TCL and GNU Readline

Christos P. Sotiriou

Contents

- TCL Library and C API
  - https://www.tcl.tk/
  - https://www.tcl.tk/man/tcl8.5/TclLib/contents.htm
- GNU Readline API
  - http://www.gnu.org/software/readline/

About TCL/TK

- TCL (Toolkit Command Language)
  - Web and desktop applications, network programming, embedded development, testing, general purpose programming, system administration, database work, and many, many more
- Dynamic, String-oriented Language
- Tk Graphical Toolkit
  - GUIs that are incredibly simple yet remarkably powerful
  - Tk canvas widget makes it easy to create displays with graphics,
  - Including powerful facilities such as bindings and tags
  - text widget provides sophisticated hypertext capabilities and more.
- Rapid Software Development

TCL Basics - 1

- Commands separated by semicolons or newlines
  - expr 20 + 10
  - tclsh is a basic TCL Shell
- Variables
  - set x 32
  - expr $x*3
- Command Substitution
  - set cmd expr
  - set x 11
  - $cmd $x*$x
  - set a 44
  - set b [expr $a*4]

TCL Basics - 2

- Quotes and Braces
  - Double-quotes allow you to specify words that contain spaces
  - set x 24
  - set y 18
  - set z "$x + $y is [expr $x + $y]“
  - z will have the value 24 + 18 is 42
- Quotes and Braces
  - (a) command and variable substitutions are performed on the text between the quotes
  - (b) the quotes themselves are not passed to the command
- Curly Braces
  - no substitutions are performed on the text between the curly braces
  - set z {$x + $y is [expr $x + $y]}
  - This command sets variable z to the value "$x + $y is [expr $x + $y]"

TCL Basics - 3

- Control Structures
  - Tcl provides a complete set of control structures including commands for conditional execution, looping, and procedures
  - Tcl control structures are just commands that take Tcl scripts as arguments
  - The example below creates a Tcl procedure called power, which raises a base to an integer power:
  - proc power {base p} {
    set result 1
    while {$p > 0} {
    set result [expr $result * $base]
    set p [expr $p - 1]
    }
    return $result
    }

TCL Basics – 4

- Examples of using proc
  - power 2 6
  - power 1.15 5
- Tcl commands are created in **three** ways
  - One group of commands is provided by the Tcl interpreter itself
    - These commands are called *builtin commands*
    - The builtin commands are present in all Tcl applications
  - The second group of commands is created using the Tcl extension mechanism
    - Tcl provides APIs that allow you to create a new command by writing a *command procedure* in C or C++ that implements the command
    - You then register the command procedure with the Tcl interpreter by telling Tcl the name of the command that the procedure implements
    - In the future, whenever that particular name is used for a Tcl command, Tcl will call your command procedure to execute the command
  - The third group of commands are those defined in TCL
    - By the proc command

TCL Basics – 5

- Other Features
  - More control structures, such as if, for, foreach, and switch
  - String manipulation, including a powerful regular expression matching facility.
    - Arbitrary-length strings can be passed around and manipulated just as easily as numbers.
  - I/O, including files on disk, network sockets, and devices such as serial ports
  - Tcl provides particularly simple facilities for socket communication over the Internet
  - File management: Tcl provides several commands for manipulating file names, reading and writing file attributes, copying files, deleting files, creating directories, and so on.
  - Subprocess invocation: you can run other applications with the exec command and communicate with them while they run.
  - Lists: Tcl makes it easy to create collections of values (lists) and manipulate them in a variety of ways.
  - Arrays: you can create structured values consisting of name-value pairs with arbitrary string values for the names and values.
  - Time and date manipulation.
  - Events: Tcl allows scripts to wait for certain events to occur, such as an elapsed time or the availability of input data on a network socket.

TCL C API

- http://www.tcl.tk/man/tcl8.5/TclLib/contents.htm
- Tcl_FindExecutable()
- Tcl_CreateInterp()
- Tcl_CreateObjCommand()
- Tcl_Eval()

GNU Readline

- The GNU Readline Library
  - Two Interfaces
    - Standard Interface – Control passed to readline()
      - char *line = readline ("Enter a line: ");
    - Alternative Interface – Event-based
- The GNU History Library
  - Command History Management
    - add_history (line);
- The GNU Readline User Interface

GNU Readline Standard Interface Example

GNU Readline Standard Interface Example – page 1

```
int main()
{
  char *text = NULL; // readline result //
  char *textexpansion; // readline result history expanded //
  int expansionresult;
  HIST_ENTRY **the_history_list; // readline commands history list - NULL terminated //
  char command[LINE_MAX]; // current command //
  unsigned long i;

  // Readline Initialisation //
  rl_completion_entry_function = NULL; // use rl_filename_completion_function(), the default filename completer //
  rl_attempted_completion_function = custom_completer;
  rl_completion_append_character = '\0';

  using_history(); // initialise history functions //
  while (1)
    {
      text = readline("PR> ");
      if (text != NULL)
              {
                expansionresult = history_expand(text, &textexpansion);
                if ((expansionresult == 0) || // no expansion //
                    (expansionresult == 2)) // do not execute //
                  {
                    add_history(text);
                    strcpy(command, text); // store command //
                  }
                else
                  {
                    add_history(textexpansion);
                    strcpy(command, textexpansion); // store command //
                  }
                free(textexpansion);
                free(text);
              }
```


GNU Readline Standard Interface Example

GNU Readline Standard Interface Example – page 2

```
...
      // handle two basic commands: history and quit //
      if (strcmp(command, "quit") == 0)
              {
                return EXIT_SUCCESS;
              }
      else if (strcmp(command, "history") == 0)
              {
                the_history_list = history_list(); // get history list //
                if (the_history_list != NULL)
                  {
                    i = 0;
                    while (*(the_history_list + i) != NULL) // history list - NULL terminated //
                      {
                        printf("%d: %s\n", (i + history_base), (*(the_history_list + i))->line);
                        i++;
                      }
                  }
              }
    }
}
```