Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
Computación II (CI2126)

LABORATORIO 2
(apuntadores a funciones)

MÉTODO DE TRABAJO: el enunciado del laboratorio será publicado con antelación a la sesión de clases, de modo que el estudiante pueda preparar sus respuestas a las preguntas planteadas en el mismo. En la clase de laboratorio, el docente dará una (1) hora para que los alumnos instrumenten sus soluciones y luego de ese lapso, procederá a discutir y exponer, para todo el grupo, las soluciones alcanzadas y las suyas propias.

Ejercicio 1: Dada la siguiente expresión escrita en lenguaje C y considerando que MAX es una constante numérica que fue definida en una instrucción al pre-procesador, indique el significado de tal declaración:

```
int (*apun_vector [MAX]) ();
```

Posible solución:
Para interpretar la declaración se aplica la regla "derecha-izquierda"; lo primero que se busca es el nombre de un identificador. En este caso se encuentra "apun_vector", luego, se examina a la derecha del mismo y se observa que se trata de un arreglo de tamaño MAX. Enseguida se mira a la izquierda para determinar el tipo de vector y se percibe la presencia del operador "*"; esto indica que se trata de un arreglo de apuntadores. Para determinar que se apunta se vuelve a examinar a la derecha y se encuentra los "( )". Esto refleja que se apunta a funciones. Solo resta que tipo de funciones, y ello se determina al mirar a la izquierda y hallando "int". Como no hay más nada a la derecha el proceso se da por culminado. Así pues, la declaración significa: un arreglo de apuntadores, llamado "apun_vector" con MAX posiciones de memoria, que señalan a funciones que devuelven enteros.

El ejercicio tiene por finalidad ayudar a que el estudiante comprenda que en la práctica diaria los programadores relacionan arreglos, funciones y apuntadores estrechamente y que aprenda a descifrar las declaraciones engorrosas que se pueden escribir en lenguaje C.

Ejercicio 2: Sean "apun_var1" y "apun_var2" dos apuntadores de tipo entero y "var" una variable entera. Asume además la siguiente instrucción de referenciación: "apun_var1=&var;" Explique si la siguiente operación que transfiere un valor derecho (valor-d) sin emplear una operación de desferenciación es válida de utilizarla:

```
apun_var2=apun_var1;
```

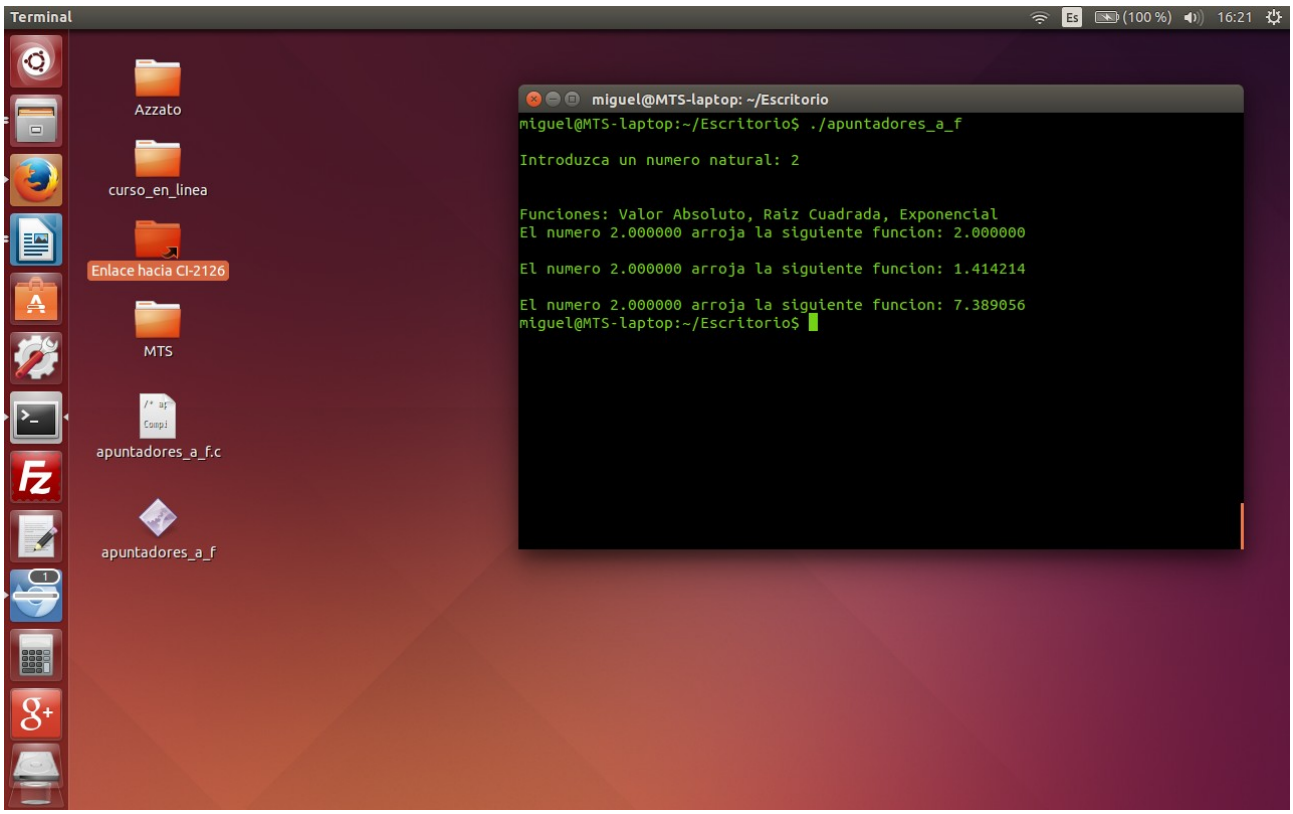
Posible solución:
Si es válida, ya que los apuntadores también son variables y en este caso su tipo es el mismo, por lo cuál ni siquiera requiere una conversión. Por lo tanto, es válido transferir su contenido (valor-d) entre ambas y no resulta necesario emplear el operador de deferenciación "*". Para este caso la instrucción hace que "apun_var1" y "apun_var2" apuntan a la misma variable "var".

El ejercicio tiene por objeto que el estudiante aclare los conceptos de variable apuntadora, valor-i, valor-d, operaciones de *referenciación* y de *dereferenciación*. Estos conceptos son útiles de conocer dado que muchos compiladores dan mensajes sobre apuntadores que los incluyen.

Ejercicio 3: Examine con cuidado el siguiente programa y después explique con detalle su funcionamiento:

```
/* apuntadores_a_f.c v1.0

Compilarlo con esta instruccion: gcc apuntadores_a_f.c -o apuntadores_a_f -lm

*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main() {

        /* --- Declaración e inicialización ---
           fabs es una función de valor absoluto
           sqrt es una función de raíz cuadrada
           exp es una función de exponente -incluye el número de Euler-
           y operación es un vector de 3 posiciones
           que almacenará funciones */
        static double (*funcion[3])(double) = {fabs, sqrt, exp};
        float x;
        int i;      /* Indice para recorrer el vector */

        fprintf(stdout, "\nIntroduzca un numero natural: ");
        scanf("%f", &x);
        fprintf(stdout, "\n\nFunciones: Valor Absoluto, Raiz Cuadrada, Exponencial");

        for (i=0; i<3 ;++i)
                fprintf(stdout, "\nEl numero %f arroja la siguiente funcion: %lf \n", x, (funcion[i])(x));

        exit (0);
}
```

Posible solución:
El código muestra cómo se puede usar un apuntador a una función. La idea es que los estudiantes comprendan que el código también se almacena en memoria y que puede ser direccionado. Incluso alterado. Todo lo que se ejecuta en un computador puede ser procesado y el lenguaje C es ideal para ello. El trasfondo de todo es que el lenguaje C provee un medio para tomar el control del computador o de otro dispositivo.

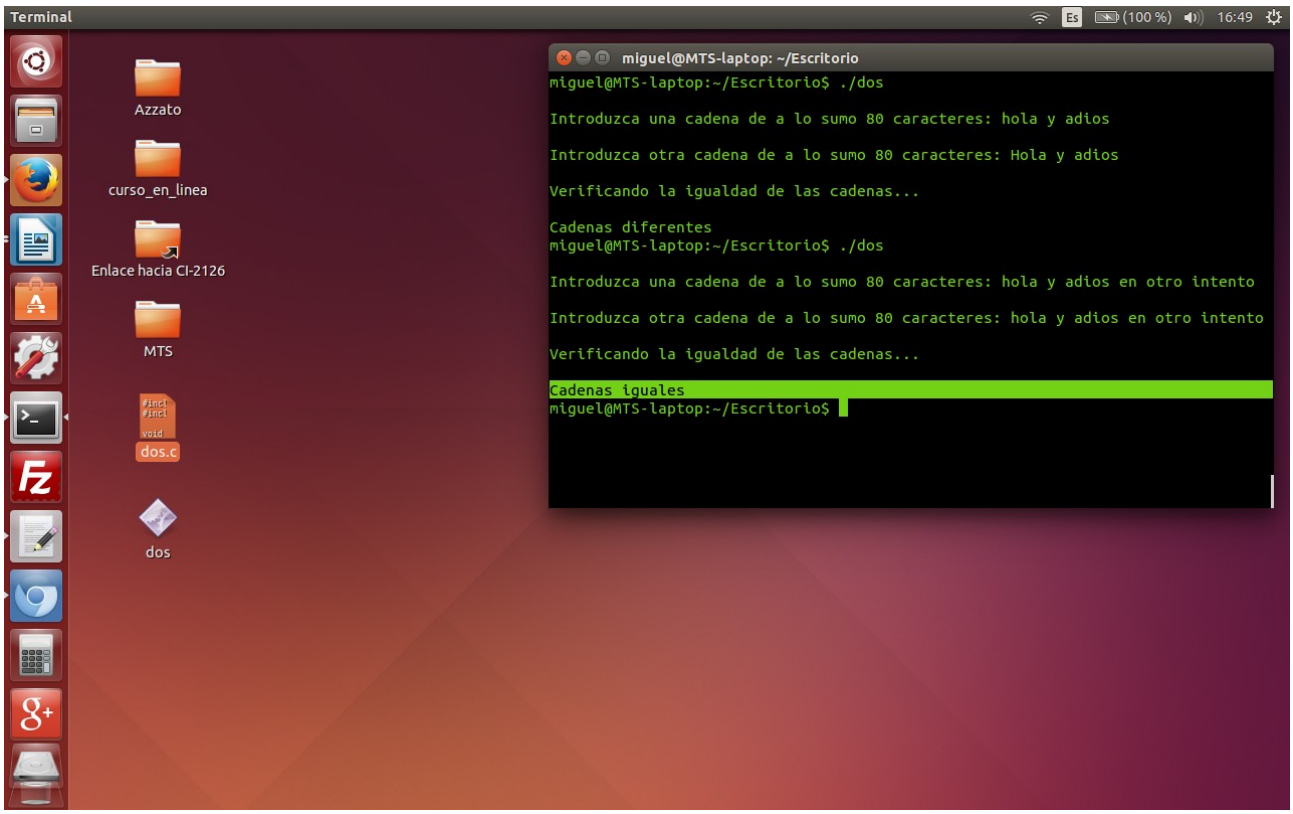


Ejercicio 4: Examine con cuidado el siguiente programa y después explique con detalle su funcionamiento:

```
#include <stdio.h>
#include <string.h>

void prueba (char *a, char *b, int (*cmp)(const char *st1, const char *st2))
{
   printf ("\nVerificando la igualdad de las cadenas...");
   (!(cmp) (a, b))? printf ("\n\nCadenas iguales\n"): printf ("\n\nCadenas diferentes\n");
}

int main ()
{
   char s1[80], s2[80];
   int (*apunt)(const char *c1, const char*c2);

   /* Asigna la dirección de la función en el apuntador */
   apunt=strcmp;

   printf ("\nIntroduzca una cadena de a lo sumo 80 caracteres: ");
   gets (s1);
   printf ("\nIntroduzca otra cadena de a lo sumo 80 caracteres: ");
   gets (s2);
   prueba (s1, s2, apunt);
}
```

Posible solución:

El código muestra un posible uso de un apuntador a una función.



MTS / 01-07-2014