Compte rendu TP3 : Nachos

# Modification

## userprog/translate.cc

### Fonction clearSC

void TranslationEntry::clearSC () {

this->secondChance = true;

}

### Fonction clearRefHistory

void TranslationEntry::clearRefHistory () {

timePageUse = 0;

historyPage = 0;

}

### Fonction setTime

unsigned int TranslationEntry::getTime () {

// return the access time here

return this->timePage;

}

### Fonction getTime

void TranslationEntry::setTimeUse(unsigned int newTime) {

timePageUse = newTime;

}

### Fonction getTimeUse

unsigned int TranslationEntry::getTimeUse () {

// return the access time here

return timePageUse;

}

### Fonction Translate

entry->historyPage |= 0x80;

entry->setTimeUse(stats->totalTicks);

## userprog/translate.h

unsigned int timePage;

unsigned int timePageUse;

unsigned char historyPage;

bool secondChance = true;

void setTimeUse (unsigned int theTime);

unsigned int getTimeUse ();

## userprog/Addrspace.cc

### Fonction SaveState

unsigned char mask;

switch (wsDeltaSize) {

case 1:

mask = 0x80;

break;

case 2:

mask = 0xC0;

break;

case 4:

mask = 0xF0;

break;

default:

mask = 0XFF;

break;

}

for (int i = 0; i < numPages; i++) {

pageTable[i].historyPage = (pageTable[i].historyPage >> 1) & mask;

}

numPages = machine->pageTableSize;

### Fonction TooManyPages

int AddrSpace::TooManyPages () {

// return the appropriate value here

return (NumPagesUsed() > 64);

}

### Fonction Fifo\_choose\_Victim

int AddrSpace::FIFO\_Choose\_Victim (int notMe) {

// You need to return an appropriate value here.

int tmp = -1;

for (size\_t i = 0; i < numPages; i++) {

if (i != notMe

&& pageTable[i].valid

&& (tmp == -1 || pageTable[i].timePage <= pageTable[tmp].timePage)) {

tmp = i;

}

}

return pageTable[tmp].physicalPage;

}

### Fonction LRU\_Choose\_Victim

int AddrSpace::LRU\_Choose\_Victim (int notMe) {

int tmp = -1;

for (size\_t i = 0; i < numPages; i++) {

if (i != notMe

&& pageTable[i].valid

&& (tmp == -1 || pageTable[i].historyPage <= pageTable[tmp].historyPage)) {

tmp = i;

}

}

return pageTable[tmp].physicalPage;

}

### Fonction SC\_choose\_victim

int AddrSpace::Fifo\_next\_victim (int act, int notMe) {

int tmp = -1;

printf("Numpage : %d \n",numPages );

for (size\_t i = 0; i < numPages; i++) {

if (i != notMe && i != act

&& pageTable[i].valid

&& pageTable[i].timePage >= pageTable[act].timePage

&& (tmp == -1 || pageTable[i].timePage <= pageTable[tmp].timePage)) {

tmp = i;

}

}

return i;

}

int AddrSpace::SC\_Choose\_Victim (int notMe) {

int i = FIFO\_Choose\_Victim (notMe);

while (1) {

if (pageTable[i].historyPage == 0) {

return pageTable[i].physicalPage;

} else if (pageTable[i].historyPage != 0

&& pageTable[i].secondChance == true) { // Donner une deuxième chance

pageTable[i].secondChance == false;

} else { // Deuxieme chance consommé

return pageTable[i].physicalPage;

}

if ((i = Fifo\_next\_victim (i, notMe)) == -1)

i = FIFO\_Choose\_Victim (notMe);

}

return 0;

}

# Résultat

## Console

## Graphique

## Access1 : nombre de page Fault en fonction du delta et de l’algorithme

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| delta | 1 | 2 | 4 | 8 |
| fifo | 137 | 137 | 137 | 137 |
| lru | 134 | 130 | 130 | 130 |
| secondChance | 131 | 131 | 131 | 131 |

## Access1 : nombre de page out en fonction du delta et de l’algorithme

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| delta | 1 | 2 | 4 | 8 |
| fifo | 69 | 69 | 69 | 69 |
| lru | 71 | 67 | 67 | 67 |
| secondChance | 67 | 67 | 67 | 67 |

## Access2 : nombre de page Fault en fonction du delta et de l’algorithme

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| delta | 1 | 2 | 4 | 8 |
| fifo | 17167 | 17167 | 17167 | 17167 |
| lru | 15899 | 15591 | 15711 | 15951 |
| secondChance | 15472 | 15472 | 15472 | 15472 |

## Access2 : nombre de page Out en fonction du delta et de l’algorithme

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| delta | 1 | 2 | 4 | 8 |
| fifo | 19583 | 19583 | 19583 | 19583 |
| lru | 15836 | 15528 | 15648 | 15888 |
| secondChance | 15408 | 15408 | 15408 | 15408 |

# Analyse