**COSC 3346 Operating Systems**

**Project #1: (guish) "Gnu" Island Shell**

1. **Purpose**

The purpose of this assignment is to prepare you on system programming and to gain some familiarity with the system call interface.  A secondary goal is to use some of the programming tools provided in the UNIX environment.

In this assignment, you will implement a UNIX shell program that can execute simple Unix commands.  A shell is simply a Command Line Interface (CLI) program that conveniently allows a user to run other programs.  Read up on your favorite shell (such as bash, csh, tcsh, sh, ...) to see what it does.

1. **Description**

**Basic Commands:**  The input to **guish** is a sequence of commands, each provided on a separate line of input text (or typed interactively at the keyboard).  The following command must be supported:

* **progname [args]**

It should run the program **progname** with the given, possibly optional, arguments. This **progname** can be any simple Unix command or executable program file name.

* **exit**

The **exit** internal command should cause the shell to exit.

* **hist**

The *hist* history internal shell command will display the (up to) 10 most recent commands entered from the user.  See details below how this should work.

* **r n**

The **r n** internal shell command should execute the **nth** command from the history list.  See details below how this should work.

1. **Rrequirements**

Your shell must support the following:

1. The command **exit** is an internal shell command.  It will cause your shell program to quit.
2. The internal shell command "hist" which displays (up to) the ten most recent commands typed into your shell. The “hist” command itself will be added to the history.  The commands will always be numbered from 1 to 10 for reference and display as a list. Notice that you need to implement a data structure to keep up to 10 most recent commands in it.
3. All commands (internal or external, successful or unsuccessful, except **r and exit**) are to be added to the history list.
4. The internal shell command "r n" where 'n' is the history number of that command in the history list. Also, the user should be able to run the most recent command again by just entering 'r'. You can assume that one space will separate the 'r' and the number of the command. Any command that is executed in this fashion should be echoed on the user's screen and the command is also placed in the history list as the next command.  The r command is not added to the history **list** but the referenced command is added to the history list.
5. Take some creative liberty is designing your shell prompt.  You could include the name of the shell, current working subdirectory, command number, etc.
6. You must check and correctly handle all return values.  This means that you need to read the **man** pages for each function to understand what the possible return values are, what errors they indicate, and what you must do when you get that error.
7. **You may not use the "system()" system function in your shell**.  However, this function call is good for prototyping and debugging.
8. **Getting Help**

You are expected to do your own work and research regarding the usage of various system calls, header files, and libraries.  Do not hesitate to ask a question if you are unclear about how some part of the assignment is supposed to work.

**Submitting Your Work**

**What should be submitted:**

Create a directory ***with your* username appended with an underscore followed by the word proj1** (Ex. For a userid dkar, the name of the directory will be dkar\_proj1). Include the following files in the directory:

1. Create a *Readme* file describing the contents of the directory.  Also include any documentation explaining the solution to the problem or answer to any questions in the *Readme* file. If you download and use some code from online sources, provide a list of references in the *Readme* file.

2. Create a *Makefile* with rules which will perform the following tasks:

a) Compile all code.

b) Clean up the directory, removing core files and object files.

c) Perform any other required tasks.

3. All source code required for the problem.

4. Any test data or configuration files required for the problem.

**How should be submitted**

1. Your work should be submitted as a **zip file** on Blackboard. This file should restore the files in a directory, **named after your username, an underscore**, and the project number (Ex: dkar\_proj1) containing the appropriate files and/or answers to the question(s), if any.
2. Unless otherwise noted in the assignment notice, your solution must compile/run/work on **riddler.tamucc.edu.**