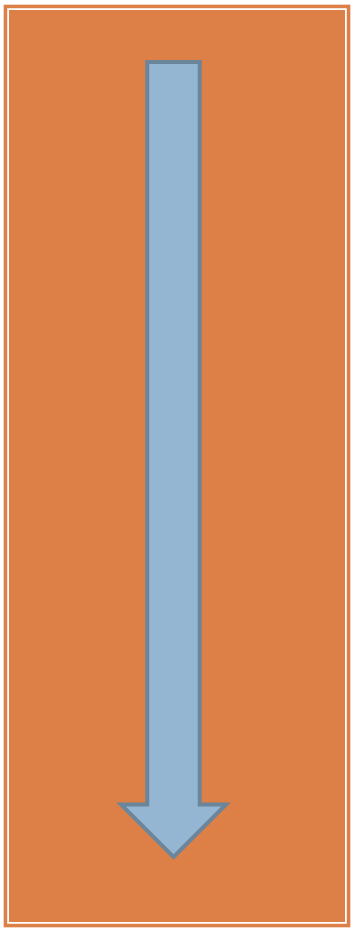
VORSTELLUNG DER DIPLOMARBEIT

14.11.2007

Thomas Werner

Inhaltsverzeichnis



Thema

„Untersuchung von Funktionsabläufen für das Remote-Thread-Debugging in eingebetteten Systemen“

Betreuer: Dipl.-Inform. S. Köhler

Prof. Dr.-Ing.habil. R. G. Spallek

Aufgabenstellung

- Literaturrecherche
- Analyse von repräsentativen Remote-Debug-Szenarien
- Konzeption und Implementierung einer Multi-Thread Erweiterung der Debug-Agentensoftware für eingebettete FTZeTW-ARM-EVAL
- Erweiterung der Target-Interface-Komponente für die UDE von PLS

Anwendungsdebugging

Ziel

- Entwicklung fehlerfreier Anwendungen für eingebettete Systeme
- Erstellung auf komfortableren Host-System
- Ausführung und Test auf gewähltem Targetsystem
- Fehlersuche auch bei gleichzeitiger Ausführung mehrerer Threads

Anwendungsdebugging

„Embedded System“

- Eingebunden in techn. Kontext
- Zweckbestimmte Funktion
- Prozessor, Speicher, Peripherie
- Oft mit eigenem Betriebssystem
- Bsp: eTW-ARM-Eval board (FTZ)
  - ARM9 Prozessor, MMU
  - Serielle und Netzwerkschnittstelle
  - Linux, angepasster Kernel 2.6.18

Anwendungsdebugging

Szenario:

- Entwicklung einer Linux-Anwendung in C für eingebettetes System
- UDE als Debugumgebung
- Target-Host-Verbindung über LAN
- GDB-Server auf Targetsystem
- Kommunikation über „Remote Serial Protocol“
- ⇒Remote-Debugging

Anwendungsdebugging

Problem

- Multithreading-Anwendungen bisher nicht unterstützt

Lösung

- Ansätze für Thread-Debugging bei GDB-Server und Remote Serial Protocol
- Erweiterung und Hinzufügen von benötigten Funktionen

Threads

Definition

- Faden, Ausführungsstrang
- Teil eines Prozesses, gemeinsame Nutzung der Betriebsmittel eines Prozesses durch dessen Threads
- Gemeinsames Code- und Daten-segment, eigener Stack und PC
- Gleicher Adressraum
- Bearbeitung von Teilaufgaben

Threads - Abgrenzung

Prozesse

- Ablauf eines Programms auf einem Prozessor
- Speicherraum und Betriebsmittel
- Mindestens ein Thread

Threads

- Teilung von Speicher und Betriebsmittel
- Geringerer Aufwand bei Threadwechsel
- Abarbeitung neben-läufiger Stränge

Threads

Kernel-thread

Userthread

Multi-threading

- Ausführungsstrang aus Sicht des Betriebssystems
- Aufteilung Kernelthreads in einzelne Stränge (Faden in Fasern) durch Anwendersoftware
- Gleichzeitige Abarbeitung mehrerer Threads

Threads - Konflikte

Kritischer Abschnitt

- Programmabschnitt mit Zugriff auf Ressourcen, die auch von parallel laufenden Threads genutzt werden
- Inkonsistenz $\Rightarrow$ exklusiver Zugriff
- Mutex
- Sequentielle Abarbeitung dieses Abschnittes

Threads - Zustände

initialized

ready

running

waiting

terminated

„zombie“

- Noch nicht gestartet
- Gestoppt, anderer Thread rechnet
- Wird gerade abgearbeitet
- Blockiert, wartet auf Ereignis
- Abarbeitung beendet
- Erzeugerthread beendet vor Erreichen des Status „terminated“

Thread-Debugging

Behandlung der Threads durch den Debugger

- Thread-Kontext als Sicht auf das System betrachten
- Visualisierung der Thread-Kontexte
- Überwachung der Synchronisation
- Wechsel des aktiven Threads
- Starten / Anhalten ausgewählter oder aller Threads

Threads unter Linux

NPTL

- Native POSIX Thread Library
- Moderne Implementierung einer Threading-Bibliothek
- POSIX-konform, unterstützt SMP
- Kompatibel zu „LinuxThreads“
- Geringer Aufwand bei Thread-erzeugung
- Eingebunden durch pthread-Bibliothek (pthread.h)

Threads unter Linux

NPTL

- Kernel verwaltet weiterhin Prozesse statt Threads (clone())
- 1:1 Abbildung auf Prozesse in Scheduling-Queue
- Kein Manager-Thread
- Kernel verantwortlich für Signale, Speicherfreigabe, Terminierung
- Synchronisation über Futex

Threads unter Linux

Nutzung der pthread-Bibliothek

- include <pthread.h>
- pthread_create()
- pthread_join()
- pthread_exit()
- pthread_mutex_lock()
- pthread_mutex_unlock()

Remote-Debugging

GDB-Server

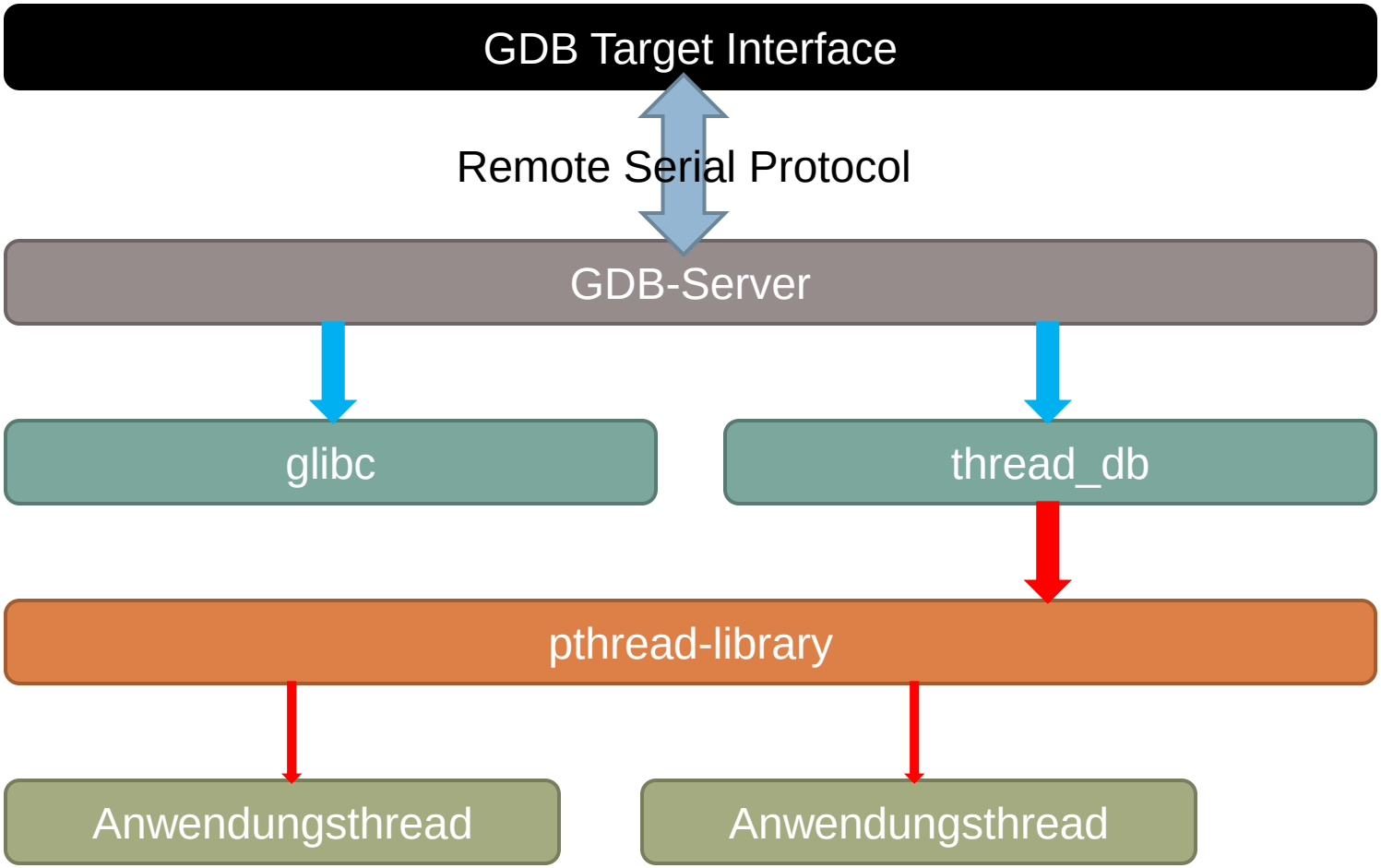
- Angepasste Version des GDB-Server aus dem GNU Projekt
- Kompilierung für das Target-System mit Toolchain
- Bereitstellung wichtiger Funktionen
- Kommunikation über Remote Serial Protocol
- $\Rightarrow$ steuerbar durch GDB oder UDE

Remote-Debugging

Remote Serial Protocol

- Paketorientiert
- Zeichenbasiert
- $ *Paketdaten* # *Checksumme*
- Übertragung aller Steuerbefehle
- z.B.: Starten, Anhalten, Fortfahren, Breakpoints, Register lesen, Speicherzugriffe, Thread-Informationen

Remote-Debugging



Implementierung - Status

Bereits in GDB-Server und Target-Interface integriert

- Alle Basisfunktionen zum Steuern des Programmablaufs
- Verwaltung des Kontext (Register lesen und anzeigen)
- qC – aktiver Thread
- qfThreadInfo – Thread-Liste

Implementierung - Status

Noch zu implemen-tieren

- qfThreadExtraInfo
- qP – thread info request

- Synchronisierungsfunktionen
- Abfrage des Thread-Status

Ausblick

- Fertigstellen aller nötigen Erweiterungen in GDB-Server und Targetinterface
- Entwicklung einer Komponente zur Visualisierung der vorhandenen Threads und deren Kontexte
- Visualisierung der Synchronisierung und Möglichkeit des manuellen Eingriffs

Quellen

- Modern Operating Systems, Andrew S. Tanenbaum
- www.wikipedia.de
- www.gnu.org
- developer.apple.com
- www.ibm.com