LKN Annexe A. Utilitaires utiles

Table of Contents

Source: Linux Kernel in a Nutshell: A. Helpful Utilities

La récupération, la construction, la mise à jour et la maintenance d'une arborescence de sources du noyau Linux impliquent de nombreuses étapes différentes. Les développeurs ont créé des programmes pour aider avec les différentes tâches de routine. Cette section décrit quelques-uns de ces outils utiles et les bases de leur utilisation.

Le développement du noyau Linux diffère à bien des égards du développement logiciel traditionnel. Certaines des exigences particulières imposées aux programmeurs du noyau incluent:

- Appliquer constamment les modifications à la cible mobile d'un calendrier de publication de développement de noyau rapide.
- Résoudre les conflits de fusion entre les modifications apportées et les modifications apportées par d'autres personnes.
- Exporter les modifications dans un format qui permet aux autres de les intégrer et de les utiliser facilement.

patch et diff

L'une des méthodes les plus courantes pour effectuer le travail du noyau consiste à utiliser les programmes de **patch** et de **diff**. Pour utiliser ces outils, deux arborescences de répertoires différentes: une arborescence `clean` et une arborescence `working` doivent être utilisées. L'arbre `clean` est une version du noyau publiée, tandis que l'arbre `working` est basée sur la même version mais contient les modifications. Ensuite, on peut utiliser **patch** et **diff** pour extraire les modifications et les transférer vers une nouvelle version du noyau.

Générer un patch d'origine

Par exemple, créer deux répertoires contenant la dernière version du noyau.

```
tar -zxf linux-2.6.19.tar.gz
mv linux-2.6.19 linux-2.6.19-dity
tar -zxf linux-2.6.19.tar.gz
ls

  linux-2.6.19 /
  linux-2.6.19-dirty /
```

Maintenant, faire toutes les différentes modifications souhaitées dans le répertoire `-dirty` et laisser le répertoire original du noyau `clean`. Une fois les modifications terminées, il faut créer un patch pour l'envoyer à d'autres personnes:

```
diff -Naur -X linux-2.6.19/Documentation/dontdiff linux-2.6.19/
linux-2.6.19-dirty/ > my_patch
```

Cela va créer un fichier appelé `my_patch` qui contient la différence entre le travail et une arborescence de noyau 2.6.19 propre. Ce patch peut ensuite être envoyé à d'autres personnes.

Nouvelles versions du noyau

Si une nouvelle version du noyau est publiée et que l'on souhaite transférer les modifications vers la nouvelle version, il faut essayer d'appliquer le correctif généré sur une version de noyau propre. Cela peut être fait dans les étapes suivantes:

1. Générer le patch d'origine, comme dans l'exemple précédent.
2. À l'aide du correctif officiel de kernel.org, déplacer l'ancienne version du noyau d'une version:

```
cd linux-2.6.19
patch -p1 < ../patch-2.6.20
cd ..
mv linux-2.6.19 linux-2.6.20
```

1. Déplacer le répertoire de travail d'une version vers l'avant en supprimant le correctif, puis appliquer la nouvelle mise à jour:

```
cd linux-2.6.19-dirty
patch -p1 -R < ../my_patch
patch -p1 < ../patch-2.6.20
cd ..
mv linux-2.4.19-dirty linux-2.6.20-dirty
```

1. Essayer d'appliquer lee patch en plus de la nouvelle mise à jour:

```
cd linux-2.6.20-dirty
patch -p1 < ../my_patch
```

Si le correctif ne s'applique pas correctement, résoudre tous les conflits qui sont créés (la commande

de correctif informe du rejet en produisant les fichiers `.rej` et `.orig` à comparer et à corriger manuellement). Ce processus de fusion peut être la partie la plus difficile si on a apporté des modifications à des parties de l'arborescence source qui ont été modifiées par d'autres personnes.

Si on utilise ce processus de développement, il est recommandé d'utiliser l'ensemble de programmes **patchutils** (disponible sur https://cyberelk.net/tim/patchutils). Ces programmes permettent de manipuler facilement les correctifs de texte de toutes sortes de manières utiles et ont permis aux développeurs du noyau d'économiser de nombreuses heures de travail fastidieux.

Gérer les patchs avec quilt

Le développement du noyau en utilisant **patch** et **diff** fonctionne généralement très bien. Mais après un certain temps, la plupart des gens en ont assez et recherchent une façon de travailler différente qui n'implique pas tant de corrections et de fusions fastidieuses. Heureusement, quelques développeurs du noyau ont proposé un programme appelé **quilt** qui gère beaucoup plus facilement le processus de manipulation d'un certain nombre de correctifs par rapport à une arborescence source externe.

L'idée de **quilt** est venue d'un ensemble de scripts écrits par Andrew Morton qu'il a utilisé pour maintenir d'abord le sous-système de gestion de la mémoire, puis plus tard l'ensemble de l'arborescence du noyau de développement. Ses scripts étaient très étroitement liés à son flux de travail, mais les idées derrière eux étaient très puissantes. Andreas Gruenbacher a repris ces idées et a créé l'outil de **quilt**.

L'idée de base derrière **quilt** est que l'on travaille avec un arbre source vierge et qu'on ajoute un tas de correctifs dessus. On peut pousser et faire disparaître différents correctifs de l'arborescence source et gérer cette liste de correctifs de manière simple.

Pour commencer, créez une arborescence source du noyau:

```
tar -zxf linux-2.6.19.tar.gz
ls

linux-2.6.19/
```

Et aller dans ce répertoire:

```
cd linux-2.6.19
```

Pour commencer, créer un répertoire appelé `patches` qui contiendra tous les patchs du noyau:

```
mkdir patches
```

Dire ensuite à **quilt** de créer un nouveau patch appelé **patch1**:

```
quilt new patch1

  Patch patches/patch1 is now on top
```

quilt doit être informé de tous les différents fichiers qui seront modifiés par ce nouveau patch. Pour ce faire, utilisez la commande **add**:

```
quilt add Makefile

  File Makefile added to patch patches/patch1
```

Modifiez le fichier `Makefile`, modifiez la ligne **EXTRAVERSION** et enregistrer la modification. Après avoir terminé, dire à **quilt** d'actualiser le patch:

```
quilt refresh

Refreshed patch patches/patch1
```

Le fichier `patches/patch1` contiendra un patch avec les modifications que l'on viens d'apporter:

```
cat patches/patch1

  Index: linux-2.6.19/Makefile
  ===================================================================
  --- linux-2.6.19.orig/Makefile
  +++ linux-2.6.19/Makefile
  @@ -1,7 +1,7 @@
   VERSION = 2
   PATCHLEVEL = 6
   SUBLEVEL = 19
  -EXTRAVERSION =
  +EXTRAVERSION = -dirty
   NAME=Crazed Snow-Weasel

   # *DOCUMENTATION*
```

On peut continuer, travailler avec ce patch unique, ou en créer un nouveau pour aller au-dessus de ce patch. Par exemple, si trois correctifs différents ont été créés, **patch1**, **patch2** et **patch3**, ils seront appliqués les uns sur les autres.

Pour voir la liste des correctifs actuellement appliqués:

```
quilt series -v

  + patches/patch1
```

```
  + patches/patch2
  = patches/patch3
```

Cette sortie montre que les trois patchs sont appliqués et que le patch actuel est **patch3**.

Si une nouvelle version du noyau est publiée et que l'on souhaite transférer les modifications vers la nouvelle version, **quilt** peut gérer cela facilement avec les étapes suivantes:

- Retirer tous les correctifs actuellement sur l'arborescence:

```
quilt pop -a

  Removing patch patches/patch3
  Restoring drivers/usb/Makefile

  Removing patch patches/patch2
  Restoring drivers/Makefile

  Removing patch patches/patch1
  Restoring Makefile

  No patches applied
```

- À l'aide du correctif officiel de kernel.org, déplacer l'ancienne version du noyau d'une version:

```
patch -p1 < ../patch-2.6.20
cd ..
mv linux-2.6.19 linux-2.6.20
```

- Maintenant, demander à **quilt** de repousser tous les patchs au-dessus du nouvel arbre:

```
quilt push

  Applying patch patches/patch1
  patching file Makefile
  Hunk #1 FAILED at 1.
  1 out of 1 hunk FAILED -- rejects in file Makefile
  Patch patches/patch1 does not apply (enforce with -f)
```

- Comme le premier patch ne s'applique pas correctement, forcer l'application du patch, puis corriger:

```
quilt push -f

  Applying patch patches/patch1
  patching file Makefile
  Hunk #1 FAILED at 1.
  1 out of 1 hunk FAILED -- saving rejects to file Makefile.rej
  Applied patch patches/patch1 (forced; needs refresh)
```

```
vim Makefile.rej Makefile
```

- Une fois le patch appliqué à la main, actualiser le patch:

```
quilt refresh

  Refreshed patch patches/patch1
```

```
 continuer à pousser les autres patchs:
```

```
quilt push

  Applying patch patches/patch2
  patching file drivers/Makefile

  Now at patch patches/patch2

quilt push

  Applying patch patches/patch3
  patching file drivers/usb/Makefile

  Now at patch patches/patch3
```

quilt a également des options qui enverront automatiquement tous les correctifs de la série par e-mail à un groupe de personnes ou à une liste de diffusion, supprimer des correctifs spécifiques au milieu de la série, monter ou descendre la série de correctifs jusqu'à ce qu'un correctif spécifique soit trouvé et de nombreuses options plus puissantes.

quilt est fortement recommandé, pour n'importe quel type de développement du noyau, même pour suivre quelques patchs, au lieu d'utiliser la méthode **diff** et **patch** plus difficile. C'est beaucoup plus simple et ça fera économiser beaucoup de temps et d'efforts.

git

git est un outil de contrôle de code source qui a été écrit à l'origine par Linus Torvalds lorsque le noyau Linux cherchait un nouveau système de contrôle de code source. Il s'agit d'un système distribué, qui diffère des systèmes de contrôle de code source traditionnels tels que **cvs** en ce qu'il n'est pas nécessaire d'être connecté à un serveur pour effectuer une validation dans le référentiel.

git est l'un des systèmes de contrôle de code source les plus puissants, flexibles et rapides actuellement disponibles, et une équipe de développement active y travaille. La page Web principale de **git** se trouve à https://git.or.cz/. Il est recommandé à tout nouvel utilisateur de **git** de parcourir les tutoriels publiés afin de se familiariser avec le fonctionnement de **git** et comment l'utiliser correctement.

Le noyau Linux est développé à l'aide de **git**, et la dernière arborescence du noyau **git** peut être trouvée sur https://www.kernel.org/git/, avec une grande liste de référentiels **git** d'autres développeurs de noyau.

Il n'est pas nécessaire d'utiliser **git** pour faire le développement du noyau Linux, mais il est très pratique pour aider à traquer les bogues du noyau. Si on signale un bogue aux développeurs du noyau Linux, ils peuvent demander d'utiliser **git bisect** afin de trouver le changement exact qui a provoqué le bogue. Si c'est le cas, suivre les instructions de la documentation de **git** pour savoir comment l'utiliser.

ketchup

ketchup est un outil très pratique utilisé pour mettre à jour ou basculer entre les différentes versions de l'arborescence des sources du noyau Linux. Il a la capacité de:

- Trouver la dernière version du noyau, téléchargez-la et décompressez-la.
- Mettre à jour une version actuellement installée de l'arborescence des sources du noyau vers toute autre version, en corrigeant l'arborescence à la version appropriée.
- Gérer le développement différent et les branches stables de l'arbre du noyau, y compris les arbres **-mm** et **-stable**.
- Télécharger les correctifs ou tarballs nécessaires pour effectuer la mise à jour, s'ils ne sont pas déjà présents sur la machine.
- Vérifier les signatures GPG de l'archive tar et des correctifs pour vérifier qu'il a téléchargé un fichier correct.

ketchup peut être trouvé à https://www.selenic.com/ketchup/ et a beaucoup de documentation supplémentaire dans le wiki à https://www.selenic.com/ketchup/wiki/.

Voici un ensemble d'étapes qui montrent à quel point il est simple d'utiliser **ketchup** pour télécharger une version spécifique du noyau, puis de le faire basculervers une autre version du noyau avec seulement un nombre minimal de commandes.

Pour que **ketchup** télécharge la version 2.6.16.24 de l'arborescence des sources du noyau dans un répertoire et renomme le répertoire pour qu'il soit identique à la version du noyau, entrer:

```
mkdir foo
cd foo
ketchup -r 2.6.16.24

  None -> 2.6.16.24
  Unpacking linux-2.6.17.tar.bz2
  Applying patch-2.6.17.bz2 -R
```

```
  Applying patch-2.6.16.24.bz2
  Current directory renamed to /home/gregkh/linux/linux-2.6.16.24
```

Maintenant, pour mettre à niveau ce noyau pour qu'il contienne la dernière version stable du noyau, entrer simplement:

```
ketchup -r 2.6

  2.6.16.24 -> 2.6.17.11
  Applying patch-2.6.16.24.bz2 -R
  Applying patch-2.6.17.bz2
  Downloading patch-2.6.17.11.bz2
  --22:21:14--
https://www.kernel.org/pub/linux/kernel/v2.6/patch-2.6.17.11.bz2
             => `/home/greg/.ketchup/patch-2.6.17.11.bz2.partial'
  Resolving www.kernel.org... 204.152.191.37, 204.152.191.5
  Connecting to www.kernel.org|204.152.191.37|:80... connected.
  HTTP request sent, awaiting response... 200 OK
  Length: 36,809 (36K) [application/x-bzip2]
  100%[====================================>] 36,809        93.32K/s
  22:21:14 (92.87 KB/s) - `/home/greg/.ketchup/patch-2.6.17.11.bz2.partial'
saved [36809/36809]
  Downloading patch-2.6.17.11.bz2.sign
  --22:21:14--
https://www.kernel.org/pub/linux/kernel/v2.6/patch-2.6.17.11.bz2.sign
             => `/home/greg/.ketchup/patch-2.6.17.11.bz2.sign.partial'
  Resolving www.kernel.org... 204.152.191.37, 204.152.191.5
  Connecting to www.kernel.org|204.152.191.37|:80... connected.
  HTTP request sent, awaiting response... 200 OK
  Length: 248 [application/pgp-signature]
  100%[====================================>] 248           --.--K/s
  22:21:14 (21.50 MB/s) -
`/home/greg/.ketchup/patch-2.6.17.11.bz2.sign.partial' saved [248/248]
  Verifying signature...
  gpg: Signature made Wed Aug 23 15:01:04 2006 PDT using DSA key ID 517D0F0E
  gpg: Good signature from "Linux Kernel Archives Verification Key
>ftpadmin@kernel.org<"
  gpg: WARNING: This key is not certified with a trusted signature!
  gpg:          There is no indication that the signature belongs to the
owner.
  Primary key fingerprint: C75D C40A 11D7 AF88 9981  ED5B C86B A06A 517D
0F0E
  Applying patch-2.6.17.11.bz2
  Current directory renamed to /home/greg/linux/tmp/x/linux-2.6.17.11
```

Cela montre que **ketchup** a automatiquement déterminé que la dernière version stable était la 2.6.17.11 et a téléchargé les fichiers de correctifs nécessaires pour accéder à cette version.

Il est fortement recommandé d'utiliser **ketchup** lorqu'on souhaite télécharger des arborescences de

source de noyau Linux. Il fait tout le travail pour trouver où sur le serveur se trouve le fichier correctif correct et appliquera automatiquement le correctif au format correct, après avoir vérifié que le fichier téléchargé est correctement signé. Combiner **ketchup** avec **quilt** et on dispose d'une configuration très puissante qui contient tout ce dont on a besoin pour gérer efficacement les sources du noyau en tant que développeur de noyau Linux.