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