CSE 3320 Operating Systems
Fall 2017, © DL, UTA, 2017

Programming Assignment 1
Introduction to OS Calls, Shell
Due: On Blackboard September 10, 2017

Description:
You will develop a small "text-based shell utility" ("ts") for Unix (or similar OS). A common complaint of the most-used UNIX (text) shells (Bourne, Korn, C) is that it is difficult to remember (long) file names, and type "long" command names. A "menu" type shell allows a user to "pick" an item (file or command)from a "menu".

You will display a simple menu, the following is a suggestion (you may change format):

```
Current Working Dir: /home/os.progs/Me
It is now: 28 August 2017, 10:58 PM

Files:          0. ts.c
                1. a.out
                2. ts.txt
                3. assignment1
                4. program2.c

Directories:    0. ..
                1. my_dir

Operation:      E  Edit
                R  Run
                C  Change Directory
                S  Sort Directory listing
                Q  Quit
```

You will read a single key "command" entered by the user, then a file or directory number or partial file name with “completion” and ts will output some system information on the terminal, or run a program (by means of systems calls).
The commands you will implement:
- Edit – opens a text editor with a file.
- Run - runs an executable program. You should handle parameters.
- Change – changes working directory.
- Sort – sorts listing by either size or date (prompt user)

You will need to implement:
- Prev, Next, and Operations need to be merged (menu fits on one screen).
- Store names in an array, Only one pass through directory.

If the "ts" program has an argument, use it as the directory starting point (working dir), like: "./ts /bin".
You should provide a reasonable user interface. (See below)
You may write this in ADA, Assembler, C, C++ or Java.(Others with permission)
You should target ts to Unix, MacOS, (or similar), or discuss alternatives with instructor.

Constraints:
- How many files and directories can you handle (max 1024)
- How long is a file name (max 2048 characters, really: limits.h, NAME_MAX)

Bonus:
- Show additional file information (date, size, read/execute)
- Use file name completion as well as a number.
- Add a pull-down menu using terminal codes or curses (ncurses).

Please, Submit ONLY to Blackboard.
All work must be your own, you may reference web sites, books, or my code but You MUST site the references.

You must submit this lab, working (or partially) by the due date.
You may (optionally) demonstrate this lab, working (or partially) to the GTA before the due date.

Your program should be well commented and documented, make sure the first
few lines of your program contain your name, this course number, and the lab number.
Your comments should reflect your design and issues in your implementation.
Your design and implementation should address error conditions:
what if an illegal command is entered or misspelled, what if a path can not be
reached, or an "executable file" is not executable.

Hints: (you may use this code)

```
/*  Some example code and prototype -
    contains many, many problems: should check for return values
    (especially system calls), handle errors, not use fixed paths,
    handle parameters, put comments, watch out for buffer overflows,
    security problems, use environment variables, etc.                  */

#include <sys/types.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <dirent.h>
#include <string.h>
#include <time.h>
#include <curses.h>

int main(void) {
    pid_t child;
    DIR * d;
    struct dirent * de;
    int i, c, k;
    char s[256], cmd[256];            /* fixed length buffers? */
    time_t t;

    while (1) {
      t = time( NULL );
      printf( "Time: %s\n", ctime( &t ));

      getcwd(s, 200);      /* why 200? What if bigger? Check for errors? */
      printf( "\nCurrent Directory: %s \n", s);

      d = opendir( "." );            /* errors? More below */
      c = 0;
      while ((de = readdir(d))){
          if ((de->d_type) & DT_DIR)
             printf( " ( %d Directory:  %s ) \n", c++, de->d_name);
      }
      closedir( d );
      d = opendir( "." );
      c = 0;
      while ((de = readdir(d))){
          if (((de->d_type) & DT_REG))
             printf( " ( %d File:  %s ) \n", c++, de->d_name);
          if ( ( c % 8 ) == 0 ) {
             printf( "Hit N for Next\n" );    /* What if only subdirs? */
             k = getchar( );    }
      }
      closedir( d );
      printf( "----------------------------------------\n" );
      c = getchar( ); getchar( );
      switch (c) {
        case 'q': exit(0); /* quit */
        case 'e': printf( "Edit what?:" );
                  scanf( "%s", s );
                  strcpy( cmd, "pico ");
                  strcat( cmd, s );
                  system( cmd );    /*this is bad, should use fork() then execv() or execl() */
                  break;
        case 'r': printf( "Run what?:" );
                  scanf( "%s", cmd );
                  system( cmd );
                  break;
        case 'c': printf( "Change To?:" );
                  scanf( "%s", cmd );
                  chdir( cmd );   /* what can go wrong ?  */
                  break;
      }
    }
}
```