Asgn0 Design Document

Jessica Pan
CruzID: jeypan

CSE 130, Fall 2019

1. Goal

The goal of this program is to implement the basic cat program, as in UNIX, without support for any flags. The program should copy data from each of the files specified on the command line to the standard output. We will call this program: dog.

2. Assumptions

I'm assuming that the commands used in the Unix cat program will be used to run the dog program. The program would also produce the proper error message to standard error if the filename isn't a file. Also, I'm assuming each file that's written as an argument in running the dog program will be separated by a space. If the arguments are grouped together then that group of words will be considered as a whole argument.

3. Design

With the assignment specifications in mind, the approach I'm taking would be to first check the arguments and see if there's more than 1 argument. The program will loop through these arguments. If the arguments consist of '-', or '--', then the program will `read()` 32KB at a time from the standard input and write the contents back into standard output. If there's only one argument, which is just `./dog`, then it's code would perform the same way as if the arguments were '-' and/or '--'.

If the arguments don't consist of the three, I just listed, then that means there are existent or nonexistent files listed as arguments. We then check if the filename is a directory or a nonexistent file, if yes, then an error message will be produced to the standard error by using `warnx()` and `warn()`. Otherwise, since we can't read a whole file in at a time, for each file, I'll repeatedly read 32KB of the file into the buffer and return the number of bytes read to the variable `bytes_read`. Then we'll write it into the standard output using the function `write()` with `bytes_read` as the count parameter. This will continue until `read()` returns 0, meaning we've reached end-of-file. I will do this to every existing file present in the arguments list. There will also be else statements that will catch any errors in the process of this program and print them to the standard error.

4. Pseudocode

Below is the pseudo code for the dog program.

```
procedure Dog
    Declare buffer of size 32000 bytes
    nbytes ← sizeof(buffer)
    if argc > 1 then
        for k ← k1,…,argc-1 do
            if (k = "-") or (k = "--") then
                while (bytes_read ← READ(stdin file descriptor, buffer, nbytes)) >= 1) do
                    WRITE(stdout file descriptor, buffer, bytes_read)
                end while
            else
                path ← argvk
                if STAT(path, &s) = 0 then
                    if argvk is a directory then
                        WARNX("%s is a directory", argvk)
                    else if argvk is a file then
                        fd ← OPEN(argvk, O_RDONLY)
                        if fd != 1 then
                            while (bytes_read ← READ(fd, buffer, nbytes)) >= 1) do
                                WRITE(stdout file descriptor, buffer, bytes_read)
                            end while
                        end if
                        CLOSE(fd)
                    else
                        WARN("%s", argvk)
                    end if
                else
                    WARN("%s", argvk)
                end if
            end if
        end for
    else
        while (bytes_read ← READ(stdin file descriptor, buffer, nbytes)) >= 1) do
            WRITE(stdout file descriptor, buffer, bytes_read)
        end while
    end if
end procedure
```