ネイティブ アプリケーションをホスティングするための RPM の構築

ここでは、ネイティブ アプリケーションをホストするための RPM の構築方法について説明します。

- ビルド環境のセットアップ，1 ページ
- ネイティブ RPM の構築，3 ページ

ビルド環境のセットアップ

ここでは、ネイティブアプリケーションホスティング対応にするビルド環境を準備し、設定する方法を説明します。

QEMU ハイパーバイザを使用したネイティブ ビルド環境の作成

ここでは、ネイティブ Wind River Linux 7.0 のビルド環境を作成し、クイック エミュレータ（QEMU）ハイパーバイザを使用して環境 ISO を実行する方法について説明します。

前提条件

- ネイティブ WRL7 ISO が含まれているシスコのリポジトリにアクセスできることを確認します。
- .iso 拡張子付きのネイティブ ISO をダウンロードします。

設定手順

1 ネイティブ WRL7 ISO を起動し、ディスク イメージにインストールします。

```
qemu-system-x86_64 -m 16G -cdrom <path-to-the-downloaded-iso-file> -net nic -net user
-hda ./wrl7.img
 -cpu core2duo -show-cursor -usb -usbdevice wacom-tablet -vga vmware
```

2 インストールされているイメージでネイティブ ビルド環境を再起動します。

```
qemu-system-x86_64 -m 16G -net nic -net user -hda ./wrl7.img -cpu core2duo -show-cursor
 -usb -usbdevice wacom-tablet -vga vmware
```

ネイティブ ビルド環境は、サードパーティ製アプリケーションをホストする準備が整っています。ユーザはネイティブ QEMU VM の VGA コンソール ポートに接続されます。

または、ユーザは VM 内で実行している SSH サービスに接続できます。

SDK シェル スクリプトを使用した相互ビルド環境の作成

WRL7 相互 SDK シェル スクリプトを使用して、相互ビルド環境をネイティブ環境の代わりとして作成することができます。Ubuntu 14.04 ホスト コンピュータなどの汎用 Linux 環境でシェルスクリプトを実行することによって、SDK をインストールできます。

前提条件

インストールを進める前に、次の要件が満たされていることを確認します。

- シスコのリポジトリの SDK へのアクセス。
- 必要に応じて、SDK にカスタマイズを構築できる機能。

インストール手順

ネイティブ アプリケーションをホスティングするための SDK をインストールするには、次の手順を実行します。

1 シスコのリポジトリから SDK をダウンロードします。

```
wget https://devhub.cisco.com/artifactory/xr600/app-dev-sdk/x86_64/
wrlinux-7.0.0.2-glibc-x86_64-intel_x86_64-wrlinux-image-glibc-std-sdk.sh
```

2 シェル スクリプトを実行して SDK をインストールします。

```
john@sjc-ads-4587:john$
./wrlinux-7.0.0.2-glibc-x86_64-intel_x86_64-wrlinux-image-glibc-std-sdk.sh
```

3 SDK のインストール先のディレクトリを入力します。

十分なストレージ領域があるターゲット ディレクトリを選択します。

```
Enter target directory for SDK
(default: /opt/windriver/wrlinux/7.0-intel-x86-64):
/nobackup/john/sdk_extract
You are about to install the SDK to "/nobackup/john/sdk_extract". Proceed[Y/n]? Y
```

正常にインストールされると、メッセージが画面に表示されます。

```
Extracting SDK...done
Setting it up...done
SDK has been successfully set up and is ready to be used.
```

ネイティブ アプリケーションのホスティングを行う SDK が正常にインストールされます。

次の作業

環境変数を設定して、この項で説明するようにそれらを検証することができます。

1. 次のコマンドを実行することによって SDK がインストールされ、環境変数が設定されているディレクトリに移動します。
   - bash シェルを使用する場合は、**. ./env.sh** コマンドを実行します。
   - 他のシェルについては、**source . ./env.sh** コマンドを実行します。

   コマンドは、SDK インストール時に抽出された環境設定ファイルを実行します。
2. **env** コマンドを実行してすべての変数値を表示し、インストールされている環境変数を検証します。
3. **env | grep CC** コマンドを実行し、次の値が割り当てられているかどうかを確認して CC 環境変数を検証します。

   ```
   CC=x86_64-wrs-linux-gcc  -m64
   --sysroot=/nobackup/john/sdk_extract/sysroots/core2-64-wrs-linux
   ```

   または、echo コマンドを使用できます。

   ```
   echo $CC
   x86_64-wrs-linux-gcc -m64
   --sysroot=/opt/windriver/wrlinux/7.0-intel-x86-64/sysroots/core2-64-wrs-linux
   ```
4. SDK がインストールされている場合、PATH 環境変数がベース ディレクトリを指しているかどうかを確認します。

   パスを確認するには、**env | grep PATH** コマンドを実行して、次のパスが表示されるかどうかをチェックします。

   ```
   PATH=<sdk_extract>/sysroots/
   x86_64-wrlinuxsdk-linux/usr/bin:
   <sdk_extract>/sysroots/x86_64-wrlinuxsdk-linux/usr/bin/x86_64-wrs-linux
   ```

   または、echo コマンドを使用できます。

   ```
   echo $PATH
   <sdk_extract>/sysroots/x86_64-wrlinuxsdk-linux/usr/bin:
   <sdk_extract>/sysroots/x86_64-wrlinuxsdk-linux/usr/bin/x86_64-wrs-linux
   ```
5. アプリケーションのソース コードを含むディレクトリに移動し、アプリケーションの構築を開始します。

（注） SDK のルート ファイル システムからすべての *.la 一時ファイルを削除する必要があります。これを実行するには、次のコマンドを使用します。

```
bash# cd  <sdk_extract>/sysroots/
bash# find . -name \*.la | xargs rm -f
```

ネイティブ RPM の構築

ここでは、ネイティブ環境または相互ビルド環境を使用してアプリケーションを構築する手順を説明します。ネイティブ ビルド環境を使用することを推奨します。

ソース コードからアプリケーションを構築するには、2 つの方法があります。

1 つは、ソース・コードのアーカイブからアプリケーションを構築する方法です。これは、この項で説明します。もう 1 つは、ソース RPM から構築する方法ですが、これは推奨されません。

前提条件

先に進む前に、次の要件が満たされていることを確認します。

- アプリケーションのビルド環境がネイティブ ビルド環境または相互ビルド環境のいずれかを使用するようにセットアップされていることを確認する。
- README ファイルを読み、アプリケーションの構築に関する構築プロセスを理解する。

設定手順

アプリケーションを構築するには、次の手順を実行します。

1. アプリケーションのソース コードが含まれているディレクトリに移動します。
2. 次のコマンドを実行して、アプリケーションを抽出します（圧縮されている場合）。

   ```
   bash-4.1$ tar xzvf tcpdump-4.7.4.tar.gz
   ```

3. ディレクトリをアプリケーション ディレクトリに変更します。

   ```
   bash-4.1$ cd tcpdump-4.7.4
   ```

4. アプリケーションを構築し、実行可能ファイルを生成します。

   ```
   tcpdump-4.7.4$ ./configure
   tcpdump-4.7.4$ make
   ```

5. ディレクトリにある実行可能ファイルを確認します。

   ```
   tcpdump-4.7.4$ ls -1 ./tcpdump
   -rwxr-xr-x 1 john eng 3677288 Jun 15 23:10 ./tcpdump
   ```

   実行ファイルは `tcpdump` としてリストされています。

実行可能ファイルは、IOS XR でアプリケーションをホストするためにパッケージ化される準備が整っています。

次の作業

IOS XR にインストールできるようにアプリケーションのバイナリをパッケージ化します。

推奨されるパッケージ化の形式は、IOS XR でホスト可能な RPM です。

RPM を構築するには、次の手順を実行します。

- .spec ファイルが必要です。
- **rpmbuild** コマンドを実行する必要があります。

次の手順を実行して、バイナリをパッケージ化します。

1. SPECS ディレクトリに .spec ファイルを作成します。

   ```
   # %define __strip /bin/true

   Name: tcpdump
   ```

```
Version: 4.7.4
Release: XR
Buildroot: %{_tmppath}/%{name}-%{version}-%{release}-root
License: Copyright (c) 2015 Cisco Systems Inc. All rights reserved.
Packager: mark
SOURCE0 : %{name}-%{version}.tar.gz
Group: 3'rd party applicaiton
Summary: Tcpdump cross compiled for WRL6

%description
This is a cross compiled version of tcpdump using IOS XR sdk for WRL7

%prep

%setup -q -n %{name}-%{version}

%build
  # This where sdk is being sourced
source /nobackup/mark/sdk_extract_18/tmp/env.sh
 ./configure
 make


 %install
rm -rf ${RPM_BUILD_ROOT}
 # make DESTDIR=$RPM_BUILD_ROOT install
mkdir -p ${RPM_BUILD_ROOT}%{_sbindir}
 install -m755 tcpdump ${RPM_BUILD_ROOT}%{_sbindir}

%files
 %defattr(-,root,root)
 %{_sbindir}/tcpdump

%pre

 %post

 %preun

 %postun

%clean
rm -rf $RPM_BUILD_ROOT
```

2 RPM を構築します。

```
mark@tenby:redhat$ cd /usr/src/redhat/SPECS/
mark@tenby:SPECS$ rpmbuild -ba tcpdump.spec
```

使用する RPM のビルドは、5.4.14 バージョンです。

3 バイナリが RPMS ディレクトリに構築されていることを確認します。

```
mark@tenby:x86_64$ pwd /usr/src/redhat/RPMS/x86_64

mark@tenby:x86_64$ ls

tcpdump-4.7.4-XR.x86_64.rpm
```

ネイティブアプリケーションをホストする準備が整いました。ネイティブアプリケーションのホスティングについては、ネイティブアプリケーションとしての iPerf の実行を参照してください。