

CSE 312 OPERATING SYSTEMS
HW-2 REPORT
200104004095
MEHMET HÜSEYİN YILDIZ

In this homework we would implement 3 page replacement algorithms:

- FIFO (First-In-First-Out)
- Second Chance
- LRU (Least Recently Used)

But to implement these algorithms we need to design a virtual memory system. To implement virtual memory I designed a virtual memory entry table. For fifo and second chance algorithms we need a queue (linked list) structure so I held all entries in a linked list. The structure of entry is like this:

```
struct TableEntry{
    int page_index;
    void* physical_address;
    bool present;                // if the entry used or not used.
    bool referenced;             // is referenced by read or write. For Fifo and Second chance
    int last_referenced_counter; // it holds the last usage time by holding a counter. For LRU

    TableEntry* nextEntry;
    TableEntry* prevEntry;
};
```

I held memory entries and disk entries in my VirtualMemoryManager class.

```
// Virtual memory manager class that manipulates virtual memory (memory and disk simulation)
class VirtualMemoryManager
{
private:
    int va_count;
    int reference_count;

    struct TableEntry* last_mem_page; // Last mem page
    struct TableEntry* last_disk_page; // Last disk page

    void* page_allocate(short disk_or_mem ); // DISK | MEMORY
    TableEntry* entry_allocate( ); // DISK | MEMORY

public:
    int hit;
    int miss;
    int page_load;
    int page_write;

    int page_size; // page size for every page in bytes.
    struct TableEntry* mem_page_table; // Memory Page table
    struct TableEntry* disk_page_table; // Disk Page table

    static MemoryManager* mem_manager; // Holds the memory_manager
    static MemoryManager* disk_manager; // Simulates the disk_manager
    static VirtualMemoryManager* activeVMM;

    VirtualMemoryManager(int page_size, MemoryManager* mem_manager1, MemoryManager* disk_manager1)
    ~VirtualMemoryManager(); // Destructor
};
```

I implemented linked list for TableEntry and their required functions like:

```
TableEntry* find_entry(short onDisk, int page_index);           // find the page entry by virtual address
TableEntry* pop_entry(bool on_disk);                           // removes the last entry of linked list and returns
void push_entry(int on_disk, TableEntry* new_entry);           // adds the given entry to head of linked_list
TableEntry* remove_entry(bool on_disk, int index);              // removes the entry with given index
```

These functions implemented for base table entry operations. They are implemented as we learned linked list in Data Structures and Algorithms course. If we want to apply the functions on disk entries then we make `on_disk = 1` and for memory entries `on_disk = 0`.

After creating these structure. Then we need some virtual memory manager class methods like:

```
VA allocate(common::size_t size);                             // allocation of pages in given size
void free(VA virtual_addr);                                   // deallocation of pages
void* operator[](VA virtual_address);                         // converts the virtual address to physical address.
bool is_on_disk(VA va);                                       // returns if corresponding physical address of given virtual address is
int* get(VA baseVa, int index);                               // returns the int index based given virtual address.
```

The VirtualMemoryManagement class basically holds 2 MemoryManagement objects which implemented in 16th wyos video. One of them is for memory pages (`mem_manager`) and the other one is for disk memory pages (`disk_manager`). Disk is simulated using MemoryManagement class.

allocate(): Allocates a page in memory by using memory manager `malloc()` function. If it has no place then it allocates in disk.

Operator[]: Converts virtual address to physical address. All reference bit or counter operations are made here. The specified replacement algorithm is called here. If there is no such virtual address, then it prints error message.

These are the basic 2 function I used. Allocation and accessing is realized with these.

The page replacement part is realized by using table entry functions like `remove_entry()`, `pop_entry()`, `push_entry()` etc.

Replacememnt Algorithms

Fifo

```
short VirtualMemoryManager::page_replace_fifo(TableEntry* disk_entry){
    TableEntry* oldMemEntry = pop_entry(0);
    page_replace(oldMemEntry, disk_entry);
    push_entry(1, oldMemEntry);
    push_entry(0, disk_entry);
}
```

As can be seen in the photo up-side, I used the basic linked list operations and page_repace() function which replaces the pages of given 2 entry.

Second Chance

```
short VirtualMemoryManager::page_replace_second_chance(TableEntry* disk_entry){
    TableEntry* oldMemEntry;
    // finding the not referenced entry
    while( (oldMemEntry = pop_entry(0))->referenced )
        push_entry(0,oldMemEntry);

    // if the entry is not referenced then we replace it
    page_replace(oldMemEntry, disk_entry);
    push_entry(1, oldMemEntry);
    push_entry(0, disk_entry);
}
```

This algorithm is very similar to fifo. There is entry bit in every entry. When a page used (with operator[]) the entry reference bit is assigned as true. After every specific entry reference all reference bits are assigned false. If the popped entry is referenced, then it just pushes to the end of linked list. It makes this operation until it find the not referenced entry. Finally replaces the pages with disk entry and pushes them to the entries.

Least Recently Used (LRU)

```
short VirtualMemoryManager::page_replace_LRU(TableEntry* disk_entry){  
  
    int lru_index = find_least_recently_used_mem_entry();  
    TableEntry* oldMemEntry = remove_entry(0,lru_index);  
  
    page_replace(oldMemEntry,disk_entry);  
    push_entry(1, oldMemEntry);  
    push_entry(0, disk_entry);  
  
}
```

For this algorithm, every entry has last_referenced_counter. Every entry counter is increased on every referencing in operator[] function. The called entry is made 0. In page_replace_LRU() function find_least_recently_used_mem_entry() function traverses all memory entries and finds the entry has maximum counter and returns its index. Then the entry is removed from the linked list and page replaced. Lastly both entries pushed to the entry lists again.

Example outputs

The screenshot displays the Visual Studio Code editor with the file `virtualmemorymanagement.h` open. The code defines constants for the replacement algorithm (LRU), sorting algorithms (Bubble, Quick, Insertion), and initial array size. It also defines a `TableEntry` struct with fields for page index, physical address, presence, referencing, and last referenced counter. The code is part of a namespace `myos`.

On the right side, there is a terminal window showing the output of the program. The output includes the initial state of the array, the result of sorting, and the final state of the array after the LRU algorithm is applied. The output also shows the creation of a VM image and the successful completion of the process.

```
Before Sort: 6 246 326 186 386 346 26 86 186 446 226 486 486 46 426 386 286 146 126 286  
6 246 326 186 386 346 26 86 186 446 226 486 486 46 426 386 286 146 126 286 6 246  
326 186 386 346 26 86 186 446 226 486 486 46 426 386 286  
After Sort: 6 6 26 26 26 26 46 46 46 46 86 86 86 186 186 186 126 126 126  
146 146 146 186 186 186 286 286 286 226 226 226 246 246 246 286  
286 286 386 386 386 326 326 326 326 346 346 346 346 386 386 386 486 486  
486 486 426 426 426 446 446 446 446 486 486 486  
Hit:1286  
Miss:17  
Page Load:17  
Page Write:19  
Array Length:76  
Media summary: 0 sessions, 0 data blocks, 0 data, 79.7g  
free  
Added to ISO image: directory '/tmp/grub.GANdE'  
xorriso : UPDATE : 574 files added in 1 seconds  
Added to ISO image: directory '/home/huseyin/Desktop  
/CSE 312 OS/Homeworks/Hw2/wyooos-p16/iso'  
xorriso : UPDATE : 578 files added in 1 seconds  
xorriso : NOTE : Copying to System Area: 512 bytes from  
file '/usr/lib/grub/i386-pc/boot_hybrid.img'  
ISO image produced: 5642 sectors  
Written to medium : 5642 sectors at LBA 0  
Writing to 'stdio:mykernel.iso' completed successfully.  
rm -rf iso  
(killall VirtualBoxVM && sleep 1) || true  
VirtualBoxVM: no process found  
VirtualBoxVM --startvm 'os_hw2' &  
huseyin@huseyin-inspiron-7570:~/os_hw2$  
orks/Hw2/wyooos-p16$
```


The screenshot shows a Visual Studio Code editor with a C++ file named `virtualmemorymanagement.cpp`. The code defines a virtual memory management system with various algorithms and data structures. It includes a `TableEntry` struct and a `Replacement_Alogithm` enum. The program is compiled and run in a terminal window, which displays the output of the program, including memory usage statistics and a list of memory entries.

```
#ifndef __MYOS_VIRTUAL_MEMORY_MANAGEMENT_H
#define __MYOS_VIRTUAL_MEMORY_MANAGEMENT_H

#define ARRAY_CONSTANT 0.6
#define REPLACEMENT_ALGORITHM 3 // 1 --> FIFO 2 --> SECOND CHANCE 3 --> LRU
#define SORT_ALGORITHM 2 // 1 --> Bubble 2 --> Quick 3 --> Insertion
#define INITIAL_ARRAY 1 // 0 --> Sorted 1 --> Random

#define ENTRY_MAX 100000
#define DISK 0
#define MEMORY 1

#define REFERENCED_CLEAN_PERIOD 5

#include <memorymanagement.h>

namespace myos
{
    enum Replacement_Alogithm{
        FIFO = 1,
        SECOND_CHANCE = 2,
        LRU = 3
    };

    //void printint(int num);

    struct TableEntry{
        int page_index;
        void* physical_address;
        bool present; // if the entry used or not used.
        bool referenced; // is referenced by read or write. For Fifo and Second
        int last_referenced_counter; // it holds the last usage time by holding a counter.
    };
}
```

The terminal output shows the program's execution, including memory usage statistics and a list of memory entries. The output is as follows:

```
mkdir iso/boot
mkdir iso/boot/grub
cp mykernel.bin iso/boot/mykernel.bin
echo 'set timeout=0' > iso/boot/gru
b/grub.cfg
echo 'set default=0' >> iso/boot/gru
b/grub.cfg

The array is smaller than memory
Before Sort:246 326 186 386 346 26 86 106 146 226 486 406 46 426 386 286 146 126
After Sort:6 246 326 186 386 346 26 86 106
Hit:321
Miss:0
Page Load:0
Page Write:0
Array Length:28

xorriso : UPDATE : 578 files added in 1 seconds
xorriso : NOTE : Copying to System Area: 512 bytes from
file '/usr/lib/grub/1386-pc/boot_hybrid.img'
ISO image produced: 5642 sectors
Written to medium : 5642 sectors at LBA 0
Writing to 'stdio:mykernel.iso' completed successfully.

rm -rf iso
(killall VirtualBoxVM && sleep 1) || true
VirtualBoxVM: no process found
VirtualBoxVM --startvm 'os hw2' &
huseyin@huseyin-Inspiron-7570:~/Desktop$
```

As can be seen when array size is smaller than memory error message is printed.

The screenshot shows a Visual Studio Code editor with a C++ file named `virtualmemorymanagement.cpp`. The code defines a virtual memory management system with various algorithms and data structures. It includes a `TableEntry` struct and a `Replacement_Alogithm` enum. The program is compiled and run in a terminal window, which displays the output of the program, including memory usage statistics and a list of memory entries.

```
#ifndef __MYOS_VIRTUAL_MEMORY_MANAGEMENT_H
#define __MYOS_VIRTUAL_MEMORY_MANAGEMENT_H

#define ARRAY_CONSTANT 2.1
#define REPLACEMENT_ALGORITHM 1 // 1 --> FIFO 2 --> SECOND CHANCE 3 --> LRU
#define SORT_ALGORITHM 3 // 1 --> Bubble 2 --> Quick 3 --> Insertion
#define INITIAL_ARRAY 1 // 0 --> Sorted 1 --> Random

#define ENTRY_MAX 100000
#define DISK 0
#define MEMORY 1

#define REFERENCED_CLEAN_PERIOD 5

#include <memorymanagement.h>

namespace myos
{
    enum Replacement_Alogithm{
        FIFO = 1,
        SECOND_CHANCE = 2,
        LRU = 3
    };

    //void printint(int num);

    struct TableEntry{
        int page_index;
        void* physical_address;
        bool present; // if the entry used or not used.
        bool referenced; // is referenced by read or write. For Fifo and Second
        int last_referenced_counter; // it holds the last usage time by holding a counter.
    };
}
```

The terminal output shows the program's execution, including memory usage statistics and a list of memory entries. The output is as follows:

```
mkdir iso
mkdir iso/boot
mkdir iso/boot/grub
cp mykernel.bin iso/boot/mykernel.bin
echo 'set timeout=0' > iso/boot/gru
b/grub.cfg
echo 'set default=0' >> iso/boot/gru
b/grub.cfg

Before Sort:220 434 122 418 150 494 142 250 430 254 62 398 310 214 382 330 290 3
74 102 70 370 234 222 118 50 294 242 450 330 54 162 30 210 14 482 38 190 174 262
270 270 34 322 310 450 94 342 150 230 354 262 290 110 314 82 230 90 474 382 470
170 334 422 18 350 394 442 358 130 154 362 498 10 114 182 438 490 274 402 178
0 134 22 218 250 194 42 58 30 454 462 198 410 414 282 138 390 74 2 378

After Sort:2 10 14 18 22 30 34 38 42 50 54 58 62 70 74 78 82 90 94 98 102 110
4 118 122 138 134 130 142 158 154 158 162 170 174 178 182 190 194 198 202 210 21
4 218 220 222 230 234 238 242 250 254 258 262 270 274 278 282 290 294 298 302 31
0 318 322 330 334 338 342 350 354 358 362 370 374 378 382 390 394 398 402 41
0 414 418 422 430 434 438 442 450 454 458 462 474 478 482 490 494 498

Hit:7972
Miss:130
Page Load:130
Page Write:134
Array Length:100

xorriso : UPDATE : 578 files added in 1 seconds
xorriso : NOTE : Copying to System Area: 512 bytes from
file '/usr/lib/grub/1386-pc/boot_hybrid.img'
ISO image produced: 5642 sectors
Written to medium : 5642 sectors at LBA 0
Writing to 'stdio:mykernel.iso' completed successfully.

rm -rf iso
(killall VirtualBoxVM && sleep 1) || true
VirtualBoxVM: no process found
VirtualBoxVM --startvm 'os hw2' &
huseyin@huseyin-Inspiron-7570:~/Desktop$
```

The screenshot shows a Visual Studio Code editor with a C++ project named 'virtualmemorymanagement'. The code defines a virtual memory management system with various algorithms and data structures. The terminal window shows the execution of a virtual machine named 'os_hw2'.

```
#ifndef __MYOS_VIRTUAL_MEMORY_MANAGEMENT_H
#define __MYOS_VIRTUAL_MEMORY_MANAGEMENT_H

#define ARRAY_CONSTANT 2.1
#define REPLACEMENT_ALGORITHM 2 // 1 --> FIFO 2 --> SECOND CHANCE 3 --> LRU
#define SORT_ALGORITHM 1 // 1 --> Bubble 2 --> Quick 3 --> Insertion
#define INITIAL_ARRAY 1 // 0 --> Sorted 1 --> Random

#define ENTRY_MAX 100000
#define DISK 0
#define MEMORY 1

#define REFERENCED_CLEAN_PERIOD 5

#include <memorymanagement.h>

namespace myos
{
    enum Replacement_Algorithm{
        FIFO = 1,
        SECOND_CHANCE = 2,
        LRU = 3
    };

    //void printint(int num);

    struct TableEntry{
        int page_index;
        void* physical_address;
        bool present; // if the entry used or not used.
        bool referenced; // is referenced by read or write. For Fifo and Second
        int last_referenced_counter; // it holds the last usage time by holding a counter.
    };
};

//void printint(int num);

struct TableEntry{
    int page_index;
    void* physical_address;
    bool present; // if the entry used or not used.
    bool referenced; // is referenced by read or write. For Fifo and Second
    int last_referenced_counter; // it holds the last usage time by holding a counter.
};

//void printint(int num);
```

The terminal window shows the execution of a virtual machine named 'os_hw2'. The output displays the state of the virtual memory management system, including the number of hits and misses, and the state of the virtual memory management system.

```
Before Sort:370 470 434 122 410 150 494 142 250 430 254 62 390 310 214 382 330 2
90 374 102 70 370 234 222 118 50 294 242 450 330 54 162 98 210 14 482 38 190 174
202 278 270 34 322 318 450 94 342 150 230 354 262 200 110 314 82 230 90 174 382
470 170 334 422 18 350 394 442 358 130 154 362 498 10 114 182 430 490 274 402
70 70 134 22 210 250 194 42 50 30 454 462 190 410 414 202 130 390 74 2

After Sort:2 10 14 18 22 30 34 38 42 50 54 58 62 70 74 78 82 90 94 98 102 110 114
4 110 122 130 134 138 142 150 154 158 162 170 174 178 182 190 194 198 202 210 214
4 218 222 230 234 238 242 250 254 258 262 270 274 278 282 290 294 298 302 310 314
4 318 322 330 334 338 342 350 354 358 362 370 374 378 382 390 394 398 402 410 414
4 418 422 430 434 438 442 450 454 458 462 470 474 478 482 490 494 498

Hit:9021
Miss:279
Page Load:279
Page Write:283
Array Length:100

libbur
79.7g
Noo'
nds
/Desktop

/CSE 312 OS/Homeworks/Hw2/woos-p16/iso'
xorriso : UPDATE : 578 files added in 1 seconds
xorriso : NOTE : Copying to System Area: 512 bytes from
file '/usr/lib/grub/i386-pc/boot/hybrid.img'
ISO image produced: 5642 sectors
Written to medium: 5642 sectors at LBA 0
Writing to 'stdio:mykernel.iso' completed successfully.

rm -rf iso
(killall VirtualBoxVM && sleep 1) || true
VirtualBoxVM --startvm 'os_hw2' &
huseyin@huseyin-inspiron-7577:~/Desktop/CSE 312 OS/Homew
orks/Hw2/woos-p16$
```

Conclusion

This is all I done in this homework. I believe that I done best. Everything is designed as we learned in the lectures. And I designed these all on my own completely. In this report I tried to be clear and short as much as possible. I hope everything is clear.