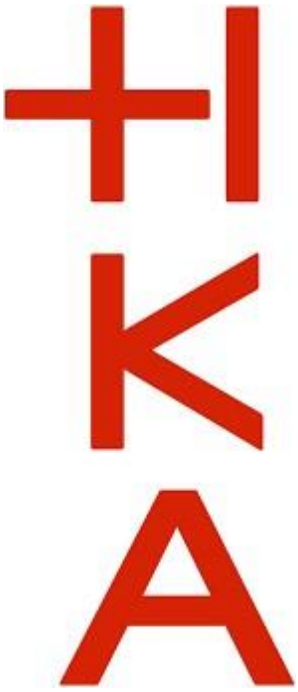
Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES



Projektarbeit

Threadsanitizer – Trace Generierung zwecks offline Analyse

Von: Martin Glauner(74770)

Dozent: Prof. Dr. Martin Sulzmann

Projektverlauf:

Während des Projekts gab, es mehrere Schwierigkeiten, die ich überwinden musste, um das Projekt erfolgreich zu kompilieren. Hier ist eine kurze Zusammenfassung dessen, was schiefgelaufen ist und wie ich die Probleme letztendlich lösen konnte.

Die meisten Fehler beim Kompilieren entstanden aufgrund von unzureichendem Arbeitsspeicher. Zunächst versuchte ich, das Projekt in einer Linux-VM auf meinem Laptop zu kompilieren. Nach mehreren Stunden erfolgloser Versuche wurde jedoch klar, dass der in der VM zugewiesene Arbeitsspeicher von 4 und 6 GB RAM nicht ausreichte.
Daraufhin entschloss ich mich, Linux direkt auf meinem Laptop zu installieren, was jedoch auch nicht zum gewünschten Ergebnis führte, da auch die 8 GB RAM meines Laptops nicht ausreichend waren, um das Projekt fehlerfrei zu bauen. Ein häufiger Fehler war hier der "Memory Exhaust". Leider traten derartige Fehler oft erst im späteren Verlauf des Kompilierprozesses auf, wodurch hier ein enormer Zeitaufwand entstanden ist.

Als Nächstes versuchte ich, das Projekt auf meinem Desktop-PC mit 32 GB RAM zu kompilieren, zunächst ebenfalls in einer VM, was jedoch auch zu Fehlern führte, die auf den Arbeitsspeicher zurückzuführen waren. Nach mehreren erfolglosen Versuchen beschloss ich, Linux direkt auf meinem Desktop-PC zu installieren.
Schließlich konnte ich das Problem mit den "Memory Exhaust"-Fehlermeldung beseitigen, indem ich alle 32 GB RAM und die 8 Kerne meines Prozessors nutzte. Allerdings stieß ich hier wieder auf hardwarebedingte Probleme, da mein Prozessor eine Vollauslastung über einen längeren Zeitraum nicht bewältigen konnte. Nachdem ich die Prozessorlast auf die Hälfte reduziert hatte, konnte ich das Projekt schließlich das erste Mal erfolgreich kompilieren.

Es gab im Verlauf des Projekts auch weitere Fehler und Probleme. Hier sind einige Beispiele:

1. Verwendung von C++ Standardbibliotheken im Projekt

Anfangs habe ich versucht, mit C++ Standardbibliotheken zu arbeiten, aber es stellte sich heraus, dass sich diese nicht so einfach einbinden ließen. Die Recherche und das Testing hier war auch hier sehr zeitintensiv. Letztlich haben wir für alles Projektrelevante nur C Bibliotheken genutzt

2. Schreiben in Files während der Laufzeit

Auch dies war bedauerlicherweise nicht möglich, durch die Instrumentierung können Operationen wie das öffnen, schließen, lesen oder schreiben in einer Datei dazu führen, dass die Ausführung des Programms mit der Ausführung der Laufzeitbibliothek vermischt wird. Dies führte bei uns zu Abstürzen. Verwenden konnten wir daher nur die eigens dafür eingebaute thread-sichere Print Funktion.

3. CMAKE Flags

CMake bereitete mir ebenfalls Probleme. Meine erste Einstellung zum Build-Typ wurde über den gesamten Prozess und mehrere Configuration-Resets hinweg als DEBUG gespeichert, leider bemerkte ich das erst relativ am Ende. Im Vergleich dauerte der Kompiliervorgang im Debug-Modus ca. 3-4 Stunden, während es im Release-Modus nur ca. 1 Stunde dauerte. Ein ähnlicher Unterschied war beim Rebuild zu beobachten (ca. 10 Minuten im Debug-Modus vs. 30 Sekunden im Release-Modus). Ich wurde schließlich durch einen Tipp auf

dieses Problem aufmerksam und konnte es nur durch das komplette Neu aufsetzen, von CMake beheben.

Wie habe ich Hilfe gefunden:

Hilfe habe ich in erster Linie durch Googeln der Fehlermeldungen erhalten. Stack overflow und Co. Waren mit Abstand am nützlichsten. Weiter konnte ich in der Google Mailing Gruppe und im llvm-board(https://discourse.llvm.org/) nützliche Informationen finden, auf fragen hier wurde jedoch leider nicht reagiert.

Trotz all der Schwierigkeiten, auf die ich während des Projekts gestoßen bin, war es letztendlich eine sehr lehrreiche Erfahrung. Durch die Fehler, die ich gemacht habe, habe ich viel über verschiedene Aspekte des Kompilierens, sowie die Verwendung von Tools wie CMake gelernt.
Während ich manchmal auch frustriert war, als ich auf Fehler stieß, erkannte ich bald, wie wichtig es ist, sich Zeit zu nehmen, um Probleme genauer zu analysieren, bevor wild drauflos probiert wird.

Alles weite Projektrelevante inklusive einer Anleitung ist im Readme des GitHub Repositorys zu finden.
https://github.com/martinglauner/llvm-project