Memory exercise

Daniel Hagimont

Daniel.Hagimont@enseeiht.fr

USTH

March 2018

Objectives

The objective of this exercise is to simulate the behavior of a virtual memory management system. This simulation should be programmed in C language and should simulate memory accesses, page faults, memory allocation and replacement strategy.

Structure

This simulation will simply consist in a C program which includes :

```
#define NB_DISK_PAGE 10
#define PAGE_SIZE 20
#define NB_MEM_PAGE 5

char Disk[NB_DISK_PAGE][PAGE_SIZE];
char Memory[NB_MEM_PAGE][PAGE_SIZE];
int PageTable[NB_DISK_PAGE];
struct {
        boolean free;
        int date;
        int npage;
} MemState[NB_MEM_PAGE];

int Date = 0;
```

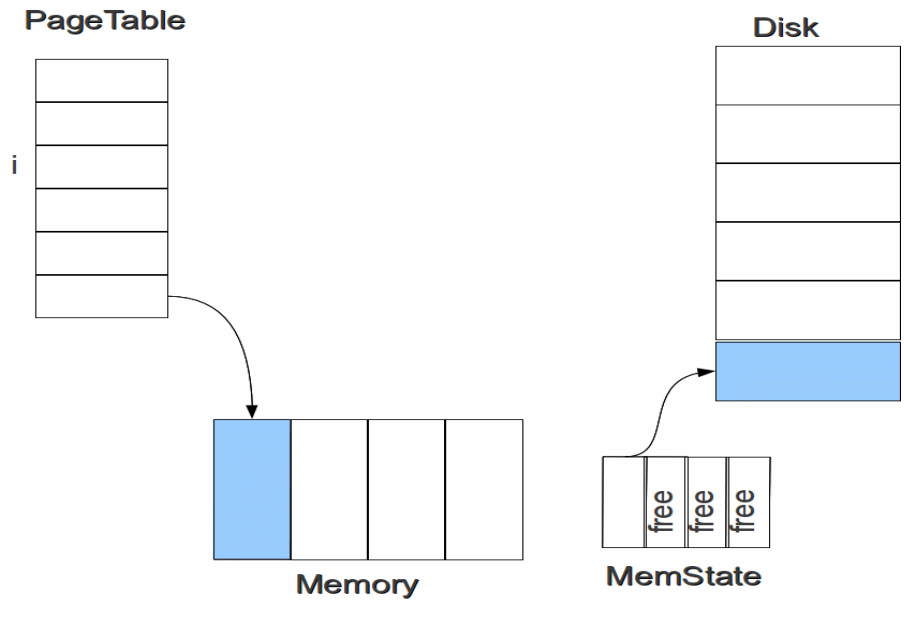
We assume that:

- a virtual address space maps a disk space (represented by the **Disk** table)
- the main memory (RAM) of the system is represented by the **Memory** table
- we manage a single address space represented by the **PageTable** table

- the state of each page in main memory is maintained in the **MemState** table. Each record in this table indicates for each page, whether the page is free or not, the date of its last access (for LRU management) and the page number it contains.

Each memory access increments the **Date** variable and updates the date field in the record associated with the accessed page in main memory.

These data structures are illustrated on the following figure:



Instructions

You have to implement the following functions:

- **void init()** - initializes all the data structures. The pages in the *Disk* table are initialized with strings ("page1", "page2", …)
- **char *memory_read(int npage)** – simulates a read memory access in a page of the virtual address space, with an indirection through the *PageTable*. It calls the *page_fault* function is necessary.
- **int memory_alloc()** - allocates a free page in main memory. Returns a free page number, -1 if no free page is found. This function is used in the implementation of the *page_fault* function.
- **void page_fault(int npage)** – resolves a page fault. This function is used in the implementation of the *memory_read* function. To resolve the page fault, this function invokes the *memory_alloc* function and it may invoke the *lru_select* function if no free page is found.

- **int lru_select()** - selects a page in main memeory for replacement. This function is used by the *page_fault* function.

The main function will use this memory management system. You can first read each page of the virtual address space and verify the it behaves consistently (you have to add traces to verify it). Then you can implement the scenario described in the lecture :

```
printf("access pages as in lecture (1,2,3,4,1,2,5,1,2,3,4,5,2,3)\n");
int serie[14] = {0,1,2,3,0,1,4,0,1,2,3,4,1,2};
for (i=0;i<14;i++)
        printf("read access page %d : %s\n",serie[i],memory_read(serie[i]));
printf("completed\n");
printf("print memory_state\n");
for (i=0;i<NB_MEM_PAGE;i++)
        printf("%d ",MemState[i].npage);
printf("\n");
```