Séance 1
Arguments ligne de commande, Résolution DNS

Programmation système

Appel système

Arguments de la ligne de commande

Résolution DNS

Conversion des entiers réseau<->hôte

Programmation système

Programmation système : appels de fonctions demandant au système d'exploitation des services de base

- gestion de la mémoire,
- gestion des périphériques,
- processus (création, communication, synchronisation),
- programmation réseau,
- E/S fichiers,
- etc

Cours ISMIN :

- Programmation système sous Linux en C
- Pré requis : langage C, les commandes de base Linux

Appel système

Système d'exploitation : tout les logiciels livrés avec la machine; Linux, Windows, …

Noyau : logiciel qui tourne an arrière-plan, qui gère la mémoire, les périphériques, les processus, le réseau, etc.

Mode utilisateur et mode noyau :
- mode utilisateur : seulement accès à la mémoire utilisateur,
- mode noyau : accès à tout l'espace mémoire, accès au hardware.

Appel système :
- requête au noyau pour accomplir une action de base (ex. créer processus),
- mis en œuvre par appel à une fonction C.

Fonction C réalisant un appel système :
1. passage en mode noyau,
2. réalisation de l'action,
3. retour en mode utilisateur.

Arguments de la ligne de commande

Exemple : application add faisant la somme de deux nombres passés sur la ligne de commande.

```
> ./add  34  526                    commande add + deux arguments
```

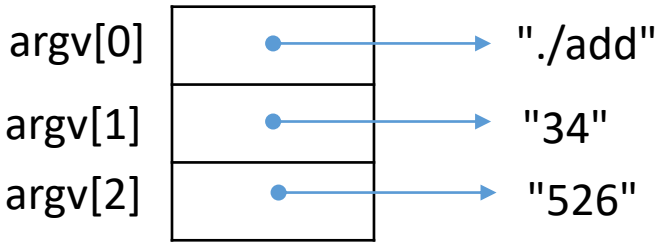
Pour récupérer les arguments dans le programme :

```
// add.c

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

argc : nb d'arguments (y compris la commande)
=> 3

argv : tableau de pointeurs sur des chaines représentant les arguments



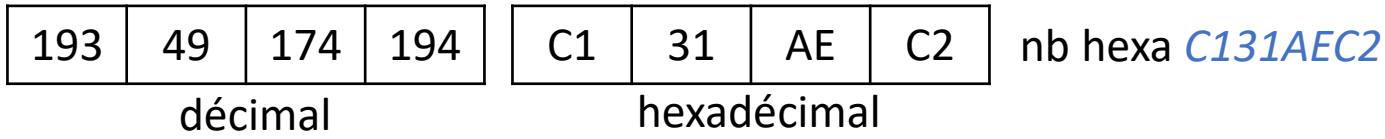
Résolution DNS

But : convertir un nom d'hôte ("www.emse.fr") et un nom de service ("http") en adresse (193.49.174.194) et port (80).

Fonction getaddrinfo :

- rend le résultat dans une structure de type *struct sockaddr_in*
- l'adresse est dans un entier 32 bits (champ *sin_addr.s_addr*)
- le port est dans un d'un entier 16 bits (champ *sin_port*)

dans la structure, l'adresse est codée en mémoire "poids forts d'abord"



nb hexa *C131AEC2*

certaines machines (ex. PC) codent "poids faibles d'abord"

interprètent comme nb hexa *C2AE31C1*

idem pour le port (16 bits)

=> nécessité de convertir adresses et ports

Conversion des entiers réseau<->hôte

Conversion ordre du **réseau** vers ordre de la machine **hôte** :

uint32_t **ntohl** (uint32_t val)	pour entiers long (32 bits)
uint16_t **ntohs** (uint16_t val)	pour entiers courts (16 bits)

Conversion ordre de la machine **hôte** vers ordre du **réseau** :

uint32_t **htonl** (uint32_t val)	pour entiers longs (32 bits)
uint16_t **htons** (uint16_t val)	pour entiers courts (16 bits)