Système d'Exploitation
Travaux Pratiques (5), Licence 2 Informatique
Gestion Mémoire
Remplacement de Pages

Pré-requis : lire le mode d'emploi de la `libmem` en annexe.

Depuis la VM Linux, recopiez le fichier `Remplacement_Pages.zip` dans votre répertoire personnel et décompactez-le (ce fichier est disponible sur Celene sur la page du cours à la section « Outils pour les Travaux Pratiques » sous le nom « Remplacement de Pages »).

```
$ unzip Remplacement_Pages.zip
```

Note : la bibliothèque `libmem` est incluse dans le fichier `Remplacement_Pages.zip`.

Dans la suite vous devez compléter les fichiers `Fifo.c` et `LRU.c`, présents dans le répertoire `Remplacement_Pages/TP`, en vous inspirant des exemples `LFU` et `Random` du mode d'emploi que vous trouverez dans le répertoire `Remplacement_Pages/libmem/algorithms`.

1. Écriture d'une Stratégie de Remplacement FIFO

1.1

Complétez le fichier `Fifo.c`

1.2

Compilez et exécutez votre programme en lui donnant en entrée la suite de références contenu dans le fichier `bench`.

```
$ make mainFifo
$ ./mainFifo < bench
```

2. Écriture d'une Stratégie de Remplacement LRU

2.1

Complétez le fichier `LRU.c`.

2.2

Compilez et exécutez votre programme en lui donnant en entrée la suite de références contenue dans le fichier `bench`.

```
$ make mainLRU
$ ./mainLRU < bench
```

3. Comparaison LRU / FIFO

Comparez les résultats obtenus. Vérifiez en exécutant la suite de références "à la main" que vos deux algorithmes implémentent bien respectivement les stratégies `LRU` et `FIFO`.

Annexe
libmem : Mode d'Emploi

La bibliothèque `libmem` permet de tester des algorithmes de remplacement de pages.
L'arborescence est la suivante :

```
libmem
|-- Makefile
|-- README
|-- algorithms
|   |-- LFU.c
|   |-- Random.c
|   `-- main.c
|-- bench
|-- bin
|-- include
|   |-- LFU.h
|   |-- Random.h
|   |-- Swapper.h
|   `-- libmem.h
|-- lib
|-- obj
`-- src
    |-- Swapper.c
    `-- libmem.c
```

Le répertoire `src` contient les sources de la bibliothèque.
Le répertoire `include` contient les fichiers d'en-tête de `libmem` (`libmem.h`, `Swapper.h`) et les fichiers propres aux exemples (`LFU.h`).

Le répertoire `algorithms` contient un exemple de main (`main.c`) ainsi que des exemples d'algorithmes de remplacement de pages :

- `LFU.c` implémente l'algorithme Least Frequently Used,
- `Ramdon.c` implémente un algorithme qui choisit une page aléatoirement.

1. Fonctions de la Librairie

Remarque : dans le code de la `libmem`, la terminologie anglaise est utilisée : `frame` = case en mémoire physique, `page` = page en mémoire virtuelle.

Structure de données

La `libmem` maintient une structure de données "`Swapper`" définie dans `Swapper.h` :

```
typedef int Page;
```

```
typedef struct Swapper {
…
      unsigned int frame_nb;     /* Nombre de cases de la mémoire physique */
      Page *     frame;          /* Tableau des cases en mémoire */
      void *     private_data;   /* Donnée privée propre à chaque stratégie */
} Swapper;
```

Le champ `frame` indique pour chaque case la page correspondante :
Pour la case `i`, si `i` est libre `frame[i] = -1`, sinon `frame[i]` = numéro de la page.

Fonctions

`libmem` fournit deux fonctions :

```
#include "Swapper.h"

int initSwapper(
      Swapper* swap,
      unsigned int nc,                        //Logical number of frames
      int (*Init)(Swapper*),                  //Init for private data
      void (*Reference)(Swapper*,unsigned int),  //Reference to keep stats
      unsigned int (*Choose)(Swapper*),       //Choose function
      void (*Finalize)(Swapper*)              //Finalize function
);
```

`initSwapper` permet d'initialiser la structure `swap` avec `nc` cases. 4 fonctions sont passées en paramètre. `Init` sera appelée à l'initialisation, `Reference` à chaque accès à une case, `Choose` à chaque défaut de page et `Finalize` à la terminaison.

Pour programmer une stratégie de remplacement de pages appelée par exemple "`MaStrategie`", il faut donc :

1. Programmer les 4 fonctions suivantes :

```
int initMaStrategie(Swapper *swap);
```

Ou la variable `swap` est celle initialisée par la fonction `initSwapper`.
Cette fonction permet éventuellement d'allouer et d'initialiser le champ `private_data` de la structure `swap`.
La fonction doit retourner 0 en cas de succès.

```
void referenceMaStrategie(Swapper *swap, unsigned int frame);
```

Fonction qui sera appelée lors d'un accès à la case mémoire numéro `frame`.

```
unsigned int chooseMaStrategie(Swapper *swap);
```

Fonction qui sera appelée lors d'un défaut de page. Elle doit retourner le numéro de la case contenant la page victime.

```
Void finalizeMaStrategie(Swapper *swap);
```

Fonction appelée à la fin du programme pour libérer ce qui a été allouée par `initMaStratégie`.

2. Appeler `initSwapper` de la manière suivante :

```
Swapper *s,
initSwapper(&s, nbcases, initMaStrategie, refenceMaStrategie,
            chooseMaStrategie, finalizeMaStrategie);
```

```
#include "libmem.h"

int   swapSimulation(Swapper *swap,FILE *f);
```

Lance la simulation des accès aux pages. `swap` doit être préalablement initialisé par `initSwapper`. Le fichier `f` contient un entier par ligne : la première ligne indique le nombre de cases de la mémoire physique et les autres lignes, la suite de références mémoire.

2. Exemple

L'exemple suivant illustre l'utilisation des primitives. On souhaite implémenter une stratégie de remplacement de page de type `LFU` (Least Frequently Used). Pour cela il faut avoir un tableau qui compte le nombre de la référence à une case. En cas de remplacement de page, il suffit de choisir la case ayant le compteur avec une valeur minimum.

`include/LFU.h` :

```
#include "Swapper.h"

int initLFUSwapper(Swapper*,unsigned int);
```

`algorithms/LFU.c` :

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

int  initLFU(Swapper*);
void referenceLFU(Swapper*,unsigned int frame);
unsigned int chooseLFU(Swapper*);
void finalizeLFU(Swapper*);

int initLFUSwapper(Swapper*swap,unsigned int frames){
    initSwapper(swap,frames,initLFU,referenceLFU,chooseLFU,finalizeLFU);
}
int  initLFU(Swapper*swap){

     /* Allouer un tableau avec un compteur d'utilisation pour
        chaque case */
```

```
	swap->private_data = calloc(swap->frame_nb,sizeof(int));
	int * use = (int*)swap->private_data;
	int i;

	/* Initialisation du tableau */

	for( i=0 ; i<swap->frame_nb ; i++ )
		use[i] = 0;
	return 0;
}

void referenceLFU(Swapper*swap,unsigned int frame){
	int i;
	int * use = (int*)swap->private_data;

	/* A chaque acces à la case frame augmente son compteur d'utilisation */
	use[frame]++;
}

unsigned int chooseLFU(Swapper*swap){
	int i, frame = 0;
	int * use = swap->private_data;

	/* Choisir la case (contenant une page) ayant le plus petit compteur */

	for ( i=0 ; i<swap->frame_nb ; i++ ){
		if( swap->frame[i] == -1 ){
			frame = i;
			break;
		}
		if( use[i] < use[frame] )
			frame = i;
	}

	use[frame] = 0;

	return frame;
}

void finalizeLFU(Swapper*swap){
	free(swap->private_data);
}
```

algorithms/main.c :

```
#include "libmem.h"
#include "LFU.h"

int main(int argc,char*argv[]){
	unsigned int frame_nb;
	Swapper    s;
```

```
	scanf("%i",&frame_nb);

	/* Initialisation du Swapper de la stratégie LFU */
	initLFUSwapper(&s,frame_nb);

	/* Lancer la simulation */
	if(swapSimulation(&s,stdin)<0){
		printf("Error during swap simulation !!!\n");
		return -1;
	}

	return 0;
}
```

3. Utilisation

La `libmem` est disponible sur Celene dans la page du cours à la section « Outils pour les Travaux Pratiques » sous le nom « Remplacement de Pages ».

Tout d'abord, copiez la `libmem` dans un répertoire personnel :

Le sous-répertoire `libmem/algorithms` contient l'exemple précédent ainsi qu'une élection aléatoire (`Random.c`). Dans `libmem` un fichier `makefile` permet de générer directement l'exécutable et la bibliothèque en exécutant les commandes suivantes :

```
$ cd libmem
$ make
```

Pour créer une nouvelle stratégie (`MaStrategie`), il faut : créer un fichier `MaStragie.h` dans le répertoire `include`, un fichier `MaStratgie.c` dans `algoritms` et modifier le fichier `main.c` dans `algorithms`.

Le fichier `libmem/bench` contient un exemple de suite de référence mémoire pour 3 cases. Pour lancer la `libmem` sur cet exemple il faut entrer :

```
$ bin/main < bench
Page 1 referenced
LFU uses:

/!\ PAGE FAULT !!! /!\
Frame 0 has been choosen
(frame 0: 1) (frame 1: _) (frame 2: _)

Page 2 referenced
LFU uses: (page:1 time:1)

/!\ PAGE FAULT !!! /!\
Frame 1 has been choosen
(frame 0: 1) (frame 1: 2) (frame 2: _)
```

```
Page 3 referenced
LFU uses: (page:1 time:1) (page:2 time:1)

/!\ PAGE FAULT !!! /!\
Frame 2 has been choosen
(frame 0: 1) (frame 1: 2) (frame 2: 3)

Page 4 referenced
LFU uses: (page:1 time:0) (page:2 time:1) (page:3 time:1)

/!\ PAGE FAULT !!! /!\
Frame 0 has been choosen
(frame 0: 4) (frame 1: 2) (frame 2: 3)

Page 3 referenced
(frame 0: 4) (frame 1: 2) (frame 2: 3)

Page 4 referenced
(frame 0: 4) (frame 1: 2) (frame 2: 3)

Page 4 referenced
(frame 0: 4) (frame 1: 2) (frame 2: 3)

4/7 ~ 57.142857%
```

Par défaut la `libmem` est en mode “verbeux”, pour désactiver le mode, il faut commenter dans le `makefile` la ligne :

```
CFLAGS=-D_DEBUG_
```

4. Pour Installer la Librairie chez soi

La librairie est disponible dans le fichier compacté « Remplacement_Pages.zip » sur Celene dans la page du cours à la section « Outils pour les Travaux Pratiques ».

Pour décompacter le fichier, il suffit de taper sur votre machine :

```
$ unzip Remplacement_Pages.zip
```

Un répertoire `libmem` est créé. Il suffit de compiler la librairie et exécuter l’exemple :

```
$ cd libmem
$ make
$ bin/main < bench
```

Merci à Mathieu Valéro pour avoir développé la libmem
Pour tout commentaire, ou rapport de “ bug ” : `Pierre.Sens@lip6.fr`