

TALLER DE LENGUAJES I

- Argumentos a la función *main*
- Compilación con *GCC*

Argumentos a la función *main*

Argumentos a la función `main`

- En los entornos que soportan al lenguaje C, existe una manera de pasarle valores a un programa que comenzará a ejecutarse. Estos valores se pasan en forma de argumentos a la función `main`.
- La utilidad de esta característica reside en la posibilidad de que el programa cambie su comportamiento en base a los argumentos pasados.

Argumentos a la función `main`

- La función `main` recibe dos argumentos:

```
int main (int argc, char * argv[])
```

- El segundo (por convención llamado `argv`) es un vector de cadenas de caracteres que contiene los argumentos.
- El primero (por convención llamado `argc`) es el número de argumentos pasados. En este caso, la dimensión del vector `argv`.

Ejemplo

- El siguiente programa imprime en pantalla los argumentos pasados a la función `main`:

```
#include <stdio.h>

int main(int argc, char *argv[]){
    int i;
    for (i = 0; i < argc; i++)
        printf("argv[%d] => %s\n", i, argv[i]);

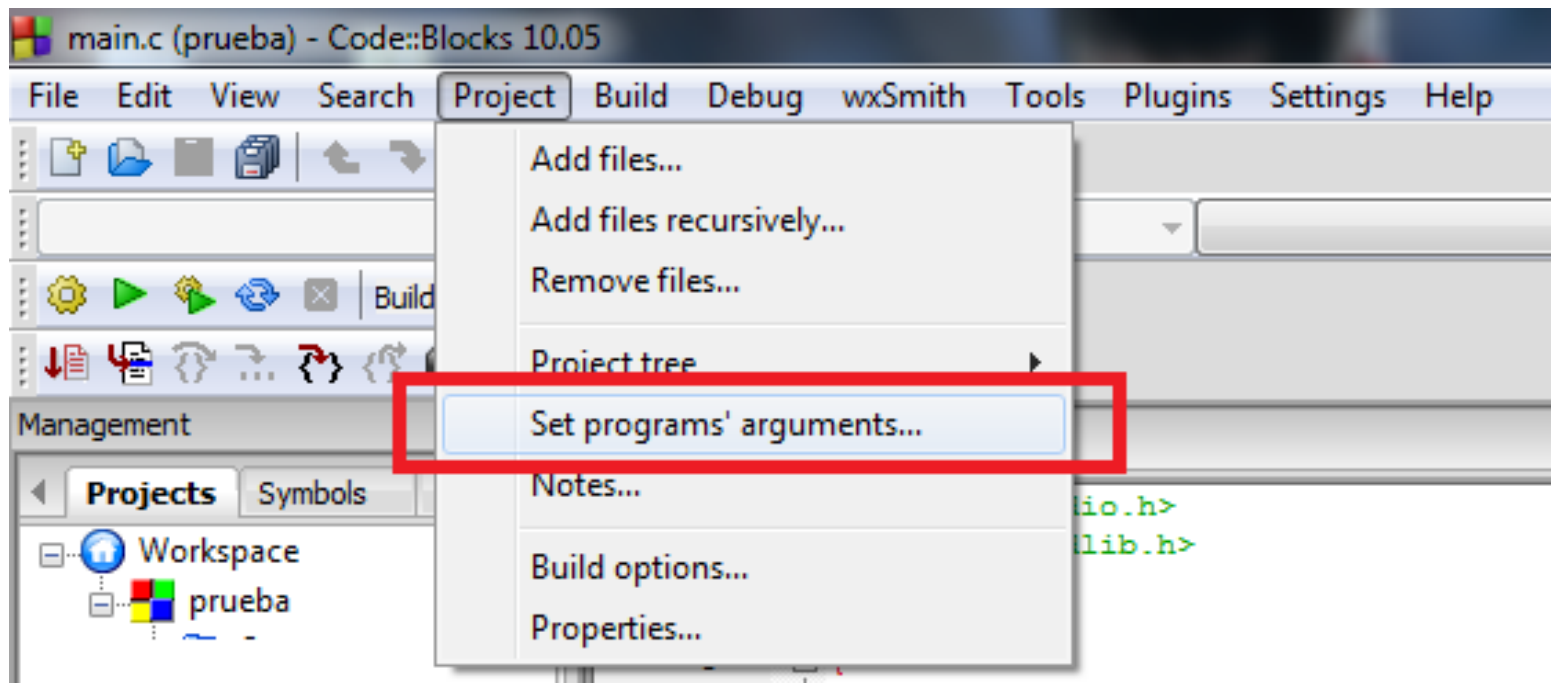
    return 0;
}
```

Argumentos a la función `main`

- Por convención `argv[0]` es la ruta completa al programa que se ejecuta, con lo cual `argc` vale por lo menos 1.
- Si `argc` es 1, entonces ningún argumento fue pasado al programa.
- Si `argc` es `n`, entonces el primer argumento opcional es `argv[1]` y el último `argv[n-1]`.
- Por último, el estándar establece que `argv[argc]` sea un puntero nulo.

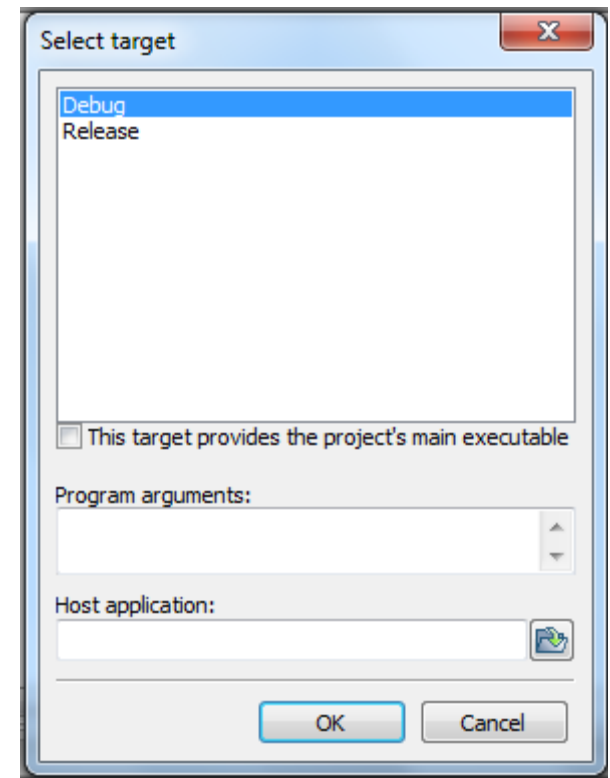
Argumentos al `main` en Code::Blocks

- Seleccionar la opción “Set programs’ arguments...” de la opción “Project” del menú superior.



Argumentos al `main` en Code::Blocks

- En primer lugar seleccione entre las opciones “Debug” y “Release” de acuerdo al modo en que esté trabajando. Luego ingrese los valores de los argumentos en la sección “Program arguments:” separados mediante un espacio. Por último, click en “OK”.

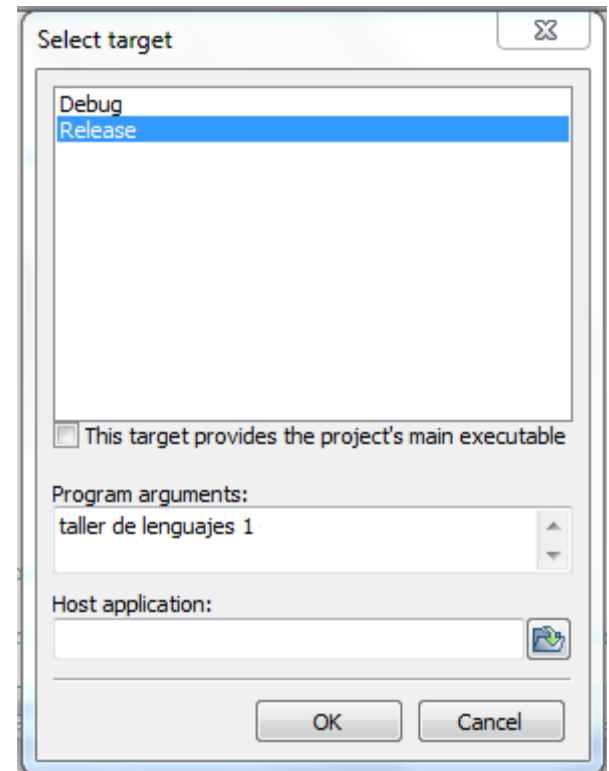


Ejemplo

Se pasan los argumentos:

- taller
- de
- lenguajes
- 1

al programa de la diapositiva 3.

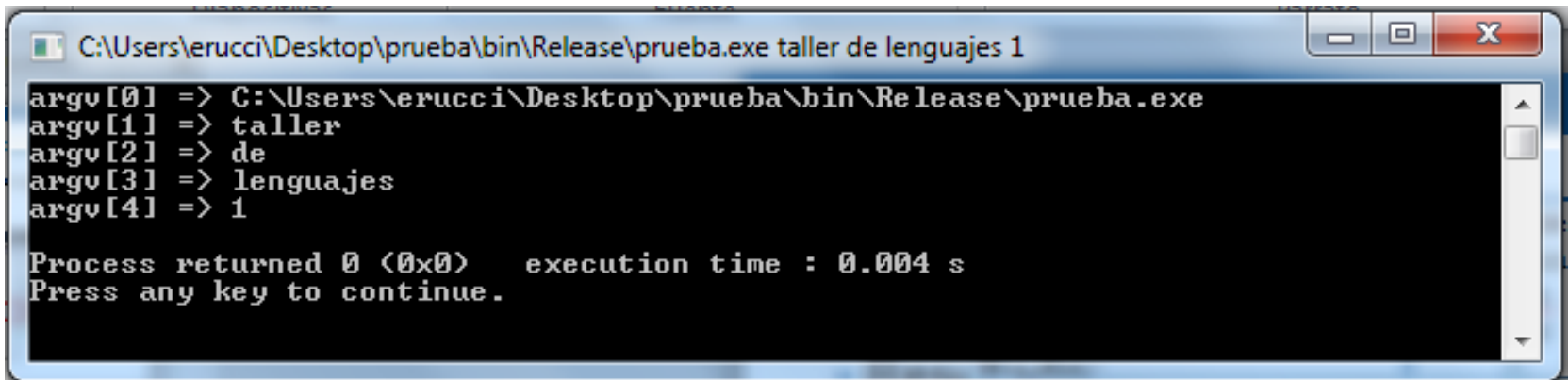


Ejemplo

- Resultado:

```
#include <stdio.h>
int main(int argc, char *argv[]){
    int i;
    for (i = 0; i < argc; i++)
        printf("argv[%d] => %s\n",i,argv[i]);

    return 0;
}
```



```
C:\Users\erucci\Desktop\prueba\bin\Release\prueba.exe taller de lenguajes 1
argv[0] => C:\Users\erucci\Desktop\prueba\bin\Release\prueba.exe
argv[1] => taller
argv[2] => de
argv[3] => lenguajes
argv[4] => 1

Process returned 0 (0x0)   execution time : 0.004 s
Press any key to continue.
```

- Es importante notar el empleo del formato `%s` para la impresión de cada argumento, ya que son cadenas de caracteres.

Compilación con *GCC*

Compilación con *GCC*

- GCC es un compilador integrado del proyecto GNU para C, C++, Objective C y Fortran;
- Es capaz de recibir un programa fuente en cualquiera de estos lenguajes y generar un programa ejecutable binario en el lenguaje de la máquina donde ha de correr.
- La sigla GCC significa "GNU Compiler Collection", aunque originalmente hacía referencia a "GNU C Compiler".

Compilación con *GCC*

- Sintaxis:

```
gcc [opción | archivo ] ...
```

- Ejemplo:

```
gcc hola.c
```

- Compila el programa C `hola.c` y genera el ejecutable `a.out`
- `a.out` es el nombre por defecto.

Compilación con GCC

- Opciones de compilación:
 - Van precedidas de un guión, como es habitual en sistemas UNIX/Linux.
 - Pueden tener varias letras.
 - No se pueden agrupar varias opciones tras un mismo guión.
 - Algunas opciones requieren además de un nombre de archivo o directorio.
 - Finalmente, se puede incluir más de un archivo fuente en el proceso de compilación.

Compilación con GCC

- Algunas opciones de compilación útiles:
- `-o archivo`
 - Permite cambiar el nombre del archivo generado.
 - Ejemplo: `gcc hola.c -o hola`
 - Compila el programa `hola.c` y genera el ejecutable `hola`.
- `-Wall`
 - Además de los errores, muestra todas las advertencias del compilador (*warnings*).
 - Ejemplo: `gcc hola.c -o hola -Wall`

Compilación con GCC

- Algunas opciones de compilación útiles:
- `-E`
 - Realiza solamente el preprocesamiento, enviando el resultado a la salida estándar.
 - Útil para visualizar el trabajo del preprocesador.
- `-Ox`
 - Indica el nivel de optimización aplicado por el compilador.
 - Ejemplo: : `gcc hola.c -o hola -O1`
 - Compila el programa `hola.c` y genera el ejecutable `hola` aplicando las optimizaciones del nivel 1.
 - Otras opciones: `O2`, `O3`, `Os`, `O0`, `O`.

Compilación con *GCC*

- Compilación con varios archivos

- Ejemplo:

```
gcc archivo1.c archivo2.c -o ejecutable
```

- Sólo uno de los archivos `.c` debe contener la función `main`.