Manuel utilisateur de l'application SystemInfoDaemon

Tables des matières

Description de l'application

L'application SystemInfoDaemon est un programme informatique qui à partir d'un processus démon récupère une requête d'un processus client puis effectue un traitement et lui renvoie le résultat. Par ailleurs, le client peut émettre autant de requête qu'il le souhaite.

À propos des requêtes, le client peut demander :

- Les informations concernant un utilisateur à partir de son uid ou de son nom (login, nom réel, groupe, répertoire dédié, shell utilisé, ...)
- Les informations sur un processus en fonction de son pid (commande, propriétaire, état, . . .)
- Toute commande système usuelle (exemple : ls, cat, pwd, …)

Pré-requis

Afin d'utiliser l'application SystemInfoDaemon, plusieurs outils sont nécessaires à son fonctionnement.

Vous devez avoir :

- **Système d'exploitation Linux :** pour éxécuter l'application écrite en langage C
- **Version du langage C :** c11 version publiée 2011 avec l'API POSIX 2008
- **Programme Make :** pour compiler et générer un fichier exécutable grâce à un ensemble d'actions écrites dans un fichier nommé Makefile.

Compilation et Exécution

Tout d'abord, il vous faut extraire l'application et vous rendre ensuite le dossier SystemInfoDaemon.

Ensuite, il faut effectuer un nettoyage afin d'être sûr que la compilation va s'effectuer. Pour cela, il faut utiliser la commande suivante dans votre terminal : **make mrproper**

Après, il faut construire les dossiers qui vont contenir les fichiers objets, la bibliothèque dynamique de la zone de mémoire partagée et les fichiers exécutables en tapant : **make start_build**

Enfin l'étape de compilation et d'exécution s'effectue en même temps.
Vous êtes dans l'obligation de lancer d'abord le démon avant le ou les clients.
En effet, car s'il n' y a pas de démon, les clients ne peuvent interagir avec un démon.
Le lancement du démon se fait avec la commande : **maker start_daemon**
Et le lancement d'un client s'effectue avec la commande : **make start_client**

Fonctionnalités

L'application est dotée de plusieurs fonctionnalités permettant au client d'interagir avec le démon d'information.

1. Information sur un processus

Tout d'abord, il faut que dans un nouveau terminal il affiche la liste des processus avec la commande **ps axu** et note un identifiant de processus (PID). Pour illustrer, voici l'exemple du PID 10.

```
ZaofuMachine@kali:~/Bureau/SystemInfoDaemon$ ps axu
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.9 167388  9324 ?        Ss   12:36   0:03 /sbin/init splash
root           2  0.0  0.0      0     0 ?        S    12:36   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   12:36   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   12:36   0:00 [rcu_par_gp]
root           6  0.0  0.0      0     0 ?        I<   12:36   0:00 [kworker/0:0H-kblockd]
root           9  0.0  0.0      0     0 ?        I<   12:36   0:00 [mm_percpu_wq]
root          10  0.0  0.0      0     0 ?        S    12:36   0:00 [ksoftirqd/0]
root          11  0.0  0.0      0     0 ?        I    12:36   0:01 [rcu_sched]
root          12  0.0  0.0      0     0 ?        S    12:36   0:00 [migration/0]
root          13  0.0  0.0      0     0 ?        S    12:36   0:00 [cpuhp/0]
root          14  0.0  0.0      0     0 ?        S    12:36   0:00 [cpuhp/1]
root          15  0.0  0.0      0     0 ?        S    12:36   0:00 [migration/1]
root          16  0.0  0.0      0     0 ?        S    12:36   0:00 [ksoftirqd/1]
root          18  0.0  0.0      0     0 ?        I<   12:36   0:00 [kworker/1:0H-kblockd]
root          20  0.0  0.0      0     0 ?        S    12:36   0:00 [kdevtmpfs]
root          21  0.0  0.0      0     0 ?        I<   12:36   0:00 [netns]
root          22  0.0  0.0      0     0 ?        S    12:36   0:00 [kauditd]
```

Ensuite, dans le terminal du client, il faut respecter le format : **info_proc PID** et vous allez pouvoir consulter les informations sur un processus. Les informations sont le nom du processus, son état, son pid et le pid de son ancêtre.

```
[15:05:32][Client] : info_proc 10
[15:05:36][Daemon] :
Name:   ksoftirqd/0
State:  S (sleeping)
Pid:    10
PPid:   2
```

2. Information sur un utilisateur

Au préalable, pour récupérer des informations sur un utilisateur vous devez avoir soit le nom de l'utilisateur ou l'identifiant de l'utilisateur. Pour illustrer, vous pouvez tester avec votre propre utilisateur. Afin de récupérer le nom d'utilisateur, il s'agit d'exécuter la commande **whoami** et d'ensuite utiliser **id -u nom d'utilisateur** pour avoir l'identifiant d'utilisateur.

```
ZaofuMachine@kali:~/Bureau/SystemInfoDaemon$ whoami
ZaofuMachine
ZaofuMachine@kali:~/Bureau/SystemInfoDaemon$ id -u ZaofuMachine
1000
```

Maintenant, vous pouvez éxécuter la commande en suivant le format **info_user <id ou nom d'utilisateur>.**

```
[15:29:22][Client] : info_user 1000
[15:29:26][Daemon] :
Nom d'utilisateur              : ZaofuMachine
Identifiant utilisateur        : 1000
Mot de passe de l'utilisateur  : x
Groupe de l'utilisateur        : 1000
Information sur l'utilisateur  : joo,,,
Répertoire home                : /home/joo
Programme shell                : /bin/bash
close pipefd[0]: Bad file descriptor
[15:29:26][Client] : info_user ZaofuMachine
[15:29:32][Daemon] :
Nom d'utilisateur              : ZaofuMachine
Identifiant utilisateur        : 1000
Mot de passe de l'utilisateur  : x
Groupe de l'utilisateur        : 1000
Information sur l'utilisateur  : joo,,,
Répertoire home                : /home/joo
Programme shell                : /bin/bash
close pipefd[0]: Bad file descriptor
```

3. Commande usuelle

L'application a aussi la possibilité d'exercer des commandes usuelles notamment ls, dir et pwd et encore bien d'autres.
Voici un exemple avec la commande dir permettant de lister le contenu du répertoire courant.

```
[15:34:09][Client] : dir
[15:34:15][Daemon] :
bin  config  inc  lib  Makefile  obj  README.md  src  Sujet.pdf
```

4. Les signaux

La gestion des signaux de terminaison sont pris en compte dans l'application.

Vous pouvez donc envoyer un signal de terminaison au client où au démon qui se chargera de libérer les ressources utilisées et de s'éteindre.

Voici la liste des signaux pris en charge :

SIGINT, SIGHUP, SIGTERM, SIGUSR1, SIGUSR2, SIGQUIT et SIGTSTP.

Voici un exemple ci-dessous où on envoie un signal au client en récupérant son pid avec la commande ps axu. Ensuite, on lui envoie un signal avec kill -SIGINT PID.

```
Fichier  Actions  Éditer  Vue  Aide

ZaofuMa+    4563  0.0  8.4 702196 84968 ?        Sl   15:00   0:01 /usr/bin/qterminal
ZaofuMa+    4566  0.0  0.5   8244  5124 pts/1    Ss   15:00   0:00 /bin/bash
ZaofuMa+    4692  0.0  8.0 701836 81196 ?        Sl   15:01   0:02 /usr/bin/qterminal
ZaofuMa+    4695  0.0  0.5   8940  5572 pts/2    Ss   15:01   0:00 /bin/bash
root        4758  0.0  0.0      0     0 ?        I    15:04   0:00 [kworker/0:1-ata_sff]
root        4827  0.0  0.0      0     0 ?        I    15:17   0:00 [kworker/u4:0-events_unbound]
root        4839  0.0  0.0      0     0 ?        I    15:25   0:00 [kworker/u4:1-ext4-rsv-conversion]
root        4849  0.0  0.0      0     0 ?        I    15:29   0:00 [kworker/0:0-events]
root        4858  0.0  0.0      0     0 ?        I    15:34   0:00 [kworker/0:2-events]
root        4906  0.0  0.0      0     0 ?        I    15:39   0:00 [kworker/0:3-events]
ZaofuMa+    4920  0.0  1.9 486944 19696 ?        SNl  15:42   0:00 /usr/lib/x86_64-linux-gnu/tumbler-1/tu
root        4935  0.0  0.0      0     0 ?        I    15:42   0:00 [kworker/u4:2-events_unbound]
ZaofuMa+    4970  0.0  0.2   5668  2224 pts/0    S+   15:43   0:00 make start_daemon
ZaofuMa+    4971  0.0  0.0  76272   896 pts/0    Sl+  15:43   0:00 ./bin/daemon ./config/app.config
ZaofuMa+    4972  0.0  0.2   5668  2196 pts/1    S+   15:43   0:00 make start_client
ZaofuMa+    4973  0.0  0.1   2532  1532 pts/1    S+   15:43   0:00 ./bin/client ./config/app.config
ZaofuMa+    4975  0.0  0.3   8720  3108 pts/2    R+   15:43   0:00 ps axu
ZaofuMachine@kali:~/Bureau/SystemInfoDaemon$ kill -SIGINT 4973
ZaofuMachine@kali:~/Bureau/SystemInfoDaemon$
```

Lorsque le signal a été envoyé, le client voit un message qui l'informe que les ressources sont libérées.

```
═══════════════════════════════════════════════════
      VALEUR DES VARIABLES DE CONFIGURATION
═══════════════════════════════════════════════════
name_empty_semaphore : /sem_empty_432572985365428 (length:26)
name_full_semaphore : /sem_full_432572985365428 (length:25)
name_mutex_semaphore : /sem_mutex_432572985365428 (length:26)
name_shm : /shm_name_589422985365427 (length:25)
buffer : 2
prefix_pipequestion : /tmp/pipequestion_d_%d (length:22)
prefix_pipeanswer : /tmp/pipeanswer_d_%d (length:20)
timeout_daemon : 360
timeout_client : 360
═══════════════════════════════════════════════════
═══════════════════════════════════════════════════
[Client] Création du tube /tmp/pipequestion_d_4973 pour l'envoi de requête
[Client] Création du tube /tmp/pipeanswer_d_4973 pour la réception de requête
En attente d'attribution d'une ressource ...
Le démon vous a attribué une ressource
[15:43:53][Client] : [Client] Toutes les ressources de configuration ont été libérées
[Client] Toutes les ressources ont été libérées
[Client] Fermeture du client
ZaofuMachine@kali:~/Bureau/SystemInfoDaemon$
```

5. Quitter un client

Afin de mettre fin à la communication, le client a la possibilité de fermer le tunnel de communication grâce à la commande **quit**.

```
[15:58:20][Client] : quit
[Client] Toutes les ressources de configuration ont été libérées
[Client] Toutes les ressources ont été libérées
[Client] Fermeture du client
```