
# Entwicklung eines Parsers zur Normalisierung und Aufarbeitung heterogener Build-Protokolle

**Projektarbeit IIb (T3_2000)**

Im Rahmen der Prüfung:
**Bachelor of Science (B. Sc.)**

des Studienganges Informatik
an der Dualen Hochschule Baden-Württemberg Karlsruhe

von
**Lukas Bailey**

| | |
|---|---|
| **Abgabedatum** | 15. September 2025 |
| **Bearbeitungszeitraum** | 30.06.2025 - 15.09.2025 |
| **Matrikelnummer, Kurs** | 8232296, TINF23B2 |
| **Ausbildungsfirma** | SAP SE<br>Dietmar-Hopp-Allee 16<br>69190 Walldorf, Deutschland |
| **Betreuer der Ausbildungsfirma** | Philipp Degler |
| **Gutachter der Dualen Hochschule** | Prof. Dr. Sebastian Ritterbusch |

# Eidesstattliche Erklärung

Ich versichere hiermit, dass ich meine Projektarbeit IIb (T3_2000) mit dem Thema:

> *Entwicklung eines Parsers zur Normalisierung und Aufarbeitung heterogener Build-Protokolle*

gemäß § 5 der „Studien- und Prüfungsordnung DHBW Technik“ vom 29. September 2017 selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Karlsruhe, 25. Februar 2026

gez.: Bailey, Lukas

# Sperrvermerk

Die nachfolgende Arbeit enthält vertrauliche Daten der:

SAP SE
Dietmar-Hopp-Allee 16
69190 Walldorf
Deutschland

Der Inhalt dieser Arbeit darf weder als Ganzes noch in Auszügen Personen außerhalb des Prüfungsprozesses und des Evaluationsverfahrens zugänglich gemacht werden, sofern keine anderslautende Genehmigung vom Dualen Partner vorliegt.

## Abstract

*- Deutsch -*

In dieser Arbeit wird ein Parser zur Normalisierung der Unterschiede in von unterschiedlichen Werkzeugen generierten Build-Protokollen vorgestellt. Dieser Parser, `log-jitsu` genannt, liest wichtige Diagnostikausgaben von unterschiedlichen Build-Prozessen unabhängig davon ein, welche Build-Systeme und Compiler verwendet wurden, und verarbeitet sie in ein Format, das wesentliche Informationen kontextuell darstellt. Um diese Anforderungen zu erfüllen, implementiert `log-jitsu` in Rust einen mehrstufigen Prozess: Zunächst wird das Protokoll eingelesen und identifiziert, welches Build-System für den Bauprozess verantwortlich war. Daraufhin werden die Eigenheiten dieses Systems normalisiert; es entstehen Zeilen, die eindeutig einem Projekt und dem Werkzeug, das sie geschrieben hat, zugeordnet werden können. Jede dieser Zeilen wird in einem Parse-Prozess verarbeitet, der unabhängig davon, wer sie produziert hat, Diagnostiken erkennen kann und alle ihre Komponenten in einem Datenformat zurückgibt, das ihren semantischen Zusammenhang wahrt. Dieses Datenformat ist ideal für die Verarbeitung und Darstellung in einer HTML-Ausgabe, die zusätzlich gezeigt wird. Es wird festgestellt, dass Rust und besonders der Parsergenerator Nom ideale Werkzeuge für diese Aufgabe sind und die Implementierung von `log-jitsu` eine gute Lösung für die Problemstellung leistet. Diese Arbeit erläutert alle nötigen Grundlagen, diskutiert die genaue Funktionsweise von `log-jitsu` und analysiert die Lösung auf einen möglichen Produktiveinsatz.

# Inhaltsverzeichnis

# Quellcodeverzeichnis

# 1 Einleitung

## 1.1 Motivation

Mit der quantifizierung von Musik durch moderne Sequenzer und Digital Audio Workstations (DAWs) ist die meiste populäre Musik exakt in mehrer Takte und Strukturen wie Intro, Refrain, Strophe, Bridge und Outro unterteilt. Durch das erkennen dieser Strukturen können unter einsatz einer einfachen Typ-3 Grammatik Remixe bestehender Songs generiert werden. Dabei können dann neue beliebig lange, situative reagierende Abmischnungen von Songs und Videos erzeugt werden.

## 1.2 Zielsetzung

Ziel dieser Arbeit ist es ein System zu erstellen, dass im ersten Schritt die Struktur sinvoll gewählter Pop-Musik Videos erkennen soll und aus diesen durch Zuordnung der Strukturelemente zu einer Typ-3 Grammatik für die menschliche Wahrnehmung stimmige Remixe zu generieren. Dabei soll eine beliebige länge angegeben werden können die ungefähr eingehalten wird. Zudem soll es möglich sein weitere nicht aus dem ursprünglichen Song stammende Elemente in den Remix einzubringen. Für die Videos sollen Abschnitte mit einem Crossfade oder anderen Übergängen ineinander greifen.

## 1.3 Aufbau der Arbeit

# Literaturverzeichnis

# Anhangsverzeichnis

# 1 Anhang: Verwendete Nom-Parserfunktionen

| Name | Beschreibung |
|---|---|
| char(a) | Akzeptiert das Zeichen `a` am Anfang der Kette |
| alt((parser1 , parser2, …)) | Testet mehrere Parserfunktionen, nur eine muss erfolgreich sein |
| many1(parser) | Wendet einen Parser so oft an, wie er erfolgreich ist; mindestens 1-mal |
| tag(A) | Akzeptiert die Zeichenkette `A` am Anfang der Kette |
| delimited(parser1, parser2, parser3) | Akzeptiert eine von parser2 akzeptierte Zeichenkette, die von zwei anderen akzeptierten Zeichenketten umgeben ist |
| separated_pair(parser1, trennzeichen, parser1) | Akzeptiert zwei von einer bestimmten Zeichenkette getrennte Zeichenketten |
| digit1 | Akzeptiert eine (mindestens ein Zeichen lange) Kette an Ziffern |
| multispace1 | Ignoriert (mindestens ein) Leerzeichen am Anfang der Kette |
| is_not(a) | Akzeptiert ein Zeichen, das nicht `a` ist |
| take_until(A) | Falls Nom die Zeichenkette `A` im Text findet, konsumiert es alles bis inklusive `A` |

## 2 Anhang: Testausgabe

```
Diagnostic {
  message: "'int readdir_r(DIR*, dirent*, dirent**)' is deprecated [-
Wdeprecated-declarations]",
  severity: Warning,
  location: SourceLocation {
    path: "/SAPDevelop/I587738/git/sapkernel/flat/nlsui3.c",
    row: Some(577),
    column: Some(61)
  },
  code: Some("   rc = readdir_r( dirp, &(c_entryBuffer.c_entry),
&c_result );\n
^\n"),
  trace: [
    In {
      context: "function 'int readdir_rU16(DIR*, direntU16*,
direntU16**)'",
      location: SourceLocation {
        path: "/SAPDevelop/I587738/git/sapkernel/flat/nlsui3.c",
        row: None,
        column: None
      }
    }
  ],
  notes: [
    Note {
      message: "declared here",
      location: Some(SourceLocation {
        path: "/sapmnt/appl_sw/sysroot-sles15/usr/include/dirent.h",
        row: Some(189),
        column: Some(12)
      }),
    code_snippet: Some(" extern int __REDIRECT (readdir_r,\n
^~~~~~~~~~\"\n"),
    trace: []
    }
  ]
}
```

Code 1: Ausgabe von log-jitsu nach Testeingabe

# 3 Anhang: Screenshots der generierten HTML-Datei

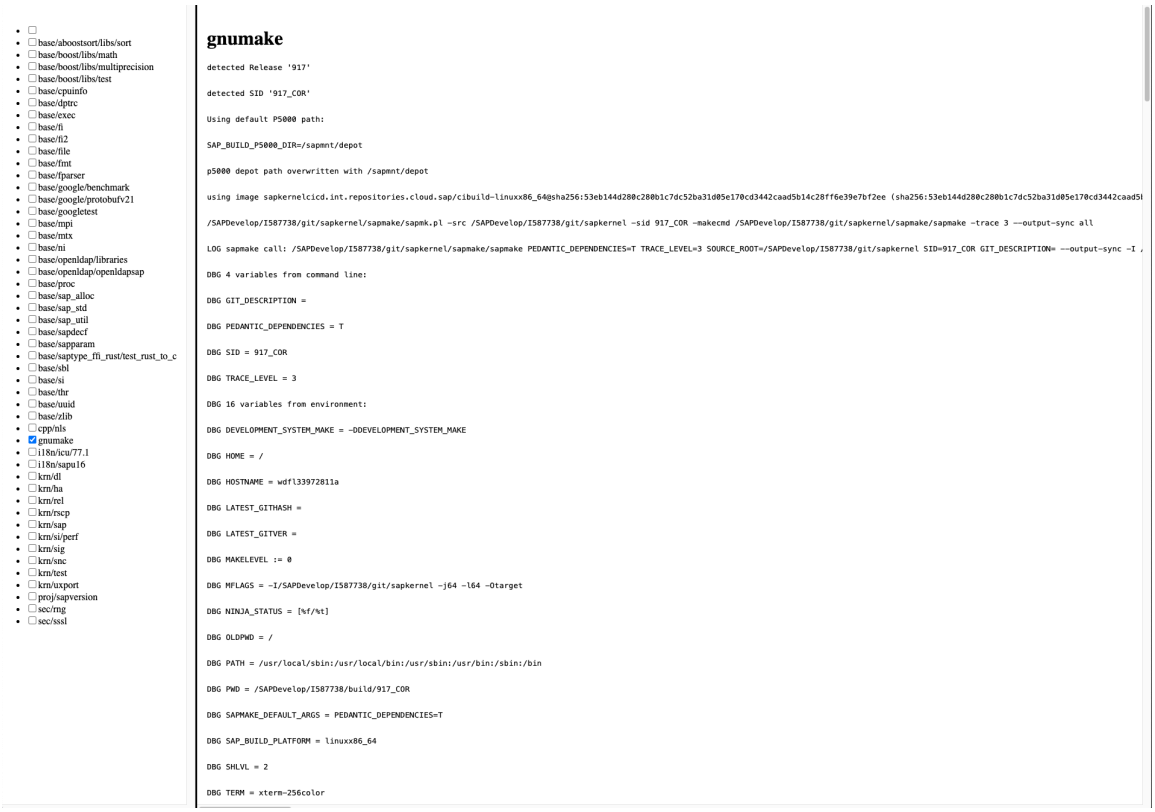


Abbildung 1: Vom Build-System generierter Output