OS 课设6

信息科学与工程学院

2019011777 计算机 19-3 刘康来

实验 6 Web 服务器内存管理

- 简单的实现了 LRU，LFU 算法，一循环队列
- 其他，没怎么写

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

typedef struct Cache {
  char *fstr;
  char buffer[BUFSIZE + 1];
  char file_buffer[FILEBUFSIZE];
  int page;
  int lru;
  int lfu;
  struct Cache *next;
} Cache;

int num_cache = 0;
int faults = 0;

Cache *head, *tail;

void inqueue(Cache *frame) { // 插入队尾
  frame->next = tail->next;
  tail->next = frame->next;
  num_cache++;
}

void outqueue(Cache *previous) {
  Cache *frame = previous->next;
  previous->next = frame->next;
  frame->next = NULL;
  free(frame);
  num_cache--;
}

Cache *findPage(int page) {
  Cache *point = head;
  for (int i = 0; i < num_cache; i++) {
    if (point->page == page)
      return point; // 返回找到的块
    point = point->next;
  }
  return NULL;
}

void LRU(char *buffer, int hit, int page) {
  Cache *now_frame;
  // 为每一个 frame 赋值，此值为其最近出现的位置，此值最小则替换。
  if ((now_frame = findPage(page)) == NULL) {
    Cache *frame;
    frame = (Cache *)malloc(sizeof(Cache));
    frame->page = page;
    frame->lru = hit;
    faults++;

    if (num_cache == 100) {
      int min_lru = head->lru;
      Cache *point = head;
      Cache *ppoint = tail;
      Cache *previous = head; // 待替换页的前一页
      for (int i = 0; i < num_cache; i++) {
        if (point->lru < min_lru) {
          min_lru = point->lru;
          previous = ppoint;
        }
        ppoint = point;
        point = point->next;
      }
      outqueue(previous);
    }

    inqueue(frame);
  } else
    now_frame->lru = hit;
}

int max_lfu = 0;
int min_lfu = 0;
// 新数据马上被淘汰，可设置一个中位数访问频率，如何取这中位数呢？在找最小访问频率同时找最大，记录之
// 第一个替换的页 lfu 没设好:
void LFU(char *buffer, int hit, int page) {
  Cache *now_frame;
  // 为每一个 frame 赋值，此值为其最近出现的位置，此值最小则替换。
  if ((now_frame = findPage(page)) == NULL) {
    Cache *frame;
    frame = (Cache *)malloc(sizeof(Cache));
    frame->page = page;
    frame->lfu = (max_lfu + min_lfu) / 2;
    faults++;

    if (num_cache == 100) {
      int min_lfu = head->lfu;
      int max_lfu = head->lfu;
      Cache *point = head;
      Cache *ppoint = tail;
      Cache *previous = head; // 待替换页的前一页
      for (int i = 0; i < num_cache; i++) {
        if (point->lfu < min_lfu) {
          min_lfu = point->lfu;
          previous = ppoint;
        }
        if (point->lfu > max_lfu) {
          max_lfu = point->lfu;
        }
        ppoint = point;
        point = point->next;
      }
      outqueue(previous);
    }

    inqueue(frame);
  } else
    now_frame->lfu++;
}

int main(void) {
  Cache *frame;
  frame = (Cache *)malloc(sizeof(Cache));
  frame->page = -1; // no page in frame.
  frame->lru = 0;
  frame->lfu = 0;
  head = frame;
  tail = frame;
  frame->next = tail;

  return 0;
}
```