Institut Galilée — Systèmes d'exploitation
Année 2022-2023 — ISPG INFO 1

Devoir de systèmes d'exploitation

Ce devoir est à rendre pour le 23 Juin 2023 au plus tard, tout délai pouvant entrainer une pénalité. Il est à réaliser en binôme.

Objectifs

L'objectif du projet est de placer l'étudiant dans un cadre applicatif global, dans lequel il devra mettre en situation des concepts vus en cours (création de processus, communication et synchronisation entre les processus).

Il s'agit de mettre en place un service de messagerie instantanée permettant à plusieurs utilisateurs d'une même machine, de se connecter au service simultanément. Un mécanisme d'abonnement permet de gérer les nouveaux utilisateurs et les sortants (abandon du service). Une fonctionalité de connexion, permet à tout instant de connaître la liste des connectés, afin de provoquer une discussion avec eux. Cette discussion passe par l'ouverture de fenêtres de dialogues. Elles permettent à deux personnes de discuter entre elles, via ces fenêtres («chat»).

Première partie

Gestion des contacts

Un premier processus, le processus de messagerie instantanée (PMI) se chargera d'interroger l'utilisateur sur ses actions qu'il entrera en ligne de commande. L'interpréteur de commandes à réaliser devra supporter les commandes suivantes :

- `e <nom>` : s'enregister avec le nom `<nom>`.
- `p <nom>` : ouvrir un dialogue avec l'utilisateur nommé `<nom>`.
- `q` : quitter le PMI.

L'ouverture de dialogue crée deux nouveaux processus dans lesquels s'éxécute un terminal `xterm` qui lance une fenêtre de discussion (une par utilisateur dans chaque terminal).

Fenêtres de dialogue

Une fois les deux terminaux lancés, ils exécutent chacun un processus de communication. Ces deux processus communiquent par des tubes (un pour chaque direction).

Le programme de discussion doit lire des messages saisis par l'utilisateur et envoyer la ligne tapée via le tube de sortie après lecture d'un retour à la ligne ('\n'). Si le texte saisi est "/quitter", la connexion aux tubes sera fermée et le programme se terminera.

À l'intérieur de ce programme de discussion, un thread (processus léger) sera constamment à l'écoute du tube d'entrée et affichera les messages dès leur réception.

Deuxième partie

La seconde partie porte sur l'enregistrement des utilisateurs. Vous simulerez un serveur grâce à un segment de mémoire partagée (SHM). Ce segment contiendra la liste des utilisateurs connectés. Chaque utilisateur lancera un PMI et s'enregistrera avec la commande `. Dux` nouvelles commandes sont à ajouter :

— `l` permettra de lister les utilisateurs connectés ;
— `d` permettra de se «déconnecter» (retirer son nom du SHM).

La commande `p` devra être modifée afin de ne pouvoir parler qu'à un utilisateur enregistré sur le «serveur». Pour cette deuixème partie, une description détaillée (nouvelles fonctions, structure du segment de mémoire partagée, synchronisation des accès, etc.) est demandée à l'écrit.

Prototypes des fonctions à utiliser

PMI

```
int main(int argc, char **argv);
```

Fonction principale, appelle `interp_commande` en boucle jusqu'a réception de la commande `q`.

```
int interp_commande(char *commande);
```

Interpréte les commandes et exécute la fontion appropriée. Renvoie 1 si la commande est différente de `q`, 0 sinon. La fonction `decoupe_mots` servira à découper la commande dans les cas où il y a des arguments (par exemple : la commande `p`).

```
int parler(char *nom);
```

Lance deux fenêtres de discussion entre l'utilisateur du PMI et un autre dont le nom est passé en argument. Retourne 0 si succès, -1 sinon.

```
int enregistrer(char *nom);
```

Enregistre le nom de l'utilisateur local (pour la seconde partie il faut l'inserer dans le SHM). Retourne 0 si succès, -1 sinon.

Dialogue

```
int main(int argc, char **argv);
```

Fonction principale, lance un thread avec la fonction lecture puis lit et envoie un message en boucle jusqu'a saisie de `"/quitter"`.

```
void* lecture(void *arg);
```

Lit en boucle sur le tube d'entrée.

Troisième partie

Cette partie est optionnelle. Il s'agira d'étendre votre application à un contexte distribué. Utilisez les `socket` pour permettre à des utilisateurs, connectés sur des machines distantes, de communiquer simultanément. Quelle solutions proposeriez vous pour enregistrer les utilisateurs connectés ?

À remettre

Votre travail devra être accompagné d'un rapport, dans un format de texte courant(PostScript, Acrobat (pdf), OpenDocument, ...), décrivant votre travail, les problèmes posés et les réponses que vous y avez apporté. Il vous est demandé également de présenter une réflexion sur les diverses modifications/améliorations envisageables et les limites de cette approche, notamment sur les limites des outils utilisés.

Le rapport n'a pas besoin d'être volumineux, allez à l'essentiel. Merci de ne pas inclure l'intégralité du code dans le rapport ; n'incluez que des parties de code, et seulement si elles sont nécessaires dans vos explications. Vous devez déposer votre projet sous la forme d'un fichier archive `nom1-nom2.tar.gz` sur l'espace moodle dédié de l'ENT. L'archive doit contenir :

— Le code source,
— Le rapport,
— Et tout fichier d'information, de jeu d'essais ou autre, pouvant être utile.

Le sujet du mail doit être intitulé Remise devoir SE. Le contenu du mail doit impérativement inclure les noms et prénoms des membres du binôme.