or many argument, and - option, for different situation of argument, using the if statement to distinguish

3.11 **Situation one: zero argument**

1 if the number of argument is zero
2 get the input from the user
3 check whether the input is valid if CTRL +D
detect exit the loop
4 print the input

5 continue to read the input from user
6 END

Exact code:

```
if (argc==1){
  n=read(0,buffer,sizeof(buffer));
    while (n>0){
      write(1,buffer,n);
      n=read(0,buffer,sizeof(buffer));
}}
```

3.12 **Situation two:one or multiple argument**

1 create a for loop to hand each argument individually in reverse order
2 using strcmp to test whether the argument is “-” or normal filename

3 if argument is not “-”, using open() to get the file descriptor and give it to int fd
4 using fd to check whether the file is opened successfully
5 using read to get the content from file descriptor into the buffer and return -1 if read Error
6 using a while a loop to read the file as many times as need until read give EOF indicator
7 using write to print the content of buffer in standard output
8 if not reach the end of loop, continue

Or

3 if argument is “-”, get the input from the user
4 using while loop with condition that the input is valid if CTRL +D detect exit the While loop
5 print the input
6 continue to read the input from user
7 while loop end
8 if not reach the end of loop, continue

Exact code

```
for (int i = argc-1; i >=1; i--){
if(strcmp(argv[i],"-")!=0){
fd=open(argv[i],O_RDONLY);
  count=read(fd, buffer, sizeof(buffer));
```

```
                If(fd==1){
                perror("error");
            exit(EXIT_FAILURE);
}

          while(count>0){
            write(1,buffer,count);
            count=read(fd, buffer, sizeof(buffer));

        }else{
            n=read(0,buffer,sizeof(buffer));
               while (n>0){
              write(1,buffer,n);
             n=read(0,buffer,sizeof(buffer));
           }
      }
```

4.0 user interface design

Users give the instruction on the command line of the terminal: such as
./dog reading the input from user and print it
./dog - reading the input from user and print it
./dog file1 file2 … print the content of file in reverse order
And the output will be printed on the standard output

5.0 Testing Issues

The program may fail to read a large text files or large mixed binary test file if file is too large