

## 8. Procesos II

Ya hemos aprendido lo más básico para **definir** e **invocar** nuevos procesos. Es importante que en este punto tengas bien clara la diferencia entre definir un proceso e invocarlo:

<b>Definir un proceso:</b>	Consiste en escribir todo su código en un bloque PROCESS - END. Sólo se hace una vez. El proceso no aparece en pantalla hasta que sea invocado..
<b>Invocar un proceso:</b>	Consiste en escribir su nombre seguido de "( );" Podemos hacerlo tantas veces como queramos. Hace que el proceso aparezca en pantalla.

Si has realizado correctamente el programa del tema anterior, habrás comprobado que el proceso disparo no era capaz de ser invocado en la misma coordenada x,y en la que se encontraba el protagonista. En este tema vamos a ver una sencilla forma de lograr esto.

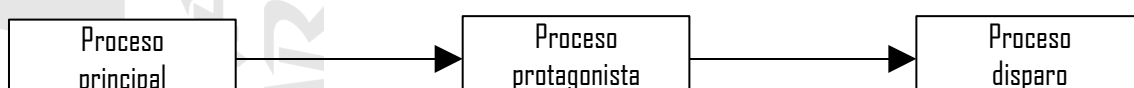
### 8.1 Jerarquía de procesos en Benu

Ahora que distinguimos entre definir e invocar un proceso, podemos comenzar con uno de los aspectos más avanzados del lenguaje Benu. Se trata de la jerarquía de procesos, y es que todos los procesos en Benu se organizan exactamente igual que un árbol genealógico.

El **padre** que gobierna toda la jerarquía de procesos es el proceso principal, ése que está definido dentro del primer bloque BEGIN - END y que por ahora se encarga de ejecutar las instrucciones set\_mode(); y load\_fpg();.

El proceso protagonista era invocado dentro del BEGIN - END del proceso principal, por tanto podemos decir que el protagonista es **hijo** del proceso principal, y obviamente, que el proceso principal es el **padre** del protagonista.

Finalmente el proceso disparo era invocado dentro del BEGIN - END del proceso protagonista, por tanto decimos que el proceso disparo es **hijo** del proceso protagonista, y de la misma manera, el proceso protagonista es **padre** del proceso disparo.



## 8.2 Herencia de procesos en Benu

Apoyándose en la jerarquía de procesos, Benu nos ofrece un mecanismo muy sencillo para hacer que ciertos procesos puedan heredar aspectos (Como graph, x, y, size, etc.) de sus padres.

Esto nos servirá para, por ejemplo, hacer que el proceso disparo herede la coordenada x, y de su padre (El protagonista) y así lograr que los disparos nos ofrezcan la sensación correcta, apareciendo siempre en la posición del protagonista, sea cual sea.

Para ello usaremos uno de los aspectos más avanzados de un proceso Benu. Se trata de un aspecto llamado **father**, que mediante el operador '.' nos permite acceder a los aspectos básicos del proceso padre según esta table:

<b>father.graph</b>	Indica el número de gráfico en el archivo FPG de nuestro proceso padre.
<b>father.x</b>	Indica la posición horizontal en píxeles de nuestro proceso padre.
<b>father.y</b>	Indica la posición vertical en píxeles de nuestro proceso padre.
...	...y lo mismo con father.size, father.angle, father.flags, etc.

Para hacer que el disparo herede la coordenada x, y de su padre (El protagonista) al ser invocado, podemos usar la asignación, asignando a su x el valor de father.x y asignando a su y el valor de father.y. Estas instrucciones estarían dentro del bloque BEGIN - END del proceso disparo, antes del bloque LOOP - END, ya que nos interesa que la herencia sólo se realice una vez nada más ser invocado el disparo.

```

PROCESS disparo()
BEGIN
    x=father.x;
    y=father.y;
    graph=45;
    LOOP
        x=x+10;
        FRAME;
    END
END
//Definición del proceso disparo:
//Inicio de las instrucciones del proceso
//Asignamos la x del padre
//Asignamos la y del padre
//Asignamos el gráfico que queramos
//A partir de aquí repetimos siempre
//Incrementamos la x en 10
//Mostramos resultado en la pantalla
//Fin de la repetición
//Fin de las instrucciones del proceso

```

Además de **father**, podemos acceder a la jerarquía en otras direcciones utilizando **son** (El último hijo de un proceso), **smallbro** (El hermano anterior de un proceso) y **bigbro** (El siguiente hermano de un proceso). También podríamos acceder a un abuelo utilizando **father.father**, pero no es necesario complicarse, ya que en la práctica nos bastará con usar **father** solamente.