Ferramentas de compilación

GPUL

Xabier Rodríguez Calvar

Ferramentas de compilación

- Compilación: *Make*
- *Java™*: *Ant*
- Grandes proxectos: *Automake–Autoconf–Libtool*

Make

- Ferramenta para axuda da compilación.
- Consegue non compilar o que xa está compilado.
- Ficheiro de entrada: `Makefile`.

Makefile

- Contido:
  - *Targets*
  - Prerrequisitos
  - Instruccións (ollo ás tabulacións)
- Exemplo:

```
all: hello

hello.o: hello.c
	gcc -c hello.c

hello: hello.o
	gcc -o hello hello.o
```

Makefile (2)

- Definición de variábeis:
  - Definición: `nome_da_variábel=definición`.
  - Uso: `$(nome_da_variábel)`.
  - Exemplo:

```
LATEX=latex -interaction=nonstopmode
SOURCES=memoria.tex introduction.tex

memoria.dvi: $(SOURCES)
        $(LATEX) memoria.tex
```

Makefile (3)

- Regras implícitas:
  - Especifícanse por *pattern-matching*.
  - Úsase o carácter %.
  - Variábeis implícitas:
    - $<: contén o prerrequisito.
    - $@: contén o *target*.
  - Exemplo:

```
CC=gcc
CFLAGS=-Wall
%.o: %.c
        $(CC) $(CFLAGS) -o $@ -c $<
```

Makefile (4)

- *Phony targets*:
  - Son os que non se corresponden cun nome de ficheiro.
  - Actívanse sempre.
  - Exemplo:

```
clean:
        rm -f *.o *~

mrproper: clean
        rm -f hello
```

Makefile (5)

- Comentarios:
  - Márcanse con # e van ata o final da liña.
  - Exemplo:

```
# Isto é un comentario
```

- Partir liñas:
  - Úsase o carácter \.
  - Exemplo:

```
SOURCES=memoria.tex \
        introduction.tex
clean:
        rm -f *.o \
        *~
```

Execución de Make

- Para lanzar *Make* basta con teclear `make` dende o shell.
- Pódese especificar tamén o *target* que se desexe executar.
  - Por defecto execútase o primeiro definido.

```
$ make all
```

Ant

- *Make* é moi complexo no caso de *Java*™ (paquetes).
- *Ant* está pensado para *Java*™.
  - Está feito en *Java*™.
- Ficheiro de entrada: `build.xml`.
- A execución faise chamando ó executábel de *Ant* pasando como parámetro o *target* que hai que executar:

```
$ ant compile
```

`build.xml`

- É un ficheiro *XML* e como tal leva a súa cabeceira:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

- Contido:
  - `project`
    - `target` (pode haber varios).
      - Orde (pode haber varias).
- Exemplo:

```
<project>
  <target>
    <orde/>
  </target>
</project>
```

`build.xml` (2)

- Propiedades:
  - Son unha serie de declaracións para facer máis sinxela a definición das tarefas.
  - Declaración:
    - `build.xml`: Usando o *tag* `property`, cos seguintes atributos:
      - `name`: nome.
      - `value`: valor desexado.

```
<property name=''sources''
        value=''sources/''/>
```

    - Na chamada a *Ant*: pasando por parámetro do seguinte xeito: `-Dname=value`.

```
$ant -Dsources=./sources compile
```

`build.xml` (3)

- Propiedade `build.compiler`:
  - Serve para seleccionar o compilador desexado:
  - Valores:
    - `modern`: compilador do *JDK 1.3* en adiante.
    - `jikes`: compilador *Jikes*.

build.xml (4)

- Etiqueta `project`:
  - Especifica o proxecto.
  - Dentro inclúense `targets`.
  - Atributos:
    - `name`: nome (opcional).
    - `default`: *target* por defecto.
    - `basedir`: directorio base (opcional).
  - Exemplo:

```
<project name=''exemplo''
    default=''compile'' basedir=''.''>
  <!-- targets -->
</project>
```

`build.xml` (5)

- Etiqueta `target`:
  - Especifica un *target* que se pode executar.
  - Dentro inclúense tarefas.
  - Atributos:
    - `name`: nome que se especificará para executalo.
    - `depends`: targets que deben executarse antes do actual (opcional).
  - Exemplo:

```
<target name=''javadoc'' depends=''compile''>
  <!-- tarefas -->
</target>
```

`build.xml` (6)

- Tarefa `javac`:
  - Compila clases *Java*.
  - Compila só o que se precise.
  - Atributos:
    - `srcdir`: directorio onde onde se atopan as clases que se deben compilar.
    - `destdir`: directorio destino.
    - `classpath`: para especificar a ruta coas clases que se precisan na compilación.
  - Exemplo:

```
<javac srcdir=''${sources}''
    destdir=''${classes}''
    classpath=''${classpath}''/>
```

`build.xml` (7)

- Tarefa `javadoc`:
  - Xera a documentación *Javadoc*.
  - Atributos:
    - `classpath`: ruta das clases necesarias.
    - `sourcepath`: directorio onde están os fontes.
    - `packagenames`: os paquetes que se desexen xerar.
    - `desdir`: onde queremos almacenar o resultado.
    - `private/package/protected/public`: nivel de detalle desexado, por defecto é `protected`.

`build.xml` (8)

- Tarefa `javadoc`:
  - Atributos:
    - `windowtitle`: título da fiestra.
    - `doctitle`: título do documento.
    - `header`: cabeceira do título.
    - `bottom`: o que aparece ó final de cada páxina.
  - Pódese incluir o *tag* `link` co attributo `href` para enlazar por exemplo contra a documentación do sistema.

build.xml (9)

- Tarefa `javadoc`:
  - Exemplo:

```
<javadoc classpath="${p.classpath}"
    sourcepath="${sources}"
    packagenames="modelo.*"
    destdir="${docs}"
    private="yes"
    windowtitle="${projectname}"
    doctitle="${projectname}"
    header="${projectname}"
    bottom="${projectname} - ${authors}"
  <link href="${j2se.docs}"/>
</javadoc>
```

`build.xml` (10)

- Tarefa `exec`:
  - Executa un comando
  - Atributos:
    - `failonerror`: se se quere que *Ant* falle se o programa non devolve un valor correcto.
    - `executable`: o nome do executábel.
    - `dir`: directorio onde se debe executar.
  - Se o programa ten argumentos pódese usar o *tag* `arg` co atributo `line`.

build.xml (11)

- Tarefa exec:
  - Exemplo:

```
<exec failonerror=''yes''
    executable=''make''
    dir=''${csources}''>
  <arg line=''all''/>
</exec>
```

`build.xml` (12)

- Outras tarefas:
  - `ant`: chama a outro ficheiro *Ant*.
  - `copy`: copia ficheiros e directorios.
  - `delete`: borra ficheiros e directorios.
  - `echo`: envía mensaxes á saída estándar.
  - `gzip`: comprime un ficheiro.
  - `mkdir`: crea un directorio.
  - `tar`: crea un ficheiro de cinta.
  - `jar`: crea un ficheiro *jar*.
  - ... : consultar no manual.

`build.xml` (13)

- Patróns de selección de ficheiros:
  - Patróns máis usuais:
    - `*`: encaixa con 0 ou máis caracteres.
    - `?`: encaixa cun carácter.
    - `**`: todos os directorios fillos.
  - Patróns que se exclúen por defecto (pódense deshabilitar con `defaultexcludes="no"`):
    - `**/*~`
    - `**/CVS/**`
    - `**/#*#`
    - ...
  - Ás veces úsanse mediante os *tags* `fileset` ou `patternset`.

Automake–Autoconf–Libtool

- Definifición complexa.
- Gáñase simplicidade en proxectos grandes.
- Usaremos os tres programas integrados:
  - *Autoconf*: comprobacións e parámetros coas macros de *M4*.
  - *Automake*: compilación.
  - *Libtool*: xeración de bibliotecas, tanto de enlace estático coma dinámico.

autogen.sh

- Usarémolo para lanzar *Automake* e *Autoconf* .
- É conveniente copialo dalgunha aplicación, preferentemente *Gnome*.
- Contido:

```
intltoolize --copy --force --automake
xml-i18n-toolize --copy --force --automake
libtoolize --force --copy
aclocal $aclocalinclude
autoheader
automake --add-missing --gnu $am_opt
autoconf
```

Cumprir con GNU

- É recomendábel incluir os seguintes ficheiros:
  - `README`
  - `INSTALL`
  - `COPYING`
  - `AUTHORS`
  - `NEWS`
  - `ChangeLog`

Autoconf

- O seu ficheiro de entrada será o `configure.in` (un por aplicación).
- Terá a seguinte estructura:
  - Inicialización.
  - Comprobacións.
  - Substitucións.
  - Creación de ficheiros.

configure.in

- Inicialización:
  - Exemplo:

```
AC_INIT(configure.in)
AM_INIT_AUTOMAKE(proba_proxecto, 0.1)
AM_MAINTAINER_MODE
AM_CONFIG_HEADER(config.h)
```

    - O `config.h` conterá definicións. Pódese inicializar no `config.h.in`.

`configure.in` (2)

- Comprobacións:
  - Hai case infinitas, pódense consultar no manual.
  - Pódense escribir novas.

configure.in (3)

- Comprobacións:
  - Exemplo:

```
AC_ISC_POSIX
AC_PROG_CC
AM_PROG_CC_STDC
AC_HEADER_STDC

AC_PROG_LIBTOOL

pkg_modules="libgnomeui-2.0"
PKG_CHECK_MODULES(PACKAGE, [$pkg_modules])
AC_SUBST(PACKAGE_CFLAGS)
AC_SUBST(PACKAGE_LIBS)
```

configure.in (4)

- Substitucións:
  - Han de marcarse como @SUBSTITUCION@.
  - Tamén se crean como variábeis.
  - Exemplo:

```
libdynamicdir="/tmp/proba_biblioteca"
LIBDYNAMIC_CFLAGS=\
    "-I$libdynamicdir/include/"
LIBDYNAMIC_LIBS=\
    "-L$libdynamicdir/lib/ -ldynamic"
AC_SUBST(LIBDYNAMIC_CFLAGS)
AC_SUBST(LIBDYNAMIC_LIBS)
```

configure.in (5)

- Creación de ficheiros:
  - Aquí especifícanse os ficheiros que han de ser creados.
  - Exemplo:

```
AC_OUTPUT([
Makefile
src/Makefile
src/text/Makefile
])
```

Automake

- Aquí especificamos como se vai construir o software.
- O ficheiro de entrada é o `Makefile.am`.
  - Hai un por cada subdirectorio.
  - Os contidos especifícanse como variábeis de *Make*, cuns nomes especiais.
  - Tamén se poden incluir *targets* no xeito usual.

Libtool

- Serve para crear bibliotecas de enlace estático ou dinámico.
- Usarémolo integrado con *Automake*.

Makefile.am

- Variábeis normais:
  - `SUBDIRS`: especificamos os subdirectorios, que se construirán primeiro.
  - `EXTRA_DIST`: ficheiros que se han de incluir a maiores na versión de distribución.
  - `INCLUDES`: especifica a ruta ós ficheiros de cabeceiras.
  - `bin_PROGRAMS`: executábeis que hai que crear.
  - `executábel_CFLAGS`: parámetros para a etapa de compilación.
  - `executábel_LDADD`: parámetros para a etapa de enlazado.

Makefile.am (2)

- Variábeis normais:
  - Exemplo:

```
SUBDIRS = text
INCLUDES = @PACKAGE_CFLAGS@ -Itext
bin_PROGRAMS = proba_proxecto

proba_proxecto_SOURCES = \
  main.c support.c support.h interface.c \
  interface.h callbacks.c callbacks.h

proba_proxecto_LDADD = \
  @PACKAGE_LIBS@ text/libtext.a \
  @LIBDYNAMIC_LIBS@ @GLIB_PACKAGE_LIBS@
```

`Makefile.am` (3)

- Variábeis para biblioteca de enlace estático:
  - `INCLUDE`: para rutas de cabeceiras.
  - `lib_LIBRARIES`: bibliotecas que se queren crear rematadas en `.a`
  - `biblioteca_include_HEADERS`: ficheiros de cabeceiras que se han de instalar posteriormente.
  - `biblioteca_includedir`: directorio onde se instalarán os ficheiros de cabeceiras.
  - `biblioteca_SOURCES`: ficheiros de código fonte para crear a biblioteca.

Makefile.am (4)

- Variábeis para biblioteca de enlace estático:
  - Exemplo:

```
INCLUDES = @GLIB_PACKAGE_CFLAGS@ \
  @LIBDYNAMIC_CFLAGS@

lib_LIBRARIES = libtext.a
libtext_a_SOURCES = text.h text.c
```

- Os nomes das bibliotecas han de poñerse completos:
  - Substitúense `.` por `_`.

```
libtext_a_SOURCES=text.c
```

`Makefile.am` (5)

- Variábeis para biblioteca de enlace dinámico:
  - `INCLUDE`: para rutas de cabeceiras.
  - `lib_LTLIBRARIES`: bibliotecas que se queren crear rematadas en `.la`
  - `biblioteca_include_HEADERS`: ficheiros de cabeceiras que se han de instalar posteriormente.
  - `biblioteca_includedir`: directorio onde se instalarán os ficheiros de cabeceiras.
  - `biblioteca_SOURCES`: ficheiros fonte.
  - `biblioteca_LIBADD`: parámetros para o enlazado.

Makefile.am (6)

- Variábeis para biblioteca de enlace dinámico:
  - Exemplo:

```
INCLUDES = @PACKAGE_CFLAGS@

lib_LTLIBRARIES = libdynamic.la

libdynamic_la_LIBADD = @PACKAGE_LIBS@
libdynamic_la_include_HEADERS = dynamic.h
libdynamic_la_includedir=$(includedir)/libdynamic

libdynamic_la_SOURCES = dynamic.c \
  $(libdynamic_la_include_HEADERS)
```

Makefile.am (7)

- Enlazado con bibliotecas do proxecto:
  - Enlázase co ficheiro `.a` ou `.la` coma se fose un ficheiro obxecto.
  - Exemplo:

```
proba_proxecto_LDADD = \
  @PACKAGE_LIBS@ text/libtext.a \
  @LIBDYNAMIC_LIBS@ @GLIB_PACKAGE_LIBS@
```

- Enlazado con bibliotecas instaladas:
  - Do xeito usual con `-l`.

- Durante o desenvolvemento:

```
$ ./autogen.sh --prefix=directorio_inst
$ make all
$ make install
```

  - Pódese abreviar:

```
$ ./autogen.sh --prefix=directorio
  && make all install
```

Execucións de Automake–Autoconf–Libtool *(2)*

- Creación de versión de distribución:

  ```
  $ make all dist
  ```

  - Crea un ficheiro `.tar.gz` para distribuir.
- Compilando a versión de distribución:

  ```
  $ ./configure && make all install
  ```
- Desinstalando unha versión instalada:

  ```
  $ make uninstall
  ```

Bibliografía

- Manual de *Make*:
  `www.gnu.org/software/make/manual/make.html`
- Manual de *Ant*: `ant.apache.org/manual`
- Transparencias de*Juan Raposo*:
  `www.tic.udc.es/f̃bellas/teaching/is/Ant_Tutorial.pdf`
- Manual de *Libtool*:
  `www.gnu.org/software/libtool/manual.html`
- Manual de *Automake*:
  `www.gnu.org/software/automake/manual`
- Manual de *Autoconf*:
  `www.gnu.org/software/autoconf/manual/autoconf-2.57`

Bibliografía (2)

- Libro de *Havoc Pennington*:
  `developer.gnome.org/doc/GGAD`
- *Autobook*: `sources.redhat.com/autobook`