Makefile et Ant

BOURGIN Jeremy[1] MANSOURI Hind[2]

[1]L2 INFORMATIQUE

[2]Techniques de Communication et de Conduite de projets

24/04/2017

Plan

plan

GNU Make

- GNU Make est un outil qui contrôle la génération d'exécutables et d'autres fichiers non-source d'un programme à partir des fichiers source du programme.
- Make obtient sa connaissance de la facon de construire notre programme à partir d'un fichier appelé Makefile.

plan

Compilation d'un fichier et édition de liens

- syntaxe des compilateurs
  - compilation basique :
    - g++ -o test fichier1.cpp fichier2.cpp
  - l'édition de liens:
    - g++ -o fichier1.o -c fichier1.cpp
    - g++ -o fichier2.o -c fichier2.cpp
    - g++ fichier1.o fichier2.o -o test

plan

Compiler avec make en édition de lien

- Syntaxe de make

```
fichier1.o: fichier1.c
TAB g++ -c fichier1.c

fichier2.o: fichier2.c
TAB g++ -c fichier2.c

all : fichier1.o fichier2.o
TAB g++ -o test fichier1.o fichier2.o
```

Définition de variables

- Variables personnalisées
  - CC
  - CFLAGS
  - LDFLAGS
  - EXEC
- Variables internes
  - $@ : Le nom de la cible
  - $¡ : Le nom de la première dépendance
  - $^ La liste des dépendances
  - $* : Le nom du fichier sans suffixe

Les régles d'inférence

- Makefile permet également de créer des règles génériques (par exemple construire un .o partir d'un .cpp) qui se verront appelées par défaut. Une telle règle se présente sous la forme suivante :

%.o: %.cpp
TAB commandes

- Construction de la liste des fichiers sources:
  - SRC= $(wildcard *.cpp)
- Génération de la liste des fichiers objets
  - OBJ= $(patsubst %.cpp,%.o,$(SRC))

Utiliser make

Une fois le fichier makefile céé, il suffit simplement de faire :
make cible

plan

Introduction

Créé par Apache

Ant a été créé par la fondation Apache en 2000

Développé en Java

Ant est multiplate-forme

Utilisation du XML

Standardisé, et simple de lecture

Utilisé pour le Java

Ant a principalement été créé pour automatiser les tâches en Java

Des IDE utilisent Ant

Netbeans et Eclipse utilisent Ant pour construire des projets

plan

Java

- créé en 1990
- Java est multiplate-forme (indépendant de la plate-forme hardware)
- Orienté objet
- Pour utiliser Java, il faut installer le client JRE
- Pour développer en Java :

Développer en Java

- Installer Java JRE et JDK
- Configurer la variable PATH
- Configurer la variable JAVA_HOME
- Un fichier pour une classe
- L'espace de nom (package) doit respecter l'emplacement du fichier dans l'arborescence du projet
- Pour compiler utiliser les commandes javac (génère un fichier binaire) et jar (génère un exécutable)

plan

fichier de construction

Pour construire un projet, Ant attends le fichier build.xml. Ce fichier doit commencer par :
<?xml version="1.0" encoding="UTF-8" ?>

Et doit comporter la balise <project> qui contiendra toute les instructions à la construction du projet

La balise project

```
<project name="hello" default="compile" dir=".">
...
</project>
```

- L'attribut name permet simplement de donner un nom un projet
- L'attribut default permet de désigner la cible par défaut
- L'attribut dir permet d'indiquer l'emplacement du projet construire

La balise property

La balise property permet de pouvoir créer des "variables" :
<property name="project.sources.dir" value="src" />
<echo>${project.sources.dir}</echo>

- Attribuer un nom à une variable avec l'attribut name
- Attribuer une valeur à une variable avec l'attribut value
- Pour utiliser une variable, il faut utiliser la syntaxe suivante : ${name}

La balise path et pathelement

La balise path va permettre de regrouper un ensemble de dossiers et fichiers :

```
<path id="classpath">
<pathelement location="projet.jar" />
<pathelement path="lib" />
</path>
```

- L'attribut id va permettre de référencer cet ensemble de dossiers et fichiers
- L'attribut location permet de désigner un fichier
- L'attribut path permet de désigner un dossier

La balise target

La balise target permet de créer des cibles :

```
<target name="init" depends="clean">
<mkdir dir="bin" />
</target>
```

- Attribuer un nom à une cible avec l'attribut name
- L'attribut depends permet d'appeler une autre cible
- Le corp comporte ce qui sera exécuté

Compiler avec la balise javac

```
<javac srcdir="src" destdir="bin" />
```

- L'attribut srcdir désigne le répertoire où se situe les fichiers sources
- L'attribut destdir désigne le répertoire où seront envoyé les fichiers compilé

Générer un exécutable avec la balise jar

<jar destfile="projet.jar" basedir="bin" />

- L'attribut destfile désigne le fichier exécutable qui sera créé
- L'attribut basedir désigne le répertoire où sont stocké les fichiers compilé

Aller plus loin

- Créer la documentation avec la balise <javadoc>
- Créer vos tests unitaires avec la balise <junit>
- Créer un seul fichier générique pour un ensemble de projet avec la balise <import>
- Vous pouvez aussi approfondir, et aller beaucoup plus loin avec Ant : http://ant.apache.org/

Utiliser Ant

Une fois le fichier build.xml céé, il suffit simplement de faire :
ant cible

Conclusion

- Utiliser make pour le c++ et le c
- Utiliser Ant pour le Java
- Le chef de projet doit gérer l'environnement de travail et les règles de compilations