Root, CyanogenMod 7, Ultra Smooth CyanogenMod 9.1 para HTC WildFire

Introducción

En este post voy a mostrar como trastear un HTC WildFire. Hace ya tiempo adquirí uno de estos terminales. Con amargura vi como se quedaba en la versión 2.2 y nunca se iba a actualizar a la versión 2.3.x de Android. Esto implicaría perder las siguientes mejoras:

- Función copiar y pegar mejorada
- Teclado rediseñado
- Gestor de descargas
- Manejo de energía mejorado y control de aplicaciones

Ciertamente la parte más importante es la del manejo de aplicaciones i gestión de la batería. Aunque la mejoras de usablidad siempre son bienvenidas, la verdad. También hay que tener en cuenta que estas no son todas las diferencias pero si las que se aplican a este terminal. Ademas esta rom nos permite personalizar muchos aspectos como transición entre

pantallas, menús… Incluye AWD Launcher que no esta nada mal y podemos personalizarla con multitud de temas.

Después de mucho pensarlo decidí dar el paso rootear el terminal e instalar alguna rom que me permitiera disfrutar de la nuevas mejoras. El proceso fué un poco complicado e implica cierto riesgo, pero la mejora es incerible. La verdad después de probar CyanogenMod aunque la rom oficial se actualizase a 2.3 no cambiaría.

Consideraciones previas

Bien en ningún caso me hago responsable de ningún daño ocasionado a algun terminal. Esta guía y las aplicaciones que se mencionan en ella carecen de cualquier tipo de garantía. Es decir, bajo vuestra responsabilidad queda lo que hagáis con vuestro terminal.

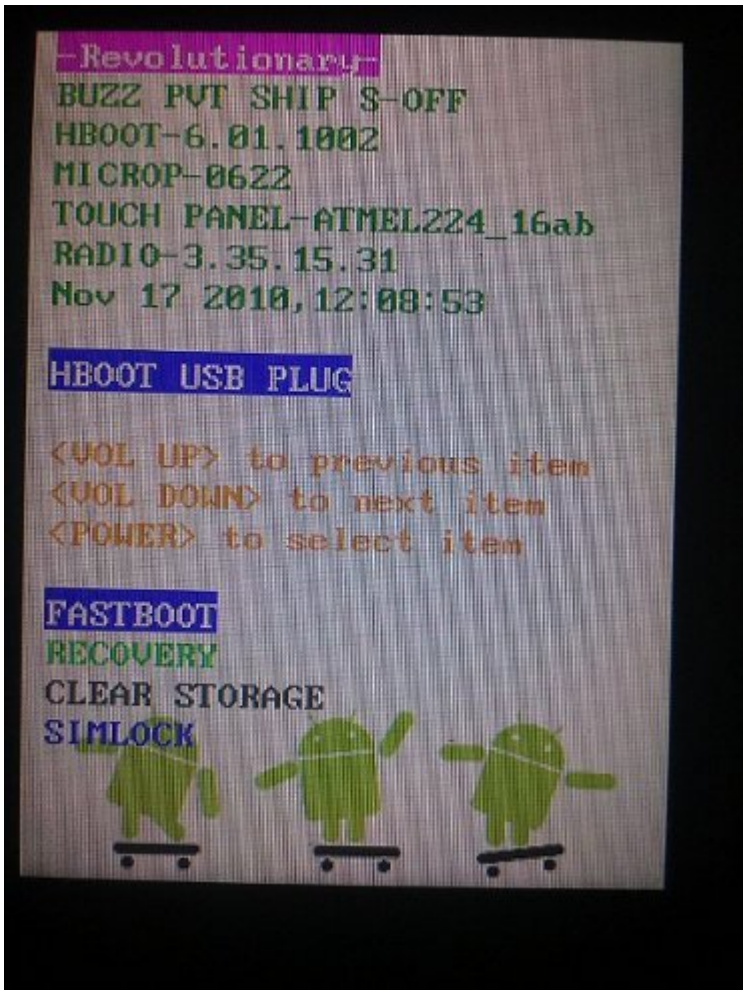
Es recomendable que la batería del termina este cargada, realizar backups y asegurarse de que en la sd tenemos espacio para estos y las bajada que vamos a realizar. A continuación mostraremos algunas aplicaciones que nos pueden ser de ayuda:

- Contactos del teléfono -> No requiere aplicación externa. Simplemente entrar en la lista de contactos, apretar el botón de menú de Android y seleccionar Importar/exportar -> Exporta a la tarjeta SD.
- Registro de llamadas -> Call Log Backup & Restore
- Registro de sms -> SMS Backup & Restore
- Aplicaciones -> MyBackup (ATENCION!! Esta es una versión de prueba funcional solo durante 30 días.)

S-OFF e instalación de un custom recovery

Esta parte es un poco complicada conviene seguir todos los pasos con paciencia y asegurándonos que realizamos cada uno correctamente.

- Debemos descargar Revolutionary.
- En cuanto empiece la descarga, la pagina se actualizará. Para poder utilizar el programa necesitamos una beta key. Para conseguirla es necesario rellenar el formulario que aparecerá introduciendo nuestra versión de HBOOT y número de serie.
    - HBOOT -> para encontrar la versión de HBOOT debemos apagar el terminal y encenderlo apretando el botón de volumen hacia abajo y el botón de encendido. Aparecerá una pantalla parecida a la siguiente en la que podemos ver la versión de HBOOT.



- El número de serie lo podemos encontrar debajo de la

batería o bien, con el terminal conectado por usb, ejecutando en un terminal de pc

./adb devices

- Una vez realizados los pasos anteriores, debemos ejecutar Revolutionary

  ./revolutionary

  desde un terminal de pc, con nuestro WildFire conectado por usb, e introducir la beta key.

- Nuestro terminal se reiniciará en poco tiempo. El proceso puede no funcionar a la primera, repetidlo. En caso de tener algún problema podéis encontrar información en la wiki de Revolutionary o en su canal de irc.

Instalar CyanogenMod 7 (Android 2.3.7) y GoogleApps

Hace pocos días el Team de CyanogenMod sacó la versión 7.2 estable. Lo cual es una gran noticia. Las versiones estables son realmente buenas. Normalmente la segunda release candidate de una versión ya suele estar muy bien.

Bien podemos realizar este proceso desde Rom Manager:

- Arrancamos la aplicación y seleccionamos **Download Rom**.
- Seguidamente en el apartado **Free** seleccionamos **CyanogenMod**.
- Seleccionamos la última stable release, en este caso la **7.2.0**. y la opción de **GoogleApps**.
- Cuando finalicen las descargas debemos seleccionar **Wipe Data** y **Wipe Cache**, seria muy recomendable realizar un backup de la rom existente si queréis hacerlo seleccionad **Backup Existing ROM**.
- La aplicación SuperUser nos preguntará si queremos dar

permisos a Rom Manager, aceptamos.
- Nuestro terminal se reiniciará en modo recovery hará un wipe de cache y data, a continuación se instalará CyanogenMod y las GoogleApps. Cuando finalice la instalación el terminal se reiniciará otra vez y CyanogenMod arrancará por primera vez.

O bien manualmente, es recomendable utilizar Rom Manager puesto que simplifica el proceso. En este caso debemos descargar:

- La rom cm-7.2.0-buzz
- Las GoogleApps gapps-gb-20110828-signed
- Una vez descargadas debemos copiarlas a la raiz de la targeta SD, entrar en Rom Manager y seleccionar **Reboot into Recovery**.
- Una vez en el recovery debemos utilizar las teclas de volumen para navegar y el botón de encendido o el trackball como enter.
- Bien si queréis realizar un backup de la rom actual (cosa recomendable porque es la original) entrad en **backup and restore** y seleccionad backup. Finalmente seleccionad **+++++Go Back+++++** para volver al menú principal.
- En el menú principal seleccionad **Wipe data/factory reset** y seguidamente **Wipe cache partition**.
- Seleccionad **Install zip from sdcard y Choose zip from sdcard**.
- Seleccionad **cm-7.2.0-buzz.zip** y esperad a que acabe el proceso de instalación.
- Bien para instalar las GoogleApps el proceso es el mismo que con la rom, simplemente debemos seleccionar **gapps-gb-20110828-signed**.
- Finalizada la instalación seleccionamos **+++++Go Back+++++** para ir al menú principal y finalmente **Reboot system now** option. Nuestro terminal se reiniciará y

CyanogenMod arrancará por primera vez

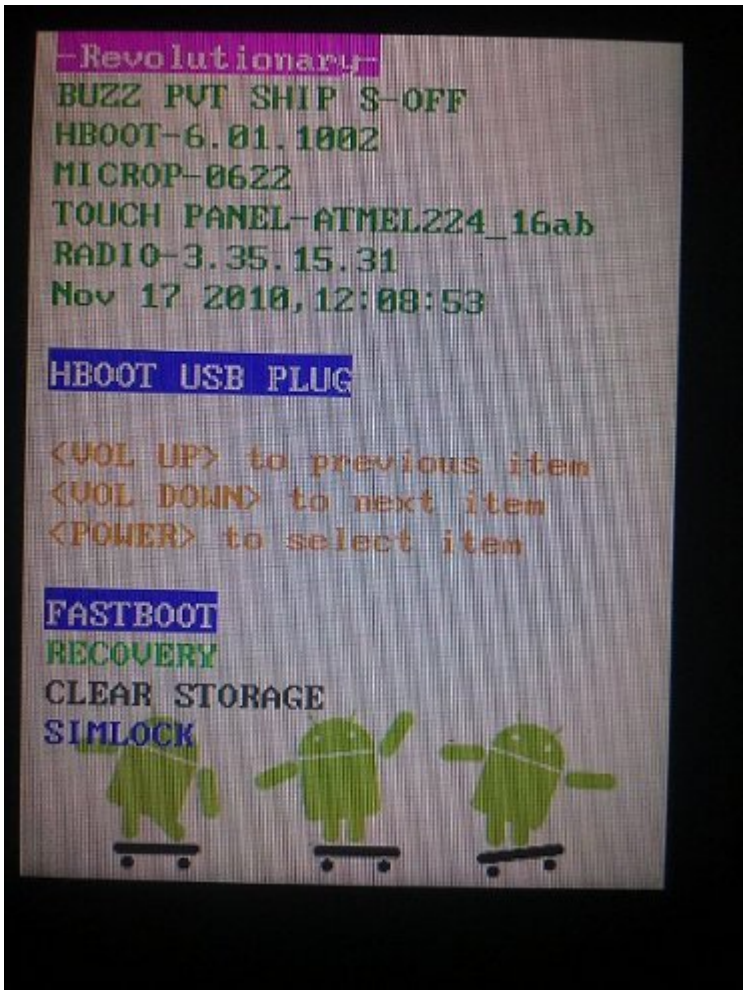
Tips & tricks

Hay algunas cosas a tener en cuenta:

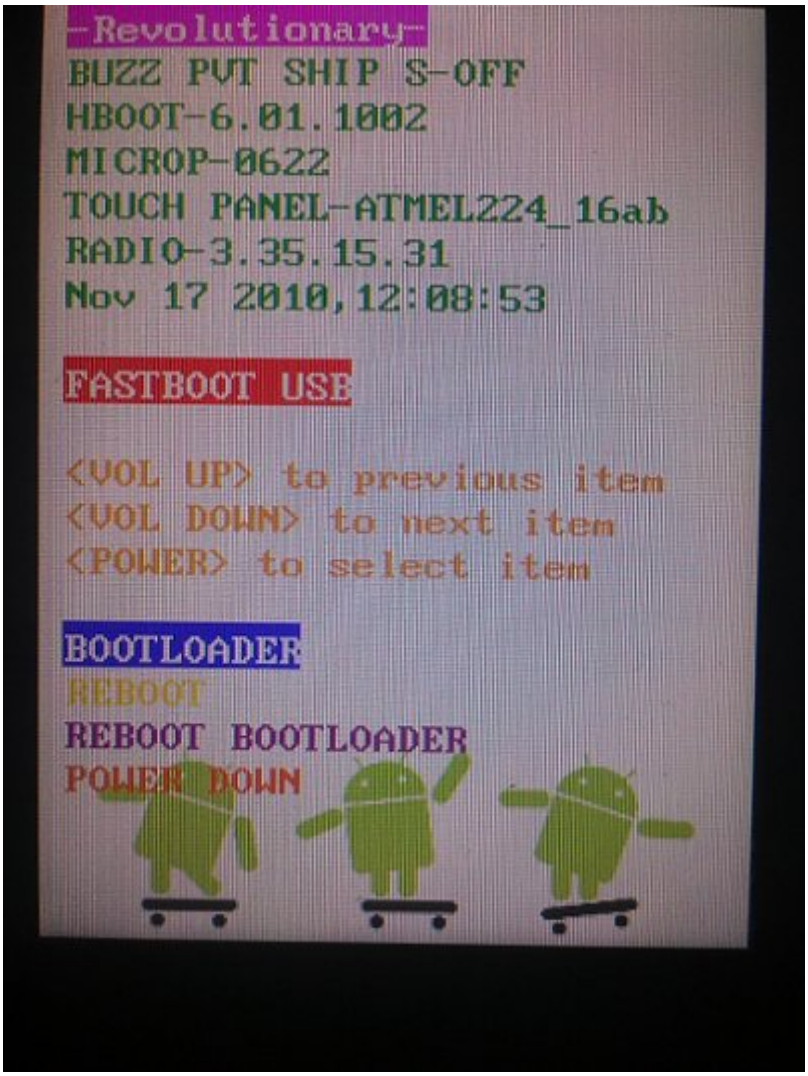
- Ubuntu no reconoce el terminal. Debéis situaros en la carpeta donde se encuentra adb. Matar el servidor y arrancarlo como root

  ```
  ./adb kill-server
  sudo ./adb start-server
  ```

- Puede que el gps no coja señal. Eso es debido a que hay que flashear una nueva radio. Podéis descargar la radio des de aquí . Una vez descargada la radio debéis descargar el siguiente ejecutable FastBoot. Debéis apagar el teléfono, y estando este conectado por usb, arrancar presionando el botón de encendido y volumen abajo. Entrareis en la siguiente pantalla.



- Con las teclas de volumen debéis seleccionar **FASTBOOT** y clicar el botón de encendido. Llegareis a esta nueva pantalla.



- Llegados a este punto des de un terminal en el pc debéis ejecutar

  ./fastboot flash radio radio.img

  Una vez acabe la instalación seleccionáis con los botones de menú **REBOOT** y presionáis el botón de encendido para reiniciar el terminal, con esto el gps debería funcionar perfectamente. Si aún así continuáis teniendo problemas podéis encontrar soporte en este foro CyanogenMod Forum.

- La aplicación de GoogleMpas se queda colgada. Desafortunadamente nuestro terminal no soporta la última versión de google mpas. Podéis descargar el apk des de aquí, es la versión 5.0.0 que funciona perfectamente.

- No se puede instalar flash. Podéis descargar una versión funcional de flash para este terminal des de aquí

Actualización a ICS

Recientemente he estado probando otra Rom, Ultra Smooth CyanogenMod 9.1.Tanto su estabilidad, eficiencia y compatibilidad con el hardware del terminal son impresionantes. La camara funciona bastante bien (excepto la gravación de video) y youtube (excepto el visionado en HD) . Algunos consejos:

– En opciones -> Performance -> Processor podéis modificar la velocidad del procesador entre 352 y 710 MHz y el governor SMARTASSV2 (estaremos realizando overclocking esto comporta cierto riesgo pero dentro de unos valores razonables).

– Podéis modificar en las opciones del launcher (menu de android -> launcher settings – Desktop settings) Wallpaper scrolling para desactivarlo.

– Podéis encontrar unas tiny google apps aquí simplemente una vez cargada la página de Google Docs persionad Ctrl+s o en el menu File->download

– En este post podéis encontrar como instalar Google VoiceSearch

– Es recomendable utilizar Seeder(consigue reducir el Lag notablemente)

Observaciones
Bien, el proceso es un poco complicado y algunos pasos son críticos, pero a mi me mereció la pena el cambio. Ahora estoy muy contento con mi terminal, hay muchas opciones que la rom oficial no ofrece, la estabilidad es muy buena así como

también el ahorro de barrería y la gestión de memoria (cuando htc sence desaparece, de las entrañas de nuestro terminal memoria ram aflora). También cabe tener en cuenta el echo de ser root. Se pueden realizar backups de todo, hay muchas aplicaciones que explotan esta funcionalidad, pero esta parte os la dejo para vosotros.
En la nueva parte para instalar ICS podemos observar varios ajustes, seamos realista estamos hablando de correr una ICS 4.0.4 en un HTC WildFire sorprendente es que funcione y lo bien que lo hace la Rom Ultra Smooth CyanogenMod 9.1, en la fuentes se encuentra el link al post de xda del creador y colaboradores de la rom.

Fuentes

- CyanogenMod
- Ultra Smooth CyanogenMod 9.1

Ruben.

Montar BlackBerry PlayBook en Ubuntu

Introducción

En este post voy a mostrar como montar una BlackBerry PlayBook a Ubuntu.Algo que a priori debería ser sencillo y lo es con un perqueño workaround.

Cuando se conecta la PlayBook mediante cable usb a Ubuntu una pantalla nos indica que se están instalando los drivers… eso no va a ocurrir en Ubuntu, simplemente cerramos esa pantalla.

Bien la idea consiste en conectarla por red, los pasos a seguir son realmente sencillos.

Preparar la PlayBook

- Iremos a Configuración -> Almacenamiento y uso compartido. En conexión usb seleccionaremos Conectarse a Windows.
- Seguidamente iremos (dentro de Configuración) a -> Sobre (la sección donde encontramos la información relacionada con el dispositivo). En Visualizar Información de la tablet seleccionaremos Red. Deberemos apuntarnos la dirección IP de la PlayBook para usarla después.

Conectarnos con Ubuntu

- Iremos a Lugares -> Conectarnos a un Servidor.
- En el formulario de conexión seleccionaremos Recurso Compartido de Windows e introduciremos la IP que anteriormente hemos apuntado y con esto accederemos mediante red al almacenamiento masivo de la PlayBooK.

Observaciones
Bien, esto no es más que un truquillo simple pero a su vez muy útil. Próximamente mostraré como portar una aplicación Android a PlayBook y las limitaciones que existen.

Ruben.

Arduino Segunda Parte

Introducción

En este segundo post vamos a realizar una pequeña aplicación que controla dos leds (uno rojo y uno verde) conectados a Arduino. Esta placa estará conectada a un pc con un servidor servidor web. En dicho servidor web alojaremos un archivo PHP que se encargará de enviar valores a través de USB al arduino. Con un pequeño programa en Android atacaremos mediante Post a ese fichero. De esta manera podremos controlar los leds desde nuestro terminal, siempre y cuando estemos en la misma red local que el servidor web.

- GNU\Linux, venditos ficheros

GNU\Linux es un sistema operativo que utiliza el núcleo Linux junto con bibliotecas y herramientas del proyecto GNU y de muchos otros proyectos/grupos de software.
Bien, en GNU\Linux el hardware esta asignado a un fichero. Así pues, solo tendremos que esribir en el fichero que identifica al puerto USB donde tenemos conectado nuestro Arduino, ni drivers ni complicaciones. Abrir en modo escritura, escribir i

cerrar el fichero, así de fácil.

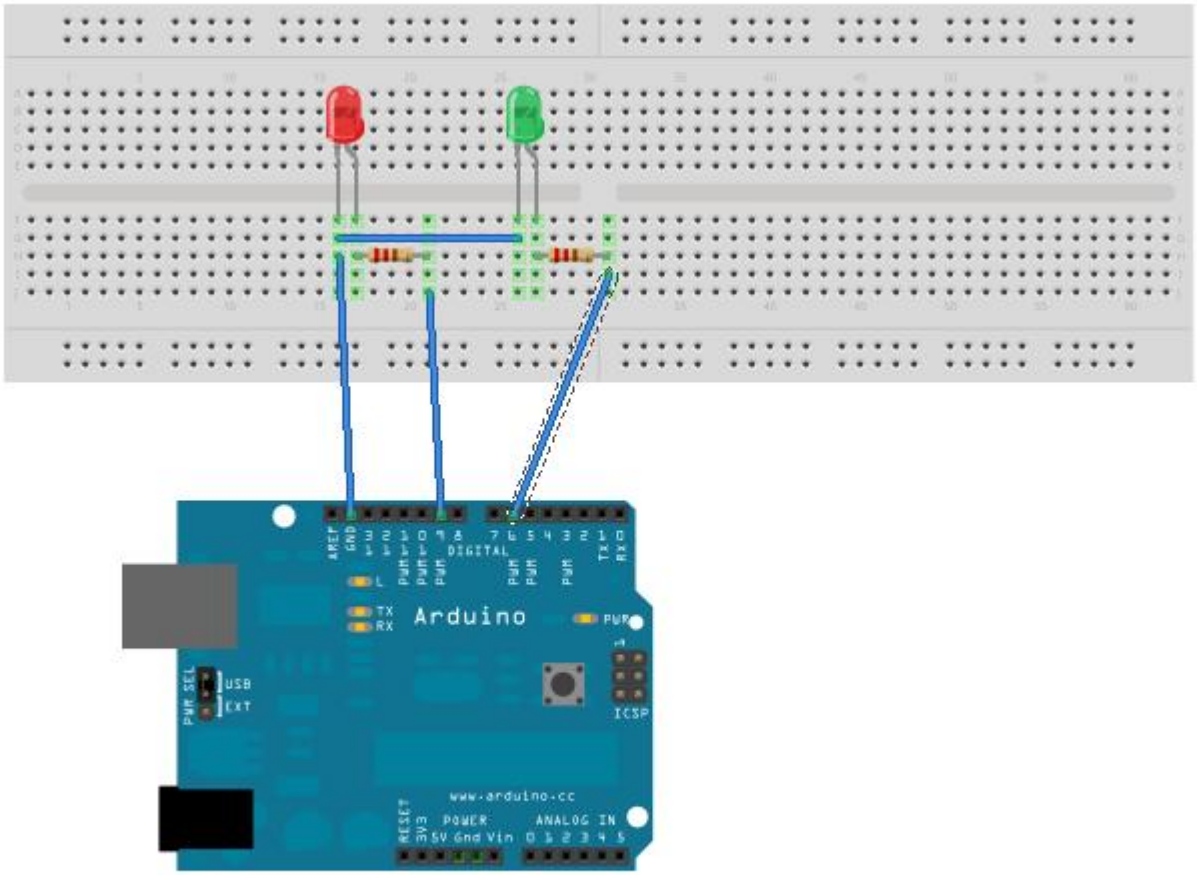
- Android

Android es un sistema operativo basado en el núcleo Linux diseñado originalmente para dispositivos móviles, tales como teléfonos inteligentes, pero que posteriormente se expandió su desarrollo para soportar otros dispositivos tales como tablet, reproductores MP3, netbook, PC, televisores, lectores de e-book e incluso, se han llegado a ver en el CES, microondas y lavadoras. Crearemos una pequeña aplicación que enviará un post a un fichero php que se encargará de escribir en el puerto USB correspondiente. Por la simplicidad de la aplicación y no ser Android objetivo del post sólo pondremos el código relativo a la ejecución del post.

Instalación de paquetes necesarios

Deberemos instalar las librerías y el IDE. Para ello podeis consultar la primera parte de la serie Arduino Arduino Primera Parte

Diseño electrico del Arduino



Código Arduino

```
int ledPin = 9;
int ledPin2 = 6;
int number_in = 0;

void setup() {
pinMode(ledPin, OUTPUT);
pinMode(ledPin2, OUTPUT);

Serial.begin(115200);
}

void loop() {
if (Serial.available() > 0) {
```

```
number_in = Serial.read();
Serial.print("He rebut: ");
Serial.println(number_in, DEC);
}

while (number_in == 49) {
digitalWrite(ledPin, HIGH);
delay(800);
digitalWrite(ledPin, LOW);
delay(800);
if (Serial.available() > 0) {
number_in = Serial.read();
Serial.print("He rebut: ");
Serial.println(number_in, DEC);
}
}

while (number_in == 51) {
digitalWrite(ledPin2, HIGH);
delay(800);
digitalWrite(ledPin2, LOW);
delay(800);
if (Serial.available() > 0) {
number_in = Serial.read();
Serial.print("He rebut: ");
Serial.println(number_in, DEC);
}
}
}
```

Servidor Web

- Deberemos dar permisos al usuario www-data (usuario por defecto del servidor Apache) para escribir en el fichero que corresponde al puerto USB:

```
sudo adduser www-data dialout
```

- En el servidor web alojaremos el fichero que se encargará de enviar las señales a Arduino:

```
if (!empty($_POST['green'])) {
$return['msg'] = "green";
// '/dev/ttyACM1' corresponde a al puerto USB deberemos cambiarlo por el que corresponda en nuestro PC
$fp =fopen("/dev/ttyACM1", "w");
fwrite($fp, chr('51'));
fclose($fp);
}else if (!empty($_POST['red'])){
$return['msg'] = "red";
$fp =fopen("/dev/ttyACM1", "w");
fwrite($fp, chr('49'));
fclose($fp);
}else if (!empty($_POST['off'])){
$return['msg'] = "off";
$fp =fopen("/dev/ttyACM1", "w");
fwrite($fp, chr('48'));
fclose($fp);
}
```

Código Android

- Encender led Verde

```
HttpClient httpclient = new DefaultHttpClient();
HttpPost httppost = new HttpPost("http://localhost/post.php");

List postValues = new ArrayList(2);
postValues.add(new BasicNameValuePair("green", "green"));
httppost.setEntity(new UrlEncodedFormEntity(postValues));
```

```
HttpResponse response = httpclient.execute(httppost);
```

- Encender led Rojo

```
HttpClient httpclient = new DefaultHttpClient();
HttpPost httppost = new HttpPost("http://localhost/post.php");

List postValues = new ArrayList(2);
postValues.add(new BasicNameValuePair("red", "red"));
httppost.setEntity(new UrlEncodedFormEntity(postValues));

HttpResponse response = httpclient.execute(httppost);
```

- Apagar leds

```
HttpClient httpclient = new DefaultHttpClient();
HttpPost httppost = new HttpPost("http://localhost/post.php");

ListpostValues = new ArrayList(2);
postValues.add(new BasicNameValuePair("off", "off"));
httppost.setEntity(new UrlEncodedFormEntity(postValues));

HttpResponse response = httpclient.execute(httppost);
```

Observaciones

Con esta segunda parte podemos ver una aproximación a las posibilidades de arduino. De esta manera podemos controlar cualquier cosa que connectemos a Arduino (las posibilidades son infinitas) y es que con este «juguete» se sabe cuando se empieza pero no cuando se acaba.

Fuentes

- Wikipedia – Android
- Android Hello World

Ruben

Arduino – Primera Parte

Introducción
En este primer post mostraremos que es Arduino, como funciona y como cargar el programa Blink. Esto nos permitirá ver el funcionamiento básico y comprobar que nuestra placa funciona correctamente.

- Arduino 

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.
El hardware consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida. Los microcontroladores más usados son el Atmega168, Atmega328, Atmega1280, ATmega8 por su sencillez y bajo coste que permiten el desarrollo de múltiples diseños. Por otro lado el software consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring y el cargador de arranque (boot loader) que corre en la placa.
Arduino se puede utilizar para desarrollar objetos

interactivos autónomos o puede ser conectado a software del ordenador (por ejemplo: Macromedia Flash, Processing, Max/MSP, Pure Data). Las placas se pueden montar a mano o adquirirse. El entorno de desarrollo integrado libre se puede descargar gratuitamente.
Al ser open-hardware, tanto su diseño como su distribución es libre. Es decir, puede utilizarse libremente para el desarrollo de cualquier tipo de proyecto sin haber adquirido ninguna licencia.

- Processing

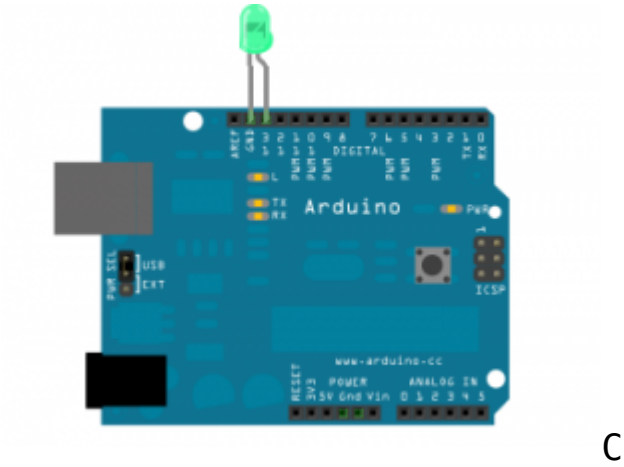
Processing es un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, de fácil utilización, y que sirve como medio para la enseñanza y producción de proyectos multimedia e interactivos de diseño digital. Fue iniciado por Ben Fry y Casey Reas a partir de reflexiones en el Aesthetics and Computation Group del MIT Media Lab dirigido por John Maeda.
Processing es desarrollado por artistas y diseñadores como una herramienta alternativa al software propietario. Puede ser utilizado tanto para aplicaciones locales así como para

aplicaciones web (Applets).
Se distribuye bajo la licencia GNU GPL.

Instalando el software necesario

Deberemos instalar las librerías y el IDE con este simple comando:
sudo apt-get install arduino arduino-core

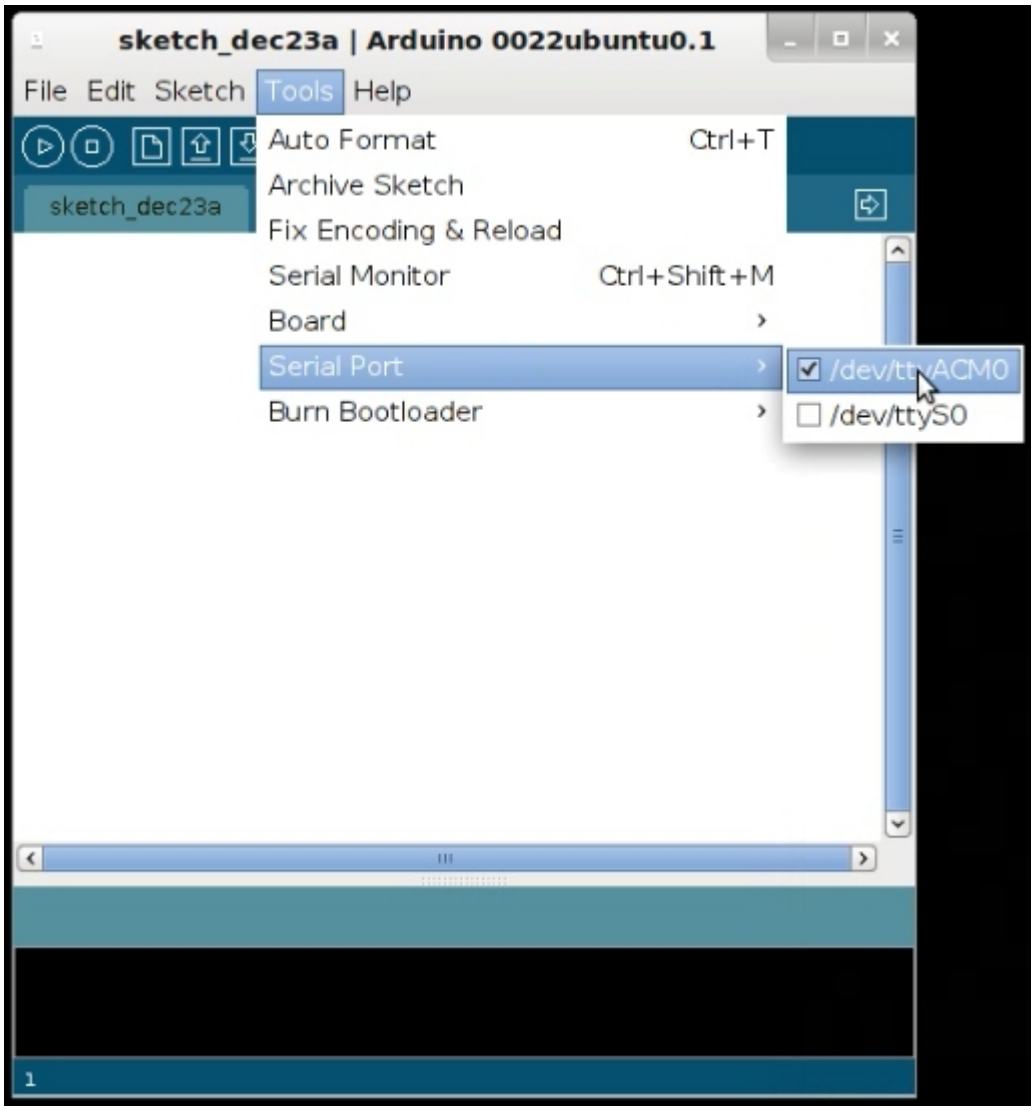
Diseño Arduino



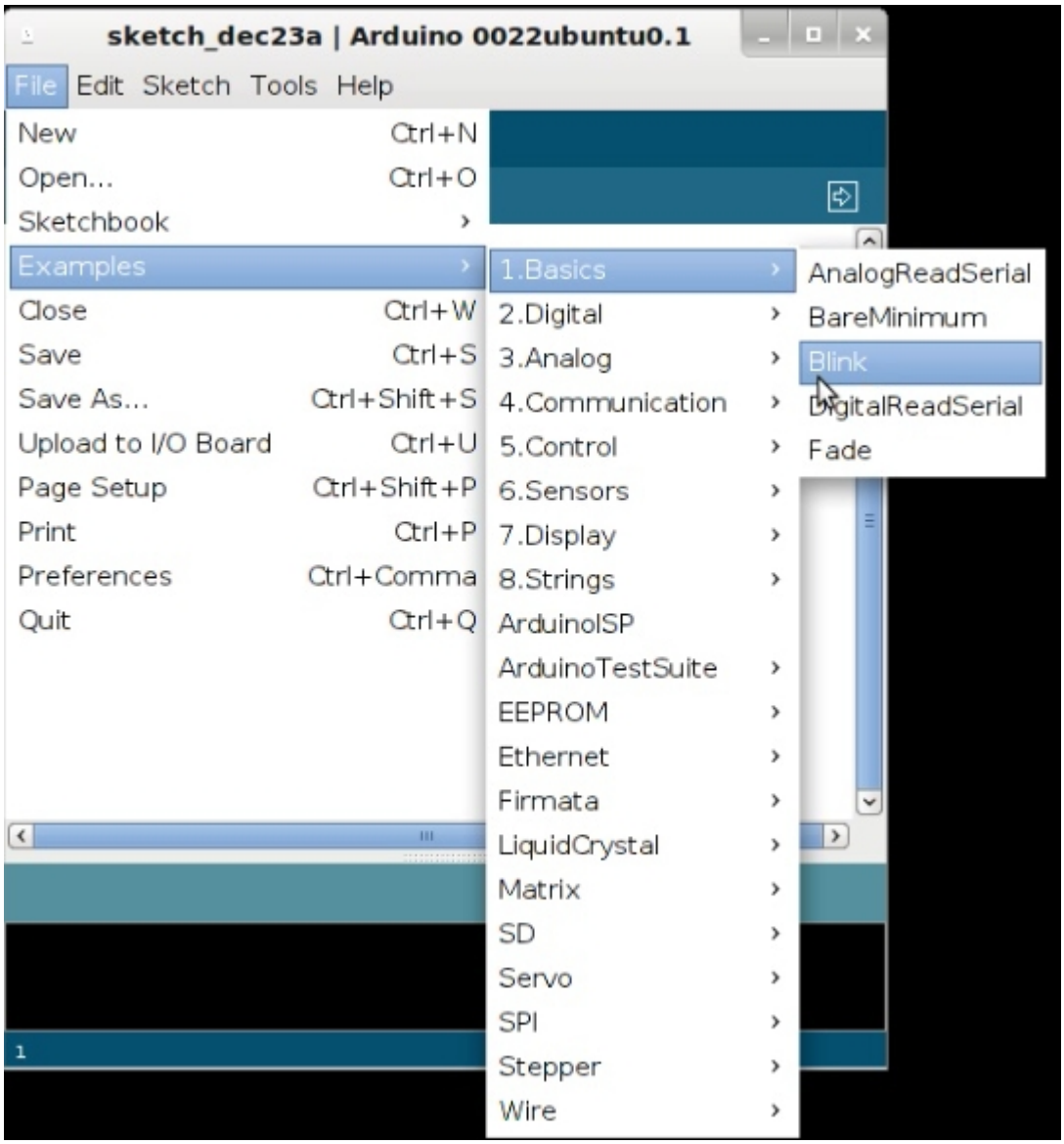
Conectaremos el led teniendo en cuenta que la patilla corta es el negativo y la patilla larga el positivo. Necesitaríamos una resistencia para que no se queme el led, por suerte el pin 13 de la placa Arduino incluye una. Así pues, conectaremos la patilla larga(positivo) al conector 13 y la corta(negativo) a la entrada de corriente GND, como se puede comprobar los dos conectores y la resistencia están puestos estrategicamente.

Cargando el programa Blink

Este programa es el más básico, simplemente produce un parpadeo del led. Conectamos el Arduino por USB. Deberemos comprobar que hemos seleccionado correctamente el puerto USB, en mi caso es el /dev/ttyACM0 tal y como se ve en la imagen.



Cargaremos el programa Blink.



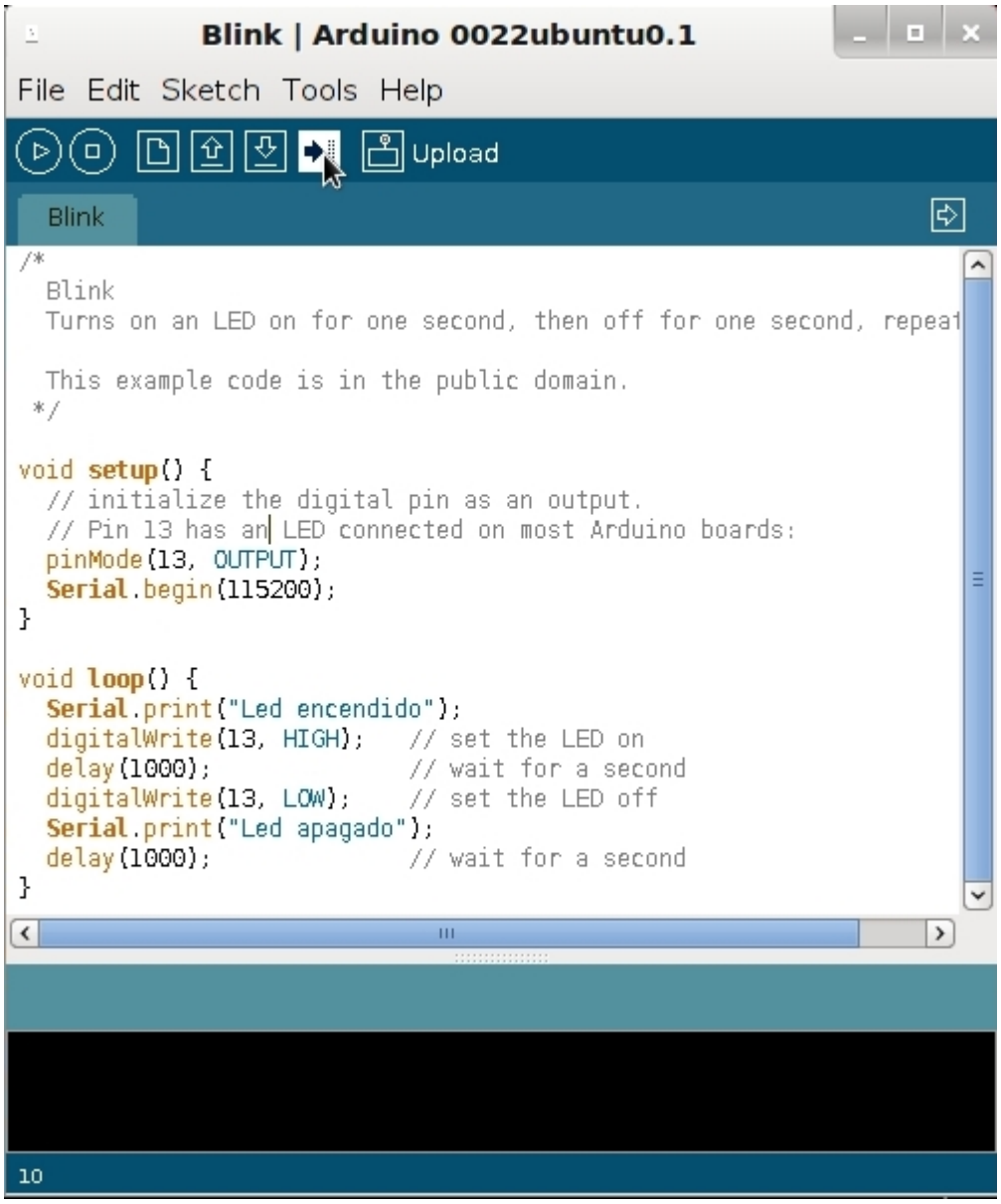
Haremos unas pequeñas modificaciones para poder obtener feedback de lo que esta ocurriendo. Processing nos dirá que el código es read-only, simplemente guardamos el código con un nuevo nombre y listo.

```
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.
```

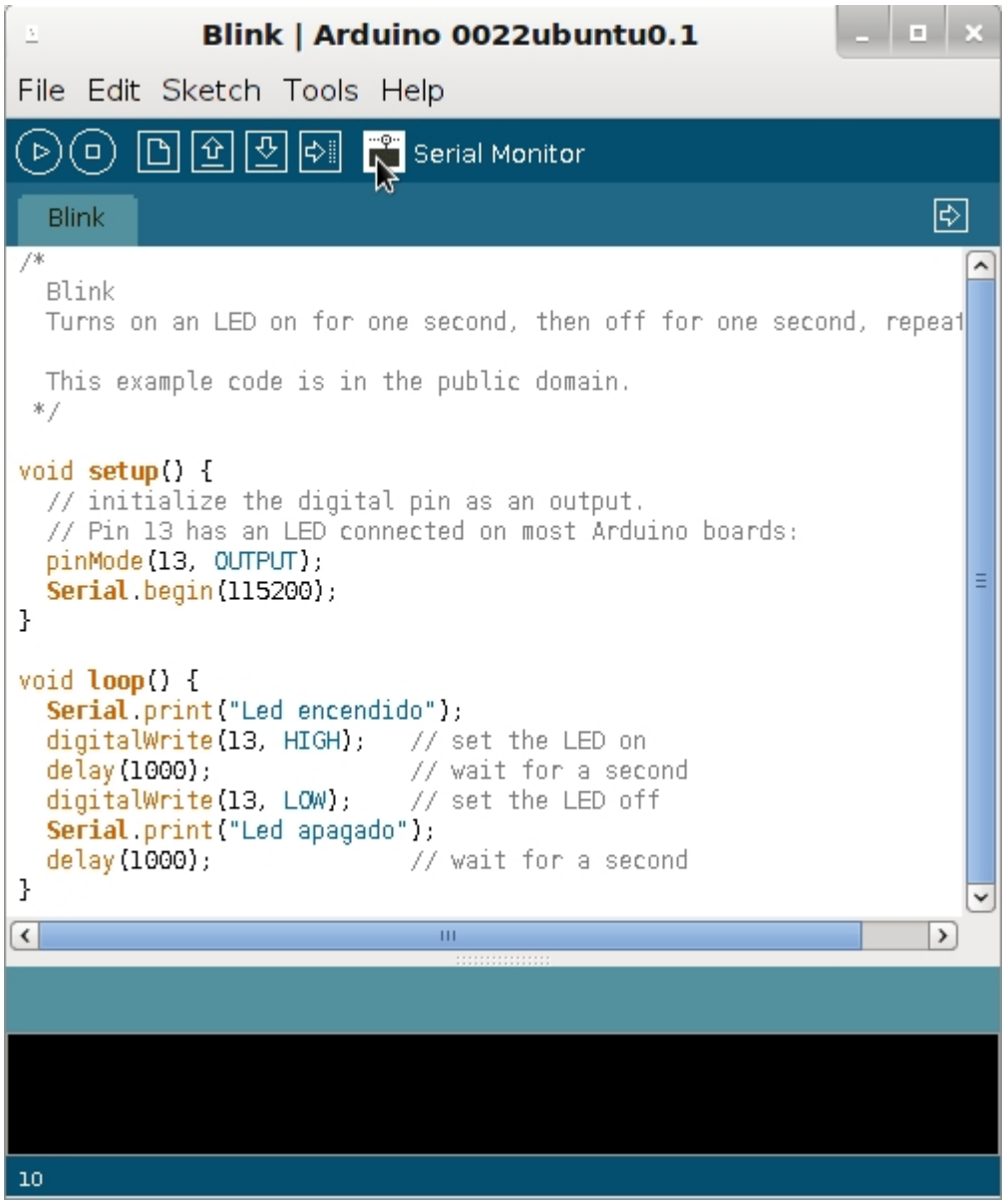
```
This example code is in the public domain.
*/

void setup() {
// initialize the digital pin as an output.
// Pin 13 has an LED connected on most Arduino boards:
pinMode(13, OUTPUT);
Serial.begin(115200); //Establecemos la velocidad de envio de
datos
}

void loop() {
Serial.print("Led encendido");
digitalWrite(13, HIGH); // set the LED on
delay(1000); // wait for a second
Serial.print("Led apagado");
digitalWrite(13, LOW); // set the LED off
delay(1000); // wait for a second
}
```

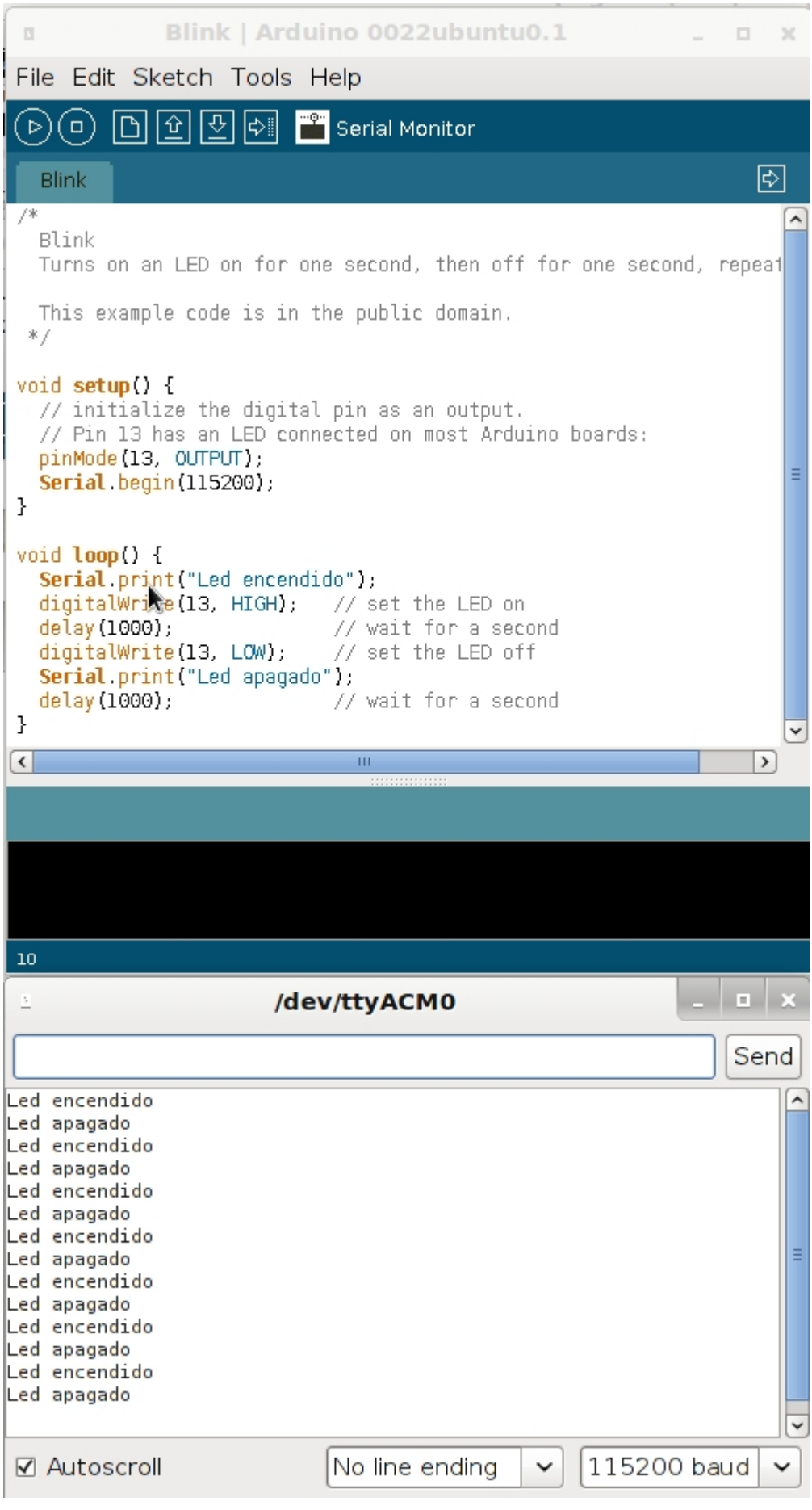
Subir el programa a Arduino y arrancar el Serial Monitor

Subiremos el programa tal y como se ve en la imagen.



Una vez cargado el programa arrancamos el Serial Monitor y podremos ver el feedback que recibimos del Arduino al mismo tiempo que nuestro led empieza a parpadear. Debemos tener en cuenta que la velocidad que hemos indicado en el codigo Serial.begin(115200) sea la misma con la que escucha el Serial monitor(Desplegables inferior derecho).


Blink | Arduino 0022ubuntu0.1
File Edit Sketch Tools Help
Serial Monitor
Blink
/*
Blink
Turns on an LED on for one second, then off for one second, repeat
This example code is in the public domain.
*/
void setup() {
// initialize the digital pin as an output.
// Pin 13 has an LED connected on most Arduino boards:
pinMode(13, OUTPUT);
Serial.begin(115200);
}
void loop() {
Serial.print("Led encendido");
digitalWrite(13, HIGH); // set the LED on
delay(1000); // wait for a second
digitalWrite(13, LOW); // set the LED off
Serial.print("Led apagado");
delay(1000); // wait for a second
}
10


Blink | Arduino 0022ubuntu0.1
File Edit Sketch Tools Help
Serial Monitor
Blink
/*
Blink
Turns on an LED on for one second, then off for one second, repeat
This example code is in the public domain.
*/
void setup() {
// initialize the digital pin as an output.
// Pin 13 has an LED connected on most Arduino boards:
pinMode(13, OUTPUT);
Serial.begin(115200);
}
void loop() {
Serial.print("Led encendido");
digitalWrite(13, HIGH); // set the LED on
delay(1000); // wait for a second
digitalWrite(13, LOW); // set the LED off
Serial.print("Led apagado");
delay(1000); // wait for a second
}
10
/dev/ttyACM0
Send
Led encendido
Led apagado
Led encendido
Led apagado
Led encendido
Led apagado
Led encendido
Led apagado
Led encendido
Led apagado
Led encendido
Led apagado
Led encendido
Led apagado
Autoscroll
No line ending
115200 baud

Observaciones

Con este primer post podemos tener un primer contacto con este espectacular juguete. Las posibilidades son muchas y se pueden encontrar miles de proyectos en la red. Próximamente, mostraré un proyecto un poco más avanzado son Arduino y más adelante el compañero Scuraki continuará con la tercera parte.

Fuentes

- Arduino Blink
- Wikipedia – Arduino

Ruben

C/C++ en Linux

Introducción

En este post voy a comentar alguna librerías y otros útiles para la programación de c/c++ en GNU\Linux. Cuando trabajamos con GNU\Linux todo programa debe tener un adecuado archivo de configuración, debe aceptar parámetros, debe poder recibir señales y actuar en función de ellas, así mismo también debería implementar un sistema de logs. También puede ser interesante acceder a MySql o tener un pequeño servidor web para mostrar información en una simple web html sin necesidad de instalar un Apache entero.

Trataremos

- Señales -> Mostraremos un pequeño codigo con el que podemos captar señales y de esta manera actuar en consecuencia.
- Dotconf -> Mostraremos un ejemplo de esta librería que biene a ser un parser para archivos de configuración estandar.
- Log4cxx -> Esta liberí a nos permite trabajar con logs comodamente, se bas en un xml de configuración donde podemos especificar como será formatación de las cadenas de loggin por pantalla hasta la configuración de logs rotativos escritos a disco.
- MySql – libmysql++ -> Con esta librería podremos realizar operaciones sobre bases de datos MySql.
- Swill Server -> Esta librería nos permitirá levantar un pequeño servidor web.
- Jansson -> Esta librería nos permitirá parsear estructuras fácilmente.

Cabe decir que no se va profundizar, esto pretende ser una aproximación a buenas «maneras» de trabajar de programar con GNU\Linux.

Señales

En esencia una señal es una notificación asíncrona enviada a un a un programa. En caso de que no se haya programado un handle el programa será el que trate las señal, sinó se ejecutará la accción por defecto para esa señal. En este ejemplo veremos como programar el handle para poder manejar las señales que recibamos.

Bien para compilarlo: g++ main.cpp -o signal

Codigo:

```
*
* main.cpp
* Author: ruben
*/

#include
#include
#include
#include
#include

void kctrlc(int signum){
printf("He recibido un ctrl + c (%d). Deberia abortar lo que
este haciendo ahora.\n", signum);
}

void ksigusr(int signum){
printf("He recibido un sigusr 1 o 2 (%d)\n", signum);
}

void ksigt(int signum){
printf("He recibido un sigterm (%d). Deberia acabar lo que
este haciendo...\n", signum);
}

void ksigalrm(int signum){
printf("He recibido un sigalrm (%d).\n", signum);
}
void ksighup(int signum){
printf("He recibido un sighup (%d). Deberia releer la
conmfiguracion de sete programa.\n", signum);
}

void kgeneric(int signum){
printf("He recibido otra senal (%d)\n", signum);
}

void ksigsegv(int signum){
printf("Segmentation fault (%d).\n", signum);
```

```
exit(1);
}

void ksigbus(int signum){
printf("bus error (%d).\n", signum);
exit(2);
}

/* Esta funcion es un distribuidor, capta las senales y en
funcion de eso
* llama a una u otra funcion.
*/
void ksighandler(int signum){
switch (signum){
case SIGINT:
kctrlc(signum);
break;
case SIGUSR1:
case SIGUSR2:
ksigusr(signum);
break;
case SIGTERM:
ksigt(signum);
break;
case SIGHUP:
ksighup(signum);
break;
case SIGALRM:
ksigalrm(signum);
break;
case SIGSEGV:
ksigsegv(signum);
break;
case SIGBUS:
ksigbus(signum);
break;
default:
```

```
kgeneric(signum);
break;
}
/* reprogramamos la senal que ha llegado para que vuelva ha
hacer lo mismo
*/
signal(signum, ksighandler);
}

int main(){
int i;
/* haciendo 'kill -l' veremos que hay 64 senales... les
reprogramaremos
* para que cuando llegue una se ejecute la funcion la funcio
ksighandler.
*/
for (i = 1; i <= 64; i++){ signal(i, ksighandler); } /*
programamos una alarma para que en 30 segundons envie un
SIGALRM (14) * al proceso */ alarm(30); /* con este bucle
infinito veremos el pid del proceso, para * poder enviarle las
senales que queramos con 'kill -num $PID' o */ while (1){
printf("Soy el pid %d y espero senales\n", getpid());
sleep(2); } }
```

Dotconf

Esta librería nos permitirá manejar fácilmente archivos standar de configuración.

Para compilar: g++ conf.cpp -o conf -ldotconf

Archvo configuración

```
# the default behaviour for dot.conf is to stop parsing as
# soon as the first unquoted, unescaped #-sign is found
# you can override it by giving the flag NO_INLINE_COMMENTS to
dotconf_create()

var1 'upper case' #inline comment 1
var2 '1' # inline comment 2
```

```
var3 '2' # inline comment 3
var4 '3' #inline comment 4
```

Archivo fuente

```
/*
* main.cpp
* Author: ruben
*/

#include
#include
#include

/*
tabsize: 4
shiftwidth: 4
*/

DOTCONF_CB(cb_noinline){
int i;
int j=0;
char sport[200];
char sport2[200];
char sport3[200];
char sport4[200];

printf("[test.conf] Have %d args\n", cmd->arg_count);

for (i = 0; i < cmd->arg_count; i++){

if(j==0){
strcpy (sport,cmd->data.list[i]);
printf("Arg: %s\n", sport);
j=1;
}else if (j== 1){
strcpy (sport2,cmd->data.list[i]);
printf("Arg: %s\n", sport2);
j=2;
```

```
}else if(j==2){
strcpy (sport3,cmd->data.list[i]);
printf("Arg: %s\n", sport3);
j=3;
}else if(j==3){
strcpy (sport4,cmd->data.list[i]);
printf("Arg: %s\n", sport4);
j=0;
}
}

return NULL;
}

static configoption_t options[] = {
{"var1", ARG_LIST, cb_noinline, NULL, 0},
{"var2", ARG_LIST, cb_noinline, NULL, 0},
{"var3", ARG_LIST, cb_noinline, NULL, 0},
{"var4", ARG_LIST, cb_noinline, NULL, 0},
LAST_OPTION
};

void readit(int flags){
configfile_t *configfile;

configfile = dotconf_create("test.conf", options, 0, flags);
if (!dotconf_command_loop(configfile))
fprintf(stderr, "Error reading config file\n");
dotconf_cleanup(configfile);
}

int main(int argc, char **argv){
//printf("Reading the configuration with NO_INLINE_COMMENTS
enabled\n");
//readit(NO_INLINE_COMMENTS);

//printf("\n\n");
printf("Reading the configuration\n");
readit(0);
```

```
//printf("%s\n",cmd->data.list[0]);

return 0;
}
```

Log4cxx

Esta librería nos permitirá gestionar los log's de nuestra aplicación adecuadamente.

Para compilar: g++ log.cpp -I/opt/include /usr/lib/liblog4cxx.a -lapr-1 -laprutil-1

Archvo configuración

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
#include
#include

using namespace log4cxx;
using namespace log4cxx::xml;
using namespace log4cxx::helpers;
```

```
// Define static logger variable
LoggerPtr loggerMyMain(Logger::getLogger( "main"));
LoggerPtr logtest(Logger::getLogger( "Function A"));

void functionA(){
LOG4CXX_INFO(logtest, "Executing functionA.");
LOG4CXX_INFO(logtest, "exiting functionA.");
}

int main(){
// Load configuration file
DOMConfigurator::configure("conf.xml");

LOG4CXX_TRACE(loggerMyMain, "this is a debug message for detailed code discovery.");
LOG4CXX_DEBUG(loggerMyMain, "this is a debug message.");
LOG4CXX_INFO (loggerMyMain, "this is a info message, ignore.");
LOG4CXX_WARN (loggerMyMain, "this is a warn message, not too bad.");
LOG4CXX_ERROR(loggerMyMain, "this is a error message, something serious is happening.");
LOG4CXX_FATAL(loggerMyMain, "this is a fatal message!!!");
functionA();

return 0;
}
```

MySql

Esta librería nos permitirá acceder y utilizar bases de datos Mysql comodamente.

Archvo configuración sql

```
-- phpMyAdmin SQL Dump
-- version 3.4.10.1deb1
```

```
-- http://www.phpmyadmin.net
--
-- Servidor: localhost
-- Temps de generació: 31-08-2012 a les 15:27:03
-- Versió del servidor: 5.5.24
-- Versió de PHP : 5.3.10-1ubuntu3.2

SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT
*/;
/*!40101                                                  SET
@OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION
*/;
/*!40101 SET NAMES utf8 */;

--
-- Base de dades: `emp`
--

-- --------------------------------------------------------

--
-- Estructura de la taula `emp`
--

CREATE TABLE IF NOT EXISTS `emp` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`name` text NOT NULL,
`surname` text NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=3 ;

--
-- Bolcant dades de la taula `emp`
--
```

```
INSERT INTO `emp` (`id`, `name`, `surname`) VALUES
(1, 'test', 'test1'),
(2, 'test2', 'test3');

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT
*/;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS
*/;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION
*/;

Para compilar: gcc conexion.cpp main.cpp -o conexion2 -Wno-
deprecated -I/usr/include/mysql -L/usr/lib/mysql -lmysqlclient
-lstdc++ -lz
/*
* main.cpp
* Author: ruben
*/

#include
#include
#include

int main(){

MYSQL_RES *result;
MYSQL_ROW row;
MYSQL *connection, mysql;

int state;

mysql_init(&mysql);

connection                                                    =
mysql_real_connect(&mysql,"localhost","user","pass","emp",0,0,
0);

if (connection == NULL){
printf("Error: %s\n",mysql_error(&;mysql));
return 1;
```

```
}

state = mysql_query(connection, "SELECT * FROM emp");

if (state !=0){
printf("Error: %s\n",mysql_error(connection));
return 1;
}

result = mysql_store_result(connection);
printf("Rows: %d\n",mysql_num_rows(result));

while ( ( row=mysql_fetch_row(result)) != NULL ){
printf("id -> %s, name -> %s, surname -> %s\n", (row[0] ?
row[0] : "NULL"), (row[1] ? row[1] : "NULL"),(row[2] ? row[2]
: "NULL"));
}

mysql_free_result(result);
mysql_close(connection);

return 0;
};
```

Sqlite3

No siempre enecesitamos guardar datos en una gestor como pueda ser MySql, aveces no requerimos de todo lo que nos puede ofrecer y con SqLite podemos tener una compensación entre recursos consumidos y funcionalidad muy buena

Archvo configuración sql

```
BEGIN TRANSACTION;
CREATE TABLE emp (id TEXT, name TEXT, surname TEXT);
INSERT INTO emp VALUES(1000,'test','test2');
COMMIT;

Para compilar: g++ main.cpp -o sqlite -lsqlite3
/*
```

```
* main.cpp
* Author: ruben
*/

#include "sqlite3.h"
#include
#include
#include
#include

#include
#include
#include

#include
#include

char missatge[1000];
char sentence[1000];
sqlite3 *db;
char *zErrMsg = 0;
int rc;

char **result;

int nrow;
int ncol;
int i;

void insertar(){

rc = sqlite3_open("emp", &db);

if( rc ){
sprintf(missatge, "Test: Can't open database %s to insert row
for DB persistence\n", sqlite3_errmsg(db));
sqlite3_close(db);
exit(1);
}
```

```
i++;

sprintf(sentence, "insert into emp ('id', 'name', 'surname')
values ('%d', 'test', 'test2')", i);

rc = sqlite3_get_table(
db,
sentence,
&result,
&nrow,
&ncol,
&zErrMsg
);

sqlite3_free_table(result);
sqlite3_close(db);

if (zErrMsg != NULL){
printf("Error:%s\n",zErrMsg);
std::cin.ignore(); // ignores the newline
std::cin.get(); //waits for character
}else{
printf("Insertado correctamente\n");
std::cin.ignore(); // ignores the newline
std::cin.get(); //waits for character
}
}

void borrar(){

rc = sqlite3_open("emp", &db);

if( rc ){
sprintf(missatge, "SpoolAndScheduler: Can't open database %s
to delete row for DB persistence\n", sqlite3_errmsg(db));
sqlite3_close(db);
exit(1);
}
```

```
std::string cons="delete from emp where id='1'";

rc = sqlite3_get_table(
db,
cons.c_str(),
&result,
&nrow,
&ncol,
&zErrMsg
);

sqlite3_free_table(result);
sqlite3_close(db);

if (zErrMsg != NULL){
printf("Error:%s\n",zErrMsg);
std::cin.ignore(); // ignores the newline
std::cin.get(); //waits for character
}else{
printf("Borrado correctamente\n");
std::cin.ignore(); // ignores the newline
std::cin.get(); //waits for character
}

}

void listar(){

rc = sqlite3_open("emp", &db);

if( rc ){
sprintf(missatge, "SpoolAndScheduler: Can't open database %s
to select rows for DB persistence\n", sqlite3_errmsg(db));
sqlite3_close(db);
exit(1);
}

std::string cons="select * from emp";//where ds='as'";

rc = sqlite3_get_table(
```

```
db,
cons.c_str(),
&result,
&nrow,
&ncol,
&zErrMsg
);

printf("\n --------RESULT-------- \n");

//printf("Voltes->%d\n",(nrow*ncol)+ncol);
int j=(nrow*ncol)+ncol;

for(i=0 ; i < j; i=i+3){ if(i>=3){
printf("i->%d - ",i);
printf("id: %s - ",result[i]);
printf("name: %s - ",result[i+1]);
printf("surname: %s\n",result[i+2]);
}
}

sqlite3_free_table(result);

sqlite3_close(db);

if (zErrMsg != NULL){
printf("Error:%s\n",zErrMsg);
std::cin.ignore(); // ignores the newline
std::cin.get(); //waits for character
}else{
std::cin.ignore(); // ignores the newline
std::cin.get(); //waits for character
}
}

int main(){

int fin = 0;
int opcion;
```

```
while (fin == 0){
system("clear");
printf("Menú simple\n\n");

printf("\t1] Insertar.\n");
printf("\t2] Borrar.\n");
printf("\t3] Listar.\n");
printf("\t4] Sortir.\n");

printf("\n\nOpción: ");
scanf("%i", &opcion);

switch(opcion){
case 1:
insertar();
break;
case 2:
borrar();
break;
case 3:
listar();
break;
case 4:
fin = 1;
break;
default:
fin = 0;
break;
}

}

return 0;
}
```

Swill Server

SWILL (***S**imple **W**eb **I**nterface **L**ink **L**ibrary*) es una librería que nos permite de manera fácil añadir una interfaz web a un

programa en C/C++. Deberemos bajarnos la librería y ejecutar los siguientes comandos para descomprimirla desde Swill e instalarla:

```
tar xvzf swill-0.3.tgz
cd SWILL-0.3
./configure
make
make install
```

Para compilar: g++ main.cpp -o swill_server -I/usr/local/include/swill/ -lswill

```
/*
* main.cpp
* Author: ruben
*/

#include

void count_to_ten(FILE *f) {
int i;
for (i = 0; i < 10; i++) { fprintf(f,"%d\n", i); } } int main() { swill_init(8181); swill_handle("ten.txt", count_to_ten, 0); swill_file("index.html",0); while (1) { swill_serve(); } swill_shutdown(); }
```

Jansson

Esta librería nos permitirá manejar fácilmente archivos json.

Para compilar: gcc main.c -o json -ljansson

Archvo json

```
{
"errors": [
{
```

```
"id": "1",
"message": "string not found"
},
{
"id": "2",
"message": "system error"
}

]
}
```

Archivo fuente

```
/*
* Author: Ruben
*/

#include
#include

#include

#define BUFFER_SIZE (256 * 1024) /* 256 KB */

#define                                          URL_FORMAT
"http://github.com/api/v2/json/commits/list/%s/%s/master"
#define URL_SIZE 256

/* Return the offset of the first newline in text or the
length of
text if there's no newline */
static int newline_offset(const char *text){
const char *newline = strchr(text, '\n');
if(!newline)
return strlen(text);
else
return (int)(newline - text);
}
```

```
struct write_result{
char *data;
int pos;
};

int main(int argc, char *argv[]){
size_t i;

json_t *root;
json_error_t error;
json_t *errors;

root = json_load_file("file.json", 0, &error);

if(!root){
fprintf(stderr, "error: on line %d: %s\n", error.line,
error.text);
return 1;
}

errors = json_object_get(root, "errors");
if(!json_is_array(errors)){
fprintf(stderr, "error: errors is not an array\n");
return 1;
}

for(i = 0; i < json_array_size(errors); i++){ json_t *error,
*id, *message; const char *message_text; error =
json_array_get(errors, i); if(!json_is_object(error)){
fprintf(stderr, "error: error %d is not an object\n", i + 1);
return 1; } id = json_object_get(error, "id");
if(!json_is_string(id)){ fprintf(stderr, "error: error %d: id
is not a string\n", i + 1); return 1; } message =
json_object_get(error, "message");
if(!json_is_string(message)) { fprintf(stderr, "error: error
%d: message is not a string\n", i + 1); return 1; }
message_text = json_string_value(message); printf("%.8s
%.*s\n", json_string_value(id), newline_offset(message_text),
message_text); } json_decref(root); return 0; }
```

Observaciones
Bien, espero que esto pueda ser de ayuda. No es más que un poco de cada. Pero al fin y al cabo estas utilidades siempre puede ir bien.

Ruben

Python XMPP Client (Twisted)

Introducción

En este post vamos a realizar un cliente de xmpp en python. Es de obligada mención Twisted Matrix, una increíble framework para python el cual facilita muchíssimo la programación enfocada a redes. En cuanto a protocolos podemos trabajar con HTTP, XMPP, NNTP, IMAP, SSH, IRC, FTP y muchos más, también podemos trabajar con varias arquitecturas (TCP, UDP, SSL/TLS, IP Multicast, Unix domain sockets).

Características

- Separación de los protocolos y transportes

El diseño de Twisted se basa en la separación completa entre los protocolos lógicos (que por lo general dependen de la conexión semántica basada en streams –flujos–, como el HTTP o POP3) y el transporte en capas físicas soportado como la semántica basada en streams (como archivos, bibliotecas sockets o SSL). La conexión entre un protocolo lógico y una capa de transporte que ocurre en el último momento posible, justo antes de la información se pase a la instancia de protocolo lógico. El protocolo lógico es informado de la instancia de capa de transporte, y puede utilizarlo para

enviar mensajes de un lado para comprobar la identidad del otro extremo. Tenga en cuenta que todavía es posible, en el código de protocolo, para consultar profundamente la capa de transporte en cuestiones de transporte (como la comprobación de un certificado SSL del lado del cliente). Naturalmente, el código de dicho protocolo, se producirá un error (lanzar una excepción) si la capa de transporte no es compatible con tales semánticas.

- Deferreds

El modelo central de aplicación para Twisted es el concepto de un deferred (predefinir algo que se usara como valor futuro). Un deferred es un valor que no se ha calculado todavía, por ejemplo, porque las necesidades de datos desde un equipo remoto. Los deferreds se pueden transferir, al igual que los objetos normales, pero no se puede pedir por su valor. Cada deferred es compatible con una cadena de devolución de llamada. Cuando el deferred toma el valor, es transferido a través de la cadena de devolución de llamada, con el resultado de cada de callback (devolución) siendo la entrada (input) para la siguiente. Esto permite que operen en los valores de un deferred sin saber lo que son. Por ejemplo, si un deferred devuelve una cadena desde un equipo remoto con una dirección IP en formato quad, un callback se puede adjuntar para traducirla a un número de 32 bits. Cualquier usuario del deferred puede ahora tratarlo como deferred de retorno de un número de 32 bits. Esto, y la capacidad de relación para definir «errbacks» (callbacks que son llamados como controladores de errores), permite que el código que se ve como si fuera de serie, mientras que todavía mantiene la abstracción por eventos.

- Soporte de Thread (hilos o subprocesos)

Twisted soporta una abstracción sobre threads en crudo usando un thread como una fuente deferred. Por lo tanto, un deferred que es retornado inmediatamente, recibirá un valor cuando

finalice el thread. Los callbacks se pueden adjuntar cuando corran en el thread principal, a fin de aliviar la necesidad de soluciones complejas de bloqueo. Un buen ejemplo de tal uso, que viene de las bibliotecas de soporte de Twisted, es usar este modelo para llamadas en bases de datos. La llamada de la base de datos misma pasa de un thread exterior, pero el análisis del resultado que sucede en el thread principal.

- Soporte de bucle de externos

Twisted se puede integrar con bucles de eventos externos, tales como los de GTK+, Qt y Cocoa (a través de PyObjC). Esto le permite el uso de Twisted como la capa de soporte de red en aplicaciones GUI, usando todas sus colecciones sin tener que añadir una sobrecarga de thread-por-socket, como lo haria cualquier biblioteca nativa de Python. Se pude integrar en proceso un completo web server con una aplicación interfaz gráfica utilizando este modelo, por ejemplo.

Cabe decir que no es objetivo de este post profundizar en twisted matrix, pues se necesitarían sería muy extenso (de echo harían falta muchos :p). También hace falta añadir, que hay una muy buena documentación en la página oficial.

Instalación

Podemos descargar el framework des de la página oficial:
Twisted Matrix Downloads
O bien odemos instalarlo fácilmente en Debian y derivados, puesto que esta en los repositorios (sino en el link de la página oficial antes citada te facilitan la PPA):

```
$>sudo apt-get install python-twisted python-twisted-words
```

El modulo twisted-words contiene las librerías de jabber.

Codigo del Cliente

```
#!/usr/bin/python
```

```
# Twisted Imports
from twisted.words.protocols.jabber import client, jid ,
xmlstream
from twisted.words.xish import domish
from twisted.internet import reactor

name = None
server = None
resource = None
password = None
me = None

thexmlstream = None
tryandregister = 1

def initOnline(xmlstream):
     # creamos los observadores hacia las respuestas xml
message y una general (podemos incluir presence, iq...)
    global factory
    print 'Initializing...'
    xmlstream.addObserver('/message', gotMessage)
    xmlstream.addObserver('/*', gotSomething)

def authd(xmlstream):
    # Autentificacion
    global thexmlstream
    thexmlstream = xmlstream
    print "we've authd!"
    print repr(xmlstream)

    # se envia la presencia a los demas clientes
    presence = domish.Element(('jabber:client', 'presence'))
    presence.addElement('status').addContent('Online')
    xmlstream.send(presence)

    initOnline(xmlstream)

def send(author, to, msg):
    # esta funcion envia los mensajes
    global thexmlstream
    message = domish.Element(('jabber:client','message'))
```

```
    message["to"] = jid.JID(to).full()
    message["from"] = jid.JID(author).full()
    message["type"] = "chat"
     message.addElement("body", "jabber:client", msg+ "- ya lo
sabia adicotalainformatica1")

    thexmlstream.send(message)

def gotMessage(el):
    # esta funcion parsea los mensajes recibidos
    global me
    # print 'Got message: %s' % str(el.attributes)
    from_id = el["from"]

    body = "empty"
    for e in el.elements():
        if e.name == "body":
            body = unicode(e.__str__())
        break

    send(me, from_id, body)

def gotSomething(el):
    # Observador general
      print 'Got something: %s -> %s' % (el.name,
str(el.attributes))

def authfailedEvent(xmlstream):
    global reactor
    print 'Auth failed!'
    reactor.stop()

def invaliduserEvent(self,xmlstream):
    print "Invalid User"

def registerfailedEvent(self,xmlstream):
    print 'Register failed!'

if __name__ == '__main__':
    #Parametrizamos la conexion
    PASSWORD = '123456'
```

```
    myJid = jid.JID('adictoalainformatica2@antitot-linux')
    me = 'adictoalainformatica2@antitot-linux'
    factory = client.XMPPClientFactory(myJid, PASSWORD)

    # Registramos las callbacks de autentificacion
    print 'register callbacks'
    factory.addBootstrap(xmlstream.STREAM_AUTHD_EVENT, authd)
factory.addBootstrap(client.BasicAuthenticator.INVALID_USER_EV
ENT, invaliduserEvent)
factory.addBootstrap(client.BasicAuthenticator.AUTH_FAILED_EVE
NT, authfailedEvent)
factory.addBootstrap(client.BasicAuthenticator.REGISTER_FAILED
_EVENT, registerfailedEvent)

    # Paarametrizamos y arrancamos el reactor (encargado de
mantener y gestionar las callbacks
    # producidas por las acciones de la conexion)
    reactor.connectTCP('localhost', 5222, factory)
```

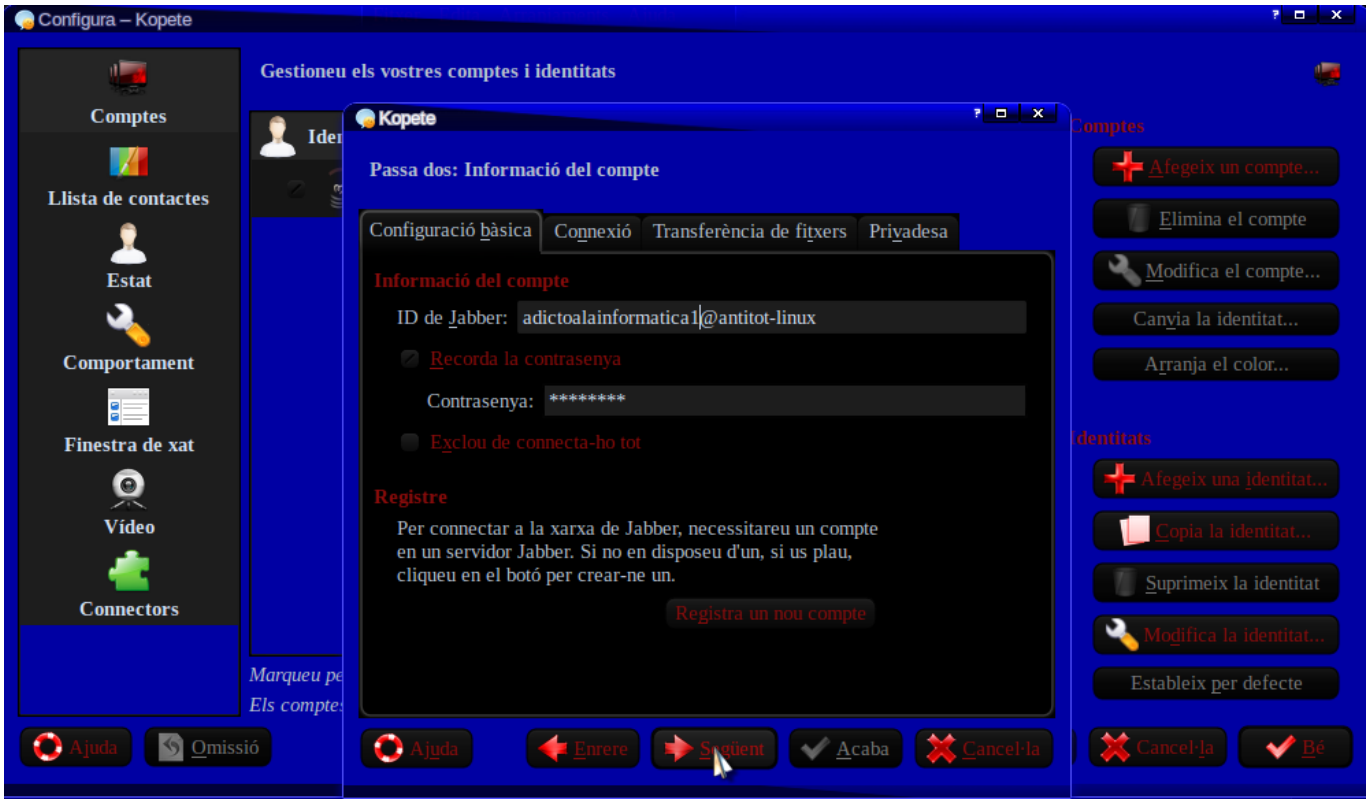
Test

Para realizar el test haremos referencia al post anteriror sobre xmpp donde configuramos y creamos dos usuarios (adictosalainformatica1, adictosalainformatica2) con el servidor openfire. Aprovecharemos los usuarios
para configurar el script e indicaremos al reactor donde esta el servidor xmpp:

```
#Parametrizamos la conexion
PASSWORD = '123456'
myJid = jid.JID('adictoalainformatica2@antitot-linux')
me = 'adictoalainformatica2@antitot-linux'
factory = client.XMPPClientFactory(myJid, PASSWORD)
reactor.connectTCP('antitot-linux', 5222, factory)
```

y cualquier cliente como por ejemplo Kopete. Abriremos Kopete e hiremos a configuraciones -> configura. Se abrirá una nueva ventana en esta pincharemos en añadir nuevo elemento.
Seguidamente deberemos escoger el protocolo:

Configuramos la cuenta:



Y ya esta, cuando des del usuario adictosalainformatica enviemos un mensaje a nuestro script éste le responderá el mismo mensaje con la coletilla » – ya lo sabia adicotalainformatica2″.

Código del cliente Xmpp

Puedes descargar el código de este post desde la cuenta de GitHub

Observaciones

Y con este post cierro el tema de redes xmpp dando una vista superficial pero intuitiva de lo que puede ofrecernos este nuevo protocolo tanto a nivel de mensajería como a nivel de distribuidor de comandos. espero que haya sido de ayuda.

Fuentes

- Twisted Matrix
- Wikipedia – Twisted Matrix

Ruben

XMPP – OpenFire

Introducción

OpenFire es un servidor de XMPP programadfo en java (multiplataforma) y con un como WebApp para administrarlo. Se encuentra bajo licencia GPL y con el puedes administrar a tus usuarios, compartir archivos, auditar mensajes, mensajes offline, mensajes broadcast, grupos, etc y además contiene plugins para ampliar sus funcionalidades.

Características

Openfire implementa las siguientes características:

- Panel de administración web
- Interfaz para agregar plugins
- SSL/TLS
- Conferencias
- Interacción con MSN, Google Talk, Yahoo messenger, AIM, ICQ
- Estadísticas del Servidor, mensajes, paquetes, etc.
- Cluster con múltiples servidores
- Transferencia de Archivos
- Compresión de datos
- Tarjetas personales con Avatar
- Mensajes offline
- Favoritos
- Autenticación vía Certificados, Kerbeos, LDAP, PAM y Radius
- Almacenamiento en Active Directory, LDAP, MS SQL, MySQL, Oracle y PostgreSQL
- SASL: ANONYMOUS, DIGEST-MD5 y Plain

Instalación

Tan simple como descargar OpenFire Downloads (en nuestro manual un deb):

```
# wget http://www.igniterealtime.org/downloads/download-landing.jsp?file=openfire/openfire_3.6.4_all.deb
```

Luego podemos instalarlo fácilmente:

```
# sudo dpkg -i openfire_3.6.4_all.deb
```

Configuración

En primera instancia debemos seguir el intuitivo instalador des de la siguiente url:

```
http://localhost:9090
```

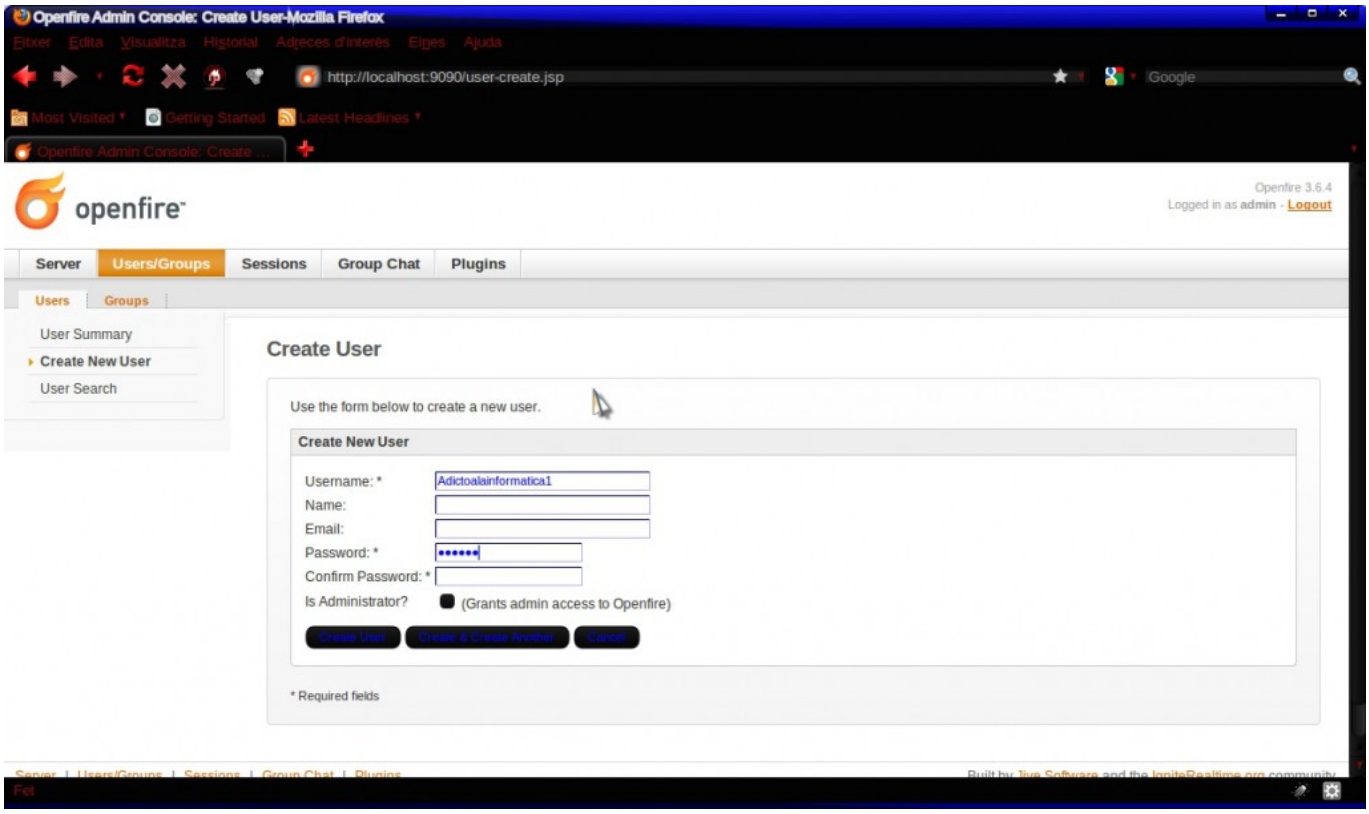
Por último deberemos realizar un restart del servidor para poder acceder al WebApp con el nuevo usuario administrador:

```
# /etc/init.d/openfire restart
```

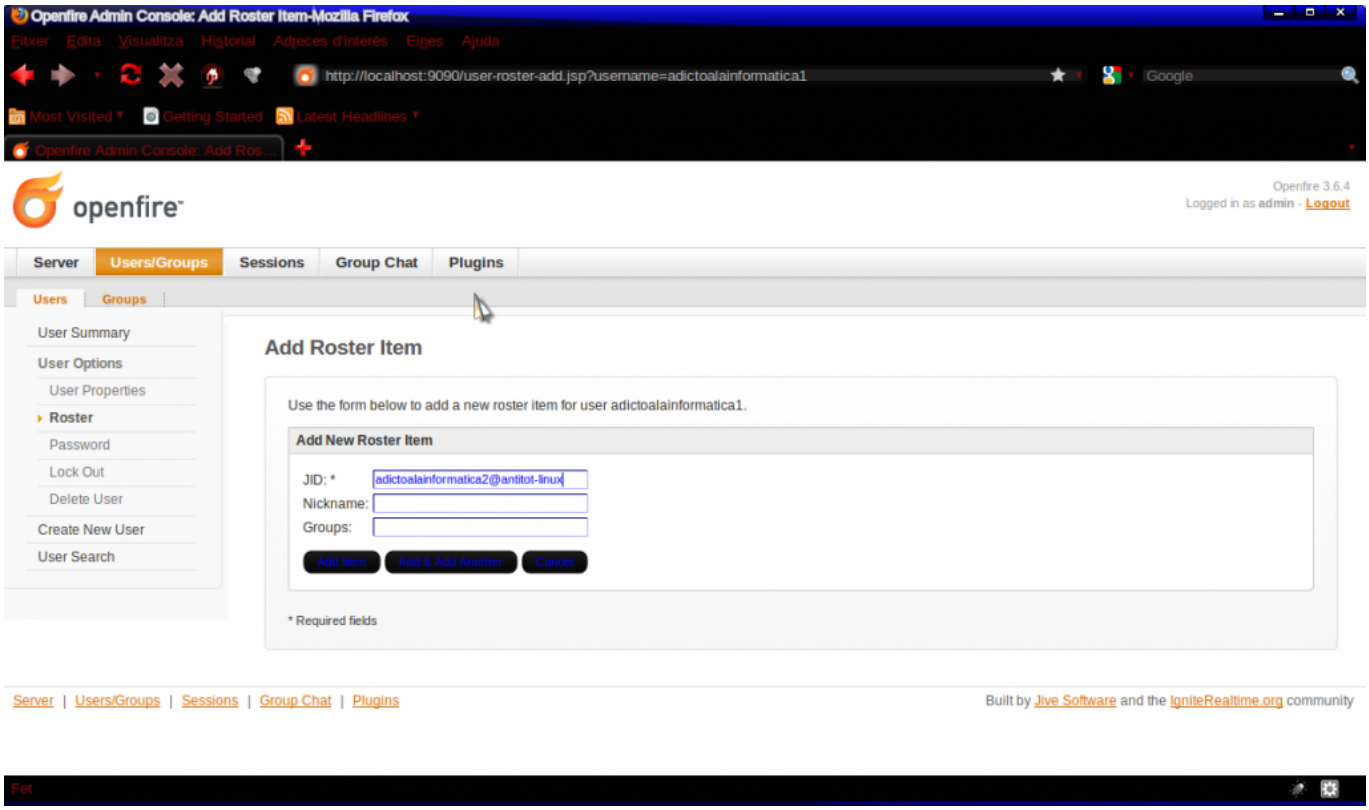
Creación de usuarios

Cuando creamos dos usuarios y queremos que estos puedan verse y enviar mensajes debemos añadirlos al roster de cada uno. Es decir, el usuario u1 debe tener en su roster al usuario u2 y u2 debe tener en su roster a u1. De esta manera podrán verse. Luego los dos usuarios deberán tener subscription both, es decir, deben estar subscritos a from y to para poder enviar y recibir mensajes de los usuarios que tienen en su roster.

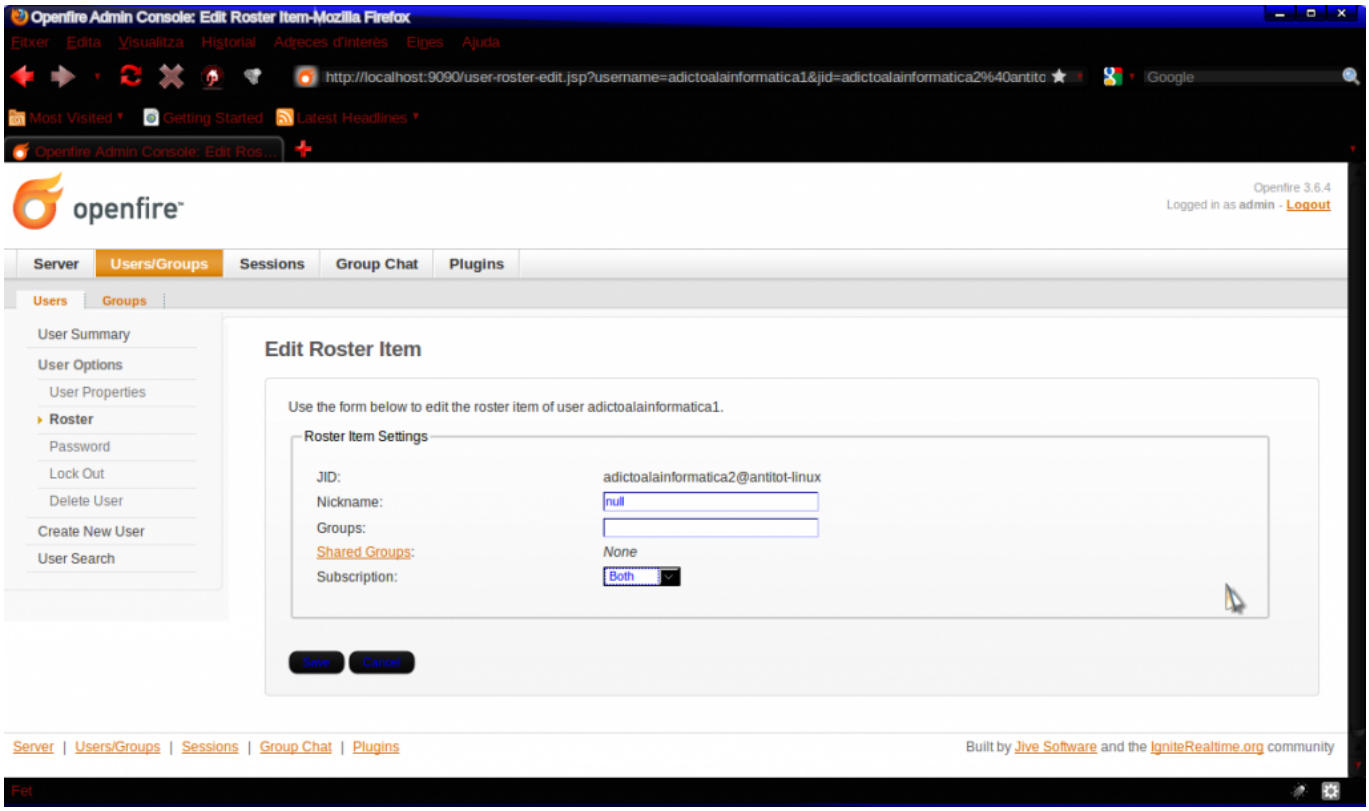
Para acceder a él tan solo debemos acceder mediante navegador web, después acceder a user/group y por último create user:



Crearemos dos usuarios adictoalainformatica1 y adictoalainformatica2 (originalidad antetodo). Despues deberemos entrar en la sección de cada usuario, en roster y por último clicar en Add Item:



Como podemos observar al usuario adictoalainformatica1 le añadimos al roster al usuario adictoalainformatica2 y debemos hacer lo respectivo para el usuario adictoalainformatica2 con adictoalainformatica1. Una vez echo esto deberemos cambiar el subscription para que los usuarios tengan completa «visibilidad» entre ellos. Debemos entrar en la sección del usuario adictoalainformatica1, luego en su roster donde veremos al usuario adictoalainformatica2. Bien, por último deberemos modificarlo y como vemos en la imagen debemos cambiar el subscription a both, en este caso los estamos haciendo para el usuario adictoalainformatica2 con respecto al usuario adictoalainformatica1 y deberemos hacerlo también a la inversa.



Ahora podríamos configurar dos clientes de xmpp (por ejemplo gajim y pidgin) con los dos usuarios y comprobar que pueden enviarse mensajes.

Observaciones

Ya hemos viso un servidor de xmpp, no es el que ofrece más opcionalidades pero para un desarrollo de nivel medio es más que suficiente. En grandes desarrollos donde tengamos una increíble cantidad de usuarios sería recomendable usar Ejabberd. Pero de todos modos con OpenFire tendremos un gran abanico de posibilidades y es mucho mas cómodo de administrar que Ejabberd. Un servidor XMPP puede ser útil tanto como para crear una red de mensajería a nivel interno de empresa como para montar un sistema de distribución de comandos. En el próximo post, ya cerrando con XMPP, mostraremos como crear un cliente con python capaz de conectarse, recibir y enviar estados y enviar y recibir mensajes.

Fuentes

- Wikipedia – Openfire

Ruben

XMPP

Introducción

XMPP(Extensible Messaging and Presence Protocol) es un protocolo estándar y abierto que se basa en el intercambio de mensajes **XML**. Inicialment va ser concebut per implementar xarxes de missatgeria instantànea . Inicialmente fue concebido para implementar redes de mensajería instantánea. Quizás las herramientas más conocidas que usan el protocolo XMPP son Jabber, GTalk y las funciones de videoconferencia y audioconferencia de Google.

Funcionalidades

- Redundancia.
- Escalabilidad.
- No hacen falta VPNs para compartir recursos dentro de una NAT.
- Soporte **SSL** y Certificados
- Backends donde se guardan los usuarios: MySQL, LDAP…
- Extensible (usa lo que se llaman **XEP** – *XMPP Extension Protocols*)
- **BOSH** permite usar XMPP sobre **HTTP**, lo que por diseño del protocolo XMPP sería un problema.

Protocolo

Vamos a desglosar, identificar y explicar las principales opciones que nos ofrece XMPP como protocolo.

- **JID:** los nodos de una red XMPP identifican a través de este identificador, que es de la forma: user @ domain / resource (ejemplo: ruben@adictosalainformatica.com/linux). Tratamiento de los JID:
    - user@example.com – conocido como JID
    - user@example.com/desktop – conocido como JID u **full JID**
- **Stanza:** los mensajes XML que se intercambian entre un servidor XMPP y un cliente se llaman Stanzas. Hay tres tipos de Stanzas:
    - **Messages:** transportan información entre nodos, los mensajes se pueden organizar en *threads.* Los hay de diferentes tipos:
        - normal
        - chat
        - groupchat
        - headline
        - error
    - **Presence:** sirven para informar de la dipsonibilitat de un recurso (online / offline):
        - away
        - do not disturb
        - extended away
        - free for chat
    - **IQ Stanza** (info query): similar a un HTTP GET / POST / PUT, sirve para pedir informaciones concretas a un nodo. Ideales para extender el protocolo. Por ejemplo, las IQ usan para saber qué recursos (usuarios) están conectados a un canal de chat. Los hay de tres tipos:
        - **get:** piden información (HTTP GET)
        - **set:** proveen información (HTTP POST / PUT)
        - **result:** devuelven información requerida o confirman que se ha aceptado un pedido 'set'.
- **Extensibility:** para que sea simple extender el protocolo, las **Stanzas** soportan **namespaces** y cualquier

elemento **XML** de una **stanza** se puede usar como un **payload,** para transportar: **XHTML** tags, Atom feeds, **XML-RPCs**, etc.

- **Roster:** lista de personas que participan en un evento.
- **Presence Subscription:** los recursos de una red (a menudo los usuarios) pueden subscribirse a otros recursos (otros usuarios) para saber si están o no disponibles en cada momento.
- **Asincronismo:** la gracia del XMPP respecto a otros protocolos como HTTP es que se trata de un protocolo asíncrono, o sea, que las conexiones se establecen durante mucho tiempo y en cualquier momento el servidor y / o el cliente pueden enviar y recibir **Stanzas** a través de este canal. Los protocolos HTTP establecen conexiones relativamente cortas donde a menudo sólo hay una petición y una respuesta después se cierra la conexión.

Jingle(add-on)

Jingle es una extensión al protocolo XMPP que permite la transferencia de información p2p. Este protocolo permite transmitir datos multimedia, habilitando servicios de VideoConferencia y de VoIP. Este protocolo fue diseñado inicialmente por Google junto con la XMPP Standards Foundation y liberado (bajo licencia similar a la de BSD).

Observaciones

Bien puede entenderse que este protocolo solo és útil para redes de mensajería instantànea, pero pensemos en sistemas de distribución de comandos. Resolvemos muchas incertidumbres facilmente, tenemos identificados los clientes, podemos enviar comandos a ciertos clientes y saber si estos clientes estan disponibles o no, por lo cual no habra un intento de envio esperando una respuesta de error simplemente no se envia, o si han recibido el comando correctamente, luego mediante respuesta del cliente podemos saber como ha ido la ejecución de cada comando por cada cliente.

Fuentes

- http://en.wikipedia.org/wiki/XMPP
- http://en.wikipedia.org/wiki/Jingle_(protocol)

Ruben

PHP ON COUCH

Introducción

Php On Couch es una librería en php para atacar bases de datos, en adelante bbdd, de CouchDB . Por el uso que le he dado he podido comprobar su estabilidad y ademas se actualiza regularmente, lo cual es de agradecer. Cabe decir también que esta bajo licencia GPL.

En este ejemplo veremos como conectar a una base de datos y extraer la información de un documento, obviamente esta librería ofrece muchísimas más funcionalidades, pero este post no pretende ser una explicación de cada una de ellas. Pretende mostrar una utilización práctica de la librería, en él se creará un gráfico con Google Chart Api a partir de datos guardados en CouchDB.

Estructura

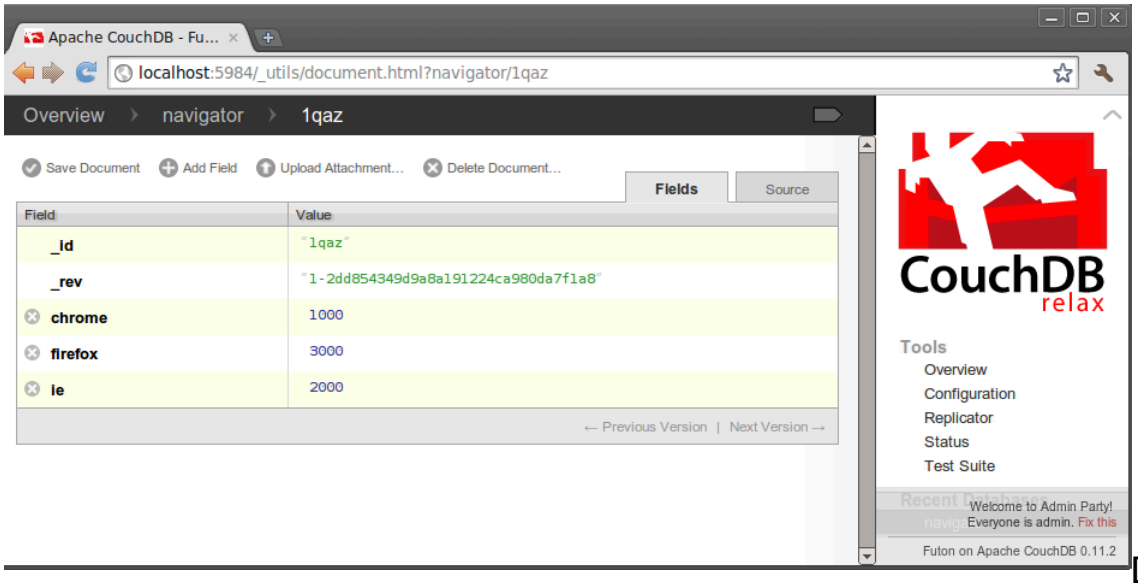
Dispondremos de los siguiente ficheros:

- index.php: archivo principal.
- couch.php: clase que utilizaremos para acceder a CouchDB.
- config.php: archivo de configuración.

Crearemos una base de datos con las siguientes

características:

- El nombre de la base de datos será navigator.
- El id del documento al que accederemos tendrá el siguiente id: 1qaz.
- El documento almacenara un valor para los siguientes tags: ie, firefox y chrome.



Debemos descargarnos la librería Php On CouchDB y estructurar el proyecto conforme la la carpeta lib, de dicha librería, se encuentre en el mismo nivel que los archivos PHP.

index.php

Este archivo incluirá la la clase couch.php, la instanciará y llamará al método chart para que este le retorne el gráfico.

```
/**
 * User: rmiguel
 * Date: 22/07/2015
 * Time: 20:00
 */

require_once "CouchWrapper.php";

print "
```

Couchdb Test

«; $con = new CouchWrapper(); print $con->chart(«600×200″,»1qaz»);

config.php

Este archivo es el típico utilizado para parametrizar las WebApp que tienen acceso a bbdd, no comentare el archivo por su simpleza y los descriptivos nombres de las variables. Dado que esto es un ejemplo, me permito la licencia de utilizar variables globales, no cabe decir que en una aplicación de uso está sería una mala implementación de uso para un archivo de configuración dada la información que contiene. Personalmente yo creo un xml que parseo con un método privado en la clase, el cual guarda el valor de los tags en variables privadas. De esta manera aseguramos que este método y las variables son usadas únicamente por la clase protegiéndolas dada la importancia de su valor.

```
/**
 * User: rmiguel
 * Date: 22/07/2015
 * Time: 20:00
 */
//Couch_db config
$COUCHDB_HOST = "127.0.0.1";
$COUCHDB_PORT = "5984";
$COUCHDB_TABLE = "navigator";
```

couch.php

Este archivo es el que contiene la clase _couch. Primero incluimos las dependencias (de la librería y el archivo de configuración) .

A continuación explicaremos la estructura de la classe:

Variables

- $client será donde se guardará la instancia cliente que ataca a la bbdd.

- $couchdsn será la url y el puerto (http://127.0.0.1:5894)
- $couch_db será la base de datos (navigator)

Métodos

- _couch -> es el constructor en él se realizará la conexión a la bbdd.
- get_doc($id) -> retornara una variable con la estructura del documento correspondiente al id que se le pasa por parámetro.
- chart -> retornará una url del tipo pie chart que atacara a google chart api para crear un gráfico en función del id de documento y la resolución que se especifique.

```
/**
 * User: rmiguel
 * Date: 22/07/2015
 * Time: 20:00
 */
//-----------------------
require_once "lib/couch.php";
require_once "lib/couchClient.php";
require_once "lib/couchDocument.php";
require_once "config.php";
//-----------------------------------
class CouchWrapper {
    public $client;
    public $couch_dns;
    public $couch_db;
    /**
     * Constructor
     */
    function __construct(){
                                                $couch_dsn
=".$GLOBALS['COUCHDB_HOST'].":".$GLOBALS['COUCHDB_PORT'];
        $couch_db = $GLOBALS['COUCHDB_TABLE'];
        try{
                                    $this->client   =   new
couchClient($couch_dsn,$couch_db);
```

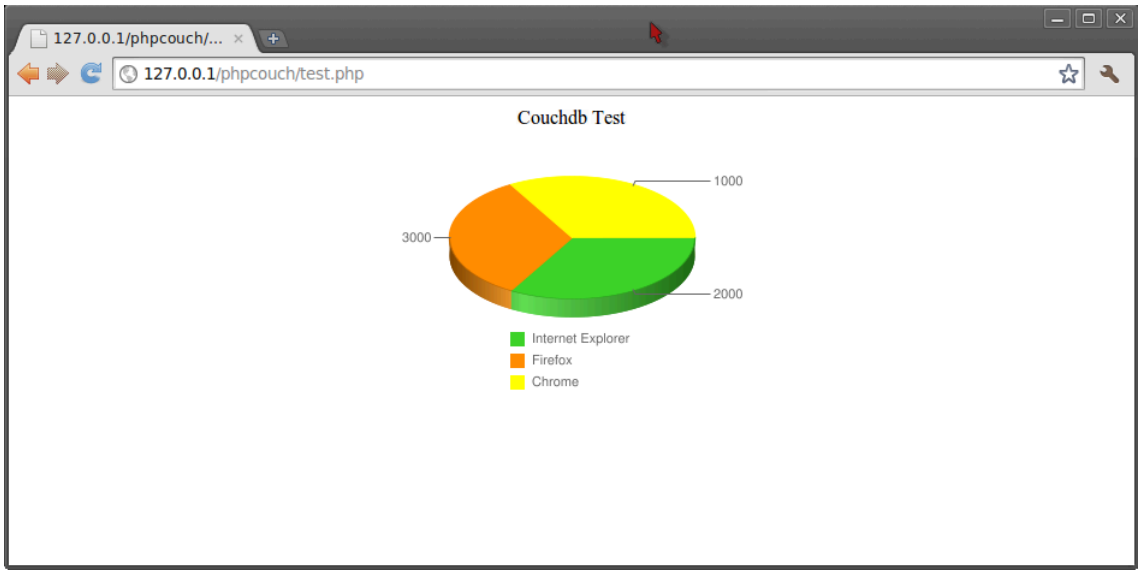
```
        } catch (Exception $e) {
                        echo  "Error:".$e->getMessage()."
(errcode=".$e->getCode().")\n";
        }
    }
    /**
     * @param $id
     * @return document values
     */
    function get_doc($id){
        try{
            $doc = $this->client->getDoc($id);
        } catch (Exception $e) {
            if ( $e->code() == 404 ) {
                echo "Document \"some_doc\" not found\n";
            } else {
                            echo "Something weird happened:
".$e->getMessage()." (errcode=".$e->getCode().")\n";
            }
            exit(1);
        }
        return $doc;
    }
    /**
     * @param $resolution
     * @param $id_document
     * @return chart image tag
     */
    function chart($resolution,$id_document){
        $doc = $this->get_doc($id_document);
        $green = $doc->ie;
        $orange = $doc->firefox;
        $yellow = $doc->chrome;
        $chart = "
```

```
«; return $chart; } }
```

Resultado

Una vez ejecutemos en nuestro servidor web index.php

deberíamos obtener este resultado si todo ha funcionado correctamente:



Observaciones

Bien con este post doy por finalizada la breve introducción al mundo de las bbdd NoSql, espero y deseo que hayan sido de ayuda.

Posts relacionados

-> NoSql

-> NoSql – CouchDB

Código fuente en GitHub

PHPonCouch GitHub

Fuentes

- Wiki couchdb -> Getting_started_with_PHP
- Google Api Char

Ruben

NoSql – CouchDB

Introducción

CouchDB es una BBDD NoSql (si tienes dudas sobre NoSql es recomendable visitar el siguiente post Bases de Datos NoSql) de documentos en formato JSON. Dichos documentos se referencian por una clave única _id a los cuales se puede acceder por HTTP y gestionar mediante javascript. Esta programado Erlang, un lenguaje de programación muy robusto, fiable y multiplataforma, que se ejecuta sobre una máquina virtual de alto rendimiento.

Claves y revisiones

Ambos campos tienen que ser únicos, no pueden estar duplicados entre otros documentos en la misma base de datos.

- _id (clave): es el identificador único e inequivoco del documento (DocID)
- _rev(revisión): es un identificador del documento para tratar su historico. Es decir, de cada documento se guardan revisiones para poder acceder a anteriores versiones de dicho documento.

Revisiones

Cada vez que modificamos un documento en CouchDB se crea una nueva revisión de este. En principo esto nos permite acceder a anteriores versiones del documento, pero puede llegar a ser un lastre. Por esa razón podemos compactar el documento, al hacerlo tansolo quedará disponible la última revisión.

En CouchDB podemos hacer consultas directas por identificador y por revisión, podemos obtener todas las revisiones activas

de un documento (si existieran) y podríamos consultar información sobre las mismas.

A partir de la versión v 0.11 se puede especificar el número de revisiones:

```
curl -X PUT -d "200" http://localhost:5984/test/_revs_limit
```

Para compactar la base de datos y así eliminar las revisiones antiguas:

```
curl -X POST http://localhost:5984/nombre_base_de_datos/_compact
```

Vistas

CouchDB permite la creación de vistas, que son el mecanismo que permite la combinación de documentos para retornar valores de varios documentos, es decir, CouchDB permite la realización de las operaciones JOIN típicas de SQL.

Replicación

La replicación entrebases de datos nos permite duplicar datos con la seguridad de no perderlos. Hay que tener en cuenta que al guardar revisiones de los documentos siempre dispondremos de todas las revisiones. Es decir, pongamos que en tenemos una base de datos bbdd1 y otra bbdd2 y cambiamos un documento común doc1, después de la replicación tendremos las revisiones de los documentos anteriores y las revisiones de los nuevos documentos en las dos bases de datos. Es nos asegura consistente persistencia de datos.

Para habilitar la replicación activa entre dos bases de datos (cada vez que hay un cambio automaticamente las bases de datos se replican) tansolo debemos ejecutar el siguiente comando:

```
curl -X POST http://localhost:5984/_replicate -d '{"source":"db", "target":"db-replica", "continuous":true}'
```

Resolución de conflictos

En CouchDB la detección y resolución de conflictos esta automatizada. En el caso de que un documento se intente actualizar en distintos nodos se guardaran las distintas versiones y poniendo como la versión más reciente la ganadora. Para solventar conflictos, CouchDB permite gestionar las versiones, pudiendo borrar veriones erroneas o reordenarlas por fecha.

Instalación

Tan simple como ejecutar en la consola:

sudo apt-get install couchdb

Luego podemos comprobar en un navegador si CouchDB ha arrancado correctamentte accediendo a la siguiente

url http://localhost:5984/

, en caso de que todo funcione correctamente recibiremos una respuesta web similar a esta dependiendo de la versión de CouchDB:

{"couchdb":"Welcome","version":"0.10.0"}

Comandos básicos mediante curl

En primera instancia deberemos instalar curl:

sudo apt-get install curl

Para visualizar todas las bases de datos:

curl http://localhost:5984/_all_dbs

Crear una base de datos:

curl -X PUT http://localhost:5984/test

Crear un nuevo documento:

curl -X PUT

```
http://localhost:5984/test/8765rftded6c29495e54cc05947f18c8af
-d '{"SO":"Linux","Distro":"Debian"}'
```

Borrar la base de datos:

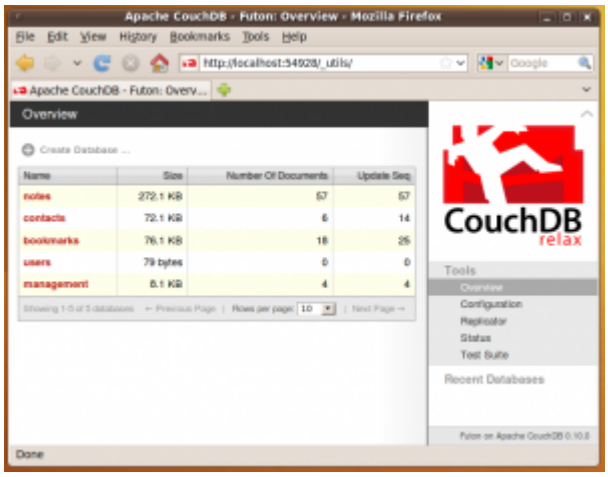
```
curl -X DELETE http://localhost:5984/test
```

Futon

Futon es una aplicativo web muy intuitivo y fácil de usar, por lo que no explicaré como realizar las operaciones antes descritas con curl , para gestionar bases de datos y sus documentos en CouchDB.

Para acceder a él tansolo debemos acceder mediante navegador web a la siguiente url:

```
http://localhost:5984/_utils/
```



Observaciones

Llegados a este punto ya tenemos un concepto básico sobre las bases de datos NoSql y la utilización de un gestor, CouchDB. El próximo post sobre este tema estará en el apartado de programación, donde se expondrá un ejemplo de acceso a CouchDB mediante PHP.

Fuentes

- http://www.oriolrius.cat
- http://en.wikipedia.org/wiki/CouchDB

Ruben