Projet AISE

BELKACEM Sarah AILAM Melisa

Fevrier 2020

1 Introduction

2 Implementation de l'alocateur

Pour implémenter notre allocateur on a suivie les étapes suivantes :
1-Nous avons creer un fichier.h nomme lib.h qui comprend les prototypes des 4 fonctions principales d'allocation memoire c'est a dire : malloc,free,calloc,realloc.
2-Nous avons créer un fichier.c nomme lib.c qui comprend les corps des 4 fonctions comme suit :

2.1 La fonction malloc

La fonction malloc permet d'allouer de la mémoire dynamiquement. Dans ce cas nous allons utliser la focntion Mmap()

```
void * malloc(size_t size){
int *plen;
int len = size + sizeof( size );
plen = mmap( 0,len, ROT_READ| ROT_WRITE, MAP_PRIVATE MAP_ANONYMOUS, 0,0 );
*plen = len;
return (void*)(&plen[1]);
}
```

Listing 1 – La fonction malloc

2.2 La fonction free

La fonction Free permet de libérer de la memoire. Dans ce cas nous allons utliser la focntion Munmap()

```
void free(void *ptr){
int *p = (int*)ptr;
int a;
p--;
a = *p; //lire la taille
munmap( (void*)p, a);
}
```

Listing 2 – La fonction Free

2.3 La fonction calloc

La fonction Calloc a la meme fonction que malloc, mais elle initialise à 0 tous les éléments de la zone mémoire.

```
void * calloc(size_t n_element, size_t element_size){
void *p;
p = malloc (n_element * element_size);
if (p == 0)
return (p);
bzero (p, n_element * element_size);
return (p);}
```

Listing 3 – La fonction Calloc

2.4 La fonction realloc

Cette fonction permet de réallouer un bloc de mémoire dans le tas (le heap en anglais). Cela veut dire que si l'espace mémoire libre qui suit le bloc à réallouer est suffisament grand, le bloc de mémoire est simplement agrandi. Par contre si l'espace libre n'est pas suffisant, un nouveau bloc de mémoire sera alloué, le contenu de la zone d'origine recopié dans la nouvelle zone et le bloc mémoire d'origine sera libéré automatiquement.

```
void *realloc(void *ptr, size_t size){
if (ptr == NULL) {
return malloc(size);}
 else if (size == 0){
free(ptr);
return NULL;}
```

Listing 4 – La fonction Realloc

3 Compilation

3.1 Interception des fonctionsf depuis une bibliothèque

gcc lib.c -fpic -shared -o libtest.so

3.2 Chargement de la bibliotheque

LDLIBRARYPATH=. ./a.out

4 Prechargement de l'allocateur

Dans ce cas on a utiliser le preload : LDPRELOAD=.libtest.so ./a.out

5 Mesure de performances et Comparaison avec les allocateurs systemes

5.1 La fonction malloc et free du systeme

Pour un tableau de 10 elements le temps d'excution : 4.000000 ms
Pour un tableau de 100 elements le temps d'excution : 5.00000 ms
Pour un tableau de 100000 elements le temps d'excution : 21.00000 ms
Pour un tableau de 10000000 elements le temps d'excution : 31.00000 ms

5.2 Notre fonction malloc et free

Pour un tableau de 10 elements le temps d'excution : 21.0000 ms
Pour un tableau de 100 elements le temps d'excution : 31.000ms
Pour un tableau de 100000 elements le temps d'excution : 34.000 ms
Pour un tableau de 10000000 elements le temps d'excution : 38.000ms

5.3 La fonction calloc et free du systeme

Pour un tableau de 10 elements le temps d'excution : 112.000ms
Pour un tableau de 100 elements le temps d'excution : 120.00ms
Pour un tableau de 100000 elements le temps d'excution : 174.00 ms
Pour un tableau de 10000000 elements le temps d'excution : 1741.00 ms

5.4 Notre fonction calloc et free

Pour un tableau de 10 elements le temps d'excution : 1375.00ms
Pour un tableau de 100 elements le temps d'excution : 2053.00ms
Pour un tableau de 100000 elements le temps d'excution : 2130.00ms
Pour un tableau de 10000000 elements le temps d'excution : 2276.00

6 Version Parallele

Pour paralleliser le code nous avons opter pour une fonction qui ceer des threads donc ona creer un code avec des threads.

```
void *thread_1(void *arg)
{
    printf("Nous sommes dans le thread.\n");

    /* Pour enlever le warning */
    (void) arg;
    pthread_exit(NULL);
}
```

Listing 5 – La fonction qui creee des threads

7 Conclusion et perspectives

Ce travail nous a permis de mettre en oeuvre tous les acquis du module aise, on a pu creer une bibliotheque qu'on a intercepter et utliser ses fonctions.Pour les perspectives ont compte mieux gerer les blocs memoires en utilisant les listes chainees.