Formato de los mensajes de correo electrónico

En la página anterior hemos hablado acerca de la manera de enviar un e-mail y veíamos la forma de insertar el cuarto parámetro de la función **mail** para incluir algunos elementos de los encabezados MIME.

El formato de los mensajes está especificado en una serie de normas conocidas como el **MIME** (**M***ultipurpose* **I***nternet* **M***ail* **E***xtensions*) en las que se establecen los contenidos y la sintaxis de las diferentes partes de un mensaje.

Recordemos que la función

mail(*dest, asunt, men, cab*)

tiene cuatro parámetros y que las especificaciones del MIME aluden a los dos últimos, es decir a *men* (el **cuerpo** del mensaje) y *cab* que es el **encabezado** del mismo.

Respecto a *dest* (destinatario) y *asunt* no se requieren más comentarios que reiterar la necesidad de incluir esos valores (e-mail del destinatario y *asunto*) bien directamente, como parámetro en la función, o a través de una variable tal como hemos comentado en la página anterior.

Cabeceras de los mensajes

Los diferentes elementos de la cabecera de un mensaje deben insertarse **siempre** separados por **saltos de línea** bien pulsando *Enter* o incluyendo la secuencia **\n** *dentro de la misma de línea*.

No pueden incluirse espacios, ni al comienzo de las nuevas líneas ni después de \n, y las *comillas* –que han de contener **todo** el encabezado– se abren delante del primero de ellos y no se cierran hasta después de haber escrito el último.

Pueden contener lo siguiente:

Date: *xxxxx*

Date: debe escribirse con esta sintaxis *exactamente*.

El parámetro *xxxxx* es una *cadena* que contendrá la fecha de envío del mensaje y que puede obtenerse a través de una de las funciones de

Las cabeceras MIME de un mensaje

Aquí tienes un ejemplo con los diferentes elementos del encabezado de un mensaje. Como ves, he incluido todos los elementos dentro de la función **mail**.

```
<?
mail("juan@mispruebas.com", "Cabeceras", "Prueba de cabeceras",
        "Date: 24 de Junio de 2001
MIME-Version: 1.0
From: Estudiante Perico<perico@mispruebas.com>
Cc:perico@mispruebas.com
Bcc:andres@mispruebas.com
Reply-To: perico@mispruebas.com
X-Mailer: PHP/".phpversion());
?>
```

ejemplo98.php

Una forma un poco más depurada del script anterior podría ser esta que incluimos aquí debajo.

Sus particularidades son las siguientes:

Recogemos los *datos* en variables e insertamos en la función **mail** esas variables
La variable **$cabecera** tiene algunas singularidades:
La vamos construyendo añadiendo subcadenas: date, from, etc. etc.
En cada subcadena dejamos *pegado* el contenido a las comillas iniciales

Ver índice | Búsqueda rápida | Página anterior | Página siguiente

```
<?
# datos del mensaje
$destinatario="juan@mispruebas.com";
$titulo="Cabeceras en variables";
$mensaje="Nueva prueba de cabeceras";
$responder="andres@mispruebas.com";
$remite="andres@mispruebas.com";
$remitente="Otra vez Andres"; //sin tilde para evitar errores de servidor
# cabeceras
$cabecera ="Date: ".date("l j F Y, G:i")."\n";
$cabecera .="MIME-Version: 1.0\n";
$cabecera .="From: ".$remitente."<".$remite.">\n";
$cabecera .="Return-path: ". $remite."\n";
$cabecera .="X-Mailer: PHP/". phpversion()."\n";

if( mail($destinatario, $titulo, $mensaje,$cabecera)){
   echo "mensaje enviado";}else{print "el mensaje no ha podido enviarse";
}
?>
```

ejemplo99.php

Algunas funciones PHP que *incorporamos* en estos ejemplos

Podrás ver en estos ejemplos algunas *funciones raras* que vamos a comentar seguidamente:

uniqid(*pre,bol*)

fecha de PHP tal como puedes ver en el ejemplo.

MIME-Version: 1.0

Este elemento de la *cabecera* especificará la versión MIME que ha de utilizar el cliente de correo para poder interpretar adecuadamente el contenido de los mensajes.

From: *remitente<e-mail>*

Este elemento de la cabecera permite indicar *el nombre del remitente* (remitente) y su *dirección e-mail* siguiendo la sintaxis que se especifica. El nombre, como un elemento independiente y la *dirección e-mail* dentro de **< >**.

¡Cuidado!

No debemos poner *comillas* ni en el nombre del remitente, ni en la dirección e-mail, ni en la fecha, etcétera.

Respecto a **Cc:** y **Bcc:** ; **Reply-To:** y **X-Mailer:** son válidos los comentarios que hemos hecho en la página anterior.

Si no se especifica lo contrario, los mensajes se envían como **texto sin formato**, pero existen opciones que permiten especificar el formato que ha de tener un mensaje.

La especificación de un formato obliga a *incluir* otro elemento en *cabecera del mensaje*:

Content-Type:

Este elemento debe ir seguido de la *especificación* en la que se indique el tipo de contenido. Tiene la sintaxis: *tipo/subtipo*

El MIME establece un gran variedad de opciones para este propósito. Hablaremos de dos de ellas:

text/plain

El **text/plain** es la opción por defecto y señala que el contenido del mensaje es de tipo **texto** (*text*) y del subtipo **sin formato** (*plain*)

text/html

Como la opción anterior, es tipo **texto**, pero en este caso, el *subtipo* es **html** con lo cual el mensaje se visualizará en formato **html** siempre que el *cliente de correo* permite esa posibilidad.

Mensajes *multiparte*

Los tipos anteriores permiten enviar *mensajes simples* (sin ficheros adjuntos) en uno u otro formato, pero el MIME nos da opciones para insertar dentro de un mismo mensaje

Genera un *identificador único* basado en la hora actual del sistema expresada en microsegundos y con una longitud de 13 caracteres.

El parámetro **pre** permite establecer una cadena o número (puede ser una *cadena vacía*) que se antepone al identificador generado por la función.

Opcionalmente permite el segundo parámetro **bol** que debe ser un valor *booleano* (TRUE ó FALSE) o también 0 ó 1.

Cuando este parámetro es **TRUE** añade al final de la cadena generada anteriormente otra subcadena numérica -generada aleatoriamente- de nueve dígitos, que *refuerza* la **unicidad** del identificador.

eregi_replace(*busca, reemplaza, cadena***)**

Busca en la cadena especificada en el parámetro **cadena** (que puede ser una cadena o una variable que contenga una cadena) las subcadenas especificadas en **busca** (pueden ser *expresiones regulares*) y sustituye esas subcadenas por el contenido del parámetro **reemplaza**.

Esta función *devuelve* la cadena modificada.

strip_tags(*cadena, excepciones***)**

Suprime **todas las etiquetas HTML** contenidas en **cadena** salvo las que se indiquen en **excepciones**.

Por ejemplo: **strip_tags($cadena, '<i><u><b>')** eliminaría todas las etiquetas HTML, salvo las indicadas aquí y sus correspondientes *cierres*.

Si no se especifican **excepciones** elimina todas las etiquetas.

base64_encode(*cadena***)**

Devuelve una cadena codificada en **base64**. Esta codificación se hace para permitir que las informaciones binarias puedan ser correctamente manipuladas por sistemas que no generan correctamente los 8 bits, tal como ocurre frecuentemente en los *cuerpos* de los mensajes de correo electrónico.

base64_decode(*cadena***)**

Realiza el proceso inverso a la anterior. **Decodifica** una cadena previamente codificada en **base64**.

chunk_split(*cadena, longitud, separador***)**

Devuelve una cadena obtenida al insertar en la *cadena* especificada -a intervalos del número de caracteres especificados en el parámetro numérico **longitud**- el contenido de una **subcadena** indicada en el parámetro **separador**.

Por defecto -cuando no se especifican los parámetros- *longitud* es igual a **76 caracteres** y el *separador* es la cadena **\r\n** (*retorno y salto de línea*).

Esta función se utiliza para convertir al formato especificado en la **RFC 2045** (especificación para MIME) las cadenas obtenidas por *base64_encode*.

Es el formato habitual de los *ficheros adjuntos* de los e-mail.

Mensaje con contenido *alternativo*

```
<?
# creamos la variables "salto" para "mayor comodidad
# un salto es la secuencia retorno de carro-nueva línea
# dos saltos es algo similar pero duplicado

$UN_SALTO="\r\n";
$DOS_SALTOS="\r\n\r\n";

# creamos el remitente, etc. y también la que parte que
```

elementos de diferentes tipos y subtipos.

Las opciones de mayor interés son las siguientes:

multipart/alternative

Es la forma de especificar que el mensaje tiene *varias partes* (*multipart*) de las que el destinatario *ha de ver una sola* (*alternative*).

Se podría utilizar en casos en los que sea necesario prever la posibilidad de que un mensaje con formato HTML pueda ser visualizado como *texto plano* cuando el *cliente de correo* no soporte HTML.

Podemos hacer un mensaje *a medida* que se presentará de una forma u otra según el *cliente* utilizado para leerlo.

multipart/mixed

Cuando en el **Content-Type** se establece el tipo *multiparte* y el subtipo *mezclado* (mixed) será cuando tengamos la posibilidad de *adjuntar ficheros* al mensaje.

Las *diferentes partes* de un mensaje deben ir separadas – tanto en modo *alternativo* como *mezclado*– y para ello hay que incluir un nuevo elemento en el encabezado. Se trata de un *separador* al que se llama **boundary**.

boundary=*cadena*

Dentro del **encabezado** y *siempre en línea aparte* (fíjate que en los ejemplos o está en línea aparte o aparece el \n) debemos incluir el elemento **boundary=** (sin símbolo de $ delante) y detrás del signo igual una cadena (en este caso **entre comillas**) que en principio puede ser una cadena cualquiera que no contenga espacios, aunque lo habitual es incluirla con el formato que podemos ver en los ejemplos.

El cuerpo del mensaje

En su formato más simple el cuerpo del mensaje contiene únicamente texto, pero cuando se trata de *multipartes* deberá contener necesariamente: los *separadores* de las diferentes partes, los *encabezados* de cada una de las partes y sus respectivos *contenidos*.

La secuencia habría de ser de este tipo:
Separador
Content-type
Content-Transfer-Encoding
**Content-Disposition
**Lectura del fichero
**Codificación
Inserción en cuerpo
Separador

```
# contiene el código HTML del mensaje

$destinatario="juan@mispruebas.com";
$titulo="Mensaje alternativo Texto Plano - HTML ";
$mensaje="<html><head></head><body bgcolor='#ff0000'>";
$mensaje .="<font face='Arial' size=6>Prueba HTML. </font>";
$mensaje .="aquí pueden ir tildes: á, é, í, ó, ú, ñ</body></html>";
$responder="andres@mispruebas.com";
$remite="andres@mispruebas.com";
$remitente="Andres Perez y Perez";
// omitimos las tildes en encabezados para evitar errores de servidor

# creamos el separador de bloques del mensaje
# anteponiento "_separador" aunque podríamos haber puesto "tiburcio"
# generamos un identificador unico utilizando un numero aleatorio
# como "semilla" y luego lo codificamos con la función md5

$separador ="_separador".md5 (uniqid (rand()));

# creamos la variable cabecera con los elementos
# ya utilizados en los ejemplos anteriores y ponemos al final
# de cada elemento UN SALTO DE LINEA

$cabecera = "Date: ".date("l j F Y, G:i").$UN_SALTO;
$cabecera .="MIME-Version: 1.0\n";
$cabecera .="From: ".$remitente."<".$remite.">".$UN_SALTO;
$cabecera .= "Return-path: ". $remite.$UN_SALTO;
$cabecera .="Cc:perico@mispruebas.com".$UN_SALTO;
$cabecera .="Reply-To: ".$remite.$UN_SALTO;
$cabecera .="X-Mailer: PHP/". phpversion().$UN_SALTO;

# AQUÍ DEFINIMOS EL CONTENIDO MULTIPART, fíjate que lo acabamos con ";"

$cabecera .="Content-Type: multipart/alternative;".$UN_SALTO;

# insertamos BOUNDARY (fíjate que dejo un espacio
# en BLANCO DELANTE y ponemos al FINAL los DOS SALTOS DE LINEA

$cabecera .=" boundary=$separador".$DOS_SALTOS;

# colocamos el primer separador(con los dos guiones delante)
# antes de insertar la primera parte del mensaje
# que es el texto plano para el caso de que el cliente de correo
# no soporte HTML

$texto_plano ="--$separador".$UN_SALTO;

# especificamos el tipo de contenido y la codificación
# e inserto DOS SALTOS AL FINAL ya que ahi acaba la cabecera de esta parte
$texto_plano .="Content-Type:text/plain; charset=\"ISO-8859-1\"".$UN_SALTO
$texto_plano .="Content-Transfer-Encoding: 7bit".$DOS_SALTOS;

# cambiamos las etiquetas "<br>" por saltos de línea
# y luego quitamos todas las etiquetas HTML del cuerpo del mensaje
# ya que el texto plano no debe llevar ese tipo de etiquetas

 $extractor= strip_tags(eregi_replace("<br>", $UN_SALTO, $mensaje));

 $texto_plano .=$extractor;

# insertamos un nuevo separador para señalar el final
# de la primera parte del mensaje y el comienzo de la segunda
# en este caso ponemos UN SALTO delante del separador ya que de lo contrar
# al componer el mensaje se uniría con la cadena texto_plano anterior
# que no tiene SALTO DE LINEA AL FINAL

$texto_html =$UN_SALTO."--$separador".$UN_SALTO;

 # especificamos el encabezado HTML para el siguiente bloque
 # y ponemos en la ultima línea los DOS SALTOS DE LINEA

$texto_html .="Content-Type:text/html; charset=\"ISO-8859-1\"".$UN_SALTO;
$texto_html .="Content-Transfer-Encoding: 7bit".$DOS_SALTOS;
#añado la cadena que contiene el mensaje
 $texto_html .= $mensaje;

# insertamos  SOLAMENTE un SALTO DE LINEA
 # estamos al funal del mensaje
```

.....
otra parte
...
Separador final
Los apartados señalados con ** sólo se incluirían en el caso de que junto con el mensaje se *adjunten* ficheros.

Content-type

Los tipos y subtipos más habituales son los siguientes:

Para incluir textos:
los ya mencionados
text/plain
text/html

Para imágenes:
según el tipo de imagen
image/jpeg
image/gif

Para sonidos:
audio/basic

Para vídeo:
video/mpeg

Para ejecutables, comprimidos y otros ficheros adjuntos:
application/octet-stream

En cualquier caso, si quieres utilizar algún otro tipo de archivo puedes *consultar en la web* las especificaciones del MIME.

Aparte de *tipo/subtipo* puede añadirse a *Content-type* -en el caso de texto- separado por *punto y coma*, la especificación del tipo de alfabeto (**charset=**) seguida del tipo de codificación (te sugerimos el "ISO-8859-1" que hace alusión al alfabeto latino).

Cuando se trata de **ficheros adjuntos** deberemos poner, después del *punto y coma*, **name=** seguido del *nombre y extensión* del fichero que se adjunta.

Content-Transfer-Encoding

Este apartado del encabezado puede especificar una de los siguientes codificaciones:

7BIT
8BIT
BASE64
BINARY
QUOTED-PRINTABLE

La transferencia codificada en **7bit** representa la codificación habitual en el formato ASCII de 7 bits.
No permite caracteres ASCII con un código mayor que 127.

Quoted-printable constituye una de las alternativas al formato ASCII de 7 bits.

Esta codificación suele usarse cuando la mayoría de los caracteres del mensaje puede escribirse con formato US-ASCII de 7 bits.

```
 $texto_html .=$UN_SALTO;

 # unimos ambas cadenas para crear el cuerpo del mensaje

  $mensaje=$texto_plano.$texto_html;

 # enviamos el mensaje utilizando


if( mail($destinatario, $titulo, $mensaje,$cabecera)){
        echo "mensaje enviado ";}else{print "ha habido errores en el envio
}

?>
```

ejemplo100.php

Mensaje con ficheros adjuntos

```
<?
# definimos estas variables igual que en el ejemplo anterior

$UN_SALTO="\r\n";
$DOS_SALTOS="\r\n\r\n";

#incluimos en varias, asunto, un texto en HTML
# remitente, etc. etc.

$destinatario="perico@mispruebas.com";
$titulo="Mensaje con dos fichero adjuntos";
$mensaje="<html><head></head><body bgcolor=\"#ff0000\">";
$mensaje .="<font face=\"Arial\" size=6>Prueba HTML </font>";
$mensaje .="</body></html>";
$responder="andres@mispruebas.com";
$remite="andres@mispruebas.com";
$remitente="Andres otra vez";

# definimos el separador de parte
# con el mismo procedimiento del ejemplo anterior

$separador = "_separador_de_trozos_".md5 (uniqid (rand()));

# insertamos los datos de la cabecera del mensaje

$cabecera = "Date: ".date("l j F Y, G:i").$UN_SALTO;
$cabecera .= "MIME-Version: 1.0".$UN_SALTO;
$cabecera .= "From: ".$remitente."<".$remite.">".$UN_SALTO;
$cabecera .= "Return-path: ". $remite.$UN_SALTO;
$cabecera .= "Reply-To: ".$remite.$UN_SALTO;
$cabecera .="X-Mailer: PHP/". phpversion().$UN_SALTO;

# especificamos el tipo de contenido mutipart/mixed
# ya que ahora insertaremos ficheros de distinto tipo

$cabecera .= "Content-Type: multipart/mixed;".$UN_SALTO;

# insertamos el valor de boundary haciéndola igual a  $separador
# y acabamos con DOS SALTOS porque es el FINAL DE LA CABECERA

$cabecera .= " boundary=$separador".$DOS_SALTOS;

/* Parte primera del envio -Mensaje en formato HTML
   ================================================

Separador inicial
------------------------------- */
$texto ="--$separador".$UN_SALTO;

/* Encabezado parcial
   ------------------  */
/* especificamos que este primer elemento
será texto y que irá codificado en formato 7 bits */
```

Prevé que los caracteres con códigos ASCII superiores 127 se expresen mediante un mecanismo especial.

Evita, entre otras cosas, que las *letras con tilde* y algunos otros *caracteres especiales* se visualicen incorrectamente. Es la forma de codificación más recomendable para *textos*.

La codificación en **base64** convierte cadenas binarias en cadenas de texto, con lo cual pueden ser enviadas de forma más segura. Es la forma de codificación habitual de las imágenes y los ficheros *exe*, *zip*, etcétera.

Content-Disposition

Se utiliza únicamente cuando se insertan ficheros adjuntos.

Permite dos opciones: **inline** o **attachment**.

Inline permite que los contenidos se visualicen junto con el cuerpo del mensaje mientras que con **attachment** sólo aparecerían como ficheros adjuntos.

Por lo que hemos podido comprobar *Outlook Express* no suele respetar esa condición y presenta siempre las imágenes en el mensaje. Sin embargo, sí funciona en los correos web.

Este elemento del encabezado lleva -*separada por punto y coma*- una segunda parte.

El **filename=**, donde se puede especificar entre comillas un nombre y una extensión (igual o distinta de la original) con la que se denominará al fichero en el mensaje recibido.

Lectura del fichero

Cuando se trata de insertar un *fichero* el proceso es el típico de lectura de ficheros, es decir:
Hay que crear el identificador de recurso del fichero en modo **lectura**.
Recoger en una variable el *buffer* de lectura.
Cerrar el fichero.

Codificación

Una vez recogido en el fichero a transmitir en una variable, el paso siguiente es *codificar* esa variable.

Utilizaremos la codificación más habitual y flexible –**base64**– que requerirá el uso de dos nuevas funciones PHP:

base64_encode
chunk_split

Mediante la primera se realiza la codificación propiamente dicha mientras que la segunda organiza el

```
$texto .="Content-Type: text/html; charset=\"ISO-8859-1\"".$UN_SALTO;
$texto .="Content-Transfer-Encoding: 7bit".$DOS_SALTOS;

/* Contenido de esta parte del mensaje
   ----------------------------------*/
# ya teniamos escrito el texto del mensaje más arriba
# simplemente lo añadimos a la cadena de texto

  $texto .= $mensaje;

  #la variable $texto recoge esta parte del documento
  # la uniremos al final con las siguientes

/*  Separador de partes
    ------------------- */

$adj1 = $UN_SALTO."--$separador".$UN_SALTO;

/* Parte segunda de mensaje -Fichero adjunto nº 1
   ================================================= */

/* Encabezado parcial
   -----------------  */
# especificamos el tipo de contenido image/jpeg
# ya que ese será el documento que vamos a enviar
# ponemos el nombre del fichero (debemos tenerlo en el servidor
# con ese mismo nombre)
# establecemos in line como disposición para que pueda ser visualizado
# directamente en el cuerpo del mensajes
# en filename le asignamos el nombre con el que queremos que sea
# recibido por el destinatario
# por ultimo especificamos la codificacion como base64

$adj1 .="Content-Type: image/jpeg;";
$adj1 .=" name=\"casa08.jpg\"".$UN_SALTO;
$adj1 .="Content-Disposition: inline; ";
$adj1 .="filename=\"leoncio.jpg\"".$UN_SALTO;
$adj1 .="Content-Transfer-Encoding: base64".$DOS_SALTOS;

/* Lectura previa del fichero a adjuntar
   ---------------------------------------  */
    # abrimos el fichero en modo lectura (r)
   # y leemos todo su contenido midiendo previamente
   # su longitud con filesize
   # recogemos en $buff el contenido del fichero
   # y cerramos después

      $fp = fopen("casa08.jpg", "r");
           $buff = fread($fp, filesize("casa08.jpg"));
       fclose($fp);

/* Codificación del fichero a adjuntar
   ---------------------------------------  */
# codificamos en base 64 y troceamos en lineas de 76 caracteres
# y añadimos el resultado a la variable adj1

$adj1 .=chunk_split(base64_encode($buff));

/*  Separador de partes
    ------------------- */

$adj2 = $UN_SALTO."--$separador".$UN_SALTO;

/* Tercera parte de mensaje -Fichero adjunto nº 2
   ================================================= */

/* Encabezado parcial
   -----------------  */
# los contenidos del encabezado son similares al caso anterior
# con la salvedad de que el contenido es ahora
# application/octet-stream ya que contiene un fichero ejecutable
# y la disposicion es attachment, no tiene sentido tratar
# de visualizar un fichero zip

$adj2 .="Content-Type: application/octet-stream;";
             $adj2 .=" name=\"apachito.zip\"".$UN_SALTO;
$adj2 .="Content-Disposition: attachment;
              filename=\"apachito.zip\"".$UN_SALTO;
```

fichero codificado en trozos, de 76 caracteres cada uno, insertando detrás de cada uno un *salto de línea*.

Si *analizas* un mensaje de correo que contenga un fichero adjunto –propiedades, ver codificación–, podrás ver esa fragmentación *tan cuidada* -un montón de líneas de texto *muy raro*- perfectamente alineadas por los márgenes por efecto de *chunk_split*.

Inserción en el cuerpo

La fase final del proceso es la de *agrupar* los diferentes *trozos* en una sola variable, que será la que se insertará como parámetro texto en la función e-mail.

¡Cuidado!

La inserción de ficheros adjuntos requiere que éstos estén disponibles en el servidor por lo que, antes de enviarlos, habrá que *subirlos* al servidor utilizando un proceso como el que hemos analizado cuando hablábamos de *Transferencia de ficheros*.

Sobre los ejemplos

Hemos incluido dos ejemplos relativos al envío de ficheros en formato HTML y con ficheros adjuntos.

No entraremos en el estudio detallado del MIME.

Quedaremos únicamente en sus aspectos funcionales en cuanto a los requerimientos de formato, separadores, etc.

Quizá te parezca *obsesivo* el hincapié que hacemos en los ejemplos sobre los detalles de la sintaxis.

Nuestra insistencia se debe al carácter *sumamente estricto* de la especificación MIME, donde un *simple salto de línea* puede ser la causa de que script deje de funcionar.

```
$adj2 .="Content-Transfer-Encoding: base64".$DOS_SALTOS;

/* Lectura previa del fichero a adjuntar
   ---------------------------------------  */
   # abrimos el fichero en modo lectura (r)
   # y leemos todo su contenido midiendo previamente
   # su longitud con filesize
   # recogemos en $buff el contenido del fichero
   # y cerramos después

      $fp = fopen("apachito.zip", "r");
         $buff = fread($fp, filesize("apachito.zip"));
      fclose($fp);

/* Codificación del fichero a adjuntar
   ---------------------------------------  */

$adj2 .=chunk_split(base64_encode($buff));

/*  Separador final YA NO HAY MAS PARTES
    ---------------------------------------- */

$adj2 .=$UN_SALTO."--$separador".$UN_SALTO;

/*  Unión de todas las PARTES
    ---------------------------------------- */
# unimos en la variable mensaje todos los elementos
# y lo hacemos por el orden en el que fueron creados

  $mensaje=$texto.$adj1.$adj2;

/*  Envio del mensaje
    ---------------------------------------- */

if(mail($destinatario, $titulo, $mensaje,$cabecera)){
        echo "mensaje enviado";}else{print "ha habido problemas";
}

?>
```

ejemplo101.php

Ejercicio nº 32

Diseña un script de forma que al cargarse la página que lo contiene se envíe –de forma automática– un mensaje de correo, indicando la fecha y hora de acceso, al usuario **juan@mispruebas.com**