Sujet du TP n° 1

Préambule

Les ressources du cours (supports, exemples et tout autre document) et les TP (sujets, fichiers de travail ainsi que les corrigés) se trouvent à l'URL: http://www.emse.fr/~lalevee/ismin/pse, qui sera dénommée **[site]** dans tous les documents fournis.

Environnement de développement

L'environnement de travail pour le cours "programmation des systèmes d'exploitation" est le suivant :

- Utilisation obligatoire du système Linux
- Édition conseillée avec **emacs**, **gedit**, etc. (ne pas utiliser **eclipse**)
- Compilation avec **gcc -Wall -o programme_executable programme.c ...**
- Outils : **make**, **gdb**, etc.

Nous vous proposons d'organiser votre répertoire de travail selon l'archive **PSE.tar**.

- Récupérez cette archive sur **PSE.tar** (elle est accessible sur le **[site]** en sélectionnant la page **Séances**)
- Ouvrez une fenêtre "Terminal", et positionnez-vous dans votre répertoire **Bureau** (ou **Desktop**)
- Vous pouvez choisir un autre lieu où extraire l'archive, qu'il faudra conserver par la suite
- Extrayez l'archive avec la commande **tar xvf PSE.tar**
- Un nouveau répertoire, **PSE**, a été créé par cette commande. Nous le nommerons **[pse]** dans les documents futurs. Il sera votre espace de travail pour les TP et votre projet

Vous trouverez dans cet espace **[pse]** :

- Un répertoire **include** contenant les fichiers **.h** des modules utilisés dans les TP. Ces fichiers contiennent la documentation des fonctions qu'ils offrent. Pour inclure tous les fichiers **.h** fournis dans les sujets des TP dans vos programmes, il vous suffira d'inclure le fichier **pse.h**.
- Un répertoire **modules** qui contient :
    - ♦ les fichiers **.c** des modules utilisés dans les TP, qui seront placés dans une bibliothèque ;
    - ♦ un fichier **Makefile** qui permet de créer la bibliothèque **libpse.a** dans le répertoire **lib**, voir le contenu de ce répertoire ci-dessous ;
    - ♦ les modules présent sont :
        - ◊ **erreur**, qui permet une gestion (simple) des affichages de messages d'erreur.
        - ◊ **resolv**, qui permet d'effectuer des résolutions DNS
        - ◊ **ligne**, qui permet de lire ou d'écrire des lignes de textes à partir de fichiers (de tout type ou presque)
        - ◊ **msg** et **msgbox**, qui permettent de gérer des messages et les files de messages
        - ◊ **datathread**, qui permet de gérer les données privées des threads
- Un répertoire **lib** dans lequel vous trouverez (après l'avoir générée) la bibliothèque de modules **libpse.a**, utilisée dans les TP. Le fichier **Makefile.inc** présent dans **[pse]** fait le nécessaire pour lier vos applications à cette bibliothèque.
- Un répertoire **exemples** dans lequel se trouvent les exemples complets des transparents du cours. Les fichiers correspondants ont un nom sous la forme **slide999.c**, où **999** est le numéro du transparent.
- Un répertoire **projet** pour l'instant vide...

- Les répertoires **TP1** à **TP8** contenant les fichiers nécessaires pour les TP. Chaque répertoire contient :
  - ♦ Un fichier **pdf**, copie du sujet du TP.
  - ♦ Un fichier **Makefile**, qui vous servira pour générer les programmes exécutables demandés dans le TP. Il utilise le fichier **Makefile.inc** décrit précédemment.
  - ♦ Selon les TP, des fichiers à compléter ou à utiliser.

Les corrigés de chaque séance sont fournis dans des archives séparées, appelées **TP1-C.tar** à **TP8-C.tar**, disponibles dès la fin du TP correspondant. Elles sont accessibles sur le **[site]** en sélectionnant la page **Séances**).

À vous de faire vivre votre espace de travail...

Placez-vous dans le répertoire **PSE/TP1**

Arguments de la fonction main (transparent 31)

La fonction **main()** peut recevoir des arguments transmis lors du démarrage du programme, en général par le *shell*. Nous utiliserons très souvent ce mécanisme pour transmettre des valeurs aux programmes de ce cours, aussi faites ces exercices consciencieusement. La fonction **main()** a la déclaration suivante en langage C :

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

- **argc** : nombre d'arguments présents (taille du tableau **argv**)
- **argv** : tableau de **argc** arguments, sous forme de chaîne de caractères

Ainsi la commande : **echo hello world** (voir transparent 31) donnera la valeur **3** à **argc** et initialisera **argv** avec les **3** chaînes de caractères **"echo"**, **"hello"** et **"world"**.
Note. Comme **argv[0]** contient toujours le nom de la commande, **argc** est toujours supérieur ou égal à 1.

Pratique 1

- placez-vous dans le répertoire **modules** et tapez **make** pour générer la bibliothèque (cette opération devra être refaite à chaque modification des modules de cette bibliothèque).
- ensuite, copiez le fichier **exemples/slide031.c** dans **TP1**, en lui donnant le nom **exercice1.c**.
- tapez **make** pour générer l'exécutable et testez-le.

Exercice 1

modifiez le fichier **exercice1.c**, de telle sorte qu'il affiche ses arguments dans la fenêtre Terminal.

Chaque programme dispose de chaînes d'environnement, c'est-à-dire d'un ensemble de variables d'environnement, sous la forme **VARIABLE=VALEUR** permettant au programme de s'adapter à un environnement particulier. Ces variables sont transmises par le *shell* qui en fait un usage important, à partir d'une variable globale **environ** déclarée de cette façon :

```
extern char ** environ;
```

Comme le nombre de variables d'environnement n'est pas connu, nous nous servirons du fait que le tableau **environ** soit terminé par un pointeur **NULL** pour le parcourir.

Exercice 2.

- affichez les variables d'environnement du programme.

- comme vous avez certainement utilisé un accès par tableau dans l'exercice 1, vous utiliserez exclusivement un accès par pointeur dans cet exercice, cela pour se remettre en condition.
note: ajoutez l'exécutable **exercice2** dans la variable **EXE** du fichier **Makefile**.

Résolution DNS (transparents 32 à 37)

Nous allons utiliser dans cette partie un module de bibliothèque, qui est placé dans le répertoire **modules**. Il réalise une résolution DNS : à partir d'un nom de domaine (par exemple **www.emse.fr**), il retournera son adresse IP. Mais avant cela, nous allons expérimenter les types et les fonctions nécessaires.

Pratique 2

- retrouvez le fichier **<netdb.h>** et cherchez la déclaration du type **struct addrinfo**.
- de même, retrouvez le fichier **<netinet/in.h>** et cherchez la déclaration du type **struct sockaddr_in**.

Pratique 3

- copiez le fichier **exemples/slide037.c** dans **TP1**, en le nommant **exercice3.c**
- vérifiez qu'il compile correctement.

Exercice 3

- affichez en format hexadécimal le contenu des champs **sin_addr.s_addr** et **sin_port**.
rappel : l'affichage en hexadécimal d'un **int** se fait avec **"%X"** et celui d'un **short** se fait avec **"%hX"**.
- que constatez-vous ?
- corrigez en utilisant les fonctions de conversion de *Network Order* vers *Host Order*.
- modifiez ce programme afin d'offrir les valeurs de **host** et de **service** avec les arguments de la fonction **main()**.
- par exemple : `./exercice4 www.google.com https`

Au retour de la fonction **getaddrinfo()**, **infos** contient l'adresse du premier élément d'une liste chaînée simple. Les éléments suivants de la liste sont pointés par le champ **ai_next**, jusqu'à la valeur **NULL** indiquant la fin de la liste.

Exercice 4.

- modifiez le programme précédent de manière à afficher toutes les adresses IP correspondant à un nom de domaine (vous placerez le paramètre **service** à **NULL**).
- utilisez **www.google.com**, qui en possède plusieurs.
- pour afficher en clair une adresse IP, vous pouvez utiliser la fonction **inet_ntop()** ; pour cela, consulter le manuel.

Pour terminer, nous allons utiliser un module de bibliothèque pour la résolution DNS, qui s'appelle **resolv**.

Exercice 5. Programmez en utilisant le module **resolv** :

- consultez et utilisez le fichier **resolv.h** qui est déjà dans le répertoire **include**
- consultez **modules**, le fichier **resolv.c** fourni, en particulier la fonction **resolv()**
- tapez **make**, si cela n'a pas été fait auparavant
- dans **TP1**, écrivez un programme qui fait une résolution DNS (domaine et service passés en paramètre de la fonction **main()**) et qui affiche l'adresse IP et le numéro de port, en utilisant le module **resolv**.