UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR | COLLÈGE STEE SCIENCES ET TECHNOLOGIES POUR L'ÉNERGIE ET L'ENVIRONNEMENT

Systèmes d'exploitation - TP n° 2 - Les files de messages

Samson Pierre <samson.pierre@univ-pau.fr>

04/02/2019

Les files de messages permettent la communication inter-processus. Pour ce TP, nous avons un processus serveur et un processus client qui communiquent entre eux à travers une file de messages. Cette file de messages peut contenir un ou plusieurs messages écrits par le client qui seront lus par le serveur.

Le travail à rendre

L'archive à rendre doit contenir exactement ces fichiers :

```
$ tar tf tp2-mauro-gaio-samson-pierre.tar.xz | sort
tp2-mauro-gaio-samson-pierre/
tp2-mauro-gaio-samson-pierre/file.ftok
tp2-mauro-gaio-samson-pierre/Makefile
tp2-mauro-gaio-samson-pierre/msq-client.c
tp2-mauro-gaio-samson-pierre/msq-common.c
tp2-mauro-gaio-samson-pierre/msq-common.h
tp2-mauro-gaio-samson-pierre/msq-message.c
tp2-mauro-gaio-samson-pierre/msq-message.h
tp2-mauro-gaio-samson-pierre/msq-server.c
$
```

Les prénoms `mauro` et `samson` ainsi que les noms `gaio` et `pierre` sont à remplacer par les vôtres. Le fichier `file.ftok` est un fichier vide. Il constitue l'un des deux éléments permettant la création d'une clé (le deuxième élément est un identifiant de projet). Cette clé est nécessaire pour obtenir une file de messages. Le fichier `Makefile` contient les règles dont se sert le programme `make`. Les fichiers `msq-message.c` et `msq-message.h` contiennent respectivement les définitions et les déclarations pour le message. Les fichiers `msq-client.c` et `msq-server.c` contiennent respectivement les définitions pour le client et le serveur. Les fichiers `msq-common.c` et `msq-common.h` contiennent respectivement les définitions et les déclarations pour ce qui n'est pas spécifique au client, au serveur et au message.

La compilation du projet doit se passer ainsi :

```
$ make
gcc -std=c89 -pedantic -Wall -Werror -D_GNU_SOURCE -g -c -o msq-common.o msq-common.c
gcc -std=c89 -pedantic -Wall -Werror -D_GNU_SOURCE -g -c -o msq-message.o msq-message.c
gcc -std=c89 -pedantic -Wall -Werror -D_GNU_SOURCE -g -o msq-client.out msq-client.c msq-common.o
msq-message.o
gcc -std=c89 -pedantic -Wall -Werror -D_GNU_SOURCE -g -o msq-server.out msq-server.c msq-common.o
msq-message.o
$
```

Vous êtes autorisés à utiliser les fonctions suivantes :

```
void exit(int status);
int fprintf(FILE *stream, const char *format, ...);
key_t ftok(const char *pathname, int proj_id);
int getchar(void);
int getopt_long(int argc, char * const argv[], const char *optstring, const struct option
*longopts, int *longindex);
struct tm *localtime(const time_t *timep);
int msgctl(int msqid, int cmd, struct msqid_ds *buf);
int msgget(key_t key, int msgflg);
ssize_t msgrcv(int msqid, void *msgp, size_t msgsz, long msgtyp, int msgflg);
int msgsnd(int msqid, const void *msgp, size_t msgsz, int msgflg);
int printf(const char *format, ...);
unsigned int sleep(unsigned int seconds);
char *strcpy(char *dest, const char *src);
size_t strlen(const char *s);
long int strtol(const char *nptr, char **endptr, int base);
time_t time(time_t *t);
int vfprintf(FILE *stream, const char *format, va_list ap);
```

Pour obtenir une bonne note, vous devez respecter ces règles :

- les fichiers doivent être encodés en UTF-8
- les fichiers ne doivent pas contenir de fautes d'orthographe
- les noms de fichiers doivent être ceux indiqués dans ce sujet
- le fichier `msq-message.h` doit être identique à celui du sujet
- les options de compilation doivent être celles précisées dans ce sujet
- les affichages à l'écran doivent correspondre à ceux du sujet
- la solution doit se rapprocher au maximum de ce qui est demandé dans ce sujet
- le code doit être correctement indenté
- le code doit être homogène concernant le nom des variables, les espaces, ...
- la mémoire allouée doit être correctement libérée
- les fonctions que vous utilisez dans votre code doivent figurer parmi celles autorisées dans ce sujet
- les valeurs de retour des fonctions pouvant échouer doivent être vérifiées afin de traiter les erreurs
- le traitement des erreurs consiste à afficher un message d'erreur personnalisé
- le format des messages d'erreur doit être identique à celui présenté dans ce sujet
- les messages d'erreur doivent être envoyés dans le flux d'erreur standard
- les autres messages seront envoyés dans le flux de sortie standard
- le code retourné par un processus rencontrant une erreur doit être 1
- le code retourné par un processus se terminant normalement doit être 0
- l'archive ne doit pas contenir d'autres fichiers que ceux demandés dans ce sujet
- l'archive doit être envoyée au plus tard le 05/04/2019 à 23:59
- l'archive doit être envoyée par e-mail à l'adresse `samson.pierre@univ-pau.fr`
- le sujet de l'e-mail doit être `SysEx - TP 2 - Mauro Gaio - Samson Pierre`
- les prénoms `Mauro` et `Samson` sont à remplacer par les vôtres
- les noms `Gaio` et `Pierre` sont à remplacer par les vôtres
- le travail est à réaliser impérativement en binôme

Le serveur

Le serveur doit proposer plusieurs options. L'option `-h` permet d'obtenir cet affichage :

```
$ ./msq-server.out -h
Usage: ./msq-server.out [OPTION]...
Receive messages from clients through a message queue.

Options:
        -h, --help
                display this help and exit
        -i, --key-proj-id=PROJ_ID
                set the key project identifier to PROJ_ID (the default value is "1")
        -p, --key-pathname=PATHNAME
                set the key pathname to PATHNAME (the default value is "file.ftok")
        -s, --seconds=SECONDS
                set the seconds between each try (the default value is "1", a value less than or
equal to 0 enables the interactive mode where the input stream is read)
        -t, --times=TIMES
                set the number of times this program tries to receive a message (the default value
is "-1", a negative value means repeat for ever)
        -v, --version
                output version information and exit

Report bugs to Mauro Gaio <mauro.gaio@univ-pau.fr> and Samson Pierre <samson.pierre@univ-pau.fr>.
$
```

Les prénoms `Mauro` et `Samson`, les noms `Gaio` et `Pierre` ainsi que les adresses e-mail `mauro.gaio@univ-pau.fr` et `samson.pierre@univ-pau.fr` sont à remplacer par les vôtres. L'option `-v` permet d'obtenir cet affichage :

```
$ ./msq-server.out -v
msq-server 20190204

Copyright (C) 2019 Mauro Gaio and Samson Pierre.

Written by Mauro Gaio <mauro.gaio@univ-pau.fr> and Samson Pierre <samson.pierre@univ-pau.fr>.
$
```

Les prénoms `Mauro` et `Samson`, les noms `Gaio` et `Pierre` ainsi que les adresses e-mail `mauro.gaio@univ-pau.fr` et `samson.pierre@univ-pau.fr` sont à remplacer par les vôtres. La version `20190204` est à remplacer par la version de votre programme. Lorsque le serveur est lancé sans option, il doit simplement attendre l'arrivée d'un message en lisant à intervalles réguliers dans la file de messages :

```
$ ./msq-server.out
proj_id = "1"
pathname = "file.ftok"
```

Lorsqu'un message est écrit par le client, le serveur doit alors l'afficher :

```
$ ./msq-server.out
proj_id = "1"
pathname = "file.ftok"
2019-02-04 08:33:57: 1: This is the default message text
```

Pour arrêter le serveur, nous devons lui envoyer le signal `SIGINT`. Pour cela, nous pouvons presser les touches `Ctrl` et `C` :

```
$ ./msq-server.out
proj_id = "1"
pathname = "file.ftok"
2019-02-04 08:33:57: 1: This is the default message text
^C
$
```

Réessayons de lancer le serveur :

```
$ ./msq-server.out
proj_id = "1"
pathname = "file.ftok"
./msq-server.out:msq-server.c:122: Unable to get the identifier of the System V message queue from
the "0x01044a37" key.
$
```

La raison de l'erreur ci-dessus est que la file de messages obtenue n'a pas été détruite. En effet, avec la commande `ipcs`, nous pouvons voir que la file est toujours présente. Pour la détruire, la commande `ipcrm` peut servir. Une fois la file détruite, nous pouvons relancer le serveur. Cette fois, essayons avec l'option `-i` :

```
$ ./msq-server.out -i 2
proj_id = "2"
pathname = "file.ftok"
```

Le serveur est en attente de messages. Le client essaye d'envoyer un message mais n'y parvient pas parce qu'il doit indiquer le même identifiant de projet que le serveur. Une fois le serveur arrêté et la file de messages détruite, nous essayons l'option `-p` :

```
$ ./msq-server.out -p toto.tok
proj_id = "1"
pathname = "toto.tok"
./msq-server.out:msq-server.c:116: Unable to create the System V IPC key from the "toto.tok"
pathname and the "1" project identifier.
$
```

La raison de l'erreur ci-dessus est que le fichier `toto.tok` n'existe pas. Une fois le fichier `toto.tok` créé, nous pouvons relancer le serveur. Cette fois, essayons avec l'option `-s` :

```
$ ./msq-server.out -s 10
proj_id = "1"
pathname = "file.ftok"
2019-02-04 09:53:14: 1: This is the default message text
2019-02-04 09:53:24: 1: This is the default message text
```

Le client a essayé d'envoyer un message à chaque seconde, cependant le serveur est configuré cette fois pour lire un message toutes les `10` secondes. C'est pour cette raison que nous voyons seulement deux messages lus et affichés par le serveur sur une période de `10` secondes. Comme indiqué dans l'aide du programme, si nous donnons une valeur inférieure ou égale à `0` à cette option, nous passons en mode interactif :

```
$ ./msq-server.out -s 0
proj_id = "1"
pathname = "file.ftok"
Press the Enter key to continue...
Press the Enter key to continue...
2019-02-04 10:13:40: 1: This is the default message text
Press the Enter key to continue...
```

Dans ce mode, le serveur attend que l'utilisateur presse la touche `Entrée`. Une fois la touche pressée, le serveur essaye de lire un message. Avant la première pression de cette touche, aucun client n'a envoyé de message ce qui explique pourquoi rien n'est affiché. Entre la première et la deuxième pression de cette touche, un client écrit un message ce qui

explique pourquoi un message est affiché. Une fois le serveur arrêté et la file de messages détruite, nous essayons l'option `-t` :

```
$ ./msq-server.out -t 3
proj_id = "1"
pathname = "file.ftok"
$
```

Le serveur essaye trois fois de lire un message puis s'arrête. Aucun client n'a envoyé de message durant ces trois tentatives donc aucun affichage de message n'est visible. Puisque cette fois, le serveur s'est arrêté par lui même, il s'est occupé de détruire la file de messages. Nous pouvons vérifier cette affirmation en lançant le serveur une nouvelle fois. Toutes les options que nous venons de voir peuvent être combinées et ceci dans n'importe quel ordre :

```
$ ./msq-server.out -t 3 -s 10 -p toto.tok -i 2
proj_id = "2"
pathname = "toto.tok"
$
```

D'autre part, chaque option courte possède sa version longue. Ainsi, la commande ci-dessus est équivalente à celle-ci :

```
$ ./msq-server.out --key-proj-id=2 --key-pathname=toto.tok --seconds=10 --times=3
proj_id = "2"
pathname = "toto.tok"
$
```

Le client

Comme le serveur, le client doit proposer plusieurs options. L'option `-h` permet d'obtenir cet affichage :

```
$ ./msq-client.out -h
Usage: ./msq-client.out [OPTION]...
Send a message to a server through a message queue.

Options:
        -h, --help
                display this help and exit
        -i, --key-proj-id=PROJ_ID
                set the key project identifier to PROJ_ID (the default value is "1")
        -p, --key-pathname=PATHNAME
                set the key pathname to PATHNAME (the default value is "file.ftok")
        -s, --seconds=SECONDS
                set the seconds between each try (the default value is "1", a value less than or
equal to 0 enables the interactive mode where the input stream is read)
        -t, --times=TIMES
                set the number of times this program tries to send a message (the default value is
"1", a negative value means repeat for ever)
        -v, --version
                output version information and exit
        -x, --message-text=TEXT
                set the message text to TEXT (the default value is "This is the default message
text")
        -y, --message-type=TYPE
                set the message type to TYPE (the default value is "1")

Report bugs to Mauro Gaio <mauro.gaio@univ-pau.fr> and Samson Pierre <samson.pierre@univ-pau.fr>.
$
```

Les prénoms `Mauro` et `Samson`, les noms `Gaio` et `Pierre` ainsi que les adresses e-mail `mauro.gaio@univ-pau.fr` et `samson.pierre@univ-pau.fr` sont à remplacer par les vôtres. L'option `-v` permet d'obtenir cet affichage :

```
$ ./msq-client.out -v
msq-client 20190204

Copyright (C) 2019 Mauro Gaio and Samson Pierre.

Written by Mauro Gaio <mauro.gaio@univ-pau.fr> and Samson Pierre <samson.pierre@univ-pau.fr>.
$
```

Les prénoms `Mauro` et `Samson`, les noms `Gaio` et `Pierre` ainsi que les adresses e-mail `mauro.gaio@univ-pau.fr` et `samson.pierre@univ-pau.fr` sont à remplacer par les vôtres. La version `20190204` est à remplacer par la version de votre programme. Lorsque le client est lancé sans option, il doit simplement essayer d'envoyer un message et l'afficher :

```
$ ./msq-client.out
proj_id = "1"
pathname = "file.ftok"
```

```
./msq-client.out:msq-client.c:162: Unable to get the identifier of the System V message queue from the "0x01044a37" key.
$
```

La raison de l'erreur ci-dessus est que nous avons oublié de lancer le serveur. Une fois le serveur lancé, nous pouvons relancer le client et obtenir cet affichage :

```
$ ./msq-client.out
proj_id = "1"
pathname = "file.ftok"
2019-02-04 11:32:24: 1: This is the default message text
$
```

Relançons le client avec l'option `-i` :

```
$ ./msq-client.out -i 2
proj_id = "2"
pathname = "file.ftok"
./msq-client.out:msq-client.c:162: Unable to get the identifier of the System V message queue from the "0x02044a37" key.
$
```

La raison de l'erreur ci-dessus est que nous avons indiqué un identifiant de projet différent de celui utilisé par le serveur. Maintenant, essayons l'option `-p` :

```
$ ./msq-client.out -p toto.tok
proj_id = "1"
pathname = "toto.tok"
./msq-client.out:msq-client.c:156: Unable to create the System V IPC key from the "toto.tok" pathname and the "1" project identifier.
$
```

La raison de l'erreur ci-dessus est que le fichier `toto.tok` n'existe pas. Une fois le fichier `toto.tok` créé, nous pouvons relancer le client :

```
$ ./msq-client.out -p toto.tok
proj_id = "1"
pathname = "toto.tok"
./msq-client.out:msq-client.c:162: Unable to get the identifier of the System V message queue from the "0x01042a52" key.
$
```

La raison de l'erreur ci-dessus est que nous avons indiqué un fichier différent de celui utilisé par le serveur. Maintenant, essayons l'option `-s` :

```
$ ./msq-client.out -s 10
proj_id = "1"
pathname = "file.ftok"
2019-02-04 12:41:25: 1: This is the default message text
$
```

Le client se comporte de la même façon que si l'option n'avait pas été utilisée. La raison est que le temps de `10` secondes est celui entre chaque message, or nous n'avons qu'un seul message à envoyer. Afin de voir l'intérêt de l'option `-s`, nous devons la combiner avec l'option `-t` :

```
$ ./msq-client.out -s 10 -t 2
proj_id = "1"
pathname = "file.ftok"
2019-02-04 13:43:34: 1: This is the default message text
2019-02-04 13:43:44: 1: This is the default message text
$
```

Ainsi, nous voyons qu'il y a un intervalle de `10` secondes entre les `2` messages envoyés. Revenons à l'option `-s` afin de passer en mode interactif grâce à une valeur inférieure ou égale à `0` :

```
$ ./msq-client.out -s 0
proj_id = "1"
pathname = "file.ftok"
2019-02-04 14:46:25: 1: This is the default message text
Press the Enter key to continue...
```

Dans ce mode, le client attend que l'utilisateur presse la touche `Entrée`. Une fois la touche pressée, le client va s'arrêter puisque nous n'avons qu'un seul message à envoyer. Cependant, si nous combinons cette option avec l'option `-t`, le programme pourra éventuellement envoyer un nouveau message une fois la touche pressée. Maintenant, essayons les options `-x` et `-y` :

```
$ ./msq-client.out -x "My message text" -y 42
proj_id = "1"
pathname = "file.ftok"
2019-02-04 15:40:31: 42: My message text
$
```

Nous voyons que ces options permettent respectivement de personnaliser le type et le texte du message à envoyer. Lorsque le type du message est négatif ou nul ou lorsque le texte du message est trop long, une erreur doit être affichée :

```
$ ./msq-client.out -x "My message text" -y 0
./msq-client.out:msq-message.c:30: Unable to set the "0" message type because its value is less than or equal to "0".
./msq-client.out:msq-client.c:141: Unable to set the "0" message type.
$
```

Toutes les options que nous venons de voir peuvent être combinées et ceci dans n'importe quel ordre. D'autre part, chaque option courte possède sa version longue.

Le message

Voici le contenu du fichier d'en-tête `msq-message.h` contenant les déclarations pour le message :

```c
/**
 * \file msq-message.h
 */
#ifndef MSQ_MESSAGE_H
#define MSQ_MESSAGE_H
/**
 * The size of the message text.
 */
#define MSQ_MESSAGE_TEXT_SIZE 64
/**
 * A message.
 */
typedef struct
{
        long type; /**< The message type. */
        char text[MSQ_MESSAGE_TEXT_SIZE]; /**< The message text. */
} msq_message_t;
/**
 * Prints a message using the "YYYY-MM-DD HH:MM:SS: type: text" format.
 * \param message The message.
 */
void msq_message_print(msq_message_t message);
/**
 * Sets the text of a message.
 * \param message The message.
 * \param text The text.
 * \return -1 on error (if the message text size is greater than MSQ_MESSAGE_TEXT_SIZE), else 0.
 */
int msq_message_set_text(msq_message_t *message, const char *text);
/**
 * Sets the type of a message.
 * \param message The message.
 * \param type The type.
 * \return -1 on error (if the message type value is less than or equal to 0), else 0.
 */
int msq_message_set_type(msq_message_t *message, long int type);
#endif
```