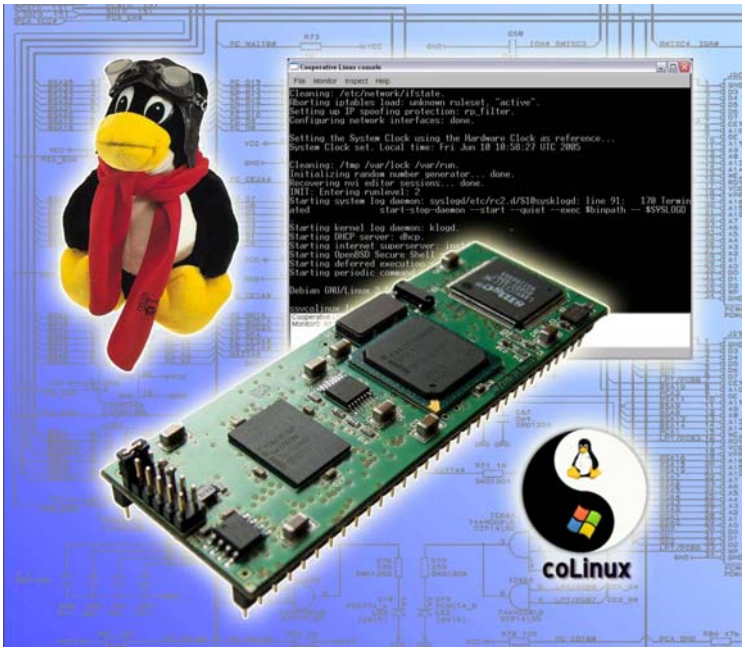
Mit coLinux unter Windows XP entwickeln

Zur Programmierung von Embedded-Linux-Systemen wird häufig auch ein Linux-PC benötigt. Dadurch entstehen für viele erfahrene Windows-Benutzer unerwartete Probleme. Mit der SSV coLinux Cross CD-ROM kann eine DIL/NetPC-basierte Embedded-Linux-Anwendung vollständig unter Windows XP entwickelt werden.

Als Embedded-Betriebssystem besitzt Linux nach wie vor eine stetig wachsende Anwendergemeinde. Auf PCs hingegen konnte sich das alternative Betriebssystem bisher nicht durchsetzen. Hier ist Microsoft-Windows das Maß der Dinge. Daran wird sich wohl auch in absehbarer Zeit nichts ändern.



Für die Softwareentwicklung nutzt der Embedded-Linux-Entwickler in den meisten Fällen den lizenzkostenfreien GCC (GNU C/C++ Compiler) nebst Zubehör. Als störend empfinden viele Anwender, dass die GNU-Werkzeuge primär für Linux-basierte PCs zur Verfügung stehen. Die Versuche den GCC unter Windows laufen zu lassen, um ein Embedded-Linux-System zu programmieren, führten bisher zu vielfältigen Problemen. Das recht verbreitete Cygwin – als Unix/Linux-Emulationsumgebung unter Windows – ist viel zu langsam und verursacht in den meisten Fällen erhebliche Bibliotheks- und Versionskonflikte. Die kommerziellen Versuche einiger Firmen, den GCC als Windows-Anwendung anzubieten, führten – trotz der hohen Kosten – nicht zu wirklich brauchbaren Produkten. Auch hier treten in der Regel ebenfalls erhebliche Bibliotheksprobleme auf. Somit bleibt vielen Anwendern nichts anderes übrig, als eine Linux-Distribution als zweites Betriebssystem auf den Windows-PCs zu bringen, um die GCC-Werkzeuge nutzen zu können. Besonders in großen Firmen ist diese technisch recht einfache Lösung nicht ohne weiteres möglich, da eine IT-

Abteilung für die PCs verantwortlich ist. Da darf man nicht einfach ein neues Betriebssystem installieren, um dieses statt Windows zu booten.

Eine weitere Hemmschwelle ist vielfach der Umgang mit einem neuen PC-Betriebssystem. Man muss sich in der neuen Umgebung erst einmal zurecht finden. Da tauchen vielen Fragen auf. Typische Beispiele sind: Wie kann ich die Netzwerkeinstellungen (IP-Adresse usw.) verändern? Wo finde ich einen Telnet-Client? Welchen Editor soll ich benutzen? Wie werden weitere Softwarekomponenten nachträglich installiert? Die Aufzählung ließe sich beliebig fortsetzen. Unter dem Strich wird erhebliche Einarbeitungszeit benötigt, um sich als sachkundiger Windows-Benutzer in der neuen Umgebung zurechtzufinden.

1. Eine typische DIL/NetPC-Entwicklungsumgebung

Die Abbildung 1 zeigt die wichtigsten Werkzeuge und deren Verbindungen für eine Embedded-Linux-Entwicklungsumgebung. Ein Linux-basierter PC dient zum Erstellen, Bearbeiten und Übersetzen der C/C++-Quellcodes mit Hilfe der GNU-Werkzeuge. Der Test erfolgt im Zielsystem, in diesem Fall ein Embedded-Linux-Modul aus der DIL/NetPC-Familie [1]. Zwischen dem Linux-Modul und dem PC gibt es eine Ethernet-LAN- und eine RS232-Verbindung. Der LAN-Link dient primär zum Dateitransfer und Benutzerdialog. Durch das Embedded-Linux kann hierfür zum Beispiel FTP (File Transfer Protocol) oder TFTP (Trivial File Transfer Protocol) benutzt werden. Für den Dialog mit der Benutzeroberfläche (Shell) eines Embedded-Linux kommt Telnet zum Einsatz. Auf dem PC wird ein Telnet-Client gestartet. Dieser bietet eine einfache (kommandozeilenorientierte) Oberfläche (CLI = Command Linux Interface) zum Embedded-Linux, um mit den Softwarekomponenten des Embedded-Systems zu kommunizieren. Die entsprechenden Server/Client-Bausteine für FTP, TFTP, Telnet usw. findet man sowohl in einem PC-Linux als auch in fast jedem Embedded-Linux.

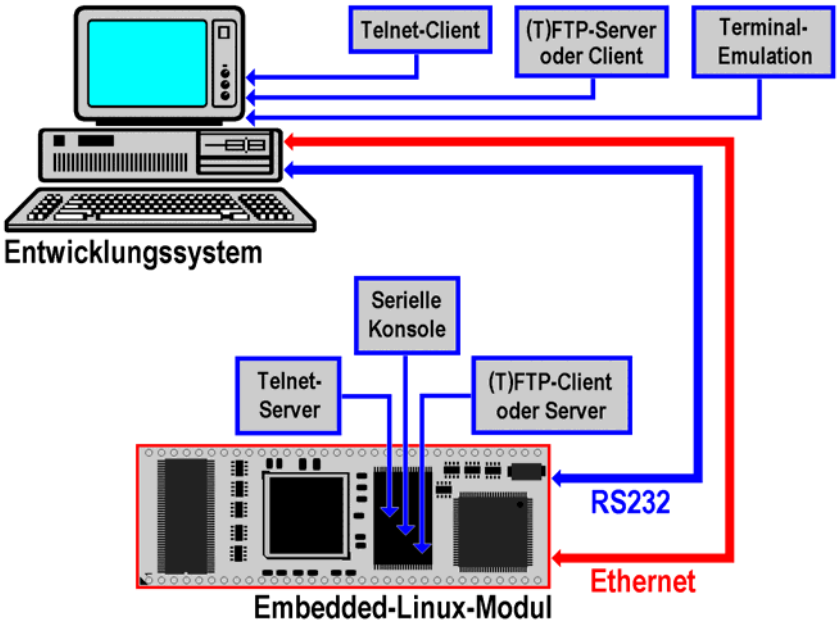


Abbildung 1: Eine typische Entwicklungsumgebung für DIL/NetPCs

Neben dem Dateitransfer- und Telnet-Dialog dient die Ethernet-LAN-Verbindung zwischen beiden Rechnern auch für das sogenannte Remote Debugging. Für die anwendungsbezogene Programmierung eines Embedded-Linux-Systems wird kein Hardware-Debugger benötigt. In den GNU-Werkzeugen findet man mit GDB und GDBserver die entsprechenden Bausteine für den ferngesteuerten Softwaretest per LAN- oder RS232-Verbindung.

Die RS332-Verbindung zwischen PC und Linux-Modul dient bei einem DIL/NetPC auch zur CLI-basierten Kommunikation mit dem Boot Loader. Unterhalb des Embedded-Linux-Betriebssystems findet man im Flash eines DIL/NetPCs einen solchen Boot Loader (zum Beispiel U-Boot bei den ARM-basierten DIL/NetPCs oder dBUG bei einem DIL/NetPCs mit Coldfire).

2. coLinux ersetzt den Linux-PC

Eine Alternative zum Einsatz eines Linux-PCs bildet das lizenzkostenfreie Cooperative Linux (coLinux). Dieses Linux-Projekt ist ein neuartiger Ansatz, um den Linux-Kernel unter Windows XP laufen zu lassen [2]. Das coLinux enthält spezielle Windows-Treiber. Dadurch kann coLinux als Gastbetriebssystem mit allen Privilegien unter Windows XP ablaufen. Es hat beispielsweise direkten Zugriff auf die Seitentabellen und schaltet den PC-Prozessor zwischen den Windows-Programmen und coLinux hin und her. Die Hardware ist soweit virtualisiert, dass es praktisch keine Zugriffskonflikte gibt.

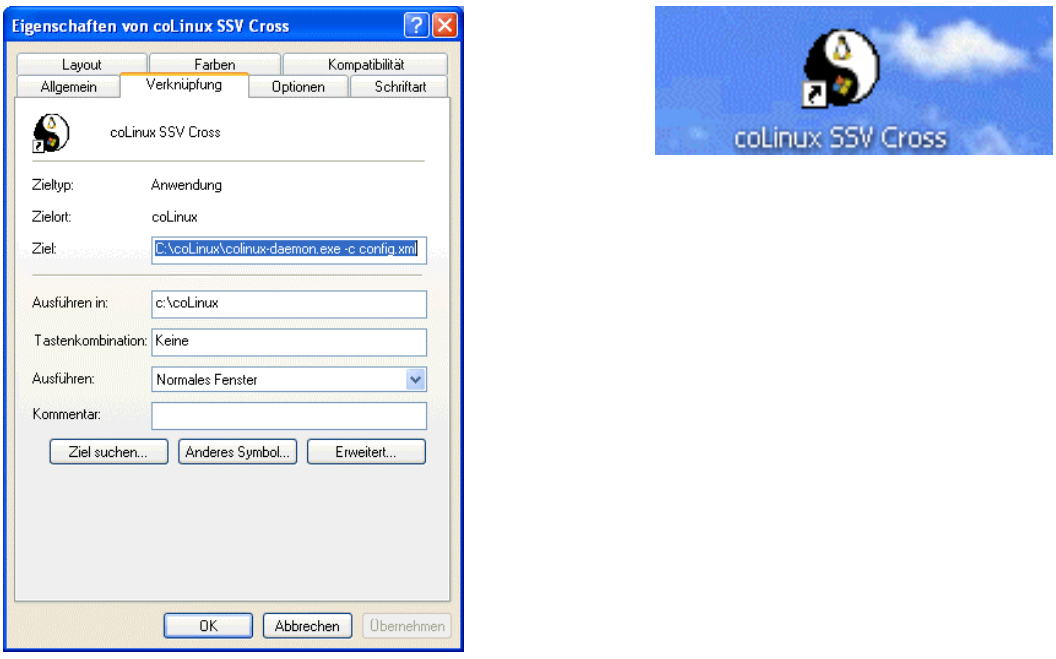


Abbildung 2: Erzeugen eines coLinux-Icons auf dem Windows-Desktop

Um coLinux zur Softwareentwicklung für DIL/NetPC-basierte Embedded-Linux-Anwendungen nutzen zu können, stellt SSV Embedded Systems seit Juni 2005 die **coLinux Cross Tools CD-ROM** zur Verfügung. Diese CD-ROM enthält in

einem Unterverzeichnis mit dem Namen *\SSV-colinux* ein vollständig vorkonfiguriertes coLinux für Windows XP-PCs. Ältere Windows-Versionen werden durch coLinux nicht unterstützt.

Die Installation des SSV-coLinux benötigt ungefähr 2 GBytes freien Speicherplatz auf der Windows XP-Festplatte. Der gesamte Installationsprozess wird über einen einzigen Schritt aktiviert: es muss lediglich die (Stapelverarbeitungs-) Datei *\SSV-colinux\setup.bat* mit Hilfe des Windows XP- (Datei-) Explorer gestartet werden. Die Installation schlägt das Zielverzeichnis *c:\programme\colinux* auf der Windows-Festplatte vor. **Verändern Sie diesen Vorschlag in *c:\colinux***.

Nachdem die einzelnen Installationsschritte automatisch durchlaufen wurden und sich das Installationsprogramm beendet hat, sollte für coLinux ein Icon auf dem Windows XP-Desktop erstellt und mit den entsprechenden Eigenschaften ausgestattet werden (Abbildung 2). Um coLinux zu starten, ist die Kommandozeile

```
c:\colinux\colinux-daemon.exe -c config.xml
```

erforderlich. Diese Befehlszeile ist auch in den Eigenschaften des coLinux-Icons abzulegen. Sie eignet sich allerdings auch, um das SSV-coLinux direkt per Windows-Eingabeaufforderung zu aktivieren.

3. Der erste Eindruck von coLinux

Die durch coLinux virtualisierte Hardware umfasst eine VGA-Textkonsole (die in einem Windows XP-Fenster dargestellt wird), Netzwerk, Festplatten (als normale Dateien unter Windows) und eine Tastatur. Bei der Auswahl der virtuellen Hardware wurde bewusst eine minimal mögliche Konfiguration gewählt, um Treiberprobleme im Linux zu vermeiden.

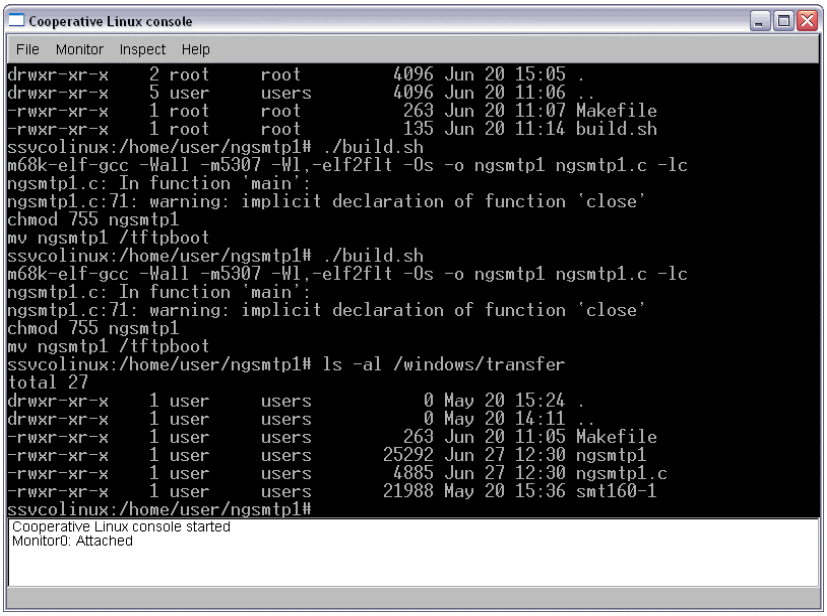


Abbildung 3: Die coLinux-VGA-Textkonsole im Windows XP-Fenster

Die Abbildung 3 zeigt die Oberfläche des auf einem Debian-Linux aufsetzenden SSV-coLinux unter Windows XP. Um diese Konfiguration zu starten, wird auf dem Windows XP-Desktop mit der Maus einfach auf den zuvor beschriebenen Icon geklickt. Wenige Sekunden später steht ein vollständiges Debian-Linux in einem Windows XP-Fenster zur Verfügung. Die Anmeldung ist über den **Benutzernamen *root*** mit dem **Passwort *root*** möglich.

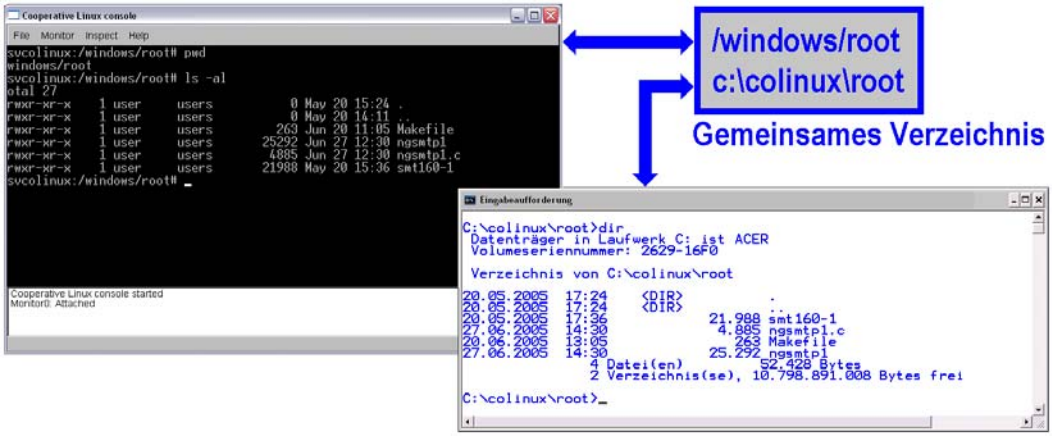


Abbildung 4: Windows und coLinux benutzen ein gemeinsames Verzeichnis

Windows und coLinux benutzen ein gemeinsames Verzeichnis auf der Festplatte (*c:\colinux* aus der Windows-Umgebung – unter coLinux ist das gleiche Verzeichnis über dem Namen */windows* ansprechbar). Sämtliche Dateien in diesem Verzeichnis sind sowohl von Windows als auch von coLinux aus zugänglich (Abbildung 4). Die gesamte Festplatte wird allerdings ausschließlich durch Windows verwaltet. Das coLinux greift nicht direkt auf die Festplatten-Hardware zu. Es ist somit keine eigene coLinux-Partition erforderlich. Der gesamte coLinux-Zugriff auf die Windows-Festplatte erfolgt über einen speziellen Treiber (coFS = coLinux File System).

Im Zusammenhang mit dem coFS ist allerdings zu beachten, dass es sich hierbei um ein asynchrones Dateisystem handelt. Die Schreibvorgänge unter coLinux bzw. Windows innerhalb des gemeinsamen Verzeichnisses werden aus Sicht des jeweils anderen Betriebssystems erst durch den nächsten Kontextwechsel (Übergabe der CPU von coLinux an Windows und umgekehrt) abgeschlossen. Aus diesem Grund sollte unter coLinux kein GCC-Makefile auf */windows* direkt zugreifen, wenn durch den Makefile Dateien benutzt werden, die zuvor unter Windows geschrieben wurden.

4. Unter coLinux per GCC übersetzen

Ein coLinux für die Embedded-Linux-Entwicklung enthält in der Regel die Original-GNU-Werkzeuge zum Übersetzen der C/C++-Programme sowie die entsprechenden Bibliotheken [3]. Im SSV-coLinux findet man – neben dem Native-GCC für x86-Code – die jeweiligen Cross-Varianten M68K-ELF-GCC (M68K-Code, Ausgabeformat ELF) und ARM-ELF-GCC (Code für verschiedene ARM-Architekturen im ELF-Ausgabeformat). Der Native-GCC kann zum Bei-

spiel für die Programmierung eines ADNP/1520 mit einem Embedded Gateway Linux benutzt werden. Zur Softwareentwicklung für die ColdFire-DIL/NetPCs unter uClinux (DNP/5280, DNP/5282 und PNP/5280) eignet sich der M68KELF-GCC. Der Linux Control DIL/NetPC DNP/7520 wird durch den ARM-ELF-GCC unterstützt. Weitere Cross-GCC-Varianten können bei Bedarf nachträglich in das SSV-coLinux eingefügt werden.

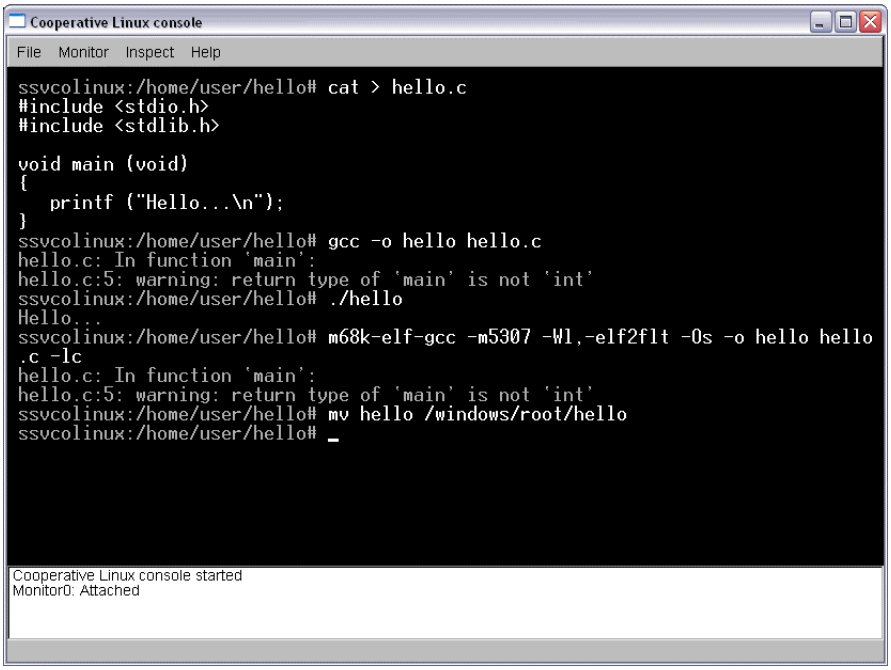


Abbildung 5: Erste Schritte mit den Compilern im SSV-coLinux

Für einen ersten Test der verschiedenen GNU-Werkzeuge einer SSV-coLinux-Installation reicht es aus, im coLinux-Fenster ein C-Programm zu erzeugen und zu übersetzen. Diese Schritte sind zum Beispiel durch die folgenden Eingaben möglich:

```
cat > hello.c
#include <stdio.h>
#include <stdlib.h>

void main (void)
{
   printf ("Hello...\n");
}
```
CTRL-D (beendet die Eingabe des C-Quelltextes)
```
gcc -o hello hello.c
```

Durch diese Eingabensequenz entsteht zunächst eine C-Quellcodedatei mit dem Namen *hello.c*. Diese Datei wird dann mit dem Native-GCC übersetzt. Das dabei entstehende Programm kann direkt im coLinux-Fenster ausgeführt werden. Die Abbildung 5 illustriert dieses Beispiel. In dieser Abbildung wird danach der M68K-ELF-GCC aufgerufen, um aus *hello.c* ein ausführbares Programm für einen ColdFire-DIL/NetPC zu erzeugen. Die unter uClinux ausführbare Datei

hello wird dann in das Verzeichnis */windows/root* verschoben. Von dort aus ist zum Beispiel ein TFTP-Dateitransfer zum DNP/5280 möglich.

5. Weitere Hilfswerkzeuge

Auf der SSV coLinux Cross Tools CD-ROM findet man einige weitere Werkzeuge für die DIL/NetPC-Programmierung. Besonders wichtig ist dabei der Open-Source-Editor PN (Programmers Notepad). Das Installationsprogramm für dieses Hilfswerkzeug ist im CD-ROM-Verzeichnis *ProgrammersNotepad* gespeichert.

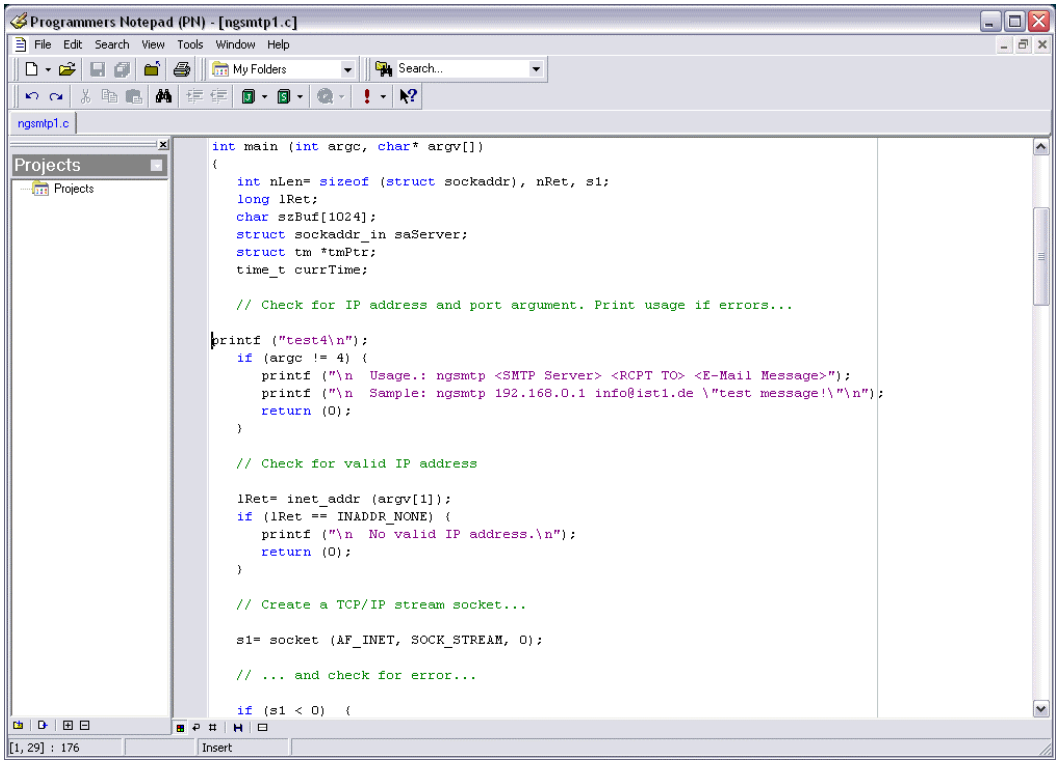


Abbildung 6: Arbeiten mit dem Open-Source-Editor PN (Programmers Notepad)

PN ist dafür vorgesehen, um aus der vertrauten Windows XP-Umgebung heraus die C/C++-Quellcodes für einen Embedded-Linux-Modul zu editieren. Die dabei entstehenden Quellcodedateien können im Windows-Verzeichnis *c:\colinux* abgespeichert werden. Sinnvollerweise sollten für die einzelnen Projekte innerhalb von *c:\colinux* weitere Unterverzeichnisse (wie zum Beispiel *c:\colinux\root* oder auch *c:\colinux\transfer*) erzeugt werden.

Ein weiteres wichtiges Hilfsmittel auf der SSV coLinux Cross Tools CD-ROM ist ein TFTP-Server mit dem Namen TFTPD. Er ist im CD-ROM-Verzeichnis *TFTP-Server-Win32* zu finden. Dieses Serverprogramm ist allerdings auch auf den Starterkit-CD-ROMs zu den entsprechenden DIL/NetPCs abgelegt. Der TFTPD dient zum TFTP-basierten Dateitransfer über eine Ethernet-LAN-Verbindung zwischen PC und Embedded-Linux-Modul. Die Abbildung 7 zeigt den TFTPD im Einsatz. Es ist zu beachten, dass dieses Programm für den Dateitransfer ein Aus-

gangsverzeichnis (Base Directory) benötigt. Dieses Verzeichnis wird über die *Browse*-Schaltfläche der TFTP-Benutzeroberfläche ausgewählt.

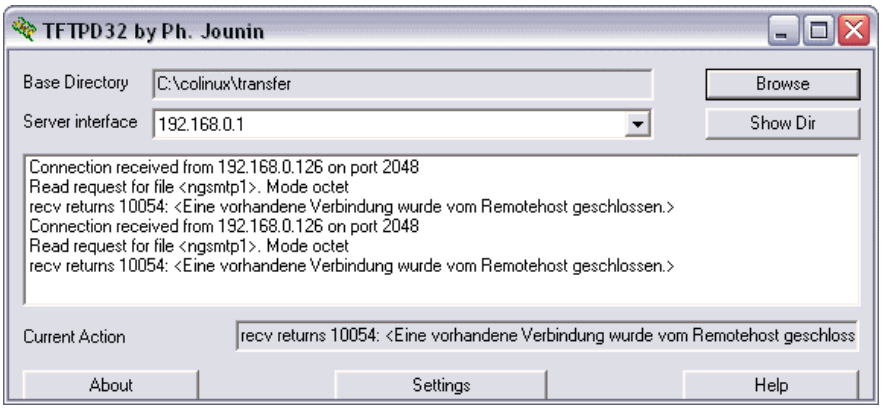


Abbildung 7: Der TFTP-Server dient für den Dateitransfer

Alle TFTP-Dateitransfers von und zum DIL/NetPC beziehen sich stets auf die Base Directory des TFTPD. Daher muss *c:\colinux* oder ein Unterverzeichnis innerhalb von *c:\colinux* für den Einsatz des TFTPD im Zusammenspiel mit coLinux als Ausgangsverzeichnis ausgewählt werden.

6. Das Zusammenspiel der Werkzeuge

Das coLinux-Fenster selbst, der Open-Source-Editor PN und der TFTP-Server TFTPD können zusammen so zum Einsatz gebracht werden, dass der Entwickler die vertraute Windows XP-Umgebung des Entwicklungs-PCs praktisch nicht mehr verlassen muss. Abbildung 8 liefert einen Überblick zum Zusammenspiel der Entwicklungswerkzeuge, die in der SSV-coLinux-basierten Entwicklungsumgebung für die DIL/NetPCs zum Einsatz kommen.

Bearbeitet werden die C/C++-Quellcodes in einem Windows-Fenster mit Hilfe von PN. Dieses Windows-Programm organisiert die Quelldateien in Projekte. Gespeichert werden alle Quellcodes im Verzeichnis *c:\colinux*, bzw. in einem Unterverzeichnis innerhalb von *c:\colinux*. Zum Übersetzen des gesamten Projekts wird im coLinux-Fenster lediglich eine Skript-Datei (Build Script) gestartet. Diese holt die entsprechenden Quellcodedateien aus dem gemeinsamen Verzeichnis, aktiviert über eine Make-Datei den GCC und schreibt nach einem erfolgreichen GCC-Lauf die ausführbaren Dateien für das Embedded-Linux-System wieder zurück in das gemeinsame Verzeichnis. Hier ein Beispiel für ein solches Build Script:

```
#!/bin/sh
cp /windows/root/ngsmtp1.c ngsmtp1.c
make all
mv /tftpboot/ngsmtp1 /windows/root/ngsmtp1
mv ngsmtp1.c ngsmtp1.bak
rm ngsmtp1.gdb
```

Die zweite Zeile des Build Scripts kopiert einen C-Quellcode mit dem Namen *ngsmtp1.c* von *c:\colinux\root (/windows/root* aus der coLinux-Sicht) in das coLinux-Verzeichnis, innerhalb welchem das Build Script gestartet wurde. Die dritte Zeile aktiviert den GCC-Make. Durch den GCC-Make-Lauf wird ein ausführbares Programm im coLinux-Verzeichnis */tftpboot* erzeugt. Dieses Programm wird über die vierte Zeile in das gemeinsame Verzeichnis *c:\colinux\root (/windows/root* aus der coLinux-Sicht) verschoben. Von dort aus kann es anschließend per TFTP zum DIL/NetPC übertragen werden. Die Ausführung eines solchen Build Scripts erfolgt jeweils aus dem coLinux-Fenster.

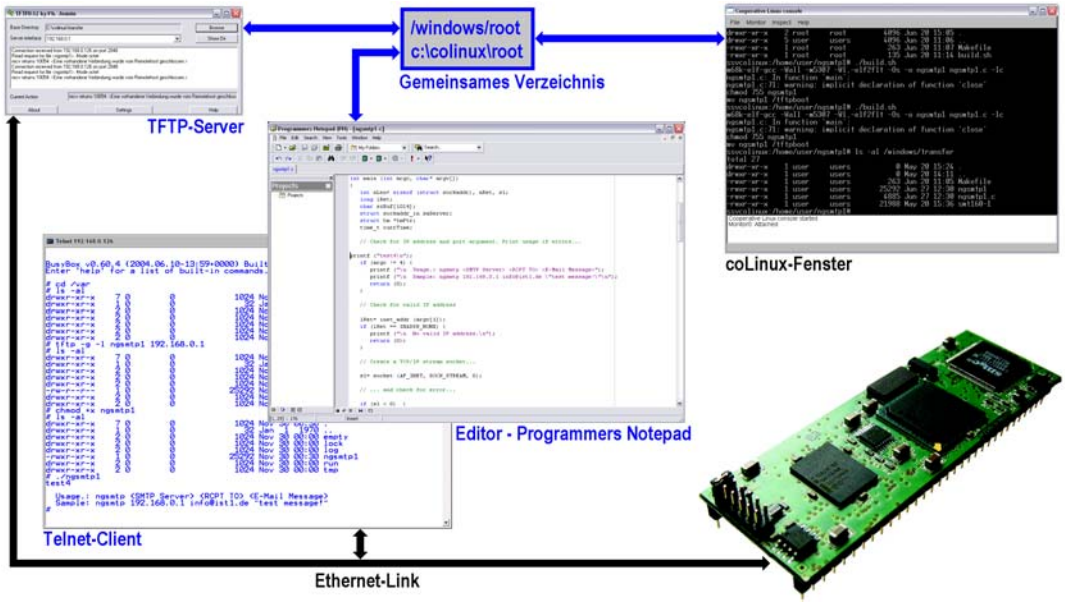


Abbildung 8: Das Zusammenspiel der Entwicklungswerkzeuge

Das Aktivieren der Skript-Datei ist die einzige Aktion, die der Benutzer im coLinux-Fenster ausführen muss. Somit entfällt praktisch jegliche Einarbeitung in Linux als PC-Betriebssystem. Für den Dateitransfer zum Embedded-Linux-System und den Shell-Zugriff kommen in der Abbildung 8 der Windows-TFTP-Server TFTPD und der Standard-XP-Telnet-Client zum Einsatz.

Abbildungen

Quellenangaben

[1] Web-Seiten zu den DIL/NetPCs: www.dilnetpc.com.
[2] Web-Seiten zu coLinux: http://www.colinux.org/
[3] Walter: Messen, Steuern, Regeln mit ARM-Mikrocontrollern. Franzis´ 2004.

Kontakt

SSV Embedded Systems
Heisterbergallee 72
D-30453 Hannover
Tel. +49-(0)511-40000-0
Fax. +49-(0)511-40000-40
E-Mail: sales@ist1.de
Web: www.ssv-embedded.de
Web: www.dilnetpc.com

Hinweise zu diesem Dokument (coLinux-APN1.Doc)

Revision	Date		Name
1.00	01.07.2005	Erste Version erstellt	KDW
1.01	05.07.2005	Einige Schreibfehler beseitigt.	KDW