6. Procesamiento del archivo de configuración.

Entre los objetivos a conseguir se encontraba que el sistema fuera lo suficientemente versátil como para permitir que el menú de usuario pudiese ser configurado en varias formas, pero a su vez se requería que el método de configuración fuera sencillo para el administrador del sistema.

El modo de configuración del sistema se ha decidido basándose en estos objetivos, y consiste en un archivo de configuración estilo Unix escrito en XML. El formato de archivo así como las marcas XML -lo que se conoce como la DTD, *Document Type Description*- han sido elegidas arbitrariamente pero con criterio, siempre persiguiendo la simplicidad.

Se describe a continuación tanto el formato del archivo de configuración como el código asociado a su procesamiento y a la representación de los datos en memoria. La parte referente al archivo de configuración se encuentra comentada igualmente en el manual de administración y configuración de OperAIT, como anexo a esta memoria.

6.1. El formato del archivo operait.conf.

Toda la configuración de la aplicación OperAIT se describe en el archivo `operait.conf`, que se leerá e interpretará al iniciarse la misma y cada vez que ésta reciba una orden de reconfiguración, como luego veremos.

Este archivo consiste en una estructura en forma de árbol XML, con las siguientes marcas XML válidas:

Marca XML	*Contenido*
`configdoc`	Engloba el documento de configuración al completo. Es la marca padre de todas.
`fileoptions`	Contiene las opciones relativas a ficheros y directorios.
`tracefile`	Fichero donde se guarda la información de monitorización o *traza*.
`wavdir`	Directorio donde se almacenan los archivos de audio.
`wavinvalopt`	Archivo de audio para “Opción no válida”.
`wavpress`	Archivo de audio para “Pulse”.

Marca XML	*Contenido*
`wav0`	Archivo de audio para “0”.
`wav1`	Archivo de audio para “1”.
`wav2`	Archivo de audio para “2”.
`wav3`	Archivo de audio para “3”.
`wav4`	Archivo de audio para “4”.
`wav5`	Archivo de audio para “5”.
`wav6`	Archivo de audio para “6”.
`wav7`	Archivo de audio para “7”.
`wav8`	Archivo de audio para “8”.
`wav9`	Archivo de audio para “9”.
`wava`	Archivo de audio para “Asterisco”.
`wavb`	Archivo de audio para “Almohadilla”.
`traceoptions`	Contiene las opciones sobre los datos que se deben monitorizar.
`progstarttime`	Si existe[3], se monitorizará el inicio del programa.
`connstarttime`	Si existe[3], se monitorizará el inicio de cada conexión.
`connendtime`	Si existe[3], se monitorizará el fin de cada conexión.
`keyboption`	Si existe[3], se monitorizará las opciones seleccionadas mediante teclado.
`reconftime`	Si existe[3], se monitorizará la reconfiguración “en caliente” del sistema.
`redirvalue`	Si existe[3], se monitorizará el código de error devuelto por la red al realizar una transferencia de llamada.
`menutree`	Contiene los datos del menú de usuario. Puede constar del atributo `label`.
`snd`	Fichero de audio a reproducir.
`option`	Submenú al que se accede si se selecciona el valor del atributo[4] `code`. Puede constar del atributo `label`.
`redir`	Número al que se debe realizar la transferencia de llamada.
`goto`	Nombre de una etiqueta definida mediante el atributo `label` de una marca anterior, y que indica el submenú que debe seguirse procesando.

Tabla 5. Marcas XML y su contenido.

6.1.1. Sintaxis del archivo de configuración.

Las marcas XML que forman la configuración del sistema deben situarse en el fichero de configuración siguiendo el las siguientes reglas:

[3] Aquí *existe* significa que la marca aparezca en el fichero. El contenido entre el inicio y el fin de la marca debe ser nulo, o se interpretará como un error.

[4] Este atributo es obligatorio.

- Todas las marcas XML deben estar contenidas dentro de la marca madre `configdoc`.
- Las marcas XML que pueden contener otras marcas, así como las posibles marcas hijas se resumen en la siguiente tabla[5].

Marca XML madre	*Posibles hijas*
`configdoc`	`fileoptions`, `traceoptions` y `menutree`.
`fileoptions`	`tracefile` y `wav*`.
`traceoptions`	`*time`, `keyboption` y `redirvalue`.
`menutree`	`snd`, `option`, `redir` y `goto`.
`option`	`snd`, `option`, `redir` y `goto`.

Tabla 6. Marcas XML contenedoras.

- Las siguientes marcas no pueden repetirse en el fichero: `configdoc`, `fileoptions`, `traceoptions`, `menutree`, `tracefile`, `wav*`.
- El campo atributo `code` de la marca XML `option` no tiene por qué ser único dentro de un mismo submenú. Es decir, puede haber marcas `option` hermanas con el mismo valor del atributo `code`. En tal caso, sólo la primera de ellas tendrá efecto a la hora de decidir el recorrido del menú cuando un usuario elija una opción de teclado.
- Todas las marcas, salvo las de opciones de traza, deben contener bien otras marcas XML -si se trata de marcas madre- bien datos de configuración -como números a los que transferir llamadas, nombres de archivo, etc.-. Las marcas de opciones de traza no deben contener ningún dato.

6.2. Comportamiento del programa.

El archivo de configuración se ha creado para permitir al usuario modificar el comportamiento del programa OperAIT. Este comportamiento podrá modificarse en varios aspectos, enumerados a continuación:

- Archivos utilizados: tanto para obtener los datos vocales que escuchará el usuario final como los archivos de traza o los directorios donde se almacenan los datos.
- Opciones de traza: los eventos que quedarán registrados en el archivo de monitorización del programa.
- Comportamiento cara al usuario final: los mensajes que el usuario escuchará, las opciones que se le ofrecerán y el resultado de cada una de ellas.

[5] En la tabla, el carácter * indica cualquier combinación de letras, al estilo de las expresiones regulares POSIX.

6.2.1. Fichero utilizados.

Para la configuración de los ficheros que utilizará el programa se utilizan todas las marcas XML contenidas dentro de la marca madre `fileoptions`. Cada una de las posibles marcas hijas indicará un fichero de datos o un directorio que utilizará el programa.

También se utilizarán los ficheros cuyo nombre coincida con los datos que estén dentro de la marca XML `snd`. Estos ficheros estarán situados en el directorio cuyo nombre coincida con los datos dentro de la marca `wavdir`.

6.2.2. Opciones de monitorización.

La configuración de las opciones de monitorización es aún más sencilla. Para conseguir monitorizar un evento determinado, basta con incluir la marca XML asociada al mismo en el archivo de configuración.

Estas marcas XML no deberán contener datos, y podrá utilizarse por lo tanto la sintaxis XML <marca/> para indicar una marca vacía.

6.2.3. Comportamiento frente al usuario.

El comportamiento del programa desde el punto de vista del usuario final del mismo es algo más complejo de comprender a partir de fichero XML.

Para explicar el comportamiento del programa desde el punto de vista del usuario se utilizará una abstracción: se supondrá que el programa analiza directamente el archivo XML para determinar su comportamiento. En realidad OperAIT analiza el archivo de configuración una sola vez y transforma estos datos XML en estructuras internas, que son las que utiliza para determinar su comportamiento. Pero, puesto que los datos XML y las estructuras internas son equivalentes, es más sencillo explicar el comportamiento de OperAIT basándonos en el archivo de configuración.

El algoritmo que utiliza el programa a la hora de comunicarse con el usuario es el siguiente:

- Se comienza a procesar el archivo de configuración comenzando por la marca XML `menutree`.
- Se recorren secuencialmente todas las marcas hijas de `menutree`, exhibiendo el siguiente comportamiento:
  - Si se encuentra una marca `snd`, se reproducirá el archivo de sonido correspondiente, no procesando la siguiente marca hermana hasta el fin de la reproducción del mismo.

  - Si se encuentra una marca `redir`, se realizará una transferencia de llamada al número de teléfono indicado. La transferencia de llamadas finaliza forzosamente la conexión, por lo que no se sigue procesando el archivo de configuración.
  - Si se encuentra una marca `goto`, se comenzará a procesar el archivo a partir de la primera marca hija de la marca `option` cuyo atributo `code` coincida con los datos contenidos en la marca `goto`. Esto es equivalente a un salto hacia una etiqueta en lenguaje de programación C.
  - Si se encuentra una marca `option`, se reproducirá primero el archivo asociado a la marca `wavpress`. Después de esto se reproducirán los archivos asociados a las teclas del teléfono (marcas `wav0` a `wav9`, `wava` y `wavb`) en función del atributo `code` asociado.
- Cuando se ha terminado de procesar una marca, se continúa -salvo saltos o transferencias de llamada- con la siguiente marca hermana. Si no hay más marcas hermanas, se vuelve a comenzar por la primera de las hermanas. Este bucle se repite hasta tres veces si el usuario no selecciona una opción correcta[6].
- La cuarta vez que se entra en un bucle (o submenú) se desconecta al usuario. Entre dos ejecuciones del bucle se establece un período de silencio de 2,048 segundos, para indicar al usuario que se vuelve a comenzar por el principio del mismo.
- Cuando el usuario selecciona por teclado una opción correcta[7], se comienza a procesar la primera marca hija de la marca `option` cuyo atributo `code` coincide con el código de opción introducido. Esto es equivalente a "entrar" en un submenú.
- Si el usuario ha introducido códigos de teclado suficientes como para descartar que vaya a realizar una selección correcta, se reproducirá el mensaje correspondiente a la marca XML `waverror`. Cuando esta termine, se continúa procesando la primera de las marcas hermanas del submenú actual. Esto cuenta como una nueva entrada en el submenú, por lo que el usuario también será desconectado realiza más de dos intentos de selección fallidos.
- No se espera nunca a que los archivos de sonido terminen de reproducirse si el usuario ha pulsado suficientes códigos de teclado como para decidir si se ha producido una selección correcta o ya no puede seleccionarse correctamente ninguna

[6] Una opción correcta es aquella cuyo código de opción coincide con alguno de los atributos code de las marcas option dentro del submenú que se está procesando.

[7] Debido al algoritmo que procesa los nodos, existen dos casos especiales a tener en cuenta a la hora de crear los archivos de configuración. Estos casos se comentan en la sección 7.3.3, página 64.

opción: en ambos casos se dejará de reproducir el mensaje actual y se efectuará la acción correspondiente (salto a un nuevo submenú o reproducción del mensaje de error).

6.3. Uso del archivo de configuración.

Puede comprobarse que construir un árbol de menú lo suficientemente complejo como para diferenciar entre personal docente y de administración, becarios, otros servicios automáticos como SNAIT, etcétera es bastante sencillo: basta grabar los archivos de sonido correspondientes a los nombres de las personas y servicios, los archivos básicos de números y mensajes "error", "pulse", etc. y construir el árbol XML asociado, reservando submenús para cada grupo accesible de personas o servicios.

Para permitir que la llamada se transfiera cuando el usuario pulse un código determinado, basta situar una única marca `redir` dentro de la marca `option` correspondiente.

Las etiquetas se utilizan para que los usuarios puedan saltar de un submenú a otro, por ejemplo para permitir volver al submenú anterior si el usuario se equivoca.

6.3.1. Caso práctico de configuración.

Para obtener una idea más clara de cómo utilizar el archivo de configuración de OperAIT para modificar el comportamiento del programa, se propondrá un ejemplo práctico de comportamiento deseado. A continuación se mostrará la solución al mismo en forma de archivo de configuración de OperAIT, y se describirán los archivos de audio que deben ser creados.

6.3.1.1. Comportamiento deseado.

Supongamos que se desea conseguir el siguiente comportamiento:

"*Deseamos que el programa OperAIT permanezca a la escucha de conexiones entrantes. Una vez se haya realizado una conexión, se registrará el momento de inicio y fin de la misma en el archivo de nombre operait.log.*

El programa se presentará al usuario como la "operadora automatizada" del área de Ingeniería Telemática y le indicará que puede contactar con el personal docente del área pulsando la tecla 1, o bien puede acceder al servicio de consulta de notas SNAIT mediante la pulsación de la tecla 2.

Si el usuario pulsa la tecla 2, se realizará una transferencia de llamada al número de teléfono 38, donde escucha la aplicación de consulta de calificaciones SNAIT.

*Si el usuario pulsa la tecla 1, se le indicará que podrá contactar con el primer profesor del área pulsando la combinación de teclas 01, con el segundo profesor pulsando la combinación de teclas 02 o volver al menú inicial mediante la pulsación de la tecla *. En el primer caso se transferirá la llamada al número 21; en el segundo se transferirá al número 22; en el tercer caso se volverá a reproducir el menú inicial, mensaje de bienvenida incluido.*

Durante todo el tiempo, se registrarán las opciones de teclado introducidas por el usuario, de forma que posteriormente pueda analizarse si el menú diseñado es demasiado complejo y da lugar a confusión o si es lo suficientemente sencillo como para no inducir a errores."

6.3.1.2. Configuración asociada.

Para conseguir este comportamiento es necesario disponer de los archivos de sonido básicos para todas las teclas del teléfono, así como archivos para los mensajes "pulse" y "error". Además, será necesario disponer de archivos para el mensaje de bienvenida y la indicación de cada una de las opciones seleccionables, por ejemplo "Hablar con el personal docente", "Realizar una consulta de notas", "Volver al menú anterior", etc.

El menú audible que se desea presentar al usuario, así como el comportamiento según las opciones seleccionadas, puede verse representado por la ilustración 8. El diagrama en forma de árbol representa, utilizando dibujos para representar nodos de tipo opción, sonido y transferencia de llamada, el menú que se presentará al usuario. Los códigos que activan cada selección y los nombres de los archivos de audio necesarios -aparte de los básicos- también se ven reflejados en el diagrama.

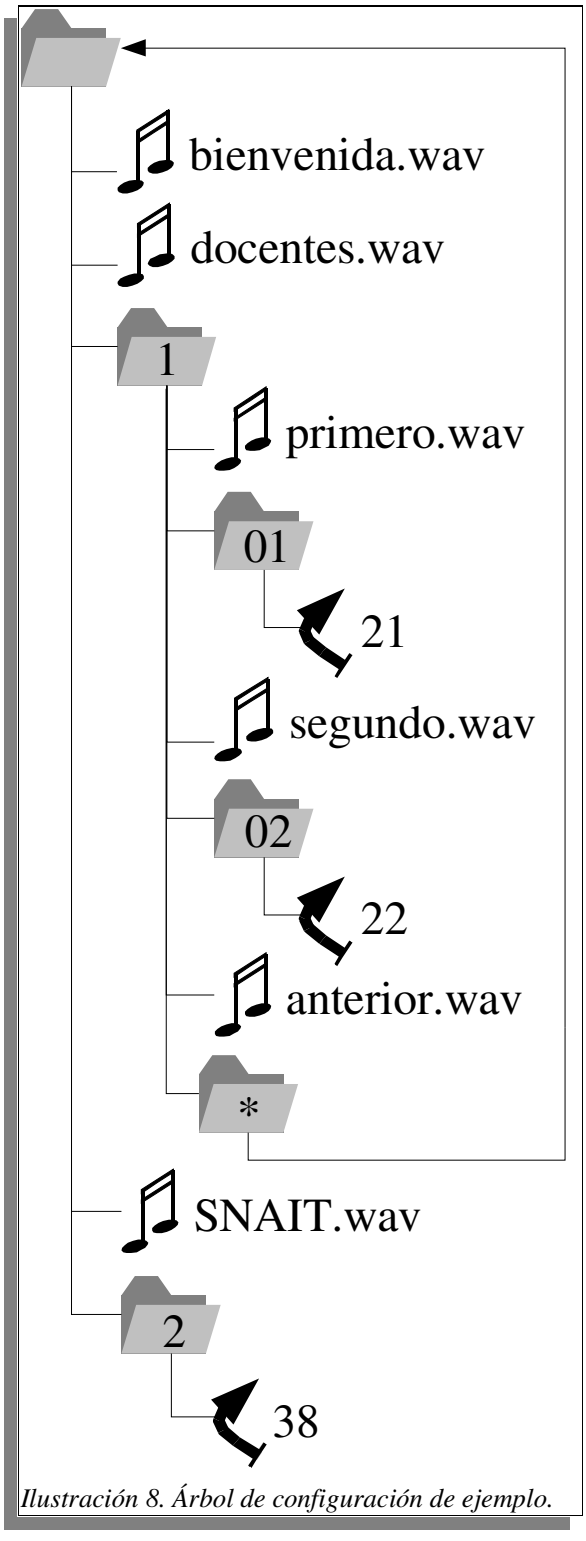

Ilustración 8. Árbol de configuración de ejemplo.

El enunciado del ejemplo indica que debe monitorizarse el inicio y el fin de cada conexión, así como las opciones de teclado introducidas por el usuario en cada instante. El archivo de monitorización debe llamarse operait.log. Supondremos que todos los archivos se encuentran dentro del directorio `/home/operait/sonidos`. Los nombres de los archivos de audio básicos utilizados serán triviales y no precisarán más explicación.

Un archivo de configuración válido asociado a este árbol y estas opciones de ficheros y monitorización puede ser el siguiente:

```
<!--  Archivo de ejemplo de configuración
      Autor: Manuel Márquez Garrido
      Fecha: 15 de julio de 2004
      Comprobado: Sí                    -->
<configdoc>
      <fileoptions>
            <tracefile>operait.log</tracefile>
            <wavdir>/home/operait/sonidos</wavdir>
            <wavinvalopt>novalido.wav</wavinvalopt>
            <wavpress>pulse.wav</wavpress>
            <wav0>0.wav</wav0>
            <wav1>1.wav</wav1>
            <wav2>2.wav</wav2>
            <wav3>3.wav</wav3>
            <wav4>4.wav</wav4>
            <wav5>5.wav</wav5>
            <wav6>6.wav</wav6>
            <wav7>7.wav</wav7>
            <wav8>8.wav</wav8>
            <wav9>9.wav</wav9>
            <wava>asterisco.wav</wava>
            <wavb>almohadilla.wav</wavb>
      </fileoptions>
      <menutree label="inicial">
            <snd>bienvenida.wav</snd>
            <snd>docentes.wav</snd>
            <option code="1">
                  <snd>primero.wav</snd>
                  <option code="01">
                        <redir>21</redir>
                  </option>
                  <snd>segundo.wav</snd>
                  <option code="02">
```

```
                        <redir>22</redir>
                  </option>
                  <snd>anterior.wav</snd>
                  <option code="*">
                        <goto>inicial</goto>
                  </option>
            </option>
            <snd>SNAIT.wav</snd>
            <option code="2">
                  <redir>38</redir>
            </option>
      </menutree>
      <traceoptions>
            <connstarttime/>
            <connendtime/>
            <keyboption/>
      </traceoptions>
</configdoc>
```

6.4. Representación interna de los datos de configuración.

Internamente, OperAIT no utiliza el árbol XML tal cual, sino que realiza una conversión de los datos contenidos en el mismo y los almacena en varias estructuras internas con las que realmente trabaja a la hora de determinar su comportamiento en cada instante. Sería muy costoso computacionalmente analizar el archivo de configuración en cada instante, por ello se utilizan estas estructuras internas de datos, mucho más simples y mucho más restringidas que un flujo de datos XML genérico.

Los datos se almacenan internamente utilizando tablas de estructuras y tablas de cadenas de caracteres. Esto se hace así porque los datos de configuración no cambian continuamente, sólo cuando el usuario ordena una reconfiguración del sistema. Resulta más sencillo calcular el tamaño de los datos y almacenarlos en dos bloques grandes de memoria dinámica a los que se accede como si fuera una tabla de estructuras que mantener listas ligadas de pequeños bloques de memoria, sobre todo a la hora de liberar la memoria.

6.4.1. Tipos de datos definidos.

Toda la configuración del sistema se referencia utilizando el tipo de datos `conf_data_struct`, cuya definición se incluye a continuación:

```
typedef struct {
      char *filenames[CONF_FILENO]; /* Nombres de los archivos */
      int traceflags[CONF_TRACEFLAGNO];  /* Opciones de traza */
```

```
      conf_node_struct *ptree;      /* Árbol de menú */
      unsigned node_number;   /* Número de nodos del árbol de menú */
      char *buffer;           /* Cadenas de caracteres utilizadas */
      unsigned buffer_length; /* Tamaño del grupo de cadenas */
} conf_data_struct;
```

Cada uno de los campos de la estructura `conf_data_struct` se describe en la siguiente tabla:

Nombre del campo	*Función*
`filenames`	Punteros al inicio de las cadenas de caracteres que almacenan el nombre de los archivos usados.
`traceflags`	Valores lógicos que indican si cada una de las opciones de monitorización está siendo registrada.
`ptree`	Puntero al inicio del árbol de menú.
`node_number`	Número de nodos del árbol de menú.
`buffer`	Bloque de memoria que almacena secuencialmente todas las cadenas de caracteres utilizadas.
`buffer_length`	Longitud de la zona de memoria que almacena secuencialmente las cadenas de caracteres.

Tabla 7. Campos de la estructura `conf_data_struct`.

Los datos referentes al menú de usuario se almacenan en una tabla de estructuras `conf_node_struct`, cuya definición se incluye a continuación:

```
typedef struct {
        unsigned nid; /* Identificador del nodo */
        unsigned fid; /* Identificador del padre */
        unsigned bid; /* Identificador del siguiente hermano */
        unsigned sid; /* Identificador del hijo */
        unsigned action; /* Acción asociada al nodo */
        char *WAV;      /* Nombre del archivo a reproducir si es un
                        sonido */
        char *code;     /* Código de activación si es una opción */
        char *redirNo;  /* Teléfono al que llamar si es una
                           transferencia de llamada */
} conf_node_struct;
```

Los campos `nid`, `fid`, `bid` y `sid` almacenarán valores de índices dentro de la tabla `ptree`, formada por estructuras de tipo `conf_node_struct`. Estos índices permiten crear una estructura lógica de árbol: cada nodo está identificado por su posición dentro de la tabla, y los índices anteriores son referencias al propio nodo, al nodo padre, al siguiente nodo hermano y al nodo hijo, respectivamente.

El campo `action` indica la función realizada por el nodo. Los posibles valores para el campo `action`, el nombre de la constante asociada declarada en el archivo `conf.h` y su función asociada se especifican en la siguiente tabla.

Número de acción	*Nombre de la acción*	*Función asociada*
1	`CONF_ACT_SND`	Reproducción de sonido
2	`CONF_ACT_OPTION`	Opción de teclado
3	`CONF_ACT_REDIR`	Transferencia de llamada
4	`CONF_ACT_GOTO`	Salto a otro nodo del árbol

Tabla 8. Tipos de acción y función asociada.

Por otra parte, los campos `WAV`, `code` y `redirNo` son punteros que apuntan al inicio de la cadena de caracteres asociada a la acción a realizar, ya sea reproducir un archivo de sonido, permitir una selección por teclado o transferir una llamada. La acción de salto a otro nodo del árbol no precisa ninguna cadena de caracteres, puesto que el identificador del nodo al que saltar es precisamente el campo `sid`.

El siguiente diagrama representa un posible estado del bloque de memoria al que apunta `ptree`.

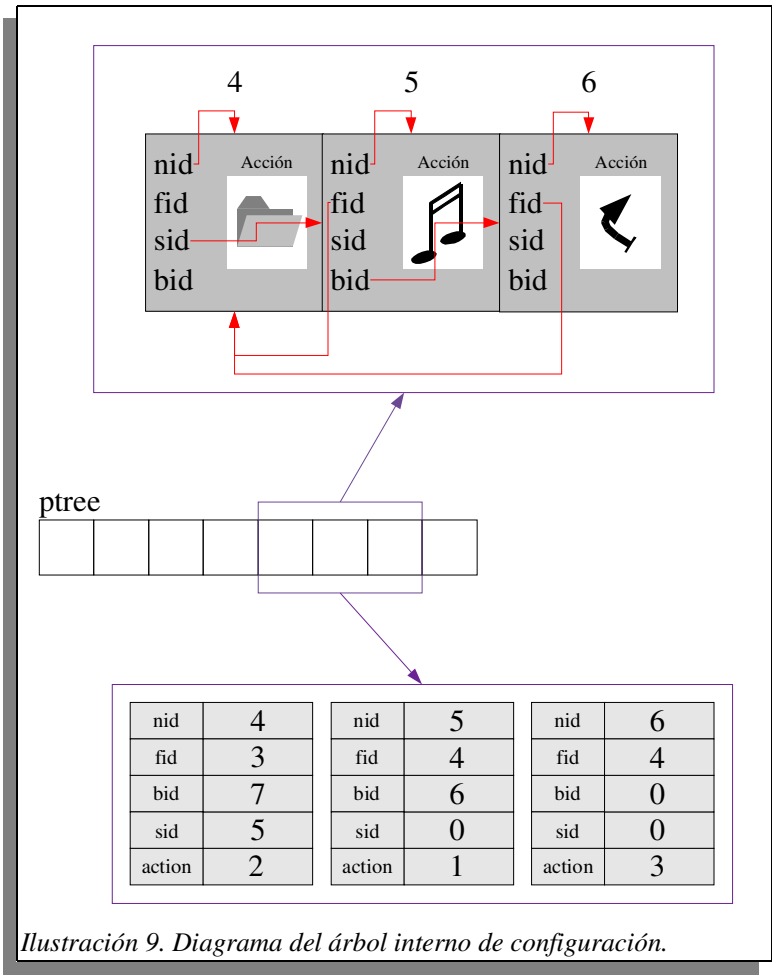

Ilustración 9. Diagrama del árbol interno de configuración.

Este caso hipotético muestra los nodos con identificadores 4, 5 y 6, los valores almacenados en sus campos más significativos y un esquema de cómo estos valores apuntan a cada otros elementos de la tabla, que representan otros nodos. El diagrama de árbol (sin identificadores de opción, número al que transferir la llamada o fichero de audio) asociado a este ejemplo sería el siguiente:

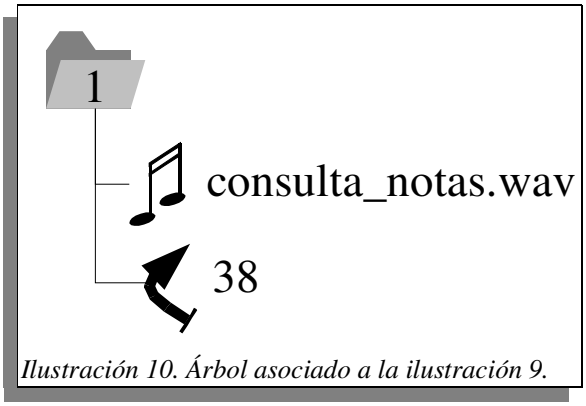

Ilustración 10. Árbol asociado a la ilustración 9.

Todas las cadenas de caracteres utilizadas se almacenan secuencialmente (terminador incluido) en forma de un sólo bloque de memoria. Cada uno de los punteros de carácter

utilizado apuntará al inicio de la cadena correspondiente dentro de este bloque. De este modo todas las cadenas se encuentran contenidas en un sólo bloque de memoria dinámica, siendo la liberación de memoria un proceso mucho más sencillo que si cada cadena utilizara su propio bloque de memoria dinámica.

Para conseguir esta simplicidad de uso de las estructuras de configuración es necesario pagar un pequeño precio: el proceso de análisis del fichero de configuración XML es algo más complejo y debe ser realizado en dos pasadas.

6.5. Análisis del archivo XML.

El archivo XML de configuración de la aplicación se analiza en dos situaciones posibles:

- Al inicio de la aplicación, para obtener la configuración inicial.
- Cada vez que el programa recibe la señal `SIGHUP`, para obtener una nueva configuración que anule la configuración anterior.

En ambos casos, se comprueba la validez sintáctica de la configuración en XML y se transforman estos datos de configuración en datos almacenados en la estructura interna de árbol definidas en el apartado anterior. La única diferencia de comportamiento sucede cuando existe un error en la configuración: en el primer caso, se indica error por la salida de errores y se termina el programa. En el segundo caso, se indica el error escribiéndolo en el archivo de monitorización y no se termina el programa, sino que se continúa utilizando la configuración anterior.

6.5.1. La biblioteca expat.

Para realizar el análisis del archivo XML se utiliza la biblioteca de procesamiento XML expat, que se distribuye con licencia libre similar a la licencia BSD. El uso que se hace de esta biblioteca está orientado a detectar qué marcas XML se utilizan en el archivo de configuración, en qué orden aparecen estas marcas y qué datos contienen las mismas. La biblioteca expat es bastante extensa y compleja, pero sólo se utiliza una parte mínima de la misma para el análisis de la configuración.

El elemento principal de la biblioteca expat es el tipo de datos opaco `XML_Parser`. Este tipo de datos se utiliza para crear una instancia de análisis de datos XML, de forma que sea posible instanciar varios analizadores desde el mismo programa, utilizando distintas variables de este tipo para cada uno de ellos. Las variables de tipo `XML_Parser` almacenan los datos utilizados internamente por la biblioteca expat de una forma transparente al usuario, que sólo

tienen que pasar como parámetro un puntero a esta variable en cada llamada a función de la biblioteca.

La creación de una instancia de análisis XML se realiza mediante la llamada a la función `XML_ParserCreate`. Esta función devuelve una variable de tipo `XML_Parser`, que será el nuevo analizador XML a utilizar. Para liberar la memoria asociada a un analizador, una vez éste no va a ser utilizado más, se invoca a la función `XML_ParserFree`. Si se desea reutilizar un analizador descartando la información que hubiera sido obtenida anteriormente se utilizará la función `XML_ParserReset`.

Para realizar el análisis se utiliza lo que se conocen como funciones *de callback*, concepto que ya se ha descrito anteriormente. Se pueden establecer distintas funciones según el evento detectado: inicio de etiqueta, fin de etiqueta, análisis de datos, etc. Para obtener la configuración se utilizan las funciones `XML_SetElementHandler` y `XML_SetCharacterDataHandler`. La primera establece las funciones manejadoras de los eventos inicio y fin de marca XML. La segunda establece la función que procesará el evento de fin de datos contenidos entre las dos etiquetas que delimitan una marca XML.

6.5.2. Algoritmo de análisis.

El algoritmo de análisis utilizado es bastante rudimentario, puesto que existen actualmente herramientas que permiten el análisis XML de forma sencilla y rápida. También podría haberse utilizado las herramientas Bison y Flex para generar un analizador LALR de la gramática basado en XML o en otra sintaxis a definir.

No se utilizó ninguno de estos métodos por desconocimiento de los mismos. Posteriormente podría haberse utilizado alguna de estas herramientas para obtener un diseño más elegante del analizador, pero puesto que éste ya estaba en funcionamiento y que no era el objetivo principal crear un analizador de gramática, se descartó prolongar el desarrollo del proyecto. Por lo tanto la comprobación de sintaxis y la extracción de datos se escribió sin más ayuda que el uso de la biblioteca expat.

El análisis del fichero se realiza en dos pasadas: en la primera se comprueba que la sintaxis del archivo es correcta, se procesan las etiquetas en busca de errores y se calcula la longitud de las zonas de memoria que almacenarán las cadenas de caracteres constantes. Asimismo se crea una lista simplemente enlazada de elementos que asocian los nombres de etiqueta al número de nodo asociado a las mismas.

En la segunda pasada se rellenan las estructuras internas que almacenarán los datos utilizando la lista de etiquetas anterior y las zonas de memoria que se habrán reservado a tal

efecto tras el cálculo realizado al final de la primera pasada. Entre la primera y la segunda pasada se invoca la función `XML_ParserReset` para reiniciar el analizar, puesto que desde el punto de vista del mismo se trata de realizar un nuevo análisis de datos XML.

Para realizar el análisis, puesto que se recorre un árbol, se utiliza una pila. Esta pila se ha implementado mediante una tabla de estructuras y tiene un límite -razonable- de 30 elementos, que correspondería con 30 niveles de anidamiento en el archivo de configuración. En cada instante se almacena la información del nodo actual en la cima de la pila, y cada uno de los nodos padres, abuelo, ... tendrán su información almacenada en el segundo, tercer, ... elemento. El siguiente diagrama muestra un ejemplo de configuración y el estado de la pila en un instante del análisis.

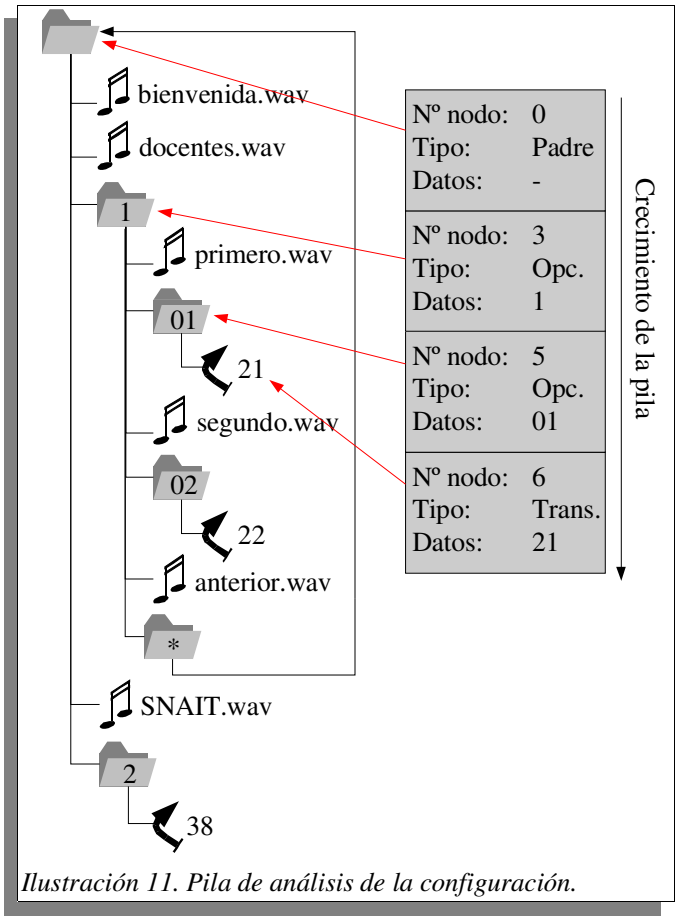

Ilustración 11. Pila de análisis de la configuración.

Puede observarse que la pila tiene en su cima un nodo de tipo “Transferencia de llamada”, cuyo nodo padre es un nodo de tipo “Opción”, que a su vez tiene un nodo de tipo “Opción” como padre y que finalmente tiene como padre al nodo inicial del árbol. De este modo, si hubiera algún error en el anidamiento de marcas XML éste sería detectado con sólo mirar el elemento anterior de la pila. Este método de análisis del árbol utilizando una pila es muy utilizado en analizadores de sintaxis.

6.6. Uso del módulo de análisis de la configuración.

Todas las rutinas de procesamiento del archivo de configuración XML se agrupan en un único módulo llamado `conf.o`. Si se desea utilizar este módulo en una aplicación -como se hace con la aplicación OperAIT- basta incluir el fichero de cabecera `conf.h` en el código fuente del programa y compilar el fichero `conf.c`, enlazando el módulo resultante `conf.o` al programa en cuestión.

El uso del módulo de análisis de configuración es bastante sencillo. La única función que se exporta es la función `GetConfiguration`, cuyo prototipo es el siguiente:

```
int GetConfiguration(int fd, conf_data_struct *pconf);
```

Los parámetros que esta función toma son un descriptor de fichero abierto, que deberá corresponder al archivo de configuración, y un puntero a una variable del tipo estructura `conf_data_struct` visto anteriormente. Los datos de la configuración obtenidos tras la ejecución de la función estarán almacenados en esta estructura.

Hay que destacar que el puntero pconf debe apuntar a una zona de memoria ya reservada, es decir, la función `GetConfiguration` no reserva la memoria para la misma. Asimismo, tampoco se cierra el descriptor de fichero asociado al archivo de configuración. Estas dos tareas debe realizarlas el programa principal.

El archivo `conf.h` define además los tipos de datos necesarios para su uso, ciertas cadenas de caracteres constantes y algunas variables para el control de errores.

6.6.1. Gestión de errores.

Durante el proceso de análisis pueden ocurrir diversos errores, que se comprueban en cada momento y se indican al usuario. Pueden distinguirse dos tipos diferenciados de condiciones de error: errores relacionados con el archivo de configuración y errores internos.

Los errores relacionados con el archivo de configuración ocurren cuando dicho archivo no cumple la sintaxis definida para el mismo. Ejemplos de errores de configuración son la inclusión de una marca madre[8] vacía, la repetición de un nombre de etiqueta en distintos campos `label`, o un nivel de anidamiento de marcas mayor a 30.

[8] Las marcas madre son, como hemos visto anteriormente, todas aquellas que deben pueden contener otras marcas XML.

Los errores internos ocurren debido al estado del sistema en el momento del análisis del fichero de configuración. Los errores internos que pueden ocurrir son la falta de memoria dinámica, necesaria para almacenar las estructuras de configuración y las estructuras temporales para el análisis, y errores de lectura del archivo debido a un fallo del sistema, por ejemplo un fallo del disco.

Si se produce algún error durante el análisis, la función `GetConfiguration` dejará de procesar el archivo de configuración y devolverá el código de error correspondiente. Los datos correspondientes al error se almacenan en la variable `conf_error_struct`, cuya declaración, junto con la definición de su tipo, se indica a continuación y se encuentra en el archivo `conf.h`.

```
struct tconf_error_struct{
        int errorcode;  /* Código de error */
        int errorline;  /* Línea donde se produjo el error */
        char errortext[CONF_MAXDATALEN]; /* Nombre del error */
} conf_error_struct;
```

De este modo, cuando el programa recibe una indicación de error en el valor de retorno de la función `GetConfiguration`, puede indicarle al usuario qué error se produjo mediante la constante de cadena de caracteres asociada y el número de la línea que provocó el error.