# OS review: Questions

> Author: 钟庸 Yong Zhong

table of contents

## II. OS Basics

### Dual Mode Operation

1. Why is dual-mode needed?
2. What is needed in the hardware to support “dual mode” operation?
3. Draw the graph of Unix system structure. You should include the layers and the interfaces in between.
4. What are the 3 ways to transit from user mode to kernel mode? Explain each.
5. Give 3 types of system calls, each with an Unix syscall example.
6. Name 3 exception:
7. Name 3 interrupts:
8. Does user application code know the address of the system call functions? If not, how does it specify which system call to invoke?
9. What are the differences and similarities between interrupts and exceptions?

### Kernel Structure

1. Explain monolithic kernel and microkernel. What are the pros and cons of each?
2. Explain framekernel
3. What is hypervisor?
4. What are the design principles of OS? (user, system)
5. What is the difference between policy and mechanism?

### OS Services

1. Name 3 OS services.
2. What does system programs provide? Name 3 examples.

## III. Processes

### Process

1. What is a process?
2. What's the difference between a program and a process?
3. Draw the diagram of process state transition. Explain each state and transition.
4. Why a process in waiting cannot directly go to running?

### system calls

1. What are the 3 ways of passing parameters of syscall?
2. Why do we need library calls? Name some library calls and syscalls respectively.

### Process creation

1. Explain the possible return values of fork().
2. What will be duplicated in fork()? What will not?
3. Will exec() return to the caller? Why?
4. Will wait() wait for all child processes? How to wait for a specific child process? How to wait for all child processes?
5. How does child process wake up parent process after child terminates?

### Kernel view of processes

1. What info is stored in PCB? Name a few.
2. How is PCB organized in memory? Explain ready queue and device queue.

3. Explain cooperate switching and non-cooperate switching.

4. What will OS do during its regaining control of CPU?

5. What will happen during context switch?

### Kernel view of fork(), exec(), and wait()

1. What will happen if parent and child write into the same memory address after fork when using COW? What about writing to the same file?

2. What is zombie process?

3. exit() and wait() is responsible for which resources clean up or routines respectively?

4. Review the difference between wait() called before child exits and after child exits.

5. The code snippet below shows the severe consequence of not killing zombie. Explain why it is harmful.

```
int main(void) {
    while( fork() );
    return 0;
}
```

### More about processes

1. Which syscall is responsible for re-parenting?

2. Explain background jobs.

3. When will real time > user + sys time? When will real time < user + sys time?

## IV. Scheduling

1. Explain CPU-bounded and I/O-bounded processes.

2. What are the criteria of CPU scheduling?

3. During which transitions can scheduling happen? Based on that, explain preemptive and non-preemptive scheduling.

4. Explain SJF.

5. Explain RR.

6. Explain Priority Scheduling.

## V. Synchronization

### Race condition & Critical Section

1. Explain race condition.

2. What are the 3 requirements for critical section implementation?

## Disable Preemption & Spin-based Lock

1. Expalin when will disable interrupt work for mutual exclusion?

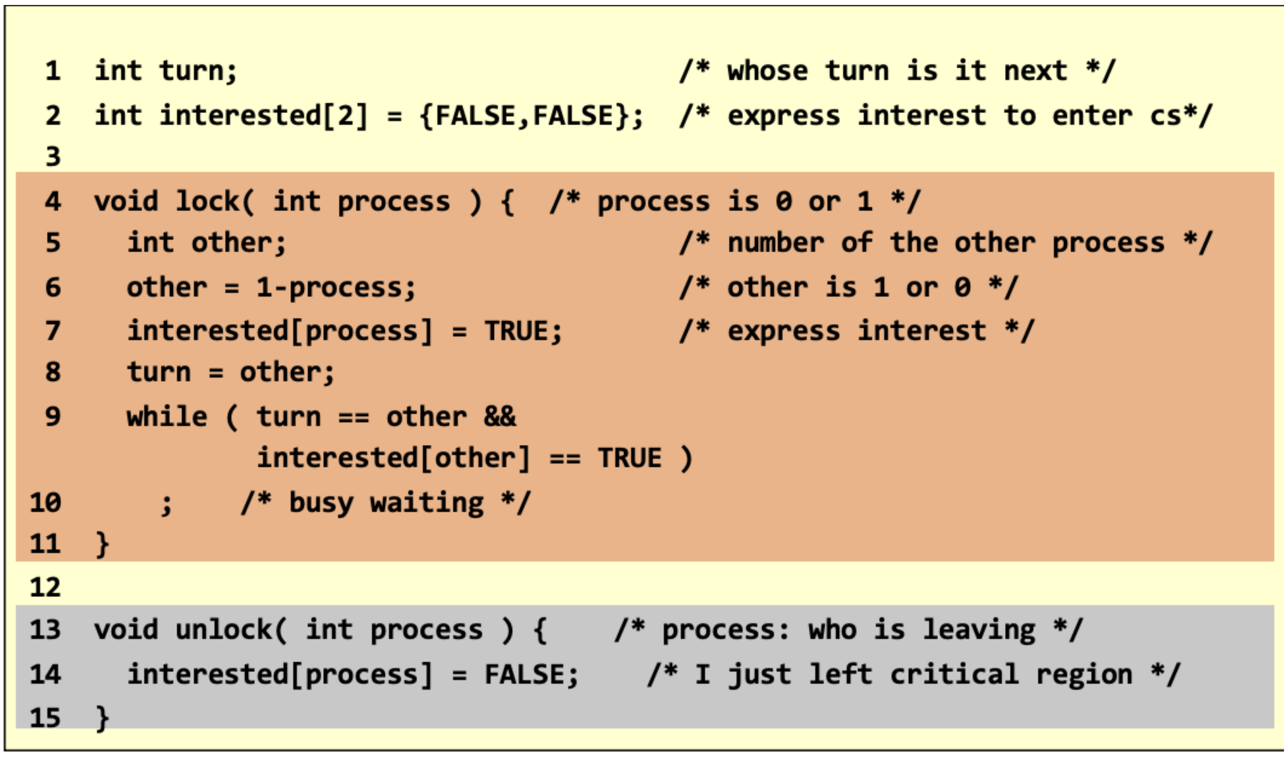
2. Why do we need atomic instructions? Are all the instructions in critical section atomic?

3. Explain strict alternating.

4. Explain progress violation in spin-based lock.

5. Why can Peterson's solution guarantee progress and bounded waiting?

6. Is peterson's algorithm correct if line 7 and line 8 are switched?



```
 1  int turn;                          /* whose turn is it next */
 2  int interested[2] = {FALSE,FALSE}; /* express interest to enter cs*/
 3
 4  void lock( int process ) {  /* process is 0 or 1 */
 5    int other;                       /* number of the other process */
 6    other = 1-process;               /* other is 1 or 0 */
 7    interested[process] = TRUE;      /* express interest */
 8    turn = other;
 9    while ( turn == other &&
              interested[other] == TRUE )
10      ;    /* busy waiting */
11  }
12
13  void unlock( int process ) {    /* process: who is leaving */
14    interested[process] = FALSE;    /* I just left critical region */
15  }
```

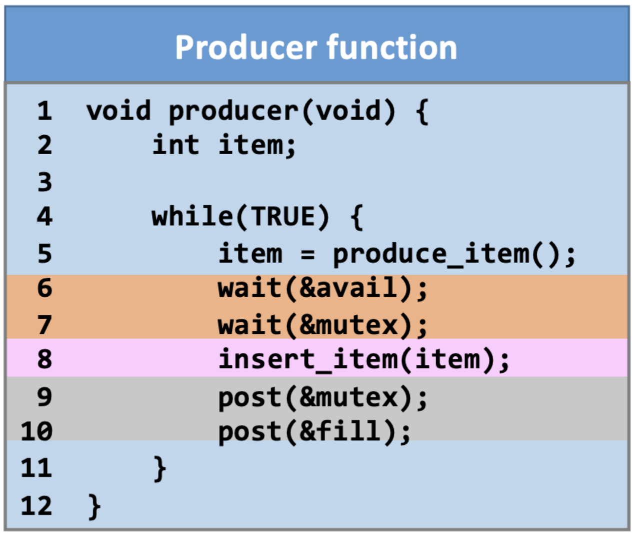
7. Explain priority inversion. How to solve it?

## Sleep-based Lock

1. Explain semophore, include semaphore struct and PV operations.

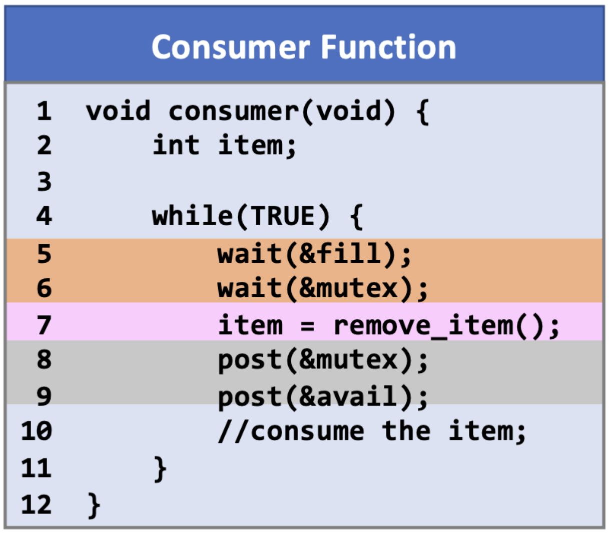
2. Suppose we are using binary semaphore. What's the initial value of sem->value? Now that sem->value = v, what dose it mean if v < 0, v = 0, v > 0?

## Producer-consumer problem



Producer function

```c
1   void producer(void) {
2       int item;
3   
4       while(TRUE) {
5           item = produce_item();
6           wait(&avail);
7           wait(&mutex);
8           insert_item(item);
9           post(&mutex);
10          post(&fill);
11      }
12  }
```

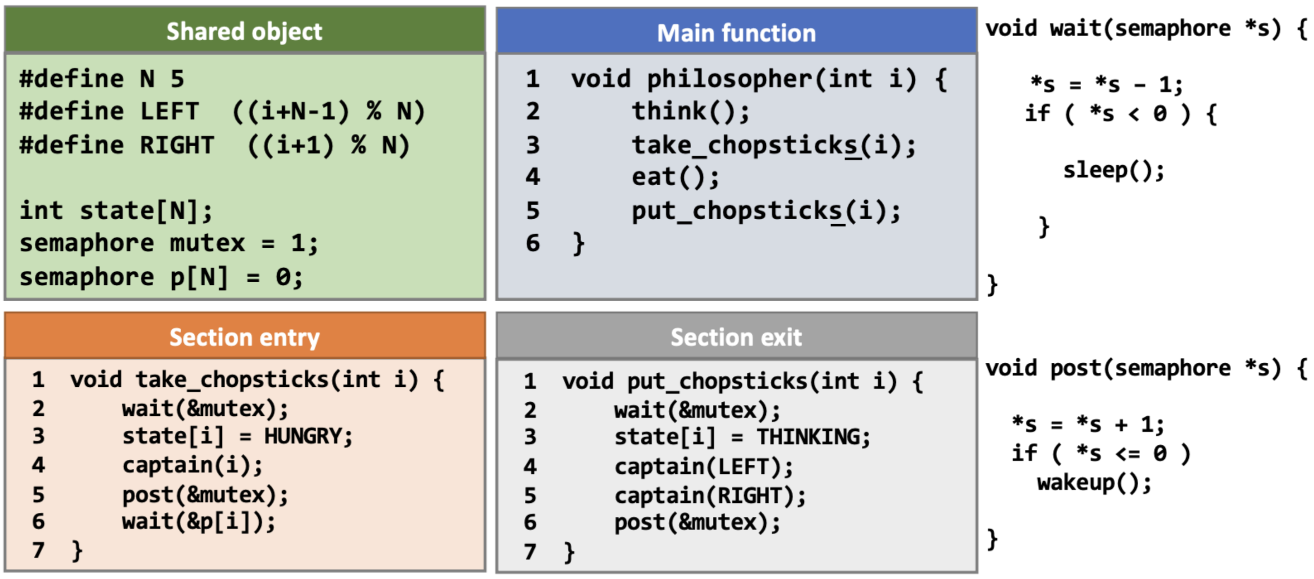


Consumer Function

```c
1   void consumer(void) {
2       int item;
3   
4       while(TRUE) {
5           wait(&fill);
6           wait(&mutex);
7           item = remove_item();
8           post(&mutex);
9           post(&avail);
10          //consume the item;
11      }
12  }
```

1. How many semaphores are needed in producer-consumer problem? Explain each semaphore's purpose.
2. Which line of code is responsible of waking up producer? Which line of code will put a consumer to sleep?
3. What if fill semaphore is not used? (producer line10 and consumer line 5 removed)
4. Can we swap wait(&avail) and wait ait(&mutex)?

## Dining philosopher problem



Shared object

```c
#define N 5
#define LEFT  ((i+N-1) % N)
#define RIGHT  ((i+1) % N)

int state[N];
semaphore mutex = 1;
semaphore p[N] = 0;
```

Main function

```c
void philosopher(int i) {
    think();
    take_chopsticks(i);
    eat();
    put_chopsticks(i);
}
```

```c
void wait(semaphore *s) {

  *s = *s - 1;
  if ( *s < 0 ) {

    sleep();

  }

}
```

Section entry

```c
void take_chopsticks(int i) {
    wait(&mutex);
    state[i] = HUNGRY;
    captain(i);
    post(&mutex);
    wait(&p[i]);
}
```

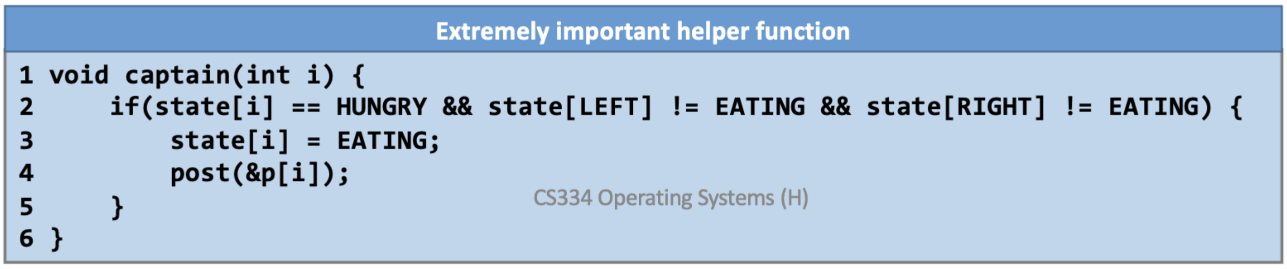
Section exit

```c
void put_chopsticks(int i) {
    wait(&mutex);
    state[i] = THINKING;
    captain(LEFT);
    captain(RIGHT);
    post(&mutex);
}
```

```c
void post(semaphore *s) {

  *s = *s + 1;
  if ( *s <= 0 )
    wakeup();

}
```



Extremely important helper function

```c
void captain(int i) {
    if(state[i] == HUNGRY && state[LEFT] != EATING && state[RIGHT] != EATING) {
        state[i] = EATING;
        post(&p[i]);
    }
}
```

1. Is it a good idea to use chopstick as semophore? Why?

2. How many somephores are needed if use philosophers as semophore? Explain each semaphore's purpose.
3. What is the invsible captain responsible for?
4. What dose p[i] == 0, p[i] == -1 mean?
5. Is it ok to swap line 4 (captain(i)) and line 5 (post(&mutex)) in entry?
6. What will happen in entry and exit?

## VI. Deadlock

### Basic Concepts

1. What's tehe relationship between deadlock and starvation?
2. What are the 4 requirements for deadlock?

### Detection

1. What are the shape and meaning of Available, Allocatoin, Request?
2. What does work and finish mean? What are they initialized to?
3. Explain the deadlock detection algorithm.
4. What actions may be taken after deadlock is detected?

### Prevention

1. In banker's algorithm, which requirements of deadlock is prevented?
2. Explain safe state and unsafe state.
3. Explain safety algorithm.
4. Explain resource-allocation graph and how to prevent deadlock with it.

## VII. Address Translation

### Memory Virtualization

1. Is the virtual address space size the same as the physical address space size? Why?
2. Name the requirements for memory virtualization and explain them.
3. Where is the translation from virtual address to physical address performed? What hardwares are involved?
4. During the address translation, what is the role of hardware and what is the role of software (OS)?

### Base and Bounds

1. Is base and bounds bind to process or to system? Why?
2. Explain the base and bounds translation mechanism.
3. What are the Limitations of base and bounds mechanism?

### Segmentation

1. Explain segmentation mechanism. What is the SegID used for? (hint: it is called enhanced base and bounds) (think: Is the base and bounds bind to process or to segment?)
2. Is the translation for stack segment different from the segment for heap and code? How?
3. What are the advantages of segmentation over base and bounds mechanism?
4. What are the disadvantages of segmentation?
5. How is the protection implemented in segmentation?

### Memory Allocation -- Linked Allocation

1. What are the 3 basic strategies for memory allocation? Explain the pros and cons of each.

## VIII. Paging

### Page and Page Table

1. Explain the paging mechanism. The virtual address is devided into which 2 parts? Why don't we need a bound register in paging?
2. What are in the PTE (Page Table Entry)? Explain each field.
3. What are the issues of single-level page table?

### Multi-level Page Table

1. How is the number of bits of L1 / L2 page table index determined? (hint: each page table fits in one page)
2. How does multi-level page table save memory space?

### Other Page Table Structures

1. Explain Inverted Page Table. What are the pros and cons?
2. Explain Hashed Page Table. What are the pros and cons?

### SV39 & IA32

1. How many levels are there in SV39 page table? Why 9 bits are used for each level index? (hint: size of PTE is 8 bytes)
2. Expalin the IA32 address translation mechanism. How many levels are there in IA32 page table? Explain the linear address.

### TLB

1. Where is TLB located?
2. What will happen during a TLB miss?
3. Name the replacement policies for TLB. Explain each.
4. What will happen to TLB during a context switch?

## IX. Demand Paging

### Demand paging mechanism

1. Who is responsible for moving data?
2. What is swap space?
3. What will happen if the present bit is 0 during address translation?

### Page Replacement Policy

1. How to calculate EAT (Effective Access Time)?
2. What are the 3 types of cache misses? Explain each.
3. Explain the following page replacement algorithms: MIN, FIFO, LRU, LFU.
4. What is Belady's anomaly? Which page replacement algorithms may suffer from it? Give an example.

### LRU Implementation

1. Explain Clock-algorithm, Enhanced clock-algorithm, Nth-chance algorithm.

### Frame Allocation

1. Explain the difference between global replacement and local replacement.
2. Explain the allocation algorithms: Equal allocation, Proportional allocation, Priority allocation.
3. Explain Thrashing. Why is thrashing a problem? How to solve thrashing?

## X. Linux Memory Management

### Address space in linux

1. Linux virtual address space is divided into which 2 parts? Explain each part.
2. Why is kernel memory mapped into the address space of each process?
3. Explain the difference between kernel logical address and kernel virtual addresses.
4. How to protect kernel space from user space access?

### Large Page

1. Explain large page. What are the pros and cons?
2. How to support large page in different ISA?

### Slab Allocator

1. Draw a diagram to explain the relationship among cache, slabs, and objects.
2. What are the states of a slab? How does a slab allocator handle allocation and deallocation of objects?

### Buddy System

1. Expalin the allocation and deallocation mechanism of buddy system.
2. How to find the buddy of a block?

## XI. IO and Storage

### Device

1. What are the 3 buses in a computer system?
2. Explain the hardware interface of a device. (status, command, data registers)

### IO Operations

1. Expalin polling.
2. Expalin interrupt.
3. How to decide between polling and interrupt?
4. Explain PIO. What are the drawbacks of PIO?

### More on interrupts

1. What are the hardware support for interrupts?
2. What are the software support for interrupts?

### DMA and drivers

1. Explain how DMA works.
2. Draw a diagram of DMA data copy from memory to disk.
3. Explain the top half and bottom half of device driver.

---

### Storage

1. What are the differences between magnetic disk and SSD?
2. Explain the structure of a magnetic disk.
3. Explain the three-stage process of magnetic disk data read/write.
4. What is the whole disk latency consist of?
5. Assume avg seek time = 5ms, rotational speed = 7200 RPM, transfer rate = 4 MB/s, sector size = 1 KB. What are the avg time reading a sector when 1) random place read 2) same cylinder read 3) next sector read?

### Disk Scheduling

1. What's the purpose of disk scheduling?
2. How to minimize the total head move distance?
3. Explain FIFO, SSTF.
4. Explain SCAN, C-SCAN.
5. Explain LOOK, C-LOOK.

## XII. File System

### Componentes & View

1. What are the 4 components of a file system?
2. What are the view of a file from User, System call, OS, Hardware?

### Disk Management

1. Expalain the entities in disk management.
2. Explain LBA.
3. Explain bitmap for free block management.
4. Explain file header for file structuring.

### Layout & Allocation

1. What should directory entry record?
2. Explain contiguous allocation. What are the problems?
3. Explain linked allocation. What are the problems?
4. How does FAT facilitate random access compared to linked allocation?

### More on FAT

1. Expalain 8 + 3 naming convention. What is the 1st byte of directory entry used for?
2. How to support LFN(Long file name) in FAT?
3. Explain appending a file in FAT.
4. Explain deleting a file in FAT.
5. Why is it possible for us to recover file in FAT?

## INode Allocation

1. What are the entities in Inode allocation?
2. What is the structure of Inode?
3. What is stored in indirect block?
4. Suppose block size = 4KB, block number size = 4 bytes, what is the max file size supported by Inode structure?

## Ext2/3/4

1. Describe the structure of block group in Ext 2/3.
2. How does ext 2/3 ensures performance and reliability using block group?
3. How does ext 2/3 facilitate file deletion?
4. How is hard link implemented in ext 2/3?
5. How does pathname length effect symbolic link implementation?