CE437: Αλγόριθμοι CAD I
Homework 1
Tcl shell’s Implementation

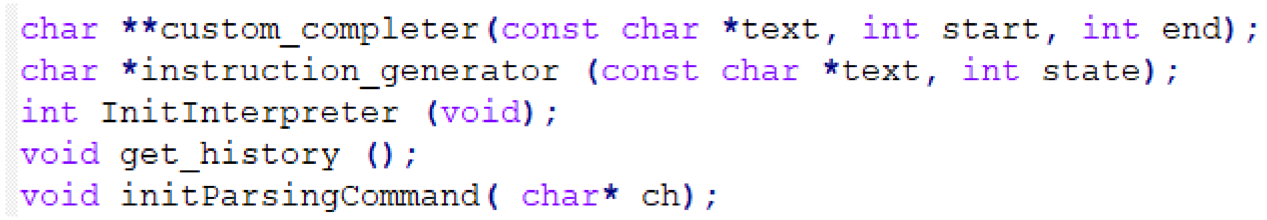
```
C:\Users\Christos Vasileiou> tcl shell
```

By Vasileiou Christos, 1983

Files’ Structure

- customTCL.c: Included function’s implementation and main thread.
  - char **custom_completer(const char *text, int start, int end);
  - char *instruction_generator (const char *text, int state);
  - int InitInterpreter (void);void get_history ();
  - void initParsingCommand(char* ch);
  - int main(int argc, char *argv[])
- Instructions.h: Included Tcl instructions in a string array.
  - static char *instructions[]
- Makefile: Linking and Compilation.
  - gcc -g customTCL.c Instructions.h -o customTCL $(LDFLAGS) $(DIRS)

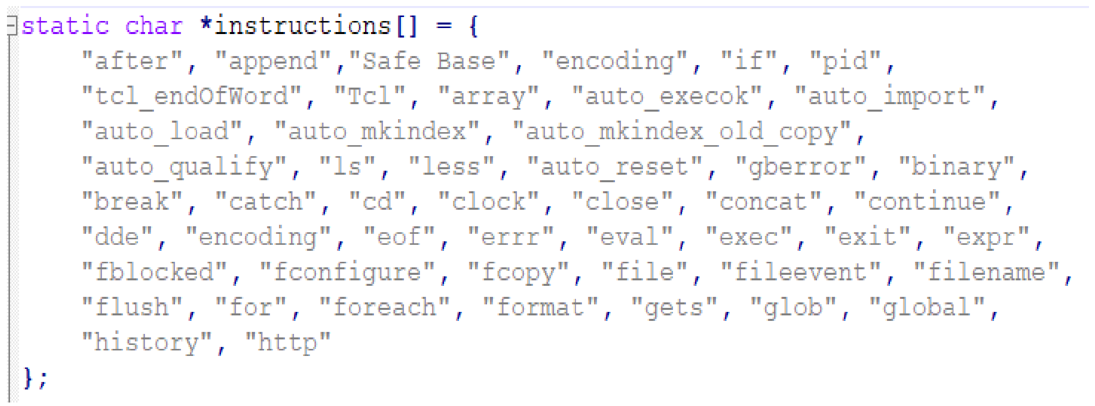
Files’ Structure



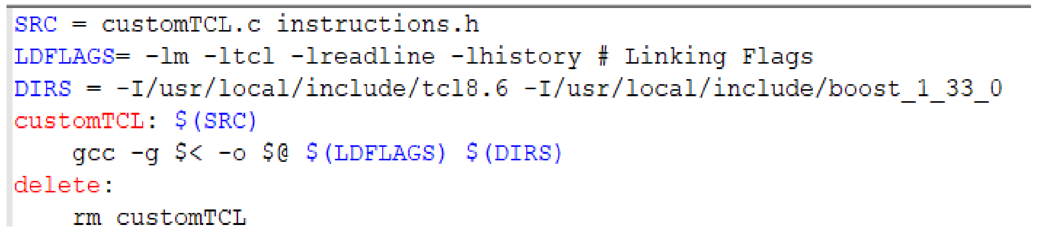
```
char **custom_completer(const char *text, int start, int end);
char *instruction_generator (const char *text, int state);
int InitInterpreter (void);
void get_history ();
void initParsingCommand( char* ch);
```

Functions’ prototypes

String array with instructions:



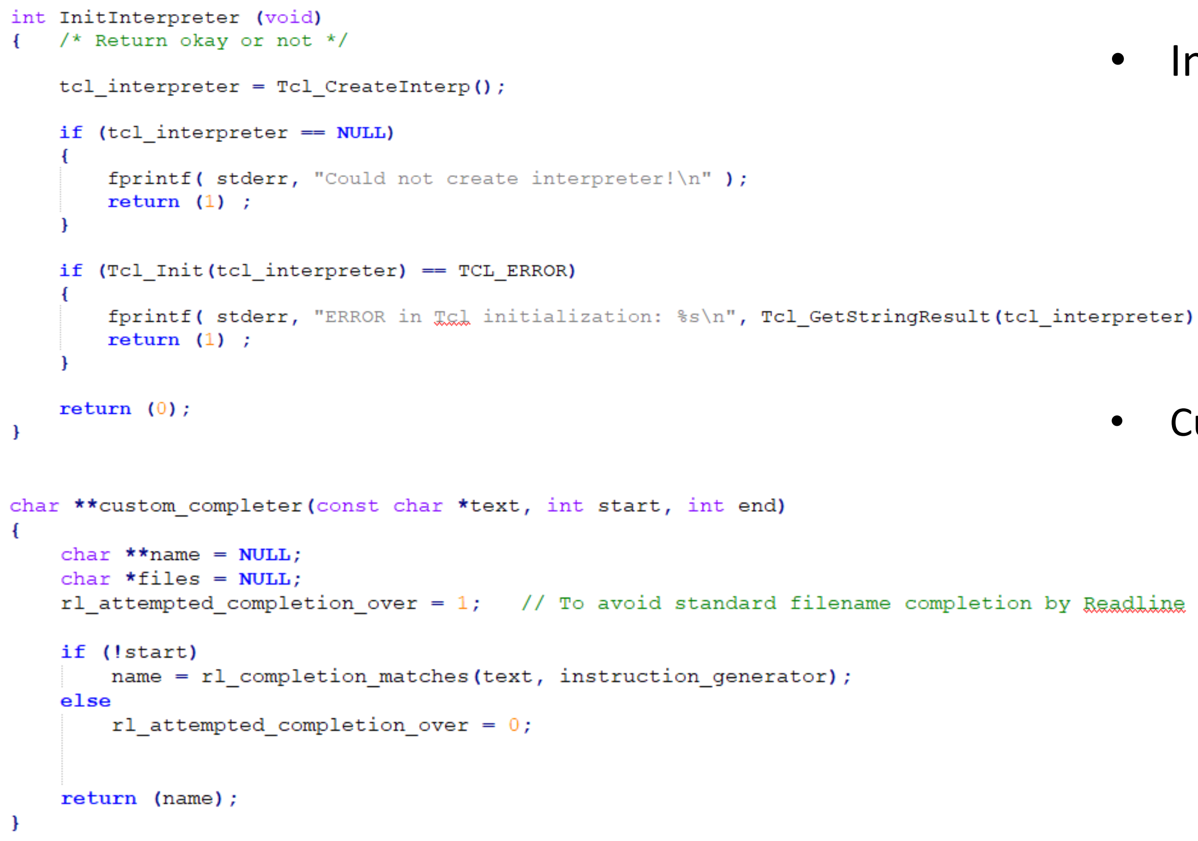
```
static char *instructions[] = {
    "after", "append","Safe Base", "encoding", "if", "pid",
    "tcl_endOfWord", "Tcl", "array", "auto_execok", "auto_import",
    "auto_load", "auto_mkindex", "auto_mkindex_old_copy",
    "auto_qualify", "ls", "less", "auto_reset", "gberror", "binary",
    "break", "catch", "cd", "clock", "close", "concat", "continue",
    "dde", "encoding", "eof", "errr", "eval", "exec", "exit", "expr",
    "fblocked", "fconfigure", "fcopy", "file", "fileevent", "filename",
    "flush", "for", "foreach", "format", "gets", "glob", "global",
    "history", "http"
};
```



```
SRC = customTCL.c instructions.h
LDFLAGS= -lm -ltcl -lreadline -lhistory # Linking Flags
DIRS = -I/usr/local/include/tcl8.6 -I/usr/local/include/boost_1_33_0
customTCL: $(SRC)
    gcc -g $< -o $@ $(LDFLAGS) $(DIRS)
delete:
    rm customTCL
```

Makefile

customTCL’s Functions

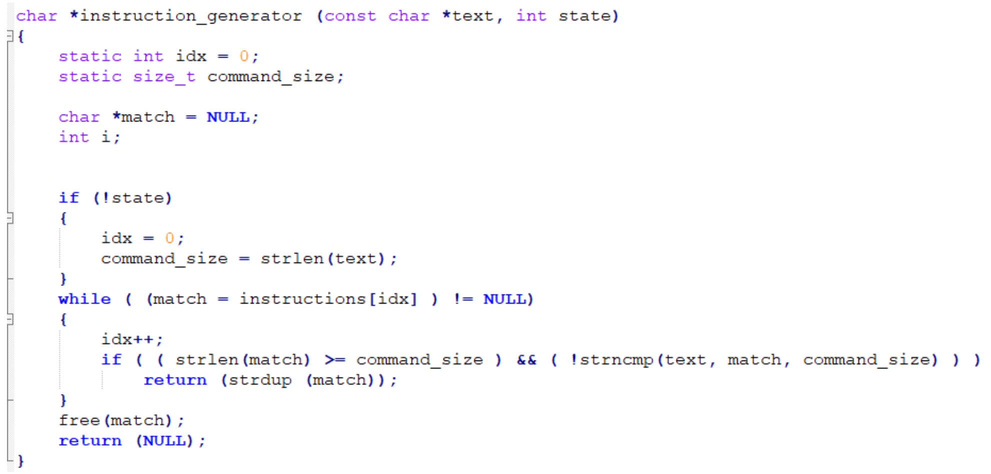


```
int InitInterpreter (void)
{   /* Return okay or not */

    tcl_interpreter = Tcl_CreateInterp();

    if (tcl_interpreter == NULL)
    {
        fprintf( stderr, "Could not create interpreter!\n" );
        return (1) ;
    }

    if (Tcl_Init(tcl_interpreter) == TCL_ERROR)
    {
        fprintf( stderr, "ERROR in Tcl initialization: %s\n", Tcl_GetStringResult(tcl_interpreter) );
        return (1) ;
    }

    return (0);
}


char **custom_completer(const char *text, int start, int end)
{
    char **name = NULL;
    char *files = NULL;
    rl_attempted_completion_over = 1;   // To avoid standard filename completion by Readline

    if (!start)
        name = rl_completion_matches(text, instruction_generator);
    else
        rl_attempted_completion_over = 0;


    return (name);
}
```

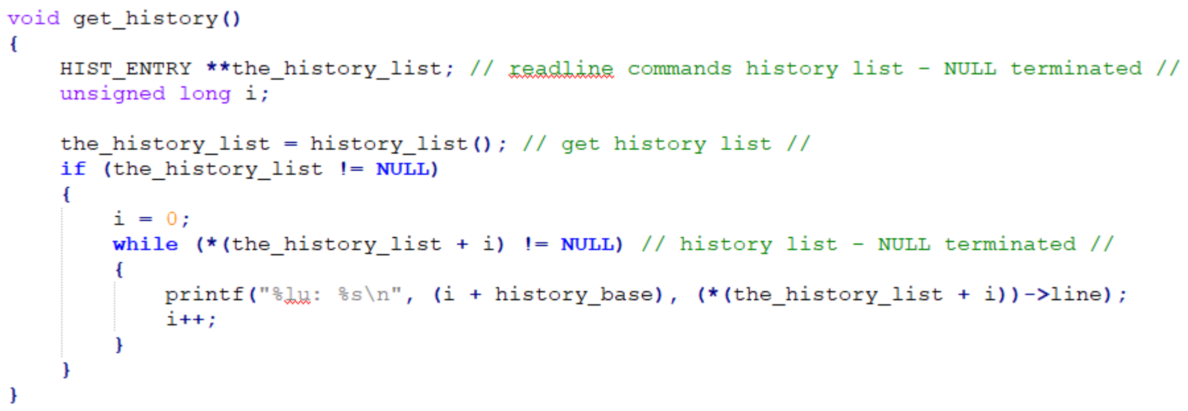
- InitInterpreter:
  - In case of successful initialization function returns 0, otherwise 1.

- Custom_completer:
  - Is responsible for completing words while tab is hitted.
  - When rl_attempted_completion_over is set to 1 function instruction_generator is called, otherwise standard file completion is enabled by Readline.

customTCL's Functions



```
char *instruction_generator (const char *text, int state)
{
    static int idx = 0;
    static size_t command_size;

    char *match = NULL;
    int i;


    if (!state)
    {
        idx = 0;
        command_size = strlen(text);
    }
    while ( (match = instructions[idx] ) != NULL)
    {
        idx++;
        if ( ( strlen(match) >= command_size ) && ( !strncmp(text, match, command_size) ) )
            return (strdup (match));
    }
    free(match);
    return (NULL);
}
```

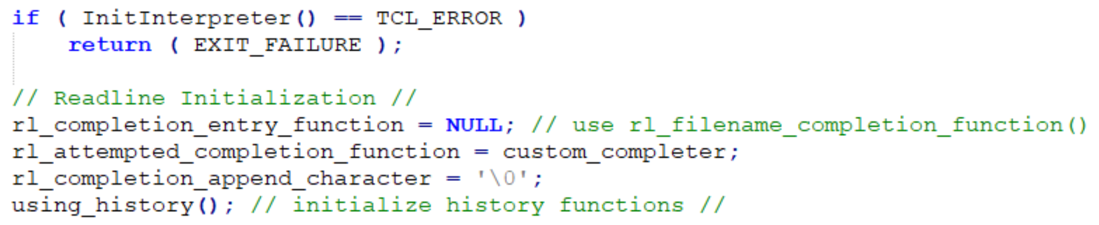


```
void get_history()
{
    HIST_ENTRY **the_history_list; // readline commands history list - NULL terminated //
    unsigned long i;

    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("%lu: %s\n", (i + history_base), (*(the_history_list + i))->line);
            i++;
        }
    }
}
```

- Instruction_generator:
  - Searches in instructions array in order to find a specific command of tcl.

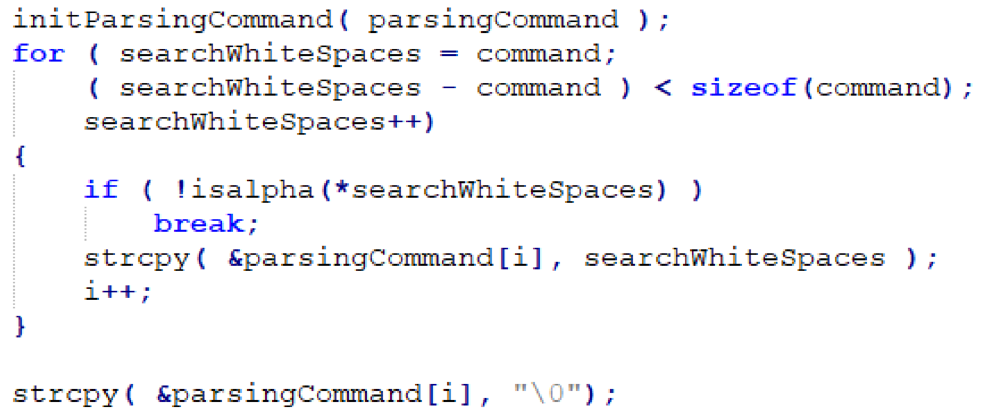
- get_history:
  - Prints all the commands that have been executed until the time it was called.

customTCL's Functions

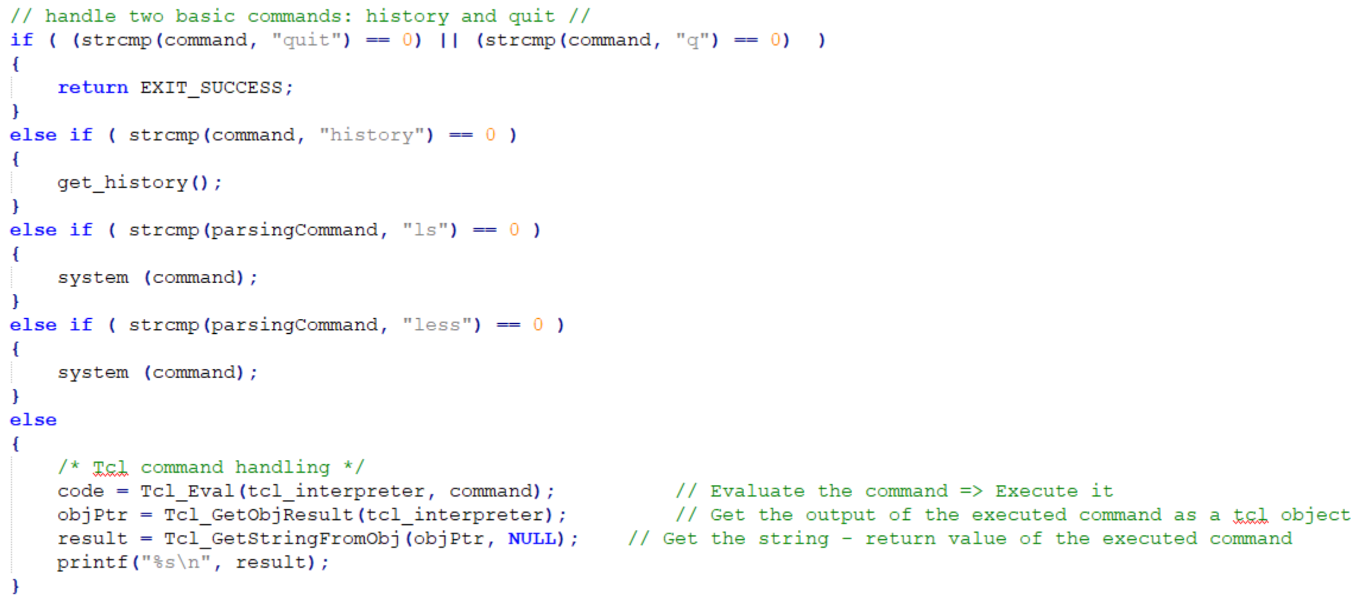


```
if ( InitInterpreter() == TCL_ERROR )
    return ( EXIT_FAILURE );

// Readline Initialization //
rl_completion_entry_function = NULL; // use rl_filename_completion_function()
rl_attempted_completion_function = custom_completer;
rl_completion_append_character = '\0';
using_history(); // initialize history functions //
```

- Main:
  - initialize all parameters of readline and history library.
  - Parsing command in order to print in bash shell.



```
initParsingCommand( parsingCommand );
for ( searchWhiteSpaces = command;
    ( searchWhiteSpaces - command ) < sizeof(command);
    searchWhiteSpaces++)
{
    if ( !isalpha(*searchWhiteSpaces) )
        break;
    strcpy( &parsingCommand[i], searchWhiteSpaces );
    i++;
}

strcpy( &parsingCommand[i], "\0");
```



```
// handle two basic commands: history and quit //
if ( (strcmp(command, "quit") == 0) || (strcmp(command, "q") == 0)  )
{
    return EXIT_SUCCESS;
}
else if ( strcmp(command, "history") == 0 )
{
    get_history();
}
else if ( strcmp(parsingCommand, "ls") == 0 )
{
    system (command);
}
else if ( strcmp(parsingCommand, "less") == 0 )
{
    system (command);
}
else
{
    /* Tcl command handling */
    code = Tcl_Eval(tcl_interpreter, command);          // Evaluate the command => Execute it
    objPtr = Tcl_GetObjResult(tcl_interpreter);         // Get the output of the executed command as a tcl object
    result = Tcl_GetStringFromObj(objPtr, NULL);    // Get the string - return value of the executed command
    printf("%s\n", result);
}
```

End of presentation.

Thank you!