CSC 3350 Systems Programming
Lab 4 – S_MAKE: Simple MAKE Utility
Due: Wednesday, May 11, 2016 in class

Write a C Console application **S_MAKE** that implements a MAKE-like utility (similar to the old-style Microsoft MAKE utility).

S_MAKE is invoked by specifying a Script filename on the command line:

```
S_MAKE     project-script-filename
```

If no script filename is given, default to a file named "PROJECT" (no filename extension).

The project script file is an text file that specifies a collection of targets (and their dependencies), and a list of commands to be executed to re-make those targets. In particular,

Target/Dependent Lines:
- begin in column one, with the target filename.
- Following the target name (freeformat, on the same line) is a colon (:) and a list (optionally empty) of filenames on which the target is dependent.

Command Lines:
- Immediately follow a target line.
- Begin with at least one white-space character.
- Specify a program command line to be executed. The command must be the name of an executable file to be run using the CreateProcess syscall.
- A target-line can be followed by several command lines.
- A blank line (or end of file) indicates the end of the sequence of command lines.

Blank Lines or **Comment Lines** (line with # in column one):
- Can appear before new target-lines, or after the last command line for a target.

Simple EXAMPLE of a makefile script:

```
x.obj:     x.c    my_pgm.h
     cl    /c   x.c

y.obj   :   y.c   my_pgm.h
     cl    /c   y.c



my_pgm.obj:   my_pgm.c    my_pgm.h
     cl    /c   my_pgm.c

my_pgm.exe  :    x.obj     y.obj     my_pgm.obj
     cl    my_pgm.obj    x.obj    y.obj
     attrib  +R   my_pgm.exe
```

Description:

For each target section, S_MAKE must determine if any of the following conditions apply:

- the target file itself doesn't exist,
- any of the dependent files has a more recent date or time of last modification than the target,
  Note that all specified dependent files on a target line must exist in the file system;
  otherwise the script is in error and S_MAKE should be terminated.
- no dependent files are specified on the target line,

If the target is "out of date" or needs to be rebuilt, then any subsequent command line(s) in that target section are to be executed one at a time.

The SMAKE program is responsible for properly executing one after the other whatever command lines are specified in the script for a target, assuming that the appropriate command environment has already been configured by the user. As command lines are executed, SMAKE must wait for each to complete and retrieve the exit status result code. If an executed (i.e., spawned) commandline returns a non-zero result code (indicating that the execution has failed), S_MAKE should be terminated at that point with a suitable error message.

Note that the example script references the "cl" command, but this is simply an example that represents a typical scenario in which a program like SMAKE might be used by a programmer to automate a "recompile" script when the programmer is working from the command line instead of within an IDE like Visual Studio. In this scenario, it is the programmer's responsibility prior to using SMAKE to configure the computer system to be capable of executing the "cl" program directly from the commandline (a setup which is not automatically configured when Visual Studio is installed).

To understand the intent of the targets and commands in the example script,

- the **x.obj** target section indicates that the x.obj target file is dependent on both x.c and my_pgm.h.
  This target section has a single command line
  ```
  cl   /c   x.c
  ```
  If executed, this would cause x.c to be compiled with the cl command, which would produce x.obj.
- similar analysis for **y.obj** and **my_pgm.obj**.
- the **my_pgm.exe** target is dependent on 3 files, x.obj, y.obj and my_pgm.obj. If my_pgm.exe is out-of-date, then both of the subsequent command lines must be executed, one after the other. Both the cl command line (causing the linkage of the three .obj files and producing the output file my_pgm.exe), and the following "attrib" command line (making the executable ReadOnly) are to be executed.

For this lab, you must develop suitable SMAKE script(s) that enable you to test out the functionality of your implementation. It is **_not_** recommended that you use a script that invokes commands like "cl" – this is a complex command requiring significant environmental setup, and unless you are already familiar with the use of "cl", you will probably have difficulty determining if it is being invoked and executed correctly. Your test makefile scripts do not really have to relate to "software project rebuild" scenarios.

Instead, you should create a testing subdirectory that contains various files (targets and dependents with varying time stamps) and a few small .exe commandline executable programs that you are familiar with (such as hexdump.exe or mydir.exe). Then design various script(s) using these files to ensure that your SMAKE program works properly, e.g., scripts that specify particular combinations of "target file" and "dependent file(s)" where the time/date stamps sometimes cause the related command lines to be executed and sometimes cause them to be skipped. You might find it useful to create a small test program like "cmdargs.c" that simply prints out its commandline arguments, and use it as an example commandline to execute for some targets.

Turn in:

- ZIP file containing your entire Visual Studio project and source code for the lab. **Before ZIPPING**, please place a copy of your lab EXECUTABLE (e.g., S_MAKE.exe) in your top-level solution directory. Submit using Blackboard file attachment for this lab assignment.
- Printed source code listing.
- Lab summary report:
  - Include brief instructions for running your lab.
  - Include a brief 1-2 paragraph summary statement that describes how well the lab works, and what you learned in doing the lab.