TP n°5 : Gestion de la mémoire dans Linux

1 La mémoire logique des processus

Le mémoire logique d'un processus est constituée d'un ensemble de *régions*. Une région est une portion contiguë de la mémoire logique. À chaque région le système associe des protections et un rôle particulier. Le but de cet exercice est de visualiser et de comprendre ces régions. Commencez par compiler (en mode `-static`) le programme suivant :

```c
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>

int une_globale;
const char* une_constante = "une-chaine";

/*****************************************************
 Afficher la carte mémoire du processus
 fichier /proc/self/maps
 *****************************************************/
void dump_maps(void) {
    FILE *f = fopen("/proc/self/maps", "r");
    assert(f != NULL);
    for(int c; ((c = fgetc(f)) != EOF); ) {
        putchar(c);
    }
    fclose(f);
    printf("Tapez␣RETURN␣>>␣");
    getchar();
}


/*****************************************************
 Afficher les adresses des variables locales sur
 trois niveaux de récursion.
 *****************************************************/
void print_local(int n) {
    int une_locale;
    printf("adresse␣de␣une_locale/%d␣=␣%012lx\n", n, (unsigned long) & une_locale);
    if (n < 4) print_local(n+1);
}


/*****************************************************
 Afficher les adresses de différents objets mémoire
 *****************************************************/
int main(int argc, char* argv[]) {
    static long une_statique = 1;

    char *alloc1 = malloc(1024L * 1024L * 10L); /* 10mo */
    printf("PID␣=␣%d\n", getpid());
    printf("adresse␣de␣une_globale␣␣␣=␣%012lx\n", (unsigned long) & une_globale);
    printf("adresse␣de␣une_statique␣=␣%012lx\n", (unsigned long) & une_statique);
    print_local(0);
    printf("adresse␣de␣alloc1␣␣␣␣␣␣␣=␣%012lx\n", (unsigned long) alloc1);
    printf("adresse␣de␣une_fonction␣=␣%012lx\n", (unsigned long) & main);
    printf("adresse␣d'une␣constante␣=␣%012lx\n", (unsigned long) une_constante);

    /* afficher la carte mémoire */
    dump_maps();

    /* allouer de la mémoire en plus */
    char *alloc2 = malloc(1024L * 1024L * 100L); /* 100mo */
    printf("adresse␣de␣alloc2␣␣␣␣␣␣␣=␣%012lx\n", (unsigned long) alloc2);
    dump_maps();

    return (EXIT_SUCCESS);
}
```

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_du_processus`

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 (*majeur* : *mineur*) et un numéro de i-node. On trouve également dans ce répertoire le fichier `statm` qui donne des statistiques sur l'utilisation de la mémoire (`man~proc` pour avoir plus de précisions).

⏩ Travail à faire :

- En comparant l'espace adressable de chaque région et les adresses données par le programme, donnez un sens à chaque région (allez sur ce site pour avoir des informations sur VDSO).
- 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 de i-node) peut-être visualisé en utilisant l'option `-i` de la commande `ls`.
- Si vous réalisez plusieurs allocations dynamiques suivies de leirs libérations, quel est le résultat sur la carte mémoire ?

2 Protection de la mémoire avec UNIX

ℹ Note : Commencez par étudier l'appel système `mprotect`. Utilisez cette fonction pour protéger une portion mémoire d'un de vos programme.

⏩ Travail à faire : La démarche est la suivante :

1. Écrivez un programme qui alloue un tableau de 16 kilo-octets (fonction `malloc`) et qui initialise ce tableau.
2. Modifiez votre programme pour protéger en écriture (puis en lecture) une portion centrale de votre tableau. Quel est le résultat ? Tentez sur votre tableau une opération interdite.

   Remarques :

   - Vous aurez sûrement besoin d'aligner un pointeur sur le début d'une page : je vous conseille d'utiliser la macro ci-dessous et la fonction `sysconf` pour récupérer la taille des pages (paramètre `_SC_PAGESIZE`).

     ```
     #define ALIGN_MIN(p,s) ((void*)( (((unsigned long)(p)) / (s)) * (s) ))
     ```

   - Vous pouvez également utiliser la fonction `aligned_alloc` pour réaliser directement des allocations alignées. Si vous effectuez ensuite des décalages de `PAGESIZE`, vous obtiendrez une adresse alignée.
3. Affichez la carte mémoire avant la protection et après. Quel est l'effet de la protection sur les régions du processus ?