Operating System

Midterm exam

1. Explain the following terms

(a) System call
  OS interface, call to the OS service
(b) Context switch
  Switch between processes for execution
(c) Microkernel OS
  An OS structure that only keeps the minimum kernel functions in kerel space, and it uses message passing to communicate between the functions in user space.
(d) Time sharing system
  Processes are forced to switch from CPU execution after a fixed time interval, so that the CPU can be shared more fairly among processes.
(e) Privilege instruction
  An instruction that can only be executed in kernel mode.

2. A process can be in one of the following states: new, terminated, waiting, running, and ready. Please draw a diagram showing the life cycle of a process and the event triggering process transition form one state to another state.

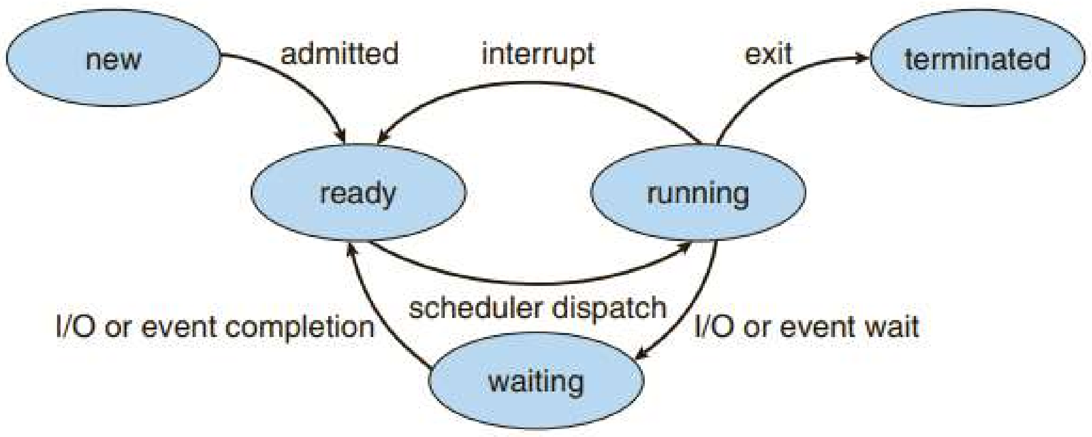

Figure 3.2 Diagram of process state.

3. Why memory address translation in modern computer first uses the segmentation, and then uses the paging?

Because segmentation can capture program's memory usage structure and pattern, so it is more convenient for users and programming.

On the other hand, paging is fixed size, so it is easier for OS to manage the physical memory space and improve memory utilization by preventing internal or external fragmentations.

4. (a) What is TLB? (b) Why TLB can reduce memory access time? (c) Why TLB must be flushed after content switch?

(a) TLB is the cache of page table.
(b) It can eliminate the memory access for page table lookup.
(c) TLB is shared among all processes. Since each process has its own page table content, TLB must be flushed after content switch to prevent from reading the wrong content from other processes.

5. (a) Explain what is dynamic loading and why it can improve memory utilization. (b) Explain what is dynamic linking and why it can improve memory utilization

(a) Dynamic loading only loads a function into memory when it is called. Since a program often contains many un-used functions for exception handling or other purposes, dynamic loading can reduce the memory usage of a program.

(b) Dynamic linking is to share library between processes at runtime.
When a process calls a dynamic linking library, the OS will first check if the library is loaded into the memory by other processes. If the library has been loaded, the process will link to that library and re-use it. Otherwise, the library will be loaded into the memory. Dynamic linking can prevent duplicated library loaded into the memory, so it can reduce memory usage.

6. (a) What is the memory-mapped file? (b) Give at least 2 pros and 2 cons about this method comparing to the file system calls. (c) Give an example use case that is suitable for using memory-mapped file.

(a) The definition of memory-mapped file :

1. It maps file into memory.
2. File is accessed by memory access (pointer) instead of file system call.

(b) The following is pros and cons

1. Pros : faster data access, easier to share file content, easier for programming, etc.
2. Cons : less secure, no access control, data loss problem, etc.

(c) I/O device input and output, like screen display, printer, etc.

7. (a) Why OS needs to break a one-level page table into multiple smaller page tables? (b) Considering "inverted page table", "hash page table", "hierarchical page table", choose the most suitable page table structure for each of the systems below. Briefly explain your reason.

I. System with scattered use memory space usage pattern.
II. System requiring fast memory access time.
III. System running large number of processes, but without page sharing.

(a) Because it is difficult to find contagious space on the physical memory to fit in large page table.

(b)
I. Hash page table, because it can effectively reduce the page table size
II. One level page table, because it only causes 1 additional memory access time for page table lookup.
III. Inverted page table, because all the processes can share the same page table, and the page sharing doesn't need to be supported.

8. (a) What is Belady's anomaly? (b) Why it is NOT a desired property for OS design? Briefly explain why LRU does not have Belady's anomaly.

(a) Page fault increases when more resources (memory frames) are given.
(b) It makes system administrator or OS harder to design the proper management mechanism or algorithm to utilize system resource properly.
(c) LRU, Least Recently Used, is a stack algorithm which means at any time the same set of pages will always in memory if we increase the number of frames.

By definition, LRU keeps the most recently used pages in memory. Therefore, if the number of frames increases, it will definitely keep the original pages, and further include the next recently used page that was not included.

9. Consider a computer with 3 memory frames to handle a reference string "3, 2, 1, 4, 2, 5, 1, 4 ,3 ,4, 1". Show the step-by-step reference result and indicate what are the references that will cause page faults using the following page replacement algorithm :

(a) FIFO

3	2	1	4	4	5	5	5	3	3	1
	3	2	1	1	4	4	4	5	5	3
		3	2	2	1	1	1	4	4	5

(b) LRU

3	2	1	4	2	5	1	4	3	4	1
	3	2	1	4	2	5	1	4	3	4
		3	2	1	4	2	5	1	1	3

(c) Optimal

3	2	1	4	4	5	5	5	3	3	3
	3	2	1	1	4	4	4	4	4	4
		3	2	2	1	1	1	1	1	1

10. Given a small process resides in page 0 (location 0 to 199) as shown below. It manipulates an integer array (4 byte per integer) that is allocated from the beginning of page 1. Assume the system only has 3 available frames, and the page size is 1000 bytes. If using pure demanding paging. Consider LRU page replacement algorithm is used. How many page faults are generated from running the program?

```
for (int i = 0; i < 1000; i++) A[i] = 1
for (int i = 999; i > -1; i--) A[i] = 0
```

1 page fault for loading the program
Each page stores 250 integers. Scan through the array needs 4 pages.
Total needs 9 pages accesses.
But the 2 pages for accessing A[1000]~A[500] won't cause page fault.
So the total page fault is 7.

11. Consider a byte addressable computer using a one-level paging scheme with a 12-bit logical address space, 4 byte page table entries, 1KB pages and a 2-entry TLB. The page-table base register access time is 0 ns. TLB access time is 10 ns and memory access time is 100 ns.

(a) How many memory address bits are needed for the page offset?
10bits (because page size is 1KB)

(b) How much memory in bytes is required to store the page table entirely in main memory? 2^2x4B = 16B (remaining 12 – 10 =2 bits)

(c) The CPU has just been context-switched to a new process whose page tables are entirely stored in main memory. If there is a one-to - one mapping between logical and physical addresses, what is the average effective access time for sequentially accessing the following set of logical address (n decimal digits): 150, 180, 2500, 1500, 1800, 150?

TLB miss : 150, 2500, 1500, 150
Time : [(10 + 100)x2 + (10 + 100x2)x4] / 6 = 176.6

(d) If logical address 18B (in hexadecimal digits) translates to physical address D8B (in hexadecimal digits), which page entry is used in the translation? What frame number is stored in that entry?

Logical address : 0001 1000 1010 => entry is 0

Physical address : 1100 1000 1010 => frame number is 3