Embedded Linux

danke Yocto und OpenEmbedded

Von Marco Israel

Dezember 2019

Inhaltsverzeichnis

Inhaltsverzeichnis ii

Abbildungsverzeichnis iii

Abkürzungsverzeichnis v

Glossar viii

1 Einleitung 1

1.1 Über Yocto, OpenEmbedded, poky, meta-daten und Bitbake 1

1.2 Docker 2

2 Setup your host 3

2.1 Erforderliche Pakete 3

2.2 Host Konfiguration 3

3 Setup Eclipse 7

3.1 Installation, Einrichtung und Plugins 7

3.2 Cross Compile und Remote Debugging Einstellungen 7

4 Source code development und debugging 9

4.1 Cross development 9

4.1.1 Entwicklung mit dem Yocto SDK 9

4.1.2 Entwicklung mit Yocto Tools 10

4.2 Remote Debugging 10

Literaturverzeichnis 11

Abbildungsverzeichnis

Abkürzungsverzeichnis

SDK Software Development Kit

PDF Portable Dokument Format

Glossary

B | D | M | R | S | Y

B

Bitbake Bitbake ist ein Framework ähnlich wie GNU Make, bestehend aus Phython Srcipten welche das Erstellen von Linux Distributionen mittels Metadaten koordiniert. . 1

D

Docker Docker ist eine Software zur virtualisierung von einzelnen Anwendungen. 2

M

Metadaten Metadaten oder Metainformationen sind Informationen über andere Daten. 1

Metadaten Metadaten oder Metainformationen sind Informationen über andere Daten. vii

Metadatendatei Eine Datei, welche Metadatens (Informationen zu anderen Daten) über ndere Daten enthält.. 1

R

Recipies Rezepte oder Anleitungen. 1

rootfs Unter Linux wird das urspüngliche *rootfs* als der Ort bezeichnet, von dem **ausgehend** alle weiteren Verzeichnissbäume eingehängt sind/werden. Beispielweise liegen direkt unterhalb des rootfs die Ordner /home, /boot /root oder /bin.. 9

S

SDK Ein Software Development Kit (SDK) ist eine gekapselte bzw. vordefinierte Umgebung zur Entwicklung von Softwarekomponenten. Die Umgebung stellt (vorkonfigurierte) Sammlungen von ausgewählten Programmierwerkzeugen oder auch Liberias bereit, die zur Entwicklung für eine Zielplattform benötigt werden.. 9

Y

Yocto Project Eine Community Gruppe welche eine Software Buildumgebung pflegt und weiterentwickelt, sowie Metadaten für diese Buildumgebung zur verfügung stellt um ein minimalistisches Linux system mit grundlegenden Tools unter der virtuallisierungsumgebung QEMU zu starten.. 1

1 Einleitung

1.1 Über Yocto, OpenEmbedded, poky, meta-daten und Bitbake

Das Yocto Project bezeichnet eine Community Gruppe welche sich zur Aufgabe gemacht hat, dass erstellen von Linux Distributionen für eingebettet System zu vereinfachen.

Zusammen mit der OpenEmbedded Community pflegt das Yocto Project eine Software Build Umgebung mit dem Namen Bitbake, bestehend aus Pythonskripten, welches das Erstellen von Linux Distributionen koordiniert. Des Weiteren pflegen beiden Communities Dateien mit Metadaten (Metadatendatei die in Form von Regeln beschreiben wie Software Pakte innerhalb für unterschiedliche Distribution und Hardware durch Bitbake gebaut werden müssen. erforderlich sind.
Portable Dokument Format (PDF)

Diese Regeln werden Recipies genannt. Regeln bzw. diese Metadaten und somit auch das Build-System Bitbake arbeiten nach einen Schichten Modell. Dabei beschreiben Meta-Daten auf unterster Schicht allgemeine grundlegende Anleitungen zum Übersetzten der wichtigsten Funktionen eines Betriebssystems. Höhere Schichten erweitern (detaillieren) oder überschreiben diese grundlegenden Rezepte in immer höheren Schichten. So setzt auf Beschreibungen der Hardware (BSP, Board Support packed Schichten) Schichten der Software- und Anwendungsschicht auf. Ein solches Schichtenmodell ermöglicht es, einzelne Schichten auszutauschen oder darunterliegende Einstellungen abzuändern. Die Yocto community stellt Meta-Daten / Recipes bereit, um ein minimalistisches Betriebssystem mit grundlegenden Linux Tools unter der Virtualisierungsumgebung QEMU für verschiedene Architekturen starten zu können. Die OpenEmbedded Community stellt Meta-Daten Rezepte bereit, die auf dieses minimalistische Betriebssystem aufsetzten und genutzt werden können um ein Betriebssystem nach eigenen Wünschen gestalten (zusammen setzten) und für Hard-

ware Plattformen konfigurieren zu können. Hierzu gehören sowohl Hardware als auch open Source Software Beschreibungen. Beide Communities pflegen und erweitern das Build Umgebung „Bitbake“, welches verschiedene Aufgaben in geregelter Reihenfolge im Multicore Betrieb durchführt:

- Herunterladen (fetchen) und Sammeln von Source Dateien (z.B. Git Repositorien, ftp Servern oder lokalen Dateien.)
- Übersetzten, konfigurieren, patchen, installieren, verifizieren usw. von Paketen auf mehreren Prozessor Kernen
- Bereitstellen von Entwicklung und Verwaltungstools

Das Minimalisten Betriebssystem der Yocto Community wird „poky“ genannt.

1.2 Docker

Bitbake benötigt, neben einem aktuellen Python 2 Interpreter, verschiedene Tools. So u.a. Git zum Herunterladen von Source Detain. Alle diese Abhängigkeiten wurden in einem Docker Container zusammengefasst. Bitbake kann unter unschädlichen Betriebssystemen und Plattformen auf gleiche Weise genutzt werden, ohne dass es und seine Abhängigkeiten neu konfiguriert werden müssen. Im Gegensatz zu anderen Virtualisierungstechniken hat Docker trotz Virtualisierungstechniken keinen großartigen Performance Verlust. Von einem Gebrauch von klassischen Virtualisierungstechniken ist aus Performance Gründen dringend abzuraten.

2 Setup your host

2.1 Erforderliche Pakete

Das yocto Docker file zeigt eine Liste aller nötigen Ubuntu Pakte die zum Arbeiten mit der Yocto / OpenEmbedded Build Umgebung auf einem Host benötigt werden, sollte nicht mit dem Docker Image gearbeitet werden wollen.

Zusätzlich sind die nachfolgenden Pakete werden zum Arbeiten auf dem lokalen Host benötigt. Nähere Informationen zu den Paketen, Quellen, Konfigurationen usw. sind auf verschiedenen Internetseiten zu finden.

- git
- docker
- TFTP Server
- NFS Server
- microcom
- eclipse
- qt5
- qt5Creator
- openssh-server

2.2 Host Konfiguration

Nachfolgende Pakete benötigen weitere Konfigurationen

Docker:

- Docker Service starten
- Docker-yocto Image bauen: „docker build -t yocto . “
- Hilfe liefert docker –help oder die Internetseite
- Image starten durch ausführen von „./run.sh bash “

TFTP Server:

- TFTP Austausch Ordner anlegen und Zugriffsrechte definieren
- stat-alone deamon (/etc/default/tftpd-hpa) oder xinitd Service (/etc/-xintd.d/tftp) konfigurieren
- Server neu starten
- **BEISPIEL** im „BSP Manual“unter phytec.de; Stichwort „Booting the Kernel from Network“(Booting_the_Kernel_from_Network)

NFS Server:

- NFS Server konfigurieren (/etc/exports)
- NFS Server neu starten
- **BEISPIEL** im „BSP Manual“unter phytec.de; Stichwort „Booting the Kernel from Network“(Booting_the_Kernel_from_Network)

Microcom

- Der Parameter –port Definiert die Serielle Schnittstelle.
- Weiteres ist unter Manual Seite zu finden.

Eclipe

- Weiters im Kapitel 3; Seite 7

QT5Creator

- Weiters im Kapitel ??; Seite ??

Yocto Areitsverzeichnis

- Erstellen eines globalen Arbeitsverzeichnisses. Z.B. /opt/yocto
- Setzten der Rechte rwx Rechte für alle user („others“).

Python2 als standart interpreter Yocto/Openembedded Tools bauen auf Python2 auf. Daher ist es nötig einen symlink oder alias auf Python2 zu setzten. Z.b. *alias python=python2*

Proxy und Routen Je nach Netzwerkinfrastruktur müssen Proxy und Netzwerkrouten auf dem lokalen Host gesetzt werden. Beispielsweise für die Tools:

- Git
- wget
- apt-get
- https_proxy und http_proxy

Informationen hierzu liefert die das Yocto Manual oder das Yocto Wiki unter dem Stichwort „**Working Working Behind a Network Proxy** "(Working_Behind_a_Network_Proxy)

3 Setup Eclipse

3.1 Installation, Einrichtung und Plugins

Zur Entwicklung von C/C++ kann Eclipse CDT verwendet werden.

Zudem sind zum Cross compilieren und Remote Debuggen sowie zum remote Deployen nachfolgende Plugins bzw zusätzliche Eclipse-Softwaremodule nötig: (*Help -> Install new Software*)

- C/C++ Remote (Over TCF/TE) Run/Debug Launcher.
- Remote System Explorer User Actions
- TM Terminal via Remote System Explorer
- TCF Target Explorer

Des weiteren stellt die Yocto download Seite downloads.yoctoproject.org ein SDK Plugin bereit, welches das Konfigurieren von Remote Einstellungen für jeweilige Entwicklungsprojekt vereinfacht und zusammenfasst. Zum installieren muss beispielsweis zu den Installationsquellen in eclipse **http://downloads.yoctoproject.org/releases/eclipse-plugin/2.6.1/oxygen/** hinzu gefügt werden.

3.2 Cross Compile und Remote Debugging Einstellungen

TODO

4 Source code development und debugging

4.1 Cross development

Es existieren verschiedene Wege um Anwendungen für Zielplattformen zu entwickeln. Yocto setzt dabei auf die beiden nachfolgenden; zum einen die Bereitstellung einer gekapselten Entwicklungsumgebung in Form eines SDKs, zum anderen die Entwicklung unter Zuhilfenahme der Buildumgebung Bitbake mitsamt seiner Tools.

Nachfolgend beide Wege im Überblick:

4.1.1 Entwicklung mit dem Yocto SDK

Dieser Weg dient der normalen Entwicklung von Softwarekomponenten. Das Yocto SDK liefert eine gekapselte Shell-Umgebung mit vordefinierten Umgebungvariablen sowie allen nötigen Header Dateien, Libraries usw. welche auf der Zielplattform vorhanden sein werden. Hierzu bildes es die Ordnerstruktur (das s.g. rootfs) der Zielplattform in einem Ordner ab.

1. **Shell Umgebungsvariablen laden** Um die SDK Umgebung zu nutzen ist es nötig, eine Konfigurationsdatei in die Shell zu laden.

Listing 4.1: Einrichten der SDK Umgebung

```
1 [user@host]/yoctopath$:source ./enviroment-setup-*.sh
```

2. **Eclipse starten** Abhängig vom Anwendungfall, anschließend in der selben Shell Eclipse oder QTCreator starten.

Listing 4.2: Start von Eclipse in der zuvor konfiguierten Umgebung

```
1 [user@host]/yoctopath$:eclipse &
```

4.1.2 Entwicklung mit Yocto Tools

Diese Software Entwicklungsart dient mehr dazu Änderungen an existierendem Sourcecode durchzuführen. Hierbei stehen zwei tools zur Verfügung:

devtool <cmd> Zum einen steht das Yocto-Tool **devtool** mit verschiedene Kommandos bereit. Eine Übersicht über Kommandos und Anwendungsbeispiele liefert die Yocto Webseite oder *devtool –help*.

bibake -c devshell <recipie> Erzeugt bzw. öffnet eine vorkonfigurierte Shell ensprechend den Metadaten eines *recipie* zu einer Source Datei.

4.2 Remote Debugging

Literaturverzeichnis