TP N°04 : Gestion de la Mémoire

Remarque : créez un répertoire réservé à vos exercices de programmation

Exercice 4.1 (La mémoire virtuelle des processus)

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. Le but de cet exercice est de visualiser et de comprendre ces régions.

Q1) Commencez par compiler (avec l'option -static) le programme suivant :

```
#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;
}
```

Q2) En comparant l'espace adressable de chaque région et les adresses données par le programme, donnez un sens à chaque région.

Q3) Même exercice mais en compilant le programme sans la directive **-static**.

Remarque : le système UNIX associe un numéro à chaque fichier sur disque. Ce numéro (appelé le numéro d'i-node) peut-être visualisé en utilisant l'option **-i** de la commande **ls**

Exercice 4.2 (Segments de processus)

Q1) Commencez par compiler le programme suivant :

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int global;
int global_init = 999;

static int global_static;

void f() {
  printf("oups\n");
}

int main(int argc, char *argv[]) {
int local;
int *p = malloc(10);
static int local_static;

printf("%18p Global              \n", &global);
printf("%18p Global static        \n", &global_static);
printf("%18p Local static         \n", &local_static);
printf("%18p Global init          \n", &global_init);
printf("%18p String literal       \n", "bonjour");
printf("%18p Function (f)         \n", &f);
printf("%18p Function (main)      \n", &main);
printf("%18p Local               \n", &local);
printf("%18p Argc                \n", &argc);
printf("%18p Argv                \n", argv);
printf("%18p Alloc               \n", p);
printf("%18p Function (lib)       \n", &printf);

free(p);
p = (void *) - 1;
printf("%18p Adr max             \n", p);

if (argc == 2)
        sleep(600);
}
```

Q2) Triez les adresses par ordre croissant et décroissant

Q3) Voir la carte mémoire de ce processus.