**PARTE I: Introducción al sistema SINTAC**

**PROLOGO**

**Presentación:**

El sistema SINTAC es un sistema integrado de desarrollo de aventuras conversacionales para ordenadores PC y compatibles. La versión actual, permite el desarrollo de aventuras de tipo conversacional en los modos gráficos de 16 y 256 colores de la tarjeta VGA.

El SINTAC está compuesto de varios programas entre los que se encuentra un intérprete‑debugger para analizar paso a paso la aventura creada, un compilador de bases de datos y un linkador que genera ficheros ejecutables de las bases de datos compiladas, además de un entorno integrado de programación que incluye su propio editor de textos y un completo sistema de ayuda en línea.

Aparte de estos que constituyen el núcleo principal del sistema se suministran algunas utilidades de apoyo como un editor de juegos de caracteres.

**Requerimientos de hardware:**

Esta versión funciona en cualquier ordenador compatible PC con al menos 512K de memoria instaladas, aunque es recomendable tener 640K sobre todo si se trabaja desde el entorno de desarrollo.

Si se utiliza el entorno integrado de programación es recomendable eliminar los programas residentes no utilizados para liberar la máxima cantidad de memoria convencional posible.

Las aventuras compiladas precisan una cantidad de memoria que varía de acuerdo al tamaño de la base de datos. El compilador informa de la memoria que ocupa cada sección una vez compilada. Todas las secciones se cargarán en memoria en el momento de ejecutar el programa excepto las tablas de mensajes que serán cargadas en el momento en que sean requeridas. Para más información consultar la sección 2.5, página 3.

Esta versión funciona solamente con tarjetas gráficas VGA o compatibles.

**INTRODUCCION**

A continuación se expondrán brevemente una serie de conceptos básicos sobre el sistema SINTAC que habrá que tener en cuenta a la hora de programar.

**La base de datos:**

La base de datos es un fichero de texto que contiene una serie de secciones, cada una de las cuales definen distintos apartados del programa, como son las descripciones de las localidades, los mensajes, los objetos y el programa que controla todo. Hay una serie de comandos, denominados condactos, que permiten la programación de complejas aventuras conversacionales. Una explicación más detallada de la base de datos puede encontrarse en el apartado correspondiente.

El fichero de texto que compone la base de datos será procesado por el compilador, el cual lo compactará y codificará creando un fichero de datos para el intérprete.

Este fichero de datos podrá ser ejecutado por el intérprete‑debugger (DS.EXE), nunca por sí mismo ya que se trata de un fichero de datos solamente, aunque por medio del linkador podemos transformarlo en un fichero ejecutable.

Para crear el fichero de texto hace falta disponer de un editor ASCII o de un procesador de textos, o usar el editor incorporado en el entorno de programación el cual incorpora capacidades especiales que facilitan la edición de las bases de datos. Si se emplea un procesador de textos habrá que tener especial cuidado al salvar el fichero. Hay que usar el modo NO DOCUMENTO o EXPORTAR el texto en formato Texto del DOS dependiendo del procesador de textos que estemos usando. Si no se hace así el procesador de textos puede incluir códigos de control y formato en el fichero que confundirían al compilador.

Una elección adecuada, para los que dispongan del sistema operativo MS‑DOS 5.00 o superior es emplear el editor (EDIT) que incorpora el propio sistema.

**Vocabulario:**

El vocabulario está compuesto por una serie de palabras a las cuales se les asigna un número y un tipo. El tipo determinará si la palabra es un nombre, un verbo, un adjetivo o una conjunción. Los verbos normalmente se usarán para indicar acciones. Los nombres serán asignados a objetos, personajes o lugares. Los adjetivos permiten que dos objetos (o personas o lugares) cuyos nombres coincidan puedan ser diferenciados. Las conjunciones actúan como separadores dentro de la frase tecleada por el jugador.

Normalmente dos palabras del mismo tipo cuyo número coincida serán consideradas como sinónimos. Esto permite dar variedad al vocabulario y que el jugador no tenga que encontrar la palabra exacta a la hora de realizar una acción.

Cuando se inserta una palabra en el vocabulario sólo se consideran como significativos los 6 primeros caracteres. Esto quiere decir que palabras como MONTAR y MONTARSE son equivalentes. Ten muy en cuenta esto cuando crees el vocabulario.

Además hay que destacar tres tipos especiales de palabras en el vocabulario: las palabras de movimiento, los nombres convertibles y los nombres propios. Las palabras de movimiento son usadas para indicar las conexiones entre las distintas localidades. Palabras de movimiento son, por ejemplo, NORTE, SUBIR, SALIR. Pueden ser tanto verbos como nombres.

Los nombre convertibles se denominan así ya que si en la sentencia lógica no aparece un verbo, pero aparece uno de estos automáticamente se convierten en verbos. Los nombres propios son usados para nombres de lugares, de personajes. Estos nombres no son sustituidos cuando se usa un verbo con terminación (COGELO, COGELA, COGELE).

**Localidades:**

Una localidad es un lugar, una habitación o un territorio en el que el jugador de la aventura puede estar en un momento determinado. Una localidad se define dentro de la base de datos mediante un texto que es el que se imprimirá cuando se pida una descripción de esa localidad. Este texto debería ser, valga la redundancia, lo más descriptivo posible; debe dar al jugador una idea exacta de dónde se encuentra.

Un aspecto importante a tener en cuenta respecto a las localidades es que estas no tiene por que tener una extensión fija. Pueden ser tan pequeñas como el interior de una maleta o tan grandes como un valle entre montañas. Este aspecto debe tomarse en cuenta a la hora de diseñar la aventura.

Además en la base de datos también se incluirá una lista con las conexiones de la localidad con las que le rodean.

**Mensajes del Sistema:**

Los mensajes del sistema son empleados, como su nombre indica, por el propio intérprete para informar de ciertas condiciones cuando se ejecuta un condacto. Así, por ejemplo, el condacto GET informa con un mensaje del tipo 'Has cogido una manzana.' cuando, efectivamente, se pudo coger la manzana. Este es un mensaje del sistema. Todos los mensajes del sistema se pueden modificar libremente o insertar nuevos. Hay 255 mensajes del sistema disponibles, del 0 al 254, de los cuales algunos deben ser fijos ya que son usados por los condactos del SINTAC. A parte de estos el programador es libre de insertar los mensajes del sistema que más le convengan o modificar los que son fijos para que se adapten a su gusto. Los mensajes del sistema se imprimen por medio del condacto SYSMESS.

Hay un carácter especial común a todo tipo de mensajes, el símbolo de subrayado (\_). Si es encontrado este carácter será sustituido por la descripción del último objeto referenciado (cuyo número está en la variable del sistema 8). Esta característica es útil para acciones genéricas del tipo COGER. Así el mensaje del sistema que indica que se ha cogido un objeto es, en realidad, 'Has cogido \_.', el cual sirve para todos los objetos existentes en el juego.

**Mensajes:**

Aparte de los 255 mensajes del sistema, existen unas tablas de mensajes disponibles para el programador. Estos mensajes son generales y pueden ser usados para cualquier propósito. Se imprimen por medio de dos condactos: MES y MESSAGE.

Existen 255 tablas de mensajes que pueden albergar hasta 255 mensajes cada una proporcionando un total de más de 65000 mensajes.

Las mismas consideraciones, en cuanto al símbolo de subrayado, hechas para los mensajes del sistema, se consideran también para estos.

El uso de tablas de mensajes puede parecer un tanto complicado pero hay que considerar que facilitan la estructuración de la base de datos. Por ejemplo podríamos usar una tabla de mensajes para los mensajes de descripción de los objetos, otra para diálogos con PSIs, etc. Además mediante el uso de tablas de mensajes se puede "romper" la barrera impuesta por la memoria disponible ya que se usa un sistema de paginación que carga en memoria sólo la tabla de mensajes que está siendo usada. Esto también tiene un inconveniente y es que se debe acceder a disco cada vez que se necesita una nueva tabla de mensajes. Esta pequeña sobrecarga se puede minimizar, como ya se ha comentado, agrupando los mensajes según su categoría. Por otro lado el tiempo de acceso al disco es mínimo ya que se usa un sistema de índices que permite acceder directamente a la tabla de mensajes requerida.

**Objetos:**

Un elemento importante de cualquier aventura conversacional son los objetos. Los objetos dentro del sistema SINTAC están definidos por un número, además cada objeto lleva asociado un nombre y un adjetivo que deben ser palabras presentes en el vocabulario. Puede haber definidos hasta 255 objetos, del 0 al 254. El número de objeto 255 queda reservado como número de objeto no válido.

Aparte de esto un objeto también tiene definido un mensaje que es el que se imprimirá en vez del símbolo de subrayado o cuando se pida una lista de objetos.

Junto a todo esto un objeto viene definido también por una serie banderas, las cuales indican si es una fuente de luz, una prenda (banderas del sistema) o cualquier otra cosa gracias a la posibilidad de definir 16 banderas de usuario para cada objeto. Las banderas de un objeto se comprueban mediante los condactos HASAT y HASNAT. Las banderas de usuario permiten definir distintas propiedades de un objeto, si es pesado o ligero, si es grande o pequeño, si es frágil o robusto,... lo cual permite una gran flexibilidad a la hora de programar una aventura.

**Procesos:**

Los procesos son el corazón de la aventura ya que ellos son los que controlan el mundo creado por el programador. En los procesos quedan reflejadas todas las posibles acciones del jugador, así como las rutina que dan vida a los personajes que interaccionan con él.

Un proceso consta de una serie de entradas, cada una de las cuales a su vez consta de una serie de condactos. Los condactos se ejecutan en orden dentro de una entrada. Cuando uno de los condactos falla (una condición no se cumple o una acción no pudo llevarse a cabo) el flujo del programa se dirige a la siguiente entrada y así sucesivamente hasta que se produzca una salida del proceso en curso. Esta salida puede estar ocasionada por que se alcanzó el último condacto de la última entrada del proceso, o por la ejecución de un condacto que ocasione la salida del proceso en curso; como por ejemplo los condactos DONE y NOTDONE. Dentro de la descripción de cada condacto se explican los efectos de que una condición se cumpla o no, o de que una acción se ejecute o no.

El sistema, cuando se inicializa, empieza la ejecución en la primera entrada del Proceso 0, esto es importante ya que todas las acciones de inicialización deben ejecutarse en este proceso (imprimir pantalla de presentación, definir las ventanas, colocar ciertas variables del sistema a sus valores adecuados, etc...).

**VARIABLES y BANDERAS**

El sistema SINTAC posee dos tipos de datos con los que se puede trabajar, se trata de las variables y las banderas. Hay disponibles para el usuario 256 de cada una de ellas (256 variables y 256 banderas). Las variables y las banderas son totalmente independientes; cuando se modifica una variable no se afecta a ninguna bandera y viceversa. La diferencia está en que las variables pueden contener valores en un rango de 0 a 255 y las banderas solamente pueden contener 0 o 1.

Las operaciones que se pueden realizar con variables son bastante diversas: darles un valor, compararlas con otras variables, sumarles o restarles un valor, etc... En cambio, las operaciones que se pueden realizar con las banderas son limitadas (pero suficientes): ponerlas a 1 o a 0 y comprobar si están a 1 o a 0.

Hay ciertas variables y banderas denominadas del sistema, que son usadas por el intérprete internamente. A continuación se listan por orden numérico:

**Variables del sistema:**

0:número de ventana activa

1:número de localidad actual

2:número de verbo de la última sentencia analizada por PARSE

3:ídem pero nombre

4:ídem pero adjetivo

5:ídem pero nombre 2

6:ídem pero adjetivo 2

7:máximo número de objetos que se pueden llevar (0=ilimitados)

8:número de objeto actual

9:primer código ASCII IBM (usado por el condacto INKEY)

10:segundo código ASCII IBM

11:variable interna usada por QUIT, SAVE y END

12:código devuelto por el condacto INPUT

13:código de salida de un programa externo ejecutado con EXTERN

14:número de columnas de caracteres en pantalla (40 o 80)

15:número de filas en pantalla (25 o 30)

16:tiempo muerto en un condacto INPUT (en segundos)

17:número de tabla de mensajes cargada en memoria

18:fila del ratón

19:columna del ratón

20:reservada

**Banderas del sistema:**

0:indicador de oscuridad (1=si, 0=no)

1:si es 1 LISTAT imprime en formato encolumnado, si es 0 imprime los objetos seguidos

2:indicador usado para evitar las inicializaciones en el Proceso 0 (DESC lo pone a 0)

3:bandera de usuario

4:puesta a 1 si la tarjeta de video es EGA o VGA (siempre estará a 1 en esta versión del sistema)

5:estará a 0 si modo 640x480x16 colores o a 1 si modo 320x200x256 colores

6:modo de temporización de INPUT, 0 si el tiempo se da para teclear la frase completa, 1 si se da hasta teclear primer carácter

7:modo de funcionamiento de LISTAT y LISTOBJ (para más información consultar las secciones correspondientes a estos condactos)

8:1 si el botón izquierdo del ratón está pulsado, 0 si no

9:1 si el botón derecho del ratón está pulsado, 0 si no

10:1 ratón activado, 0 ratón desactivado

11 a 20:reservadas

**INDIRECCION**

El término indirección se refiere a la posibilidad de usar los valores almacenados en las variables como parámetros de los condactos. De esta forma se consigue que los parámetros indireccionados dependan del contenido de las variables que se usan para indireccionar.

Todos los parámetros de un condacto pueden ser indireccionados. Por ejemplo:

DESC 1, describirá siempre la localidad número 1, en cambio,

DESC [1], describirá la localidad cuyo número esté contenido en la variable número 1.

Para indicar que un parámetro es indireccionado se escribe entre corchetes []; así se indica que se usará el contenido de la variable cuyo número está entre corchetes, en vez del número directamente.

La indirección permite mucha flexibilidad a la hora de programar ya que mediante ella se pueden crear bucles. Por ejemplo:

\_ \_ LET 100 101

$repite

\_ \_ LET [100] 10

INC 100

EQ 100 106

DONE

\_ \_ SKIP $repite

Esta secuencia de condactos pone a 10 las variables 101 a 105 usando la indirección para crear un sencillo bucle. La línea clave es en la que aparece:

\_ \_ LET [100] 10

Mediante esta línea conseguimos que la variable cuyo número está en la variable 100 sea puesta a 10. Esto que puede sonar muy lioso es bastante fácil de asimilar. Asegúrate que entiendes lo que hace este pequeño ejemplo.

**ETIQUETAS Y SALTOS**

El sistema SINTAC posee la capacidad de realizar saltos desde un punto de un proceso a otro punto dentro del mismo proceso. Esto se consigue mediante el uso del condacto SKIP. El condacto SKIP debe ir seguido siempre de una etiqueta. Las etiquetas se colocan al inicio de una entrada, y solamente al inicio. No está permitido colocarlas en mitad de una entrada.

Una etiqueta se define por el símbolo $ seguido del nombre de la etiqueta. Aunque el nombre de la etiqueta puede ser tan largo como una línea de texto, sólo los primeros 14 caracteres son significativos. El nombre de la etiqueta puede contener todos los caracteres alfabéticos y numéricos así como los siguientes caracteres especiales: '+', '-', '\*' y '\_'

Así son válidas etiquetas como $saltar\_aquí, $ad+d, $coger-cosa, $123-456.

Las etiquetas son locales al proceso. Es decir, sólo existen dentro del proceso en el que se definieron. Así se pueden tener dos etiquetas de nombre igual en procesos diferentes sin que se interfieran entre sí; el condacto SKIP saltará a la etiqueta que se encuentre en su mismo proceso.

El máximo número de etiquetas que puede haber dentro de un proceso es de 100; además el máximo número de llamadas, dentro de un proceso, a etiquetas definidas posteriormente al condacto SKIP que las llama es de 100. Esto último es los que denominaremos saltos 'forward'

**VENTANAS**

Las ventanas permiten dividir la pantalla en zonas separadas e independientes unas de otras. Estas zonas de pantalla son rectangulares con unas dimensiones definidas mediante el condacto DEFWIN. Además cada ventana tiene unos colores por defecto que son también definidos por este condacto. El máximo número de ventanas que pueden ser definidas es de 10, numeradas de la 0 a la 9.

Todos los textos que se impriman se mandarán a la ventana activa la cual se selecciona por medio del condacto WINDOW. La ventana es activa hasta que se cambie por medio de otro condacto WINDOW o hasta que se modifique, por cualquier medio, el contenido de la variable del sistema 0.

Además cada ventana tiene unos colores independientes, así los condactos INK y PAPER sólo afectan a la ventana activa. Estos condacto seleccionan los colores actuales de impresión en la ventana activa. Los colores de la ventana activa sólo se modificarán si se cambian por medio de estos condactos, o si se ejecuta alguno de los condactos siguientes:

- CLWque restaurará los colores de la ventana a los colores que se dieron en el condacto DEFWIN

- DEFWINque colocará los colores de la ventana al color especificado en sus parámetros

Las ventanas pueden tener borde; inicialmente cuando son creadas por el condacto DEFWIN las ventanas no tienen borde. Mediante el condacto WBORDER se puede definir un tipo de borde para la ventana. Si se quiere definir una ventana con borde es aconsejable colocar el condacto WBORDER inmediatamente a continuación del DEFWIN que la define.

**MODOS GRAFICOS Y LOS COLORES**

Los ordenadores de la familia PC y compatibles permiten una amplia gama de colores si se dispone de la configuración adecuada para visualizarlos, es decir de una tarjeta gráfica de color VGA y de un monitor que permita visualizar la pantalla en color.

Bajo estos requisitos se podrán visualizar un máximo de 16 colores simultáneamente en el modo 640x480 (MODE 0) y hasta 256 colores simultáneamente en pantalla en el modo 320x200 (MODE 1).

Estos son los dos únicos modos que soporta la versión actual del sistema SINTAC.

El número real de colores disponibles es mayor, en concreto 262144, lo que ocurre que sólo un número muy reducido de ellos (16 o 256 según el modo) se pueden visualizar a la vez en pantalla.

El número del color (0 a 15 o 0 a 255) realmente es un índice a una paleta de colores, esta paleta tendrá capacidad para 16 o 256 colores y en ella estarán almacenadas las 'verdaderas' representaciones de los colores.

Un color se puede definir mediante sus componentes de color rojo, verde y azul o componentes RGB. En la tarjeta VGA estas componentes pueden tomar valores de 0 a 63 cada una. Así el color negro se representa por (0 0 0) y el blanco intenso por (63 63 63), serán pues estos valores los que estarán almacenados en la paleta de color.

En algunos programas de dibujo cada componente de color RGB se representa mediante un porcentaje del 0 al 100%. Es fácil establecer una relación entre porcentaje y valor real sabiendo que el 0% se corresponde a una componente de valor 0 y el 100% a una de valor 63. Mediante una regla de tres podemos a partir del porcentaje sacar el valor real de la componente tal y como lo entiende la tarjeta VGA. Así una componente del 20% en el programa de dibujo da un valor real de (63\*20)/100=12.6 que redondeando hacia abajo nos da 12.

¿Y a qué viene todo esto?. Pues que si tienes un programa de dibujo que te permite ajustar la paleta de colores (Deluxe Paint, Paint Brush, ...) puedes probar combinaciones de color hasta encontrar la que más te guste, anotar los porcentajes de cada una de sus componentes y luego usarlos en el condacto REMAPC, aplicando la conversión descrita, para modificar uno de los colores de la paleta.

Los primeros 16 colores disponibles inicialmente en la paleta se listan a continuación:

**SIN BRILLO CON BRILLO**

**Valor Color Valor Color**

0 Negro 8 Gris

1 Azul 9 Azul pálido

2 Verde 10 Verde pálido

3 Cían 11 Cían pálido

4 Rojo 12 Rojo pálido

5 Magenta 13 Magenta pálido

6 Marrón 14 Amarillo

7 Blanco 15 Blanco intenso

**PARTE II: Compilador, intérprete‑debugger y linkador‑**

**EL COMPILADOR**

**Cómo compilar la base de datos:**

***Los ficheros:***

El compilador permite codificar el fichero de texto que compone la base de datos y transformarlo en un formato compacto listo para ser ejecutado mediante el intérprete. Para compilar una base de datos se debe entrar al compilador con la siguiente línea desde el sistema operativo:

CS [base\_de\_datos] [fichero\_salida]

El parámetro *base\_de\_datos* es el nombre de la base de datos que queremos compilar. El parámetro *fichero\_salida* es el nombre del fichero que contendrá la base de datos compilada. Si hemos suministrado en la línea de llamada el nombre de la base de datos, como nombre de fichero de salida se cogerá este sustituyendo su extensión por DAT.

***Los mensajes de error:***

Si se produce algún error durante la compilación se mostrarán los errores producidos. Se presenta una lista de los errores junto al número de línea de la base de datos donde se produjeron (0 si no se aplica número de línea, por ejemplo cuando se produce un error de *apertura de fichero*).

El compilador reserva cierta cantidad de memoria para compilar las secciones. Si aparece el mensaje de error de *memoria insuficiente* habrá que liberar memoria eliminando programas residentes como controladores de ratón, emuladores de memoria expandida, discos ram,... e intentar compilar de nuevo.

***El proceso de compilación:***

Durante la compilación se irán presentando una serie de mensajes informativos sobre lo que está haciendo el compilador en ese momento. En concreto se mostrará una lista de secciones. Una marca indicará la sección que está siendo compilada. Cuando se termine se mostrará cierta información referente a la sección. El primer número indica la cantidad de palabras si es el vocabulario, el número de mensajes del sistema, tablas de mensajes, localidades, procesos, objetos. El segundo número indica lo que ocupa la sección en bytes una vez compilada, en el caso de tablas de mensajes indica el número total de mensajes compilados.

**Estructura de la base de datos:**

Como ya se comentó anteriormente la base de datos es un fichero de texto, el cual puede ser creado por cualquier editor o procesador de textos, o mejor desde el entorno de programación.

En la base de datos, las distintas secciones del programa están claramente delimitadas unas de otras; esto es así para que el compilador sepa de que sección se trata a la hora de compilar y codificar la base de datos.

Normalmente el inicio de una sección se indica mediante la secuencia de caracteres *\XXX*, donde *XXX* indica el tipo de sección. Los distintos tipos de secciones se listan a continuación:

\VOC : sección de vocabulario

\OBJ : sección de descripción de objetos

\LOC : sección de localidades

\MSG : sección de mensajes

\MSY : sección de mensajes del sistema

\PRO : sección de procesos

En la sección de mensajes habrá tantos \MSG como tablas de mensajes haya en la base de datos, así mismo en la sección de procesos habrá tantos \PRO como procesos haya en el programa.

El final de una sección también debe ser indicado; esto se realiza mediante la secuencia de caracteres \END. Estas secuencias de caracteres, tanto la de inicio como la de final, deben siempre estar colocadas en líneas separadas; así:

\LOC

..... aquí irían las descripciones de localidades .....

\END

**Comentarios en la base de datos:**

Los comentarios normalmente deben ir precedidos de un punto y coma (;). Se pueden colocar comentarios en prácticamente cualquier lugar de la base de datos teniendo en cuenta que cualquier línea que comience por punto y coma es considerada un comentario. Se debe tener cuidado con esto ya que cualquier línea de un texto de descripción de localidad, mensaje, mensaje del sistema o descripción de objeto, que contenga como primer carácter un punto y coma es considerada como comentario. Por ejemplo:

@001 Esta descripción de localidad no es correcta

; ya que esta línea es considerada como comentario

y por tanto la siguiente línea real de texto es esta

Además se pueden colocar comentarios tras la definición de cualquier elemento como palabras de vocabulario, conexiones, objetos, condactos; incluso detrás de las secuencias de inicio y final de sección:

\VOC

; sección de vocabulario

NORTE 5 Nombre ; verbo de movimiento

**Constantes simbólicas:**

Dentro de la base de datos se pueden definir constantes simbólicas. Una constante simbólica puede ser usada en cualquier lugar de la base de datos donde sería usado un número, con la comodidad de que es más fácil recordar un nombre que un número.

Las constantes simbólicas pueden estar definidas en cualquier lugar de la base de datos; tanto al principio, como al final, como en medio de cualquier sección.

El compilador realiza una primera pasada en busca de constantes y las va almacenando en una tabla. El máximo número de constantes es de 1000.

La definición de una constante debe ir en una línea que debe ser de la forma:

\\nombre\_constante valor

donde los caracteres '\\' indican el inicio de la definición de constante, *nombre\_constante* es el nombre que queremos asignar a la constante y que no debe tener más de 14 caracteres de longitud. El campo *valor* es un valor numérico de 0 a 255 que queremos asignar a la constante.

El nombre de la constante NO DEBE empezar con un número y puede contener letras, números y los caracteres '\_', '+', '-' y '\*'.

No es necesario definir una constante antes del lugar de uso de la misma; una constante que esté definida al final de la base de datos puede ser usada incluso al principio de esta.

A continuación unos ejemplos de uso de constantes simbólicas:

\\HABLAR\_PSIS 15

...

\PRO 3

DECIR \_ PROCESS HABLAR\_PSIS

DONE

...

\END

\PRO HABLAR\_PSIS

; constantes con los números de mensajes usados en

; conversaciones

\\MENS\_CAMARERO 18

\\MENS\_CRIADA 23

\\VAR\_CAMARERO 100 ; variable con localidad de

; camarero

\\VAR\_CRIADA 101 ; ídem para la criada

\_ CAMARERO EQ [VAR\_CAMARERO] [1]

MESSAGE 0 MENS\_CAMARERO

DONE

\_ CRIADA EQ [VAR\_CRIADA] [1]

MESSAGE 0 MENS\_CRIADA

DONE

...

\END

\\VAR\_INPUT 12 ; variable con el código

; devuelto por INPUT

\\TECLA\_F1 255 ; código de tecla F1

\\TECLA\_F2 254 ; código de tecla F2

\\INVENTARIO 24 ; proceso que imprime inventario

\\MIRAR 25 ; proceso que imprime

; descripción

\PRO 23

\_ \_ EQ VAR\_INPUT TECLA\_F1

PROCESS INVENTARIO

DONE

\_ \_ EQ VAR\_INPUT TECLA\_F2

PROCESS MIRAR

DONE

...

\END

Hay una serie de constantes simbólicas con un significado especial; estas son las siguientes:

- V\_MOVsu valor indica el máximo número de palabra de vocabulario que será considerado como verbo de movimiento. Su valor por defecto es 14.

- N\_CONVsu valor indica el máximo número de palabra de vocabulario que será considerado como nombre convertible. Su valor por defecto es 20.

- N\_PROPsu valor indica el máximo número de palabra de vocabulario que será considerado como nombre propio. Su valor por defecto es 50.

Si estas constantes no son definidas se tomará su valor por defecto y se producirá un mensaje de aviso.

**Descripción de las distintas secciones:**

Seguidamente se describen una a una las distintas secciones. Los campos encerrados entre corchetes ([]) son opcionales y pueden o no aparecer. El resto de los campos deben aparecer siempre, además en el orden en que se indican. Los campos deben ir separados unos de otros por uno o más espacios en blanco o por uno o más códigos de tabulación.

***Sección de Vocabulario:***

La sección de vocabulario se inicia con \VOC seguido de una serie de líneas de la forma:

palabra número tipo [;comentario]

Los distintos campos se comentan a continuación:

- *palabra*es la palabra que se quiere insertar en el vocabulario. Puede estar escrita tanto en mayúsculas como en minúsculas, el compilador la convertirá automáticamente a mayúsculas. Sólo son significativos los 6 primeros caracteres.

- *número*es el número que se quiere asignar a la palabra en el rango 1 a 254 inclusive.

- *tipo*indica el tipo de palabra de que se trata. El tipo de palabra puede ser:

V para verbo

N para nombre

A para adjetivo

C para conjunción

Puede estar en mayúscula o minúscula indiferentemente. Si en este campo se especifica más de un carácter, sólo se considera el primero. Así son equivalentes: Nombre, nombre, nomb, nada y n; verbo, V, v, verdadero y verb; etc...

Unos ejemplos de definición de la sección de vocabulario:

\VOC

NORTE 1 nombre

SUR 2 n

SUBIR 9 verbo

; acción de coger cosas

COGER 20 verbo ; coger

coge 20 VERB

CoJo 20 V

COGELO 20 verbosio

COGeLa 20 verbum

PREGUNtar 29 verbo

vieja 1 adj

Y 1 conjunción

luego 1 conj.

\END

Dentro del campo palabra sólo son admitidos como caracteres válidos letras y números (ABC..XYZ, 01..9) nunca otro tipo de caracteres. Además no se podrán insertar en el vocabulario palabras repetidas, es decir, cuyos 6 primeros caracteres del campo palabra coincidan.

A parte de esto hay que tener en cuenta que el número asignado a una palabra puede modificar el tipo de esta; así:

- Nombres y verbos cuyo número sea menor que el valor de la constante V\_MOV son considerados como palabras de movimiento.

- Los nombres cuyo número sea menor que el valor de la constante N\_CONV son considerados nombres convertibles.

- Los nombres cuyo número sea menor que el valor de la constante N\_PROP son considerados nombres propios.

Para más información ver la sección 8.4, página 13.

Dos palabras del mismo tipo cuyo número coincida son considerados como sinónimos; pueden ser usadas indistintamente una de otra.

Si el campo palabra de un verbo tiene menos de 6 letras se deben añadir las correspondientes entradas de ese mismo verbo con terminación LA, LO o LE. Por ejemplo:

DEJAR 21 verbo

DEJARL 21 verbo

DEJA 21 verbo

DEJALO 21 verbo

DEJALA 21 verbo

DEJALE 21 verbo

PONER 22 verbo

PONERL 22 verbo

PON 22 verbo

PONLO 22 verbo

PONLA 22 verbo

Como se puede observar si un verbo tiene al menos 5 letras en su campo palabra sólo hay que insertar una entrada más añadiendo una 'L' final. En cambio, si tiene 4 o menos letras en su campo palabra hay que insertar el verbo con su correspondiente terminación. En el ejemplo anterior se reconocerían las formas con terminación PONLO y PONLA del verbo PON, pero no la forma PONLE ya que esta no ha sido insertada.

***Secciones de Mensajes y Mensajes del Sistema:***

Estas dos secciones por su similitud se explican juntas. La sección de Mensajes se inicia con la secuencia de caracteres \MSG nnn, donde nnn es el número de tabla de mensajes, y la sección de Mensajes del Sistema con la secuencia \MSY. Seguidamente se definirán cada uno de los mensajes de la forma:

@nnn ..... Texto del mensaje .....

que puede ocupar varias líneas

seguidas. Al final se pondrá el carácter@

Se inicia el mensaje con el carácter @ seguido de un número en el rango 0 a 255. Los mensajes deben ir en orden numérico ascendente es decir primero el @1, luego el @2 y por último el @254. No es necesario que estén definidos los 255 mensajes, puede haber 'huecos', pero aquellos que se definan deben estar ordenados. Así podemos definir el mensaje 1 y luego el 10 sin definir el 2, el 3, ..., el 9; pero no podemos colocar primero el mensaje 10 y luego el 1, deben ir en orden numérico ascendente.

Seguidamente, separado por un espacio o por una tabulación, escribiremos el mensaje que puede ocupar varias líneas consecutivas. Hay que tener en cuenta que los retornos de carro no se insertarán como parte del mensaje, así el mensaje:

@1 Este mensaje se imprimirá

en una sola línea.@

Se imprimirá en pantalla:

Este mensaje se imprimirá en una sola línea.

Además las sucesivas líneas a partir de la primera se tomarán tal cual desde su primera columna, si hay espacios iniciales estos se respetarán, si no sencillamente no se insertarán. La primera línea es un tanto especial ya que el texto de esta se tomará a partir del carácter siguiente al espacio o tabulación que sigue al campo @nnn. Así si queremos un espacio inicial al principio del mensaje deberemos poner:

@1 Este mensaje tiene un espacio inicial.@

El cual se imprimirá con un espacio al inicio. Además hay un carácter con significado especial, el carácter (|) cuyo código ASCII es el 124. Este carácter será sustituido por un avance de línea con lo que el resto del mensaje se imprimirá en la siguiente línea de pantalla. Por ejemplo:

@1 Esto va en la primera línea|

y esto en la segunda|

esto en la tercera.@

Se imprimirá en pantalla:

Esto va en la primera línea

y esto en la segunda

esto en la tercera.

Se pueden insertar cuantos caracteres de avance de línea sean necesarios sin límite alguno. Recordar también que cualquier línea que comience por una serie de espacios o tabulaciones seguidos de un punto y coma (;) se considera un comentario. Así mismo las líneas en blanco se desechan.

En cuanto a las tablas de mensajes decir que deben estar colocadas en orden, no vale colocar, por ejemplo, la tabla 10 antes que la 2. Lo que si se acepta es que haya tablas sin definir. Por ejemplo se puede definir la tabla de mensajes 1 y luego la 5 sin haber definido las tablas de la 2 a la 4.

Seguidamente se incluyen unos cuantos ejemplos de secciones de mensajes:

\MSG 0 ; tabla de mensajes 0

@01 Mensaje con espacio inicial.@

@002 Mensaje con avance de línea|intermedio.@

@4 este mensaje no dará error aunque su número sea el 4 y no el 3@

@5 @ ; mensaje nulo

@6 Antes del número de mensaje puede haber cuantos espacios o tabulaciones se precisen.@

@007 ; esto no se considera comentario y se introducirá como parte del mensaje.

; en cambio esto es un comentario que no se incluirá

; ni esto tampoco

@

@8 Este mensaje en realidad ocupa solamente

dos líneas, ya que las dos anteriores son desechadas.@

\END

***Sección de Localidades:***

La sección de localidades se inicia con la secuencia \LOC seguida de la descripción de localidad de la forma:

@nnn ..... Texto de localidad .....

que igual que los textos de mensajes

puede ocupar varias líneas.@

Luego van las conexiones de la forma:

# palabra\_de\_movimiento nnn [;comentario]

# palabra\_de\_movimiento nnn [;comentario]

# palabra\_de\_movimiento nnn [;comentario]

...

El texto de la localidad se define de la misma forma que los mensajes (ver sección 8.5.2, página 16). Tras el texto de localidad se colocarán las conexiones. Cada conexión está compuesta de una palabra de movimiento seguida de uno o varios espacios, o una o varias tabulaciones, y después el número de localidad con la que conecta en el rango 0 a 251.

Cada conexión debe ir en una línea separada, comenzando con varios espacios o tabulaciones opcionales, seguidos del carácter (#). Se considera finalizada la lista de conexiones para esa localidad cuando se encuentra una línea que no sea ni comentario, ni esté en blanco y cuyo primer carácter no blanco (ni espacio ni tabulación) sea distinto del carácter (#).

La lista de conexiones es opcional y puede aparecer o no. A continuación se incluyen algunos ejemplos de definiciones de localidades:

\LOC

@0 Estás en un castillo medieval. Ves una herrumbrosa

armadura apoyada contra una pared.@

# NORTE 1 ; al patio de armas

# SUR 4 ; al salón del trono

# SUBIR 6 ; al torreón

@1 El patio de armas está vacío y desolado. Desde aquí ves la

puerta de entrada al castillo.@

# SUR 0 ; al interior del castillo

; Ahora van unas cuantas localidades sin usar

; que definiremos posteriormente

; la localidad 2 la reservamos para los establos

; la localidad 3 la reservamos para la armería

@4 El salón del trono está lleno de polvo y telarañas.@

# NORTE 0

@5 Estás en el fondo de un oscuro pozo. Parece que no hay

salida.@

@6 Desde el torreón divisas el desolado paisaje exterior.@

# BAJAR 0

\END

***Sección de Objetos:***

La sección de objetos se inicia con \OBJ seguida de la descripción de los objetos de la forma:

@nnn nombre adjetivo l.i [tipo] banderas\_usuario

texto\_de\_descripción\_del\_objeto

El campo *@nnn* indica el número del objeto en el rango 0 a 254 (el 255 es un número de objeto reservado). Seguidamente y separado por espacios o tabulaciones va el nombre del objeto el cual debe ser un nombre insertado en el vocabulario. Luego va el adjetivo el cual deber ser, también, un adjetivo válido del vocabulario o el símbolo del subrayado (\_), que indica que no hay adjetivo. Tras este va la localidad inicial, un número de localidad válido o los valores especiales para objeto no creado (252), objeto puesto (253) u objeto cogido (254).

Tras la localidad inicial va el tipo del objeto que viene indicado por una serie de caracteres separados por espacios o tabulaciones. Estos caracteres puede ir en mayúscula o minúscula indiferentemente; y son:

- Ppara indicar que el objeto es una prenda; es decir puede ser puesto y quitado (WEAR, REMOVE).

- Lpara indicar que el objeto es una fuente de luz; es decir que permite la descripción de localidades oscuras cuando está presente.

Si aparecen varios caracteres seguidos, sin separación, se tomará solamente el primero de ellos; así: *p*, *P*, *prenda*, *Poner* son equivalentes, también: *l*, *L*, *luz*, *luminosos* son equivalentes.

Luego van las banderas de usuario que son una secuencia de 16 caracteres seguidos, sin espacios intermedios. El carácter (x) o (X) se usa para indicar bandera activa, el carácter (o) u (O) se usa para indicar bandera inactiva. El orden de colocación de las banderas de usuario es de izquierda a derecha, con el primer carácter indicando el estado de la bandera 0 y el último el de la bandera 15.

En la línea siguiente a las definiciones del objeto va el texto de descripción, el cual será impreso en los listados de objetos así como en sustitución al símbolo de subrayado (\_) que aparece en los mensajes. El texto de descripción del objeto no puede ocupar más de una línea.

A continuación unos cuantos ejemplos de definición de objetos:

\OBJ

; Pesado

; Pequeño

; Elástico

; Inflamable

;

@0 LINTERNA VIEJA 1 L OOOOOOOOOOOOOOOO

una vieja linterna

@1 ESPADA \_ 2 ooooooooooooooox

una espada

@2 GUANTE ROJO 252 P ooooooooOOOOOOXO

un guante de lana rojo

@3 GUANTE VERDE 3 prenda OOOOooooOOOOxxxo

un guante verde, de nylon

\END

***Secciones de Procesos:***

Las secciones de procesos se inician mediante \PRO nnn, donde nnn es el número de proceso a definir. Habrá pues tantas secciones de procesos como procesos contenga el programa. Los procesos deben ir colocados en orden numérico ascendente y deben empezar a numerarse desde 0 en adelante, aunque como en el caso de los mensajes puede haber 'huecos' o procesos sin definir. El campo nnn debe ser un número en el rango 0 a 255. Tras el inicio del proceso irán un número variable de entradas de la forma:

verbo nombre condacto [;comentario]

condacto [;comentario]

...

verbo nombre condacto [;comentario]

condacto [;comentario]

...

... ...

Donde el campo verbo es un verbo o un nombre convertible del vocabulario o el símbolo de subrayado (\_), el campo nombre es un nombre válido del vocabulario o el símbolo de subrayado (\_) y el campo condacto se refiere a un condacto con sus parámetros. El primer carácter del campo verbo debe ser el primer carácter de la línea ya que si hay espacios o tabulaciones antes, la entrada se considera inválida. Luego van los condactos; tras el campo nombre siempre debe haber un condacto. Los condactos deben ir en líneas separadas y antecedidos por espacios o tabulaciones. Se considera el final de la entrada cuando se encuentra una línea (que no sea comentario) cuyo primer carácter no es ni un espacio ni una tabulación; es decir cuando se encuentra una etiqueta, el inicio de otra entrada o el final de la sección de proceso (\END).

Por ejemplo:

\PRO 0 ; se ejecuta tras cada inicialización y DESC

\_ \_ PROCESS 4 ; inicializaciones

\_ \_ NOTZERO 2 ; si hay que saltar

; acciones DESC

SKIP $buc\_princ

\_ \_ SET 2 ; aquí llega si es inicio

; o DESC

; Ejecuta diversas acciones tras una descripción.

\_ \_ NORESP

\_ \_ LISTOBJ ; lista objetos

PROCESS 1

$buc\_princ

\_ \_ SYSMESS 6

INPUT ; pide frase de entrada

SKIP $buc\_princ

\_ \_ NEWLINE

...

\END

\PRO 001

\END

\PRO 5

; este es el proceso donde se analizan las frase del jugador

\_ \_ PARSE ; analiza frase del

; jugador

NOTDONE

\_ \_ RESP

...

INVEN \_ SYSMESS 14 ; inventario

LISTAT 254

SYSMESS 15

LISTAT 253

DONE

COGER \_ AUTOG

DONE

...

\END

**EL INTERPRETE‑DEBUGGER‑**

**Conceptos básicos:**

El debugger (o mejor intérprete‑debugger) permite la ejecución paso a paso de la base de datos, presentando en cada momento el estado de las banderas y variables así como el condacto que está siendo ejecutado.

El debugger dispone de una serie de opciones para modificar banderas y variables, ejecutar paso a paso, etc... que serán explicadas en detalle más adelante.

**Ejecutar la base de datos compilada:**

***Cómo entrar en el intérprete‑debugger‑:***

La ejecución de la base de datos compilada se realiza entrando al intérprete‑debugger, desde el sistema operativo, mediante el comando:

DS nombre\_de\_fichero [opciones]

El *nombre\_de\_fichero* es el nombre del fichero que contiene la base de datos compilada. Si no se da el nombre del fichero se producirá un mensaje de error:

*Falta nombre de fichero*

Además se puede especificar la opción */lxx* o */Lxx*, donde *xx* es el número de línea de pantalla en la que queremos que se presente la ventana de depuración. Por ejemplo:

DS aventura /L5

***Mensajes de error:***

Si se produce algún error durante la apertura del fichero con la base de datos compilada (no existe, por ejemplo), aparecerá el mensaje:

*Error de apertura fichero de entrada*

Si el fichero existe pero no es un fichero conteniendo una base de datos correcta se dará el mensaje de error:

*Fichero de entrada no válido*

Si se produce algún error durante la lectura de la base de datos compilada aparecerá el error:

*Error en fichero de entrada*

El error:

*No hay suficiente memoria*

indica que la cantidad de memoria disponible es insuficiente para ejecutar la base de datos compilada.

Principalmente estos son todos los mensajes de error referentes al fichero que contiene la base de datos compilada.

**La ventana de depuración:**

La ventana de depuración es una pequeña franja de pantalla en la cual aparece la información del proceso de depuración. Puede ser colocada en cualquier posición de pantalla horizontalmente sin más que especificar el parámetro adecuado a la hora de entrar en el debugger (ver sección 9.2.1, página 22).

La información que presenta la ventana de depuración es la siguiente:

PRO nnn Var xxx=xxx \_\_\_\_\_\_ Band xxx=x

ddddd: VERBO NOMBRE

... condacto + parámetros ...

Donde *nnn* indica el número de proceso actual, luego van la variable y bandera que están siendo visualizadas actualmente junto con sus valores.

El campo *ddddd* indica la dirección del condacto en curso (esto es útil para calcular donde saltará un condacto SKIP). Seguidamente van el VERBO y el NOMBRE de la entrada actual si estamos en su primera línea, si no estos campos aparecerán en blanco. Después va el condacto que se va a ejecutar y sus parámetros, los cuales si van entre corchetes ([]) indicará que están indireccionados (ver sección 4, página 7).

Junto al valor de la variable aparecerá, si se trata de una variable relacionada con la sentencia lógica, la palabra de vocabulario correspondiente al contenido de la variable.

**Opciones del debugger:**

En principio las opciones disponibles son las siguientes:

Var. Band. Pant. Desact. Salir

- La opción V permite visualizar y editar variables.

- La opción B permite visualizar y editar banderas.

- La opción P quita momentáneamente la ventana del debugger de la pantalla, espera hasta que se pulse una tecla y luego la restaura. Esto sirve para visualizar los efectos de algún condacto de salida por pantalla.

- La opción D desactiva el paso a paso.

- La opción S sale al sistema operativo.

Para regresar al paso a paso se deberá pulsar F10 durante la ejecución de la aventura. Además otro método de regresar al paso a paso es colocando un condacto DEBUG 1 en algún lugar de la aventura.

***Visualizar y editar variables:***

Pulsando V desde las opciones principales pasaremos al modo de edición de variables, aparecerá la siguiente línea de menú:

Otra Modificar Fin

Se puede pasar a la siguiente variable pulsando el cursor arriba y a la anterior con el cursor abajo.

- La opción O permite seleccionar una variable, para lo cual aparecerá un cursor en el campo correspondiente y se podrá introducir el número de la nueva variable.

- La opción M permite modificar el valor de la variable. Igualmente aparecerá un cursor en el campo correspondiente al valor de la variable que nos permitirá introducir el nuevo valor.

- La opción F permite regresar a las opciones principales.

***Visualizar y editar banderas:***

Pulsando B desde las opciones principales aparece la siguiente línea reemplazando las opciones anteriores:

Otra Modificar Fin

Se puede pasar a la siguiente bandera pulsando el cursor arriba y a la anterior con el cursor abajo.

- La opción O permite seleccionar una bandera, para lo cual aparecerá un cursor en el campo correspondiente y se podrá introducir el número de la nueva bandera.

- La opción M permite modificar el valor de la bandera. Igualmente aparecerá un cursor en el campo correspondiente al valor de la bandera que nos permitirá introducir el nuevo valor (0 para desactivarla y 1 para activarla).

- La opción F permite regresar a las opciones principales.

**Frase y sentencia lógica:**

A lo largo de este manual consideraremos que frase es la línea que teclea el jugador cuando esta es solicitada por el condacto INPUT, en cambio sentencia lógica es la conversión de la frase en códigos numéricos por medio del analizador (condacto PARSE). Por ejemplo, si el jugador tecleó la frase:

COGER MARTILLO Y CLAVAR EL CLAVO.

El condacto PARSE extraerá de ella la sentencia lógica, así el primer condacto PARSE que se ejecute después de teclear la frase extraerá la siguiente sentencia lógica:

**Verbo Nombre Adjetivo Nombre2 Adjetivo2**

COGER MARTILLO ninguno ninguno ninguno

El siguiente condacto PARSE que se ejecute extraerá la sentencia lógica:

**Verbo Nombre Adjetivo Nombre2 Adjetivo2**

CLAVAR CLAVO ninguno ninguno ninguno

**La sentencia lógica:**

***Acciones encadenadas:***

Una frase puede contener varias sentencias lógicas, esto permite la introducción de acciones encadenadas; por ejemplo:

COGER PIEDRA GRANDE, LANZARLA AL GUARDIA, IR AL OESTE Y COGER MARTILLO

Esta frase será descompuesta por sucesivos condactos PARSE en las siguientes sentencias lógicas:

**Verbo Nombre Adjetivo Nombre2 Adjetivo2**

**1.** COGER PIEDRA GRANDE ninguno ninguno

**2.** LANZAR PIEDRA GRANDE GUARDIA ninguno

**3.** OESTE OESTE ninguno ninguno ninguno

**4.** COGER MARTILLO ninguno ninguno ninguno

Merece la pena observar las sentencias lógicas 2 y 3. La sentencia lógica 2 hace uso de una potente característica del analizador; el uso de verbos con terminación. Esto permite que el verbo LANZARLA se refiera al último nombre y adjetivo referidos, en este caso PIEDRA GRANDE.

La sentencia lógica 3, puede observarse que tiene repetida la misma palabra en el verbo y en el nombre. Esto ocurrirá si IR no está definida en el vocabulario y si OESTE es un nombre convertible. ¿Por qué sucede esto?. El analizador, cuando encuentra un nombre convertible en la frase y no encontró antes un verbo u otro nombre convertible, duplica el nombre encontrado en el verbo y en el nombre de la sentencia lógica. Esto permite que se más tarde se encuentra una entrada del tipo:

\_ OESTE ...

DONE

o del tipo:

OESTE \_ ...

DONE

Se ejecutará con esta sentencia lógica siempre que la comprobación verbo-nombre al inicio de cada entrada este activada (condacto RESP); si no estuviese activada esta comprobación las entradas descritas arriba se ejecutaría siempre, independientemente de la sentencia lógica actual.

***Variables del sistema asociadas a la sentencia lógica:***

La sentencia lógica se almacena en una serie de variables del sistema que se indican a continuación:

2:número de verbo de la última sentencia analizada por PARSE

3:ídem pero nombre

4:ídem pero adjetivo

5:ídem pero nombre 2

6:ídem pero adjetivo 2

Los números de verbo, nombre y adjetivo son aquellos que se especificaron cuando la palabra se introdujo en el vocabulario. Los números contenidos en estas variables son los que se emplearán para realizar las comprobaciones de verbo‑nombre para saber si una entrada se ha de ejecutar.

***Introducción de la sentencia lógica:***

La forma tradicional de introducir la sentencia lógica es a través de la frase tecleada por el jugador mediante el condacto INPUT, la cual luego es descompuesta por medio del condacto PARSE. Esto no quiere decir que sea la única forma posible de introducir la sentencia lógica.

Es posible crear una aventura dirigida por menús de opciones sustituyendo el condacto INPUT por un sistema de introducción a base de opciones, quizá usando el condacto INKEY. Usando este sistema alternativo habrá que colocar directamente, en las variables correspondientes, la sentencia lógica adecuada para que se ejecuten las entradas correctas. Así si se tecleó la opción de COGER objeto, habría que colocar en la variable 2 el número del verbo COGER, en la variable 3 el número del nombre del objeto y en la variable 4 el número del adjetivo del objeto; las variables 5 y 6 sería conveniente que estuviesen a 255 (ninguna palabra).

Con un poco de imaginación y con los condactos disponibles del SINTAC se pueden conseguir resultados excepcionales.

***Diálogo con personajes:***

El condacto PARSE permite, además de analizar la frase del jugador, extraer de la misma diálogos con personajes. Normalmente un diálogo con personajes es una frase dentro de la frase tecleada por el jugador, la cual queremos que, en presencia de cierto personaje provoque alguna acción en este. Así por ejemplo:

DECIR A JUAN "COGE LA MANGUERA"

Queremos que cuando le digamos esto al personaje Juan, este coja la manguera si es posible (o si quiere).

El condacto PARSE analizaría la anterior frase hasta el primer separador, en este caso serían las comillas dobles ("), dando la sentencia lógica:

**Verbo Nombre Adjetivo Nombre2 Adjetivo2**

DECIR JUAN ninguno ninguno ninguno

Esta sentencia lógica la usaríamos para ejecutar el proceso con las posibles respuestas de Juan, de la forma:

DECIR JUAN PROCESS 8 ; proceso de diálogos de

; Juan

DONE

El proceso con las posibles respuestas de Juan debería contener un condacto PARSE que analice el texto entre comillas y produzca una respuesta adecuada, así:

\PRO 8

\_ \_ PARSE

MESSAGE 0 '¿Qué dices?'

DONE

COGE MANGUERA ... ; condactos para coger manguera

MESSAGE 0 'Juan: Vale.'

DONE

... ; aquí otras posibles frases que deba entender Juan

\_ \_ MESSAGE 0 '¿Qué dices?'

DONE

\END

Esta es una posibilidad, desde luego no es la única. Además el condacto PARSE no distingue si el texto para Juan va entre comillas o no, sólo busca un separador (ver condacto PARSE). Por lo tanto las siguiente frases son equivalentes entre sí:

DECIR A JUAN "COGE LA MANGUERA"

DECIR A JUAN 'COGE LA MANGUERA

DECIR A JUAN, COGE LA MANGUERA

DECIR A JUAN: COGE LA MANGUERA.

DECIR A JUAN; COGE LA MANGUERA

Hay muchas otras posibles combinaciones, lo cual da bastante flexibilidad al jugador para elegir la que más se ajuste a sus gustos.

**LINKADOR**

El linkador que se suministra con el sistema SINTAC permite convertir las bases de datos compiladas en ficheros ejecutables. Para ello se suministra un módulo 'runtime' (SINTAC.RUN) que contiene una versión del intérprete que puede ser unida a la base de datos para conseguir un fichero ejecutable independiente.

El linkador se ejecuta con:

LKS [base\_de\_datos\_compilada] [fichero\_exe]

El parámetro *base\_de\_datos\_compilada* es el nombre del fichero que contiene la base de datos ya compilada. El parámetro *fichero\_exe* es el nombre del fichero ejecutable que queremos generar.

Si hemos suministrado el la línea de llamada el nombre de la base de datos, como nombre de fichero ejecutable se cogerá este sustituyendo su extensión por EXE.

Una vez se finaliza el proceso de linkado, y si no se produjo ningún error, tendremos una versión ejecutable de nuestra base de datos.