1.Übung

1.Aufgabe:

3.

TFTP = Trivial File Transfer Protocol:

TFTP ist ein sehr einfaches Dateiübertragungsprotokoll. Es unterstützt lediglich das

Lesen und Schreiben von Dateien. Nicht vorhanden sind viele Funktionen des mächtigeren

FTP wie etwa Rechtevergabe mittels chmod.

DHCP = Dynamic Host Configuration Protocol:

DHCP ist ein Kommunikationsprotokoll, um die Zuweisung der Netzwerkkonfiguration an

Clients durch einen Server zu ermöglichen.

NFS = Network File System:

Das NFS ist ein von Sun Microsystems entwickeltes Protokoll, das den Zugriff auf Dateien über ein Netzwerk ermöglicht.

Dabei werden die Dateien nicht wie z. B. bei FTP übertragen, sondern die Benutzer können auf Dateien, die sich auf einem entfernten Rechner befinden, so zugreifen, als ob sie auf ihrer lokalen Festplatte abgespeichert wären.

Abfolge der Interaktionen zwischen dem Board und den Servern:

Der Desktop-PC (Server) führt die Netzwerkkonfiguration mittels DHCP aus und stellt die Funktionen des TFTP und NFS für den Client (das Board) zur Verfügung.

Das Board verbindet sich mit den Servern, lädt das Linux-Kernel-Image vom PC und Daten können ausgetauscht werden.

5.

Das Programm UBoot ist für die ersten Ausgaben über die serielle Schnittstelle verantwortlich. Im weiteren Verlauf wird das Image des Linux-Kernels geladen und gestartet. Das Board verbindet sich mit dem NFS des Desktop-PCs und der Scheduler wird registriert. Danach warden 3 Partitionen für den Collibri-Flash angelegt (Bootloader, Kernel, Filesystem) und Collibri wird gestartet.

CPU-Informationen:

root@colibri:~# cat /proc/cpuinfo
Processor : XScale-PXA270 rev 7 (v5l)
BogoMIPS : 311.29
Features : swp half thumb fastmult edsp
CPU implementer : 0x69
CPU architecture: 5TE

CPU variant : 0x0
CPU part : 0x411
CPU revision : 7
Cache type : undefined 5
Cache clean : undefined 5
Cache lockdown : undefined 5
Cache format : Harvard
I size : 32768
I assoc : 32
I line length : 32
I sets : 32
D size : 32768
D assoc : 32
D line length : 32
D sets : 32

Hardware : Toradex Colibri Module
Revision : 0000
Serial : 00000000002d1400

2.Aufgabe:

1.

Versionen:

gcc (Debian 4.4.5-8) 4.4.5

arm-linux-gcc (GCC) 3.3.2

2.

GNU ld version 2.14.90.0.7 20031029

GNU assembler 2.14.90.0.7 20031029

GNU gdb 6.7.1

3.

root@debian:/media/USB Basti/EOS# file HelloWorldGCC

HelloWorldGCC: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.18, not stripped

root@debian:/media/USB Basti/EOS# file HelloWorldARM

HelloWorldARM: ELF 32-bit LSB executable, ARM, version 1, dynamically linked (uses shared libs), for GNU/Linux 2.0.0, not stripped

ELF bedeutet Executable and Linkable Format. Es handelt sich dabei um ein Standardformat für ausführbare Dateien. Es ist das Standardbinärdateiformat für Unix-Systeme.

Dynamically linked bedeutet, dass Abhängigkeiten wie z.B. DLLs zur Laufzeit geladen werden. Im Vergleich dazu werden beim statischen Linken die Abhängigkeiten direkt in die ausführbare Datei kompiliert.

Not stripped bedeutet, dass die Datei Debug- und anderweitige Informationen enthält. Mit strip können die zusätzlichen Informationen entfernt werden.

4.

Diese Option bedeutet, dass das Programm nicht gelinkt wird, sondern nur kompiliert. Es wird dabei eine Objektdatei erzeugt.

root@debian:/media/USB Basti/EOS/Uebung1/HelloWorld# file HelloWorld.o

HelloWorld.o: ELF 32-bit LSB relocatable, ARM, version 1, not stripped

5.

Sie werden mit der /etc/exports Datei konfiguriert.

Inhalt: /root/pxadev/rootfs 192.168.1.0/255.255.255.0(rw,no_root_squash,no_subtree_check)

3.Aufgabe

Zeitmessung PC:

root@debian:/tmp/RandomNumbers# time ./RandomNumbers

real 0m0.129s

user 0m0.128s

sys 0m0.000s

Zeitmessung ARM:

root@colibri:/var/tmp# time ./RandomNumbersARM

real 0m 6.61s

user 0m 4.49s

sys 0m 0.10s

RandomNumbers.c:

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

int randLimited(int limit)
{
	return rand() % limit;
}

int main()
{
	int i = 0;
	int numOfRandomNumbers = 1000000;
	int limit = 1000;

	FILE* pFile;
	pFile = fopen("/tmp/randomNumbers.txt","w");

	for (i=0; i<numOfRandomNumbers; i++)
	{
	if (pFile!=NULL)
	{
	fprintf(pFile, "%d\n", randLimited(limit));
	}
	}
	fclose(pFile);

	return 0;
}
```