# Overview

(1) Solve the following:
   A. How much is 2^13 in decimals?
   B. How much is roughly 1 billion in power of 2?
   C. How much is 2^64/2^21 in power of 2?
   D. How much is 128MB/4KB ?
   E. How much is log2(8192)?

(2) When measuring I/O throughput, what is the difference between the units
   (1) MBps and Mbps
   (2) KBps and Kbps?

(3) How much are these units in decimals?
   (1) Pico
   (2) Nano
   (3) Micro
   (4) Milli
   (5) Kilo
   (6) Mega
   (7) Giga
   (8) Tera
   (9) Peta

   Hint:

   For metric system:

   See http://www.chemteam.info/Metric/Metric-Prefixes.html

   For size of information in computers see:
   https://web.stanford.edu/class/cs101/bits-gigabytes.html

(4) Replace "?" below with the correct answer
   A. 1 Nanosecond = ? seconds
   B. 500 Milliseconds = ? seconds
   C. 4 KB = ? bytes
   D. 4 Kilometers = ? meters
   E. 4Kbps = ? bits per second

(5) What is an Operating System? List its primary responsibilities.

(6) What are the three (or four) different ways in which OS code can be invoked? Explain.

(7) What is an Instruction Set Architecture (ISA)? What is the difference between User ISA and System ISA?

(8) What is a system call? How is a system call different from ordinary function calls?

(9) Explain the following interfaces in a computer system
   (a) Instruction Set Architecture (ISA)
   (b) User Instruction Set Architecture (User ISA),
   (c) System ISA,

(d) Application Binary Interface (ABI).
(e) Application Programmers' Interface (API)

(10) Why doesn't a program (executable binary) that is compiled on the linux machine execute on a Windows machine, even if the underlying CPU hardware is the same (say x86)?

(11) Let's say that you are asked to modify the Linux OS so that programs and libraries compiled on Windows OS could run *natively* on Linux, meaning they should be executed as normal programs (without using any emulator or virtual machine). What would be your high-level approach?

(12) What hardware mechanism does x86 ISA provide to ensure that Operating System's code and data are protected from user-level processes?

(13) What is the role of privilege levels (defined by the ISA) in a computer system? How many privilege levels are defined in the x86 ISA? In which privilege level does the OS execute?

(14) What is the difference between a hardware interrupt, a software interrupt (trap), and an exception? Give examples of each.

# Processes

(1) (a) What is a process? (b) How is a process different from a program?

(2) In the memory layout of a typical process, why do stack and heap grow towards each other (as opposed to growing in the same direction)?

(3) In terms of call-return behavior, how are the fork() and exec() system calls different from other system calls?

(4) (a) Describe the process lifecycle illustrating the states and transitions. (b) Which transitions occur when a process (i) is pre-empted? (ii) voluntarily yields the CPU?

(5) What is a zombie process? Why does the Operating System maintain the state of zombie processes? List two ways in which a parent process can prevent a child process from becoming a zombie.

(6) Why are frequent context switches expensive in terms of system performance?

(7) What is cold-start penalty? What are some ways to reduce it?

# Threads

1. What are threads? How do they differ from processes? How are they similar?
2. What state do threads share? What state is different?
3. Why does context-switching between threads incur less overhead than between processes?
4. Briefly explain
   (a) User-level threads
   (b) Kernel-level threads
   (c) Local thread scheduling
   (d) Global thread scheduling
5. What are the benefits and disadvantages of using user-level and kernel-level threads?
6. What combinations or user/kernel threads and global/local scheduling are feasible and why?
7. What kind of applications benefit the most from kernel-level threads support? What kind of applications benefit most from user-level threads support? Explain why with examples?
8. Explain how a web server could use threads to improve concurrency when serving client requests.
9. What happens if a thread in a multi-threaded process crashes? How can you improve the robustness (fault-tolerance) of a multi-threaded application?
10. When would you prefer (a) event-based programming (b) threads-based programming? Why?

11. Event-driven programming
    (a) What is the "event-driven" programming model?
    (b) What does the structure of a typical event-driven program look like?
    (c) When would you prefer an event-driven programming model over a thread-based programming model?

12. What is the problem with long-running event handlers? How do threads solve this problem?

## IPC

1. List any five inter-process communication mechanisms, with a one line description for each?

2. When using a pipe for inter-process communication, why should a process promptly close any unused write descriptors to the pipe? Also give an example of what happens if it doesn't.

3. Let's say a **chain of filters** refers to a series of commands whose standard inputs and standard outputs are linked by pipes. For example,

   "ps -elf | grep bash | more"

   is a chain with three commands.

   In the general case,

   "command1 | command2 | command 3 | ... | command K"

   is a chain of filters with K commands.

   Suppose you were implementing a shell (e.g. csh, bash, tcsh, ksh, etc.), how would you go about supporting a chain of filters with ***arbitrary*** number of commands? Explain. Don't write actual code.

4. What's the difference between byte-stream vs. message oriented communication?

# System Calls

1. What is a system call? How do system calls differ from ordinary function calls?

2. What steps take place when a system call is invoked by a process?

3. What is a system call table? Why is it needed? OR What role does it play in OS security?

4. Explain the CPU-privilege transitions during a system call.

5. (a) Why do some operating systems, such as Linux, map themselves (i.e. the kernel code and data) into the address space of each process? (b) What is the alternative?

# Kernel Modules

1. Why are memory access errors (such as segmentation fault and null pointer dereferencing) more dangerous in kernel space than in user space?

2. What are kernel modules?

3. What are some advantages of kernel modules?