Índice de contenido

Introducción

Qué es InPAWS

Inpaws es una herramienta escrita en C++, que a partir de un archivo de texto con una sintaxis determinada (explicada a lo largo de este documento), genera un archivo con extensión .tap que contiene una base de datos que se puede cargar en cualquier versión del Professional Adventure Writing System de Gilsoft (en adelante PAWS) del Spectrum.

Dicho de forma más resumida, permite crear aventuras para el Spectrum sin necesidad de utilizar el editor de PAWS.

Características de InPAWS

- Generación de Bases de Datos (aventuras) PAWS en fichero .tap a partir de un archivo fuente de texto en lenguaje Inpaws.
- Permite incorporar y usar los gráficos y los juegos de caracteres generados en el editor de PAWS.
- Permite definir y utilizar nombres para Localidades, Objetos, Mensajes y Flags.
- Constantes con nombre para utilizar en los condactos.
- Comentarios para mejorar la legibilidad del código.
- Soporte a caracteres internacionales o códigos ASCII no soportados por PAW: puedes usar eñes, tildes, y demás caracteres de tu idioma directamente en los literales de texto. Simplemente indícale a Inpaws por qué códigos debe sustituirlos al crear la aventura.
- Mejor algoritmo de compresión que PAWS original (no siempre).
- Definición de mensajes directamente en los condactos. Es decir, que puedes utilizar MESSAGE “Es un collar la mar de bonito” en vez de MESSAGE 34 o MESSAGE MesCollar (también permitidos).
- Estructura más lógica y ordenada: las conexiones se definen con cada localidad, así como las palabras, peso y localidad incial de cada objeto van junto a su declaración.
- Permite definir partes de las Respuestas, Procesos, Vocabulario y Mensajes según se vayan necesitando: no necesitan estar todos definidos en el mismo lugar.

- Extracción de aventuras de volcados .Z80 o .SNA directamente a código fuente Inpaws modificable y recompilable por Inpaws.

Carencias actuales de Inpaws

- La más importante, soporte para aventuras en 128K.
- Macros parametrizables para agilizar la implementación de tareas comunes
- ¡Muchas cosas que al autor no se le ocurrieron!

Ejecución y opciones de InPAWS

Si ejecutamos InPAWS sin especificar parámetros, nos muestra el siguiente mensaje que nos valdría para explicar sus opciones:

Sintaxis: inpaws <comando> <fichero de entrada> [opciones]

Comandos:
c: Compilar a fichero tap (nombre por defecto: <fichero de entrada>.tap)
cd: Compilar a fichero fuente .SCE de PAW-PC
cm: Compilar a fichero fuente .SCE de PAW-CPM (Amstrad CPC)
e: Extraer fuente de .SNA o .Z80 con una aventura de PAW 48k
cp: Sólo generar fichero preprocesado, sin compilar (DEPURACION)
eg: Extraer gráficos de .SNA o .Z80 con una aventura de PAW 48k
ec: Extraer juegos de caracteres de .SNA o .Z80 con una aventura de PAW 48k

Opciones:
-o <fichero de salida>: Nombre dado al fichero generado por inpaws
-s : Listado detallado de símbolos tras la compilación

Compilar [c]

A partir del fichero de entrada especificado, InPAWS generará una base de datos PAWS que es posible cargar en un emulador de Spectrum corriendo cualquier versión de PAWS. Para ello, una vez generado el .tap, deberemos ir al menún de PAWS y teclear la opción J (Load Base de Datos). La base de datos con nuestra aventura se cargará en el Spectrum, momento a partir del cual el tratamiento sería igual que con cualquier aventura creada en PAWS. Podremos probar la aventura o grabarla cuando la consideremos finalizada. El nombre del fichero .TAP resultante si no se especifica ninguno será el mismo que el de entrada con extensión .TAP.

Compilar a .SCE de PAW-PC [cd]

A partir del fichero de entrada especificado, InPAWS generará un fichero fuente compatible con la versión para PC de PAW (PCPAW). Este fichero fuente será necesario compilarlo con el compilador de PC (pawcomp) antes de ver la aventura. Si no se especifica nombre de archivo, se establecerá por defecto el mismo que el de entrada con extensión .SCE.

Compilar a .SCE de PAW-CPM(Amstrad) [cm]

A partir del fichero de entrada especificado, InPAWS generará un fichero fuente compatible con la versión para CPM de PAW. Este fichero fuente será necesario compilarlo con el compilador de CPM (pawcomp) antes de ver la aventura. Por defecto establece en la sección /CTL que el destino será la unidad de disco A. Si no se especifica nombre de archivo, se establecerá por defecto el mismo que el de entrada con extensión .SCE.

Extraer [e]

Nos permite extraer una aventura creada con cualquier versión de PAWS, almacenada en un volcado de memoria de 48k .Z80 o .SNA. El fichero resultante está en lenguaje InPAWS, que podremos modificar posteriormete y volver a compilar, y contiene toda la aventura incluyendo Gráficos y caracteres (sets, UDG y Shades). Como es lógico no incluye cualquier proceso o función añadido mediante la opción EXTERN. Es decir, los EXTERN estarán ahí, pero no harán nada a menos que vuelvas a cargarlos en la zona de memoria asignada para ellos una vez grabada la aventura. Si no se especifica el fichero de salida, vuelca el resultado a la salida estándar.

Generar Preprocesado [cp]

Este comando nos permite generar el fichero preprocesado a partir del cual se genera el .TAP, pero sin dar el último paso de generarlo. Este fichero contiene una representación intermedia de InPAWS donde no existen símbolos y la estructura es más compacta y parecida a lo que sería una aventura de PAWS. Se proporciona por motivos de depuración, o si quieres ver como quedará tu aventura finalmente en PAWS. Si no se especifica el fichero de salida, vuelca el resultado a la salida estándar.

Extraer gráficos [eg]

Extrae sólamente la parte de gráficos de un volcado .z80 o .sna con una aventura PAWS de 48k, para poder incorporarlos a tu aventura. Consulta el apartado de gráficos para ver cómo utilizarla.

Extraer caracteres [ec]

Extrae sólamente la parte de juegos de caracteres/udg/shades de un volcado .z80 o .sna con una aventura PAWS de 48k, para poder incorporarlos a tu aventura. Consulta el apartado de Caracteres para ver cómo utilizarla.

Opciones

Fichero de salida [-o]

Permite especificar el nombre del fichero resultante de los comandos anteriores.

Listado detallado de símbolos [-s]

Una vez hayamos creado una aventura y la estemos probando, nos encontraremos con el difícil problema de saber los números de localidad, objeto, flag y mensajes que ha asignado InPAWS, y que son necesarios para cargar las banderas durante el proceso de depuración en PAWS de Spectrum. Para solventar este problema, esta opción genera un listado de todos los símbolos definidos y su valor, al finalizar la compilación.

Introducción a la sintaxis de InPAWS

Estructura de una aventura InPAWS

InPAWS no obliga a mantener una estructura concreta en cuanto a la definición de los elementos de una aventura. Puedes ir definiendo las localidades, mensajes, objetos, etc a medida que los vayas necesitando o manteniendo la estructura que consideres más clara. Únicamente hay unos mínimos que PAWS de Gilsoft también te exige, y que son los siguientes:

- Al menos una localidad
- Al menos un objeto

- Al menos un mensaje de usuario
- Mensajes del sistema básicos para las respuestas por defecto

Si no se satisface alguna de las condiciones anteriores, InPAWS te lo notificará, impidiéndote compilar la aventura. Por lo demás gozas de bastante libertad para definir tu aventura de la forma que prefieras.

Sensibilidad a mayúsculas

InPAWS es insensible a las mayúsculas, tanto para palabras reservadas de la sintaxis como para los símbolos. Dicho de otro modo, para definir una localidad utilizas la palabra reservada LOCATION, pero también podrías utilizar Location, location o LocATiOn. Eso mismo aplica a todos los símbolos y palabras de vocabulario definidas en tu aventura.

Cadenas de texto

Lo dicho en este apartado aplica a las descripciones de las localidades, objetos, mensajes y mensajes del sistema. PAWS permite incorporar ciertos caracteres de control (los llamados ESCC) dentro de los mensajes para obtener diversos resultados: cambiar el color del texto, cambiar el color de fondo, cambiar el juego de caracteres. Inpaws reproduce este comportamiento mediante códigos de control entre llaves dentro de las cadenas entrecomilladas:

{<código>}:

Permite introducir el código ASCII representado por <código> dentro de la cadena de texto. Por ejemplo {65} se correspondería con la letra 'A'. La utilidad de este control es que puedes meter también códigos no representables (no visibles) que imprimen ciertas características al texto que viene a continuación. Algunos de ellos son:

- ***{0-5}***: Selecciona el set de caracteres a utilizar en el resto del mensaje. Ejemplo “A partir de aquí {1}Utilizo el set 1”
- ***{7}***: Nueva línea. Aunque para esto hay una forma más sencilla, consistente en el carácter '^'.
- ***{16}{<color>}***: Cambia el color del texto que viene a continuación. Ejemplo: “Este texto es {16}{4}VERDE”
- ***{17}{<color>}***: Cambia el color de fondo del texto que viene a continuación: Ejemplo: “Este texto tiene fondo {17}{6}AMARILLO”
- ***{18}{<0-1>}***: Efecto de parpadeo (flash) en el texto que viene a continuación.
- ***{19}{<0-1>}***: Efecto de brillo en el texto que viene a continuación.
- ***{20}{<0-1>}***: Efecto de inversión del texto a continuación: el color del texto pasa a ser el del fondo y viceversa.

^ *:*

Se utiliza para generar un retorno de carro o nueva linea dentro del texto. Ejemplo: “Linea 1^Linea 2”

\” *:*

Representa una comilla dentro del texto. Esto es para diferenciarla del final de la cadena de texto. Ejemplo: “El ogro dice: \”Dame aaaargooo\””

\^ *:*

Representa al carácter '^' dentro del texto. Es para que el compilador lo distinga de un carácter de nueva línea. Ejemplo: “2\^3 es igual a 8”

\{:

Representa al carácter de apertura de llaves dentro del texto. Se utiliza para difernciarlo del inicio de un código de control para código ASCII. Ejemplo: "\{Esto es un texto entre llaves}".

Algunas consideraciones sobre las aventuras creadas con Inpaws

Por lo que el autor de InPAWS ha podido comprobar, el formato de PAW es bastante universal no habiéndose observado diferencias entre una y otra versión de PAWS en cuanto a estructura de una base de datos. Esto asegura que no vas a encontrar problemas al cargar tu aventura en la versión de PAWS que prefieras.

Sin embargo sí que existen diferencias, algunas muy importantes, en la forma en cómo las diferentes versiones del PAWS de Gilsoft interpretan la misma aventura. La más evidente es la que hay entre las versiones en español e inglés (uso de las eñes, adjetivos, pronombres). Pero hay otras más sutiles que deberás tener en cuenta, no sólo por el idioma. Por ejemplo, a partir de la versión 16 el tratamiento de las conversaciones con otros personajes (comando PARSE) tiene ciertas modificaciones. Otra diferencia importante es que la versión española de PAW tiene un bug que impide que al imprimir un carácter _ en un mensaje lo sustituya por el nombre del objeto actual SIN el artículo, como hacen las versiones en inglés.

En definitiva, debes tener en cuenta todos estos detalles al crear tu aventura. Mi recomendación es que antes de empezar decidas para qué versión de PAWS estás creando la aventura y te ciñas a ella hasta el final.

Un efecto colateral de lo anterior es que al extraer aventuras de volcados y volver a recompilar, si no las ejecutas en la misma versión de PAWS para la que fueron creadas, pueden hacer cosas raras. Esto no es un error de Inpaws.

Sintaxis de InPAWS

Localidades

LOCATION [<nombre de localidad>] | [<numero de localidad>]
{
"Descripcion de localidad";
[CONNECTIONS { <dirección> TO <destino> [<dirección> TO <destino> ...] };
}

Nombre de la localidad

Al definir una localidad se la puede identificar por un nombre, un código o ambos:

- LOCATION Casa { ... } // Define una casa, InPAWS asignará un número automáticamente
- LOCATION 13 { ... } // Define la localidad 13, sin nombre
- LOCATION Casa 13 {...} // Define una casa, y además tendrá el código 13 en PAWS

El último caso permite referirte a esa localidad por su nombre pero obligamos a Inpaws a que asigne ese número de localidad al generar la aventura. Si no especificamos número de localidad, PAW asignará uno automáticamente.

Lo más importante a tener en cuenta en la forma de definir una localidad es que a partir de ese momento sólo podremos referirnos a ella por el símbolo o valor que le hayamos asignado. Por ejemplo, si a la primera localidad definida no le asignamos número, no podremos referirnos a ella en el resto del código como localidad 0, aún cuando estamos bastante seguros de que InPAWS le va a asignar ese código. Si definimos una localidad por nombre Y número, tendremos que referirnos a ella por el nombre en el resto del código.

Algunas consideraciones más en cuanto a localidades:

- Recomendable asignar a la localidad de inicio el número 0, independientemente de que le pongamos nombre o no. Con ello nos aseguramos de que es la primera que va a visitar el jugador cuando empiece la aventura.
- Recomendable definir un código de localidad para las que actúen como contenedores de objetos, así como definir el mismo código para el objeto contenedor. Esto debe ser así por la forma que tiene PAW de tratar objetos contedores, en los que el contenido se almacena en la localidad con el mismo código que el objeto.
- Recomendable definir un código de localidad para las localidades con gráficos, ya que el soporte que da InPAWS a los gráficos es muy simple y no permite asignar nombres a gráficos individuales.
- Los códigos de localidad necesitan guardar una correlación. Esto quiere decir que empezarán en la cero e irán asignándose uno detrás de otro. Si asignas un código demasiado alto a una localidad, puede ser que InPAWS no encuentre suficientes localidades de las que no tienen código asignado para llegar a ese número de forma correlativa. En tal caso se suspenderá la compilación y se avisará de esta situación mediante el mensaje de error: “no se pudo establecer una definición correlativa de los elementos”.

Descripción de la localidad

Es la descripción que muestra PAW cuando entras en esa localidad. Consulta el apartado de cadenas de texto para ver qué posibilidades tienes.

Conexiones

Conexiones que parten desde esta localidad a las demás. La dirección debe ser una palabra de vocabulario (verbo o nombre < 14) y el destino, el nombre o código de la localidad de destino. Se pueden indicar las conexiones necesarias simplemente indicando una a continuación de otra. Importante cerrar este apartado con } y punto y coma.

Esta claúsula se puede omitir por completo si no hay conexiones obvias desde esta localidad a las demás.

Algunos ejemplos de localidades

```
LOCATION Inicial 0
{
“{16}{2}EL CAOS REPTANTE{16}{7}^Una aventura por Pepe Pérez^”;
}

LOCATION Pradera
{
“Estás en una agradable pradera bañada por los rayos del sol del amanecer.”;
CONNECTIONS { N TO Riachuelo S TO 2 ENTRAR TO Casita };
}

LOCATION 2
{
“Estás a la entrada de una cueva.”;
CONNECTIONS { N TO Pradera};
}
```

Objetos

```
OBJECT [<nombre de objeto>] | [<número de objeto>]
{
“Descripcion del objeto”;
[INITIALLYAT <localidad> | CARRIED | WORN | NOTCREATED;]
[WORDS <nombre> <adjetivo>;]
[WEIGHT <peso>;]
[PROPERTY [CLOTHING] [,CONTAINER] ;]
}
```

Nombre del objeto

Al nombre del objeto aplican las mismas reglas que para las localidades. Se pueden definir por nombre, por código, o por ambos. En el caso de indicar un código, InPAWS fuerza a que ese sea el código generado en la aventura final. En el resto del código nos referiremos a este objeto por el código o nombre que le hayamos puesto.

Algunas consideraciones sobre los objetos:

- También aplicable la recomendación de dar un código a los objetos contenedores, que sea el mismo de la localidad a la que están asociados.
- Recomendable definir como objeto 0 (le pongamos nombre o no) aquel que vaya a representar una fuente de luz en el juego, debido a la forma de PAW de tratar la luz y la oscuridad.
- Los códigos de los objetos necesitan guardar una correlación. Esto quiere decir que empezarán en el cero e irán asignándose uno detrás de otro. Si asignas un código demasiado alto a un objeto, puede ser que InPAWS no encuentre suficientes objetos de los que no tienen código asignado para llegar a ese número de forma correlativa. En tal caso se suspenderá la compilación y se avisará de esta situación, mediante el mensaje de error: “no se pudo establecer una definición correlativa de los elementos”.

INITIALLYAT

Localidad inicial del objeto. Se indicará el nombre de la localidad, o bien el código, o bien alguna de las palabras reservadas CARRIED (objeto en posesión del jugador), WORN (objeto llevado puesto por el jugador) o bien NOTCREATED (objeto inexistente). Por defecto, si no se especifica esta cláusula, el objeto se iniciará como NOTCREATED.

WORDS

Palabras del vocabulario por la que nos referimos al objeto en la aventura. Se indicará el nombre y el adjetivo que describen al objeto, o bien un carácter _ cuando alguno de ellos no aplique. Por defecto, si no se especifica esta cláusula, el objeto se asociará con las palabras vacías nombre:_ y adjetivo:_.

WEIGHT

Peso del objeto. Un valor entre 0 y 63. Caso de no especificarse esta cláusula, InPAWS asignará un peso por defecto de 1 al objeto.

PROPERTY

Propiedades especiales del objeto. Aquí se indicará la lista de propiedades, si las hay, que pueda tener el objeto. Concretamente si es una prenda que se pueda poner o quitar (CLOTHING) o es un

contenedor que puede almacenar otros objetos (CONTAINER). Por defecto los objetos no tendrán ninguna de estas dos propiedades.

Algunos ejemplos de objetos

```
OBJECT linterna 0
{
"Una linterna encendida";
INITIALLYAT CARRIED;
WORDS LINTE ENCE;
}

OBJECT espada
{
"Una espada triunfadora";
INITIALLYAT Lago;
WORDS ESPAD _;
WEIGHT 4;
}

OBJECT baul 1
{
"Un baul";
INITIALLYAT 5;
WORDS BAUL _;
WEIGHT 15;
PROPERTY CONTAINER;
}

OBJECT mochila 2
{
"Una mochila verde";
INITIALLYAT WORN;
WORDS MOCHI VERDE;
WEIGHT 2;
PROPERTY CONTAINER, CLOTHING;
}

OBJECT 3
{
"Un objeto soso";
}
```

Vocabulario

```
VOCABULARY
{
<tipo> [<número>] : <sinónimo1> [, <sinónimo2>...];
...
[<tipo> [<número>] : <sinónimo1> [, <sinónimo2>...];]
}
```

Explicación de los conceptos

Tipo: un tipo de plabra de entre los siguientes: NOUN, VERB, ADJECTIVE, PREPOSITION, ADVERB, CONJUNCTION, PRONOUN.

Número: el código que queremos asignar a la palabra. Este dato es opcional. Si no se indica, InPAWS asignará uno automáticamente, que será a partir de 50 para nombres, a partir de 20 para verbos, y a partir de 2 para el resto de palabras.

Sinónimos: lista de palabras entrecomilladas, separadas por comas que representan a la misma palabra para el código y tipo definidos al principio.

Algunas consideraciones sobre el vocabulario:

- Se pueden definir una o más palabras en cada bloque de vocabulario.
- Se pueden definir varios bloques de vocabulario en diferentes sitios, a medida que se van necesitando. Por ejemplo puedes definir un bloque justo después de un objeto para las palabras que describen al objeto, o tras definir una localidad para nombrar los elementos manipulables de esa localidad. También puedes hacerlo de la forma tradicional y poner todo el vocabulario junto.
- Para palabras especiales de PAW (MOVIMIENTO, CONVERT y NOMBRE PROPIO), deberás asignar tú sus códigos. En ese sentido pon especial atención en no asignar el mismo código a palabras diferentes a lo largo de tu aventura, o estarías definiendo sinónimos sin darte cuenta. Mi recomendación es que este tipo de palabras las definas todas en un mismo bloque para evitar esta confusión.
- En el PAW español se permite introducir la Ñ durante el juego (symbol shift+d, o CTRL+d en la mayoría de emuladores). Este carácter estaba representado por la barra invertida, que por tanto también había que redefinir en el set de caracteres usado para darle forma de eñe. Inpaws permite definir palabras de vocabulario con Ñ, las cuales transformará en '\' a la hora de volcar al TAP con la aventura. Tenlo en cuenta si vas a utilizar vocabulario con esta letra.
- A diferencia de localidades, objetos y mensajes, los códigos de palabra no necesitan guardar correlación. Puedes asignar el código que quieras a las palabras, por ejemplo podrías empezar en la 254, aunque no haya ninguna con código 253.

Algunos ejemplos de vocabulario

```
VOCABULARY { NOUN: “mochi”, “bolsa”; VERB: “lanza”, “tira”, “arroj”; }
VOCABULARY
{
NOUN 11: “PROA”, “PR”; // Un nombre que representa una dirección
NOUN 12: “POPA”, “PO”; // Otro
NOUN 17: “LISTA”; // Un nombre que podemos utilizar como verbo
NOUN 40: “FRODO”; // Nombre propio
PREPOSITION: “EN”, “DENTRO”;
}
```

Mensajes de usuario y Mensajes del sistema

```
MESSAGES
{
<numero de mensaje> | <nombre de mensaje> : “Texto del mensaje”;
...
[<numero de mensaje> | <nombre de mensaje> : “Texto del mensaje”; ]
}

SYSMESSAGES
{
<numero de mensaje> | <nombre de mensaje> : “Texto del mensaje”;
...
[<numero de mensaje> | <nombre de mensaje> : “Texto del mensaje”; ]
}
```

Aspectos generales de los mensajes

Los mensajes de InPAWS se pueden definir por número o por nombre. A diferencia de objetos y localidades, no existe la posibilidad de definir ambos. Debes decidir si a lo largo del código los vas a identificar por un nombre, al que InPAWS asignará automáticamente un número al compilar, o bien forzar a que tenga un número concreto.

En cuanto al texto, sigue las mismas reglas de formato que se han explicado en el apartado de cadenas de texto.

Puedes definir uno o varios mensajes en cada bloque MESSAGES o SYSMESSAGES. Asimismo, puedes ir definiendo bloques de mensajes a medida que los vayas necesitando, no siendo necesario agruparlos todos en un mismo bloque.

PAW establece la obligación de definir al menos un mensaje de usuario y 54 mensajes del sistema que son utilizados por los comandos automáticos de PAWS (AUTOG, AUTOD, listado de objetos, etc). Los mensajes del sistema los puedes modificar a tu gusto, pero no podrás compilar una aventura que tenga menos de ese número.

Definición de mensajes en los condactos

InPAWS permite una forma de definir mensajes muy potente, que consiste en utilizar directamente el texto que queremos visualizar tras uno de los condactos que muestran mensajes (MESSAGE, MES y SYSMESS). En este caso, InPAWS crea el mensaje nuevo, le asigna un número y convierte, una vez compilada la aventura, el literal del mensaje en ese número.

Ejemplo (de la tabla de respuestas):

```
EXAMI ESPAD: PRESENT Espada MESSAGE “De acero duro y afilado.” DONE;
```

Por supuesto esta forma de definir mensajes está sujeta a todas las restricciones de la forma “tradicional”, es decir, contabiliza como un mensaje más en las tablas de mensajes y si llegamos al límite impuesto por PAW de 255, no nos dejará compilar la aventura.

InPAWS intenta hacer economía en esta forma de definir mensajes. Para ello, si se utiliza dos o más veces el mismo mensaje en los condactos, sólo se definirá el primero de ellos, el resto detectará la existencia de un mensaje con el mismo texto y simplemente hará referencia al mismo código de mensaje. Esto sólo es válido si los mensajes son EXACTAMENTE iguales. Baste que varíe un sólo carácter para que InPAWS genere un nuevo mensaje con lo que estamos escribiendo.

Ejemplo:

ESCUCHA BOSQUE: AT bosque MESSAGE “No oyes nada fuera de lo común.” DONE;

ESCUCHA _ : MESSAGE “No oyes nada fuera de lo común.” DONE;

Lo anterior sólo va a generar un mensaje al que ambas entradas de proceso harán referencia en la aventura generada.

Algunos ejemplos de mensajes

MESSAGE

{

0: “Este es un mensaje con código fijo.”;

Mimensaje: “A este mensaje hay que referirse por su nombre.”;

}

SYSMESSAGES

{

MsgArcon: “En el arcón hay: “;

57: “Este es el mensaje 57, en PAW y en InPAWS.”;

}

Colores y Set de caracteres por defecto

DEFAULTS

{

[CHARSET: <0-5>;]

[INK: <0-9>;]

[PAPER: <0-9>;]

[FLASH: <0-1>;]

[BRIGHT: <0-1>;]

[INVERSE: <0-1>;]

[OVER: <0-1>;]

[BORDER: <0-7>;]

}

Correspondería a la opción “B-Colores de Fondo” de PAW, y permite establecer los valores por defecto para tinta, fondo, brillo, flash, borde y juego de caracteres, que se utilizarán en la aventura. Estos son los valores que PAW asignará a cada localidad salvo que el programador los modifique con alguno de los condactos al efecto, o mediante códigos de control en los mensajes.

Todos ellos son opcionales (de hecho el bloque completo es opcional). Si no se especifican, adoptan los siguientes valores por defecto:

DEFAULTS

{

CHARSET: 0;

INK: 7;

PAPER: 0;

FLASH: 0;

BRIGHT: 0;

INVERSE: 0;

OVER: 0;

BORDER: 0;

}

Flags y constantes

FLAG <nombre del flag> [<numero del flag>];

CONSTANT <nombre de constante> <valor de constante>;

PAW trae por defecto 256 flags y permite asignar constantes numéricas en casi todos los condactos (LET, EQ, ISAT, ...). Para facilitar la tarea de identificar los valores asignados a estos flags así como la función que cumplen los flags en la aventura, InPAWS permite definir nombres para ambos.

Para definir un FLAG tenemos dos posibilidades. La primera es indicar simplemente un nombre en cuyo caso PAW asignará un número automáticamente, de los disponibles para el usuario (del 11 al 28, y del 60 en adelante). La otra forma de definir los flag es especificar qué número de flag queremos asignarle al nombre. En este caso, InPAWS permite apuntar a uno de los FLAGS "reservados", asumiendo que el programador se hace responsible del uso que le va a dar a ese flag. Esto permite que en el resto del código nos podamos referir a LocalidadActual en vez de al número 38 cuando queramos comprobar la localidad del jugador (por ejemplo).

Aquí es importante resaltar que cuando definimos un FLAG, lo único que estamos haciendo es darle un nombre, ya que los flags *están todos definidos por defecto*. Esto quiere decir que sin necesidad de definir ningún flag, podríamos utilizarlos en los condactos, simplemente refiriéndonos a ellos por su número.

En el caso de constantes es obligatorio asignarles un valor. Las constantes se pueden utilizar en los condactos en sustitución de los valores numéricos, por ejemplo:

```
FLAG dineroDisponible;
CONSTANT maxDinero 100;
_ _ : GT dineroDisponible maxDinero MESSAGE "¡Soy rico!"
etc
```

Algunos ejemplos de flags y constantes

FLAG locaJugador 38;

FLAG objetoActual 51;

FLAG MiNuevoFlag;

FLAG OtroFlag 100;

CONSTANT MiValor 150;

Procesos y Respuestas

```
RESPONSE
{
<verbo entrada> <nombre entrada>
  [| <verbo entrada> <nombre entrada>...]: <lista de condactos> ;
...
[ <verbo entrada> <nombre entrada>
  [| <verbo entrada> <nombre entrada>...]: <lista de condactos> ; ]
}

PROCESS <nombre de proceso> | <numero de proceso>; // Declaración de un proceso
PROCESS <nombre de proceso> | <numero de proceso>
{
<verbo entrada> <nombre entrada>
  [| <verbo entrada> <nombre entrada>...] : <lista de condactos> ;
...
[ <verbo entrada> <nombre entrada>
  [| <verbo entrada> <nombre entrada>...]: <lista de condactos> ; ]
}
```

Sintaxis de los procesos

Los procesos en Inpaws se identifican por un número, en cuyo caso será el número asignado en la aventura generada, o bien por un nombre. No es posible asignar ambos. En el resto del código deberemos referirnos a ese proceso por el identificativo asignado (nombre o número). En el caso de asignar un nombre al proceso, InPAWS le asignará de forma automática un número durante el proceso de compilación.

Verbo entrada: el verbo que se busca en la tabla de procesos o repuesta en función de lo teclado por el jugador. Puede adoptar los valores especiales _ y * para indicar “cualquier entrada”, al igual que en PAWS.

Nombre entrada: el primer nombre que se busca en la tabla de procesos o repuesta en función de lo teclado por el jugador. Puede adoptar los valores especiales _ y * para indicar “cualquier entrada”, al igual que en PAWS.

Lista de condactos: la lista de condactos que se ejecuta al encontrar la entradas verbo-nombre del encabezado.

Declaración de los procesos

Como caso excepcional dentro de la sintaxis de InPAWS, es obligatorio que los procesos creados por el usuario (a partir del 3) se declaren antes de su utilización. Caso de definir parte de un proceso que no ha sido declarado, fallará la compilación con un mensaje de error indicándolo.

Opciones múltiples, sinónimos y uso de operador O

Inpaws permite especificar opciones múltiples (también llamadas sinónimos) en las entradas de procesos, así como en los condactos dentro de esas entradas que sean condiciones. Para ello se debe utilizar el carácter de *pipe* o tubería '|' entre las entradas o condactos que queremos que se evalúen de una forma alternativa.

- Utilización de sinónimos dentro para los encabezados de las entradas de procesos/respuestas: se indicarán las tuplas verbo-nombre para las cuales queremos repetir el cuerpo de la entrada en la aventura generada, separando esas tuplas por el carácter |.
    - Ejemplo: USAR LLAVE|ABRIR PUERTA: <lista de condactos>;
- Utilización de opciones alternativas dentro del cuerpo de la entrada: separar las opciones alternativas (que sean condiciones) mediante el carácter |. Se generarán tantas entradas como opciones consideradas, teniendo en cuenta todas las combinaciones posibles.
    - Ejemplo: ABRIR PUERTA: CARRIED llave|ISAT llave abrigo SET abierta ...;

Se pueden combinar ambas formas de especificar opciones alternativas:

USAR LLAVE |ABRIR PUERTA: CARRIED llave|ISAT llave abrigo SET abierta ...;

así como especificar tantas opciones alternativas dentro del cuerpo de los condactos como sean necesarias. Aquí es necesario tener en cuenta que cuando InPAWS encuentra una o varias de estas opciones dentro de una entrada de procesos o respuesta, va a generar tantas entradas en la aventura generada para PAW como la multiplicación del número de opciones especificadas, de forma que se cubran todas las posibilidades con todas. En el caso del último ejemplo estaríamos generando en la aventura final: 2 opciones de encabezado X 2 opciones de cuerpo de entrada = 4 entradas en la tabla.

Otros aspectos importantes a tener en cuenta

- InPAWS tiene definidos por defecto los procesos 1 y 2, a los que no se podrá cambiar el nombre y por tanto debes referirte a ellos por sus números en tu código.
- Del resto de códigos asignados, InPAWS intentará establecer una definición correlativa de los procesos definidos, al igual que se hace en objetos, localidades y mensajes. Si esta no es posible, el proceso de compilación finalizará con el mensaje de error: “no se pudo establecer una definición correlativa de los elementos”.
- No es necesario definir todo el proceso/respuesta en el mismo sitio, puedes ir ampliando el proceso/respuesta a medida que lo vayas necesitando. Por ejemplo, puedes definir las respuestas que afectan a una localidad u objeto justo después de su definición. En definitiva, se pueden abrir varios bloques referidos al mismo proceso/respuesta en distintas partes del código. El compilador agrupará todas ellas durante el proceso de generación de la aventura.

Orden de las entradas en una tabla de procesos o respuestas

Al procesar una tabla de respuestas o procesos, PAWS recorre las tablas de principio a fin buscando una entrada hasta que una de ellas ejecuta el condacto “DONE”. El orden en el que están almacenadas las entradas es importante, y debe conocerlo el programador, ya que dependiendo de la entrada que se procese primero, la tabla se comportará de una forma u otra.

PAWS ordena las entradas dentro de un proceso o tabla de respuesta por código de verbo primero, y por código de nombre después. Las palabras reservadas * y _ representan los códigos 1 y 255 respectivamente. Es importante resaltar aquí que se trata de una ordenación POR CÓDIGO, no por alfabeto.

Para igual verbo y nombre, el orden que establece PAWS es el de introducción de esas entradas, es decir, si tenemos varias entradas EXAMI ESPAD, la primera que aparecerá en la tabla de respuestas, y que por tanto se comprobará a la hora de buscar una respuesta adecuada a la entrada del jugador, es la que primero se haya introducido, y así sucesivamente.

El comportamiento de InPAWS en relación a lo anterior respeta la filosofía de PAW, pero conviene tener en cuenta las diferencias entre una definición en fichero de texto y lo que muestra el editor de PAW de Gilsoft para evitar confusiones:

- El fichero de texto que contiene tu aventura InPAWS tendrá las entradas en el orden en que tú las hayas ido escribiendo, sin importar el código de palabra, pero ten en cuenta que al generar la aventura se seguirá respetando la filosofía de PAW de ordenar por código, y por tanto ese orden podría cambiar.
- Para entradas de InPAWS de igual verbo y nombre, aparecerá primero en la aventura aquella que esté primero en tu fichero de texto con el fuente.
- Al definir las palabras del vocabulario sin asignarles número, no podrás hacer presunciones de cuál de ellas tendrá un código menor (recuerda que el orden no es alfabético). Por tanto si quieres controlar totalmente el orden en que se recorrerán las tablas de procesos/respuestas, deberás asignar número a las palabras de vocabulario que definas y para las que quieras asegurar un orden concreto.

Parámetros en los condactos

Los parámetros utilizados en los condactos deberán ser coherentes con el tipo de dato que tengan definido. Consulta la especificación de PAW para conocerlos (en WOS tienes toda la documentación). Atendiendo al tipo de parámetro, basándonos en la nomenclatura utilizada por Gilsoft, deberemos proporcionar lo siguiente:

LOCNO / LOCNO+: el código o nombre de una localidad, tal como la hayamos definido. En el caso de LOCNO+, podremos utilizar las localidades "especiales": NOTCREATED (252), CARRIED(254) y WORN(253), así como la 255 que siempre representa la localización actual del jugador.

MESNO / SYSNO: el código o nombre del mensaje tal y como los hayamos definido en un bloque MESSAGES o SYSMESSAGES, o bien un literal de cadena que representa al mensaje a mostrar.

FLAGNO: el código o nombre del flag. Recuerda que para utilizar los flags por código no es necesario haberlos definido.

PROCNO: el código o nombre del proceso, tal y como lo hemos declarado. Los procesos 1 y 2 están definidos por defecto y no se les puede cambiar el nombre.

WORD: una palabra definida en el vocabulario, o las palabras de vocabulario reservadas _ y *.

VALOR: en la documentación de PAW viene indicado como un rango de valores: 0-255, 0-7, etc. En el caso de uno de estos valores, podemos indicarle directamente el valor numérico que vamos a utilizar, pero también podemos utilizar palabras de vocabulario, nombres de localidad, nombres de objeto, o constantes.

Gráficos

InPAWS es una herramienta para generar bases de datos PAWS a partir de ficheros de texto, y como tal no puede crear gráficos para tus aventuras. La solución adoptada en InPAWS para poder incorporar gráficos es la de crearlos en PAW para luego importarlos a tu fichero fuente en un formato de listado de datos, que luego InPAWS codificará dentro del fichero resultante al finalizar

la compilación.

Para incorporar gráficos a tus aventuras podrás seguir dos enfoques diferentes: crear toda la aventura y añadir los gráficos al final (Enfoque 1),o bien ir añadiendo gráficos a medida que vas creando la aventura y las localidades que van a mostrar gráfico (Enfoque 2). Explico la forma de hacerlo en cada caso:

Enfoque 1 (más estricto)

1) Escribe tu aventura en Inpaws con todas las localidades, objetos, mensajes, etc. Pruébala y decide cuando está completa para incorporarle los gráficos.
2) Compila la aventura con InPAWS, utilizando la opción de compilación -s para que puedas obtener un listado de los códigos asignados a las localidades para las que hayas definido un nombre.
3) Carga tu aventura en el editor de PAWS del Spectrum y añade los gráficos a las localidades.
4) Una vez compruebes que todo está correcto, graba la aventura (opción A-Save Aventura) a un tap, resetea el spectrum en modo 48k cárgala como lo harías si fueras a jugarla. Una vez ejecutándose en el Spectrum graba un snapshot con el nombre: graficos.z80.
5) Vuelve a ejecutar InPAWS, pero esta vez utiliza la siguiente sintaxis: inpaws eg graficos.z80 -o graficos.txt.
6) Abre el graficos.txt, contiene un bloque GRAPHICS que deberás sustituir en el fichero fuente de tu aventura borrando cualquier bloque graphics anterior.
7) Y ya está. Cada vez que vuelvas a compilar tu aventura, aparecerá con los gráficos que has definido. A partir de este momento, puedes modificar tu aventura añadiendo objetos, cambiando mensajes, metiendo partes nuevas, cualquier cambio EXCEPTO modificar la estructura de las localidades en cuanto a cantidad y orden en el que las has ido definiendo. Si modificas esto puedes generar una inconsistencia entre los números de localidad y los gráficos asociados. Puedes no obstante modificar las descripciones y conexiones sin temor.

Enfoque 2 (más flexible)

Este es el modo que yo recomiendo, ya que no te obliga a definir la aventura por completo antes de añadirle gráficos. Deberás seguir los siguientes pasos:

1) Empieza a escribir tu aventura, de la forma habitual, pero asigna un número a TODAS las localidades. InPAWS tiene la posibilidad de asignar tanto nombre como número a las localidades, úsala. Únicamente deberás llevar un control de qué códigos has ido asignando para ver cual sería el siguiente. Lo más importante aquí es: no cambies esos códigos a lo largo del proceso. Una vez hayas asignado un código a una localidad, cíñete a él hasta el final.
2) Cuando hayas definido una localidad que pueda tener un gráfico (o subrutina), o simplemente en el momento que decidas incorporar algunos gráficos, aunque no esté toda la aventura finalizada, compílalo a un TAP y cargalo en el editor de PAWS.
3) Crea tus gráficos, graba la aventura, genera un snapshot e importa los gráficos de la misma forma que se explicó en el enfoque 1.
4) Sigue desarrollando tu aventura (que ahora tendrá los gráficos que creaste), y si de nuevo deseas incorporar gráficos para las nuevas localidades que hayas definido, vuelve a repetir el proceso desde el paso 2.
5) Se acabó. Este enfoque es más flexible pero te obliga a asignarle tú los códigos a todas las

localidades (no sólo a las que tienen gráficos, ya que las demás podrían ser subrutinas gráficas).

Juegos de caracteres

Al igual que ocurre con los gráficos, la definición de los UDGs, Shades y juegos de caracteres deberás hacerla en PAWS para luego importarlos dentro de tu aventura en un bloque CHARACTERS. En este caso, los juegos de caracteres no están asociados a ningún otro elemento del juego, por lo que podrás realizar este proceso cuando consideres más conveniente, todas las veces que te haga falta, sin temor a generar ninguna inconsistencia en el resultado. Los pasos para incorporar datos de juegos de caracteres, udg y shades son los siguientes:

1) Abre el PAWS de Spectrum (no necesitas cargarle ninguna aventura) y define los juegos de caracteres, udg y shades que consideres necesarios.
2) Graba la aventura resultante (opción A-Save Aventura). Resetea el emulador a modo 48k y carga la aventura como si fueras a jugarla.
3) Graba un snapshot al fichero: carac.z80.
4) Ejecuta InPAWS de la siguiente manera: inpaws ec carac.z80 -o carac.txt
5) Abre tu aventura y el carac.txt y sustituye todo el bloque CHARACTERS de tu aventura por el que se encuentra en el fichero carac.txt.
6) Fin. Si necesitas hacer modificaciones en los juegos de caracteres, compila tu aventura (que ahora contendrá los nuevos juegos de caracteres), cárgala en el editor de PAWS, haz las modificaciones pertinentes y vuelve al paso 2.

Directivas del preprocesador

Para los no iniciados en programación, una directiva es un comando no perteneciente a la sintaxis de un lenguaje, que instruye o canaliza al compilador en la forma en que debe procesar el código fuente de la sintaxis. Inpaws provee un soporte básico de directivas con dos objetivos básicos:

- Poder dividir el código fuente en varios ficheros con objeto de clarificar el código.
- Decidir, en función del hardware para el que vayamos a generar la aventura, si procesar o no ciertas partes del código.

Todas las directivas del preprocesador de Inpaws comienzan por el carácter #.

Inclusión de ficheros

Con objeto de evitar manejar ficheros muy grandes con nuestros fuentes, es posible separar la aventura en varios ficheros (por ejemplo uno por cada zona del juego) y juntarlos todos al compilar la aventura. Para ello utilizaremos la directiva #INCLUDE:

#INCLUDE “<nombre de fichero a incluir>”

Una vez procesada esta directiva, el compilador seguirá leyendo del nuevo fichero hasta que finalice, momento en el que continuará la compilación donde lo dejó antes del #INCLUDE.

No hay inconveniente en incluir nuevos ficheros dentro de los ficheros incluidos (includes dentro de includes), hasta un nivel indeterminado. No obstante tendrás que prestar atención a no hacer includes “anidados”, aunque esta situación te será avisada por el compilador.

No se pueden incluir ficheros en directorios diferentes al del fichero principal (el que damos como parámetro al invocar el comando), por lo que cualquier nombre de fichero con ruta establecida será rechazado por el preprocesador.

Compilación condicional

El objetivo de la compilación condicional es forzar o ignorar la compilación de ciertas partes del fichero fuente dependiendo de la existencia de ciertas "banderas" o variables del preprocesador. En el caso de Inpaws, el soporte a compilación condicional viene dado por la posibilidad de generar aventuras para varias plataformas (Spectrum, PC o Amstrad CPC). En cuyo caso el programador tendrá que tener en cuenta las particularidades de cada versión decidiendo en cada momento qué partes deben compilarse y cuales no en función de lo anterior. No obstante no sólo está restringido a estos casos, y el programador podrá usarlo para los fines que considere.

#DEFINE <variable>

La variable nombrada será definida y tenida en cuenta como existente en el resto de la compilación. Las variables de preprocesador en Inpaws no tienen valor asignados, su definición cumple simplemente el objetivo de declararlas como existentes, para su posterior comprobación en directivas #IFDEF e #IFNDEF.

Inpaws tiene predefinidas ciertas variables en función del comando de compilación que se haya indicado. Para la opción de compilación "c" (Spectrum) existirá una variable PAWSPECTRUM. En el caso de la compilación para PC (opción "cd") existirá una variable predefinida PAWPC. Finalmente si estamos compilando para CPC, la variable predefinida por el preprocesador será PAWCPM.

#IFDEF <variable>

... Código ...

#ENDIF

El trozo de código entre las directivas #IFDEF y #ENDIF sólo será procesado si está declarada la variable de preprocesador especificada tras la directiva #IFDEF.

#IFNDEF <variable>

... Código ...

#ENDIF

El trozo de código entre las directivas #IFNDEF y #ENDIF sólo será procesado si NO está declarada la variable de preprocesador especificada tras la directiva #IFNDEF.

No se permite anidamiento de #IFDEF/#IFNDEF dentro del mismo fichero. Tampoco se soporta la directiva #ELSE.

Caracteres internacionales

SUBCHAR "<carácter>" "<cadena de sustitución>";

Problemática con los caracteres internacionales

El juego de caracteres ASCII que maneja el Spectrum tiene un "espectro visible" que sólo llega hasta el código 127 (algo más para los UDG). Todo lo que esté por encima de ese código en los modernos PC, no es representable en un Spectrum per sé. PAW utiliza estos códigos por encima del 127 para generar las abreviaturas.

Esto implica que si en tu aventura escribes alegremente un mensaje en tu editor de texto como "Estás en un camión añil", luego, al ejecutarla en el emulador te mostrará algo como "Est STOP s en un cami NOTn a\il" o incluso peor.

Para solventar este problema, lo que se hace en PAW es redefinir ciertos UDG o caracteres del juego de caracteres por defecto que no se utilicen (como el %, $ o la barra invertida) y que sí los

muestra el Spectrum para representar esos extraños códigos internacionales. Una vez definidos estos códigos tendrías que indicarlos en el sitio del mensaje donde vayan a ser utilizados. En el ejemplo anterior, quedaría en tu editor algo como “Est{144}s en un cami{158}n a\nil”.

Lo anterior permitiría mostrar caracteres en nuestro idoma en la aventurqa final, pero tiene el inconveniente de dificultar el proceso de creación de la misma, ya que nos obliga, a la hora de escribir los mensajes, a tener en cuenta todos estos caracteres especiales.

Para facilitar esta labor de conversión, y poder escribir los mensajes como lo harías en cualquier editor de texto, InPAWS te permite especificar para ciertos caracteres, por qué código o cadena debe sustituirlos antes de generar la aventura. Si por ejemplo has definido la “á” mediante el UDG 'A', correspondiente al código 144, simplemente indica al principio de tu aventura:

SUBCHAR “á” “{144}”;

Esto hace que antes de volcar los mensajes al .tap resultante de la compilación (de hecho antes de comprimir el texto), se haga una sustitución de todas las “á” por “{144}”. A partir de ese momento si definimos un mensaje con el texto “Estás en un sitio muy bonito”, InPAWS lo tratará como si hubieras tecleado “Est{144}s en un sitio muy bonito”.

El carácter de sustitución sólo puede tener longitud 1, es decir, ser realmente un sólo carácter. En cambio la cadena de sustitución podrá tener la extensión que consideres, pero recuerda que debe cumplir las normas de codificación para el resto de cadenas “normales”, ya que va a formar parte de varias de ellas.

Algunos ejemplos de sustitución de caracteres:

SUBCHAR "¿" "{146}";

SUBCHAR "á" "{144}";

SUBCHAR "é" "{148}";

SUBCHAR "í" "{152}";

SUBCHAR "ó" "{158}";

SUBCHAR "ú" "{159}";

SUBCHAR "¡" "{147}";

SUBCHAR "ñ" "{149}";

SUBCHAR "Ñ" "{150}";

SUBCHAR "ü" "{151}";

Agradecimientos

A Graeme Yeandle y Tim Gilberts por crear PAW.

A los autores de la herramienta UNPAWS: José Luis Cebrian (versión original), Carlos Sánchez (Versión en Pascal y cargador del formato Z80) y Alexander Katz (128k y soporte de gráficos), por la labor arqueológica realizada en el formato de una base de datos PAW, sin cuyo destripe, esta herramienta no hubiera sido posible. Disculpas a los tres por haber puesto en la mesa de disección los fuentes de vuestra herramienta.

Contactar con el autor

En el momento de redactar este documento (febrero de 2009) podeis hacer llegar vuestras sugerencias, quejas, errores encontrados, o cheques al portador a la siguiente dirección de correo:

lane.mastodon [en] gmail.com

Última modificación: 20 de febrero de 2009