Makefile

- % - it matches one or more characters in a string. This match is called the stem.

Automatic Variables

- These variables have values computed afresh for each rule that is executed, based on the target and prerequisites(dependancies) of the rule.
- The scope of automatic variable is limited to only single rule. They only have values within the recipe.
- Cannot be used them anywhere within the target list of a rule (Dependancy line).
- $@
  - The file name of the target of the rule.
  - is the name of whichever target caused the rule's recipe to be run.
- $<
  - The name of the first prerequisite.
- $^
  - The names of all the prerequisites, with spaces between them.

GCC - GNU C Compiler

- Set of tools/programs used to compile C program.
- These tools are used to develop C programs and SDK (Software Development Kit).
- Many of these tools are used in sequence and also called as tool-chain.
- Tools
  - Pre-processor (cpp)
  - Compiler (cc1)
  - Assembler (as)
  - Linker (ld)
  - Debugger (gdb)
  - Objdump (objdump)
  - etc.
- "gcc" is front-end for compilation & linking tools.
- gcc internally invokes Pre-processor, Compiler, Assembler and Linker.
  - gcc -E --> Pre-procssor
  - gcc -c --> Compiler
  - gcc -S --> Assembler
  - gcc --> Linker

"gcc" options

- -o output_file --> give output file name.
- -E --> show Pre-procssor output
- -c --> Compile only (.o)
- -S --> Create assembly output file (.s)
- -std --> specify C standard
  - -std=c89 --> ANSI standard
  - -std=c99 --> ANSI standard

-
  - -std=gnu89 --> C89 with GNU extensions
  - -std=gnu99 --> C99 with GNU extensions (used in Linux device driver development)
- -g --> Debugger level (Higher level --> Higher debug info --> Higher .out file size)
  - -ggdb1
  - -ggdb2 (default)
  - -ggdb3
- -Wxxx --> Warning flags
  - -Wall --> show all warnings
  - -Werror --> treat warning as error (do not create .out file)
- -Ox --> Optimization
  - -O0 --> No optimization
  - -O1
  - -O2
  - -O3 --> Highest optimization
  - -Os --> Optimization for size (compact low level code generated)
- -D --> define symbol, symbolic constant or macro
  - -DPI=3.142
  - -D'BV(n)=(1<<(n))'
- -I --> Include standard dir path.
  - -I/usr/include --> find standard header files into "/usr/include"
  - -I. --> find standard header files into current directory
  - #include <file.h> --> will be searched in standard directory (or -I dirpath)
- -L --> Library standard dir path
  - Standard library: libc.so (by default linked) --> -lc
  - Math library: libm.so (need to link separately) --> -lm
  - Standard libraries are available in /usr/lib (depends on Linux).
  - -L/usr/lib --> file .so/.a files into "/usr/lib".

Debugging

- Debugging is process of finding bugs (logical errors) in the programs.
- It also helps understanding flow of execution of the program.
- Debugger needs symbol & source code info to be present in executable file.
  - Need to compile program so that debugging can be done.
  - -g --> enable debugging (add symbol & source code info in executable file).
- Debugger enable executing the program step by step and monitor values of each variable.
- Debugger in GCC tool-chain is "gdb".

Debugging Steps

- step 1: Compile program to enable debugging.
  - gcc -g
  - Makefile: CFLGAS = -g
- step 2: Start debugger.
  - gdb main.out
- step 3: Give gdb commands to debug step by step.
  - Set a breakpoint (point from which you want to debug step by step).

    - break file.c line_number
    - break function_name
  - Start debugging process (it will auto stop on first breakpoints)
    - run
  - Execute step by step
    - next - execute the function but do not show fn code line by line (Step Over)
    - step - execute the function line by line (Step Into)
    - cont - execute directly till next breakpoint
  - Monitor variables
    - display varname - print var contents after each step
    - print varname - print var content once
    - Backtrace
  - Source code
    - list - show 10 lines of code
  - Stop debugging
    - quit