1 Einleitung

Herzlich willkommen zu dieser Studie zur Analyse des Einflusses von Entwicklern auf den Energieverbrauch von Programmen. Im Folgenden werden Sie verschiedene Aufgaben bearbeiten und die von uns bereitgestellten Werkzeuge nutzen. Ihr Ziel ist es, den Energieverbrauch der vorgegebenen Programme zu optimieren. Sollten während der Bearbeitung Fragen oder Unklarheiten auftreten, zögern Sie bitte nicht, diese direkt an uns zu richten.

2 Arbeitsumgebung

Vor sich finden Sie einen Rechner, den wir extra für die Studie eingerichtet haben. Zusätzlich zu diesem Aufgabenblatt haben Sie einen Zettel mit Zugangsdaten zu ihrem Account erhalten. Loggen Sie sich bitte damit ein.
Nach dem Login öffnet sich automatisch der Dateimanager in dem Ordner `/Dokumente/study` Innerhalb

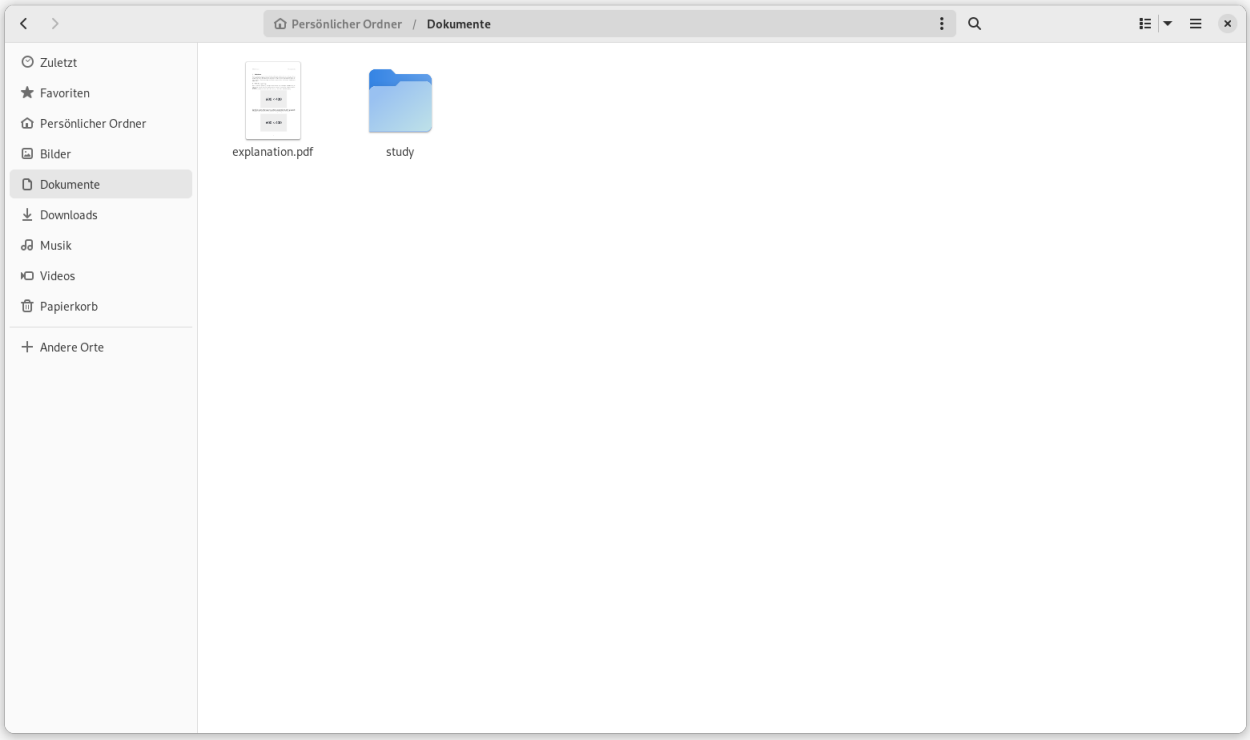


Abbildung 1: Der Ordner study enthält Ihre Arbeitsumgebung

dieses Ordners finden den Unterordner `tasks`, der die einzelnen Projekte enthält die Sie im Rahmen der Studie bearbeiten sollen . Zusätzlich dazu befinden sich in dem Ordner neben einer digitalen Kopie dieses Blatts weitere hilfreiche Dokumente, um die Bearbeitung der Aufgaben zu unterstützen.

In der Schnellzugriffsleiste haben wir bereits Verknüpfungen zu der verwendbaren Software installiert unter anderem auch Visual Studio Code um die Aufgaben bearbeiten zu können. Wir haben in Visual Studio Code einige Extensions installiert, um die Bearbeitung der Aufgaben sinnvoll zu ermöglichen.
Bitte nutzen Sie für die Bearbeitung der Aufgaben ausschliesslich Visual Studio Code!

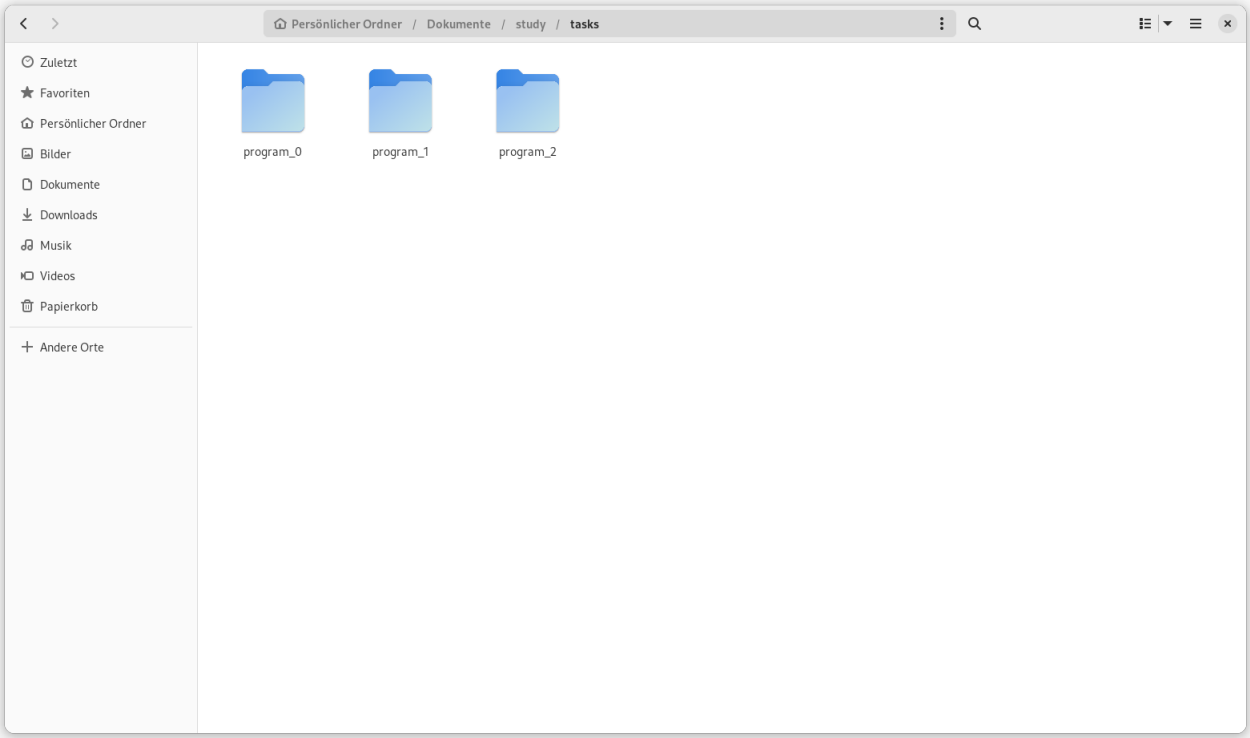


Abbildung 2: Der Ordner Tasks mit den einzelnen Programmen

3 Buildscript

Um die Bearbeitung der Aufgaben so einfach wie möglich zu gestalten, stellen wir Ihnen ein Shell-Script bereit, das die Kompilierung der Aufgaben übernimmt. In jedem Aufgaben-Unterordner finden Sie die Datei `builder.sh`. Diese erlaubt folgende Befehle:

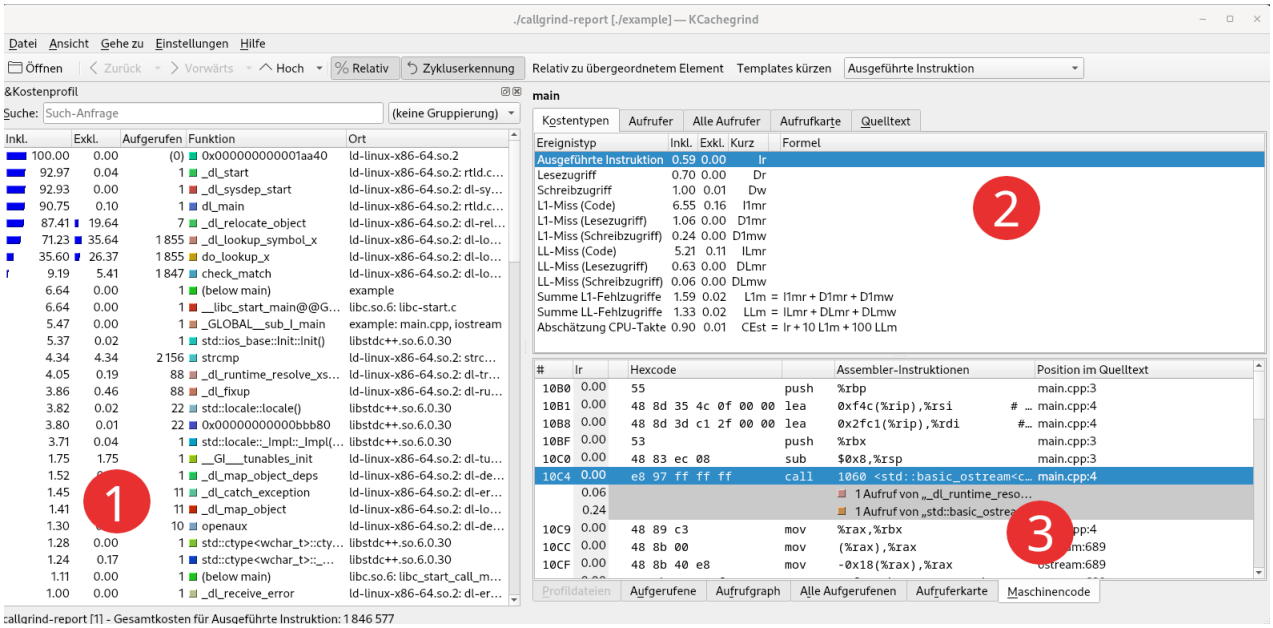
- `./builder.sh build`
  Kompiliert die Anwendung und erzeugt eine Ausführbare Anwendung im Ordner `target/`
- `./builder.sh profile`
  Profiled das Programm mithilfe von `kCacheGrind`. Benötigt einen erfolgreichen Build!
- `./builder.sh clean`
  Löscht das `target/` Verzeichnis, um einen Clean-Build, ohne gecachte Dateien zu ermöglichen

4 Profiler

Um die Bearbeitung der Aufgaben zu unterstützen, stellen wir Ihnen den Profiler `valgrind` bereit, der über das Tool `kCacheGrind` visualisiert wird. Diese Visualisierung kann für jede Aufgabe mithilfe des Build-Scripts gestartet werden und erlaubt es Ihnen, bestimmte Insights über Ihr Programm zu erhalten.

4.1 Beispiel

Im Folgenden sehen Sie einen Screenshot aus dem Profiler für ein Beispielprogramm:



Die diversen Programmdetails werden im Folgenden genauer erläutert:

1) Hier werden die Funktionen, die im Programm aufgerufen werden, aufgelistet. Neben Funktionen des Programms selber tauchen hier auch Funktionen auf, die durch das Betriebssystem bereitgestellt werden. Jede Funktion wird über zwei Parameter genauer beschrieben:

   - `Inkl.` Beschreibt den relativen Anteil an Kosten, der durch aufgerufene Funktionen erzeugt wird.
   - `Exkl.` Beschreibt den relativen Anteil an Kosten, der durch die Funktionen selbst erzeugt wird.

2) In diesem Teilfenster werden weitere Programmdetails nach dem `inkl.` und `exkl.` Modell beschrieben. Hier kann zum Beispiel nachvollzogen werden, ob die aufgerufenen Funktionen mehr Lesezugriffe durchführen als die Funktion selber. Auch kann hier zum Beispiel die Verteilung der geschätzten CPU-Takte eingesehen werden.
   Die Verteilungen für Aufrufer der ausgewählten Funktion können auch detailliert in einer Aufrufkarte angezeigt werden.

3) Der untere Teil der rechten Seite beschreibt Details zu den Funktionen, die durch die ausgewählte Funktion aufgerufen werden. Insbesondere steht hier ein Callgraph und eine Aufruferkarte bereit.

Weitere Details über `kCachegrind` können Sie dem offiziellen Handbuch entnehmen.

5 Aufgabe

Ihre Aufgabe ist nun die Bearbeitung der einzelnen Programme in dem Ordner `tasks`. Ziel ist dabei die Reduktion des Energieverbrauchs der Programme. Sie dürfen diese nach Belieben abändern, ohne die Grundfunktion zu beeinflussen. D.h. ein `Hello-World`-Programm sollte nach ihrer Bearbeitung weiterhin “Hello World” ausgeben. Bearbeiten Sie die Programme nacheinander und achten Sie auf die Compilierbarkeit, die sie über das Script `builder.sh` testen können. Sollten Sie den Energieverbrauch nicht weiter optimieren können, wechseln sie gerne zum nächsten Programm. Zum Öffnen eines Programms nutzen sie am Besten die eingebaute Funktion von Visual Studio Code über `File -> Open Folder`. Zur Bearbeitung der drei Programme haben Sie 30 Min. Zeit. Sollten Sie früher fertig sein, melden Sie sich bitte.