

**FH Aachen**

**Fachbereich Elektrotechnik und Informationstechnik**

Embedded Systems SS2022

Referat

**Linux Tracing**

**Marcel Ochsendorf, Muhammed Parlak**

# Inhalt

<b>1</b>	<b>Einleitung</b>	<b>1</b>
1.1	Tracing . . . . .	1
1.2	Ursprung . . . . .	1
<b>2</b>	<b>Grundlagen</b>	<b>2</b>
2.1	Events . . . . .	2
2.2	Kprobes . . . . .	2
2.2.1	CPU-Traps . . . . .	2
<b>3</b>	<b>Tracing auf Mikrokontrollern</b>	<b>3</b>
<b>4</b>	<b>Tools</b>	<b>4</b>
4.1	Trace-Log Aufzeichnung . . . . .	4
4.1.1	ftrace . . . . .	4
4.2	Visualisierung . . . . .	4
4.2.1	Kernelshark . . . . .	4
<b>5</b>	<b>Interpretation des Kernel-Trace Ergebnisses</b>	<b>6</b>
<b>6</b>	<b>Beispiel der Identifikation von Laufzeitproblemen</b>	<b>7</b>
6.1	Ausgangsszenario . . . . .	7
6.2	Aufzeichnung mittels ftrace . . . . .	7
6.3	Visualisierung und Beurteilung des Trace-Logs mittels kernelshark . . . .	7
	<b>Literaturverzeichnis</b>	<b>8</b>
	<b>Abbildungsverzeichnis</b>	<b>9</b>
	<b>Tabellenverzeichnis</b>	<b>10</b>

# 1 Einleitung

Tracing ist die spezielle Verwendung der Protokollierung zur Aufzeichnung von Informationen über den Ausführungsablauf eines Programms. Oft werden mit eigenständig hinzugefügte Print-Messages der Code debuggt. Somit verfolgt man die Anweisungen mit einem eigenem tracing-System. Linux bringt einige eigenständige Tools mit, mit denen es möglich ist Vorgänge innerhalb von einem Embedded-System nachvollziehen und analysieren zu können. Die Linux-Tracing Funktionalität und die bestehenden Tools, welche im Linux-Kernel integriert sind, helfen so dabei bei der Identifikation von Laufzeiten, Nebenläufigkeiten und der Untersuchung von Latenzproblemen,

## 1.1 Tracing

Nachfolgend wird erleutert und an einem Beispiel demonstriert, wie das Linux-Tracing bei der Identifikation von Laufzeitproblemen eingesetzt werden kann.

## 1.2 Ursprung

## **2 Grundlagen**

### **2.1 Events**

### **2.2 Kprobes**

Kprobes können dazu verwendet werden, Laufzeit und Performance-Daten des Kernels zu sammeln. Der Vorteil and diesen ist, dass diese Daten ohne Unterbrechnung der Ausführung auf CPU-Instruktions-Ebene aggregiert werden können, anders wie bei dem Debuggen eines Programms mittels Breakpoints.

#### **2.2.1 CPU-Traps**

## **3 Tracing auf Mikrokontrollern**

# **4 Tools**

Allgemein sind keine speziellen Programme notwendig um die Laufzeiteigenschaften eines Programms aufzuzeichnen. Der Linux-Kernel bringt bereits alle nötigen Funktionalitäten mit. Jedoch gibt es Tools die eine visuelle Darstellung der aufgezeichneten Events ermöglichen.

## **4.1 Trace-Log Aufzeichnung**

### **4.1.1 ftrace**

## **4.2 Visualisierung**

### **4.2.1 Kernelshark**

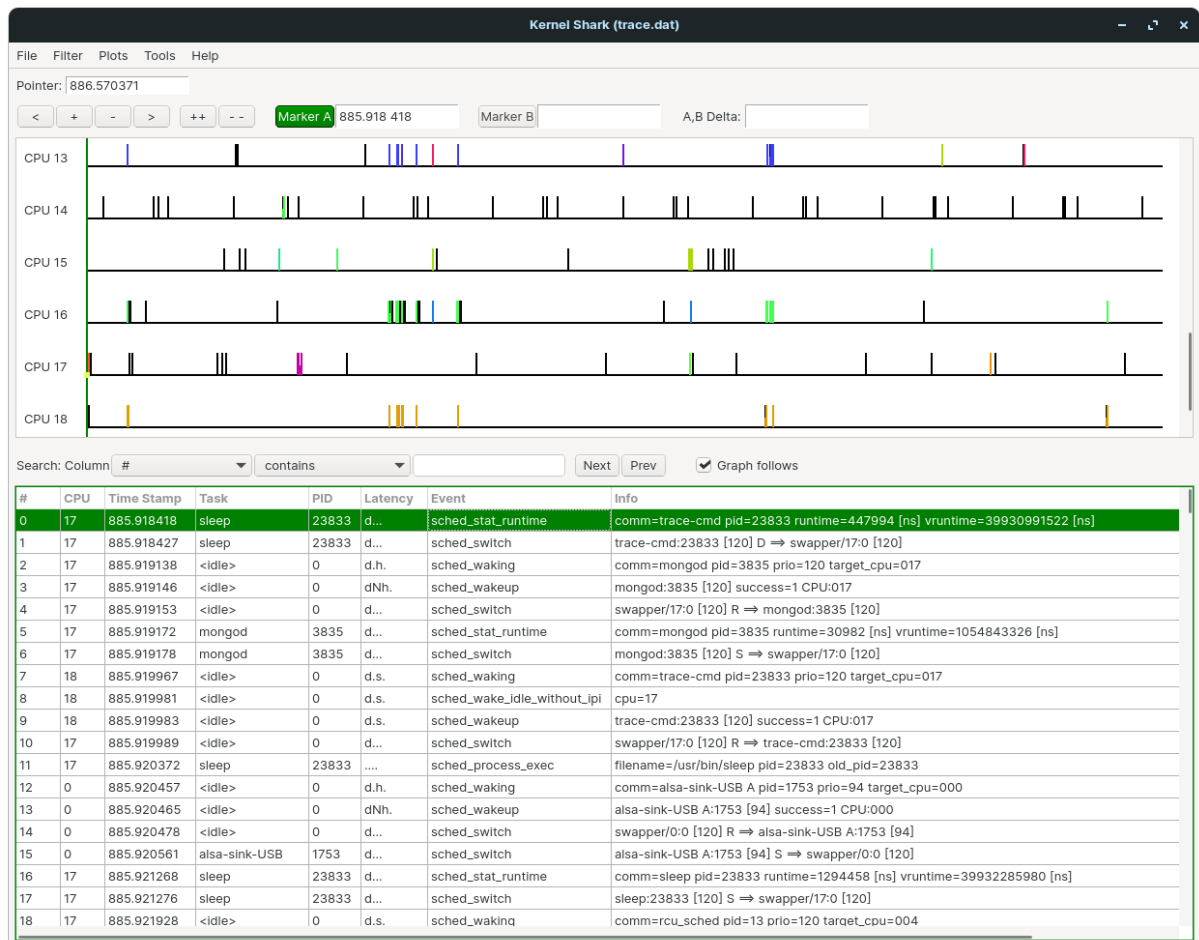


Bild 4-1: Kernelshark

## **5 Interpretation des Kernel-Trace Ergebnisses**

## **6 Beispiel der Identifikation von Laufzeitproblemen**

### **6.1 Ausgangsszenario**

Als Ausgangspunkt dieses Beispiels, soll das Laufzeitverhalten eines Programms auf einem Linux-System analysiert werden. Die zugrunde liegende Software wurde bisher nur auf einem Linux-Realtime Kernel verwendet, jedoch erfordert die Implementation neuer Features eine neuere Kernel-Version, welche noch nicht als RT-Version auf dem System zur Verfügung steht. Somit soll ermittelt werden, ob die unmodifizierte Software eins zu eins auf dem neuen System lauffähig ist und die Laufzeitandorderungen erfüllt.

### **6.2 Aufzeichnung mittels ftrace**

### **6.3 Visualisierung und Beurteilung des Trace-Logs mittels kernelshark**

# Literaturverzeichnis

# Abbildungsverzeichnis

4-1	Kernelshark	5
-----	-------------	---

# **Tabellenverzeichnis**