Paso de Borland Turbo C (bajo DOS) a Anjuta (Linux)[1].

Anjuta es un entorno de desarrollo de C que podemos encontrar en cualquier distribución de GNU/Linux. Si nuestra distribución no dispone de ella, es fácil encontrarla en Internet, e instalarla en nuestro sistema GNU/Linux.

Anjuta utiliza el compilador de GNU gcc.

Todas aquellas personas que están acostumbradas a hacer uso del Borland Turbo C, pueden tener algunos problemas a la hora de compilar el código de sus programas.

A continuación iré describiendo los problemas (y soluciones) que he ido encontrando.

SOBRE LAS FUNCIONES:

1.- Función clrscr();
2.- Función pow();
3.- Función scanf(). No funciona.
4.- Funciones getch(), getche(), getchar()... Las dos primeras no son ANSI C.
5.- Función getchar(). Aparentemente no hace nada.
6.- Generación de números aleatorios (randomize/random) vs (srand/rand).
7.- main(int argc, char *argv[]). Ventajas de gcc respecto a TurboC.
8.- funciones atoi, itoa, sprintf.

SOBRE EL MANEJO DEL ENTORNO DE PROGRAMACIÓN.

101.- Cómo lograr que funcione el depurador del Anjuta.

SOBRE LAS BIBLIOTECAS (LIBRERÍAS).

201.- Librería CONIO.H . No funciona. No es ANSI C.
202.- Cómo hacer que el compilador gcc incluya otras librerías.

SOBRE LAS ESTRUCTURAS (Ventajas de GCC).

301.- Array (estático) de estructuras. Intercambio de dos elementos del array.

1 Última revisión (4) 16 de marzo de 2006.

Al pasar de TurboC a Anjuta, me han surgido algunos problemas, que he descrito en este documento.

1.- función clrscr();

Función: borra la pantalla.

Problema: no funciona.

Motivo: esta función se encuentra en la librería conio.h. Esta librería no es ANSI C. Por tanto no la podemos utilizar con ANJUTA.

Solución:

- El ANSI C no dispone de función para borrar la pantalla.
- Podemos utilizar la función "system" que permite ejecutar comandos del sistema
- Ejemplo: system("clear");

2.- función pow();

Función: función potencia.

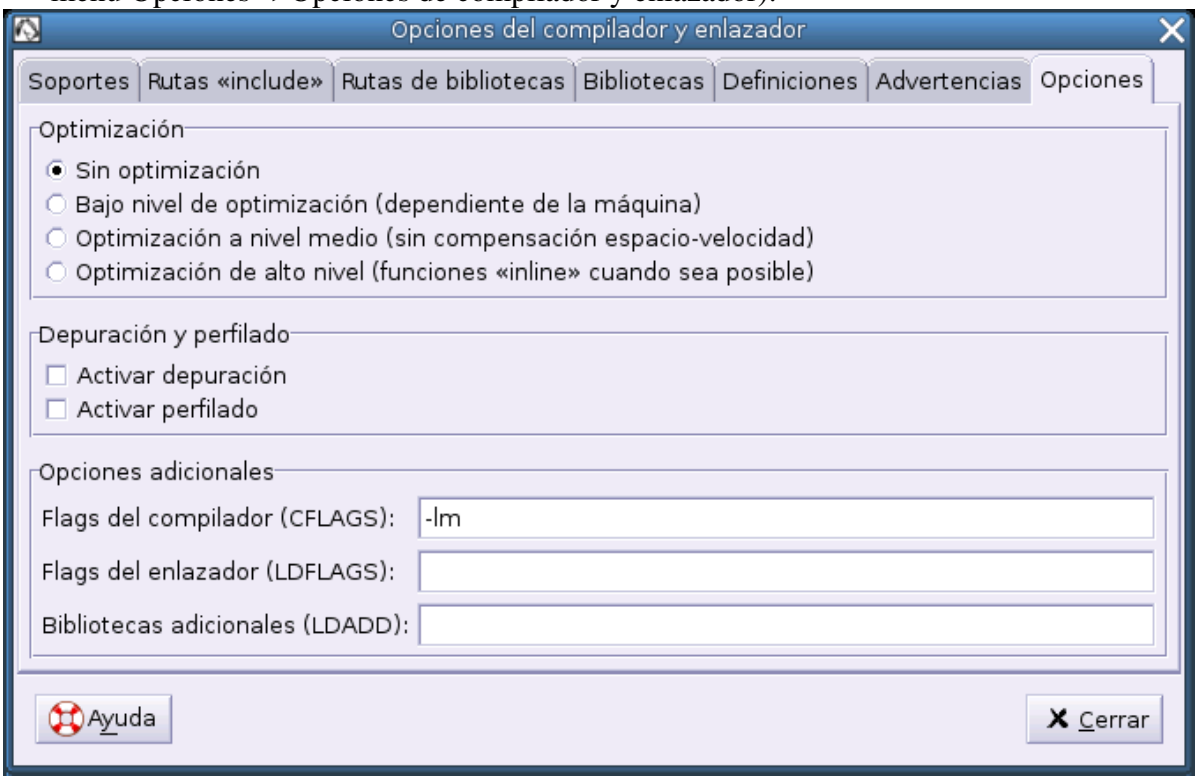
Problema:

- Anjuta no encuentra math.h

Motivo:

- Esta función se encuentra en la librería math.h (que sí es ANSI).
- Por tanto debemos añadir #include <math.h> en la sección de directivas.
- Pero aún así, Anjuta no encuentra dicha librería.

Solución: Añade el parámetro -lm tal y como se muestra en la siguiente imagen (Acceder al menú Opciones-->Opciones de compilador y enlazador):



- Hemos de recurrir a librerías. Por ejemplo: utilizar la librería ncurses.
  - Para utilizar la librería:
    - acudir a la opción del menú (Anjuta) Opciones-->Opciones del compilador y enlazador-->Bibliotecas.
    - Elegir la librería ncurses, y pulsar el botón añadir.

3.- función scanf();

Función:

- **captura datos de la entrada estándar.**

Problema:

- Cuando queremos leer un carácter y almacenarlo en una variable de tipo carácter, la siguiente vez que utilizamos scanf, la función no lee.

Motivo:

- quedan restos en el buffer del teclado.
- Esto se debe a que, en el primer scanf, al darle al <enter> estamos introduciendo ese carácter '\n' que queda almacenado en el buffer del teclado.

Solución:

- Con el Borland turboC 3.0 lo que hacíamos era vaciar el buffer del teclado. Podemos hacer uso de fflush(stdin);
- Pero ahora, con Anjuta no funciona. Tenemos una solución muy simple: decirle al scanf que no nos coja los caracteres en blanco que haya antes del carácter válido. ¿Qué cómo se hace esto?. Simplemente dejando un espacio en blanco antes del %c. Así:
    - printf("Introduce un carácter: "); scanf(" %c",&car);

4.- función getch(), getche() y getchar(). Las dos primeras no son ANSI C.

Función: Estas funciones se parecen mucho entre sí. Permiten capturar un carácter.

Problema: Las dos primeras no pueden ser utilizadas en Linux.

Motivo: Pertenecen a la librería conio.h de Borland.

Solución:

- No hacer uso de getche() ni de getch().
- getchar() sí es ANSI-C por lo tanto, sí es reconocida por gcc.
- Hemos de tener en cuenta que getchar() equivale a un scanf("%c",). Ejemplo:
    - caracter=getchar(); equivale a
    - scanf("%c",&caracter);
    - donde "caracter" es una variable de tipo carácter.

5.- Función getchar(). Aparentemente no hace nada.

Problema: Aparentemente no funciona. No captura el carácter deseado.

Motivo: Se trata del mismo problema que hemos descrito con "scanf". Leed el punto "3.- Función scanf(). No funciona" para más información.

6.- Generación de números aleatorios con Anjuta (gcc).

Descripción: La generación de números aleatorios en TurboC se suele hacer con las funciones randomize (para generar la semilla aleatoria) , y random(n) para generar el número aletorio entre 0 y n-1.

Problema: No funcionan en linux.

Motivo: Estas funciones no son ANSI C. El gcc no las reconoce.

Solución: Utilizaremos funciones que sí funcionan en Linux, por cumplir con ANSI C. Estas funciones son srand() y rand(). Estas funciones se encuentran en stdlib.h El uso de estas funciones difiere ligeramente a randomize() y random(). Veámoslas:

- **srand():** genera una semilla aleatoria. Necesita que le pasemos un argumento. Podemos hacer que la genere basándose en la señal del reloj. Así podemos hacer:
  - srand(time(NULL));
- **rand();** genera un número aleatorio. No le podemos indicar el rango de números que queremos que genere. Por tanto, combinándolo con la ayuda del operador módulo (%), obtenemos el mismo efecto:
  - Ejemplo rand()%10 --> generará números enteros. Al aplicarle el %10, éstos serán entre 0 y 9.

Ejemplo: generar un número aleatorio entre 11 y 20.

Con Borland TurboC	#include <stdio.h>
	#include <stdlib.h>
	void main(void)
	{
	int alea;
	randomize();
	alea=random(10)+11; //random(10) genera de 0 a 9.
	....
	}

Con Anjuta (gcc)	#include <stdio.h>
	#include <stdlib.h>
	int main(void)
	{
	int alea;
	srand(time(NULL));
	alea=random()%10+11; //random()%10 genera de 0 a 9.
	....
	return(0);
	}

7.- main(int argc, char *argv[]). Ventajas de gcc respecto a TurboC.

Ejemplo: En Turbo C sería impensable hacer algo como:

```
int main(int argc, char *argv[])
{
 int vector[argc-1];
 ....
 return(0);
}
```

Es decir, haciendo uso de Borland Turbo C, no podemos utilizar "argc" (número de argumentos que se le pasan a la función principal) como tamaño de un vector.
Sin embargo, gcc lo acepta sin problemas.

8.- funciones atoi, itoa, sprintf.

atoi	• **int atoi(char *)** es una función ANSI C. Se puede utilizar sin problemas.
	  • ¿Qué permite esta función?: esta función permite transformar una cadena de caracteres en un número entero.
	  • Ejemplo:
	  int n;
	  char cad[]="143";
	
	  n=atoi(n);
	  //La variable "n" tomará el valor entero 143.

itoa	• itoa(char *, int,int) es una función que no pertenece al ANSI C. Por tanto, no puede ser utilizada. En su lugar, podemos utilizar sprintf.
	• Esta función transforma un número entero, en una cadena de caracteres.
	• En TurboC actuábamos de la siguiente manera:
	  int n=153;
	  char cad[5];
	
	  itoa(cad,n,10);
	  //La variable "cad" tomará la cadena "153".
	• Si hacemos uso de ella con gcc, obtendremos un error. Para solucionar este problema, podemos hacer uso de la función **sprintf**, que comentamos a continuación.

sprintf	• sprintf funciona de forma muy parecida a la función printf.
	• Simplemente, en lugar de imprimir en pantalla, imprime en una cadena de caracteres.
	• Por tanto, si tenemos un número y queremos pasarlo a cadena de caracteres, podemos hacer uso de esta función. Veamos el siguiente ejemplo:
	  int n=153;
	  char cad[5];
	
	  sprintf(cad,"%i",n);
	  //La variable "cad" tomará la cadena "153".

101.- Depurador;

Función:

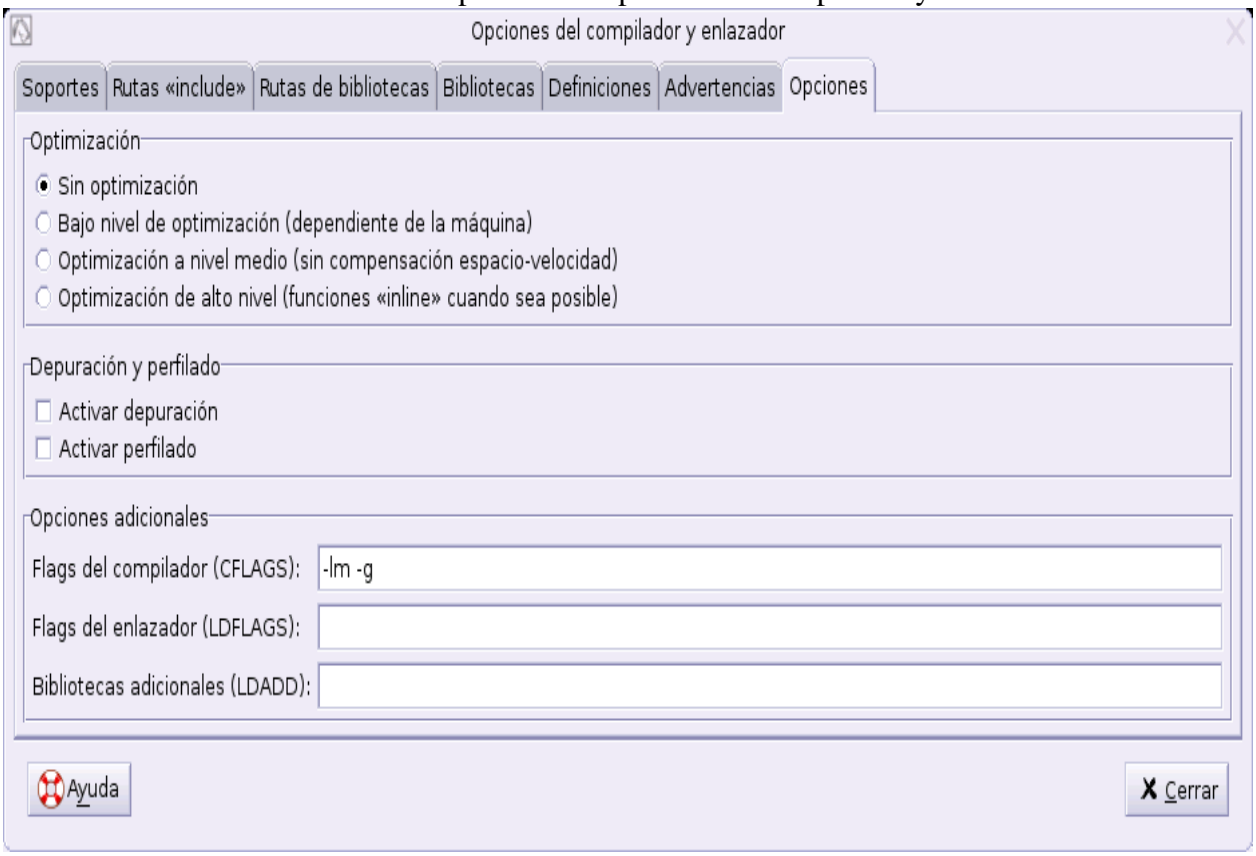
Problema: no funciona el depurador.

Motivo:

- Es necesario que el programa ejecutable contenga la tabla de símbolos.

Solución:

- Al compilar el programa, hemos de añadir la opción -g al compilador, tanto si lo hacemos con gcc directamente, o lo hacemos con Anjuta. Si lo hacemos con Anjuta, la configuración se hace de la siguiente forma:
    - Acceder al menú Opciones-->Opciones de compilador y enlazador:



201.- Librería CONIO.H . No funciona. No es ANSI C.

Problema: No es posible utilizar las funciones pertenecientes a CONIO.H

Funciones implicadas: todas las de conio.h. Por ejemplo:

- clrscr() (comentada la solución en el punto "1.- Función clrscr()" de este documento.
- getch(), getche() (comentada la solución en el punto "4.- Función getch()" de este documento.

Motivo: "La librería" conio.h pertenece al compilador de Borland. No es ANSI C.

Solución: Debemos evitar hacer uso de funciones que no son ANSI C.

202.- Cómo hacer que el compilador gcc incluya otras librerías.

Problema: necesito hacer uso de una librería. Pero el compilador no hace caso. Por ejemplo, la "librería" #include <math.h>

Motivo: es necesario indicar explícitamente al compilador, que queremos enlazar nuestro programa con las otras librerías.

Solución: la solución ha sido desarrollada en el punto "2.- Función pow" de este manual.

301.- Array (estático) de estructuras. Intercambio de dos elementos del array. Ejemplo:

Tenemos la siguiente estructura:

```
struct s_ejemplo
{
        int campo1;
        char campo2[10];
        float campo3;
};
```

Tenemos la siguiente función:

```
int funcionejemplo()
{
        struct s_ejemplo vector[100];
        struct s_ejemplo temp;

        //Imaginemos que deseo cambiar el elemento 10 de mi vector, por el elemento 20.
        //En Anjuta  podemos proceder así:

        temp=vector[10];
        vector[10]=vector[20];
        vector[20]=temp;
}
```

En TurboC, necesitamos más código (campo por campo):

```
temp.campo1=vector[10].campo1;
strcpy(temp.campo2,vector[10].campo2);
temp.campo3=vector[10].campo3;

vector[10].campo1=vector[20].campo1;
strcpy(vector[10].campo2,vector[20].campo2);
vector[10].campo3=vector[20].campo3;

vector[20].campo1=temp.campo1;
strcpy(vector[20].campo2,temp.campo2);
vector[20].campo3=temp.campo3;
```