ECE 362 – Embedded Operating Systems

Assignment 1

Due: Jan 17th in D2L dropbox (zip format, not rar)

You may discuss this assignment with other students. But do not share code! This first assignment is intended to provide a module that you may use in future assignments. Note that not all details may be provided in the description below. Part of the assignment is to ask questions.

Each student is individually responsible for:

1) C code and a Makefile (zip up your source code hierarchy as needed).

2) A brief description of the overall program design (I recommend a text readme file, but a pdf or doc file would be fine as well). This might be done by providing a couple lists. One would enumerate the functions and what they do. The other would enumerate any significant variables/data structures and how they are manipulated.

Include an “all” target in the makefile that compiles everything required by the assignment. To determine whether your program works correctly, we will execute “make all” and then run your program (“shell”) against a collection of (unknown to you) test cases.

Circular history buffer: Your program should provide a function (or set of functions) to support a bash-like history capability. The program should repeatedly display a prompt and then accept a line of input that is stored in a circular buffer. The circular buffer should be able to hold 5 lines (this should be a constant you can change later).

Have your function assign numbers, starting at 1, to each input line received and print the command number as part of the prompt (for example: “10> “). Prior to storing an input line, your program should interpret special commands:

- *!x* means repeat the (absolute) input line numbered x (but only if it is one of saved commands --- there may be less than 5). In addition to storing a copy of the command, display the original input line or display an error message if the argument x is invalid.

- *exit* means terminate the program.

- *history* means print the saved commands (again, there may be less than 5).

- *parse n* means tokenize input line number *n*. This input line shouldn’t be stored in the buffer, but instead display each word (token) on a separate line.