Projet

Objectifs

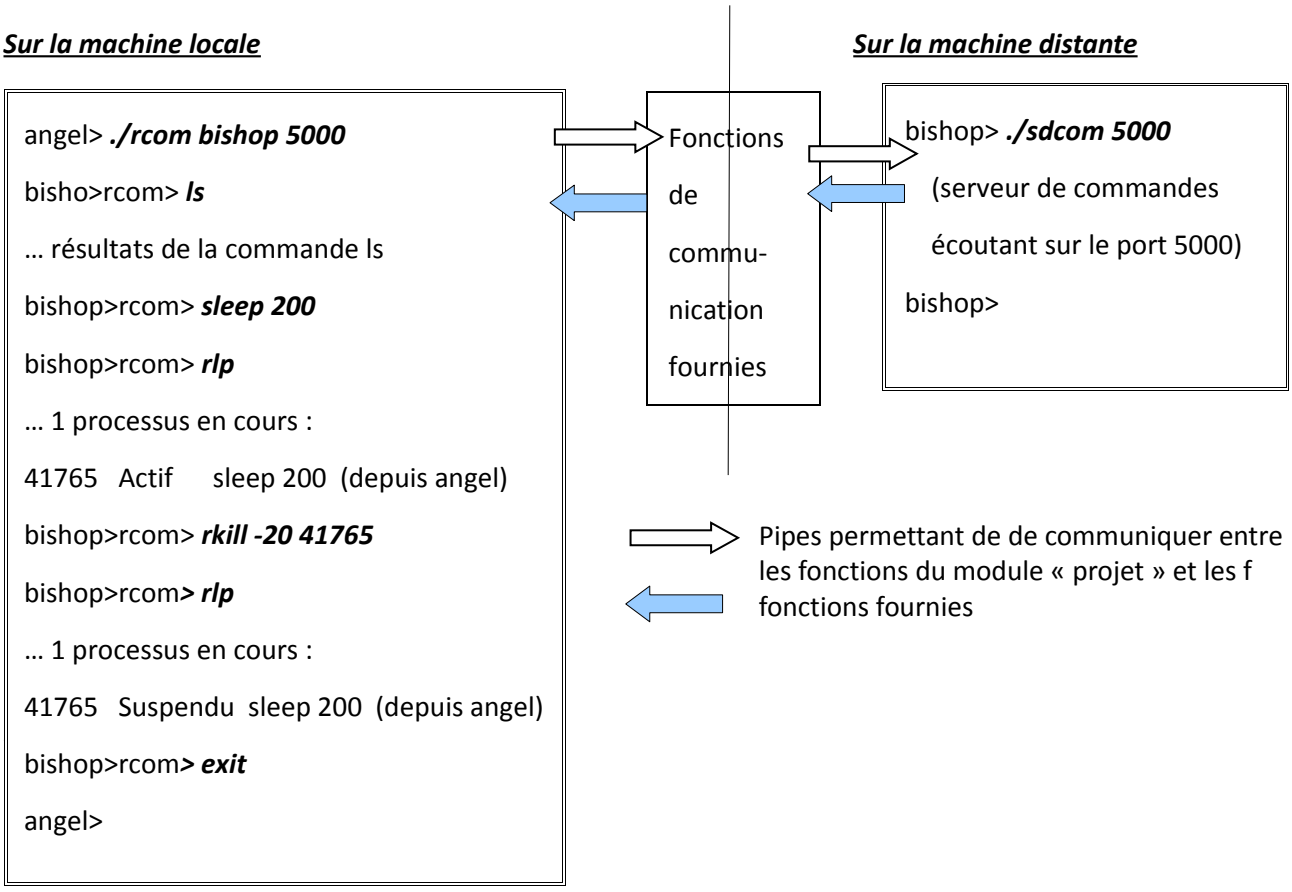
1. Développer une application permettant de mettre en œuvre les principaux concepts systèmes étudiés en TD et en TP : gestion des processus, synchronisation, et communication entre processus.
2. Approfondir la programmation en langage C : manipulation de structures chaînées, programmation modulaire.

Méthodologie

1. Le projet sera découpé en étapes fortement liées aux sujets des TD et des TP,
2. Chaque étape est abordée lors du TP concerné, et est complétée, par l'étudiant, en dehors des séances de TP,
3. La régularité et la progressivité du travail constituent un élément clé du projet : certains TP, liés à des aspects importants du projet, seront notés (investissement, avancement, qualité), et comptent pour 1/3 dans la note du projet (voir calendrier sur moodle).
4. La qualité doit être privilégiée. Il vaut mieux faire 75% du projet en respectant les règles de programmation et de présentation, plutôt que de chercher à faire 100% du projet avec un code et des tests de mauvaise qualité.

Sujet

Le projet consiste à développer, progressivement, un shell à distance permettant de :
- lancer des processus sur d'autres ordinateurs du réseau et récupérer leurs résultats,
- visualiser leurs caractéristiques : nom, pid, état, ...
- leur envoyer des signaux : suspension, relance, arrêt,



1. La commande ***sdcom num_port*** lance un processus permanent qui :
   - lance un serveur à l'écoute de toute communication entrante sur le port num_port (fonction scok_server fournie)
   - analyse, dans une boucle infinie, toute demande entrante et exécute la commande correspondante :
     * toute commande externe (shell, exécutable), tout en l'enregistrant dans la liste des processus en cours
     * rlp : envoie au demandeur la liste des commandes en cours : nom, pid, état, machine source de la commande
     * kill -num pid : envoie le signal indiqué au processus (s'il est dans la liste) et actualise son état

2. La commande ***rcom nom_machine numport*** :
   - ouvre une communication (sock_client fournie) avec la machine indiquée en argument sur le port num_port (il faut qu'une session sdcom num_port soit déjà ouverte),
   - boucle en lisant :
     * toute commande lue au clavier et l'envoie à sdcom sur la machine distante (seule la commande exit est interprétée localement pour arrêter le processus rcom)
     * tout message ou résultat provenant de l'application sdcom

   Ceci nécessite une écoute non bloquante de l'ensemble des descripteurs (entrée standard et pipes de communication avec sock_client). Les pipes seront vus en TD et TP7, et le mécanisme d'écoute non bloquante sera vu en TD et TP8.

3. L'application ***sdcom*** gère une liste des commandes en cours, en mettant à jour leurs caractéristiques et leur état, et doit donc implanter les traitements permettant de récupérer les changements d'état des fils. La commande rlp permet d'afficher la liste des commandes et leurs caractéristiques.

Etapes

Le poids de chaque étape dans le projet est indiqué entre (), avec un total =10.
Chaque étape est liée aux concepts manipulés en TP de la semaine précédente.

Etape1 (poids = 2) : Ecrire et tester le noyau de l'application qui :
- Lit une commande au clavier (fonction de lecture d'une commande simple fournie)
- Exécute les actions correspondantes :
  1. « exit » : arrêt de l'application
  2. « commande » [arguments] : lance l'exécution de la « commande »

La fonction strcasecmp (char *s1, char *s2) renvoie 0 si les deux chaines s1 et s2 sont égales, sans tenir compte de la casse. Le module cmds.h présente les fonctions de saisie, et principalement la fonction lire_cmd qui lit une ligne de commande, la découpe en différents champs, et renvoie des pointeurs sur ces derniers dans le tableau passé en paramètre.

Ecrire et tester le code qui permet au processus principal de prendre en compte, ***de façon non bloquante***, les changements d'état de ses fils (suspension, reprise, arrêt, …), d'en rendre compte à l'écran.

Etape2 (poids = 2) : Terminer l'implémentation du module « liste_proc », qui permet, en manipulant un type abstrait « fiche_proc » de gérer l'ensemble des processus en cours, avec les fonctions suivantes :

- inserer_fin : insère un nouveau processus en fin de liste
- appartient : vérifie si un processus appartient à la liste
- lire_etat_proc : renvoie l'état du processus passé en paramètre
- modifier_etat_proc : modifie l'état du processus passé en paramètre
- afficher_liste_proc : affiche les processus de la liste et leurs caractéristiques : numéro, état (ACTIF,

SUSPENDU), nom de la commande, (et plus tard, nom utilisateur et nom machine source)
- supprimer_proc : supprime le processus portant le numéro « numproc » de la liste
- detruire_liste : détruit la liste

- Compléter l'étape 1 avec l'enregistrement des caractéristiques des processus dans la liste

- Ajouter la commande « rlp » qui permet d'afficher l'ensemble des processus en cours
- Ajouter la commande kill permettant d'envoyer un signal à un processus de la liste, avec modification de l'état consécutive au signal.

Les parties 1 et 2 sont à rendre pour le jeudi 31 mars 2OH :
- archive envoyée par mail à l'enseignant de TP (Nom_ProjetP1.tar)
- contenant le code commenté, ainsi qu'une copie d'écran d'un test représentatif.

Etape3 (poids = 2) : Deux fonctions de communication sont fournies :

- « sock_server » contenant des fonctions de lancement et d'arrêt du serveur de communication par socket.
  * Le serveur utilise un numéro de port, qui doit être assez grand – par exemple 7000- pour ne pas entrer en conflit avec d'autres applications utilisant des ports d'entrées/sorties,
  * Il peut accepter plusieurs communication avec des clients différents,
  * Il communique avec le processus qui l'appelle avec des pipes crées par ce dernier.

- « sock_client » permettant de communiquer avec « sock_server » sur une machine distante.

Transformer le code des étapes 1 et 2, pour créer l'application sdcom.
Compléter avec le programme rcom.

Etape4 (poids = 2) : Les différents messages (saisis au clavier, reçus depuis le serveur de communication, résultats de commande, etc.) pouvant se mélanger, il devient utile d'organiser cela avec une écoute non bloquante des différentes entrées. Compléter les codes de sdcom et de rcom par le mécanisme qui permet de réaliser cette écoute non bloquante sur l'entrée clavier et les communications entrantes.

Etape5 (2 points) : ***au choix.*** Ajouter une autre fonctionnalité, au choix, au code précédent :
- exécution d'une commande avec redirection (« > ») des résultats vers un fichier . Quand ils sont présents, « < » et « > » sont détectés par la fonction « lire_cmd » et renvoyés dans le champ pointé par le argv[] correspondant.
- ou enregistrement de l'historique des commandes dans un fichier
- ou une toute autre fonctionnalité personnelle (limitation des connexions à des utilisateurs autorisés, ...)

Livraison : Semaine du 11/5 – mardi 10 mai 18H

- Rapport : un document simple doit présenter l'architecture de l'application (modules, leur contenu, et leurs interconnexions), le mode opératoire et la liste des commandes implantées.
- Les modules bien commentés (surtout l'entête des fonctions).
- une copie d'écran d'un jeu de test représentatif

- Une archive portant le nom « SEC16_projet_groupe_nometudiant »

Test du projet : Semaine du 16/5 – jour du TP

- Présence obligatoire – ordre de passage par tirage au sort.

Tout commentaire sur le sujet et sa présentation peut être envoyé par mail à hamrouni@n7.fr

Les explications techniques peuvent être demandées à l'enseignant de TP.