1. Operating system architecture.
   * What are main tasks solved by an operating systems (services)?
   * What is operating system kernel?
   * Explain differences between monolithic and microkernel model of OS kernel. What models are used in Linux/MacOS/Windows?
   * Explain the idea of *kernel* and *user* modes of a processor.
   * What is a system call and how is it implemented?
2. C programming language. GNU C Library (glibc).
   * C language: brief history, what tasks it solves, advantages over assembly language.
   * What data types are supported in C? What is a pointer? How does sizeof work?
   * How to assign and how to dereference a pointer? How does address arithmetic work?
   * What is *glibc* (GNU C Library) and what tasks does it solve?
   * How are strings are represented in C? What functions to manipulate with strings do you know?
   * Explain the main idea of pattern-matching and regular expressions.
3. Executable and Linkable Format (ELF).
   * What is ELF? What kings of ELF object files do you know? What is their purpose?
   * What data do ELF object files contain?
   * Explain the idea of symbols and symbol table. What kinds of symbols do you know?
   * What happens when several object files are linked together (explain the idea of symbol resolution and relocation)?
   * What does it mean strong and weak symbol? Explain symbol resolution rules.
   * Explain the idea of position-independent code (PIC).
4. Compiling/linking/loading. Static and dynamic libraries.
   * List the compiler stages (steps to turn a C source file to an executable file).
   * What is done at the linking stage? What is the meaning of static and global keywords?
   * Explain the idea of static and shared libraries.
   * Explain the idea of run-time loading of shared libraries. What are the advantages of shared libraries?
   * Explain the idea of library interpositioning (compile time, link lime, load/run time).
   * What tasks are solved with the help of Make files? What are *target*, *source*, and *recipe* in a Make file?
5. Memory management.
   * Memory layout of a program: What memory segments do you know? What purposes do they serve?
   * What ways to allocate memory do you know?
   * How dynamic memory allocation via malloc/free is implemented (using what data structures)?
   * Give definitions of payload, fragmentation, and placement strategies.
   * What is the purpose of the sbrk system call?
6. Filesystems. Linux folder structure.
   * What Linux file types do you know?
   * Explain the purpose of the following Linux folders: /home, /bin, /sbin, /usr, /proc, /dev, /media.
   * What is Virtual File Systems (VFS) and what functions does it provide?
   * What are the parts of a Linux disk?
   * What is *inode*? What data does it contain?
   * What is a link? How to create it (what utility tool to use)? What is the difference between hard and symbol links?
7. System calls / system utilities / Shell (Bash).
   * Explain connection between system calls, system utilities and Bash.
   * What is Bash and what tasks does solves?
   * How to get a manual on Linux system utilities and system calls?
   * How (using what special symbols) to access command-line arguments in Bash?
   * What is the role of exit code in a program (e.g. 0 vs. -1)? Who exit code can be checked in Bash scripts?
   * Name Linux utilities that solve these tasks:
     + print current directory;
     + change current directory;
     + print the list of files/folders in the current directory;
     + create new folder;
     + copy file/folder;
     + remove file/folder;
     + move/rename file/folder;
     + print full path to a utility file (e.g. full path to gcc).
   * What Bash commands are used to read user input to a variable and to print variable values?
8. File input/output. Pipes and redirection.
   * What system calls are used to read/write data from/to files?
   * What *glibs* (C Standard Library) function to work with files do you know? Their advantages over system calls?
   * What is a file descriptor? What is descriptor table? What is open file table?
   * List three standard streams of a Linux process and their descriptors.
   * How to redirect process I/O from a terminal to a file?
   * How to connect I/O of two processes?
   * What is a pipe? What system calls are used to manage pipes?
9. Processes.
   * What is a process? What parts does it contains (its layout in memory)?
   * List the states of the process and describe how it changes states.
   * What is Process Control Block (PCB)? What information does it contain?
   * Explain how CPU switches between processes (context switch).
   * Explain the main idea of short-term, long-term, and medium-term schedulers. What is process swapping?
   * Describe the idea of process creation with system calls fork and exec. What is the role of system call wait?
   * How to see the list of running processes in Linux (what system utilities do you know)?
10. Threads and synchronization.
    * Give a definition of a thread. Explain the difference between a process and a thread.
    * Explain the main idea of consumer-producer problem.
    * Explain the idea of critical section and mutual exclusion.
    * How thread synchronization is supported in hardware?
    * List system calls that are used to manage threads in Linux (pthreads).
    * Explain the main idea of mutexes and conditional variables (pthreads).
    * What is a deadlock?
11. Permissions.
    * What are main attributes of a Linux user and group?
    * What access rights do you know? What permission groups do you know? How to view file permissions (what utility tool to use)?
    * How to change file permissions (what utility tool to use)? E.g. add write permission to *group*, remove read permission from *other*.
    * Explain the setuid/setguid permissions.
    * Explain the sticky bit permission.
12. Inter-process communications: signals.
    * Give a definition of a signal. What signals to you know (name and purpose)?
    * How to send a signal to a process (system call and utility tool)?
    * How to set up a custom handler for a signal? It is possible to do this for all signals?
    * Explain the idea of foreground and background processes. How to run a background process?
    * How to switch a process from foreground to background and vice versa?
13. Inter-process communications: message queues, memory mapping, shared memory.
    * Explain the main idea of two models of inter-process communication: shared memory and messages.
    * Describe main features of POSIX message queues. What system calls are used to manage POSIX message queues?
    * How subscribe to get a notification (a signal) when a message is available in the queue?
    * Explain the idea of mapping file into memory? What system call is used for this?
    * Describe main idea of POSIX shared memory. That system calls are used to manage it?
14. Network. Sockets and TCP/IP.
    * Explain the concept of a client-server application.
    * Explain the idea of a network protocol. What is a network packet and what information does it contain?
    * What protocols does the TCP/IP family include?
    * What is MAC address? What is IP address? How domain name (e.g. www.hse.ru) is converted to an IP address?
    * Explain the difference between TCP and UDP. What advantages and disadvantages do they have?
    * What is socket? What system calls are done by the client and the server to establish a communication?
    * What is a port? Give examples of network protocols you know and ports they use.