1 Goal

The goal of this program is to replicate the program cat with no extra command line flags. The program dog will take in no files (read from std input, print to std output), one file, or two or more files, including a ‘’-” representing not a file, but to read from std in and print to std out.

2 Assumptions

This program should be ran on Linux 18.04. I’m assuming that c strings are allowed. I have not gotten the read write error to appear on any files tested by me, but I assume it should work.

3 Design

The general approach I’m taking is in the main function I will create an 32kb sized buffer. I will first check if no command line arguments were given and if they were not given, I will call the myStdIn function to take user input and output it back using the buffer created in main. It will terminate when read returns 0, indicating it reached the eof/cntr-d. Then after that check I will loop through the rest of the arguments given on the command line (if given) when running the program. This loop will check if a ‘-’ was given, if so, it will call myStdIn function. If not, the program will call readFile on the current filename in the argument array which will open and output the file to std out using the buffer created in main. It will break out of the loop when read returns 0, indicating eof. If open returns an error, I will use warn to print an error to stderr. I will import errno.h and use the errno variable to help with specific error messages. I will also check for a read write error, that is when the bytes read don’t equal the amount of bytes written.

4 Pseudocode

This is the pseudocode for main and any functions I use. Python like indenting for scope.

```
Main:
    Create buffer of size N. N<=32kb
    Check if argument array is length 1:
        myStdInput(buffer)
    For I = 1; i<argc; i++:
        If (‘-’ == argv[i]):
            myStdInput(buffer)
```

```
            Else:
                readFile(argv,i,buffer)
End Main


readFile (argv[], curIndex, buf):
    Fd = 0
    N_char = 0
    Fd = Open file at cur index
    If (Fd == -1):
        Call warnx -- errno
        return
    While (n_char = read(from file to buf) ) != 0):
        If (n_char = -1):
            Warn -- errno
            Break from WHILE
        endN_char = write(write from buf to stdout);
        If (n_char != endN_char):
            Warn -- input/output error
    close(fd)
End readFile


myStdInput(buf):
    N_char = 0
    endN_char = 0

    while((n_char = read(stdIN to buf)) !=0):
        endN_char = write(buf to stdout)
        If (n_char != endN_char):
            Warn -- input/output error

End myStdInput
```