Titre du projet : Mini Serveur (version simplifiée)

Objectif du projet :

- **Concevoir et réaliser un Mini Serveur en en C sous système de la famille Linux** (soit sur une des distribution Linux, ou sur une des distribution de 'Linux Like' (Mac Os-X, ..))

Aspects techniques du projet :

- Ce projet sera limité et simplifié en termes de fonctionnalités, de volumes (données et instances applicatives), et de protocoles utilisés. Il représente les premières étapes d'établissement de connexion et de mise en place pour une communication entre clients via un serveur.
- Ce projet permet, sur une base simple, de mettre en œuvre des processus d'échanges et de synchronisation en processus. Les choix des technologies employées pour ces échanges seront à justifier.

Objectif fonctionnel du projet :

- Un ensemble de robots potentiellement activables sont connectés par un réseau.
- Chacun des robots à un rôle actif ou passif (ceci est un simple statut pour le projet indiquant fonctionnellement une activité propre ou une attente de réception d'information).
- Le serveur offre un service de supervision des clients connectés et de leur état (actif/inactif qui correspond fonctionnellement à une acceptation de recevoir un message d'un autre client, ou non). Cette dernière partie de communication entre clients n'étant pas gérée dans cette version simplifiée du projet.
- Les fonctions à mettre en place (sur le serveur) :
    - gérer la liste des robots avec l'état actif ou passif,
    - gérer le changement d'état,
    - permettre l'affichage de la liste des états des robots.

Contraintes des services à réaliser :

- Le service de communication entre les clients et le serveur, des états des clients et de mise à jour de ces données qui permettront fonctionnellement pour une version plus large de ce projet la communication entre clients doit être le plus rapide possible (l'objectif étant que les robots ainsi connectés puissent être connus sur le serveur dès qu'une mise à jour de l'état du robot est demandée par celui-ci).
- Le service d'affichage des états des clients à vocation de contrôle (interface graphique de supervision) pour un suivi simple (il ne s'agit pas de gestion d'alertes avec un contrôle visuel, mais simplement une trace écran de l'état des clients connectés). L'affichage peut être en mode texte dans un terminal simple.

Terminologie :

- Un robot sera représenté par un programme qui se connecte au serveur (il devient client).
- Un robot qui se connecte au serveur pour se faire connaître sera nommé client (du serveur) dans ce projet.

Les fonctionnalités du serveur à réaliser :

- Le serveur permet de gérer les clients : fonctions de connexion et de déconnexion, et fonction d'identification (numéro client).
- Le serveur permet de gérer la liste de l'état des clients connectés : fonctions de mise à jour de l'état et d'affichage.
-

Les fonctionnalités d'un client à réaliser :

- Ses fonctions sont :
  - connexion au serveur
  - déconnexion du serveur
  - information au serveur de passage à état actif
  - information au serveur de passage à état inactif

Les spécifications techniques pour la réalisation :

- L'état du client
  - deux valeurs : état actif ou état inactif.
  - valeur état : de type entier (1 ou 0) (voir sur `send_state` plus loin le formalisme de codage dans les messages)
  - valeur par défaut : à la connexion d'un client, par défaut l'état du client est défini comme inactif.

- Le principe de passage à état actif ou à l'état inactif pour un client :
  - C'est le client qui informe le serveur de sa demande de passage à état actif ou inactif.

- L'identification d'un client :
  - Un client sera identifié par un numéro interne qui est un numéro unique, incrémenté par ordre d'arrivée des demandes de connexion, et dont la gestion à l'aide d'une base de données ainsi que la gestion d'une limite max ne seront pas abordées dans ce projet.
  - Le client reçoit du serveur son identification, (numéro client) , qui sera à indiquer dans les messages envoyés (messages de changement d'état)
  - identifiant client : de type entier (de 1 à 255) (voir sur `send_state, send_num` plus loin le formalisme de codage dans les messages)

- L'affichage de la liste des clients connectés :
  - C'est un service qui peut être effectué en trace en continu par le serveur suivant un paramètre de démarrage du serveur et qui peut aussi être demandé par un programme externe.
  - Il y a donc deux services d'affichage réalisés par le serveur :
    - affichage en mode trace en continu paramétrable au lancement du serveur.
    - affichage à la demande : soit une liste, soit pour un client donné
    - L'affichage est simple de type de celui de la commande 'ps' :

      N° ETAT

      … .

- le protocole de communication entre le serveur et le client est le suivant :

  - Les messages du client vers le serveur :
    - Le principe fonctionnel des messages est le suivant :
      - `connexion` : permet d'établir la connexion,
      - `déconnexion` : permet de rompre la connexion,
      - `send_state` : permet de faire connaitre au serveur son état

  - Le message d' envoi de l'état (`send_state`) étant le seul type de message que le client peut envoyer, il ne sera pas nécessaire de spécifier un type de message : seul sont envoyés l'état le numéro de client. Le codage des valeurs entières sera fait en mode caractère dans ce message de deux caractères :
    - 1er caractère : numéro de client : '1' à '255'
    - 2ème caractère : état : '0' = inactif, '1' = actif

- Les messages du serveur vers le client :
  - Le principe fonctionnel des messages est le suivant :
    - `send_num` : permet de confirmer la connexion, et de renvoyer l'identifiant client. le message comprend un caractère :
- Le message d'envoi de l'état (`send_num`) étant le seul type de message que le serveur peut envoyer, il ne sera pas nécessaire de spécifier un type de message : seul est envoyé le numéro de client. Le codage des valeurs entières sera fait en mode caractère dans ce message d'un caractère :
  - 1er caractère : numéro de client : '1' à '255'

- Les messages seront envoyés par un protocole de communication de type socket.

- le protocole de communication entre le serveur et un processus d'affichage est le suivant :

- Les messages du processus d'affichage vers le serveur :
  - Le principe fonctionnel des messages est le suivant :
    - `get_list` : permet de demander la liste des clients
    - `get_state` : permet de demander l'état d'un client

  - Les messages du processus d'affichage vers le serveur étant au nombre de deux mais de type similaire, la restriction en volume donnant un codage de numéro de client sur un caractère possible, et un client étant par définition connecté, il n'est donc pas nécessaire de spécifier un type de message. Le format des messages est donc le suivant:
    - 1er caractère : numéro de client : '0' à '255' ('0' = demande pour la liste de tous les clients)

- Les messages du serveur vers le processus d'affichage:
  - Le principe fonctionnel des messages est le suivant :
    - `send_list` : permet de renvoyer la liste des clients
    - `send_state` : permet de renvoyer l'état d'un client

  - Les messages du processus d'affichage vers le serveur étant au nombre de deux mais de type similaires, et la restriction en volume donnant un codage de numéro de client sur un caractère possible, et un client étant par définition connecté il n'est donc pas nécessaire de spécifier un type de message. Le format des messages est donc le suivant:
    - `send_list` : permet de renvoyer la liste des clients
      - 1er caractère : numéro de client : '0' ('0' = réponse pour la liste de tous les clients)
      - caractères suivant : suite de deux caractères (numéro de client, état) correspondants au format de message `send_state`

      - NB : le protocole est simple, les volumes de données, faibles, dans votre choix d'implémentation, il vous sera facile de trouver un moyen d'échange d'information entre ces deux processus n'excédant pas 512 octets 255*2 +1. Le protocole n'a donc pas nécessité de spécifier des début de liste, suite de liste et fin de liste.

    - `send_state` : permet de renvoyer la l'état d'un client

      - 1er caractère : numéro de client : '1' à '255'
      - 2ème caractère : état : '0' = inactif, '1' = actif

Les étapes d'implémentation et le calendrier :

- Des étapes d'implémentations peuvent être suivies pour la réalisation du projet et permettre d'une part un découpage par lot et d'autre part une validation intermédiaire en termes de fonctionnalités (une présentation de 75% des fonctionnalités sera à effectuer la semaine du 9 juillet 2015, la soutenance étant la semaine du 20 juillet 2015). Les fonctionnalités techniques détaillées servant de base à ce découpage, seront données en annexe à ce syllabus projet.

Les choix d'implémentation :

Les choix d'implémentation sont libres, et sont multiples. Ils sont à justifier en fonctions des contraintes et données du projet. Des IPC (mémoire partagée, tubes, ..), des protocoles de communisation (de type socket (UPD/TCP)) et de technologies de gestion de thread et ou de processus peuvent être utilisées pour le projet.

Base de connaissance

Livres

Programmation système en C sous Linux

Signaux, processus, threads, IPC et sockets

Auteur : Christophe Blaess
Editeur : Eyrolles, 3eme edition

Références

POSIX : http://www.gnu.org/software/libc/manual/html_node/POSIX.html

Volume de travail estimé : 10h

Travail en groupe : de 2 à 4 personnes

Historique du document

- Création du projet initial : le 17 avril 2015
- Réduction du projet en version simplifiée : le 5 juin 2015
- audience : publique