

Projet : Architecture système

Trouvez la dernière mise à jour du projet ici : [https://github.com/Anthony-Jhoiro/Archi-Sys_sophia]

Structure de fichier

```
.
├── daemon
│   ├── daemon.c          # Fichier source principal pour le démon
│   ├── daemon.h          # Header pour les fonctions du démon
│   └── Makefile          # Makefile pour compiler le démon
├── invoker
│   ├── help.txt          # Texte à afficher avec la commande --help
│   ├── invoker.c         # Fichier source principal pour l'invocateur
│   ├── invoker.h         # Header pour l'invocateur
│   └── Makefile          # Makefile pour compiler l'invocateur
├── rapport.md            # Version Markdown de ce rapport
├── rapport.pdf           # Version PDF de ce rapport
├── README.md            # README du projet
└── tools
    ├── constants.h       # Constantes du projets
    ├── fifo_tools.c      # Sources pour les outils sur les fifos
    ├── log_tools.c       # Sources pour les outils sur les logs
    ├── Makefile          # Makefile pour les tools
    ├── tools.c           # Sources principale pour les outils
    └── tools.h           # Header pour les outils
```

Le dossier tools contient tout le code susceptible d'être utilisé par les 2 programmes.

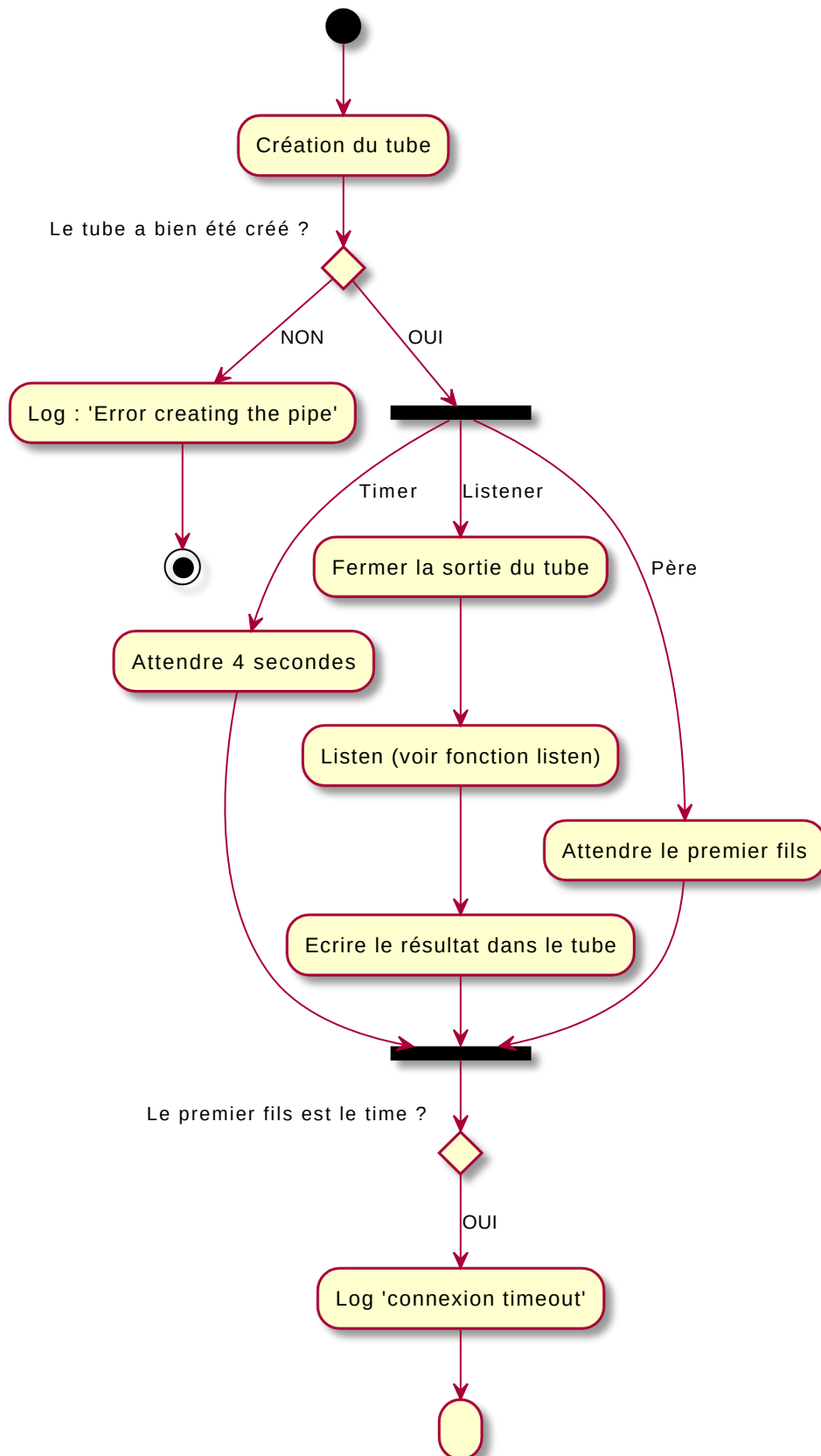
Fonctionnement principale

L'utilisateur va directement interagir avec le programme invocateur. Il va pouvoir executer différentes actions pour controller le démon ainsi qu'executer diverses actions sur le temps.

La communications avec le démon se fait par une fifo dont le nom est définit dans `tools/constants.h`.

Listener

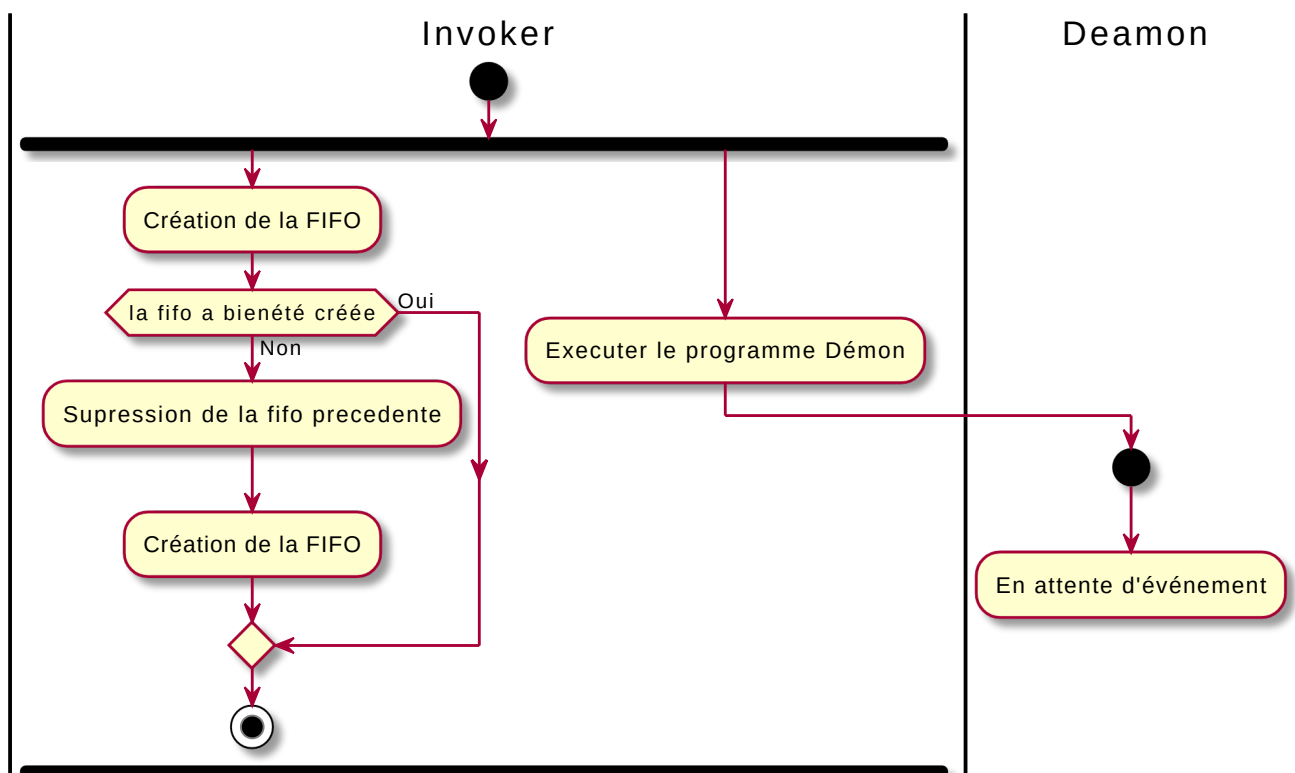
Quand le programme invocateur est en écoute, un timer est lancé en même temps. Il permet de renvoyer une erreur dans le cas où la fifo existe mais que le démon ne répond pas.



Commandes de l'invocateur

Commande `--start`

La commande `--start` permet de créer un démon. Pour ce faire, elle commence par créer un processus fils qui exécutera le programme démon. Puis, dans le processus père, on crée la FIFO. En cas d'échec dans la création de la FIFO, on essaie de supprimer une éventuelle FIFO existante avec la fonction `unlink()` puis on tente à nouveau de créer la FIFO, ci il y a un nouvel échec, on affiche l'erreur à l'utilisateur. Le processus père n'attend pas le fils afin que le démon puisse de nouveau écouter le programme invocateur.



Commande `--stop`

La commande `--stop` permet d'arrêter un démon. Pour commencer, elle vérifie que la FIFO existe puis envoie *KILL* au démon avant de supprimer la fifo avec la fonction `unlink()`. Elle échoue si la FIFO n'existe pas. Il n'y a pas de vérification pour être sûr que le démon est bien arrêté, on peut utiliser la commande `--state`.

Welcome to PlantUML!

If you use this software, you accept its license.
(details by typing `license keyword`)

You can start with a simple UML Diagram like:

```
Bob->Alice: Hello
```

Or

```
class Example
```

You will find more information about PlantUML syntax on <https://plantuml.com>



```
PlantUML 1.2021.01beta3
```

```
[From string (line 2) ]
```

```
@startuml
```

```
!include tools.iuml
```

```
cannot include tools.iuml
```

Commande `--restart`

La commande `--restart` permet de relancer un démon. Pour ce faire, on lance la fonction de la commande `--stop` puis cette de la commande `--start`. La première fonction n'est pas bloquante donc si le démon n'était pas actif, il sera simplement créé.

Welcome to PlantUML!

If you use this software, you accept its license.
(details by typing `license keyword`)

You can start with a simple UML Diagram like:

```
Bob->Alice: Hello
```

Or

```
class Example
```

You will find more information about PlantUML syntax on <https://plantuml.com>



```
PlantUML 1.2021.01beta3
```

```
[From string (line 2) ]
```

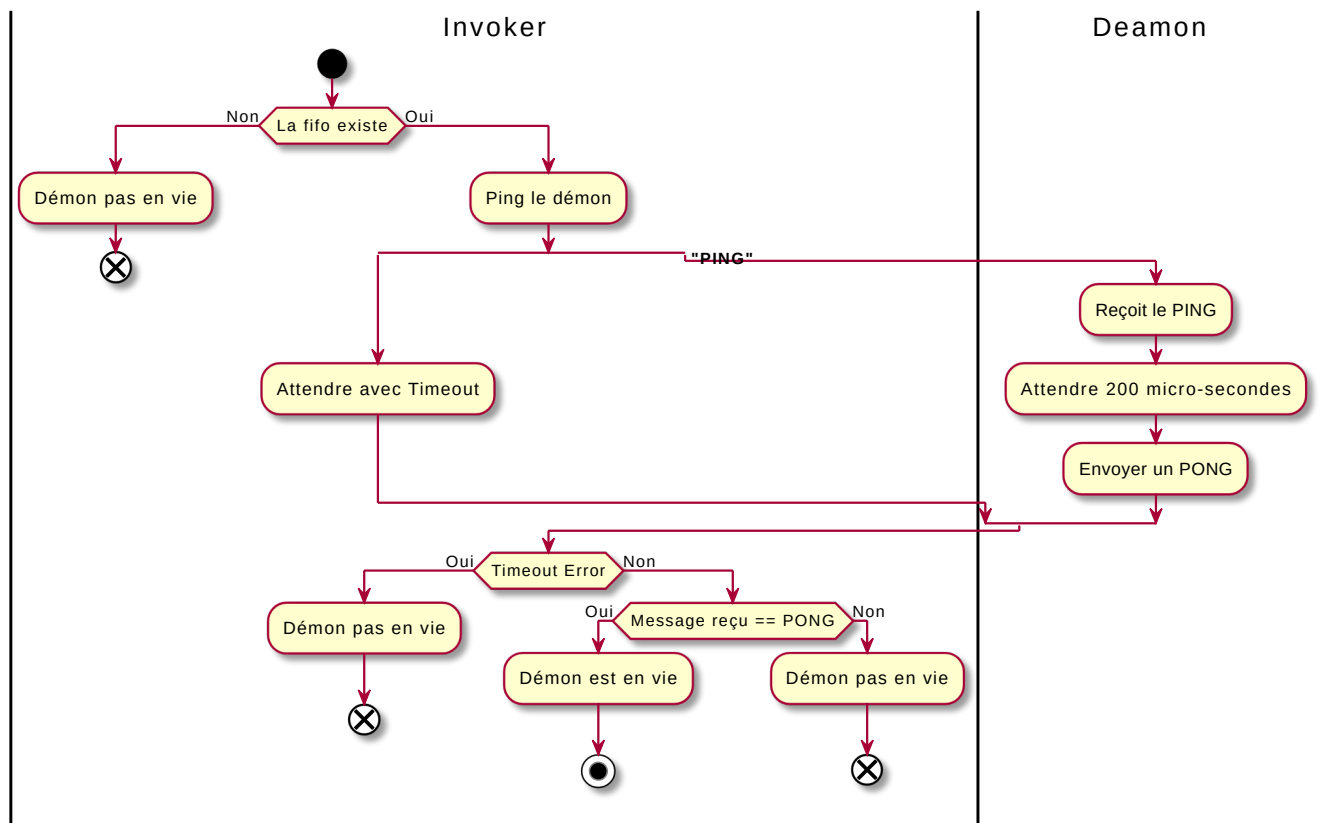
```
@startuml
```

```
!include tools.iuml
```

```
cannot include tools.iuml
```

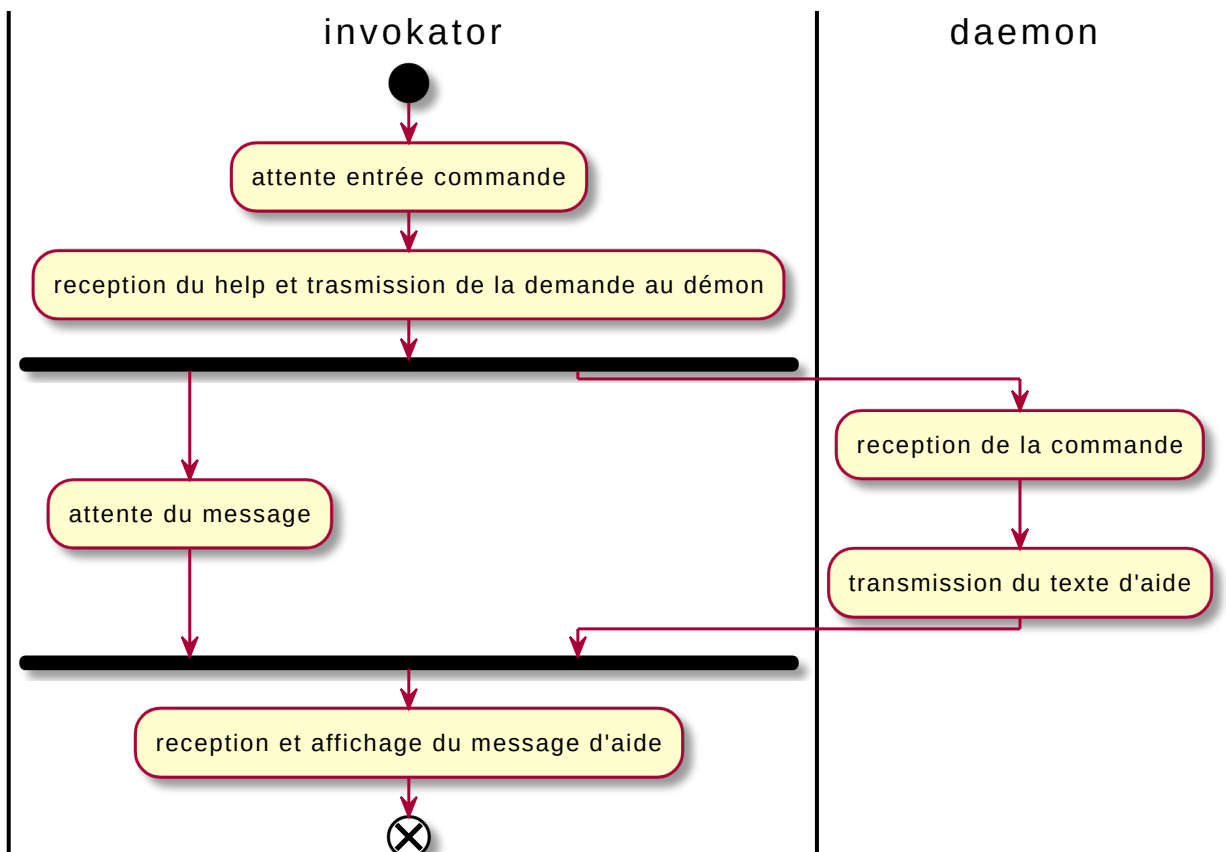
Commande `--state`

La commande `--state` permet de savoir si un démon est actif ou non. Elle commence par vérifier si la FIFO existe puis elle envoie un *PING* au démon par cette FIFO. En recevant un *PING*, le démon va renvoyer un *PONG*, il y a un timeout de 200 microsecondes pour permettre à l'invocateur de passer en écoute. Il aurait été possible de mettre l'invocateur en écoute avec un thread ou un fork mais après des tests, nous nous sommes rendu compte que c'était encore plus long. Une fois le *PONG* reçu, l'invocateur informe l'utilisateur que le démon est actif.



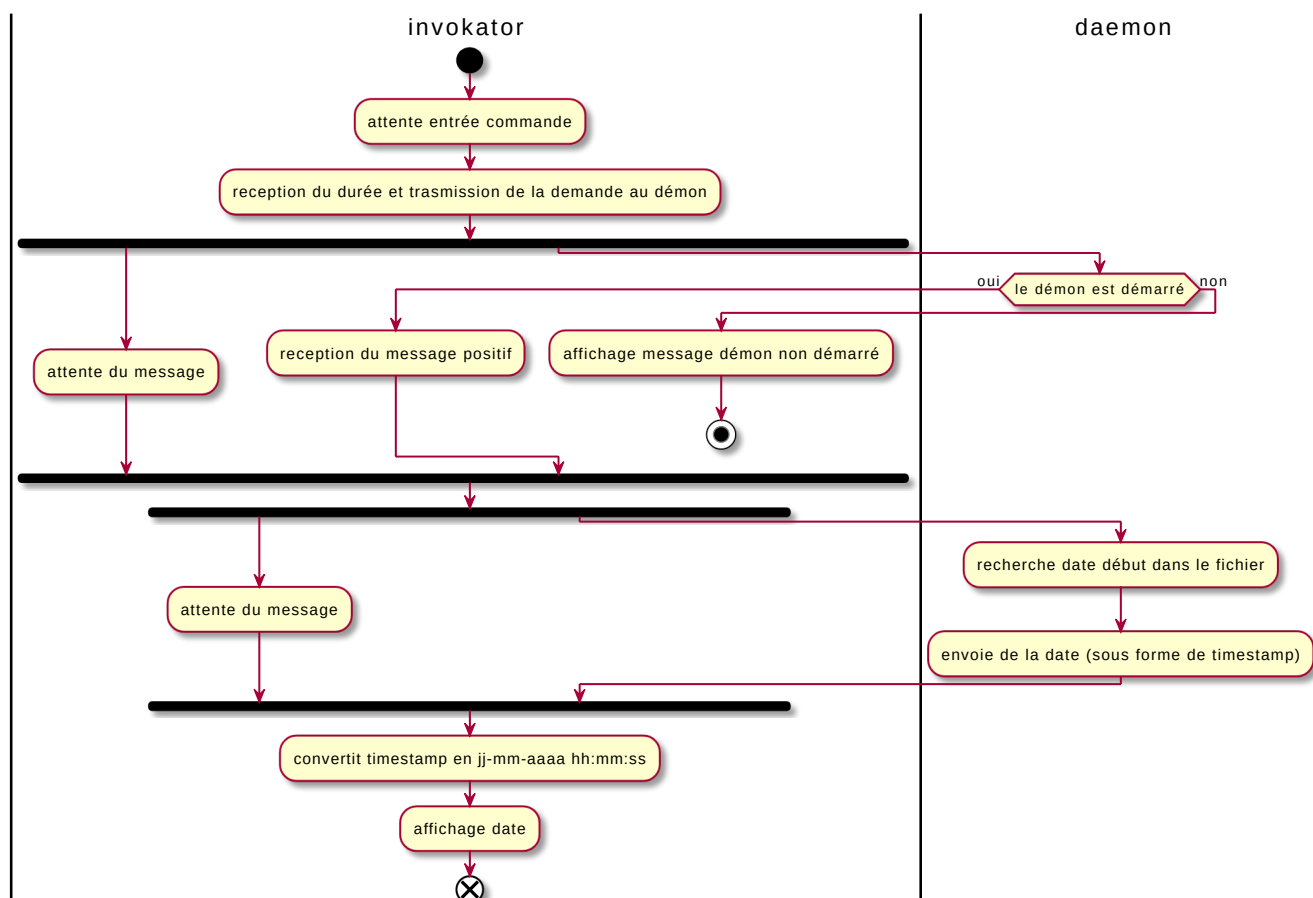
Commande `--help`

La commande `--help` sert à afficher l'aide pour utiliser l'invocateur. Elle affiche simplement le contenu du fichier `help.txt` situé dans le dossier contenant les sources de l'invocateur.



Commande `--date`

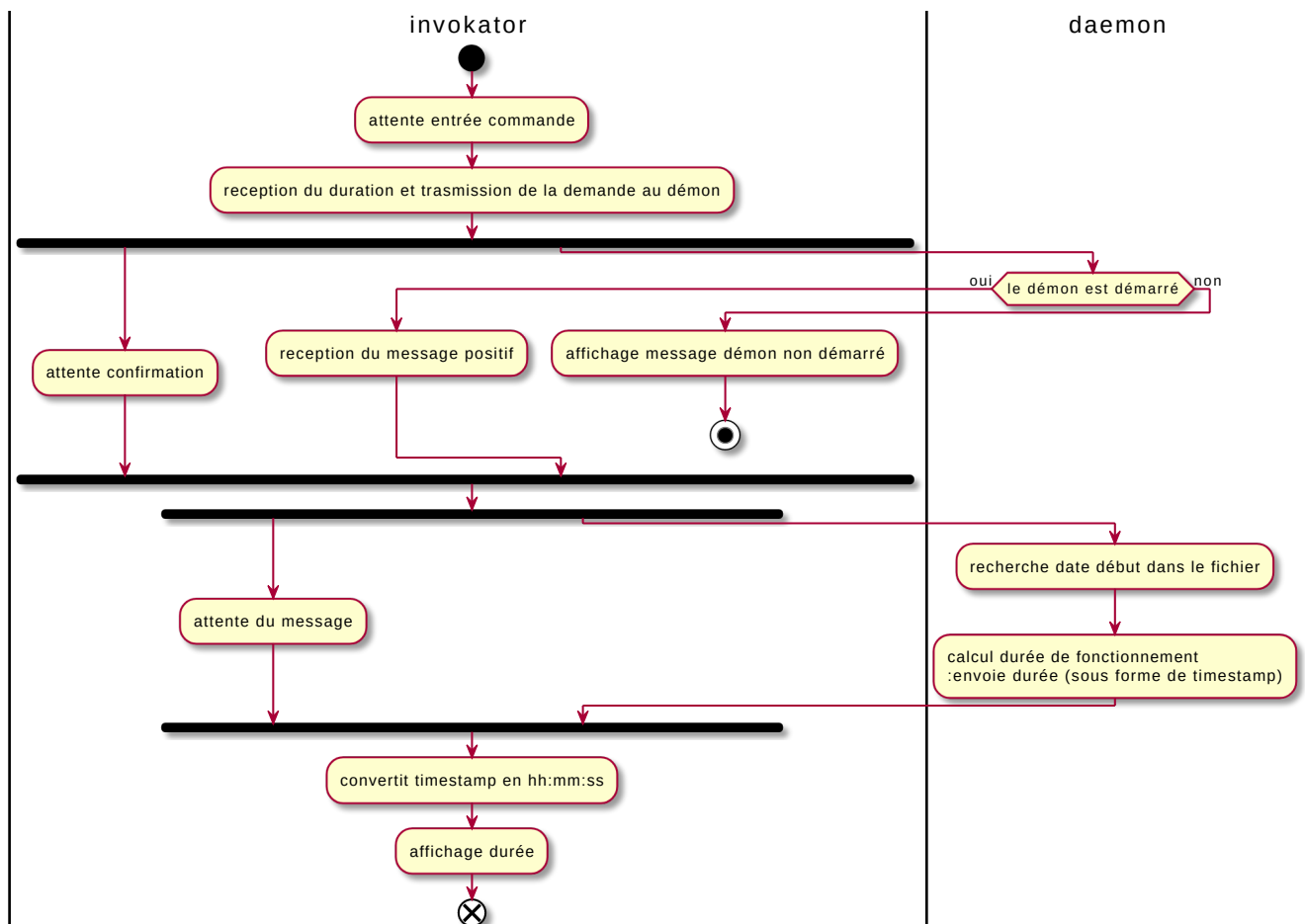
La commande `--date` sert à récupérer la date depuis l'invocateur par le démon. Pour ce faire, le programme invocateur vérifie que la FIFO existe puis envoie *TIME* au démon. Quand le démon reçoit *TIME*, il renvoie la date à l'invocateur par la FIFO. L'invocateur affiche ensuite la date envoyée par le démon.



-d=5

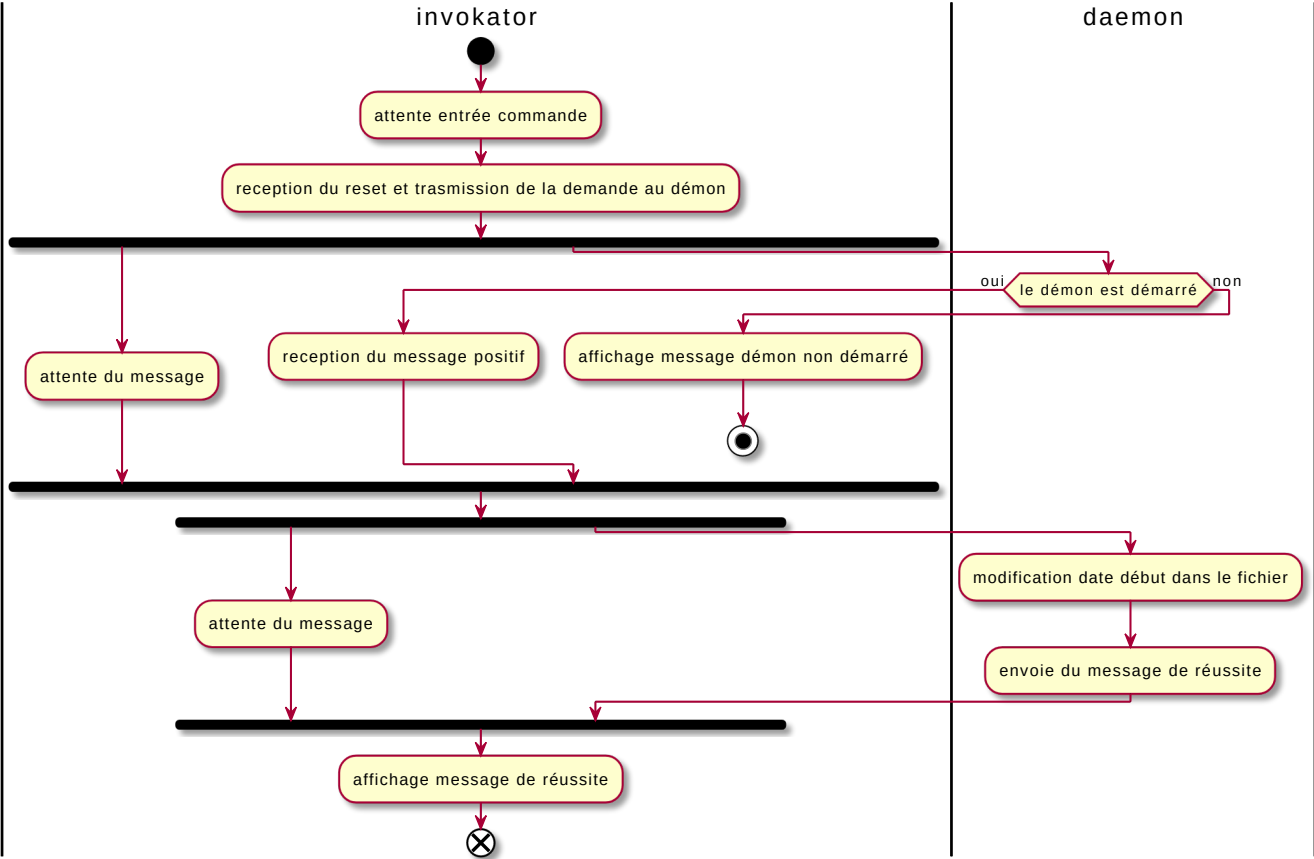
Commande `--duration`

La commande `--duration` sert à obtenir la durée depuis laquelle le démon est en vie ou alors la durée depuis le dernier "reset" (voir commande `--reset`). Pour ce faire, le programme invocateur vérifie que la FIFO existe puis envoie *DURATION* au démon. Quand le démon reçoit *DURATION*, il renvoie la différence entre la "date de base" et la date actuelle à l'invocateur par la FIFO. Cette différence est exprimée en seconde. L'invocateur affiche ensuite la durée envoyée par le démon.



Commande `--reset`

La commande `--reset` permet de modifier la "date de base" du démon. Pour ce faire, le programme invocateur vérifie que la FIFO existe puis envoie *RESET* au démon. Quand le démon reçoit *RESET*, il modifie sa "date de base" pour qu'elle corresponde à la date actuelle.



Partage des tâches