Un diccionario persistente se almacena en un archivo en formato binario. El diccionario se representa mediante un árbol binario de búsqueda en donde cada nodo del árbol utiliza un área contigua en el archivo con los siguientes campos:

- *izq*: Un entero de 4 bytes (en formato binario *little endian*) correspondiente al desplazamiento a partir del cual se encuentra la raíz del subárbol izquierdo. Es -1 si no posee subárbol izquierdo. El desplazamiento 0 corresponde a la raíz del árbol.
- *der*: Como *izq* pero para el subárbol derecho.
- *tam_llave*: Un entero de 2 bytes (en formato binario *little endian*) correspondiente al tamaño en caracteres de la llave almacenada en el nodo.
- *tam_valor*: Un entero de 2 bytes (en formato binario *little endian*) correspondiente al tamaño en caracteres del valor asociado a la llave de ese nodo.
- *llave*: Secuencia de *tam_llave* bytes correspondiente a la llave (codificada en ascii). No incluye terminación de string.
- *valor*: Secuencia de *tam_valor* bytes correspondiente al valor (codificado en ascii). No incluye terminación de string.

Un ejemplo de diccionario es el archivo *defs.dicc* incluido en los archivos adjuntos. No es legible al examinarlo con un editor de texto porque es un archivo binario, no de texto. Aún así podrá distinguir llaves y valores, pero no desplazamientos ni tamaños. El programa *revisar.c* incluido verifica la consistencia de un diccionario y muestra su contenido. Estudie este programa, ya que le servirá para resolver esta tarea. Compile y ejecute *revisar.c* con este comando:

*make revisar*

En esta tarea se pide programar el comando *./definir.bin* que recibe como parámetros el archivo que guarda un diccionario, una llave y una definición. El comando busca eficientemente la llave en el diccionario, si la llave no existe usted deberá agregarla donde corresponda, en caso contrario usted deberá reportar un error sin modificar el archivo. Debe programar este comando en el archivo *definir.c*. Ejemplo de ejecución:

*./definir.bin new-dic.dicc perro mamifero domestico de la familia de los canidos*

El test de prueba verifica que su solución haga exactamente lo mismo que *./prof.ref-$(arch)*. En particular revisa que la salida estándar, la salida estándar de errores y el código de retorno sean iguales.

Observe que en esta tarea **Ud. debe programar la función *main***. En las tareas anteriores la función *main* estaba dada en el archivo con los tests de prueba de su solución.

## *Requerimientos:*

- Debe ser eficiente: Minimice la cantidad de nodos que se necesita leer para encontrar la llave, aprovechando que es un árbol binario de búsqueda. No puede leer secuencialmente el archivo. Use *fseek* para posicionarse directamente en el desplazamientos que está en los campos *izq* o *der*. La raíz del árbol está en la posición 0.
- Si no puede leer el diccionario, debe diagnosticar el error con la función *perror*.
- Si la llave no existe o no se recibe la cantidad de parámetros correcta, debe diagnosticar el error en la salida estándar de errores, con el mismo mensaje que *./prof.ref-$(arch)*.

## *Instrucciones*

Descargue *t4.zip* de U-cursos y descomprímalo. Ejecute el comando *make* sin parámetros en el directorio *T4* para recibir instrucciones adicionales.

## *Entrega*

Ud. solo debe entregar por medio de U-cursos el archivo *definir.zip* generado por el comando *make zip*. **A continuación es muy importante que descargue de U-cursos el mismo archivo que subió, luego descargue nuevamente los archivos adjuntos y vuelva a probar la tarea tal cual como la entregó**. Esto es para evitar que Ud. reciba un 1.0 en su tarea porque entregó los archivos equivocados. Créame, sucede a menudo por ahorrarse esta verificación. Se descontará medio punto por día de atraso. No se consideran los días de receso, sábados, domingos o festivos.