

INFORMÁTICA

Práctica 3. Depuración de programas.

Depuración. Depurador DDD.

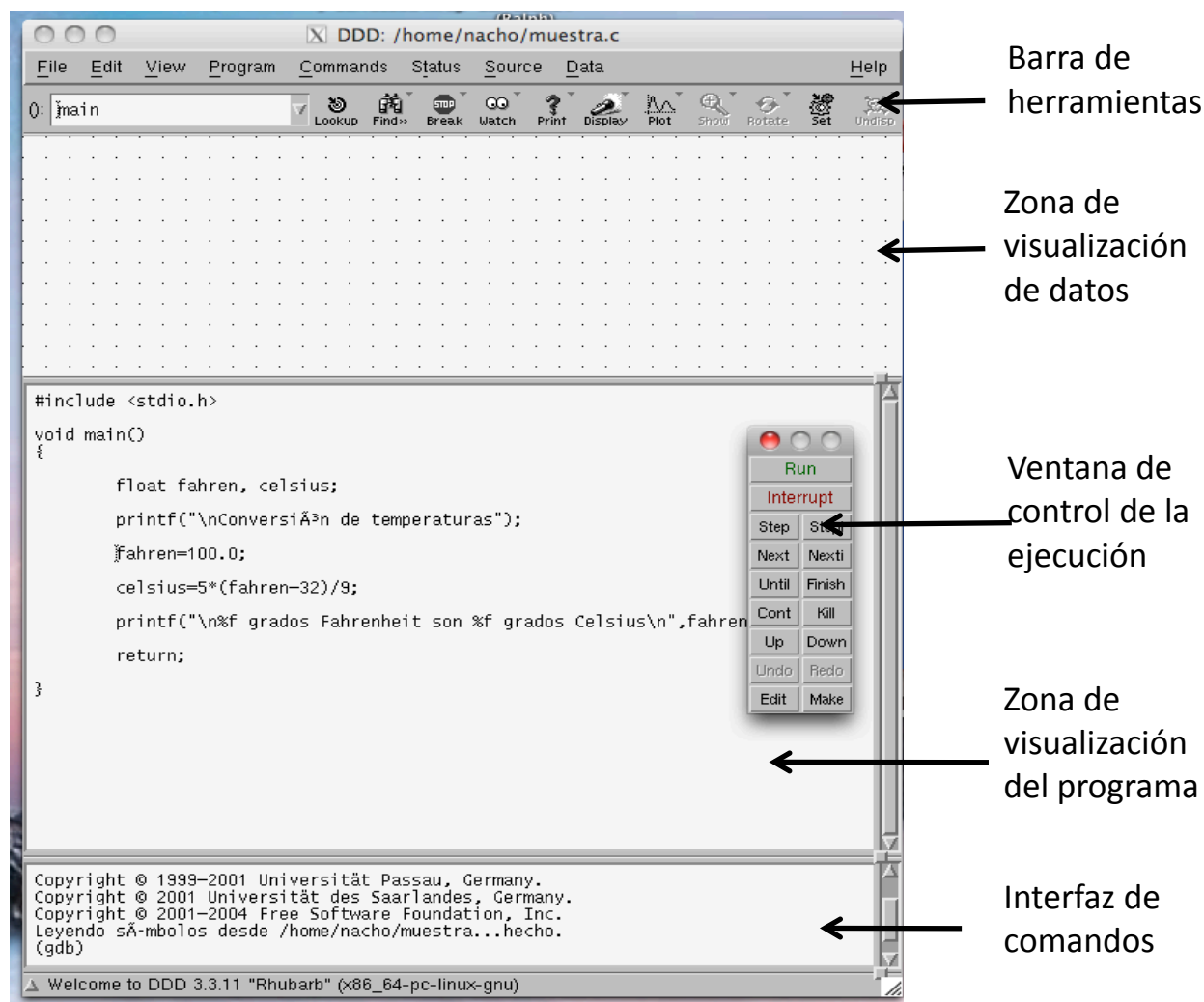
Es bastante frecuente que los programas contengan errores de diseño o de programación, por lo que una herramienta imprescindible es el depurador. El depurador permite ejecutar un programa paso a paso (instrucción a instrucción) y analizar el valor de las variables en cada momento, para ver en qué momento la ejecución se separa de lo previsto. Es pertinente recordar aquí la máxima:

“Un programa hace lo que dices, no lo que quieres”.

El depurador que vamos a utilizar se llama Data Display Debugger (o ddd). Se puede arrancar desde el escritorio, o desde un interfaz de comandos, tecleando:

```
bash-ln.05$ ddd muestra.c
```

La ventana del depurador será parecida a lo que se ve en la siguiente figura:



Una vez cargado el programa, lo primero es ponerlo en marcha. Para ello, se pone un punto de parada (*breakpoint*) en una línea del mismo (por ejemplo, en el primer `printf`) haciendo click en la línea en la que se encuentra, al principio. Eso llevará el cursor a esa posición. A continuación se pulsa el botón *Break* de la barra de herramientas. Para desactivarlo, basta con volver a pulsar el

botón 'Break'. Una vez puesto el punto de parada, se echa a correr el programa pulsando el botón 'Run' de la ventana de control de la ejecución. El programa se parará en la línea para la que hemos creado un punto de parada.

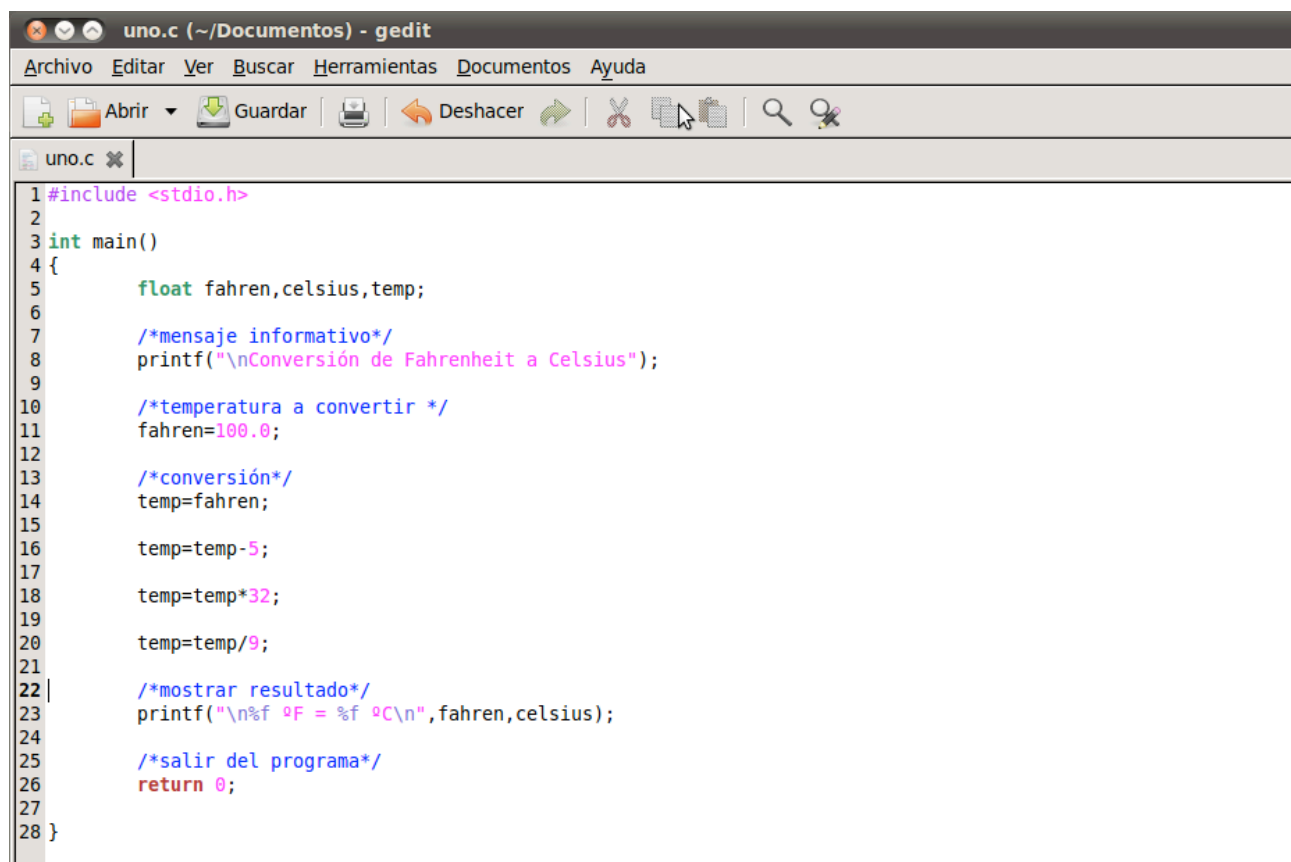
Es posible ver el valor de variables seleccionándolas con el ratón (haciendo click sobre ellas en la zona de visualización de programa) y, a continuación, pulsando el botón 'Display'. Eso hará que aparezcan en la zona de visualización de datos, y se vayan mostrando los valores que van tomando.

Para ejecutar una línea del programa, se pulsa bien el botón 'Step', bien el botón 'Next', ambos en la ventana de control de la ejecución. La diferencia entre ambos radica en el tratamiento de las funciones. Cuando se llega a una función, 'Next' ejecuta toda las instrucciones de la función de una vez, es decir, con una única pulsación del botón, mientras que cada pulsación de 'Step' ejecuta una única instrucción de la función.

Además, es posible saltarse zonas completas de código simplemente poniendo un punto de parada en la instrucción donde termina la zona que nos queremos saltar, y echando a correr el programa con el botón 'Cont'. La ventana de control de la ejecución tiene otras opciones que es interesante explorar. Para ello, consultar la bibliografía de esta práctica, situada al final de este cuaderno.

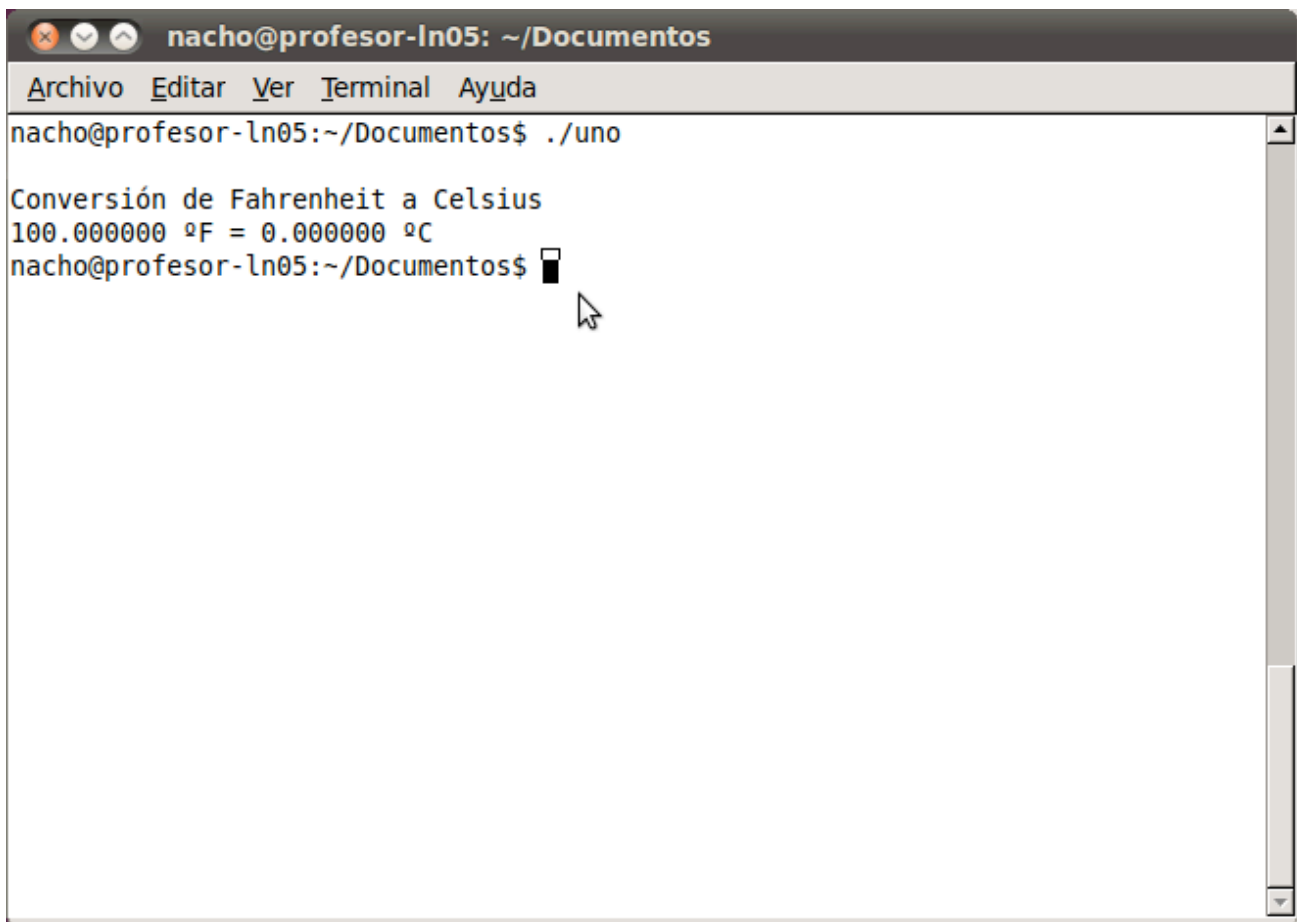
Utilización de ddd para depurar un programa.

En esta sección se muestra cómo utilizar ddd para encontrar los errores de un programa. El punto de partida es el siguiente programa:



```
1 #include <stdio.h>
2
3 int main()
4 {
5     float fahren,celsius,temp;
6
7     /*mensaje informativo*/
8     printf("\\nConversión de Fahrenheit a Celsius");
9
10    /*temperatura a convertir */
11    fahren=100.0;
12
13    /*conversión*/
14    temp=fahren;
15
16    temp=temp-5;
17
18    temp=temp*32;
19
20    temp=temp/9;
21
22    /*mostrar resultado*/
23    printf("\\n%f °F = %f °C\\n", fahrenheit, celsius);
24
25    /*salir del programa*/
26    return 0;
27
28 }
```

Se pretende que realice la conversión de una temperatura expresada en grados Fahrenheit, contenida en la variable `fahren`, a grados Celsius, y la muestre por pantalla. Si se compila y ejecuta, el resultado es el siguiente:



```
nacho@profesor-ln05: ~/Documentos
Archivo  Editar  Ver  Terminal  Ayuda
nacho@profesor-ln05:~/Documentos$ ./uno
Conversión de Fahrenheit a Celsius
100.000000 °F = 0.000000 °C
nacho@profesor-ln05:~/Documentos$
```

Como se puede comprobar, no es el correcto, ya que 100°F no son 0°C, por lo que el programa no funciona bien. Para encontrar los errores, lo primero es lanzar el depurador. En la siguiente imagen, se han puesto dos puntos de parada, uno en la primera sentencia del programa, y otro justo en la llamada a la función `printf` que muestra los resultados. Además, se visualiza el valor de las variables `fahren` y `celsius`. Las instrucciones para hacer todo esto se encuentran más arriba en este documento.

El programa se ha dejado correr hasta el segundo punto de parada, y se puede ver que la variable `celsius`, que es la que se muestra, contiene el valor 0, por lo que la función `printf` está funcionando correctamente. El problema ya salta a la vista: los cálculos de la conversión se realizan con la variable `temp`, y el valor final no se traslada a la variable `celsius`, que, por lo tanto, mantiene su valor inicial.

DDD: /home/nacho/Documentos/uno.c

File Edit View Program Commands Status Source Data Help

(): fahrenheit

1: celsius	2: fahrenheit
0	100

```
#include <stdio.h>

int main()
{
    float fahrenheit,celsius,temp;

    /*mensaje informativo*/
    printf("\nConversion de Fahrenheit a Celsius");

    /*temperatura a convertir */
    fahrenheit=100.0;

    /*conversion*/
    temp=fahrenheit;

    temp=temp-5;

    temp=temp*32;

    temp=temp/9;

    /*mostrar resultado*/
    printf("\n%f °F = %f °C\n",fahrenheit,celsius);

    /*salir del programa*/
    return 0;
}
```

(gdb) cont

Breakpoint 2, main () at uno.c:23
(gdb)

▲ Breakpoint 2, main () at uno.c:23

Para saber si la variable temp contiene el valor correcto, y al menos la conversión se ha realizado correctamente, se visualiza su valor en el depurador:

DDD: /home/nacho/Documentos/uno.c

File Edit View Program Commands Status Source Data Help

(): uno.c:6

1: celsius 0
2: fahrenheit 100
3: temp 337.777771

```
#include <stdio.h>

int main()
{
    float fahrenheit,celsius,temp;

    /*mensaje informativo*/
    printf("\nConversion de Fahrenheit a Celsius");

    /*temperatura a convertir */
    fahrenheit=100.0;

    /*conversion */
    temp=fahrenheit;

    temp=temp-5;

    temp=temp*32;

    temp=temp/9;

    /*mostrar resultado*/
    printf("\n%f °F = %f °C\n", fahrenheit,celsius);

    /*salir del programa*/
    return 0;
}
```

Breakpoint 2, main () at uno.c:23
(gdb) graph display temp
(gdb)

Display 3: temp (enabled, scope main, address 0x7fffffff254)

El valor que adquiere, 337.777771, no es el correcto, luego en la conversión de la temperatura hay otro error. Se puede depurar el proceso de conversión reiniciando la ejecución del programa (botón 'Run'), y poniendo puntos de parada en cada sentencia de la conversión. Esto no es estrictamente necesario, ya que se puede ejecutar sentencia a sentencia con el botón 'Step'. Ambos métodos son igual de válidos.

DDD: /home/nacho/Documentos/uno.c

File Edit View Program Commands Status Source Data Help

(): uno.c:14

1: celsius	2: fahrenheit
0	100

3: temp
100

```
#include <stdio.h>
int main()
{
    float fahrenheit,celsius,temp;

    /*mensaje informativo*/
    printf("\nConversion de Fahrenheit a Celsius");

    /*temperatura a convertir */
    fahrenheit=100.0;

    /*conversion */
    temp=fahrenheit;

    temp=temp-5;

    temp=temp*32;

    temp=temp/9;

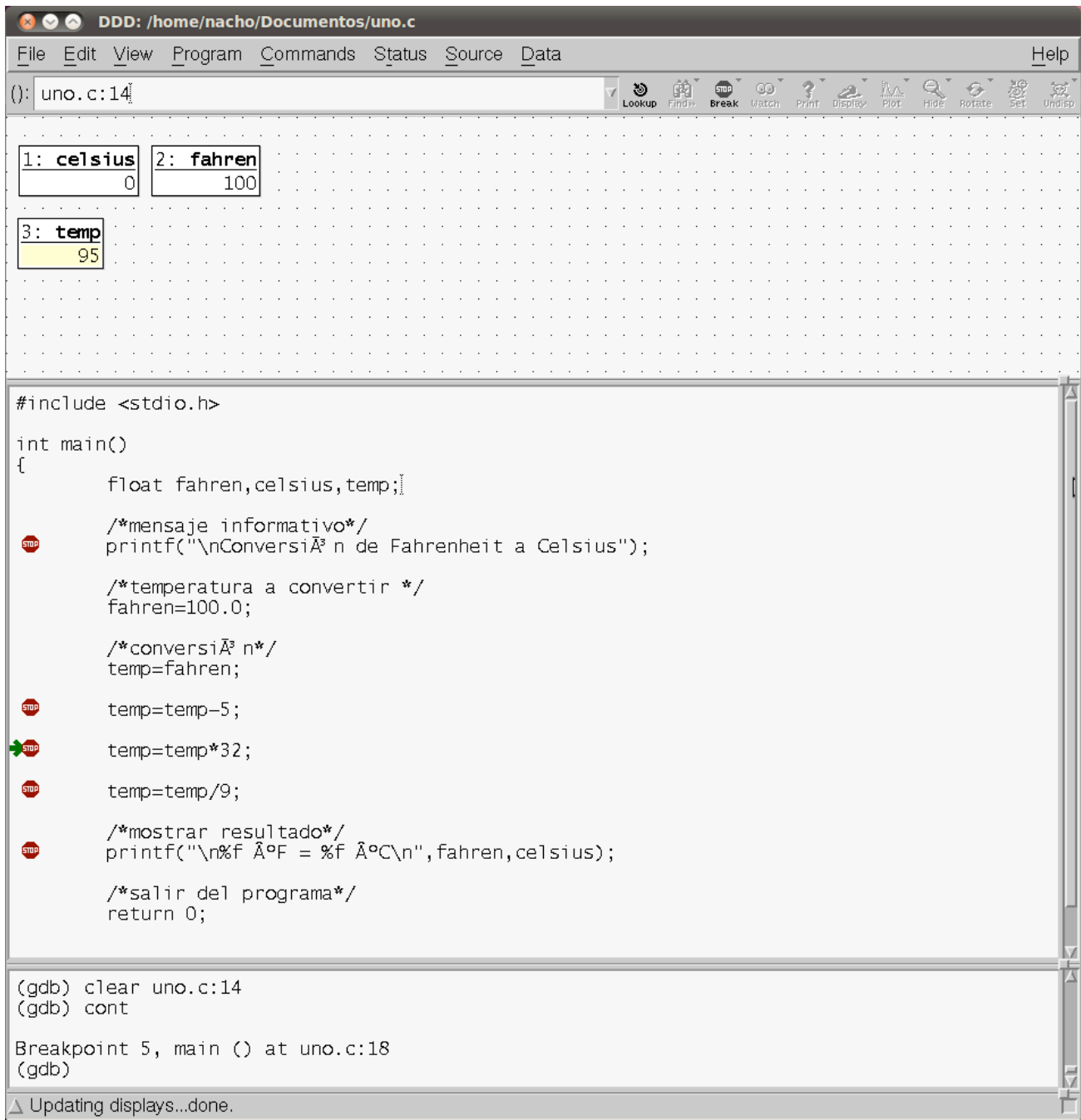
    /*mostrar resultado*/
    printf("\n%f °F = %f °C\n", fahrenheit,celsius);

    /*salir del programa*/
    return 0;
}
```

(gdb) cont

Breakpoint 4, main () at uno.c:16
(gdb) clear uno.c:14
(gdb)

Se puede ir ejecutando el programa para comprobar el proceso de conversión de Fahrenheit a Celsius. Si se usan puntos de parada, para pasar de cada uno al siguiente, se debe pulsar el botón 'Cont'. La siguiente captura muestra uno de los pasos intermedios de este proceso.



En algún momento de la ejecución, el programador atento se dará cuenta de que la conversión de Fahrenheit a Celsius no es la correcta. La conversión correcta se muestra en la primera figura de esta práctica, en concreto:

$$\text{celsius} = 5 * (\text{fahrenheit} - 32) / 9;$$

La siguiente figura muestra el código con la conversión correcta y la asignación de la variable `temp` a la variable `celsius`, para que se muestre el valor correcto por pantalla al llamar a la función `printf`.


```
uno.c (~/Documentos) - gedit
Archivo  Editar  Ver  Buscar  Herramientas  Documentos  Ayuda
Abrir  Guardar  Deshacer
uno.c x
1 #include <stdio.h>
2
3 int main()
4 {
5     float fahrenheit,celsius,temp;
6
7     /*mensaje informativo*/
8     printf("\nConversión de Fahrenheit a Celsius");
9
10    /*temperatura a convertir */
11    fahrenheit=100.0;
12
13    /*conversión*/
14    temp=fahrenheit;
15
16    temp=temp-32;
17
18    temp=temp*5;
19
20    temp=temp/9;
21
22    celsius=temp;|
23
24    /*mostrar resultado*/
25    printf("\n%f °F = %f °C\n",fahrenheit,celsius);
26
27    /*salir del programa*/
28    return 0;
29
30 }
```

C Ancho de la tabulación: 8 Ln 22, Col 22 INS

Si se compila y ejecuta este programa, el resultado de la ejecución es el correcto. La siguiente figura muestra el resultado de la depuración. Se puede ver, en el panel del interfaz de comandos (el de abajo del todo), el volcado de pantalla de la función `printf`, con el valor correcto. Esto es lo que se vería en la ventana del interfaz de comandos si se ejecutara el programa directamente desde el *prompt*. Además, las variables `fahrenheit` y `celsius` tienen los valores adecuados.

DDD: /home/nacho/Documentos/uno.c

File Edit View Program Commands Status Source Data Help

(): uno.c:28

1: **fahrenheit**
100

2: **celsius**
37.777786

```

float fahrenheit,celsius,temp;

/*mensaje informativo*/
printf("\nConversion de Fahrenheit a Celsius");

/*temperatura a convertir */
fahrenheit=100.0;

/*conversion */
temp=fahrenheit;

temp=temp-32;

temp=temp*5;

temp=temp/9;

celsius=temp;

/*mostrar resultado*/
printf("\n%f F = %f C\n",fahrenheit,celsius);

/*salir del programa*/
return 0;
}

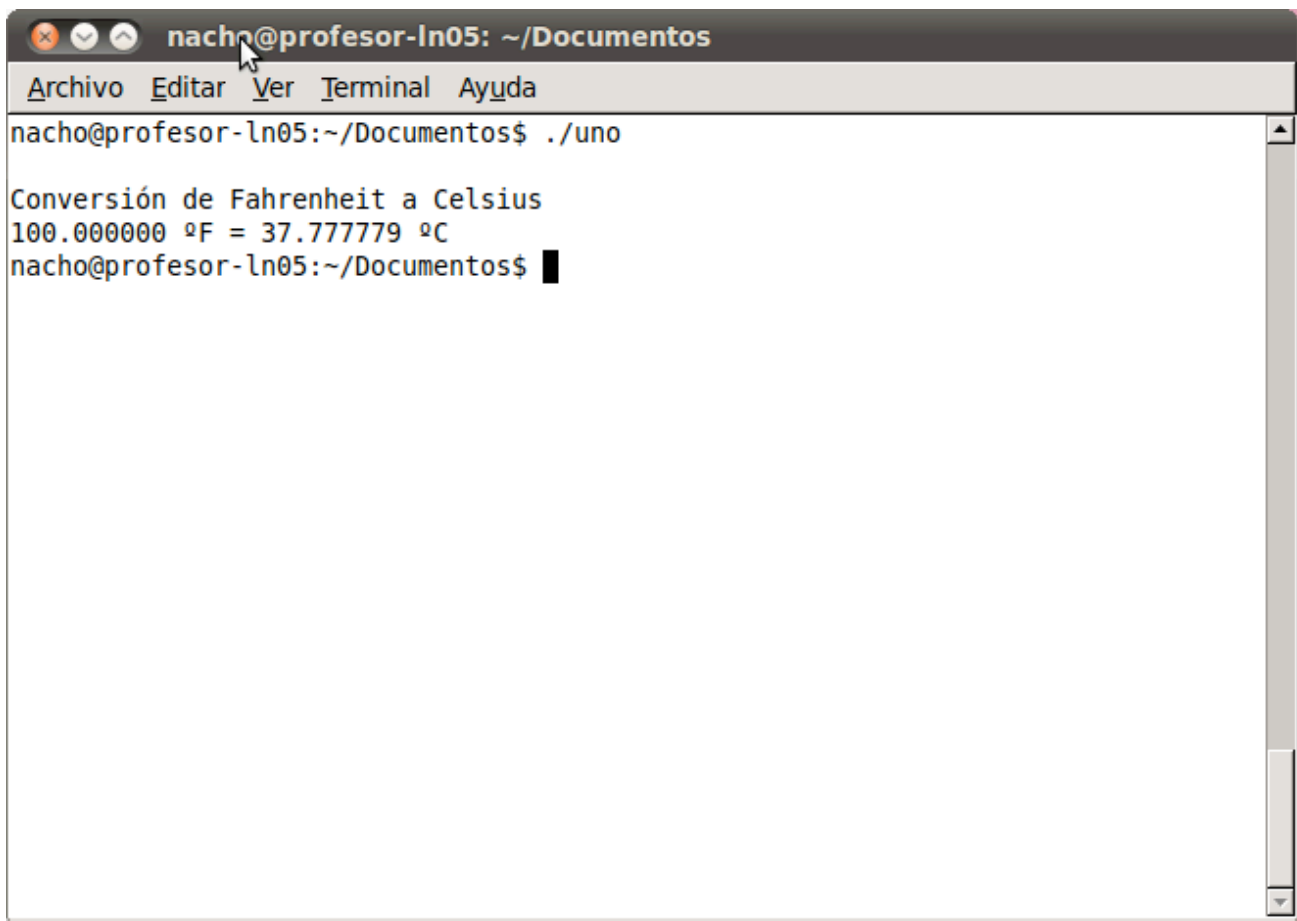
```

Conversion de Fahrenheit a Celsius
100.000000 F = 37.777779 C

Breakpoint 2, main () at uno.c:28
(gdb)

Execution window has been closed...done.

A continuación, precisamente la ventana del interfaz de comandos con la invocación al programa, y el resultado correcto.



```
nacho@profesor-ln05: ~/Documentos
Archivo Editar Ver Terminal Ayuda
nacho@profesor-ln05:~/Documentos$ ./uno
Conversión de Fahrenheit a Celsius
100.000000 °F = 37.777779 °C
nacho@profesor-ln05:~/Documentos$ █
```

Depuración. Depurador gdb.

El depurador DDD, visto en el apartado anterior, no es más que una interfaz gráfica entre el usuario y el depurador real, el `gdb`. Éste último es el que hace realmente la depuración, y DDD se limita a transferir información y comandos entre él y el usuario. En algunas ocasiones, es interesante manejar el depurador `gdb` directamente, por lo que en esta sección se va a explicar, de forma un poco superficial, su utilización.

Dado que DDD recurre a `gdb` para hacer el trabajo de depuración, evidentemente todas las cosas que se han visto en el apartado anterior son posibles aquí: ejecutar instrucción a instrucción, ver el valor de variables, gestionar la entrada/salida del programa en depuración, fijar puntos de parada, etc. La única diferencia (y lo que lo hace, tal vez, un poco más gravoso) es que `gdb` se maneja desde el terminal, no tiene interfaz gráfica. Es decir, las acciones de depuración se desencadenan por medio de comandos que se deben teclear en el terminal.

Lo primero, pues, es arrancar el depurador. En un terminal se teclea:

```
bash-ln.05$ gdb
```

y esto hace que el depurador arranque. El *'prompt'* cambia, para indicar que ya no estamos interactuando con el interfaz de comandos sino con el depurador. Y a partir de aquí se repite un patrón básico: introducir comando, recibir resultado. El primero es indicar cuál es el programa que se desea depurar. En nuestro caso, `muestra`. (Dado que para depurar el programa es necesario referirse a las instrucciones del mismo, es aconsejable abrirlo en un editor de texto y activar la opción para mostrar el número de línea). El comando para cargar nuestro programa es `file`.

```
(gdb) file muestra
```

Esto prepara el depurador para procesar el programa `muestra`. El proceso de depuración en `gdb` es idéntico al que se hace con `DDD`. Lo único necesario es conocer los comandos de `gdb` que corresponden a los botones de `DDD`. Por ejemplo, para poner un punto de parada en la sentencia `return 0` de la línea 28 (tomamos como referencia la versión corregida del programa `muestra.c` que aparece más arriba), el comando es:

```
(gdb) b 28
```

Para ejecutar el programa tenemos 4 comandos: el comando `run`, que lo ejecuta de forma continua desde el principio, el comando `step`, que ejecuta una instrucción, el comando `next`, que ejecuta una línea de código (y si es una llamada a función, la ejecuta entera), y el comando `cont`, que reanuda la ejecución de forma continua desde donde se encuentre parado el programa. Como se ve, son las mismas opciones que ofrece `DDD`. La ejecución continua, al igual que ocurre con `DDD`, se interrumpe con el primer punto de parada que se encuentre o con el final del programa.

Al interrumpirse la ejecución, el depurador muestra el número de línea y la instrucción que ejecutaría a continuación. Por ejemplo, con el punto de parada en la línea 28 mostraría:

```
Breakpoint 1, main() at muestra.c:28
28      return 0;
(gdb)
```

La primera línea indica que es el primer punto de parada que tenemos, situado en la función `main()` en la línea 28 del archivo `muestra.c`. A continuación, la línea en cuestión, y por último, el *prompt* espera el siguiente comando.

También es posible visualizar variables. En este caso tenemos dos opciones: ver puntualmente el valor que toma, o seguir la evolución de la variable. Para lo primero tenemos el comando `print`, y para lo segundo el comando `display`. El primero muestra el valor actual de una variable una única vez, y el segundo hace que se muestre cada vez que se interrumpe la ejecución del programa. Por ejemplo, si antes de ejecutar el programa usamos el comando `display` para seguir la evolución de las variables `temp` y `fahren`, cuando se para el programa en la línea 28 `gdb` mostraría:

```
Breakpoint 1, main() at muestra.c:28
28      return 0;
2: temp = 37.77777778
1: fahren = 100.0
(gdb)
```

Es posible visualizar también cadenas de caracteres, números enteros, caracteres sueltos, etc.

Por último, los comandos `kill` y `quit` terminan la ejecución del programa en depuración. `kill` se queda dentro del depurador (para volver a depurarlo, por ejemplo) y `quit` sale.

El depurador `gdb` es un programa muy completo con muchas opciones. Es muy recomendable revisar su página del manual. Además, en la página de la asignatura está disponible una hoja de referencia rápida con un resumen de los comandos y una breve descripción de los mismos.

EJERCICIOS.

Encontrar los errores en los siguientes programas utilizando los depuradores DDD o `gdb`:

Programa 1: *El programa debe comparar dos variables y mostrar por pantalla la mayor de ellas. En caso de que sean iguales, además, debe indicarlo.*

Código:

```
#include <stdio.h>

int main()
{
    int var1, var2;

    var1=10;
    var2=5;

    if (var1>=var2)
        printf("\nLa variable mayor tiene el valor: %d",var2);
    else
        printf("\nLa variable mayor tiene el valor: %d",var1);

    if(var1=var2)
        printf("\nAmbas variables valen lo mismo: %d\n",var1);

    return 0;
}
```

Programa 2: *El programa debe calcular la media aritmética de los valores de un array de números enteros.*

Código:

```
#include <stdio.h>

int main()
{
    int lista[10]= {3,5,2,1,6,2,3,1,5,9};
    int suma,i,j;
    float media;

    for(i=1;i<=10;i++)
        suma=suma+lista[j];

    media=suma/2;

    printf("\nLa media aritmética de la lista es: %f\n",media);

    return 0;
}
```

Programa 3: *El programa debe crear un array de números enteros a partir de otro dado, de forma que, en cada posición del nuevo array se almacene la suma de todos los elementos del array antiguo hasta esa posición inclusive.*

Ejemplo:

si el array original es

{2, 1, 4, 7, 2}

el nuevo array sería:

{2, 3(2+1), 7(2+1+4), 14(2+1+4+7), 16(2+1+4+7+2)}

Código:

```
#include <stdio.h>

int main()
{
    int original[10]= {3,5,2,1,6,2,3,1,5,9};
    int nuevo[10];
    int suma,i,j;

    printf("\nOriginal\tNuevo\n");

    for(i=1;i<=10;i++)
        printf("%d\t",original[i]);
        suma=suma+original[i];
        printf("%d\n",suma);
        nuevo[i]=suma;

    return 0;
}
```

Bibliografía.

- Tutorial de ddd en castellano: <http://www.linuxfocus.org/Castellano/January1998/article20.html>
- Manual (en inglés): <http://www.gnu.org/software/ddd/manual/pdf/ddd.pdf>