TP n°1 : Gestion de la mémoire

Exercice 1 :

1. Exécutez la commande : **cat /proc/meminfo**
   /proc/meminfo est un fichier qui fournit des informations détaillées sur l'utilisation de la mémoire du système, des informations telles que la quantité totale de mémoire disponible, la quantité de mémoire libre, la quantité de mémoire utilisée par différents types de processus et d'autres informations utiles sur la mémoire système peuvent etre obtenu
2. Exécutez la commande : **free**
   free permet de fournir des informations sur l'utilisation de la mémoire système. Lorsqu'elle est exécutée sans aucun argument, elle affiche la quantité totale de mémoire physique, utilisée, libre, partagée entre plusieurs processus, utilisée pour les caches du noyau, ainsi la quantité totale et utilisée du swap, **free** utilise **/proc/meminfo**
   pour obtenir un nouvel affichage toutes les 5 secondes : **watch -n 5 free**
3. Pour afficher sur une seule ligne la quantité de mémoire disponible sur le système : **grep MemAvailable /proc/meminfo | awk '{print $2}'**
4. Exécutez la commande : **vmstat**
   vmstat fournit des informations sur la mémoire virtuelle, la mémoire du noyau, les processus en cours d'exécution, les entrées/sorties, la charge du système etc
5. Exécutez la commande : **cat /proc/<pid>/status,** que permet-elle d'obtenir ? (Utilisez la commande top pour choisir un processus)
6. Executez **pmap <PID>** ensuite **cat /proc/<PID>/maps**
   Les commande **pmap** et **/proc/<PID>/maps** permettent d'obtenir des informations sur la carte mémoire du processus spécifié par PID qui montrent les differentes régions du processus

Remarque : la mémoire virtuelle d'un processus est constituée d'un ensemble de régions. Une région est une portion contiguë de la mémoire virtuelle d'un processus. A chaque région le système associe des protections et un rôle particulier.

Exercice 2 : Compiler le programme suivant en utilisant -static : **gcc -static -o exo2 exo2.c**
Remarque : **-static** est utilisé pour lier statiquement les bibliothèques lors de la création de l'exécutable.
Recompiler le programme sans utiliser **-static** et expliquez la différence

```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
int une_globale;
int une_autre=4;
char* alloc;
int main()
{int une_locale;
alloc = malloc(1024L * 1024L * 10L); /* 10mo */
printf("PID = %d\n", getpid());
printf("adresse de une_globale = %8lx\n", (unsigned long) &
une_globale);
printf("adresse de une_autre_globale = %8lx\n", (unsigned long) & une_autre);
printf("adresse de une_locale = %8lx\n", (unsigned long) & une_locale);
printf("adresse de alloc = %8lx\n", (unsigned long) alloc);
printf("adresse de main = %8lx\n", (unsigned long) & main);
printf("adresse de printf = %8lx\n", (unsigned long) & printf);
/* afficher la carte mémoire */
sprintf(alloc, "cat /proc/%d/maps", getpid());
system(alloc);
return 0; }
```

L'exécution de ce programme provoque l'affichage d'un PID et d'une série d'adresses. Dans le système Linux, les utilisateurs sont capables d'obtenir beaucoup d'informations sur les processus. Ces informations se trouvent dans le répertoire : **/proc/<PID>**
On y trouve notamment le fichier **maps** qui donne la liste des régions associées à ce processus. Pour chaque région nous trouvons son espace adressable, les protections, l'offset (le décalage), le numéro du périphérique et un numéro de i-node.
Exécutez la commande : **ls -i <nom fichier>** pour afficher le numéro de l'i-node associé au fichier

Exercice 3 : Compiler et exécutez le programme suivant
l'appel system **sysconf** permet de récupérer divers paramètres système en utilisant différentes constantes prédéfinies

```
#include<stdio.h>
#include<unistd.h>
int main ()
{
long max_files = sysconf(_SC_OPEN_MAX);
long page_size = sysconf(_SC_PAGESIZE);
long page_physique = sysconf(_SC_PHYS_PAGES);
long page_physique_dispo = sysconf(_SC_AVPHYS_PAGES);
printf("MAX. fichiers ouverts %ld \n",max_files);
printf("Taille page %ld octets \n",page_size);
printf("Nb. pages(cadres) physiques totales %ld \n",page_physique);
printf("Nb. pages(cadres) physique disponible %ld \n",page_physique_dispo);
return 0;}
```