Hoja de Ruta Headers / Reglas de [Kieras, 2012]

Resumen v0.1 de R.O. 10/09/2017

1. Cada módulo con su .c /.h corresponde a una funcionalidad separada
2. Siempre usar "include guards" en los .h (`#ifndef MODULO1_H; #define MODULO1_H; …resto del header…; #endif`)
3. Todas las declaraciones necesaria para usar un módulo deben aparecer en su .h – Este .h se usa para acceder al módulo.
4. EL .h solo contiene DECLARACIONES (`struct, func() prototypes, extern var globals`), en el .c van las DEFINICIONES (de las `func(),` de las globals, y sus inicializaciones) + el módulo .c debe #incluir el .h
5. Las variables globales para todo el programa, se DECLARAN como extern en el .h (como dice la regla 4) y se DEFINEN e INICIALIZAN al principio del .c de ese módulo. Los otros módulos solo #incluyen el .h de ese módulo. (en .h va: `extern int pepe;` al principio del .c va: `int pepe=0;`)
6. Las variables y elementos de uso interno del .c deben sacarse del .h – se declaran al principio del .c como static. (ej. `pepe2` solo usado por tolaspi.c, se declara al principio de ese archivo como `static int pepe2 = 0`; el tolaspi.h ni se entera, y el resto del programa tampoco)
7. Cada archivo .h debe incluir solamente otros .h que sean requeridos para compilar correctamente, pero no mas. (ej. si las funciones de tolaspi.c requieren math .h, #incluilrlo al principio del .c, ni mencionar eso en el .h)
8. Si solo se requiere en tolaspi.h un único elemento X de cadorna.h, se puede declarar sólo ese elemento X en tolaspi.h y no cadorna.h completo. Esto se llama a veces "forward declaration"
9. El .h debería compilar correctamente por sí mismo. Conviene testear en un test.c incluyendo uno a uno todos los .h, a ver si compilan correctamente y declararon todas sus partes requeridas (esto evita extraños errores de dependencia, al modificar archivos)
10. Un archivo tolaspi.c debe #incluir al principio a tolaspi.h, y después cualquier otro archivo requerido
11. Nunca hacer un #include de un archivo .c !! (como en PWRC2 original FATFS)

REFERENCIAS:

[Kieras, 2012], David Kieras, EECS Dept. University of Michigan, 19/12/2012 – originalmente escrito para C++, pero utilizado para C indistintamente.