DM510 Operating Systems
Spring 2026

Exercise Sheet 6

Complete before tutorial on Thursday, March 26th

Learning goals

Be able to

- apply first-fit, best-fit, worst-fit
- translate from logical to physical addresses
- explain different forms of page tables
- explain internal and external fragmentation
- apply page replacement algorithms (LRU, FIFO, Optimal) on a reference string and analyze page faults
- interpret the impact of page faults on system performance
- use `malloc()` and `free()` understand memory leaks

Chapters 9+10

Exercise 1. Given six free memory sections of 300 KB, 600 KB, 350 KB, 200 KB, 750 KB, and 125 KB in this order, how would the first-fit, best-fit, and worst-fit algorithms place processes of size 115 KB, 500 KB, 358 KB, 200 KB, and 375 KB in this order?

Exercise 2. Why are page sizes always powers of 2?

Exercise 3. Consider a logical address space of 64 pages of size 1024 each, mapped onto a physical memory of 32 frames.

a. How many bits are there in the logical address?

b. How many bits are there in the physical address?

Exercise 4. Consider an operating system with 21 bit logical addresses, which runs on an embedded device with only 16-bit physical addresses. It also has a 2-KB page size.

a. How many entries are there in each of the following? A conventional, single-level page table; An inverted page table.

b. What is the maximum amount of physical memory?

Exercise 5. Consider the page table for a system with 12-bit logical and physical addresses and 256-byte pages, all addresses given in hexadecimal.

Page	Page Frame
0	–
1	2
2	C
3	A
4	–
5	4
6	3
7	–
8	B
9	0

The list of free page frames is D, E, F (that is, D is at the head of the list, E is second, and F is last). A dash for a page frame indicates that the page is not in memory. The following logical addresses are accessed in this order. Convert them to their corresponding physical addresses.

- 9EF, 111, 700, 0FF

Exercise 6. Consider the following page reference string:

$$1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6$$

How many page faults would occur for the following replacement algorithms, assuming three or six frames? Remember that all frames are initially empty, so your first unique pages will cost one fault each.

a. LRU replacement

b. FIFO replacement

c. Optimal replacement

Exercise 7. Describe the connection between locality of reference (similar addresses being referenced closely in time) and page faults.

Exercise 8. Consider a demand-paged computer system where the degree of multiprogramming is currently fixed at four (four processes run at the same time). The system was recently measured to determine utilization of the CPU and the paging disk. Three alternative results are shown below. For each case, what is happening? Can the degree of multiprogramming be increased to increase the CPU utilization? Is the paging helping?

a. CPU utilization 13 percent; disk utilization 97 percent

b. CPU utilization 87 percent; disk utilization 3 percent

c. CPU utilization 13 percent; disk utilization 3 percent

1 C Programming

Exercise 9. A LIFO list is a data structure where the last element that was inserted will be removed first. It can be implemented as a linked list. Implement the following functions and pay attention to correct usage of `malloc()` and `free()`.

```
struct list_entry {
        int data;
        struct list_entry* next;
};

struct list {
        struct list_entry* top;
};

void list_insert_top(struct list* l, int val);

int list_remove_top(struct list* l);

struct list* list_new();

void list_delete(struct list* l);
```

Exercise 10. In the previous exercise, would the program still execute correctly if you omitted `free()`? Would this lead to other problems? Why is it not necessary to free memory in other programming languages like Java and Python?

Exercise 11. The concept of fragmentation was introduced for continuous allocation in the lecture.

a. Discuss whether fragmentation is a problem for heap allocation using `malloc()` as well.

b. Discuss whether fragmentation is a problem for stack allocation (for example, the memory needed to store local variables in function calls).

Exercise 12. One approach to avoid explicit memory management is reference counting. Imagine that after creating an object with `malloc()`, the program automatically counts the number of pointers to it: when the pointer value is copied, this number is increased; when a pointer is overwritten or released (for example, it is a local variable and the containing function exits), then the number is decreased. When the number of references reaches zero, the memory is freed. C++ implements this idea very conveniently under the name of *smart pointers*, see https://en.cppreference.com/book/intro/smart_pointers. The details of reference counting are hidden from the programmer.
Discuss whether this solves the problem of memory leaks. Compare it with the alternatives.