Fundamentos del Software

Relación de Problemas 3. Compilación y Enlazado de Programas

1. Un procesador (CPU) puede interpretar y ejecutar directamente las instrucciones de un programa en:
- (a) Lenguaje de alto nivel de tipo intérprete.
- (b) Lenguaje ensamblador o en lenguaje máquina, cualquiera de los dos.
- (c) Sólo lenguaje máquina.
- (d) En pseudocódigo o en lenguaje ensamblador.

2. ¿Es lo mismo un token que un lexema? Muestre algún ejemplo.

3. ¿El compilador es la única utilidad necesaria para generar un programa ejecutable en una computadora?

4. El análisis léxico es una etapa de la compilación cuyo objetivo es:
- (a) Extraer la estructura de cada sentencia, reconociendo los componentes léxicos (tokens) del lenguaje.
- (b) Descomponer el programa fuente en sus componentes léxicos (tokens).
- (c) Extraer el significado de las distintas construcciones sintácticas y elementos terminales.
- (d) Sintetizar el programa objeto.

5. El análisis sintáctico es una etapa de la compilación cuyo objetivo es:
- (a) Extraer la estructura de cada sentencia, reconociendo los componentes léxicos (tokens) del lenguaje.
- (b) Descomponer el programa fuente en sus componentes léxicos (tokens).
- (c) Extraer el significado de las distintas construcciones sintácticas y elementos terminales.
- (d) Sintetizar el programa objeto.

6. Para el siguiente código que aparece a la izquierda en lenguaje C++ (fichero `test.cpp`), indique el nombre de la fase en la que el compilador produce el mensaje de error que aparece a la derecha y explique la naturaleza del mismo:

```
01 int main (void)
02 {
03     int i;
04     char* j;
05
06     j = i;
07
08     if (i == 0)
09         i += ;
10
11     ¬;
12
13     return 0;
14 }
```

a) `test.cpp:9: error: expected primary-expression before ';' token`

b) `test.cpp:6: error: invalid conversion from 'int' to 'char*'`

c) `test.cpp:11: error: stray '\302' in program`

7. Muestre un ejemplo a partir de una sentencia en lenguaje C++ en la que un error léxico origine un error sintáctico derivado y otro error léxico que no derive en error sintáctico.

8. Muestre un ejemplo a partir una sentencia de en lenguaje C++ en la que un error léxico origine un error sintáctico y semántico derivados y otro error léxico que no los derive.

9. ¿Sería siempre posible realizar la depuración de un archivo objeto? Razone la respuesta.

10. Dado un programa escrito en lenguaje ensamblador de una arquitectura concreta, ¿sería directamente interpretable ese código por esa computadora? En caso contrario ¿qué habría que hacer?

11. ¿Sería necesario usar siempre el enlazador para obtener un programa ejecutable?

12. Dado un único archivo objeto, ¿podría ser siempre un programa ejecutable y correcto simplemente añadiendo la información de cabecera necesaria?

13. Dado un programa ejecutable que requiere de una biblioteca dinámica, ¿por qué no es necesario recompilar el código fuente de dicho programa si se modifica la biblioteca?

14. Indique en qué fase del proceso de traducción y ejecución de un programa se realizará cada una de las siguientes tareas:
   (a) Enlazar una biblioteca estática.
   (b) Eliminar los comentarios del código fuente.
   (c) Mensaje de error de que una variable no ha sido declarada.
   (d) Enlazar una biblioteca dinámica.

15. Indique en qué fase o fases del proceso de compilación de un lenguaje de programación de alto nivel se detectarían los siguientes errores:
   (a) Una variable no está definida.
   (b) Aparece un carácter o símbolo no esperado.
   (c) Aparecen dos identificadores consecutivos.
   (d) Aparecen dos funciones denominadas bajo el mismo nombre.
   (e) Aparece el final de un bloque de sentencias pero no el inicio del mismo.
   (f) Aparece un paréntesis cerrado y no se ha podido emparejar con su correspondiente paréntesis abierto.
   (g) Una llamada a una función que no ha sido definida.
   (h) En la palabra reservada `main` aparece un carácter extraño no esperado, por ejemplo `mai¿n`.

16. ¿Todo error sintáctico origina un error semántico? En caso contrario, demuéstrelo usando algún contraejemplo.