TEC | Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores

Introducción a los Sistemas Embebidos

Taller 3

Profesor:

Luis Alberto Chavarría Zamora

Estudiante:

Jose Ignacio Granados Marín

Grupo 1

Semestre II 2022

Investigación

¿Qué es GNU Make?

Según Sarker (2020) define GNU Make como una herramienta que ayuda a generar programas ejecutables y archivos no fuente a partir de un determinado código fuente. Además, Make obtiene su conocimiento de cómo crear los programas y ejecutables a partir de un archivo llamado Makefile, el cual enumera cada uno de los archivos no fuente y la forma de ejecutarlos a partir de otros archivos.

¿Cuáles son los componentes más importantes de un archivo Makefile?

Un artículo publicado por GNU menciona que un archivo Makefile puede contener:

- Reglas explícitas, las cuales indican cuándo y cómo rehacer uno o más archivos.
- Reglas implícitas, las cuales indican cuándo y cómo rehacer una clase de archivos en función de sus nombres.
- Variables, las cuales especifican ciertos valores que pueden ser sustituidos en cualquier momento.
- Directivas, las cuales corresponden instrucciones encargadas de realizar acciones específicas. Dentro de las directivas más comunes, se encuentran:
    - Lecturas de otro archivo Makefile.
    - Toma de decisión entre utilizar o ignorar una parte del archivo Makefile.
    - Definición de variables.
- Comentarios.

¿Cómo se define (asignaciones) y utilizan los macros dentro de un archivo Makefile? Brinde un ejemplo.

Un artículo publicado por Idera Inc, se refiere a un macro como un string que se expande cada vez que se llama desde un fichero Makefile. Dichas herramientas pueden representar nombres de ficheros, opciones del compilador, programas a ejecutar, directorios de búsqueda, directorios de escritura, entre muchos otros.

Por su parte, la sintaxis de una macro está dada por:

$$< MacroName > = < expansion\_text >$$

donde *MacroName* es nombre del macro y *expansion_text* representa la instrucción del macro. Algunos ejemplos de dichas definiciones, se muestran a continuación:

$$SOURCE = f1.cpp\ f2.cpp\ f3.cpp$$

$$CFLAGS = \$(DEBUG) - c$$

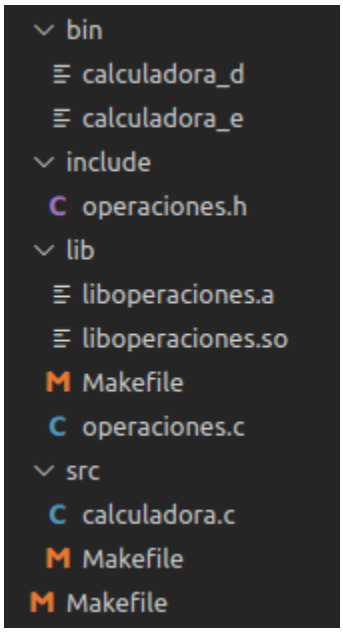
$$DEBUG = -g$$

¿Qué utilidad tienen los macros que hacen referencia a herramientas del toolchain?

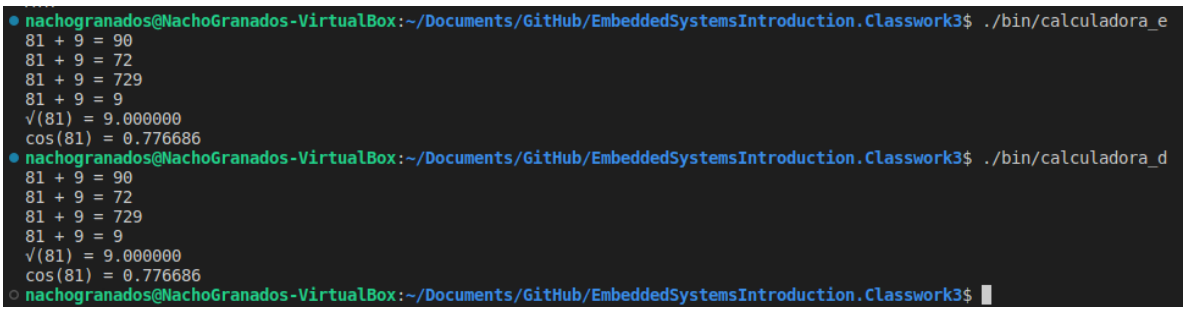
Un artículo publicado por CE Programming menciona que aquella definición de macros dentro de un archivo Makefile ofrecen la posibilidad de comprimir programas en un ejecutable autoextraíble.

Ejercicios prácticos

Distribución final de la biblioteca:



Ejecución de la biblioteca estática y dinámica respectivamente:



Referencias

CE Programming. (2017). Makefile Options — CE C/C++ Toolchain documentation. https://ce-programming.github.io/toolchain/static/makefile-options.html

GNU. (2015). Makefile Contents (GNU make). https://www.gnu.org/software/make/manual/html_node/Makefile-Contents.html#:%7E:text=Makefiles%20contain%20five%20kinds%20of,files%2C%20called%20the%20rule%27s%20targets.

Idera Inc. (2015, 13 abril). MAKE Macros - RAD Studio. Embarcadero. https://docwiki.embarcadero.com/RADStudio/Sydney/en/MAKE_Macros

Sarker, S. (2020, 19 enero). GNU Make Tutorial. Linuxhint. https://linuxhint.com/gnu-make-tutorial/