CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

METHODES DE PROGRAMMATION SYSTEMES

UE NSY103

NANCY – METZ

PROJET

Année 2011 – 2012, deuxième semestre

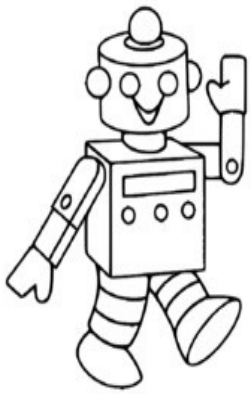
Coefficient : 1/3

Travail à réaliser au plus tard avant l'examen de juin 2012, et à remettre en version électronique :

- au surveillant au début de l'examen final (avant la distribution du sujet),
- ou à envoyer avant l'examen à E. Desvigne : emmanuel@desvigne.org

Mise à jour d'un robot-jouet

La société « *Blanchet* » a commercialisé un jouet du nom de « *Robby* », ayant l'apparence d'un robot :



Ce jouet communiquant via wifi permettait à ses propriétaires de programmer :

- la rotation des bras gauches et droits,
- l'allumage de trois LEDs situées sur le torse,
- et la diffusion de phrases exprimées via un synthétiseur vocal.

Pour ce, après avoir inclus une ligne « `#include "api_robby.h"` », le programmeur avait accès aux fonctions suivantes de l'API de télécommande de Robby :

- `int etat_LED1()` : retourne 0 si la led n°1 est éteinte, 1 si elle est allumée ;
- `int etat_LED2()` : retourne 0 si la led n°2 est éteinte, 1 si elle est allumée ;
- `int etat_LED3()` : retourne 0 si la led n°3 est éteinte, 1 si elle est allumée ;
- `void allume_LED1()` : allume la LED n°1 si elle ne l'est pas déjà ;
- `void allume_LED2()` : allume la LED n°2 si elle ne l'est pas déjà ;
- `void allume_LED3()` : allume la LED n°3 si elle ne l'est pas déjà ;
- `void eteint_LED1()` : éteint si nécessaire la LED n°1 ;
- `void eteint_LED2()` : éteint si nécessaire la LED n°2 ;
- `void eteint_LED3()` : éteint si nécessaire la LED n°3 ;
- `int etat_bras_gauche()` : donne l'angle (en degré) de rotation du bras gauche (la valeur de retour est ainsi un entier compris entre 0 et 359) ;
- `int etat_bras_droit()` : donne l'angle (en degré) de rotation du bras droit (la valeur de retour est ainsi un entier compris entre 0 et 359) ;
- `void rotate_bras_gauche(int a)` : positionne le bras gauche à un angle de « `a` » degrés (le paramètre « `a` » doit être un entier compris entre 0 et 359, sinon, la fonction ne fait rien). Cette fonction est bloquante et ne se termine que lorsque le bras a la position vouloue ;
- `void rotate_bras_droit(int a)` : positionne le bras droit à un angle de « `a` » degrés (le paramètre « `a` » doit être un entier compris entre 0 et 359, sinon, la fonction ne fait rien). Cette fonction est bloquante et ne se termine que lorsque le bras a la position vouloue ;
- `void dicte(char *s)` : utilise la synthèse vocale pour prononcer la phrase contenue dans la chaîne de caractères « `s` ». Cette fonction est bloquante et ne rend la main à la fonction appelante que lorsque toute la phrase est dictée par le robot.

Ce jouet a eu un succès commercial non négligeable, mais au dire de ses utilisateurs, il souffrait d'un inconvénient majeur : il n'était pas multitâche. Autrement-dit, il ne pouvait exécuter qu'un et un seul programme à la fois, toutes les fonctions systèmes liées à la multiprogrammation ayant été bridées.

Deux ans après la naissance de Robby, la société Blanchet a été rachetée par la société « *Open Up* ». L'idée du repreneur est de sortir « *Robby version 2.0* », ce nouveau robot devant ne pas avoir l'inconvénient « monotâche » de son prédécesseur.

Dans les laboratoires d'Open Up, une version spéciale de Robby est utilisée, dans laquelle les fonctions fork() et pthread_create() ne sont pas bridées. Après quelques jeux d'essais, il apparaît que :

- si deux programmes qui tournent en parallèle cherchent à faire tourner le même bras en même temps, le robot plante. Même plantage général si deux programmes cherchent en même temps à parler, ou à changer l'état d'une même LED ;
- par contre, un programme peut très bien demander au robot de dicter une phrase pendant qu'un autre programme demande de faire tourner un bras, un autre programme faisant tourner l'autre bras, et que 3 autres programmes titillent l'allumage des 3 LEDs, sans que le système soit instable. Autrement-dit, tous les organes du robot peuvent être commandés en parallèle par un et un seul programme ;
- enfin, l'appel aux fonctions `etat_xxx()` peut être effectué en tout instant sans perturber le système.

Suite aux résultats de ces tests, le staff d'Open Up décide de commercialiser « Robby version 2.0 », une version multiprogrammée de Robby. Dans cette version :

- les fonctions de l'ancienne version de Robby ne seront plus accessibles, et les primitives liées à la gestion des processus et des threads restent bridées pour les acquéreurs du jouet ;
- par contre, une fonction :
  `int v2_nouveau_programme(void (*nouv_prog)(void))`
  est mise à la disposition des utilisateurs pour autoriser plusieurs programmes à tourner en parallèle. Ainsi, lorsqu'un utilisateur voudra qu'une fonction `mon_prog()` soit lancée en parallèle de ses autres fonctions déjà lancées, il fera un appel à :
  `v2_nouveau_programme(mon_prog);`
  Un maximum de 10 programmes utilisateurs pourront tourner en simultané. Si l'utilisateur cherche à en lancer un onzième, la fonction `v2_nouveau_programme()` retournera une valeur nulle (alors qu'elle retourne un entier non nul quand tout se passe bien) ;
- caque programme utilisateur mettra fin à son exécution par l'appel à la fonction :
  `void v2_fin_programme(void).`

Ainsi, typiquement, voici ce qu'un utilisateur de « Robby version 2.0 » aura à programmer pour lancer deux programmes en parallèle :

```
/**********************/
/* début de l'exemple */
/**********************/

/* définition des nouvelles fonctions de l'API version 2 : */
#include "v2_api_robby.h"

void mon_programme_1()
{
    /* bla bla bla ...  (utilisation de fonctions en v2_xxx() ) */
    v2_fin_programme();
}
```

```
void mon_programme_2()
{
    /* bla bla bla ...  (utilisation de fonctions en v2_xxx() ) */
    v2_fin_programme();
}

int main()
{
    /* bla bla bla */
    if (v2_nouveau_programme(mon_programme_1) == 0)
    {
        /* traitement de l'erreur */
    }
    if (v2_nouveau_programme(mon_programme_2) == 0)
    {
        /* traitement de l'erreur */
    }
    /* bla bla bla */
    exit(0) ;
}
/********************/
/* fin de l'exemple */
/********************/
```

Une nouvelle API version 2 est ainsi proposée aux utilisateurs. Toutes les fonctions de cette API commencent par le préfixe « `v2_` ». En plus des fonctions `v2_nouveau_programme()` et `v2_fin_programme()` déjà décrites, cette API contiendra les fonctions suivantes :

- `int v2_etat_LED(int n)` retournera 0 si la LED numéro « `n` » est éteinte, 1 sinon ;
- `void v2_eclaire_LED(int n, int e)` éteindra la LED numéro « `n` » si « `e` » vaut 0, et l'allumera si « `e` » est différent de 0 ;
- `int v2_etat_bras(int b)` retourne une valeur entière comprise entre 0 et 359, qui exprime l'angle de rotation actuel (en degré) du bras droit si « `b` » vaut 0, et l'angle de rotation (toujours en degré) du bras gauche si « `b` » différent de 0 ;
- `void v2_rotate(int b, int a)` forcera le robot à positionner son bras droit (si « `b` » vaut 0) ou gauche (si « `b` » non nul) selon un angle de « `a` » degrés (« `a` » devant être compris entre 0 et 359) ;
- `void v2_dicte(char *s)` : utilise la synthèse vocale pour prononcer la phrase contenue dans la chaîne de caractères « `s` ».

La dictée vocale et la rotation des bras restent des fonctions bloquantes. De plus, si plusieurs programmes cherchent simultanément à utiliser le même organe, les ordres seront mis en file d'attente, et seront exécutés les uns après les autres (débloquant ainsi au fur et à mesure les programme qui se trouvaient bloqués).

Enfin, dans cette API version 2, de nouvelles fonctionnalités voient le jour :

- des nouvelles fonctions de l'API doivent permettre de connaître l'état du système. Elles permettront par exemple de savoir combien de programmes utilisateurs ont déjà été lancés depuis le démarrage du jouet, et combien de programmes utilisateurs tournent actuellement en parallèle ;
- de plus, l'API doit contenir des fonctions et mécanisme permettant à deux programmes de communiquer. Le staff d'Open Up reste flou sur cet aspect, laissant toute liberté aux programmeurs de l'API pour proposer les outils qu'ils veulent afin de résoudre ce besoin de communication.

Couverture fonctionnelle du travail à réaliser par les auditeurs de l'UE NSY103

Vous êtes le programmeur système qui a à charge de développer l'API version 2 de « Robby version 2.0 ». Pour ce, vous avez à votre disposition l'ensemble de toutes les primitives de Linux étudiées dans l'UE NSY 103, ainsi que l'accès à toutes les fonctions de l'API version 1.

Vous devez par conséquent implémenter les fonctions `v2_nouveau_programme()`, `v2_fin_programme()`, `v2_etat_LED()`, `v2_eclaire_LED()`, `v2_etat_bras()`, `v2_rotate()`, et `v2_dicte()`.

De plus, vous avez à proposer des fonctions permettant de donner à l'utilisateur l'état du système, et permettant à deux programmes de communiquer. Vous avez une totale liberté concernant ces deux derniers points. Toutefois, votre choix devra être argumenté, les fonctions devront être documentées, et l'utilisateur devra connaître les limites inhérentes à ces choix.

Barème

La moitié de la note sera évaluée sur la description et l'argumentaire du choix de l'architecture du système cible. C'est pourquoi dans son rapport, le candidat décrira l'architecture finale du système, en précisant (et argumentant le choix) des mécanismes qu'il met en œuvre pour résoudre chaque problématique de communication, de multitâche, d'accès aux données partagées, etc.

L'autre moitié de la note sera attribuée sur l'évaluation des programmes constituant l'API version 2 proposée par le candidat. Ces programmes seront être écrits en pseudo code ou en langage C.

Bonus : si le candidat fournit du code en C opérationnel (et non un pseudo code), un « bonus » lui sera attribué. Ce bonus pourra rattraper une éventuelle moyenne tangente à 10 après l'examen.

Modalités de remise du projet

Le projet devra être impérativement remis sous forme électronique (sur CD-Rom ou par e-mail) AVANT l'examen final du NSY103. Dans le pire des cas, un CD-ROM pourra être remis à la personne qui surveille l'examen AVANT la distribution des sujets. Le candidat signera alors une feuille d'émargement. Sinon, l'ensemble du projet pourra être envoyé par email à l'adresse emmanuel@desvigne.org avant l'examen. Chaque réception d'e-mail sera suivie d'accusé de réception de la part du correcteur, indiquant que le projet a bien été reçu.

Le rapport du projet (obligatoire) pourra être rédigé dans un document en texte brut (ASCII ou UTF8), ou en RTF, ou au format MS-Word doc 1997-2003, ou préférentiellement au format Oasis Open Document (format odt). Les algorithmes pourront faire partie de ce document, ou être mis dans des fichiers texte séparés. Enfin, l'éventuel code source en C sera fourni dans des fichiers séparés ayant une extension ".c", ".h", etc.
L'ensemble de tous ces documents pourra être intégré dans un seul fichier (archive 7Z, ZIP, RAR, .tar.gz, ou .tar.bz2).