Compile Options

- include:

```
#include <tcl.h>
#include <tk.h>
```

- compile options under g++:

```
-I/software/tcl-7.4/include
-I/software/tk-4.0/include
-I/software/x11r5_dev/Include
-L/software/tcl-7.4/lib
-L/software/tk-4.0/lib
-L/software/x11r5_dev/lib
-ltk
-ltcl
-lX11
```

Starting the Interpreter

- setup call to Tk_Main does not return and hands the control over to the interpreter

```
// prototype for the initialization function
int InitProc( Tcl_Interp *interp );

// declare an array for two strings
char *ppszArg[2];

// allocate strings and set their contents
ppszArg[0] = (char *)malloc( sizeof( char ) * 12 );
ppszArg[1] = (char *)malloc( sizeof( char ) * 12 );
strcpy( ppszArg[0], "Hello World" );
strcpy( ppszArg[1], "./hello.tcl" );

// the following call does not return
Tk_Main( 2, ppszArg, InitProc );
```

Initializtion Function

- pass the Tk_Main function an initializer which is called before the interpreter starts – do call-back registration here

```
int InitProc( Tcl_Interp *interp ) {
    int iRet = Tcl_Init( interp );   // init tcl
    if( iRet != TCL_OK )
    {  fprintf( stderr, "Unable to Initialize tcl!\n" );
       return( iRet );
    } // end if

    iRet = Tk_Init( interp );   // init tk
    if( iRet != TCL_OK)
       return( iRet );

    return( TCL_OK );
} // end InitProc
```

Function Calls & Registration

- all functions to be registered have to have the following prototype:

```
int Myfunc( ClientData Data, Tcl_Interp *pInterp, int argc, char *argv[] );
```

- call-back resgistration looks like:

```
Tcl_CreateCommand( interp, "hey_there", Myfunc, (ClientData)NULL,
    (Tcl_CmdDeleteProc *)NULL );
```

Variable Access

- getting a variable:

```
set say_hello_to "World"
```

⋮

```
char sHelloTo[30];

// after this call sHelloTo should contain "World"
strncpy( sHelloTo, Tcl_GetVar( pInterp, "say_hello_to", 0 ), 29 );
```

- setting a variable:

```
Tcl_SetVar( pInterp, "say_hello_to", "World", 0 );
```