Command Launcher

Arborescence du projet

```
.
|-- cmdl.c                      # Sources du client
|-- cmdld.c                     # Sources du daemon
|-- cmdld.conf                  # Fichier de configuration du daemon
|-- inc                         # -- Répertoire contenant les en-têtes des modules
|   |-- common.h                # Définitions communes utilisées par le client et le daemon
|   |-- config.h                # En-tête du module de configuration
|   |-- squeue.h                # En-tête du module de file synchronisée
|-- LICENSE                     # Licence MIT
|-- Makefile                    # Makefile
|-- README.md                   # README
|-- src                         # -- Répertoire contenant les sources des modules
|   |-- config.c                # Sources du module de configuration
|   |-- squeue.c                #  Sources du module de file synchronisée
|-- test                        # -- Répertoire contenant les sources des programmes de test
    |-- test.sh                 # Script shell de test global
    |-- test_squeue.c           # Programme de test du module de file synchronisée
```

Compilation

La cible par défaut de `make` est le client (`cmdl`) et le daemon (`cmdld`). Il est aussi possible de compiler le programme de test de la file synchronisée.

```
$ make
$ make test
```

`cmdl` et `cmdld` se trouvent à la racine du projet, tandis que `test_squeue` reste dans le répertoire dédié aux tests.

Configuration du daemon

Le nombre de workers et la taille maximale de la file synchronisée peuvent être modifiés au travers du fichier de configuration `cmdld.conf`.

Utilisation

Le daemon peut être lancé et arrêté avec les commandes :

```
$ ./cmdld start
$ ./cmdld stop
```

Les clients peuvent maintenant envoyer des commandes :

```
$ ./cmdl 'pwd'
$ ./cmdl 'sleep 5'
```

Il est possible d'envoyer des commandes plus complexes en passant par un shell. Par exemple avec bash :

```
$ ./cmdl 'bash -c -- for x in $(seq 10); do echo $x; done'
$ ./cmdl 'bash -c -- echo $SHELL && whoami'
```

Tester le programme

Puisque le daemon est détaché de tout terminal, il affiche des informations au moyen de logs systèmes (`syslog`). Sur un système utilisant `systemd` (la plupart des distributions Linux modernes), ces logs sont gérés avec la commande `journalctl`. Pour suivre en temps réél les logs du daemon :

```
$ journalctl -f --identifier=cmdld --priority=6
```

Note : changer la priorité à 7 permet d'afficher les messages de debug.

Le script `test/test.sh` lance X commandes `sleep`, X étant le nombre de workers du daemon.

Exemple de logs après avoir configuré le daemon avec 4 workers, lancé `sh test/test.sh` et demandé l'exécution de `echo 'hello world'` en parallèle :

```
$ journalctl -n 12 --no-hostname --identifier=cmdld --priority=6
-- Journal begins at Tue 2020-10-27 23:12:53 CET, ends at Sat 2020-12-26 18:48:02 CET. --
déc. 26 18:42:38 cmdld[54222]: [maind] daemon started with 4 workers
déc. 26 18:43:41 cmdld[54498]: [wk#00] started job '/bin/sleep 2'
déc. 26 18:43:41 cmdld[54499]: [wk#01] started job '/bin/sleep 4'
déc. 26 18:43:41 cmdld[54501]: [wk#02] started job '/bin/sleep 6'
déc. 26 18:43:41 cmdld[54503]: [wk#03] started job '/bin/sleep 8'
déc. 26 18:43:43 cmdld[54222]: [wk#00] finished job '/bin/sleep 2' (2s) with status 0
déc. 26 18:43:45 cmdld[54222]: [wk#01] finished job '/bin/sleep 4' (4s) with status 0
déc. 26 18:43:46 cmdld[54520]: [wk#00] started job 'echo 'hello world''
déc. 26 18:43:46 cmdld[54222]: [wk#00] finished job 'echo 'hello world'' (0s) with status 0
déc. 26 18:43:47 cmdld[54222]: [wk#02] finished job '/bin/sleep 6' (6s) with status 0
déc. 26 18:43:49 cmdld[54222]: [wk#03] finished job '/bin/sleep 8' (8s) with status 0
déc. 26 18:46:52 cmdld[54222]: [maind] daemon terminated
```