



Analyse de Programmes Mystères

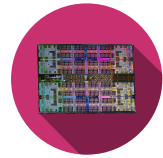
Labo Architecture des ordinateurs

Contenu

1 Objectifs	1
2 Installation	2
2.1 hyperfine et time	2
2.2 btop	2
2.3 Vérifier l'installation	3
3 Analyse de programmes mystères	4
3.1 Analyse avec hyperfine	5

1 | Objectifs

L'objectif de ce mini-labo est d'analyser des programmes inconnus avec des outils de débogage tels que *Activity Monitor* (MacOs), *Task Manager* (Windows) ou *btop* (Linux, MacOS, Windows) ainsi que des outils d'analyse de performance tels que **hyperfine** (Linux, MacOS, Windows) ou **time** (Linux, MacOS).



2 | Installation

Tout d'abord, vous devez installer les différents outils que nous utiliserons pour les tests de performance.

Pour simplifier l'installation, les outils sont installés via les gestionnaires de paquets suivants :

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 et time

hyperfine est une application de benchmarking en ligne de commande.

hyperfine est disponible sous <https://github.com/sharkdp/hyperfine?tab=readme-ov-file#installation>.

MacOS	Windows
brew install hyperfine	scoop install hyperfine

Le programme **time** est déjà installé sur MacOS et Linux, il n'est pas disponible sur Windows.

2.2 btop

btop est un outil de surveillance du système basé sur le terminal similaire aux outils intégrés tels que *Task Manager* (Windows) ou *Activity Monitor* (MacOS).

btop est disponible sous <https://github.com/aristocratos/btop?tab=readme-ov-file#installation> pour Linux et Mac. Pour Windows, utiliser le fork **btop4win** <https://github.com/aristocratos/btop4win?tab=readme-ov-file#installation>.

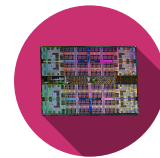
MacOS	Windows
brew install btop	scoop install btop-lhm



2.3 Vérifier l'installation

Pour vérifier que l'installation a réussi, exécutez les commandes suivantes dans un terminal :

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



3 | Analyse de programmes mystères

Les programmes binaires dans le dossier `car_labs/dbg/rust-mystery/release/` peuvent être exécutés avec différents paramètres.

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

Options:

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



Selon le système d'exploitation, un autre fichier binaire doit être exécuté.

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

Adaptez en conséquence les commandes ci-dessous.

L'option `-m` ou `--mystery` attend une valeur de **1** à **5**. Chaque valeur entraîne un comportement différent du programme.

Le programme peut être exécuté directement avec l'option `-m`.

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

Les programmes ne durent que très peu de temps. Pour mesurer la vitesse d'exécution, nous utilisons l'outil **hyperfine** ainsi que **bttop**.

Commencez par démarrer **bttop** dans un terminal séparé pour surveiller l'utilisation du système. Ensuite, vous pouvez exécuter le programme respectif avec **hyperfine**. Par exemple, 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"
```



Exécutez toutes les variantes `-m 1` à `-m 5` du programme et analysez la sortie de **hyperfine** et **bttop**.



3.1 Analyse avec hyperfine

À la fin d'un benchmark **hyperfine**, un résumé de la vitesse d'exécution est affiché.

```
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
```



Ces informations sont aussi disponibles dans les fichiers markdown exportés par **hyperfine**, ici **mystery-4.md**.

Dans ce cas, le programme a été exécuté 43 fois. En moyenne une exécution a duré **64.9ms** avec une variation de **13.3ms**. La durée d'exécution minimale était de **58.3ms** et la maximale de **147.6ms**.

Les valeurs **User** et **System** sont également importantes. Ces valeurs indiquent combien de temps le programme a passé dans l'espace utilisateur et système. Dans ce cas, **14.8ms** et **12.5ms**.



A quoi correspondent concrètement les valeurs **User** et **System**? En quoi diffèrent-elles l'une de l'autre ?



Pourquoi le cumul de **User** et **System** n'est pas égal au temps total d'exécution ?

Regardez le temps d'exécution, les valeurs utilisateur et système pour tous les programmes. En parallèle, observez la consommation CPU, GPU, mémoire ... avec **htop**.

Essayez de comprendre pourquoi les programmes prennent des durées différentes.



Donnez une hypothèse sur les opérations que peuvent être entrain d'effectuer chaque programme.

Argumentez à l'aide des valeurs / consommations mesurées.