HEIG VD | REDS Institut Reconfigurable and Embedded Digital Systems

Systèmes d'exploitation (SYE)

Profs Daniel Rossier, Alexandre Corbaz, Fiorenzo Gamba
Assistants : Anthony Jaccard, Basile Cuneo, Jean-Pierre Miceli

Algorithmes de remplacement de page

lab13 (Semaine du 08.01.2024)

Objectifs de laboratoire

Ce laboratoire porte sur les différents algorithmes de remplacement de page dans la mémoire physique lorsque nécessaire. Le but ici est d'implémenter l'algorithme LRU (*Least Recently Used*) ainsi qu'une version simplifiée de *WSClock* dans un simulateur permettant de générer des accès sur des pages mémoires

Récupération et rendu du laboratoire

Pour récupérer la branche de ce laboratoire, utilisez les commandes:

1. "*git fetch upstream*"
2. "*git checkout labXX"*

Ce laboratoire sera à rendre selon la méthode vue lors du précédent laboratoire, à savoir

1. Ajouter les fichiers modifiés avec "*git add <vos_fichiers>"*
2. Commiter les changements avec "*git commit –m "<message de commit>"*
3. Envoyer les changements sur votre repo gitlab avec "*git push –u origin"* ou "*git push —set-upstream origin"*

Etape 1 – Implémentation de l'algorithme LRU

Cette étape consiste à implémenter l'algorithme LRU « classique ». La configuration de la mémoire est la suivante : **3 pages physiques** et **16 pages virtuelles**.

Pour cela, le fichier « ***usr/host/memreplace.c*** » déclare une table de page gérant 16 pages virtuelles (variable globale « ***page_table*** ») et initialise au démarrage du programme la table des pages avec les pages 0, 1 et 2 en RAM et la 3 en « *swap* ». La fonction « *main()* » implémente déjà une base pour l'étape 2 et 3 ; pour cette étape, il s'agit de considérer le code lorsque la condition LRU est « vraie ».

a) Créer un tableau **TDU** (Temps de Dernière Utilisation) intégrant la notion de compteur pour chaque page virtuelle (max 16 pages).

b) Implémenter un compteur global **TVC** (Temps Virtuel Courant) qui sera incrémenté à chaque accès à une page et mettre à jour le compteur de la page accédée avec la valeur courante du compteur global.

c) Etudier et éditer la fonction « ***main()*** » afin d'appeler la fonction « *incCompteur()* » lors de l'accès à une page par le processus (dans notre situation, un seul processus est présent, c'est le programme lui-même).

d) Enfin, écrire le contenu de la fonction « ***replaceLRU()*** » qui recherche la page avec la plus petite valeur de compteur (page la plus ancienne)

⇒ Mettre le bit « *valid* » à 0 et le bit « *swap* » à 1 de la page qui est déchargée.
⇒ Dans la fonction « ***main()*** », faire l'inverse pour la page chargée.

e) Tester et valider en exécutant la commande « ***memreplace LRU*** **»** ; un ensemble de 7 pages simulées sont accédées par le processus selon la séquence suivante : **5 2 1 4 2 2 3 6**

```
$ ./memreplace.elf LRU
RAM : [0] [1] [2]
SWAP : [3]
Enter the page to be accessed:
```

⇒ Le résultat final attendu est :

```
RAM : [2] [3] [6]
SWAP : [0] [1] [4] [5]
```

f) Simuler le comportement sur papier de la version « classique » de l'algorithme LRU et comparez avec l'affichage obtenu pour valider l'implémentation.

Etape 2 – Evolution vers l'algorithme WSClock

Dans l'étape précédente, la page déchargée est la page présente en RAM avec l'accès le plus ancien. Cette étape propose d'affiner la sélection de la page en appliquant la fenêtre du *working set* issue de l'algorithme WSClock avec les estampilles temporelles (*timestamps*) des pages. La configuration de la mémoire est la suivante : 3 pages physiques et 16 pages virtuelles.

Cette étape est réalisée dans la partie consacrée à l'algorithme WSC dans la fonction « ***main()*** ».

a) Utiliser le compteur représentant le **TVC** (Temps Virtuel Courant), que nous simplifierons en l'incrémentant à chaque accès à une page**.**

b) Utiliser le tableau intégrant la notion de **TDU** (Temps Dernière Utilisation) des pages lorsqu'elles sont référencées (max 16).

c) Ecrire la fonction « ***updateTDU()*** » qui a pour but de mettre à jour les **TDU** des pages à **TVC** lorsque le bit R vaut 1. Adapter la partie *WSC de la* fonction « ***main()*** » en conséquence lors de l'appel à une page pour que la fonction soit appelée à chaque fois.

d) Editer la fonction « ***main()*** » afin de modifier les propriétés suivantes de la page lors de son appel par le processus :

- Bit de référence défini à 1
- Bit de validité défini à 1

e) Ecrire le contenu de la fonction « ***replaceWSC()*** », qui définit la page à décharger dans le cas où la mémoire physique est pleine et que la page recherchée n'est pas présente, en considérant l'approche suivante :

- Appliquer une seconde chance si R vaut 1, en passant R à 0. Si R vaut 0, tester si la page est dans l'ensemble de travail avec la fenêtre d'observation *delta =2*. Si la page est hors de l'ensemble de travail, décharger la page.

- Si aucune page ne remplit les conditions pour être déchargée, supprimer la première page en RAM (**Ne pas oublier de** passer le bit « *valid* » à 0 et le bit « *swap* » à 1.

f) Tester et valider en exécutant « ./***memreplace.elf WSC*** ».

⇒ Utiliser la même séquence que pour l'étape précédente. Le résultat final attendu est :

```
RAM : [3] [4] [6]
SWAP : [0] [1] [2] [5]
```