



# Analyse von Mystery-Programmen

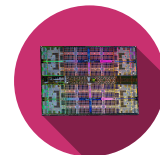
## Mini-Labor Computerarchitektur

## Inhalt

1	Ziel .....	1
2	Installation .....	2
2.1	<b>hyperfine</b> und <b>time</b> .....	2
2.2	<b>bt</b> op .....	2
2.3	Überprüfen der Installation .....	2
3	Analyse der Mystery-Programme .....	4
3.1	Analyse mithilfe von <b>hyperfine</b> .....	5

## 1 | Ziel

Ziel dieses Mini-Labors ist es, unbekannte Programme zu Analysieren mit Debug Tools wie der *Aktivitätsanzeige* (MacOs), *Task Manager* (Windows) oder *bt*op (Linux, MacOS, Windows) sowie und Leistungsanalyse Tools wie **hyperfine** (Linux, MacOS, Windows) oder **time** (Linux, MacOS).



## 2 | Installation

Zuerst müssen Sie die verschiedenen Tools installieren, die wir für die Leistungstests verwenden werden.

Um die Installation zu vereinfachen, werden die Tools über die folgenden Paketmanager installiert:

MacOS "**brew**", bash:

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Windows "**scoop**", powershell:

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser  
Invoke-RestMethod -Uri https://get.scoop.sh | Invoke-Expression
```

### 2.1 hyperfine und time

**hyperfine** ist eine Kommandozeile benchmarking Applikation.

**hyperfine** ist verfügbar unter <https://github.com/sharkdp/hyperfine?tab=readme-ov-file#installation>.

MacOS	Windows
<code>brew install hyperfine</code>	<code>scoop install hyperfine</code>

Das Program **time** ist auf MacOS und Linux bereits vorinstalliert, auf Windows ist dieses nicht verfügbar.

### 2.2 btop

**btop** ist ein Terminal basiertes Systemmonitoring Tool ähnlich wie eingebaute Tools wie *Task Manager* (Windows) oder *Activity Monitor* (MacOS).

**btop** ist verfügbar unter <https://github.com/aristocratos/btop?tab=readme-ov-file#installation> für Linux und Mac. Für Windows, verwenden Sie den Fork **btop4win** <https://github.com/aristocratos/btop4win?tab=readme-ov-file#installation>.

MacOS	Windows
<code>brew install btop</code>	<code>scoop install btop-lhm</code>

### 2.3 Überprüfen der Installation

Um zu überprüfen, ob die Installation erfolgreich war, führen Sie die folgenden Befehle in einem Terminal aus:



```
hyperfine --version
btop      --version
time time          # Linux MacOS only
```



### 3 | Analyse der Mystery-Programme

Die Binär-Program im Ordner `car_labs/dbg/release/` lässt sich mit verschiedenen Parametern ausführen.

```
Usage: rust_mystery_v1_0_0_Mac_AARCH64 [OPTIONS]
```

Options:

```
-m, --mystery <MYSTERY>
-h, --help                Print help
-V, --version              Print version
```



Je nach Betriebssystem muss ein andere Binärdatei ausgeführt werden.

- `rust_mystery_v1_0_0_Mac_AARCH64` für MacOS
- `rust_mystery_v1_0_0_Linux_x64` für Linux
- `rust_mystery_v1_0_0_Windows_x64.exe` für Windows

**Passen Sie die untenstehenden Befehle demensprechend an.**

Die Option `-m` oder `--mystery` erwartet einen Wert von **1** bis **5**. Jeder Wert führt zu einem anderen Verhalten des Programms.

Das Program kann direkt mit der Option `-m` ausgeführt werden.

```
./rust_mystery_v1_0_0_Mac_AARCH64 -m 1
```

Die Programme dauern jeweils nur eine kleine Zeit. Um die Ausführungsgeschwindigkeit zu messen, verwenden wir das Tool **hyperfine** sowie **bttop**.

Starten sie zuerst in einem separatem Terminal **bttop** um die Systemauslastung zu überwachen. Danach können Sie das jeweilige Program mit **hyperfine** ausführen. z.B Mystery 4:

```
hyperfine --warmup 3 --export-markdown mystery-4.md --show-output --min-runs 10
"release/rust_mystery_v1_0_0_Mac_AARCH64 -m 4"
```



Führen Sie alle Varianten `-m 1` bis `-m 5` des Programmes aus und Analysieren Sie die Ausgabe von **hyperfine** und **bttop**.



### 3.1 Analyse mithilfe von hyperfine

Am Ende eines **hyperfine**-Benchmarks wird eine Zusammenfassung der Ausführungsgeschwindigkeit angezeigt.

```
Time (mean ± σ):      64.9 ms ± 13.3 ms    [User: 14.8 ms, System: 12.5 ms]
Range (min ... max):  58.3 ms ... 147.6 ms  43 runs
```



Diese Informationen sind auch in den von **hyperfine** exportierten Markdown-Dateien verfügbar, hier **mystery-4.md**.

In diesem Fall wurde das Program 43 mal ausgeführt. Durchschnittlich dauerte eine Ausführung **64.9ms** mit einer Variation von **13.3ms**. Die Minimale Ausführungsdauer war **58.3ms** und die Maximale **147.6ms**.

Wichtig sind auch die Werte **User** und **System**. Diese Werte geben an, wie lange das Program in der User- und System-Space verbraucht hat. In diesem Fall **14.8ms** und **12.5ms**.



Was bedeuten die Werte **User** und **System** konkret? Inwiefern unterscheiden sie sich voneinander?



Warum ist die Summe von **User** und **System** nicht gleich der Gesamtlaufzeit?

Schauen Sie sich für alle Programme die Ausführungszeit, User und System Werte an. Parallel dazu, beobachten Sie die CPU, GPU, Speicher ... mit **btcp**.

Versuchen Sie zu verstehen, warum die Programme unterschiedlich lange dauern.



Geben Sie eine Hypothese zu den Operationen ab, die jedes Programm möglicherweise ausführt.

Argumentieren Sie anhand der gemessenen Werte / Verbräuche.