

Preguntas teóricas tipo final

Ing. Ignacio Bonelli, Ing. Héctor Levi, Leandro Kollenberger

Sábado 17 de Octubre de 2015

Preguntas

1. ¿Puede ser usada la función *write* para escribir en un archivo abierto con *fopen*? ¿Por qué?
2. ¿Cuál es la principal diferencia entre una biblioteca *shared* y una estática?
3. ¿A qué equivale el nombre de una función en C?
4. ¿Cuanto lugar ocupa (en memoria) un puntero a función?
5. ¿Cuál es la principal diferencia entre el protocolo TCP y el UDP?
6. Describa brevemente el proceso de compilación. ¿De qué se encarga el *linker*?
7. ¿Cómo se especifica la inclusión de una librería en nuestro programa?
8. ¿A qué se le denomina ámbito de una variable?
9. Indique cuál es el tamaño del tipo de dato *varios*. Justifique.

```
union varios {  
    int entero;  
    float flotante;  
    char letra;  
    double z;  
    void (*leandro)(void);  
}
```

10. Considere el siguiente bloque de código, cuya distribución en memoria se muestra en la figura:

0xF000	arr[0]	int arr[4] = {2500, 3500, 4500, 5500};
0xF004	arr[1]	int *ptr1, *ptr2;
0xF008	arr[2]	ptr1 = arr;
0xF00C	arr[3]	ptr2 = &arr[2];

Escriba la salida del programa al lado de cada línea de código. Las instrucciones se ejecutan en secuencia.

```
printf("%d\n", (ptr2-ptr1));
printf("%d\n", ((int)ptr2-(int)ptr1));
printf("%d\n", (*ptr2-*ptr1));
printf("%x\n", ptr2);
printf("%x\n", --ptr2);
printf("%x\n", ptr2++);
printf("%d\n", *(ptr2++));
printf("%x\n", ptr2);
```

11. Dados los siguientes bloques de código, indique si compilan (con o sin warnings), si funcionan correctamente, y justifique las posibles correcciones que realizaría para que tengan el resultado correcto deseado.

- (a)

```
int a = 10;
int *ptri = NULL;
double x = 5.0;
double *ptrf = NULL;
ptri = &a;
ptrf = &x;
ptrf = ptri;
```
- (b)

```
char *ptr;
*ptr = 'a';
```
- (c)

```
int n;
int *ptr = &n;
ptr = 9;
```
- (d)

```
int *ptr = NULL;
*ptr = 9;
```

12. Escriba la salida de pantalla que genera el siguiente bloque de código:

```
int main(void) {
    int arreglo[3][2] = { {8,2}, {20,3}, {5,10}};
    int **ptr;
    int fila, col;
    arreglo[2][1] = 12;
    ptr = arreglo;
    printf("%d\n", **ptr);
    fila = 2;
    col = 1;
    printf("%d\n" ptr[fila][2*col]);
    ptr = &arreglo[1];
    printf("%d\n", **ptr);
    return 0;
}
```

13. Elija las opciones que considere correctas, justifique las falsas:

El archivo `stdio.h` que se utiliza mediante la sentencia `#include <stdio.h>...`

- (a) Contiene código compilado, pero sin linkear.
- (b) Contiene funciones compiladas de entrada y salida.
- (c) Al incluirlo se copian solo los prototipos de aquellas funciones utilizadas en nuestro código fuente.
- (d) En la etapa de linkeo se copia el contenido completo del archivo a nuestro programa fuente.
- (e) Contiene prototipos de funciones.
- (f) Contiene texto.
- (g) Al compilar, en la etapa de precompilación, se copia el contenido completo del archivo a nuestro programa fuente.

14. Dado el siguiente código, que fue compilado con el comando `gcc ./programa.c -o ./programa.bin -Wall`, indique:

- Si al compilar el programa se encontrarán *warnings*.

- Si al compilar el programa se encontrarán *errores*.
- Si el compilador generará o no el archivo ejecutable.
- En caso de que el archivo ejecutable se genere, indique qué resultado se vería en la pantalla si se lo ejecuta.
- En caso de que el archivo ejecutable no se genere, indique qué modificaría para corregirlo y por qué.

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

void main(void) {
    float numero = 3.2;
    int *ptr;
    ptr = &numero;
    printf("%f\n", numero);
    printf("%f\n", sqrt(*ptr));
    return 0;
}
```

15. Indique la salida de pantalla del siguiente bloque de código:

```
... funeval(          ,          ); // completa el prototipo
int main(void) {
    int x = 5;
    int y = 3;
    funeval(&x, y);
    printf("%d, %d\n", x, y);
}
void funeval(int *c, int d) {
    int aux;
    aux = *c;
    *c = d;
    d = aux;
}
```

16. Dado el siguiente bloque de código, indique cuál es la respuesta correcta y justifique:

```
int *entero;
float *decimal;
```

- `sizeof(entero) > sizeof(decimal)`
- `sizeof(entero) = sizeof(decimal)`
- `sizeof(entero) < sizeof(decimal)`

17. ¿Cuál es la diferencia entre un header y una librería? ¿Qué contiene cada uno? ¿Donde especificamos la inclusión de un header en nuestro programa?

18. Explique la diferencia entre una variable local y una global.

Respuestas

1. No, debido a que *write* opera con *file descriptors*, mientras que *fopen* trabaja con estructuras *FILE*.
2. Una librería estática es un tipo de librería que es linkeada de manera estática, es decir, su código binario se embebe en la aplicación en el momento de compilación. En contraste, una librería *shared* presenta una única copia de sí en los directorios del sistema, y su contenido se carga en memoria al llamar a la aplicación que la requiera.
3. El nombre de una función equivale a su dirección de comienzo en memoria.
4. Un puntero a función ocupa el mismo tamaño que cualquier tipo de puntero, el tamaño del bus de direcciones del microprocesador.
5. El protocolo TCP incluye un sistema de control de flujo y confiabilidad de paquetes, es decir, controla que todos los paquetes lleguen al destino indicado, de manera sincrónica. Por otro lado, UDP no presenta estos controles de transferencia, por lo tanto es menos confiable y más rápido que TCP.
6. El proceso de compilación comienza con el *preprocesador de C*, que se encarga de resolver ciertas directivas en el código, como por ejemplo *#include*, o *#define*. Luego de esto, el compilador se encarga de transformar ese código C en lenguaje *assembly* de la arquitectura destino, para que el *assembler* lo transforme en código máquina. Por último, el *linker* se encarga de enlazar las diferentes unidades de compilación con las librerías del sistema, y la librería estándar de C, si correspondiese.
7. Además de requerir los prototipos de las funciones a utilizar, necesitamos indicarle al linker el nombre de la librería que queremos incluir.
8. El ámbito de una variable indica el fragmento de código en el que esa variable puede ser accedida. En el caso de una variable local, por ejemplo, sólo reside en memoria hasta que la función retorne, salvo que sea declarada como *static*.
9. El tipo de dato *varios* ocupa 8 *bytes*, debido a que es una unión, y su tamaño está dado por la variable de mayor tamaño, en este caso *double* z.
10.

8
8
2000
F008
F004
F004
5500
F00C
11. (a) Compilará, aunque con warnings, ya que *ptrf* es un puntero a *double*, y *ptri* es un puntero a *int*.
(b) Compilará sin *warnings*, pero funcionará incorrectamente, ya que se le está asignando un valor al contenido de lo apuntado por una variable tipo puntero sin inicializar.
(c) Compilará con warnings, ya que 9 es un valor del tipo *int*, y *ptr* contiene una dirección de memoria. Este código funcionará correctamente (a *ptr* se le asignará la dirección de memoria 9) pero probablemente éste no sea el resultado deseado. En ese caso, deberá agregarse el operador de desreferencia, de manera que quede **ptr = 9*, y se le asigne el valor 9 a la variable *n*.
(d) El código compilará correctamente y sin warnings, pero al ejecutarse provocará una violación de segmento, ya que se está intentando de asignarle un valor a lo apuntado por un puntero marcado previamente como *NULL*.
12.

8
12
20
13. (a) Falso, *stdio.h* contiene prototipos de funciones.
(b) Falso, ídem anterior.
(c) Falso, se copia todo el contenido del archivo, incluyendo todos los prototipos incluidos en él.
(d) Falso, se copia en la etapa de precompilación.
(e) Verdadero.

- (f) Verdadero.
- (g) Verdadero.
14. El programa no compilará, debido a que falta indicarle al linker la necesidad de incluir la librería matemática, con el parámetro `-lm`. Luego de agregar esto el programa compila, pero con un *warning* en la línea que dice `ptr = &numero`, debido a que `ptr` es un puntero a `int`, pero `numero` es un `float`. El resultado en pantalla será el valor 3.2, y el resultado de su raíz cuadrada.
 15. El prototipo resultante será `void funeval(int *, int);`. El resultado por pantalla del programa será 3, 3.
 16. `sizeof(entero) = sizeof(decimal)`, debido a que ambos son punteros y ocupan el mismo espacio en memoria.
 17. Un *header* incluye los prototipos de las funciones que son incluidas en las librerías, mientras que las librerías contienen el código máquina compilado y listo para linkear con la aplicación. El *header* se incluye al comienzo de nuestro archivo fuente, con la directiva `#include` del preprocesador de C.
 18. Una variable local es únicamente accesible dentro de la función en la que fue declarada, mientras que una variable global es accesible por todas las funciones que formen parte de la misma unidad de compilación.