Buildroot: generic-package

Table of Contents

- Buildroot: generic-package

Gentargets est l'infrastructure pour les packages avec des systèmes de construction spécifiques, c'est à dire tous les packages dont le système de construction n'est pas l'un des standards, comme 'autotools' ou 'CMake'. Cela inclut généralement les packages dont la construction système est basé sur des Makefiles écrits à la main ou des scripts shell.

Description générale

```
01: ################################################################
02: #
03: # libfoo
04: #
05: ################################################################
06: LIBFOO_VERSION = 1.0
07: LIBFOO_SOURCE = libfoo-$(LIBFOO_VERSION).tar.gz
08: LIBFOO_SITE = http://www.foosoftware.org/download
09: LIBFOO_INSTALL_STAGING = YES
10: LIBFOO_DEPENDENCIES = host-libaaa libbbb
11:
12: define LIBFOO_BUILD_CMDS
13:     $(MAKE) CC=$(TARGET_CC) LD=$(TARGET_LD) -C $(@D) all
14: endef
15:
16: define LIBFOO_INSTALL_STAGING_CMDS
17:     $(INSTALL) -D -m 0755 $(@D)/libfoo.a $(STAGING_DIR)/usr/lib/libfoo.a
18:     $(INSTALL) -D -m 0644 $(@D)/foo.h $(STAGING_DIR)/usr/include/foo.h
19:     $(INSTALL) -D -m 0755 $(@D)/libfoo.so* $(STAGING_DIR)/usr/lib
20: endef
21:
22: define LIBFOO_INSTALL_TARGET_CMDS
23:     $(INSTALL) -D -m 0755 $(@D)/libfoo.so* $(TARGET_DIR)/usr/lib
24:     $(INSTALL) -d -m 0755 $(TARGET_DIR)/etc/foo.d
25: endef
26:
27: $(eval $(generic-package))
```

Le Makefile commence aux lignes 6 à 8 avec les informations de métadonnées suivantes:

- **LIBFOO_VERSION**: version du package,
- **LIBFOO_SOURCE**: nom du tarball contenant le packag,
- **LIBFOO_SITE**: l'emplacement Internet où le tarball peut être téléchargé.

Toutes les variables doivent commencer par le même préfixe, **LIBFOO_** dans ce cas. Ce préfixe est toujours le nom du package en majuscule.

À la ligne 9, on spécifie que le paquet veut installer quelque chose à l'espace de staging. Ceci est souvent nécessaire pour les bibliothèques, car elles doivent installer les fichiers d'en-tête et d'autres fichiers de développement dans l'espace de staging.

Cela garantira que les commandes répertoriées dans la variable **LIBFOO_INSTALL_STAGING_CMDS** sera exécutée.

À la ligne 10, on spécifie la liste des dépendances sur lesquelles repose ce paquet. Ces dépendances sont répertoriées en termes de noms de packages en minuscules, qui peuvent être des packages pour la cible (sans le préfixe **host-**) ou des packages pour l'hôte (avec le préfixe **host-**)).

Buildroot s'assurera que tous ces packages sont compilés et installés 'avant' que le paquet actuel ne commence sa configuration.

Le reste du Makefile définit ce qui doit être fait aux différents étapes de la configuration, de la compilation et de l'installation du package:

- **LIBFOO_BUILD_CMDS** indique les étapes à suivre pour construire le paquet.
- **LIBFOO_INSTALL_STAGING_CMDS** indique les étapes qui doivent être effectuées pour installer le package dans l'espace de staging.
- **LIBFOO_INSTALL_TARGET_CMDS** indique les étapes à suivre pour installer le package dans l'espace cible.

Toutes ces étapes reposent sur la variable **$(@D)**, qui contient le répertoire où le code source du package a été extrait.

Enfin, à la ligne 27, on appelle les **GENTARGETS** qui génère, en fonction des variables définies précédemment, tous les Code Makefile nécessaire pour faire fonctionner le paquet.

Macro generic-package

Il existe deux variantes de la cible générique. La macro **generic-package** est utilisé pour les packages à compiler de manière croisée pour la cible. la macro **host-generic-package** est utilisée pour les packages hôtes, compilés nativement pour l'hôte.

Il est possible d'appeler les deux en un seul fichier **.mk** : une fois pour créer les règles pour générer un package cible et une fois pour créer les règles permettant de générer un package hôte :

```
$(eval $(generic-package))
$(eval $(host-generic-package)):
```

Cela peut être utile si la compilation du package cible nécessite certains outils à installer sur l'hôte. Si le nom du paquet est **libfoo**, alors le nom du paquet pour la cible est aussi **libfoo**, alors que le nom du paquet pour l'hôte est **host-libfoo**. Ces noms doivent être utilisés dans les variables

DEPENDANCES d'autres packages, si elles dépendent de **libfoo** ou **host-libfoo**.

L'appel à la macro **generic-package** doit être à la fin du fichier **.mk**, après toutes les définitions de variables.

Pour le package cible, le **generic-package** utilise les variables définies par le fichier .mk et préfixé par le nom du package en majuscule : **LIBFOO_**. *Pour le package hôte, il utilise le* ***HOST_LIBFOO_***. Pour 'certaines' variables, si la variable préfixée **HOST_LIBFOO_** ne le fait pas, l'infrastructure du package utilise la variable correspondante préfixés par **LIBFOO_**. Ceci est fait pour les variables susceptibles d'avoir la même valeur pour les packages cible et hôte.

Variables de métadonnées

La liste des variables pouvant être défini dans un fichier **.mk** pour donner des métadonnées l'information est (en supposant que le nom du paquet est **libfoo**) :

- **LIBFOO_VERSION**, obligatoire, doit contenir la version du paquet. Si **HOST_LIBFOO_VERSION** n'existe pas, il est supposé être le même que **LIBFOO_VERSION**. Il peut aussi s'agir d'une branche ou balise Subversion ou Git, pour les packages récupérés directement à partir de leur système de contrôle de révision, par exemple : **LIBFOO_VERSION = 0.1.2**
- **LIBFOO_SOURCE** peut contenir le nom de l'archive du paquet. Si **HOST_LIBFOO_SOURCE** n'est pas spécifié, **LIBFOO_VERSION** est utilisé par défaut. Si aucun n'est spécifié, alors la valeur est supposée être **nomdupaquet-$(LIBFOO_VERSION).tar.gz**. Par exemple : **LIBFOO_SOURCE = foobar-$(LIBFOO_VERSION).tar.bz2**
- **LIBFOO_PATCH** peut contenir le nom d'un patch, qui sera téléchargé depuis le même emplacement que l'archive indiquée dans **LIBFOO_SOURCE**. Si **HOST_LIBFOO_PATCH** n'est pas spécifié, **LIBFOO_PATCH** est utilisé par défaut. Un autre mécanisme est également disponible pour patcher un paquet : tous les fichiers du formulaire **packagename-packageversion-description.patch** présent dans le répertoire du package à l'intérieur de Buildroot seront appliqués au package après extraction.
- * **LIBFOO_SITE** peut contenir l'emplacement Internet du paquet. Ce peut être soit l'emplacement HTTP ou FTP d'une archive tar, soit l'URL d'un référentiel Git ou Subversion (voir **LIBFOO_SITE_METHOD** ci-dessous). Si **HOST_LIBFOO_SITE** n'est pas spécifié, **LIBFOO_SITE** est utilisé par défaut . Si aucun n'est spécifié, l'emplacement est supposé être `http://$(BR2_SOURCEFORGE_MIRROR).dl.sourceforge.net/sourceforge/packagename`. Par exemple : `LIBFOO\_SITE=http://www.libfoosoftware.org/libfoo`, `LIBFOO\_SITE=http://svn.xiph.org/trunk/Tremor/`
- **LIBFOO_SITE_METHOD** peut contenir la méthode pour récupérer le paquet code source. Il peut s'agir de **wget** (pour les téléchargements FTP/HTTP d'archives), **svn**, **git** ou **bzr**. Lorsqu'il n'est pas spécifié, il est deviné à partir de l'URL donnée dans **LIBFOO_SITE** : **svn:**\/\/, **git:**\/\/ et les URL **bzr:**\/\/ utiliseront les méthodes **svn**, **git** et **bzr** respectivement. Tous les autres types d'URL utiliseront la méthode **wget**. Donc par exemple, dans le cas d'un package dont le code source est disponible

via le référentiel Subversion sur HTTP, un “doit” spécifier **LIBFOO_SITE_METHOD=svn**. Pour les méthodes **svn** et **git**, Buildroot clone le référentiel qui est ensuite tarballé et stocké dans le cache de téléchargement. Les prochaines builds ne devront pas cloner à nouveau, mais utilisera l'archive

directement. Lorsque **HOST_LIBFOO_SITE_METHOD** n'est pas spécifié, il prend par défaut la valeur de **LIBFOO_SITE_METHOD**.

- **LIBFOO_DEPENDENCIES** liste les dépendances (en terme de nom de package) requis pour que le package cible actuel soit compilé. Ces dépendances sont garanties d'être compilées et

installé avant le démarrage de la configuration du package actuel. De la même manière, **HOST_LIBFOO_DEPENDENCIES** répertorie la dépendance pour le package hôte actuel.

- **LIBFOO_INSTALL_STAGING** peut être défini sur **YES** ou **NO** (par défaut). Si il est défini sur **YES**, puis les commandes dans les variables **LIBFOO_INSTALL_STAGING_CMDS** sont exécutées pour installer le package dans le staging.
- **LIBFOO_INSTALL_TARGET** peut être défini sur **YES** (par défaut) ou **NO**. Si défini sur **YES**, les commandes dans les variables **LIBFOO_INSTALL_TARGET_CMDS** sont exécutées pour installer le package dans la cible.

La méthode recommandée pour définir ces variables consiste à utiliser les éléments est la syntaxe suivante:

```
LIBFOO_VERSION = 2.32
```

Variables d'action

Variables qui définissent ce qui doit être fait au moment différentes étapes du processus de construction.

- **LIBFOO_CONFIGURE_CMDS**, permet de lister les actions à effectuer pour configurer le paquet avant sa compilation
- **LIBFOO_BUILD_CMDS**, permet de lister les actions à effectuer pour compiler le paquet
- **HOST_LIBFOO_INSTALL_CMDS**, permet de lister les actions à effectuer pour installer le package, lorsque le package est un package hôte. le package doit installer ses fichiers dans le répertoire donné par **$(HOST_DIR)**. Tous les fichiers, y compris les fichiers de développement tels que les en-têtes doivent être installés, car d'autres packages peuvent être compilés en plus de ce paquet.
- **LIBFOO_INSTALL_TARGET_CMDS**, permet de lister les actions à effectué pour installer le package dans le répertoire cible, lorsque le package est un package cible. Le paquet doit installer ses fichiers pour le répertoire donné par **$(TARGET_DIR)**. Seuls les fichiers requis pour 'documentation' et 'exécution' du package doivent être installée. Les fichiers d'en-tête ne doivent pas être installés, ils seront copiés à la cible, si l'option `fichiers de développement dans le système de fichiers cible` est sélectionnée.
- **LIBFOO_INSTALL_STAGING_CMDS**, permet de lister les actions à effectuée pour installer le package dans le répertoire intermédiaire, lorsque le package est un package cible. Le paquet doit installer ses fichiers pour la directory donné par **$(STAGING_DIR)**. Tous les fichiers de développement doivent être installés, car ils pourraient être nécessaires pour compiler d'autres paquets.
- **LIBFOO_CLEAN_CMDS**, permet de lister les actions à effectuer pour nettoyer le répertoire de construction du package.
- **LIBFOO_UNINSTALL_TARGET_CMDS**, permet de lister les actions à désinstaller le package du répertoire cible **$(TARGET_DIR)**
- **LIBFOO_UNINSTALL_STAGING_CMDS**, utilisé pour lister les actions nécessaires pour

désinstaller le package du répertoire intermédiaire **$(STAGING_DIR)**.

La meilleure façon de définir ces variables est :

```
define LIBFOO_CONFIGURE_CMDS
    action 1
    action 2
    action 3
endef
```

Dans les définitions d'action, on peut utiliser les variables suivantes :

- **$(@D)**, qui contient le répertoire dans lequel la source du paquet code a été décompressé.
- **$(TARGET_CC)**, **$(TARGET_LD)**, etc. pour obtenir la cible utilitaires de compilation croisée
- **$(TARGET_CROSS)** pour obtenir le préfixe de la chaîne d'outils de compilation croisée
- Bien sûr les variables, **$(HOST_DIR)**, **$(STAGING_DIR)** et **$(TARGET_DIR)** pour installer correctement les packages.

Utilisation des hooks

La dernière caractéristique de l'infrastructure générique est la possibilité d'ajouter des hookss. Ceux-ci définissent d'autres actions à effectuer après les étapes existantes. La plupart des crochets ne sont pas vraiment utiles pour les packages génériques, puisque le fichier **.mk** a déjà un contrôle total sur les actions effectuées à chaque étape de la construction du colis. Les crochets sont plus utiles pour les paquets utilisant l'infrastructure autotools décrite ci-dessous.

L'exception est **LIBFOO_POST_PATCH_HOOKS**. Patcher package n'est pas définissable par l'utilisateur, donc **LIBFOO_POST_PATCH_HOOKS** sera utile pour les packages génériques.

Les points d'accroche suivants sont disponibles :

- **LIBFOO_POST_PATCH_HOOKS**
- **LIBFOO_PRE_CONFIGURE_HOOKS**
- **LIBFOO_POST_CONFIGURE_HOOKS**
- **LIBFOO_POST_BUILD_HOOKS**
- **LIBFOO_POST_INSTALL_HOOKS** (pour les packages hôtes uniquement)
- **LIBFOO_POST_INSTALL_STAGING_HOOKS** (pour les packages cibles uniquement)
- **LIBFOO_POST_INSTALL_TARGET_HOOKS** (pour les packages cibles uniquement)

Ces variables sont des “listes” de noms de variables contenant des actions à effectué à ce point d'accroche. Cela permet à plusieurs hooks d'être enregistré à un point d'accroche donné. Voici un exemple:

```
define LIBFOO_POST_PATCH_FIXUP
    action1
    action2
endef
LIBFOO_POST_PATCH_HOOKS += LIBFOO_POST_PATCH_FIXUP
```