Національний технічний університет України
“Київський політехнічний інститут
імені Ігоря Сікорського”
Факультет інформатики і обчислювальної техніки
Кафедра обчислювальної техніки

# Лабораторна робота №1

з дисципліни “ Системне програмне забезпечення”
на тему
"Алгоритми заміщення сторінок віртуальної пам’яті"

Виконав:
Студент 4 курсу
групи ІО-62
Лавріненко Н.Т.

Перевірив: Сімоненко А.В.

Київ 2019

# The Not Recently Used Page Replacement Algorithm

Щоб операційна система могла збирати корисну статистику про те, які сторінки використовуються, а які ні, більшість комп'ютерів з віртуальною пам'яттю мають по два біти стану, пов'язані з кожною сторінкою. R встановлюється, коли на сторінку посилається (читається чи пишеться). M встановлюється, коли сторінка записується (тобто модифікується). Біти містяться у кожному записі таблиці сторінок. Важливо усвідомити, що ці біти повинні оновлюватися в кожному посиланні на пам'ять, тому важливо, щоб вони встановлювались обладнанням. Після того, як біт встановлено в 1, він залишається 1, поки операційна система не скине його на 0 у програмному забезпеченні.

Якщо в апараті немає цих бітів, їх можна моделювати наступним чином. Коли процес запускається, всі записи його сторінок у таблиці позначені як такі, що не є в пам'яті. Щойно на будь-яку сторінку посилається, відбудеться помилка сторінки. Потім операційна система встановлює біт R (у своїх внутрішніх таблицях), змінює запис таблиці сторінки, щоб вказувати на правильну сторінку, в режимі «ПРОЧИТАТИ ТОЛЬКО» та перезавантажує інструкцію. Якщо згодом сторінка буде записана, відбудеться інша помилка сторінки, що дозволить операційній системі встановити M біт і змінити режим сторінки на READ / WRITE.

## Лістинг:

### main.c

```
#include "allocator.h"

int main(void)
{
	init_mem(PAGE_NUMBER);
	mem_dump();
	remove(SWAP_FILE);
	testing(400, 100);
	mem_dump();
return (0);
}
```

### init_mem.c

```
#include "allocator.h"

t_physical_page *init_physical(void)
{
   return (calloc(1, sizeof(t_physical_page)));
}
```

```
void    init_mem(int num_pages)
{
   int i;

   i = 0;
   g_mem = calloc(num_pages, sizeof(t_virtual_page));
   for (i; i < num_pages; i++)
   {
      g_mem[i].physizal_page = init_physical();
      g_mem[i].id = i;
   }
}
```

## testing.c

```
#include "allocator.h"

void          free_reference()
{
   int i;

   i = 0;
   for (i; i < PAGE_NUMBER; i++)
      g_mem[i].reference = 0;
}

void    testing(int num_of_iteration, int system_timer)
{
   int           i;
   t_virtual_page  *page;

   i = 0;
   srand(time(NULL));
   for (i; i < num_of_iteration; i++)
   {
      page = choose_page();
      page->modify = rand() % 2;
      page->reference = rand() % 2;
      if (i == system_timer)
      {
         free_reference();
         system_timer += i;
      }
   }
}
```

## nru.c

```
#include "allocator.h"

int		func_1_1(int a, int b)
{
   return (a & b);
}

int		func_1_0(int a, int b)
{
   return (a && !b);
}

int		func_0_1(int a, int b)
{
   return (!a && b);
}

int		func_0_0(int a, int b)
{
   return (!a && !b);
}

t_virtual_page  *check_bits(int func(int, int))
{
   int i;

   i =0;
   for (i; i < PAGE_NUMBER; i++)
   {
      if (func(g_mem[i].modify, g_mem[i].reference))
         return (&g_mem[i]);
   }
   return (NULL);
}

t_virtual_page  *choose_page(void)
{
   t_virtual_page *page;

   page = check_bits(func_0_0);
   page = !page ? check_bits(func_0_1) : page;
   if (!page)
   {
      page = check_bits(func_1_0);
      page = !page ? check_bits(func_1_1) : page;
      save_swap(page);
      page->modify = 0;
```

```
      page->reference = 0;
   }
   return(page);
}
```

## Тестування

Для тестування було взято 10 сторінок, які вибиралися процесорами 400 тактів. (Системный час поновлення R біту 10тактів)

## swap

```
SWAP id = 0, phys addr = 0x55e34d880310
R = 1 M = 1
SWAP id = 0, phys addr = 0x55e34d880310
R = 1 M = 1
SWAP id = 0, phys addr = 0x55e34d880310
R = 1 M = 1
SWAP id = 9, phys addr = 0x55e34d880430
R = 0 M = 1
SWAP id = 8, phys addr = 0x55e34d880410
R = 0 M = 1
SWAP id = 7, phys addr = 0x55e34d8803f0
R = 0 M = 1
SWAP id = 6, phys addr = 0x55e34d8803d0
R = 0 M = 1
SWAP id = 5, phys addr = 0x55e34d8803b0
R = 0 M = 1
SWAP id = 4, phys addr = 0x55e34d880390
R = 0 M = 1
SWAP id = 3, phys addr = 0x55e34d880370
R = 0 M = 1
```