Noyau d'un système d'exploitation INF2610

Chapitre 7 : Gestion de la mémoire (annexe 2)

Département de génie informatique et génie logiciel

POLYTECHNIQUE MONTRÉAL

AFFILIÉE À L'UNIVERSITÉ DE MONTRÉAL

Hiver 2014

Exemple 1 : Espace virtuel d'un processus

Où seront stockées les données des variables déclarées (cas de C) ?

// http://ilay.org/yann/articles/mem/mem0.html (à lire avant le prochain lab) //

memoirevirtuelle.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
static int i_stat = 4; /* Stocké dans le segment data */
int  i_glob;          /* Stocké dans le segment bss  */
int *pi_pg;           /* Stocké dans le segment bss  */
int main(int nargs, char **args) {
int *pi_loc;      /* dans la frame 1 de la pile */
void *sbrk0 =(void *) sbrk(0);  /*l'adresse de base avant le 1er malloc */
if (!(pi_loc = (int *) malloc(sizeof(int) * 16)))
          return 1;
if (!(pi_pg  = (int *) malloc(sizeof(int) *  8)))
     {    free(pi_loc);
          return 2;
     }
```

Exemple 1 : Espace virtuel d'un processus
Où seront stockées les données des variables déclarées ?

```
// afficher les adresses
printf("adresse de i_stat = 0x%08x (zone programme, segment data)\n", &i_stat);

printf("adresse de i_glob = 0x%08x (zone programme, segment bss)\n",  &i_glob);

printf("adresse de pi_pg  = 0x%08x (zone programme, segment bss)\n",  &pi_pg);

printf("adresse de main   = 0x%08x (zone programme, segment text)\n", main);

printf("adresse de nargs  = 0x%08x (pile frame 1)\n", &nargs);

printf("adresse de args   = 0x%08x (pile frame 1)\n", &args);

printf("adresse de pi_loc = 0x%08x (pile frame 1)\n", &pi_loc);

printf("sbrk(0) (heap)    = 0x%08x (tas)\n", sbrk0);

printf("pi_loc            = 0x%08x (tas)\n", pi_loc);

printf("pi_pg             = 0x%08x (tas)\n", pi_pg);
```

Exemple 1 : Espace virtuel d'un processus

Où seront stockées les données des variables déclarées ?

```
//  récupérer le contenu /proc/pid/maps du processus
    char buf[128];
    printf("Affichage du fichier /proc/%d/maps\n",getpid());
    sprintf(buf,"/proc/%d/maps",getpid());
    int fd1 = open(buf,O_RDONLY);
    while (read(fd1,buf,128)>0) write(1, buf,128);
    write(1, "\n",2);
    close(fd1);
    free(pi_pg);
    free(pi_loc);
    return 0;
}
```

Exemple 1 : Espace virtuel d'un processus
Où seront stockées les données des variables déclarées ?

```
jupiter$ g++ memoirevirtuelle.cpp -o memoirevirtuelle

jupiter$ ./memoirevirtuelle

adresse de i_stat = 0x006012d4 (zone programme, segment data)

adresse de i_glob = 0x006012e0 (zone programme, segment bss)

adresse de pi_pg  = 0x006012e8 (zone programme, segment bss)

adresse de main   = 0x0040096c (zone programme, segment text)

adresse de nargs  = 0xfd058cac (pile frame 1)

adresse de args   = 0xfd058ca0 (pile frame 1)

adresse de pi_loc = 0xfd058d38 (pile frame 1)

sbrk(0) (heap)    = 0x00c24000 (tas)

pi_loc            = 0x00c24010 (tas)

pi_pg             = 0x00c24060 (tas)
```

Exemple 1 : Espace virtuel d'un processus
Où seront stockées les données des variables déclarées ?

Affichage du fichier /proc/2850/maps

00400000-00402000 r-xp 00000000 00:26 125178476 memoirevirtuelle

00601000-00602000 rw-p 00001000 00:26 125178476 memoirevirtuelle

00c24000-00c45000 rw-p 00000000 00:00 0 [heap]

….

Les champs sont : adresses (deb et fin), permissions (p pour privé et Copy-On-Write),

offset, périph, i-nœud, chemin d'accès (pour localiser le texte et les données initialisées)

Lancez une seconde fois le programme.
Est-ce que vous obtenez les mêmes adresses ?

Exemple 2 : Exécution d'un programme (évolution de la pile d'exécution)

(Livre sur les SE de Bort LAMIROY, Lourent NAJMAN, , Hugues TALBOT)

Le code assembleur dépend de l'architecture. La commande g++ -S prog.cpp produit le code dans prog.s.

Trois paramètres suffisent pour caractériser l'état d'exécution d'un programme. Ces paramètres sont mémorisés dans des registres. Par exemple, pour Intel x86, ces paramètres sont :

- %cs contient l'adresse de base(le premier mot mémoire) du programme.
- %eip est le compteur ordinal. Il pointe en permanence sur la prochaine instruction à exécuter. Il est initialisé à l'adresse de la fonction main.
- %esp est le pointeur de pile. Il évolue dynamiquement avec l'utilisation de la pile, au fur et à mesure des allocations de données intermédiaires (données locales, les paramètres d'une fonction, etc.). On utilise souvent un autre registre %ebp (Extended Base Pointer) pour y stocker à chaque appel à une fonction, l'adresse du sommet de la pile. À la fin de l'exécution de la fonction, le contenu de %ebp est copié dans %esp.

Exemple 2 : Exécution d'un programme (évolution de la pile d'exécution)

(Livre sur les SE de Bort LAMIROY, Lourent NAJMAN, , Hugues TALBOT)

```
int addition(int a) {return a+5; }
static int arg1= 5;
static int arg2= 1;
int main() {
        int param= arg1 + arg2;
        arg2 = addition(param);
}
```

Instructions ass.	Fonction addition
pushl %ebp	Sauvegarder le contenu de %ebp (avant l'exécution de la fonction) dans la pile.
movl %esp, %ebp	Mettre à jour %ebp
movl 8(%ebp), %eax	Charger dans %eax la valeur du paramètre a (qui se trouve à 8 octets de de l'adresse contenu dans %ebp).
addl $5 %eax	Ajouter de 5 à %eax
popl %ebp	Récupérer dans %ebp, le contenu de %ebp sauvegardé avant l'exécution de la fonction.
ret	Retour à l'appelant (dépiler %eip, …)

Exemple 2 : Exécution d'un programme (évolution de la pile d'exécution)

(Livre sur les SE de Bort LAMIROY, Lourent NAJMAN, , Hugues TALBOT)

```
int addition(int a)
 {   return a+5; }
static int arg1= 5;
static int arg2= 1;
int main() {
    int param= arg1 + arg2;
    arg2 = addition(param);
}
```

Instructions ass.	Fonction main
pushl %ebp	Sauvegarder le contenu de %ebp (avant l'exécution de la fonction) dans la pile.
movl %esp, %ebp	Mettre à jour %ebp
subl $4, %esp	Réserver 4 octets pour param. %esp contient l'adresse de param ((%ebp)-4)
movl arg1, %edx	%edx = arg1
movl arg2, %eax	%eax = arg2
leal (%edx,%eax), %eax	%eax= %edx + %eax
movl %eax, -4(%ebp)	param = %eax
pushl -4(%ebp)	Empiler param comme argument
call addition	Appeler la fonction addition (empiler %eip puis aller au début de la fonction)
addl $4, %esp	Mettre à jour %esp (dépiler l'argument)
movl %eax, argv2	arg2 = %eax
leave	Restorer %ebp et %esp.
ret	Retour à l'appelant (dépiler %eip,…)

Exemple 3 : Alignement de la mémoire

(Livre sur les SE de Bort LAMIROY, Lourent NAJMAN, , Hugues TALBOT)

```
typedef struct { double valeur; short int index; char code; char id;} S1;
typedef struct {char id; short int index; double valeur; char code;} S2;
#include <stdio.h>
#include <stdlib.h>
int main()
{  S1* s1 = (S1*) malloc(sizeof(S1));
   S2* s2 = (S2*) malloc(sizeof(S2));
   char * a_S1 = (char*) s1;
   char* a_S2 = (char*) s2;
   printf("Taille d'un double = %lu\n", sizeof(double));
   printf("Taille d'un entier court = %lu\n", sizeof(short int));
   printf ("Taille d'un caractère = %lu \n", sizeof(char));

   printf("\n Taille de S1 (double,short,char,char)= %lu\n", sizeof(S1));
   printf("Champ \t Offset \t Taille \n");
   printf("1 \t %ld \t %lu\n", ((char*)(&(s1->valeur))) - a_S1,sizeof(double));
   printf("2 \t %ld \t %lu\n", ((char*)(&(s1->index))) - a_S1,sizeof(short int));
   printf("3 \t %ld \t %lu\n", ((char*)(&(s1->code))) - a_S1,sizeof(char));
   printf("4 \t %ld \t %lu\n", ((char*)(&(s1->id))) - a_S1,sizeof(char));
```

Exemple 3 : Alignement de la mémoire

(Livre sur les SE de Bort LAMIROY, Lourent NAJMAN, , Hugues TALBOT)

```
   printf("\n Taille de S2 (char,short,double,char)= %lu\n", sizeof(S2));
   printf("Champ \t Offset \t Taille \n");
   printf("1 \t %ld \t %lu\n", ((char*)(&(s2->id))) - a_S2,sizeof(char));
   printf("2 \t %ld \t %lu\n", ((char*)(&(s2->index))) - a_S2,sizeof(short int));
   printf("3 \t %ld \t %lu\n", ((char*)(&(s2->valeur))) - a_S2,sizeof(double));
   printf("4 \t %ld \t %lu\n", ((char*)(&(s2->code))) - a_S2,sizeof(char));
 return 0;
}
```

```
// cas Mac OS X 10.8.5
jupiter$ ./alignement
Taille d'un double = 8
Taille d'un entier court = 2
Taille d'un caractère = 1
```

char	aucun
short int	adresse paire
double	Adresse multiple de 8
struct	Adresse multiple de 8

```
Taille de S1 (double,short,char,char)= 16
Champ   Offset    Taille
1    0      8
2    8      2
3    10     1
4   11      1
```

```
Taille de S2 (char,short,double,char)= 24
Champ   Offset    Taille
1      0     1
2      2     2
3      8     8
4      16    1
```

Exemple 4 : Parcours d'un tableau multidimensionnel

```
#include <stdio.h>
#include <stdlib.h>
int main()
{  char A[512][4096];
   int i,j;
   for(i=0; i<512; i++)
     for(j=0; j<4096; j++)
               A[i][j] = 'A';
   return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>
int main()
{  char A[512][4096];
   int i,j;
   for(j=0; j<4096; j++)
     for(i=0; i<512; i++)
               A[i][j] = 'A';
   return 0;
}
```

```
jupiter$ time ./parcours_Lig_Lig
real 0m0.017s
user0m0.010s
sys  0m0.003s

jupiter$ time ./parcours_Col_Col
real 0m0.035s
user0m0.030s
sys  0m0.003s
```