RevistaSL

El software libre hecho revista

02

PROGRAMACIÓN SL

Echandole el ojo a: Anjuta IDE

Mis primeros pasos con C# y Mono

Aprende en 5min a programar con Python

Licenciamiento

Software libre... revista libre

creative commons

COMMONS DEED

Attribution 2.5 Mexico

You are free:

- to copy, distribute, display, and perform the work
- to make derivative works
- to make commercial use of the work

Under the following conditions:

Attribution. You must attribute the work in the manner specified by the author or licensor.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

This is a human-readable summary of the Legal Code (the full license).

Disclaimer

Editorial SL

Tras pantomima

Han pasado varios meses desde la última publicación de la RevistaSL, un proyecto fue bien recibido por la comunidad mexicana de software libre. Después de tan largo periodo de letargo, la revista resurge con un staff renovado pero con el mismo propósito de difusión de la cultura del software libre. Como cualquier proyecto de la comunidad, éste se realiza en el poco tiempo libre que tiene cada uno de los colaboradores y como cualquier proyecto de software libre, cualquiera puede ayudar a mejorar el producto final, es por eso que los artículos se liberan bajo licencias libres, siendo congruentes con nuestra filosofía.

El software libre no existiría sin las herramientas fundamentales para crearlo, es por eso que hemos dedicado buena parte de este número a la programación.

El año pasado realizamos la entrevista a los creadores de ienjinia, un proyecto que tiene como fin acercar a los jóvenes a la programación, escrito enteramente en Java, uno de los lenguajes que soporta Anjuta, un IDE (Entorno Integrado de Desarrollo) que facilita la creación de aplicaciones en Gtk; además de Java, Anjuta soporta C, C++, Perl y Python, este último lo hemos revisado en un breve artículo para despertar el interés de los curiosos. También incluimos un artículo sobre C# y Mono, la alternativa libre a la plataforma .NET que ha tomado mucho auge en la comunidad de desarrolladores de software libre. Los temas de seguridad siempre son de interés para mucha gente y para comprobar qué tan buenas son tus contraseñas puedes usar el password cracker programado en Perl que viene en este número. Para los programadores que tengan experiencia en Pascal les resultará agradable toparse con el artículo dedicado al kernel Toro. La salida de este número no sería posible si no se realizaran eventos por y para la comunidad ya que en éstos tuvimos la oportunidad de conocer gente interesada en el proyecto y que ahora es parte del staff. Incluimos las reseñas de los útlimos eventos realizados en el país.

Todos los miembros del staff esperamos que con la salida de este número la publicación de la revista se haga de forma periódica.
Disfrútala y compártela.

Julio M. Acuña Carrillo
Editor de GenteSL

a nombre de:
Tec. Gonzalo Javier González Rodriguez
Editor en Jefe
ggonzalez@linux-chetumal.org.mx

Staff SL

Editor en Jefe
Tec. Gonzalo Javier González Rodríguez

Editor Vistazo SL
Carlos Augusto Lozano Vargas

Editor Gente SL
Lic. Julio Acuña Carrillo

Eventos SL
Tec. Victor Hugo Cordova Madrid

Programación SL
Lic. José Luis Galicia Sánchez
Lic. Jesus Antonio Balam Jiménez

Diseño SL
Diseñador Grafico Edgar Guerra
TSU Josué Gutíerrez Hernández

Web SL
Tec. Eyden Barboza Varela

Agradecimientos

Comunidad Linux Chetumal
www.linux-chetumal.org.mx

RTM Security Team
www.zonartm.org

Grupo de Usuarios Linux del Estado de Hidalgo
www.guleh.org

Comunidad OpenSource en Cancún
www.tucancunix.org

Colectivo Estudiantil de Química UNAM
colectivoquimica.dnsalias.org

Vistazo SL
El software bajo la lupa

Anjuta IDE

Carlos Augusto Lozano Vargas
vendetta@zonartm.org

Hola!,, saludos desde la pachanguera Ciudad de México.

En esta ocasión hemos tomado como tema central de esta edición la programación; la verdad cuando Gonzalo me dijo el tema iba a tirar la toalla y casi no hay Vitazo SL, por que la programación de plano no es algo en lo que haya incursionado mucho; pero a la vez es algo que es muy recomendable dominar, y en la escuela estoy en la parte introductoria a la programación, lo que me hizo reflexionar y empezar el articulo.

En mi caso mis primeros acercamientos a la programación fueron con Perl, que me recomendó un amigo, sin embargo en la escuela aprender a programar es cosa de C, y para hacerlo usamos C++BuilderX que es un IDE desarrollado por Borland, y como supusieron, no es libre; y en busca de libertad me encontré con Anjuta que es de lo que les platicare. Pero vamos por partes.

¿Qué es un IDE?

IDE son las siglas en ingles de Integrated Development Environment algo así como Ambiente Integrado de Desarrollo, y es un conjunto de herramientas útiles al programador colocadas en un todo para no tener que estar usando las herramientas por separado y de diferentes autores. Los componentes mas comunes de un IDE son: un editor de texto, un compilador, un interprete, un depurador y después vendrán los plus que cada IDE tenga como característica especial.

Algunos IDE pueden tener soporte para varios lenguajes de programación y otros están especializados para un solo lenguaje.

Del IDE que les platicare mas a fondo es Anjuta IDE; este es un proyecto libre que busca ofrecer un IDE para C y C++, aunque también se soportan otros lenguajes como Perl. Anjuta esta escrito para GTK+ y GNOME.

Haciendonos de Anjuta

Muchas de las distribuciones mas populares de GNU/Linux ya traen entre sus paquetes a Anjunta, y sera facil de instalar usando sus respectivas herramientas (apt-get, urmpi, emerge, etc.)

Sin embargo desde el sitio web del proyecto podemos descargar el tarball para instalarlo por nosotros mismos.

Lo primero que debemos hacer una vez que lo hayamos descargado es extraer el contenido del tarball

```
$ tar –xvzf anjuta-1.2.4.tar.gz
```

Después de esta instrucción desde la linea de comandos, entraremos al directorio en donde se encuentra todo el contenido del tarball

```
$ cd ./anjuta-1.2.4a
```

Algo que siempre recomiendo en la instalación de cualquier cosa es el leer el README.TXT, esto siempre es importante en todo lo que hagamos.

Bien, continuando con la instalación deberemos de teclar las siguientes instrucciones desde la linea de comandos.

```
$ ./configure
$ make
$ make install
```

Y listo, con esto abremos terminado de costruir Anjuta IDE sin importar si esta o no empaquetado para nuestra distribución.

Echandole un ojo a Anjuta

La primer vez que inciemos Anjuta veremos un IDE sencillo, bien ordenado y que si no supieramos nada de software alternativo, bien podria pasar por un IDE de esos comerciales.

Al dar click en nuevo nos aparecera un dialogo que nos preguntara el tipo de archivo a crear, un .c, .cpp, .pl, etc. Algo que me llamo la atención de esta parte es que podemos activar una casilla con la cual nuestro nuevo programa ya aparecera comentado con la leyenda de la GPL, :D asi desde antes de escribir el código ya nuestro programa es libre.

Anjuta tiene todo lo que podemos esperar de un herramienta que nos ayudara a programar, como es un asistente que nos guiara paso a paso en la creación de nuestro proyectos; o igual y ya tenemos un proyecto echo y queremos importarlo, bueno, pues Anjunta cuenta con esta opción; y claro, ya hablando de las caracteristicas del editor pues tendremos el ya muy caracteristico resaltado de sintaxis, identado y el autocompletado de instrucciones y variables.

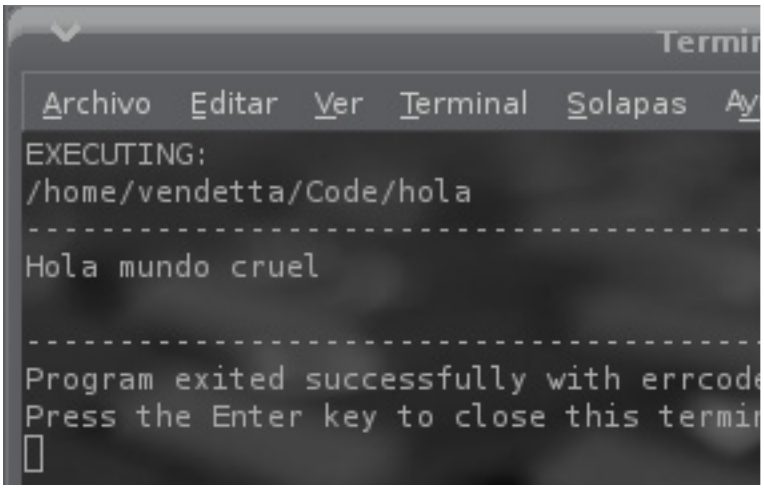
Un ejemplito en Anjuta

Anjuta es tan sencillo que realmente no hay mucho que decir, pues todo no lo pone a nuestro alcance, así que mejor mostrare un ejemplo sencillo sobre como trabajar en Anjuta.

Lo mas sencillo, pues el "Hola mundo", asi que simplemente escogemos que queremos un "C FILE" y empezamos a escribir el codigo:

```
#include <stdio.h>
main(){
printf("Hola mundo cruel\n");}
```

Guardamos nuestro archivo, nos vamos al menú Construir>Construir, Construir>Compilar y despues Construir>Ejecutar y listo podremos ver como se abre una terminal para ver ejecutado nuestro "Hola mundo".

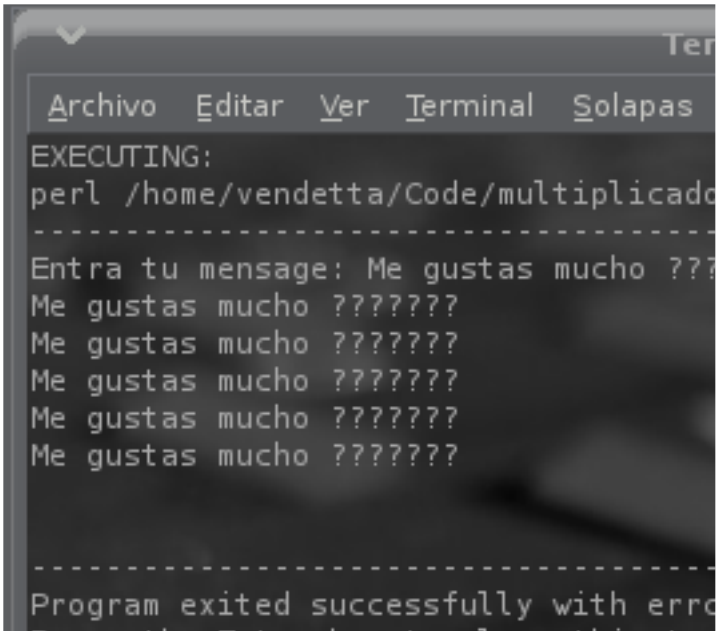


Como ven, mucho mas sencillo que en C++BuilderX.

Pero ahora veamos un ejemplo con un lenguaje no compilado, como es el caso de Perl. Aqui vamos a hacer un pequeño script para mostrarle a alguien que somos unos adoradores de su belleza :P

```
#!/usr/bin/perl

print "Entra tu mensage: ";
$message=<STDIN>;
chomp($message);
$message="$message \n" x5;
print "$message\n";
```

Aqui como no necesitamos compilar iremos directamente a Construir>Ejecutar para ver nuestro script funcionando.

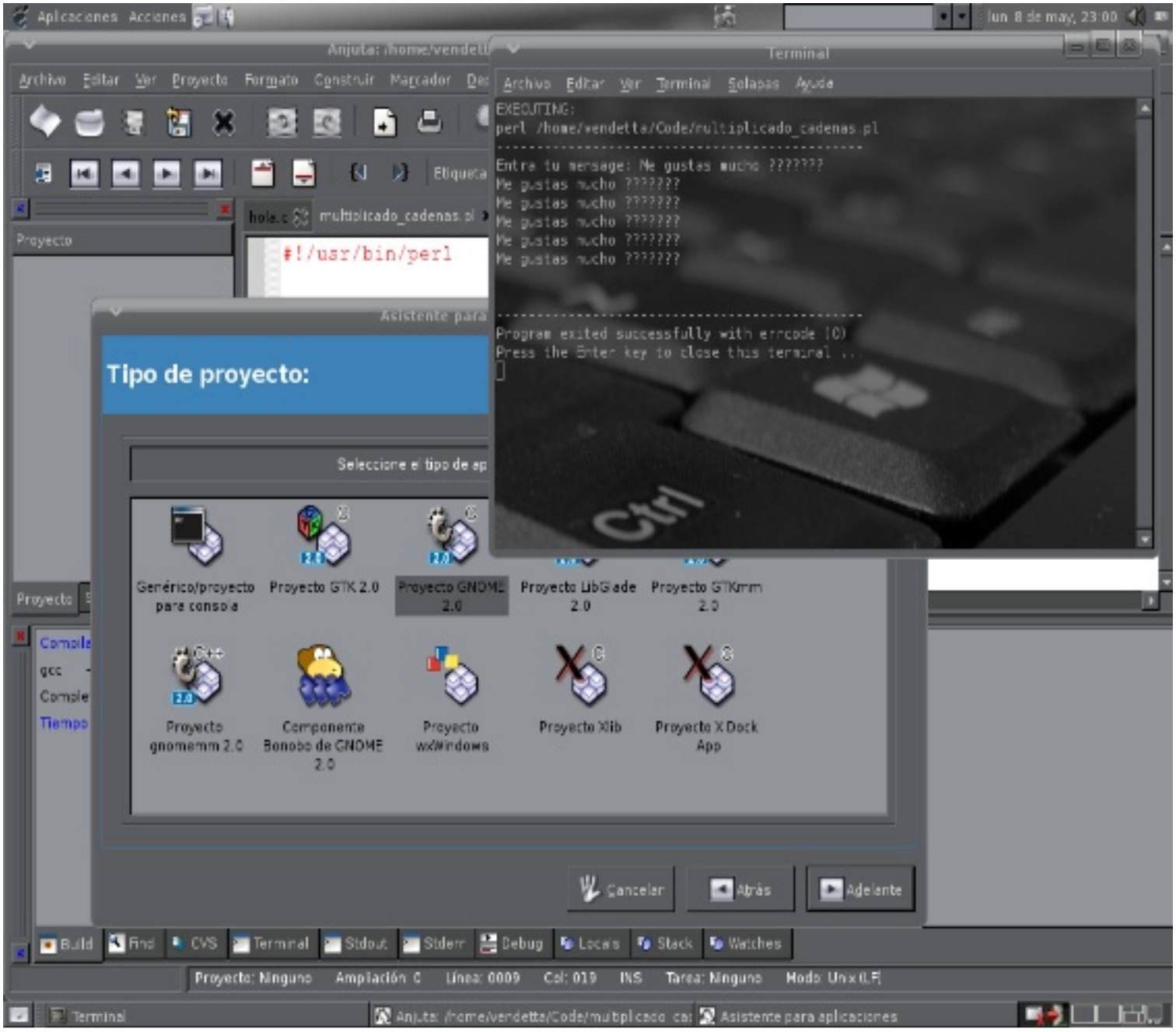


Y listo, podremos ver en este caso nuestro mensage multiplicado varias veces.

Estos ejemplos son muy sencillos, pero solo es para ilustrar que Anjuta es capaz de hacer todo tipo de cosas.

Ah! cierto, aun no pongo la screenshot de Anjuta, pero era para que ustedes mismos se animaran a instalarlo y mirarlo, pero ya que insisten aqui les presento como se ve Anjuta IDE.

Nos leemos en el proximo Vistazo SL ;)



Código SL
#include <free_code.h>

Mis primeros pasos con C# y Mono

TSU Josué Gutierrez Hernández
eusoj19.blacknash@gmail.com

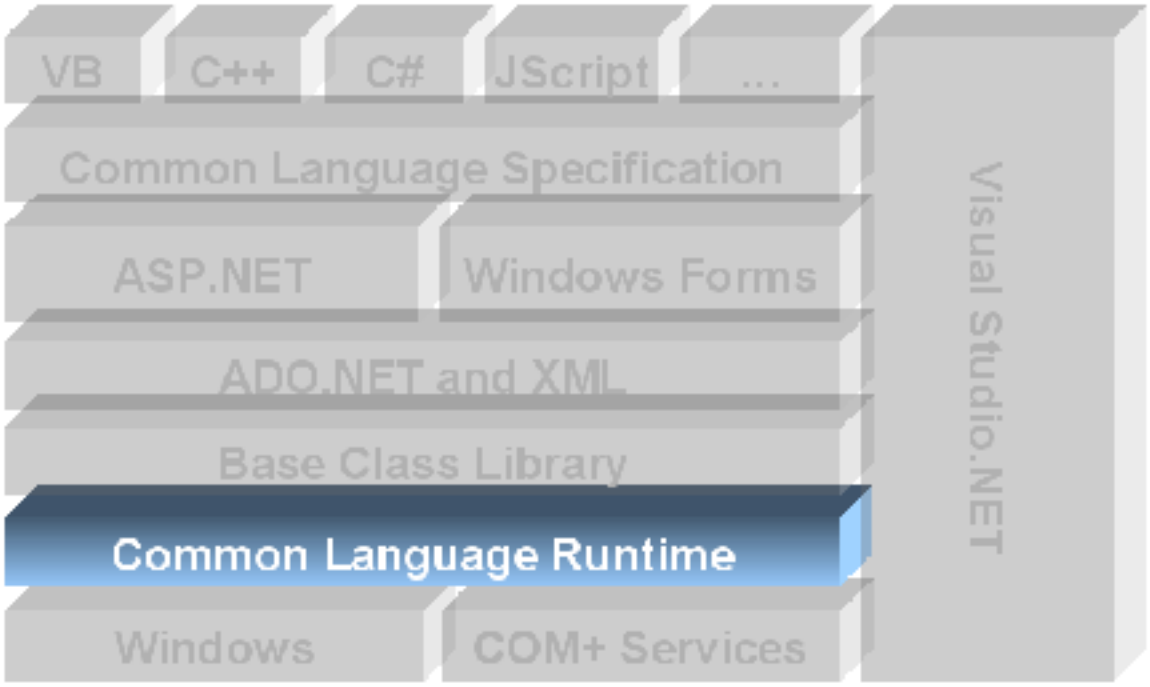
La teoría ...

En este primer articulo se pretende dar un pequeño vistazo de lo que es el Proyecto Mono, comprender que es el .NET y dar nuestros primeros pasos programando en C#.

Ahora que sabemos que es lo que trataremos, comencemos por entender que es el .NET y por que es considerada una de las innovaciones tecnológicas mas importantes lanzadas por Microsoft.

... el cual constituye la base de la plataforma .Net y denota la infraestructura sobre la cual se reúnen un conjunto de lenguajes, herramientas y servicios que simplifican el desarrollo de aplicaciones en entorno de ejecución distribuido.

Los principales componentes del Framework son:
-> El conjunto de Lenguajes
-> La Base Class Library (Biblioteca de Clases Base)
-> Common Language Runtime (Entorno Común de Ejecución para lenguajes)



.NET

Es el proyecto para desarrollo de Software con una énfasis en la transparencia de redes, y que corre bajo un “Framework”...

Siendo este ultimo el verdadero corazón del .NET, ya que es el entorno de ejecución en el que se cargan las aplicaciones que nosotros desarrollamos, sin importar el lenguaje e inclusive permitiendo una comunicación real entre ellos.

Ya que al compilar no se genera código de maquina sino CLI, el cual correrá en la maquina virtual.

MONO

Es el proyecto de código abierto impulsado por Novell para crear un grupo de herramientas libres, basadas en GNU/Linux y compatibles con .NET, de igual manera que el proyecto propietario cuenta con una maquina virtual para ejecutar el CLI, BCL .

Esto significa por ejemplo, que si defines una clase que haga una manipulación algebraica en C#, esa clase puede ser reutilizada en cualquier lenguaje que soporte el "CLI". Puede crear una clase en C#, una subclase en C++ e instanciar esa clase en un programa en Eiffel.

Ahora que ya conocemos la teoria vamos a realizar nuestro primer programa en C# utilizando mono.

Dependiendo de la distribución que se este utilizando es la manera en la que instalaran mono, si no saben cual manejador de paquetes poseen pueden bajar el mono-installer para Linux de go-mono.com y seguir las sencillas instrucciones.

Bien, ya que tenemos instalado mono en nuestro sistema GNU/Linux vamos a realizar nuestro clásico Hola Mundo, para ello podemos utilizar nuestro IDE que proporciona mono o algún editor que sea de nuestro agrado.

Nuestro código a utilizar en el primer ejemplo es el que se encuentra a continuación:

```
using System;

public class HolaMundo{
static void Main()
{
Console.WriteLine("HOLA MUNDO");
}
}
```

Guardamos nuestro archivo como holamundo.cs y para compilar en nuestra consola tecleamos

```
$mcs holamundo.cs
```

Esto nos generara un .exe. y para ejecutarlo en nuestra consola tecleamos

```
$mono holamundo.exe
```

Como resultado nos escribirá en la consola la leyenda “HOLA MUNDO”.

Como se ve en el ejemplo anterior, la compilación no es nada complicada, y la creación de un programa mucho menos. Ahora en el siguiente ejemplo vamos a crear una pequeña ventana el cual contendrá un botón que de igual manera escribirá “HOLA MUNDO” al presionarlo. Y el código es el siguiente :

```
using System;
using Gtk;

class MyWindow {
static void Main ()
{
Application.Init ();
Window myWindow = new Window ("Titulo de Ventana");
myWindow.DeleteEvent += Cierrame;
myWindow.SetDefaultSize (300, 200);

Button button = new Button ("Click");
button.Clicked += HolaMundo;
myWindow.Add (button);
myWindow.ShowAll ();
Application.Run ();
}
```

```
    static void HolaMundo(object
sender,EventArgs e)
     {
          Console.WriteLine("HOLA MUNDO");
     }

     static void Cierrame(object sender,
DeleteEventArgs e)
     {
        Application.Quit();
     }
}
```

Ok ... esto se va poniendo mas complejo... vamos a explicar un poco cada sección de código.

Application.Init() va a marcar como su nombre lo dice el inicio de nuestra aplicación.

Después de ello declaramos un objeto de tipo ventana, pasándole como parámetro el titulo que tendrá. La siguiente linea nos va a servir para cuando demos click en la "X" que cierra la aplicación, realmente la termine y no se quede colgada la ventana, y la tercera simplemente estable su tamaño.

Window myWindow = new Window ("Titulo de Ventana");
myWindow.DeleteEvent += Cierrame;
myWindow.SetDefaultSize (300, 200);

Después declaramos nuestro botón, pasándole como parámetro lo que queremos que aparezca en su leyenda, después de ellos asignamos al evento del click la función de hola mundo. Agregamos el botón a la ventana y para finalizar mostramos y corremos la aplicación.

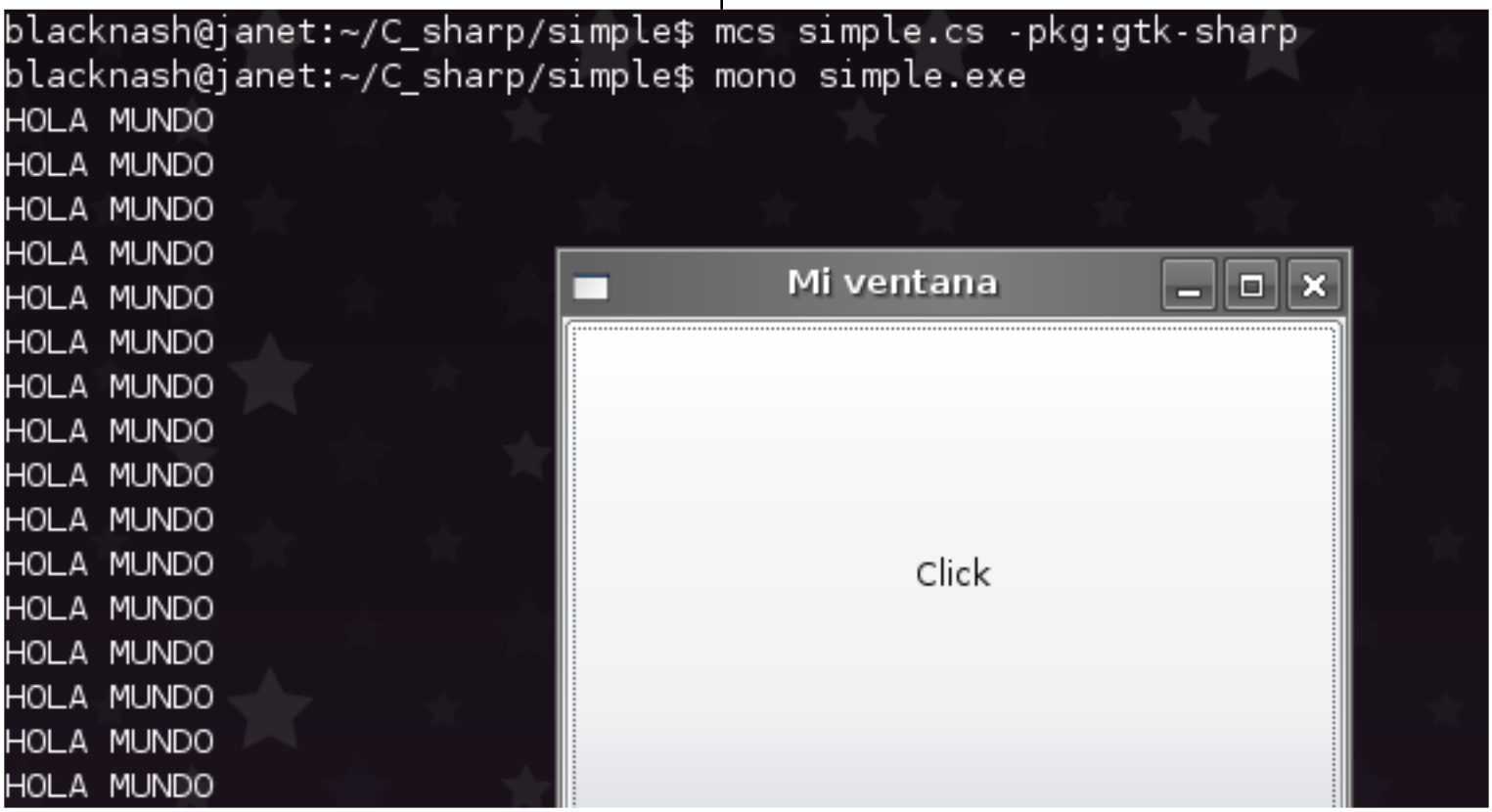
Button button = new Button ("Click");
button.Clicked += ClickButton;
myWindow.Add (button);
myWindow.ShowAll ();
Application.Run ();

Guardamos nuestro código como simple.cs y la instrucción para poder compilarlo es la siguiente*:

```
$mcs simple.cs -pkg:gtk-sharp
```

y de la misma manera ejecutamos con

```
$mono simple.exe
```



*Para poder compilar esta aplicación grafica tenemos que tener instalado GTK. Como conclusiones finales,aprendimos que es realmenente el .NET y el uso de una herramienta libre para poder desarrollarlo.

en 5 minutos

Julio Acuña Carrillo
urkonn@gmail.com

Uno de los lenguajes de programación que mas auge esta teniendo en la actualidad es Python, un lenguaje de script creado por Guido van Rossum, y no es por casualidad ya que Python permite un desarrollo de aplicaciones bastante rápido, su sintáxis es muy clara y las buenas costumbres de programación se hacen una regla. Es ideal como un primer lenguaje de programación ya que permite asimilar los conceptos importantes sin tener que aprender cosas truculentas propias de la sintáxis.

Python es un lenguaje de muy alto nivel orientado a objetos, todo en Python es un objeto, extensible, puede ser usado como un lenguaje "pegamento" embebiendo código escrito en otros lenguajes, portable, ya que una aplicación escrita en Python corre en virtualmente cualquier sistema operativo.

Echando un vistazo al clásico programa con el que se inicia el estudio de cualquier lenguaje, el famoso hola mundo, se puede ver su simplicidad.

```
>> print "hola mundo"
hola mundo
```

No podría ser mas fácil...

Control de flujo con:

while
if
elif
else
for

Tipos de datos:
string
a = 'cadena'
int
n = 42
float
i = 3.2

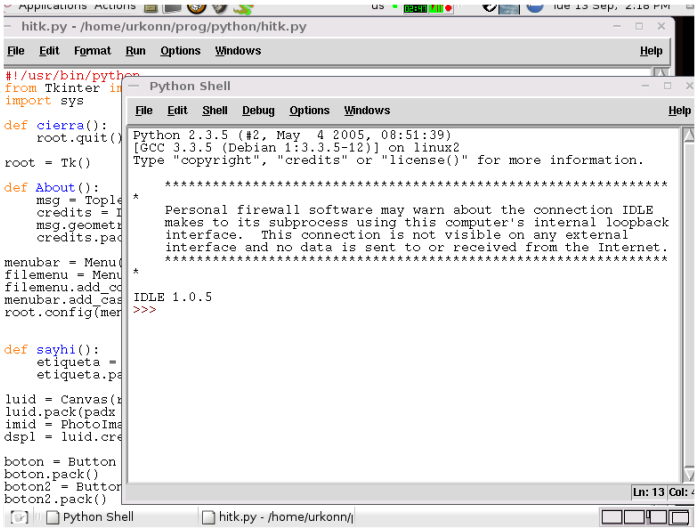
Tipos de datos secuenciales:
listas
lista = ['palabra', 5, '42']
diccionarios
dicc = {'login':'usuario',
'passwd':'secreto'}
tuplas
tupla = 43, 'monty', 'python'

Invocamos el intérprete tecleando desde la línea de comandos:

```
$python
Python 2.4.1 (#1, Jun 30 2005, 01:39:23)
[GCC 2.95.2 20000220 (Debian GNU/Linux)] on linux2
Type "help", "copyright", "credits" or "license"
for more information.
>>>
```

Desde aqui podemos empezar a probar nuestro código, pero si queremos que nuestros programas perduren, necesitaremos un editor de textos. Python trae un ambiente integrado de desarrollo (IDE) en el que es muy cómodo empezar a programar llamado idle, muy probablemente el nombre de este se debe al apellido del actor Eric Idle, integrante de los Monty Python, de donde proviene el nombre de este lenguaje.

No es necesario haber visto el programa de televisión Monty Python's Flying Circus, del cual Guido es fan, sin embargo se pueden entender de donde vienen los ejemplos usados en el tutorial de Python y disfrutar de los chistes.



Intentemos algunos ejemplos.

Python como calculadora

```
>>> a = 5 + 5
>>> a
10
```

Manejando diccionarios

```
>>> d = {'servidor':'localhost', 'base de datos':'postgres', 'uid':'usuario', 'passwd':'secreto'}
>>>d['servidor']
'localhost'
```

Operaciones con cadenas

```
>>> hola = 'Python! '
>>> print hola * 5
Python! Python! Python! Python! Python!
```

Un programa interactivo simple

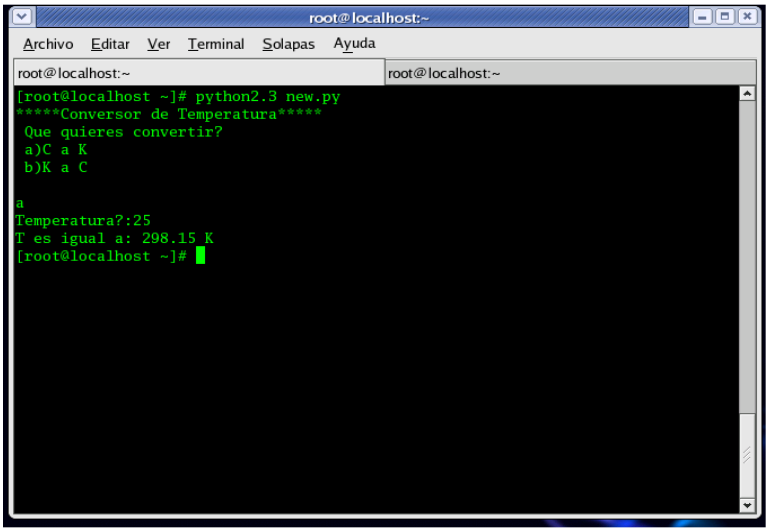
```
#! /usr/bin/python

print "*****Conversor de Temperatura*****\n Que quieres convertir?\n a)C a K\n b)K a C\n"

v = raw_input ()          #pregunta por la opcion
t = input ("Temperatura?:")       #pide por los datos
if v == "a":
print "T es igual a:", 273.15 + t,"K"
if v == "b":
print "T es igual a:", t - 273.15,"C"
```

Las variables se asignan con un signo = y se pueden declarar en cualquier momento, el signo == denota que estamos comparando.

Para correr un programa en python solo se tiene que teclear desde la línea de comandos

```
$python nombre_del_programa.py
```

La extensión py es la mas usada para indicar que es un script de Python. Invito al lector a que pruebe los ejemplos en su propia computadora.



Bueno todo esto esta muy bonito pero vamos a ver algo mas emocionante, que tal el mismo ejemplo de "hola mundo" pero en versión gráfica.

```
 #!/usr/bin/python
from Tkinter import *      #importamos el
módulo Tkinter para crear la
interfaz gráfica
import sys    #importamos el módulo sys
def cierra():#definimos una función para
terminar el programa
    root.quit()

root = Tk()   #creamos una ventana principal

def di_hola():      #definimos una función
para que pinte un mensaje en la ventana
    etiqueta = Label (fg = 'blue', text =
'Hola mundo')
    etiqueta.pack()
boton = Button (root, text = 'Empezar',
command = di_hola)  #botón asociado a la
función di_hola
boton.pack()
boton2 = Button (root, text = 'Cerrar',
command = cierra)   #botón asociado a la
función cierra
boton2.pack()



root.mainloop()
```

Vamos explicando poco a poco este programita.
La primera línea del programa es para que el sistema operativo pueda ejecutar el programa llamando al intérprete de python que esta en /usr/bin/python o en el path que se le haya dado al compilarlo, si tenemos bien configurado el sistema nuestro programa correrá con tan solo darle click.

En Python el símbolo # denota un comentario y todo lo que va después no es tomado en cuenta por el programa y se utiliza para que el programa sea mas fácil de revisar, ya sea por nosotros o por otros programadores.

En las primeras líneas importamos los módulos Tkinter y sys, el primero es una herramienta para construir interfases gráficas que viene con la distribución...

... de Python, y el segundo es un módulo estándar que también esta en la instalación por default. Uno mismo puede crear sus propios módulos o importar los módulos creados por terceros con este método. No es necesario importar todos los módulos al principio del programa, pero como este es un programa pequeño y por claridad, lo hacemos asi.

En la parte

```
def cierra():
    root.quit
```

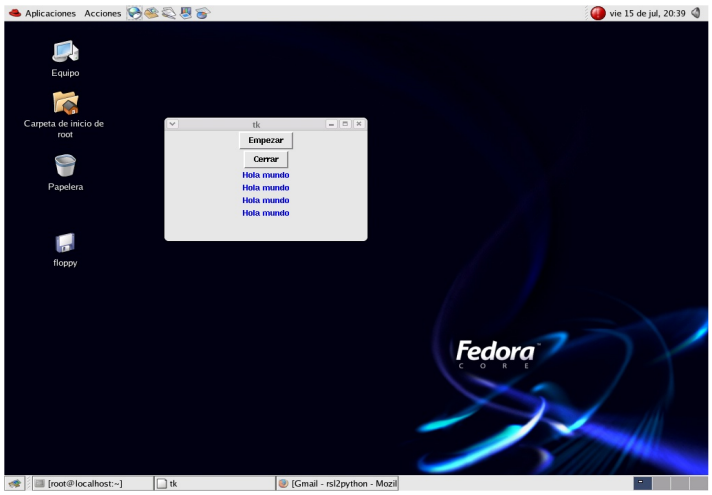
la indentación no solo es por claridad, es necesaria para que pueda ejecutarse el programa, por eso se dice que python hace regla las buenas costumbres de programación.

Los widgets que se usaron para la creación de este programa son Label y Button, pero existen otros widgets que se pueden utilizar en una aplicaciónmas grande y con mas funcionalidades. Aunque aquí utilizamos Tkinter, se pueden utilizar otras herramientas para construir interfases gráficas para Python como GTK, wxWindows, Qt, entre otras.Para los amantes de la plataforma Java existe una implementación de Python llamada Jython que ejecuta Python sobre la máquina virtual de Java.

Python no se queda fuera de los CGI, se puede utilizar para ejecutar aplicaciones sobre el servidor. También se puede utilizar PSP que es la versión pythonesca del ASP para construir páginas web dinámicas.

¿Juegos escritos con Python?, por supuesto. El proyecto Pygame hace posible escribir juegos de manera muy sencilla.

Como podemos ver es bastante rápido, versátil y relativamente sencillo el desarrollo de aplicaciones en Python, no por nada empresas como Google o proyectos como Ubuntu, por mencionar algunos, lo estan utilizando para su desarrollo.



Referencias:

http://www.python.org
http://effbot.org
http://vex.net/parnassus/
http://greenteapress.com/thinkpython/
http://diveintopython.org
http://www.pygame.org

Seguridad SL
Policias de la red

Creando nuestro propio password cracker

Dr. Victor Salazar Vera
ksaver@ksavers.cjb.net

Antiguamente las contraseñas de los sistemas *nix se guardaban encriptadas en el archivo /etc/passwd, y cualquier usuario del sistema lícito u "ocasional" podía leer tal archivo, con las consecuencias de seguridad que ello suponía. Por aquellas épocas, el poder de procesamiento de las computadoras no era para nada como en las de hoy, y "crackear" una contraseña bien escogida era una tarea muy difícil para el usuario común...

Los tiempos han cambiado, los sistemas se han vuelto más seguros, más potentes por lo que este método de encriptación de contraseñas se ha vuelto obsoleto, pero aún hoy en día existen sistemas que usan este antiguo método de encriptado, aunque se van haciendo cada vez más escasos, y en sistemas modernos se usan variantes de MD5, acompañado casi siempre de "shadow" passowrds.

Un ejemplo de archivo /etc/passwd:

```
$cat /etc/passwd
root:x0MmCh104WpfI:0:0::/root:/bin/bash
bin:!:1:1:bin:/bin:
adm:!:3:4:adm:/var/log:nobody:x:99:99:nobody:/:
jsmith:7rw6tPeKns97A:1000:100:jsmith,,,:/home/j
smith:/bin/sh
hacker:S3eudhPFntltc:1001:100:hacker,,,:/home/h
acker:/bin/tcsh
louis:l4fALit8qmwxM:1002:100:louis,,,:/home/lou
is:/bin/bash
```

Prof. Ronald Rivest
Creador del MD5

Mmmh.. Tenemos un antiguo archivo de contraseñas, donde las contraseñas encriptadas se encuentran en el mismo archivo, a la vista de todos (Es la cadena de 13 caracteres que está entre los primeros y segundos ":")... Podemos hacer cosas interesantes con ésto...

Hay miles de password crackers por todas partes en la red... lo cual indica que es demasiado fácil crear este tipo de programas... o que hay mucha gente por ahí, sin mucho que hacer...

Pero, nuestro objetivo es crearnos uno para nosotros, no queremos usar el que creo un tal "C001-B14ckIc3" de no sé que oscuro rincón del planeta, no... eso no es para nosotros, nosotros nos podemos crar nuestro propio password cracker, así que, a ello.

Veamos: Sabemos que una vez encriptada la contraseña, NO existe forma de volver atrás, es decir, de desencriptarla, ya que se trata de un algoritmo de encriptación "asimétrico" (una variante de DES, para los más curiosos: http://es.wikipedia.org/wiki/DES)...

Lo que la mayoría de los programas usados para crackear este tipo de contraseñas hacen es encriptar una palabra, comparar el resultado de la encriptación con la contraseña que se desea crackear, y... Voilá! Es lo que se conoce comúnmente "Ataque por diccionario".

Por lo tanto, primero necesitaremos crear un "programa" que se encargue de encriptar alguna palabra que le digamos y nos devuelva la palabra, encriptada.

Aqui es donde entra Perl, eidtemos el siguiente fichero, (con cualquier editor de texto "plano", como vi, emacs, pico, etc):

```
**********************************************
#!/usr/bin/perl
# llamaremos a este pedazo de código "mkpass.pl"
if (!@ARGV)
{
print "Usage: $0 [salt] [word] ";
exit;
}
else
{
$crypted=crypt $ARGV[1],$ARGV[0];   #He aquí el
meollo
print "$crypted ";
}
**********************************************
```

Le damos permisos de ejecución:

```
$ chmod +x mkpass.pl
```

Y lo ejecutamos, sin argumentos:

```
$./mkpass.pl
Usage: ./mkpass.pl [salt] [word]
```

Ok, nos dice que necesitamos pasarle una palabra ("word"), que sera la que queremos encriptar, y un tal "salt"... Que diablos es esto de "Salt"?? Pues las contraseñas en *nix se encriptan usando un "Salt" (algunos lo traducen como "Sal", otros como "Salto", nosotros lo dejaremos como "Salt", para no confundirnos :), que son dos caracteres, generados de manera aleatoria por el sistema al momento de crear una cuenta de usuario y encriptar la contraseña correspondiente.

Como cada "Salt" será diferente y aleatorio para cada password, TODAS las contraseñas encriptadas serán distintas, aunque en texto plano fueran idénticas. Veamos un ejemplo:

```
$./mkpass.pl  SS  password
SSi1glKdQJnsg
$
```

Nos devuelve una cadena de 13 carateres, que es la contraseña, encriptada igual que en cualquier sistema *nix (antiguo claro, ya comentamos que ahora se usa MD5, que es más... seguro?)

Si observan, los 2 primeros caracteres de la cadena devuelta, corresponde al "Salt", en este caso "SS", que le pasamos como parámetro a nustro script perl (ahora usamos "SS", pero puede ser lo que nosotros queramos, por el momento). Veamos otro ejemplo, con un "Salt" diferente:

```
$./mkpass.pl  x8  password
x8D96j0w3RzYE
$
```

Ok, observamos cómo para la misma contraseña, nos devuelve una cadena totalmente distinta! Precisamente para eso nos sirve el "Salt".

Ahora, observemos que en el archivo de contraseñas que vimos arriba, todas las cadenas que corresponden a la contraseña encriptada son diferentes, y los 2 primeros caracteres de cada una también son diferentes (x0, 7r, S3, l4). Tenemos entonces, en el archivo 4 contraseñas, y 4 "Salts" diferentes.

Esta información será necesaria para la creación nuestro "password cracker", ya que debemos encriptar las palabras en texto "claro" de nuestro diccionario con el "Salt" de la contraseña que queremos crackear, ya que si no, jamás funcionará.

Analizando el problema, observamos que:

Tenemos una lista de contraseñas encriptadas, todas diferentes, cada una usando el mismo sistema de encriptacion, pero con "Salts" diferentes. Por otra parte, tenemos una lista de palabras (diccionario) que usaremos para comprobar cada una de las contraseñas encriptadas (Si no tenemos uno, lo creamos, o descargamos uno de la web, por ejemplo desde: http://www.geocities.com/ksaversoft/download/DICT1.txt)

El planteamiento que debemos hacer, para crear nuestro programa será algo como ésto:

"Encripta una palabra en texto claro del diccionario "X", usando el "Salt" de la contraseña a crackear, compara el resultado con la contraseña encriptada del archivo "Y", si son iguales, entonces informa con un mensaje en pantalla, luego si hay más contraseñas en el archivo "Y", pasa a la siguiente...".

De manera esquemática, seria algo como ésto:

Obviamente, aquí nos estamos saltando algunos pasos importantes, como leer los respectivos archivos, de contraseñas encriptadas y el diccionario, pero damos por echo que se entiende.

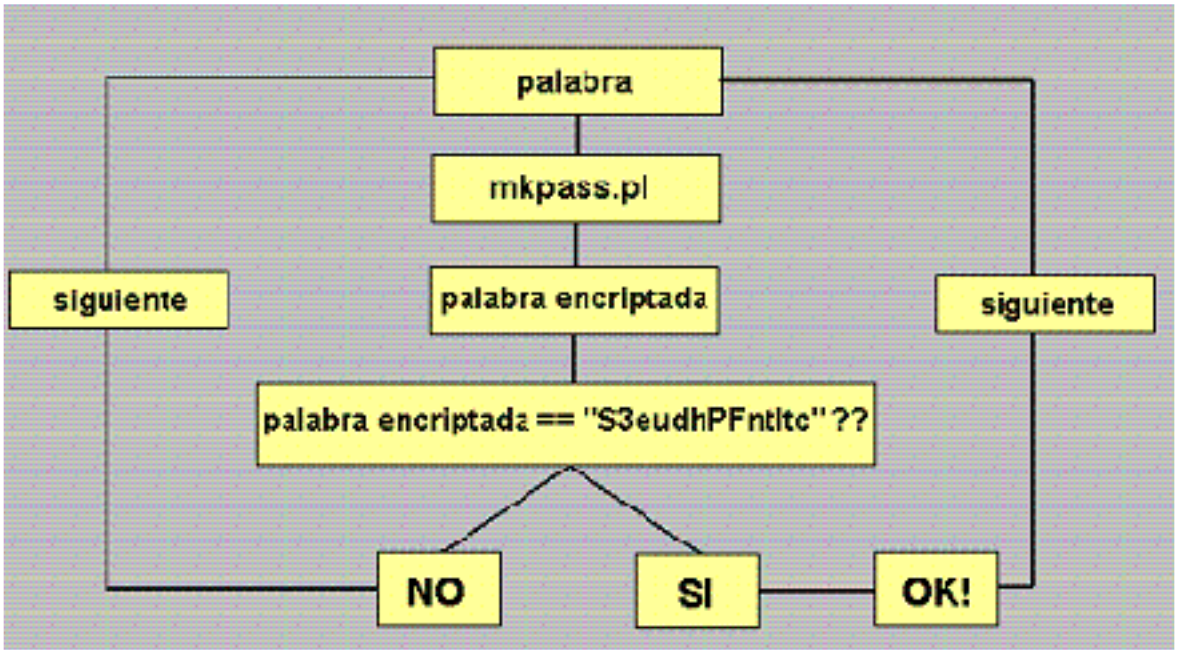
Veamos el procedimiento, paso a paso:

Creamos una lista con las contraseñas que queremos crackear:

```
$cat> pass.txt
x0MmCh104WpfI
7rw6tPeKns97A
S3eudhPFntltc
l4fALit8qmwxM
[ctrl+d]
$
```

Y tenemos un diccionario, al que llamaremos por ejemplo "dict.txt":

```
$cat dict.txt
...
root
admin
hacker
h4x0r
super
sex
...
$
```



Esquema del script

Ahora, para la automatización de los pasos de encriptación y comparación, usaremos un script Bash, que crearemos con cualquier editor de texto "plano", y que usará al script mkpass.pl que creamos antes:

```
**************************
#!/bin/bash

#Abrimos el archivo de contraseñas encriptadas, y asignamos cada línea de éste a la variable PASS,

#con un bucle for.
#El primer parámetro que el programa recibirá ($1), por lo tanto, será el archivo de passwords

#encriptados que deseamos comprobar.

for PASS in `cat $1`
do
#Que en cristiano sería algo como: para cada línea que resulte de un `cat $1`, hacer esto...

#(observar las comillas invertidas, que significa que el comando "cat $1" se ejecutará, y

#la variable "PASS" contendrá una línea del resultado en esa "vuelta" del for...)

SALT=${PASS:0:2}

#Esta línea asigna un "Substring" o subcadena de la variable $PASS, que va del caracter

#número cero al dos, quedando entonces con los primeros 2 caracteres del password

#encriptado o "Salt" lo que nos servirá para encriptar las palabras de nuestro diccionario.

#Abrimos un segundo loop for, con el segundo archivo, que corresponderá al diccionario de palabras

#en texto claro y lo recorremos con un "cat", asignando en cada vuelta del for una línea de dicho

#archivo a la variable "TEXT".

#Note que debemos "Anidar" este segundo "for", para que se compruebe una palabra del diccionario

#por cada password existente en el archivo a comprobar...


for TEXT in `cat $2`
do

KRYPTED=`./mkpass.pl $SALT $TEXT`

#En esta  línea, llamamos al script perl que hicimos antes, mkpass.pl
#para que encripte el texto contenido en la variable $TEXT con el "Salt"
#de la contraseña contenida en la variable $PASS.

if [ $KRYPTED = $PASS ]
then
echo -e "[+] $PASS --> $TEXT"
fi
#Aquí hacemos la comparación de las cadenas: "si la cadena contenida en la variable $KRYPTED
#(que es lo que nosotros encriptamos) es igual que la cadena contenida en $PASS (que es la

#contraseña que queremos obtener), entonces escribe en pantalla un mensaje que contenga

#la contraseña encriptada y el texto en claro al que equivale (obtenido de nuestro diccionario)..."

done
done
#Cerramos el primer y segundo bucles "for", con "done" (hecho).
**************************
```

Le podemos agregar más funcionalidades, depende que tan complejo lo queramos hacer, pero para un uso básico, con lo que tenemos funcionará bien. Lo que podríamos añadirle por lo pronto sería unas líneas al inicio para que nos compruebe los parámetros que le pasamos, nos muestre una ayuda en caso de no pasar parámetros, y termine de inmediato:

```
if [ -z $1 ] & [ -z $2 ]; then
echo -e "$0, by $USER. Sintaxis: $0
[password_file] [dictionary_file]"
exit
fi
```

Y ya está, guardamos el archivo, (podemos quitar los comentarios, que inician con un "#"), le damos permisos de ejecución con chmod, y lo probamos:

```
$chmod +x minicrack.sh
$./minicrack.sh
minicrack.sh  [password_file] [dictionary_file]
$
```

Nos aparece la "ayuda", lo ejecutamos como nos indica, con los archivos de password como primer parámetro, y el diccionario como segundo parámetro:

```
$./minicrack.sh  pass.txt  dict.txt
7rw6tPeKns97A --> 'secret'
S3eudhPFntltc --> 'h4x0r'
l4fALit8qmwxM --> 'superman'
...
$
```

Funciona!!, nos ha "crackeado" 3 de las 4 contraseñas de nuestro archivo (dependerá de el diccionario que usemos).

En esta ocasión usamos un script "externo", escrito en lenguaje Perl, pero podemos usar cualquier otro lenguaje, por ejemplo con un programa escrito en C aumentaríamos la velocidad y el rendimiento de nuestro programa, o hacerlo de una enorme complejidad, pero dejamos el campo abierto, por si alguien quiere hacerlo... y será material para otro mini-tutorial.

El código "final" de nuestro script quedaría más o menos así:

```
**************************
#!/bin/bash
# script para "crackear" passwords de *nix (DES)
# comparandolos con una lista de palabras
# (ataque por diccionario)
# by $USER 03.03.06

if [ -z $1 ] & [ -z $2 ]; then
echo -e "$0, by $USER. Sintaxis: $0 "
exit
else
for PASS in `cat $1`
do
SALT=${PASS:0:2}

for TEXT in `cat $2`
do
KRYPTED=`./mkpass.pl $SALT $TEXT`

if [ $KRYPTED = $PASS ]; then
echo "$PASS --> '$TEXT'"
fi
done
done
fi
********************
```

El Dr. Victor Salazar Vera es fundador y ex-integrante del RTM Security Group, participante del Festival Latinoamericano de Instalación de Software Libre y a echo varias traducciones de articulos en ingles para diferentes sitios.

Uno de sus proyectos mas importantes es Cracknix, un password cracker desarrollado en Perl que aplica mucho de los conocimientos expuestos en este articulo.

Gente SL

Los famosos del ratón y el teclado

El equipo de Ienjinia

Lic. Julio Acuña Carrillo
urkonn@gmail.com

En esta ocasión Revista SL tuvo la oportunidad de entrevistar a los desarrolladores Gerardo Horvilleur (GH), Elmer Garduño (EG) y Jorge Vargas (JV), los cuales forman un importante equipo de desarrollo en México, con proyectos libres. Pero bueno, ya los conoceremos mas durante la entrevista.

RSL-Nos podrían platicar quiénes son, a qué se dedican.

GH- ¡¿Quiénes somos y a que nos dedicamos?!, pues escribimos software, Elmer trabaja en una empresa que se llama México Analytica y Jorge y yo trabajamos por nuestra cuenta como consultores en desarrollo de Java y entre los tres en nuestros ratos libres hacemos software libre. Ahorita tenemos dos proyectos mas o menos importantes, uno es el jrMan que es una implementación del software de PIXAR , Photo Realistic RenderMan pero en versión software libre y escrito en Java y el otro es ienjinia que es una herramienta para ayudar a los adolescentes a aprender a escribir software dándoles un modelo simplificado de las computadoras y permitiéndoles que aprendan a escribir software haciendo videojuegos.

RSL- ¿Cómo surgió su interés por las computadoras?

JV– Por las computadoras!, hijoles pues la verdad es que mas que interés nació por necesidad, en la escuela nos dijeron: sabes que hay que aprender a usarlas, obviamente con paquetería, te estoy hablando de hace 16 o 17 años y ahi vi que era una herramienta que era padre usar, te entretenía, podías hacer cosas interesantes y ...

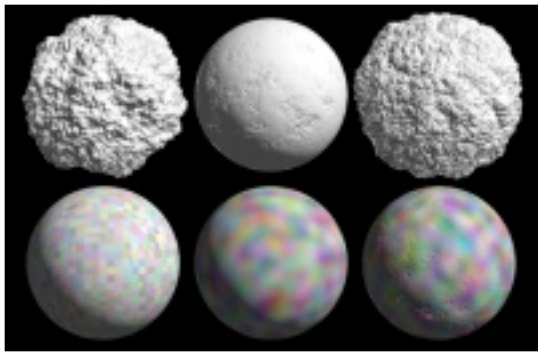


Imagen rendereada en 52 s usando un Duron a 1Ghz

... pues aunque no me dediqué a estudiar eso, pues me gustó y desde entonces me puse a hacer sistemitas, programitas desde DBase, Clipper, BASIC, hasta pasando por C, Visual BASIC y ahora Java.

EG- Yo tenía que hacer una animación en 3D Studio y ahi fue cuando empecé a usar la computadora, a descomponerla y volverla a componer, entre a la prepa cuando aprendí a usar la computadora y aprender un poquito lo que eran los lenguajes y asi entre a la universidad y después a trabajar y ahi a programar y me gusta, interesantes, padres las computadoras.

GH-Pues a mi siempre me llamaron la atención, desde muy chico, cuando empecé a aprender de computadoras fue cuando tenía 10 años, nada mas que en esa época todavía no existían las computadoras personales, entonces no tenía acceso a una computadora, lo que hacía era leer libros de lo que conseguía, de programación, de electrónica; empecé lo que es muy raro, yo mismo aprendiendo álgebra booleana cuando tenía 10 años, eso fue hace 34 años.

RSL-¿Y cuál fué su primer acercamiento al software libre?

GH- Por ahi de 1987, en una empresa en la que trabajaba y en la que también trabajaba mi hermano en la que vendíamos máquinas multiusuario con UNIX, en un viaje en el que fue mi hermano a Estados Unidos, regresó con unas cintas magnéticas de 9 tracks que le habían regalado llenas de software que se distribuía gratis en Usenet y empezamos a jugar con eso. Después en 1990 pusimos una empresa que se dedicaba a vender workstations de Sun a Universidades y centros de investigación y empezamos a distribuir con las workstations unos CDs que se llamaban Primetime Freeware que venían con mucho software que se distribuía en Internet y en Usenet y pues ahí empezamos a ver cosas interesantes, estaba el Perl, Emacs, el GCC, y empezamos a jugar con eso, a compilarlo a instalarlo, a jugar y a ver qué había...

JV– Bueno realmente creo que fue cuando en la universidad un amigo me dijo: “Oye hay algo que sirve y que se llama Linux y ya no tenemos que pelarnos con el MS-DOS”, entonces me empezó a informar porque yo no tenía PC en ese entonces y cuando ibamos a su casa probábamos lo que había ahí, un poquito de ver qué eran las herramientas de awk, él se metió un poco mas y empezaba a ver que los drivers para las tarjetas de audio y demás, porque nos interesaba la música y de ahí yo me salté y me seguí con Linux, yo creo que ese fue el acercamiento con software libre.

EG- Nosotros teníamos en la empresa una plataforma toda en Windows pero nos causaba muchos problemas y para desarrollar sobre todo era una pesadilla, Gerardo antes trabajaba en esa empresa, era mi jefe y fue ahi donde instalé Linux y fué una maravilla, desde entonces desarrollar se volvió muy amigable y bueno ahora en la empresa utilizamos muchisimo software libre como plataforma de soporte en toda la parte de servidor.

RSL-¿Cuánto tiempo llevan trabajando juntos?
GH- Desde el año 2000 mas o menos

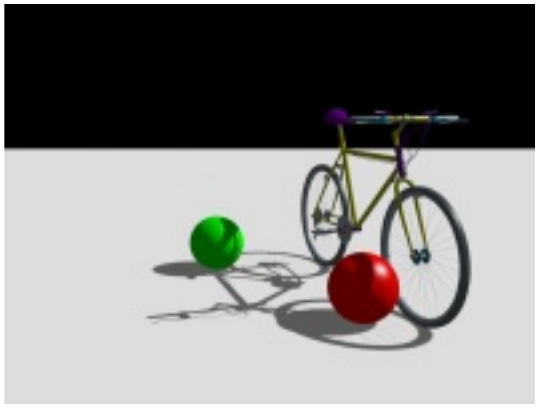


Imagen rendereada en 11 min usando un Duron a 1Ghz

RSL- Bueno y ¿por qué Java?

GH- Por muchas razones, básicamente en lo que trabajamos los tres es desarrollando software en Java, no andamos metidos en lo que anda la mayoría de la gente, con los cjb's y todo eso, aunque ocasionalmente para un cliente hacemos algo así. Java es un lenguaje bastante interesante donde muchas personas no le han entendido, de hecho lo interesante de Java no es lenguaje, el lenguaje no tiene nada en especial, tiene una sintaxis que está mas o menos práctica pero ya eso es todo, la parte interesante de Java es su máquina virtual, que realmente hay que exparimentarla y vivirla para darse cuenta todo lo que tiene que es una inovación muy grande; mucha gente, como muchas de las partes ya existían en otras como por ejemplo los intérpretes de código P para Pascal, el garbage collection ya está en otros lenguajes, creen que ya conocen lo que tiene, no pero tiene un modelo de cómo de cómo implementar totalmente diferente que da una flexibilidad tremenda, es mas, de hecho aunque usamos Java y la máquina virtual de Java, no siempre usamos el lenguaje Java, para varios proyectos hemos usado el Jython, que es una implementación de Python sobre la máquina virtual, en varios otros hemos estado usando Kawa que es Scheme sobre la máquina virtual, por cierto hay otra cosa interesante, Scheme pues es un dialecto de Lisp, todo mundo anda con la idea de que como Lisp es interactivo es interpretado y pues no, en la práctica todas las operaciones son compiladas, el Kawa también es un compilador, entonces yo escribo mis expresiones en Lisp, ...

... el Kawa las compila a bytecodes de la máquina virtual de Java y la máquina virtual de Java, igual hay que tener una buena versión de Hotspot, lo traduce ya a las partes críticas a lenguaje máquina a la hora de la ejecución.

¡Esta imagen tiene unicamente 3 primitivas!. Todo el detalle esta echo con mapa de textura y mapeo de desplazamiento. Fue rendereada en 3min 36 s usando un Athlon a 650.

RSL-Hace aproximadamente un par de años surgió el jrMan, ¿qué nos pueden platicar acerca de él?

EG- Es un proyecto muy interesante, nosotros fuimos los primeros impresionados cuando lo vimos funcionar. Estabamos trabajando en otro proyecto en el que necesitábamos hacer un lenguaje que compilara algunas cosas genéricas, unos vectores genéricos y ahí empezamos a platicar de que hacía falta un lenguaje donde podías multiplicar y hacer cosas con vectores y matrices genericas y empezamos a desarrollarlo, nos juntamos los martes en la noche a trabajar en una cosa que estaba muy interesante, es un proyecto donde hay de todo, matemáticas, programación muy avanzada, compiladores, un proyecto muy llamativo.

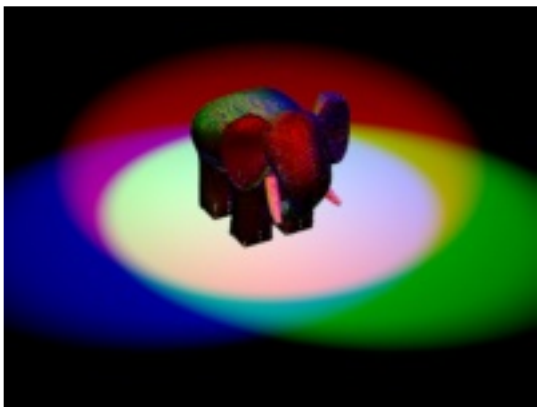
GH-Trabajamos mucho el primer año, todo el 2003, el 2004 estuvimos haciendo algunas cosas que no han quedado en ningún release pero que se quedaron olvidados porque tuvimos mucho trabajo y sacamos el proyecto de ienjinia y ahorita hace un par de semanas tuvimos una junta en la que vimos como retomar y seguir el desarrollo de jrMan porque no hicimos ningún release en el 2004, el último release, la versión 0.3, es de diciembre del 2003, entonces hay cosas nuevas que estan en el CVS pero nos han faltado algunos detallitos y no hemos sacado la versión 0.4, entonces estamos trabajando en la versión 0.4 y en la 0.5; de hecho ienjinia, la idea se nos ocurrió en octubre del 2004, en noviembre sacamos el release 0.1, despues para diciembre sacamos el 0.3, y en principios de este año es cuando sacamos el release 0.4 y cuando le dije a mi hijo de diez años que habíamos sacado el release 0.4 de ienjinia me regañó, me dijo: “¿y qué pasó con el release 0.4 de jrMan?”. Ahora algo muy interesante es que mucha gente anda con la idea de que Java es un lenguaje que es lento a la hora de la ejecución, probablemente han estado trabajando en algo de enterprise con los cjb's que a mi no me convencen mucho y no han visto el lenguaje y la máquina virtual en sí. Para hacer render se necesita CPU a morir, se necesita algo eficiente, de hecho hay un proyecto que también es una implementación de Renderman usando el mismo algoritmo REYES que es el que usa PIXAR que es software libre que se llama Aqsis, llevan mucho mas tiempo que nosotros, como 5 años desde que empezaron, ellos son un equipo como de 26 personas o algo asi, escrito en C++, pues cada vez que hemos probado lo que tenemos de jrMan contra su software en C++, típicamente jrMan va de 2 a 4 veces mas rápido.

JV- Además es un proyecto que llama mucho la atención porque estas viendo el resultado de una gran intensidad de cómputo, cuando les platicas que se generan tantos millones de polígonos pues le sorprende mucho a la gente y lo está viendo porque lo está dibujando y pues vé que a partir de una definición y un texto se generan imagenes muy padres, entonces yo creo que es parte del atractivo que tiene el proyecto y que cuando les dices que está en Java pues muchos no lo creían, decían: “¿cómo en Java?, ¿tan rápido?”

EG- En muchos de los sitios donde está dicen: In JAVA! con admiraciones y con mayúsculas, no lo pueden creer.

GH- Ahora hay otra cosa interesante, la gente que está metida en gráficos hay como dos bandos, las que hacen render estilo como el RenderMan y los que hacen gráficas en tiempo real con OpenGL o DirectX, últimamente ha sido impresionante la velocidad a la que se han ido acelerando los aceleradores gráficos el poder de cómputo que tienen es tremendo y cada año andan diciendo: "el año próximo vamos a poder hacer RenderMan en tiempo real", no tienen idea de lo que estan hablando, por mass rápidos que sean, hay una gran diferencia entre el tipo de imágenes que se generan en tiempo real y lo que hace un render como RenderMan; es decir a la hora que se hace el render para un cuadro de película como Buscando a Nemo o Los Increíbles usando máquinas rápidas, estan hablando de que van de unas cuantas horas a mas de 24 horas de render pero lo mas impresionante es que manejan las gráficas a un nivel de abstracción mucho mas alto, no andan manejando polígonos, manejan NURBS, subdivision surfaces que permiten representar de manera muy compacta la geometría de detalles también tienen los shaders procedurales, que las nuevas tarjetas aceleradoras tienen sus shaders y sus shading language que se ejecutan en los pixel shaders pero estan limitados a los métodos, a las rutinas que pueden meter, un shader que se usa para una película puede tener varios miles de líneas, no cabe en un acelerador gráfico, y aún compactando todo eso, el formato RIB de RenderMan permite guardar la información muy compacta, una imágen típica de película un RIB, la descripción geométrica es de 1 a 2 Gigas mas a parte 2 o 3 GB de texturas y a parte lo sacan en una resolución mucho mayor a cualquier pantalla de televisión o monitor de computadora, entonces algo interesante sobre eso es que el chiste de RenderMan es, el algoritmo REYES que usan es llegar a esa calidad, digamos ahí el objetivo es calidad y después mayor velocidad, mientras que los aceleradores graficos es primero velocidad y despúes la mejor calidad que logremos a por lo menos de 30 a 60 cuadros por segundo...

... entonces, las imágenes de demo que vienen con jrMan y las que se ven en nuestro website son pequeñas pruebas, nosotros para probar jrMan hacemos imágenes mucho mas complejas, hemos hecho imágenes que es gigantesco la cantidad de poliígonos que se renderean; una manera en la funciona REYES es que divide toda la superficie obtenida en micropolígonos, hemos generado imágenes que tienen del orden de arriba de mil millones de micropolígonos para probar que realmente funciona, es decir, no es un juguetito.



Escena rendereada en 3min 30 s con un Duron a 1Ghz

RSL- ¿Creen que en el futuro alguna empresa tome su proyecto para hacer, no sé, lo que está haciendo por ejemplo PIXAR actualmente?

GH- Pues a mi me agradaría mucho, aunque estamos un poco temprano como para eso, yo creo que si seguimos trabajando talvez para mediados del 2006 tendremos algo que ya se use para renderear en serio, que tiene todos los features de RenderMan para poder renderear y de eso, dándole a la gente tiempo para que trabaje en él y que lo afinemos, se podría usar. Bueno ahora una ventaja que tiene es que el Photo Realistic RenderMan de PIXAR aun ahorita con sus nuevos precios, sigue costando 3mil dólares o algo así por CPU, el render para una película es de unos cuantos miles de CPU, potencialmente alguien se podría ahorrar unos cuantos millones de dólares en licencias de software, otra ventaja es que los fuentes están disponibles sin restricciones, los fuentes de RenderMan, por ejemplo Industrial Light & Magic los tiene, pero esta muy limitado quien puede verlos y meterles mano.

JV- Sería terminarlo para que tuviera todos los features y entonces yo creo que si la gente se animaría, porque en el estado en el que está pues faltan cosas muy importantes como motion blur y cosas que le dan la parte realística al rendereo, ahorita podrían hacer imágenes fijas que saldrían bastante decentes, pero ya algo en movimiento sería algo muy complicado, bueno realmente es imposible

GH- Bueno saldría el movimiento pero se vería muy salteado porque le falta el motion blur, quizá ahorita los shaders hay que escribirlos en Java

RSL- Me imagino que ienjinia surgió de la inquietud de hacer una consola como el Commodore 64 o el Apple][, ¿alguien de ustedes tuvo una Commodore 64?

EG- Gerardo ha tenido todas las consolas de la historia (risas)

GH- Yo programé bastante en Apple][, programando en Pascal y en Assembler 6502, en Atari 800 accesando al hardware de gráficas, me inspiré bastante en el hardware de gráficas del Atari 800 para algunas cosas, tomé algunas cosas de la arquitectura de la Commodore 64, que esa no la usé tanto, si llegué a hacer algunas cuantas cosas en Assembler marcando los chips, nos basamos un poco en la información que hay de cómo era el Nintendo original, el ColecoVision que tenía un chip de Texas Instruments, el 9918, el mecanismo de sprites y de tiles que manejamos es muy parecido, es una mezcla de varios de esos, hicimos como una abstracción, diseñamos el chip virtual y lo simulamos, por ejemplo eso está interesante, en la consola virtual de ienjinia y en el devkit estamos en software, en Java, simulando el chip y el chip esta dibujando en la pantalla con todo, con sprites 25 veces por segundo, simula la interrupción de vertical blank para poder llamar a una rutina y además estamos generando el audio en 4 canales independientes y también lo estamos sintetizando en software, hay un thread que está simulando el hardware de video y otro thread que está simulando el audio y aún en una máquina no muy reciente funciona sin problemas y está escrito en Java.

RSL- ¿Qué mas nos pueden platicar acerca de ienjinia?

JV- Bueno yo creo que ienjinia nace de la necesidad de generar gente que sepa programar, que es un gran problema que tenemos en México porque nos hemos dado cuenta, principalmente en nuestros trabajos, que llegas a una empresa que se dedica a desarrollar software y la verdad es que muy pocos realmente saben programar, que su nivel de abstracción, de todo lo que tu quieras es muy bajo, no pueden sacar software de calidad ni algoritmos correctos y el esfuerzo que se hizo con ienjinia fue precisamente darles una manera amigable, que se pudiera interesar la gente jóven y que obviamente si sabes programar un juego de video pues puedes programar todo.

GH- También otra cosa es que hemos visto que hay una crisis en desarrollo de software, pero casi todo lo que hacen aqui es que tratan de arreglarlo con soluciones administrativas, metodologías y vamos ahora a hacer el diseño yo creo que primero deberían enfocarse en formar programadores que sepan programar; esto de saber programar es como saber leer y escribir, se supone que ya un porcentaje muy alto de la población en México sabe leer y escribir, pero eso quiere decir que pueden escribir un recado sin demasiadas faltas de ortografía y que pueden leer el nombre de una calle, los pones a leer un texto de una página, deja tu un libro, nada mas una página y no son capaces de entender lo que leyeron, bueno pues la gran mayoría de los programadores están así, gente que se gana la vida como programador, son programadores profesionales, pues saben hacer un pequeño loop o unos if's pero de ahí no pasan, no tienen idea de lo que son estructuras de datos, base algoritmos o diferentes lenguajes...

... entonces hay que empezar a tener gente que sepa escribir software y también a empezar a propagar la idea de que un programador no es un obrero, es decir, ese concepto de fábricas de software de que hay alguien que lo diseñe y luego se lo pasan a los programadores que lo fabriquen, no funciona así, no puedes separar la programación del diseño eso es totalmente falso y hay que empezar a matar esa idea que causa mucho daño.

EG- Y otra cosa en la que pensamos cuando lo diseñamos, pensando en cómo le ibamos a hacer para que fuera accesible para los adolescentes, entonces estuvimos buscando algunas herramientas que había por ahí para la enseñanza y acabamos pensando que lo mejor era escribir la propia y el enfoque que le dimos es que fuera superatractivo para los adolescentes, para que cualquiera, yo también quise, aprender a hacer un video juego. Entonces por un lado la motivación es esa, de que aprendan y por otro lado que sean juegos de video para que sea superatractivo y además ves los resultados, te pones enfrente del devkit y le pones un comando y estas haciendo que la pantalla pinte lo que tu dijiste y eso es superatractivo porque te retroalimenta luego luego y eso da muy buen resultado.

GH- Nos ha dado resultados muy interesantes, está enfocado a adolescentes, lo hemos probado con varios niños, también en adolescentes, les ha gustado mucho y con el ITAM hace unos meses, la semana después de semana santa, en la última semana de marzo en el ITAM les regalaron a algunos alumnos de preparatoria que estaban interesados, gratis un curso que dimos nosotros de ienjinia, que fueron cuatro tardes, y de prácticamente de cero, en cuatro tardes ya estuvieron programando un video juego bastante interesante, un programa ya de cierto tamaño y que lo estuvimos haciendo junto con ellos, fueron aprendiendo de programación, en el ITAM les gustó bastante el resultado y empezando en agosto, los alumnos que entren al ITAM a una carrera de computación, su primer materia de programación y lo primero que van a ver va a ser con ienjinia.

RSL- ¿Y cómo fue concebido el ipl, el lenguaje de ienjinia?

EG- De muchas noches en que Gerardo trataba de arreglar un error... (risas)

GH- Empezamos usando el BeanShell que es un lenguaje de scripting para Java, que es realmente un intérprete de Java, tu le pudes poner código en Java y te lo interpreta, pero tiene la opción de un modo simplificado en el cual no necesitas declarar los tipos de las variables y argumentos, entonces es como si fuera una especie de BASIC o Python en cuanto a eso, entonces dijimos: esto esta perfecto, para un curso de introducción no importan los tipos, la parte de objetos no nos interesa tanto, nos interesan los conceptos básicos de programación, el BeanShell con su sintaxis simplificada funciona y para la primer versión del devkit lo estuvimos usando con BeanShell y a la hora en que empezamos a preparar material de cursos, pues ya tenemos la lección uno, dos y tres, para la lección cuatro empezamos a introducir arreglos y al estar haciendo las lecciones me topé con un bug que no entendía porque no jalaba el programa, hasta que me dí cuenta de que por ahí como que se transparentaba la parte de como emular Java, entonces había cosas que la semántica no era muy clara y dijimos: esto no sirve para los estudiantes, estuvimos volviendo a analizar lenguajes, volvimos a analizar que si usamos Jython, o el Jrubi, hasta pensamos en Scheme, dijimos: esto no lo van a querer usar los profesores ytotal, dijimos: bueno la opción mas sencilla es el lenguaje tal como lo estamos usando en BeanShell, la misma sintaxis, nada mas que haga lo que debería de hacer a nivel básico...

... no que trate de emular Java si no lo que esperamos y pues fue implementar el ipl, la primera versión del ipl fue una semana de trabajo mas o menos, después otra semana integrarlo con la consola, el devkit, corregirle unos bugs y ya teniendolo fuimos poniéndole otras extensiones, tiene cosas avanzadas, una variable puede contener una lambda estilo Lisp, se puede pasar un método como argumento a otro método, puedes hacer un método que regrese nuevos métodos; que un estudiante que está empezando no necesita saber eso, no le afecta, pero a la hora que empiece a adelantar mas, no le va a quedar corto el lenguaje, por cierto algo que hicimos en el site de ienjinia, para poder ir haciendo la documentación mas rápido, pasamos a usar un wiki, quitamos nuestras páginas estáticas de html y lo pasamos todo a wiki, estamos usando MediaWiki, el mismo de la wikipedia, pusimos lo mismo que teníamos del website, los FAQ's y todo eso, lo que tenemos preparado del material de cursos, las tres primeras lecciones, casi la cuatro, ya las pusimos online, tienen varios errores, siguen siendo detalles de las lecciones que estan todavía enfocados a BeanShell que tenemos que actualizar, pero la idea es estarla actualizando, el que le interese puede downloadear el software gratis, el que tenga el runtime de Java le va a funcionar, y están las lecciones online para que puedan ir aprendiendo, la idea es darles oportunidad a los que tengan potencial, darles acceso a la información, porque creo que es uno de los problemas que tenemos en México, no creo que la gente sea mas tonta o mas inteligente que en otros países, lo que se necesita es el acceso a la información y lo hicimos preocupándonos por México pero en cualquier lugar del mundo lo pueden ver, es decir el website está en inglés, es chistoso, y los cursos están en español.

EG- Los cursos están orientados a gente que no sabe nada, que van empezando y están en un lenguaje familiar, lo hicimos tratando de que sea lo mas amigable posible para que tu llegues, lo leas y puedas solo, aunque nadie te esté dando soporte, aprender a programar y a programar bien, porque también nos preocupamos por el estilo que es muy importante.

GH- Bueno estamos en contra de eso de te voy a enseñar a programar cómo deberías de programar bien con buen estilo, está mezclado en el curso, a medida que vamos explicando qué hacen las diferentes instrucciones, en qué consiste programar y que vamos haciendo un programa, les vamos explicando cómo conviene estructurarlo, que conviene ponerle nombres a los métodos y a las variables que signifiquen algo, vamos mezclando la explicación de qué es programar con las buenas prácticas, y nos gustaría pensar que cuando decimos buenas prácticas nos referimos realmente a prácticas que son buenas, no como muchos lugares en la industria que dicen buenas prácticas y se refieren a lo que hacen los demás, que en muchos casos no tienen nada de buenas prácticas.

JV- Y obviamente si está en Java puedes bajar la consola y correrla en lo que quieras, mientras tengas el Runtime pues correrá, igual si usas Linux pues ahi lo tienes, si usas Windows, en nuestras Mac funciona sin problemas.

GH- ahorita estamos usando Mac para desarrollo pero funciona perfecto en Windows, funciona perfecto en Linux.

EG- Ahorita una cosa que nos falta son los juegos para demostrar lo que se puede hacer, aunque es una consola que simula a las consolas de 8 bits, en cuanto a las gráficas, en la parte de cómputo usa el procesador de la computadora pero la parte gráfica se ve como una de 8 bits y los juegos se verán como los de 8 bits de aquella época pero también eso fue intencional, que aprendas a programar, no que te andes preocupando por que tus gráficos sean los mas padres, porque se perdería todo el tiempo, porque ahi te vas con los gráficos...

JV- y nunca será obsoleta, porque bien lo dice el logo fue creada obsoleta por diseño, si salen mejores máquinas no importa, si tienes el Runtime lo puedes correr y seguir desarrollando, no importa que tengas una máquina diferente, mas avanzada, siempre será compatible; entonces lo interesante de los cursos es que las definiciones están ocultas, lo típico que nos pasa siempre cuando estudiamos es: a ver me tengo que aprender todo este cuaderno...

... de definiciones acerca de estas cosas, no, dentro del curso estan todas las definiciones que se van necesitando en el orden en el que se vayan requiriendo pero de alguna manera el alumno ni cuenta se dá que ya se aprendió una definición, ni hay un renglón que me lo tengo que memorizar, no, no, no, simplemente lo aprendí porque lo necesité.

GH- y lo estas ocupando para desarrollar un juego.

RSL- ¿En qué forma se puede colaborar con el proyecto?

GH- En muchas cosas, pueden escribir juegos, los podemos poner alli para que se puedan downloadear, estamos esperando a que haya gente que nada mas use la consola virtual y que downloadee juegos, entre mas pues habrá mas habrá otros alicientes y diferentes motivos para hacer juegos, les va a dar gusto que la gente juegue sus juegos. Pusimos toda la documentación en Wiki para que si alguien quiere hacer una lección o poner en español la parte que ahorita tenemos en inglés o ayudar a documentar mas, la idea es que todo el mundo pueda participar alli.

EG- Otra cosa con las lecciones es que las pensamos hasta la lección cuatro...

GH- bueno como hasta la lección seis como una especie de tronco común y que se bifurquen, por ejemplo que hay unas lecciones que se van mas a cómo manipulas a nivel hardware el chip, porque hay unos métodos de alto nivel que te permiten manipular el hardware de audio pero hay que empezar las lecciones que expliquen la notación hexadecimal, el sistema binario, el directorio de memoria y que vayan directo a cómo usar el chip y sacarle provecho, otras que hablen mas de estructuras de datos, otras sobre algoritmos de inteligencia artificial para los juegos, otras por ejemplo sobre la física, estaría muy divertido que alguien hiciera un pinball, que las gráficas lo permiten y todo pero no está nada trivial.

EG- Y está abierto a que si alguien tiene conocimientos acerca de alguna rama en particular, pues está abierto a que tome alguno de los tracks de una rama y que nos ayude con eso.

JV- Y bueno es opensource entonces si alguien quiere colaborar y seguir ampliando lo que tenemos, que lo veo un poco reducido de alguna manera porque va a llegar un momento en que la consola va a estar terminada, pero obviamente la intención es enseñar pues a lo mejor que hay mejores métodos, que hay mejores ejemplos.

GH- Alguien con un curso de compiladores puede bajar los fuentes del ipl y ahi estan los fuentes completos del compilador, porque el ipl realmente compila a una representación interna a un código intermedio que es el que después ya interpreta pero hay un compilador completo.

RSL- ¿Creen que haga falta difusión del software libre en general?

JV- Como concepto yo creo que no...

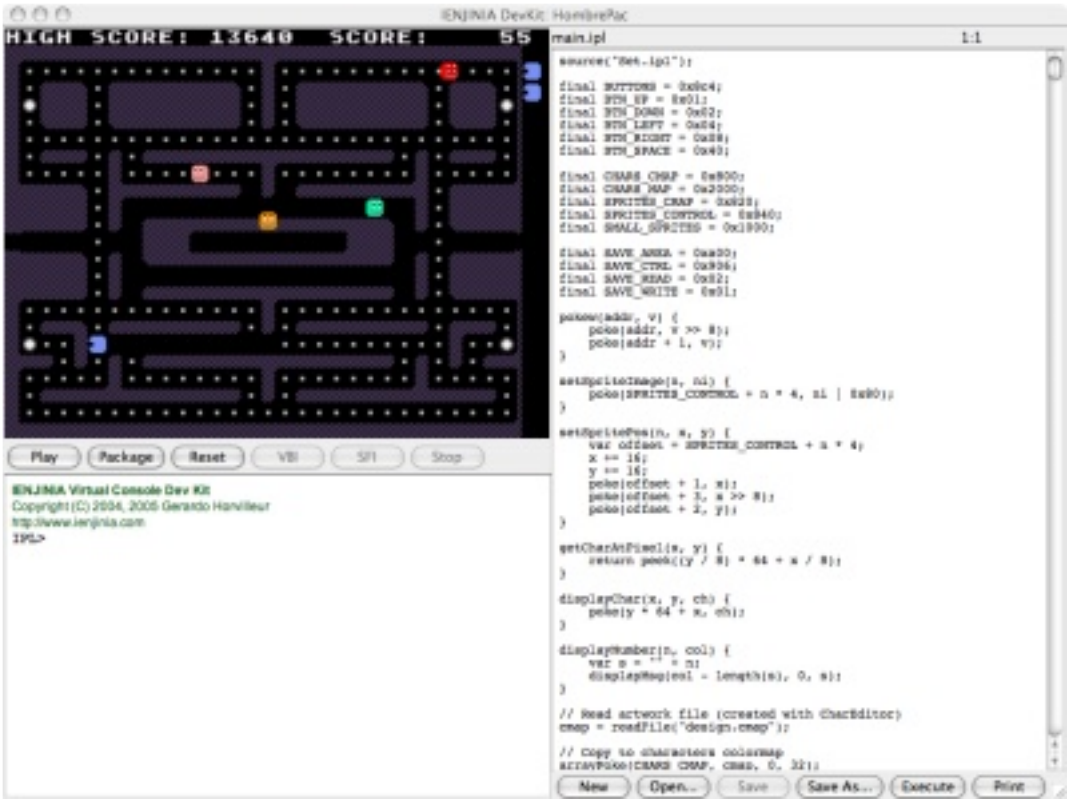
EG- Bueno depende de los niveles, entre los programadores si hay una buena difusión, bueno entre algunos...

GH- yo creo que si, que hay que difundirlo mas por un lado, porque no todos saben de él, segundo, hay que hacer entender que software libre no es igual a software gratis, que gratis es un punto importante, interesante y muy padre pero no es el único, es decir a los que quieren aprender a programar bien, aparte de usar ienjinia se me hace un buen lugar para empezar, pero no se pueden quedar toda su vida en el ipl, conviene que aprendan varios otros lenguajes que downloadeen otros proyectos, que los compilen, que traten de modificarlos, que vean cómo están escritos, ahi pueden aprender mucho sobre ingeniería de software, por ejemplo prácticas que aquí en México prácticamente nadie usa, como usar control de versiones, usar builds automáticos, ya sea con make si están en C, o usando el Ant si están en Java o el asdf si están usando CommonLisp que vayan viendo los conceptos de testing

EG- que no sólo es gratis, que vayan viendo el código de los demás, de gente que sabe mucho de programar, te ayuda a aprender, eso es muy importante y además si le haces cambios como dice Gerardo pues ya estás mucho mas avanzado, pero es muy importante que en proyectos enormes como todo lo de Apache que puedes ver el código y es un software muy bien hecho.

JV- Quizás, ya reflexionando, la difusión a lo mejor va hacia la parte negocios, porque hay mucha gente que todavía le dá miedo o no se anima a usar software libre porque piensa que tiene vicios, piensa que no hay un respaldo, hay una especie de “No, software libre no, porque mi empresa es de misión crítica” o “yo quiero que todo funcione bien y el software libre no me lo garantiza”, que es la creencia que tiene alguna gente.

GH- De hecho el software libre, hay de todo, igual que el software comercial, al menos ahi tengo la opción de no estar amarrado a un proveedor, existe la opción de arreglarlo, no quiere decir que sea trivial, pero tengo la opción de arreglarlo si el proveedor no me responde y también muchas veces no quieren usar software libre porque no hay alguien que se haga responsable, pues la próxima vez que instalen Windows en una máquina que vean la licencia, Microsoft tampoco se hace responsable de los problemas, es decir, yo puedo tener una empresa que quiebre porque algún software comercial no jaló y la licencia dice que ellos no se hacen responsables por lo que pase si el software no jala, entonces allí no existe ninguna diferencia. Pero yo creo que sería mas importante, hay mucho software que se desarrolla, por ejemplo una empresa de repente por alguna razón se dá cuenta que ninguno de los sistemas de nómina que existen le sirven, entonces tiene que estar desarrollando su propio sistema de nómina, que se den cuenta que les conviene publicarlo como software libre, lo mas probable que pase es que no pase nada, pero si por suerte otras empresas lo empiezan a usar, muchas veces dicen para que lo voy a hacer yo, para qué lo que yo gasté para darles ventaja a otras empresas que tengan un software de nómina; pero que se den cuenta de que entre mas personas lo esten usando, van a repartirse el costo de mantenimiento entre todos y típicamente en una empresa yo me dedico a hacer otra cosa, no a hacer software de nómina, entonces si yo invertí en hacerlo, lo pongo como software libre y otras empresas lo empiezan a usar porque les conviene y las mejoras que ellos le pongan, yo también las puedo usar y después somos ya veinte o treinta empresas que lo estamos usando, el costo de mantenimiento del software se dividió ya entre veinte, todos ganamos. Por otro lado, todo el software que se desarrolla para instituciones de gobierno, hacen licitaciones en las que gastan cada año decenas de millones de dólares en desarrollo de software, todo ese software debería tener una licencia de software libre, para empezar porque vemos cada dependencia de gobierno licitando para hacer lo mismo que está haciendo la vecina, hay mucho de lo que es la integración y customizar para la empresa, pero eso no es ingeniería de software, eso es ingeniería de negocios, pero lo que es el desarrollo del software en sí...



Consola Virtual Ienjinia

... todo lo están haciendo desde cero, todos; fuera software libre, para empezar si todas las dependencias de gobierno podrían hacer de una licitación pues decir: bueno vamos a usar este y lo aprovechamos; por otro lado eso es algo que se pagó con dinero público, eso le pertenece a toda la nación, eso debería de ser libre, de dónde se sacaron los fondos, no hay ninguna razón para que ese sea propietario.

RSL- Mencionaron otro proyecto además de jrMan y de ienjinia, ¿cuál es este proyecto?

GH- Tenemos un proyecto que lo hicimos poco, es mas, lo poco que hicimos relacionado con el proyecto nunca estuvo en CVS, lo que estuvo en CVS es otra cosa, que es el Java OWL que tiene que ver con el semantic web, se nos ocurrió que la manera de representación semántica en la web, podría ser muy bueno para representar información sobre software libre, metadata acerca del software, y en Java se presta a que está muy fácil de generar, entonces la idea es que a la hora que busque un software que haga algo o muchas veces no quiero todo el paquete, quiero una función, un método, un objeto que sirva para algo... si ese tiene buen metadata que se puede buscar en internet, me puede ayudar a localizarlo, por otro lado muchas veces digo aquí tengo una clase que me pasé a xml pero me espera algo que sea un imput source, pero yo lo que tengo es un archivo, entonces en una de esas como qué cadenita de clases tengo que armar para que mi xml parser lea mi archivo, es decir la información está puesta como metadata que la computadora puede procesar y encontrar en el proyecto cuál es la cadenita de clase que hay que armar. Entonces ese algún día lo seguiremos, de entrada estaría bien publicar lo que ya tenemos de software pero no está en nuestras prioridades, entre lo que hacemos del trabajo, entre el ienjinia que a la mera hora se volvieron, el ienjinia solito son tres programas, está la consola, el devkit y el chareditor, además de eso está la parte de lenguaje, que está el lenguaje ipl, y está la base de datos idb, entonces allí van cinco programas; por otro lado está jrMan que está el renderer y está el compilador de shaders que tenemos que sacar, entonces el de Java OWL lo vamos a dejar dormido ahorita un rato

JV- hasta que nos volvamos millonarios y podamos dedicarle

GH- podamos dedicarle fulltime a esto

EG- hay que trabajar además, no?

JV- hay que gastar dinero en otras cosas

RSL- ¿Algo más que quieran agregar?

GH- Se aceptan donativos (risas)

JV- Después de este esfuerzo se aceptan donativos, no nos han donado un centavo

Enlaces de interes:

http://www.ienjinia.com
http://www.jrman.org

La Revista SL agradece a Gerardo, Elmer y Jorge por las facilidades y su tiempo. Desde aqui un saludo y nuestros mejores deseos para sus proyectos.

Festival Latinoamericano de Instalación de Software Libre

Carlos Augusto Lozano Vargas
vendetta@zonartm.org

Comite Organizador FLISol Ciudad de México en el CECyT Juan de Dios Batiz - IPN

El pasado 25 de Marzo se llevó acabo una edición mas del Festival Latinoamericano de Instalación de Software Libre (FliSoL), evento que reunío a trece paises con decenas de ciudades que al mismo tiempo realizarón actividades de instalación de software libre, talleres, conferencias y hasta torneos.

México no se podia quedar atras y varias ciudades del país participarón, como fuerón la Ciudad de México, Guadalajara, Apizaco, Tijuana, Ciudad Nezahualcoyotl, Ecatepec, Cuautla, Toluca y Baja California.

Laboratorio de Inestigación y Desarrollo de Software Libre (LIDSOL-UNAM)

Entre las actividades que se realizarón en las diferentes sedes del país, estuvierón las fiestas de instalación, conferencias, platicas informales, talleres, muestras de proyectos e incluso torneos de juegos bajo licencias libres.

Grupo de Usuarios Linux del Estado de Hidalgo (GULEH)

De acuerdo a los registros en el sitio web www.flisolmexico.info se tuvierón 512 asistentes a nivel nacional, sin embargo la sede IPN de la Ciudad de México por si misma superó ese número de asistentes, por lo que se estima que alrededor del pais se pudieron tener de 1200 a 1500 asistentes, participando en las actividades del FLISOL 2006. Se espera el proximo año que mas grupos, patrocinadores y asistentes se unan a las celebraciones.

Se lo que hicierón el Chetumal pasado (1era parte)

Tec. Victor Hugo Cordova Madrid
vhcordova@linux-chetumal.org.mx

Permítanme presentarme, mi nombre es Víctor Córdova pero pues ViChO esta bien ;)

Voy a hablar de lo que fue el “1er Congreso de Software Libre Chetumal, Q.R. - Belice” ó CongresoFS (Frontera Sur) en su versión corta, el cual se llevo a cabo en la ciudad de Chetumal capital del estado de Quintana Roo, México los días 7, 8 y 9 de Abril del año en curso en las instalaciones del Teatro “Constituyentes del '74” (http://www.iqc.gob.mx/teatro_constituyentes_del_74.htm).

Este evento fue iniciativa de la Comunidad de usuarios Linux de Chetumal (CLC http://www.linux-chetumal.org.mx), organizado conjuntamente con la Comunidad Linux de Belice (http://www.linux.bz) y la Universidad de Quintana Roo (http://www.uqroo.mx).

No entraré mucho en detalles de la organización del evento, ya que de eso no se trata este artículo, pero pues a lo mejor en una siguiente edición de la RevistaSL podría escribir un How To “Realizando tu Congreso” =P ya que el vivir una experiencia como organizador te hace ver como se manejan y como se deben llevar a cabo las cosas, siempre hay detalles que se pasan por alto y un artículo así podría servir de ayuda para personas que se quieran aventar el compromiso de organizar un evento de tal magnitud por primera vez, pero bueno dejemos eso un poco de lado y entremos a lo que nos ocupa que es hablar de lo que fue este evento.

El Congreso dio inicio el día Viernes 7 de Abril de 2006, a las 09:00 horas comenzamos con lo que fue el registro de asistentes en el Teatro...

Poster oficial del CongresoFS

... ya que legaron grupos de otros lugares de la república como Veracruz y Tabasco, este proceso fue un poco más tardado de lo que teníamos pensado pero todo salió bien.

Alrededor de las 10:30 horas se realizó la inauguración del evento dando pie así a la primera conferencia “Free and Open Source Software” la cual impartieron nuestros compañeros de Linux Belice, tratando temas meramente básicos acerca de lo que es la filosofía del Open Source y Software Libre, para que nuestros asistentes puedan tener una idea concisa de los temas que se iban a tratar con el transcurso de los días.

Andres Vargas recibiendo el reconocimiento FS

La segunda platica fue alrededor del medio día y el título de la misma fue: “Montando un servidor web casero” que nuestro amigo Andres Vargas (zodman) se encargó de presentar. En esta conferencia se explicó como montar y configurar un servidor con los servicios Apache2, MySQL, PHP4, FTP y SSH, explicando las configuraciones más importantes de estos servicios (de otra forma una sola hora no hubiera sido suficiente para poderlo explicar), además se explicó que se manejó como servidor web “casero” porque se sustituyen algunos servicios que implican costo como lo es una IP estática y se habló de la configuración de dominios sin emplear bind, para luego aplicar dichas configuraciones al servidor casero, una muy buena platica por parte de zodman.

Al mismo tiempo que se llevaba a cabo la platica de zodman en el Teatro, en el mismo edificio, pero en el “Salón Latinoamericano” se impartía el taller de Ruby on Rails, el cual fue taller substituto del que teníamos programado (FirebirdSQL) ya que el ponente no pudo llegar, pero el Ing. Fabien Mannessier nos hizo el favor de dar el taller antes mencionado de Ruby on Rails, el cual duró alrededor de 2 horas, de las 12:00 a las 14:00 horas.

Fabien Mannessier impartiendo el taller de Ruby on Rails

A las 13:00 horas en el teatro se impartió la conferencia: “Montanto un servidor Samba+ldap para administrar usuarios en un laboratorio de computo (Caso en el ITSLP)”, por parte de nuestro buen compañero Hugo Franciso González Robledo...

esta plática trato acerca de un caso de éxito de la implementación de un servidor de archivos (samba) y Lightweight Directory Access Protocol (LDAP) para administrar los usuarios de los laboratorios de computo del Instituto Tecnológico de San Luis Potosí. Hugo manejo el tema con bastante soltura y creo que fue muy claro para todos los asistentes.

A las 14:00 horas dimos un descanso de 2 horas, para que tanto los asistentes como ponentes y Staff fueran a tomar sus alimentos, asearse y descansar un poco, a las 16:00 horas reanudamos las actividades, esto fue así los 3 días del evento.

Hugo González Robledo

A las 16:00 horas ya de regreso al Congreso en el Teatro el Ing. Fabien Mannessier Belva (catedrático del ITESM campos Toluca) impartió la conferencia: “Desarrollo Ágil de Aplicaciones Web con Ruby on Rails” la cual trato sobre un breve recorrido de los métodos de desarrollo de aplicaciones para llegar a una introducción de lo que son los métodos ágiles y luego se presentó Ruby y Ruby on Rails como las herramientas que permiten implantar de manera exitosa estos métodos de desarrollo. La verdad que esta plática tuvo un gran éxito ya que habían varias personas entre el público que se dedicaban al desarrollo de aplicaciones web, ya sea con PHP, PERL, etc., y quedaron encantados al ver las maravillas que podían hacer en un período muy corto de tiempo con Ruby on Rails. Aprovechó para mandarle un cordial saludo al Ing. Fabien y desearle el mayor de los éxitos en todos los proyectos que emprenda, esperamos con ansias esa traducción del libro de Ruby ;)

Al mismo tiempo se llevaba a cabo el taller: “Introducción a iptables – NAT y Filtrado de paquetes” impartido por el Ing. Edgar Jesús Sánchez Uicab en el cual se hizo una descripción breve del protocolo TCP, como se conforman los paquetes y algunos conceptos esenciales de redes.

Descripción del filtrado de paquetes en Linux, que parte del sistema lo maneja. Iptables, ¿que es iptables?, como interactúa con el núcleo y una explicación de cómo se usa. Filtrado de paquetes con iptables, explicación de cómo usar iptables para filtrar paquetes. NAT con iptables, ¿Qué es NAT?, y como funciona en Linux, se explicaría como manejar NAT con iptables en sus distintos tipos. El taller duró cerca de 2 horas.

A las 17:00 horas comenzó la conferencia: "Sistemas Expertos e Inteligencia Artificial" por parte de nuestros buenos amigos de IcenetX (Gaper, BRIO y Ayzax, un saludo para los tres). En esta plática se pretendió hacer una análisis profundo sobre lo que son los sistemas expertos, como funcionan y en que nos ayudaran en un futuro, como se están desarrollando y su estrecha relación con la Inteligencia artificial. De esta misma se derivara un análisis sobre el tema Inteligencia Artificial desde como trabaja hasta como se aplica. Definición de IA y Sistemas expertos Inferencias y lógica de desarrollo para IA Sistemas expertos al detalle Sistemas Inteligentes (software con IA) lenguajes de desarrollo (Prolog y Lisp).

En punto de las 18:00 horas en el Salón Latinoamericano daba comienzo el taller: "Transacciones a una BD MySql con Pear DB" a cargo de Adib Saavedra Bocanegra, miembro del GULITESCO de Coatzacoalcos, Veracruz.

A la misma hora pero en el auditorio del Teatro, Leo Utskot (Director General de Copyleft México) nos hacía el favor de exponer la conferencia: "¿Qué significa Web 2.0 y cómo podemos implementarlo con software libre?", se revisaron los orígenes del término, que significa realmente y revisaron las tecnologías atrás de este cambio exponiendo ejemplos concretos de implementaciones de tecnología Web 2.0 y explicaciones de como software libre fue utilizado para implementarlos.

Finalmente a las 19:00 horas se llevo a cabo la Conferencia Magistral del día titulada: "Presentación del proyecto Gekko", impartida por José Carlos Nieto Jarquin (xiam) y Josué Gutiérrez Hernández (BlacKNasH)...

Leo Utskot, director de Copyleft

desarrolladores de este interesante CMS. Se explicó ¿Qué es Gekkoware? ¿Qué es un CMS? Se mostraron las características del proyecto. Por que se eligió PHP y MySQL. Se habló de cómo inicio el proyecto Gekkoware y como ha sido el desarrollo del mismo a lo largo de sus versiones. La verdad es que la Conferencia Magistral fue todo un éxito y el público quedo encantado con Gekko (http://www.gekkoware.org/), hubó bastante interés por el proyecto y cada vez son más sitios, como este, donde se esta implementando este poderoso CMS que actualmente esta en su versión 0.6.4 recientemente lanzada.

Así es como concluimos de manera muy exitosa el primer día de actividades del CongresoFS y por el momento es todo, espero que esta primera entrega haya sido del agrado de todos ustedes y en las siguientes ediciones de la RevistaSL publicaremos la crónica de los otros dos días del CongresoFS.

Un saludo muy cordial y recuerden que: "La Tolerancia es parte de la Libertad".

Jose Carlos Nieto y Josúe Gutierrez. Proyecto Gekko

Proyectos SL

Trabajando desde las cuevas...

Generando un kernel con Freepascal

Matias Vara
matiasvara@yahoo.com

Bueno tal vez el titulo no sea muy descriptivo pero básicamente lo que tratare de explicar es como generar un ejecutable en freepascal sin que este utilice llamadas al sistema , llevando obviamente a la caída del sistema .

A la hora de comenzar a escribir un S.O. la principal duda q se me planteo fue que compilador iba a utilizar . Elegí Freepascal , porque además de gustarme el lenguaje me era fácil portarlo puesto que es independiente del S.O. sobre el cual corra , esto no significa que no utilice llamadas al sistema sino que posee un capa de abstracción de llamadas primitivas las cuales son adaptadas al S.O. sobre el cual corra , es por eso que al compilador no le interesa el S.O. sobre el que se ejecuta .

En el caso de la versión 1.0.6 de FP utilizando el extensor go32v2 el tema es bastante fácil . Cuando se utilizan procedimientos y funciones dependientes del S.O. como por ejemplo writeln , write , read ,etc . (no así sizeof () , len() , etc .) el compilador enlazada nuestro ejecutable al archivo objeto donde se encuentran estas llamadas , básicamente lo que se hizo en Toro es capturar estas llamadas , creando el archivo Lib/fpclib/fpclib.pas con estas llamadas y linkeandolo junto con todo el kernel .
Por ejemplo :

llamada muy utilizada para el tratamiento de strings

```
procedure
int_strconcat(s1,s2:pointer);[public,alias:'FPC_SHORTSTR_C
ONCAT'];
var
s11, s21 : byte;
type
pstring = ^string;
begin
if (s1=nil) or (s2=nil) then
exit;
s11:=length(pstring(s1)^);
s21:=length(pstring(s2)^);
if s11+s21>255 then
s11:=255-s21;
move(pstring(s1)^[1],pstring(s2)^[s21+1],s11);
pstring(s2)^[0]:=chr(s11+s21);
end;
```

También se podría haber linkeado directamente el archivo objeto de FP donde se encontrase la llamada , pero haría crecer el kernel enormemente!!! .

En las unidades mientras no se utilicen llamadas que depende del S.O. o que provengan de unidades externas no hay problema , el compilador no realizara llamadas al sistema .
La cosa cambia con un ejecutable . Cuando se compila el archivo kernel/kernel.pas , este ya no es una unidad sino que es un programa , para volverlo “limpio” de llamadas al sistema lo que se hace es simplemente quitarlas del código assembler :

Extracto del archivo /kernel/Makefile .

```
$(FPC) $(FPC_FLAGS) kernel.pas

$(GREP) -iv "INIT$$SYSTEM" kernel.s > kernel.tmq
$(GREP) -iv "FPC_INITIALIZEUNITS" kernel.tmq >
kernel.tmp
$(GREP) -iv "FPC_DO_EXIT" kernel.tmp > kernel.kkk
$(GREP) -iv "OBJPAS" kernel.kkk > kernel.s
```

Estas son los símbolos que deben ser retirados y que llevan a la caída del kernel cuando bootea (a la caída me refiero a una excepción , muy común la 13)

Una vez compilados todos los archivos objetos son linkeados con ld , puede pasar que tire errores de símbolos desconocidos , es decir hay funciones o procedimiento que están llamando a otros que no se encuentran en los archivos objetos linkeados . Lo mas posible es que haya errores en el tipeado de nombres , o que el head de la función no haya sido declara como publica . Si todo esto no es correcto habría que ver si es una llamada a librerías del compilador , si es así , se debería buscar el procedimiento o función en el código de FP , copiarlo al archivo lib/fpclib/fpclib.pas y declararlo como publico con el alias devuelto como error en ld .

FPC 2.0.0 sobre win32 y linux :

Sobre estas plataformas la cosa no me resulto fácil , lamento informa que no he llegado a una versión booteable del kernel de toro compilado sobre win32 , es por eso que las distribuciones se harán por ahora para go32v2 y para linux , o tal ves solo para linux puesto que se dejaron producir versiones de FPC para DOS .

Para el caso de FPC 2.0.0 sobre linux , al compilar el archivo kernel/kernel.pas solo se quita el símbolo:

```
$(GREP) -i 'FPC_INITIALIZEUNITS' -v kernel.s > kernel.q
```

Para el caso de llamadas al sistema lo que opte hacer ahora fue directamente el linkeo de los archivos objeto que contienen las llamadas utilizadas por el compilador estas son :

```
/usr/lib/fpc/2.0.0/units/i386-linux/rtl/prt0.o
/usr/lib/fpc/2.0.0/units/i386-linux/rtl/system.o
/usr/lib/fpc/2.0.0/units/i386-linux/rtl/objpas.o
```

Lo que hace es incrementar enormemente el archivo toro-1.1 , que posee casi el triple del mismo archivo generado con el FPC para DOS .

La unidad Lib/fpclib/fpclib.pas deja de utilizarse para el FPC sobre linux . Por supuesto , posiblemente dentro de estos archivos objetos hayan funciones que realicen llamadas al sistema , pero no causaran problemas mientras no se las llame :) .

Espero que este articulo les sirva para saber como se puede generar un ejecutable totalmente booteable (por ejemplo un elf para grub o un binario para un booteador propio) sin que posibles llamadas del compilador lleven a excepciones no deseadas .

Aclaración! si el elf generado va a ser utilizado por un booteador como grub tiene que ser creado bajo la norma multiboot sino grub no podrá levantarlo , toro fue creado bajo ese Standard , en próximos documentos describiré como hacer un elf que pueda ser levantado con grub .

Colas ligadas en Toro

Matias E. Vara
matiasvara@yahoo.com

Las colas ligadas son un método eficaz para mantener agrupados una cantidad ilimitada de elementos . En Toro son utilizadas para agrupar los procesos , los timers , los superbloques , los inodos , los buffers , etc. , se le ha dado gran cantida de uso .

Podemos clasificarlas en dos tipo : simplemente ligadas o doblemente ligadas .

Por lo general las estructuras que se encuentran en una cola simplemente ligada posee solo un campo que apunta a la siguiente estructura y la ultima de la cola posee este campo a un puntero nulo , es decir solo pueden ser recorridas en un solo sentido .
A diferencia de estas , las doblemente ligadas posee dos campos una punteando a la siguiente estructura y otra a la anterior , pudiéndose así recorrerla en ambos sentidos y de manera cíclica .

La principal diferencia es para què van a ser utilizada . Por un lado las simplemente ligadas posee un campo menos , es decir ocupan menos memoria , pero si en la cola se están agregando y quitando elementos continuamente se consume mucha cpu , porque por cada elemento que debo quitar debo recorrer toda la cola para encontrar el anterior! .
Este problema se soluciona creando un nuevo campo apuntando al elemento anterior , así surgen las doblemente ligadas .
Es por eso que las colas que mantiene a los procesos , inodos , timers ,etc. , son colas doblemente ligadas , mientras que las colas que mantiene a los buffers que deben ser escritos a disco se encuentran en colas simplemente ligadas , cada buffer es agregado al comienzo y cuando deben ser quitados por la llamada sync() , se recorre la lista desde el inicio sacando de a uno todos .

Todo muy lindo , pero el hecho de que sean tan utilizadas hace necesarios procedimiento muy rápidos y eficientes , y este es el motivo del articulo .
Lo que se trato de crear fue procedimientos generales para el tratamiento de colas ligadas , tal como lo hace linux , pero en freepascal .

Todo el código del manejo de colas ligadas se encuentra en el archivo Include/Head/list.h

Este como se ve no es una unidad sino solo un archivo que debe ser incluido en la unidad luego de IMPLEMENTATION

Para que funcione deben ser declarados en la unidad cuatro símbolos ,de la manera tradicional , {$DEFINE Use_Tail} , estos son :

Use_Tail : le indica al compilador que el código de manejo de las lista debe ser incluido

nodo_struct : Debe contener la estructura de los elementos dentro de la cola , por ejemplo para una cola ligada de procesos valdría :

```
{$DEFINE nodo_struct = p_tarea_struc}
```

Siempre se considera a nodo_struct como un puntero .

nodo_tail : Puntea al elemento cabecera de la lista , puede estar definido o no .

next_nodo y prev_nodo : Estas estructuras poseen los campo dentro de los elementos de la lista que puntean al siguiente elemento y al anterior , por ejemplo para el caso de cola de procesos , estos valdrían :

```
{$DEFINE next_nodo = next_tarea }
{$DEFINE prev_nodo = prev_tarea}
```

El procedimiento para encolar un elemento , si se ha definido nodo_tail :

procedure Push_Node(Nodo : nodo_struct) ; inline ;

o en el caso de que no se haya definido nodo_tail :

procedure Push_Node(Nodo : nodo_struct;var Nodo_Tail : nodo_struct);inline;

La diferencia de que se defina o no nodo_tail , es la capacidad de trabajar en una unidad con mas de una cola ligada , puesto que de lo contrario se definiría un único valor para el símbolo nodo_tail , con esto se especifica el nodo cabecera de la cola en cada llamada .

Siempre el elemento es agregado al comienzo de la cola .

Ojo! nodo_tail no es una estructura sino solo un puntero al primer elemento de la cola .

Para quitar un elemento sucede lo mismo :

El procedimiento para quitar un elemento , si se ha definido nodo_tail :

```
procedure Pop_Node(Nodo : nodo_struct );inline;
```

o en el caso de que no se haya definido nodo_tail :

```
procedure Pop_Node(Nodo : nodo_struct;var Nodo_tail : nodo_struct);inline;
```

Como se ve todos los procedimiento son declarados como inline para acelerar su ejecución

Para hacer un poco mas “entendible” el código suelo definir otro símbolo que oculte a estos procedimientos :

```
{$DEFINE push_buffer = push_node }
{$DEFINE pop_buffer = pop_node }
```

Lo bueno de esto que definiendo un par de símbolos ya tenes todo el código para la manipulación de listas ligadas sin importar las estructuras , campos , etc .

Para el caso de colas simplemente ligadas , se debe definir el símbolo Use_Simple_Tail
para comenzar a trabajar con ella .
Igual que en el caso de las doblemente ligadas se cuenta con los procedimientos :

para agregar un elemento :

```
procedure Push_Snode (Nodo , sNodo_Tail : snode_struct);inline;
```

y para quitar un elemento :

```
procedure Pop_Snode (Nodo,sNodo_Tail : snode_struct);inline;
```

Ahora los símbolos nodo_struct y nodo_tail pasan a llamarse snode_tail y snode_struct respectivamente

Espero que les haya servido este pequeño articulo .
Una saludo

Matias Vara, es un argentino que actualmente estudia la universidad. Empezo a escribir Toro en el 2003 debido al interes que encontro en el foro de Hispabyte

El sitio web de Toro es: http://toro.sourceforge.net

Bytez! SL

Sentimentalismo friki...

Despues de mas de ocho meses de no haber liberado un número de la revista, mis compañeros editores no podian creer cuando en mi blog escribi que en dos semanas saldria el nuevo número, ¿como es posible el trabajo de meses hacerlo en un par de semanas?, peor aun, no teniamos Staff SL, faltaban demaciadas cosas, con tanto tiempo se perdierón articulos que teniamos para este número, etc., un gran reto, pero el número esta aqui y ustedes lo tienen archivado en su computadora.

Pero este número que tiene un diseño sencillo, puede que información de poco o mucho interés para ustedes, es el fruto del trabajo de dementes partidarios del software libre; para todos ellos quiero aprovechar este espacio, para agradecerles su trabajo, a todos los editores que en vez de ponerse a trabajar o estudiar se tomarón el tiempo de sentarse a escribir sus articulos, a los diseñadores que terminan con los ojos rojos pero siempre en la mañana se encuentran e-mails de ellos con sus colaboraciones, al webmaster que a pesar de estar en medio de examenes tuvo tiempo de preparar todo para tener el enlace de descarga, a los colaboradores que enviarón sus articulos y sobre todo al editor en jefe, gracias a quien estoy en medio de la noche y con las manos atrofiadas escribiendo esto :)

Sin embargo, y a pesar de este gran exfuerzo necesitamos apoyo, necesitamos colaboraciones para poder seguir mostrando el trabajo de la comunidad a la comunidad, necesitamos de su ayuda para que el proyecto se difunda a todos los rincones en donde entiendan el español. También, como se ha mencionado varias veces, nuestro sueño es ver esta revista impresa, para ello solicitamos su apoyo en forma de donaciones o patrocinio, la revista lo agradecera en todas las formas que esten a nuestro alcance.

Bueno, despues de tanto sentimiento electronico me despido, agradeciendole a ti querido lector, que nos lees y que esperamos sea de tu agrado el presente archivo.

Carlos Augusto Lozano Vargas
vendetta@hackertm.org