Instituto Tecnológico de Costa Rica
Área Académica de Ingeniería en Computadores
Introducción a los Sistemas Embebidos
Prof. Ing. Jeferson González Gómez

Tutorial: Introducción a GNU Build System (Autotools)

Basado en: `http://www.lrde.epita.fr/~adl/autotools.html`

1. Introducción

Este tutorial presenta un enfoque introductorio al uso de las herramientas del sistema de construcción de GNU, también llamado **Autotools**. En el desarrollo de este tutorial se describirá el funcionamiento de *Autotools* tanto en modo usuario, como en modo desarrollador. Además, se aplicará *Autotools* en la creación de un programa simple y una biblioteca estática y dinámica. A continuación se presentan algunos conceptos importantes a considerar:

Autotools

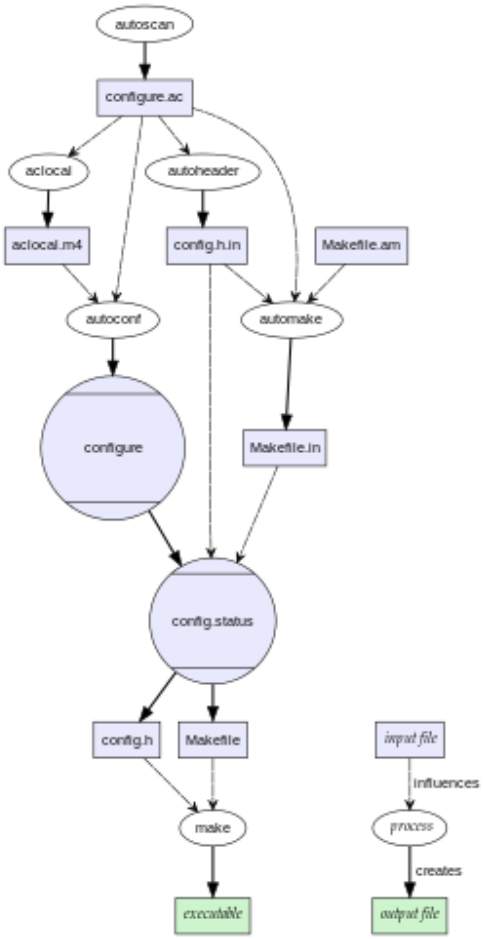
El sistema de construcción de GNU (GNU Build system), comúnmente conocido como Autotools, corresponde a un conjunto de herramientas del proyecto GNU centradas en la construcción de paquetes de código fuente con alta portabilidad para sistemas basados en Unix.

Autotools viene a complementar las herramientas de GNU, de forma que la construcción de paquetes de código fuente sea automática, lo que reduce tiempo y complejidad en creación de Makefiles y scripts para paquetes de código abierto.

Autotools está compuesto, entre otros, por tres herramientas principales: *Autoconf*, *Automake* y *Libtool*.

- **Autoconf**: Esta herramienta permite generar un script de configuración (*configure*) que va a caracterizar el proceso de construcción de un código fuente particular. En este script se verifica la existencia y compatibilidad del compilador, la presencia de bibliotecas dinámicas, archivos de encabezado, ambiente de cross-compilación, etc). En este script, además, se genera **automáticamente** los archivos Makefiles necesarios para la construcción del paquete de código abierto.
- **Automake**: Esta es la herramienta encargada propiamente de la construcción automática de Makefiles. Además se encarga de rastrear dependencias de manera automática.
- **Libtool**: Esta herramienta se encarga de manejar la creación de bibliotecas estáticas y dinámicas de manera abstracta y automática. El principal objetivo de Libtool es abstraer los diferentes tipos de archivos de bibliotecas compartidas, por medio de un formato de archivo genérico (libtool archive - **.la**) que envuelve los demás formatos.

De esta forma, *Autotools* facilita la creación de paquetes de software, al abstraer la construcción propiamente, así como el manejo de archivos de cabecera, bibliotecas, Makefiles, dependencias, etc.
A continuación se presenta un gráfico que describe el proceso completo de construcción. Más adelante se explicará la función de los archivos y herramientas descritos en la figura.



**By Jdthood - Own work, CC BY-SA 3.0,
`https://commons.wikimedia.org/w/index.php?curid=15581407`

2. Autotools: Modo usuario

Para entender el funcionamiento de Autotools, primero se procederá a utilizarlo en modo usuario, para instalar un paquete de código abierto. El paquete será la biblioteca *libpng*, utilizada para el soporte del formato PNG en imágenes, y pieza fundamental de cualquier biblioteca de procesamiento de imágenes.

A continuación se describen los pasos para la instalación de la biblioteca *libpng* utilizando Autotools, en modo de usuario.

Nota: esta biblioteca probablemente se encuentre instalada en su sistema, pero se hará una instalación manual en un directorio convencional, para no interferir con el funcionamiento del sistema.

1. En el directorio home (/home/usuario) se creará un directorio llamado libpng.
2. Dentro de libpng se descargará el paquete de la biblioteca:

```
$ wget https://sourceforge.net/projects/libpng/files/libpng16/1.6.34/libpng-1.6.34.tar.gz
```

3. Se descomprime el archivo descargado

```
$ tar -xvf libpng-1.6.34.tar.gz
$ cd libpng-1.6.34
$ ls
```

Al ejecutar *ls* se pueden visualizar los archivos de construcción del paquete de la biblioteca. El archivo más importante, en este punto, es el script *configure*, que permitirá la construcción automática de la biblioteca. Se puede ejecutar *$ cat configure* para visualizar el contenido del script si se desea. Cabe recordar que este script es autogenerado por medio de Autoconf.

4. Preparación de la instalación: Autotools permite que los directorios de instalación y construcción sean distintos. Para el directorio de construcción se creará un directorio dentro de libpng-1.6.34 llamado build. Esta es una práctica común en la mayoría de construcciones de paquetes. El directorio de instalación se definirá dentro del directorio build. Por defecto, Autotools usa */usr/* como directorio de instalación, pero para no interferir con el sistema se creará un directorio *usr* dentro del directorio build.

```
$ mkdir build
$ cd build
$ mkdir usr
```

5. Ejecución de script de configuración:

```
$ ../configure --prefix=/home/usuario/libpng/libpng-1.6.34/build/usr
```

El comando - *-prefix=...* permite establecer la ruta en la que se desea instalar el paquete. Esta debe establecerse explícitamente y no se permite el uso de direccionamiento relativo. Al ejecutarse el script se verifica el compilador y las diferentes dependencias y se genera (entre otros) un archivo Makefile que permitirá la construcción final de la biblioteca. Esto se puede verificar con *ls*.

6. Construcción de la biblioteca: Al crear el Makefile automáticamente, para construir el paquete solo se debe ejecutar el comando *make*.

```
$ make
```

7. Instalación: La instalación de la biblioteca se realizará en el directorio *usr*, dentro del directorio *build*

```
$ make install
$ cd usr
$ ls
```

Al realizar *ls* se puede visualizar el contenido de la instalación. Esta biblioteca específicamente crea un directorio **bin** con los archivos binarios o ejecutables que son parte del programa, un directorio **lib** que contiene las bibliotecas estática (.a) y dinámicas (.so, .la), un directorio **include** que contiene los archivos de cabecera del código de la biblioteca y un directorio **share** que contiene la documentación de la biblioteca. Cabe destacar que esta estructura de directorios es estandar y corresponde con la estructura de archivos de Linux. De haber omitido el comando - *-prefix*, la biblioteca se habría instalado en el directorio */usr* del sistema, y el contenido de cada subdirectorio (bin, include, etc) se habría instalado en su correspondiente del sistema (/usr/bin, /usr/include, etc).

Para verificar la instalación se puede ejecutar alguno de los binarios, dentro de bin (*$ ./libpng16-config*)

3. Programa simple con Autotools: Hello World

Para esta sección se utilizará Autotools en modo de desarrollador para la construcción de un programa simple *Hello world*, para esto se utilizará el siguiente código fuente helloworld.c

```
#include <config.h>
#include <stdio.h>
int main (void)
{
puts (" Hello World ! " ) ;
puts (" This is " PACKAGE_STRING ". " ) ;
return 0;
}
```

A continuación se muestran los pasos para construir el paquete de código para el programa. *helloworld*.

1. Preparación de la estructura de archivos: En el directorio home (/home/usuario) se deberá crear un directorio para el paquete, llamado *helloworld*. Dentro de *helloworld* deberá crearse un directorio llamado *src*. Se deberá copiar el archivo helloworld.c al directorio *src*.

2. Archivos *template*: Para poder utilizar Autotools se requieren archivos específicos (adicionales al código fuente). Para este caso se requerirán 3 archivos: configure.ac, Makefile.am, src/Makefile.am. A continuación se presenta el contenido de cada uno.
   ****configure.ac****

   ```
   AC_INIT([helloworld], [1.0], [mail@mailprovider.com])
   AM_INIT_AUTOMAKE([foreign -Wall -Werror])
   AC_PROG_CC
   AC_CONFIG_HEADERS([config.h])
   AC_CONFIG_FILES([Makefile src/Makefile])
   AC_OUTPUT
   ```

   Este archivo es el punto de entrada de Autoconf para la generación automática de los archivos y scripts necesarios para la construcción. El comando *AC_INIT([helloworld], [1.0], [mail@mailprovider.com])* inicializa Autoconf, se especifica el nombre del paquete, la versión y la dirección de correo para el reporte de errores/bugs. El comando *AM_INIT_AUTOMAKE([foreign -Wall -Werror])* inicializa Automake y habilita la generación de warnings en la construcción. El comando *AC_PROG_CC* revisa por el compilador de C. LEl comando *AC_CONFIG_HEADERS([config.h])* declara config.h como salida de headers. Una de las ventajas de Autotools es que agrupa todas las definiciones y macros generales en un solo header. El comando *AC_CONFIG_FILES([Makefile src/Makefile])* establece qué archivos serán la salida de Autoconf. En este caso corresponde al Makefile general y el Makefile del código fuente. Finalmente, El comando *AC_OUTPUT* establece los archivos como salida.

   ****Makefile.am****

   ```
   SUBDIRS = src
   ```

   Este corresponde al archivo de entrada de Automake para la generación del Makefile principal (ubicado en /helloworld/Makefile. En este archivo de entrada de Automake se

define los subdirectorios sobre los que construir recursivamente.

****src/Makefile.am****

```
bin_PROGRAMS = hello
hello_SOURCES = helloworld.c
```

Este documento corresponde al archivo de entrada de Automake para el Makefile del código fuente. La variable *bin_PROGRAMS = hello* establece el nombre del binario a producir (hello) y lo define como programa a ser almacenado en el directorio bin. La variable *hello_SOURCES = helloworld.c* establece explícitamente cuál es el código fuente a compilar para la generación del binario *hello* y lo agrega a la lista de códigos fuente.

Nota: Cabe agregar que cualquier tipo de macros y banderas de compilación/enlace pueden agregarse como entrada para ser consideradas en los Makefiles (-L../, -I./, etc). Esto se ejemplificará más adelante.

3. Generación de scripts de configuración y archivos: Autotools tiene diferentes herramientas para la generación automatizada de los Makefiles y scripts de configuración, por medio de Automake y Autoconf. Para el caso de Autoconf se debe ejecutar el comando *autoreconf* que ejecuta a su vez otros comandos de Autoconf en el orden correcto. Este comando genera todos los archivos intermedios y finalmente produce el script *configure.* que será ejecutado por el usuario final. Entonces, en el directorio helloworld se ejecuta:

```
autoreconf --install
```

Al ejecutar *ls* se puede verificar la creación de los diferentes archivos, específicamente el script *configure*

4. Verificación: Para verificar el funcionamiento de esta sección se pueden seguir los pasos 4-7 de la sección 2, en modo usuario.

5. Creación del paquete de código con la aplicación: Para generar un paquete estandar de código abierto con Autotools, luego de seguidos los pasos anteriores, dentro del directorio build se utiliza el comando:

```
make distcheck
```

Este comando genera la versión empaquetada del programa simple helloworld (helloworld-1.0.tar.gz), con los archivos necesarios para su distribución a un potencial usuario.

4. Biblioteca simple con Autotools

En esta sección se utilizará Libtool, como parte de Autotools, para la generación de una biblioteca simple *libhello.la* (libhello.so).

Como primera parte para la generación de la biblioteca se debe contar con los códigos fuentes tanto de la biblioteca como de la aplicación, para esto se asumirá una estructura estandar de archivos. En esta estructura los archivos requeridos son: *src/main.c* (código fuente de aplicación),

lib/say.c (código fuente de la biblioteca) y *include/say.h* (archivo cabecera de la biblioteca). El contenido de los archivo es:

*****src/main.c*****

```
#include <say.h>

int main(int argc, char const *argv[])
{
say_hello();
return 0;
}
```

*****lib/say.c*****

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

void say_hello(void)
{
puts("Hello World!");
puts("This is " PACKAGE_STRING".");
}
```

*****include/say.h*****

```
 void say_hello(void);
```

A continuación se muestran los pasos para construir el paquete de código para la biblioteca.

1. Archivos *template*: Similar al punto anterior, al utilizar Autotools se requieren archivos adicionales. Para este caso se requerirán 4 archivos: configure.ac, Makefile.am, src/Makefile.am y lib/Makefile.am. A continuación se presenta el contenido de cada uno.
   ****configure.ac****

   ```
   AC_INIT([libhello], [1.0], [mail@emailprovider])
   AC_CONFIG_AUX_DIR([build-aux])
   AC_CONFIG_MACRO_DIR([m4])
   AM_INIT_AUTOMAKE([foreign -Wall -Werror])
   AC_PROG_CC
   AM_PROG_AR
   LT_INIT
   AC_CONFIG_HEADERS([config.h])
   AC_CONFIG_FILES([Makefile lib/Makefile src/Makefile])
   AC_OUTPUT
   ```

   Igual que en el punto anterior, este archivo es el punto de entrada de Autoconf para la generación automática de los archivos y scripts necesarios para la construcción. El comando *AC_CONFIG_MACRO_DIR([m4])* es requerido por Libtool para la ejecución de macros. El comando *LT_INIT* inicializa Libtool para su uso.

****Makefile.am****

```
SUBDIRS = lib src
ACLOCAL_AMFLAGS = -I m4
```

En el archivo, la variable *ACLOCAL_AMFLAGS = -I m4* establece banderas para la ejecución de comandos para el Libtool.

****lib/Makefile.am****

```
lib_LTLIBRARIES = libhello.la
include_HEADERS = $(top_srcdir)/include/say.h
AM_CPPFLAGS = -I$(top_srcdir)/include
libhello_la_SOURCES = say.c
```

Este archivo corresponde a la entrada para el Makefile que construirá la biblioteca. La variable *lib_LTLIBRARIES = libhello.la* incluye la biblioteca *libhello.la* a la lista de construcción y la agrega al directorio correspondiente (lib). La variable *include_HEADERS = $(top_srcdir)/include/say.h* agrega el archivo de cabecera al directorio *include* en la instalación. La variable *AM_CPPFLAGS = -I$(top_srcdir)/include* establece las banderas del compilador. Para este caso, se define la bandera *-I* que denota la ubicación de los archivos cabecera de la biblioteca; en este comando se utiliza la variable $(top_srcdir) para hacer referencia al directorio de código fuente de la biblioteca en la definición de la ruta.

****src/Makefile.am****

```
AM_CPPFLAGS = -I$(top_srcdir)/include
bin_PROGRAMS = hello
hello_SOURCES = main.c
hello_LDADD = $(top_builddir)/lib/libhello.la
```

Este documento corresponde al archivo de entrada de Automake para el Makefile del código fuente del programa que usa la biblioteca. La variable *AM_CPPFLAGS = -I$(top_srcdir)/include* establece las banderas del compilador. La variable *hello_LDADD = $(top_builddir)/lib/libhello.la* permite enlazar el programa con la biblioteca construida.

2. Generación de scripts de configuración y archivos: Al igual que la sección anterior, el comando *autoreconf* se encarga de la generación.

```
autoreconf --install
```

3. Verificación: Para verificar el funcionamiento de esta sección se pueden seguir los pasos 4-7 de la sección 2

4. Creación del paquete de código con la aplicación.

```
make distcheck
```

5. Evaluación

5.1. Descripción

Debe crear una biblioteca, en lenguaje C, la cuál ofrecerá cinco funciones matemáticas: suma, resta, multiplicación, división y raíz cuadrada. Para este caso, debe utilizar Autotools tanto para la generación de las biblioteca, como de las aplicaciones que las verifican. La estructura de la solución de este Ejercicio es la siguiente:

- Archivo empaquetado de la biblioteca creada con Autotools (ejemplo *libhello-1.0.tar.gz*), siguiendo el formato presentado en el tutorial. La estructura de archivos deberá ser la estandar (bin, lib, include, src, etc).

5.2. Entregable

(Subir al tecDigital)

- Único archivo .tar (incluya su nombre como parte nombre del archivo) con el archivo empaquetado de la biblioteca descrito anteriormente.

Material adicional

****`http://www.gnu.org/software/automake/manual/automake.html`

****`http://www.lrde.epita.fr/~adl/autotools.html`