Dokumentation
doc120, Version 0.1

Das Styx-Protokoll und Matlab

6. Februar 2013

Patrick Frenzel

Kurzfassung

Dieses Dokument soll einen kurzen Einblick in das Styx-Protokoll, sowie dessen Verwendung mit Matlab, geben.

Inhaltsverzeichnis

1 Das Styx-Protokoll

Das Styx Protokoll (auch 9P) ist ein Netzwerk-Protokoll und Teil des Plan 9 Betriebssystems von Bell Labs [1]. Von diesem Protokoll werden zwei Implementierungen verwendet.

Zum einen das estyx-Protokoll. Diese Implementierung ist vorallem für embedded-devices entwickelt worden. Es stellt einen Server auf dem remote-device bereit. Das angepasste estyx Protokoll steht als binary von TI zur Verfügung und muss für die aktuelle Anwendung nicht angepasst werden. Sollte dies allerdings doch einmal notwendig sein, steht der Quelltext, in allgemeiner Form zur Verfügung. Dieser Quelltext müsste dann für das entsprechende Device angepasst werden.

Zum anderen das cstyx-Protokoll. Dieses stellt eine client-Library zur Verfügung, die eine Kommunikation mit dem estyx-Server ermöglicht. Es gibt bereits Implementierungen dieses Protokolls in Java,Python und LabView. Die Python Implementierung beinhaltet eine, zur direkten Kommunikation mit dem Server. Von TI wird empfohlen hierfür Python v2.4 zu Verwenden. Neuere Versionen könnten zu Problemen führen.

Zur Kommunikation über die USB-Schnittstelle setzt cstyx auf die Shared-Library libusb auf. Da diese Bibliothek sowohl für Linux, Windows und MAC zur Verfügung steht ist das Protokoll weitest gehend unabhängig. Normalerweise müsste deshalb eine Compilierung des Quellcodes für das entsprechende System genügen. Dies ist leider nicht der Fall. Da einerseits einige Fehler im Quelltext enthalten sind. Diese Fehler sind wahrscheinlich darauf zurück zu führen, dass der Quellcode über vier Jahre alt ist [2] und der Compiler frühere Warnungen nun als Fehler erkennt. Dazu kommt noch, dass das Protokoll in Matlab teilweise zu abstürzen führt.

Deshalb wird in der folgenden Dokumentation das Vorgehen zur Erstellung der Shared-Library unter Linux für Matlab dokumentiert.

[1] `http://www.plan9.bell-labs.com/wiki/plan9/plan_9_wiki/`
[2] `http://developer.berlios.de/projects/cstyx/`

2 Regel zur Erstellung eines USB-Device unter Linux

Unter Linux ist es zunächst erforderlich eine udev Regel anzulegen, damit man später auch die nötigen Rechte hat um auf das USB-Device zuzugreifen.

Dazu muss eine *.rules Datei im Verzeichnis '/etc/udev/rules.d' erstellt werden. z.B. "40-TI_USB.rules". Die vorangestellte Nummer sollte kleiner sein als die bereits vorhandenen System-Regeln, um zu gewährleisten, dass die eigene Regel zuerst erkannt und umgesetzt wird. Eine Regel kann z.B. wie folgt aussehen:

```
1 SUBSYSTEM=="usb", SYSFS{idVendor}=="0451",   //Bedingungen
2 SYMLINK+="TI-USB-device", OWNER="hugo"       //Anweisungen
```

Die erste Bedingung ist zwingend erforderlich um zu kennzeichnen, dass es sich um ein USB-Gerät handelt. Mit der zweiten Bedingung wird das USB Gerät genauer spezifiziert. Die Vendor-ID kann mithilfe des Linux-Kommando

```
1 lsusb -v
```

ermittelt werden. Werden mehrere TI-Devices verwendet empfiehlt es sich auch noch nach der Produkt-ID zu unterscheiden.

Die erste Anweisung legt einen symbolischen Link im Ordner "/dev"für das entsprechende Gerät, mit dem Namen "TI-USB-device", an. *OWNER* bestimmt nun noch den Besitzer des Gerätes.

Nachdem die udev-Regel erstellt wird muss der udev-service noch neugestartet werden. Dies kann entweder über einen Neustart des Systems erfolgen oder über die Kommando-Zeile mithilfe von:

```
1 service udev reload
```

Abschließend muss das Gerät noch Neugestartet werden. Nun müsste im Ordner ein symbolischer Link, mit dem Namen *TI-USB-device*, erscheinen.

Nun kann jedes Programm des Benutzers auf das entsprechende USB-Gerät zugreifen.

3 Compilieren der cstyx Library

Zum Compilieren wird der *g++ Compiler* benötigt. Der eingfachheit halber wird die Kompilierung der Shared-Library mithilfe der Linux-Konsole erläutert.

Cstyx bietet die Möglichkeit einen Test-Server in die Shared-Library einzubinden. Auf diese Option wird an dieser Stelle allerdings verzichtet. Da dies nicht so trivial ist und für die aktuelle Anwendung auch nicht nötig.

Beim erstmaligen Compilieren werden einige Fehlermeldungen ausgegeben. Wie diese zu beheben sind wird im folgenden Abschnitt erläutert.

3.1 Änderungen am cstyx Quelltext

In der Datei *cstyx.cc* ist eine falsche Bibliothek eingebunden. Hier muss die Bibliothek *string* durch Bibliothek ***c****string* ersetzt werden. In der Datei *styxmsg.cc* fehlt diese Bibliothek komplett und muss hier noch mit eingebunden werden.

In der Header-Datei *cstyx.h* ist zudem die Funktion *cstyx_path_isdir* falsch deklariert. Hier muss die entsprechende Zeile durch die Funktionsdeklaration der Source-File ersetzt werden.

In der Header-Datei *Fid.hh* muss die Zeile 62:

```
1 Fid\& Fid::operator=(const Fid\& f);
```

durch

```
1 Fid\& operator=(const Fid\& f);
```

ersetzt werden.

Zusätzlich ist die Funktion *cstyx_path_isdir* falsch im Header deklariert. Hier muss die Funktionsdeklaration des Header mit der des Source-Code überschrieben werden.

Zu guter Letzt muss noch die *usb.h* Header-Datei ersetzt werden. Der enthaltene Header passt nicht zu der libusb Library **TODO: Für windows!?**. Der passende Header kann z.B. hier entnommen werden. Dabei ist zu beachten, dass die Version 0.x auszuwählen ist. Die 1.x Version scheint nicht kompatibel zu sein. Eine ausführliche Dokumentation dieser Libary befindet sich auf der Seite `http://www.libusb.org/`

Wenn man nun die Shared-Library erzeugt, werden nur noch 2 Warnungen ausgegeben, die allerdings Ignoriert werden können.