Quiz Submissions - Week 7 Quiz

✕

Timothy Holmes (username: THOLME15)

Attempt 2

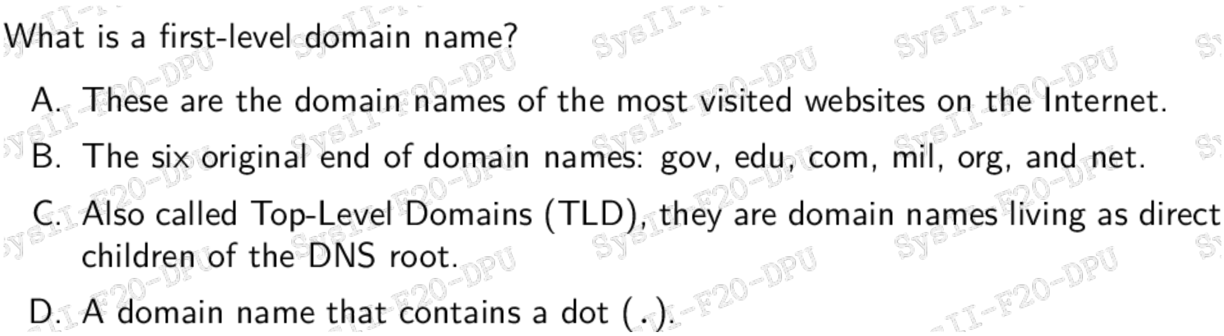
Written: Oct 26, 2020 8:22 PM - Oct 26, 2020 8:26 PM

Submission View

Released: Sep 7, 2020 10:15 PM

Network Programming

Question 1 — 1 / 1 point



What is a first-level domain name?

A. These are the domain names of the most visited websites on the Internet.

B. The six original end of domain names: gov, edu, com, mil, org, and net.

C. Also called Top-Level Domains (TLD), they are domain names living as direct children of the DNS root.

D. A domain name that contains a dot (.).

○ Answer A.

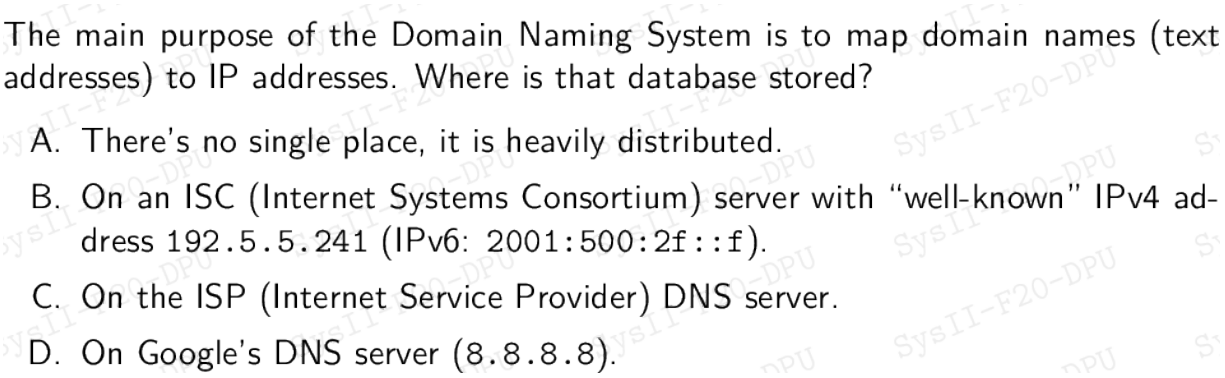
○ Answer B.

✓ ○ Answer C.

○ Answer D.

▷ View Feedback

Question 2 — 1 / 1 point



The main purpose of the Domain Naming System is to map domain names (text addresses) to IP addresses. Where is that database stored?

A. There's no single place, it is heavily distributed.

B. On an ISC (Internet Systems Consortium) server with "well-known" IPv4 address `192.5.5.241` (IPv6: `2001:500:2f::f`).

C. On the ISP (Internet Service Provider) DNS server.

D. On Google's DNS server (`8.8.8.8`).

✓ ○ Answer A.

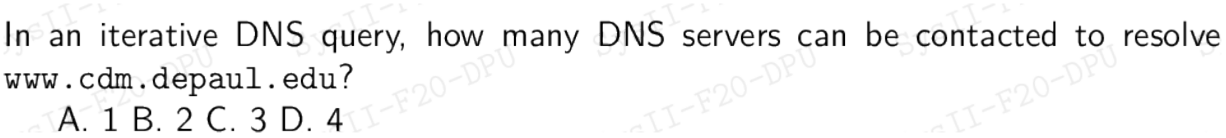
○ Answer B.

○ Answer C.

○ Answer D.

▷ View Feedback

Question 3 1 / 1 point



In an iterative DNS query, how many DNS servers can be contacted to resolve `www.cdm.depaul.edu`?
A. 1 B. 2 C. 3 D. 4

○ Answer A.

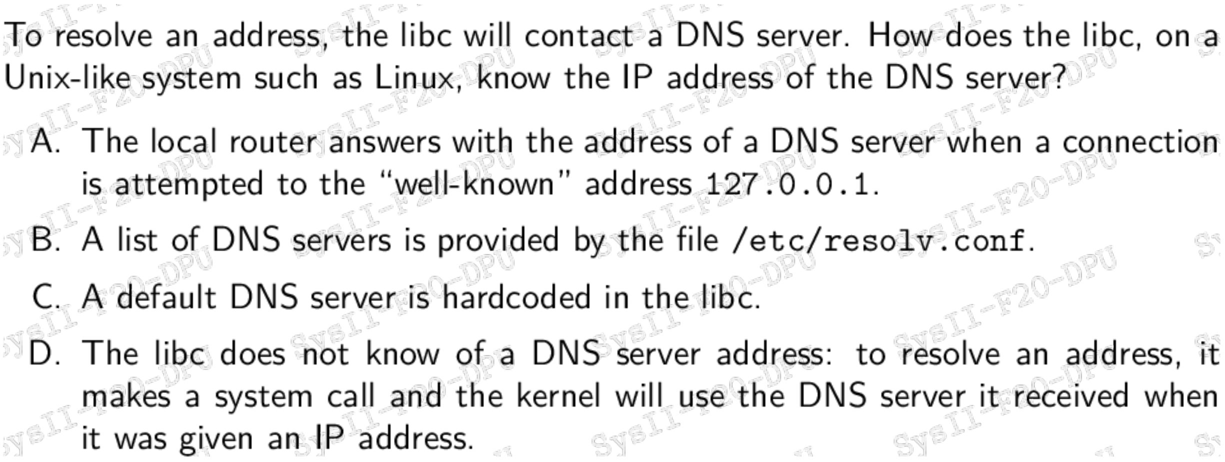
○ Answer B.

○ Answer C.

✓ ○ Answer D.

▷ View Feedback

Question 4 1 / 1 point



To resolve an address, the libc will contact a DNS server. How does the libc, on a Unix-like system such as Linux, know the IP address of the DNS server?

A. The local router answers with the address of a DNS server when a connection is attempted to the "well-known" address `127.0.0.1`.

B. A list of DNS servers is provided by the file `/etc/resolv.conf`.

C. A default DNS server is hardcoded in the libc.

D. The libc does not know of a DNS server address: to resolve an address, it makes a system call and the kernel will use the DNS server it received when it was given an IP address.

○ Answer A.

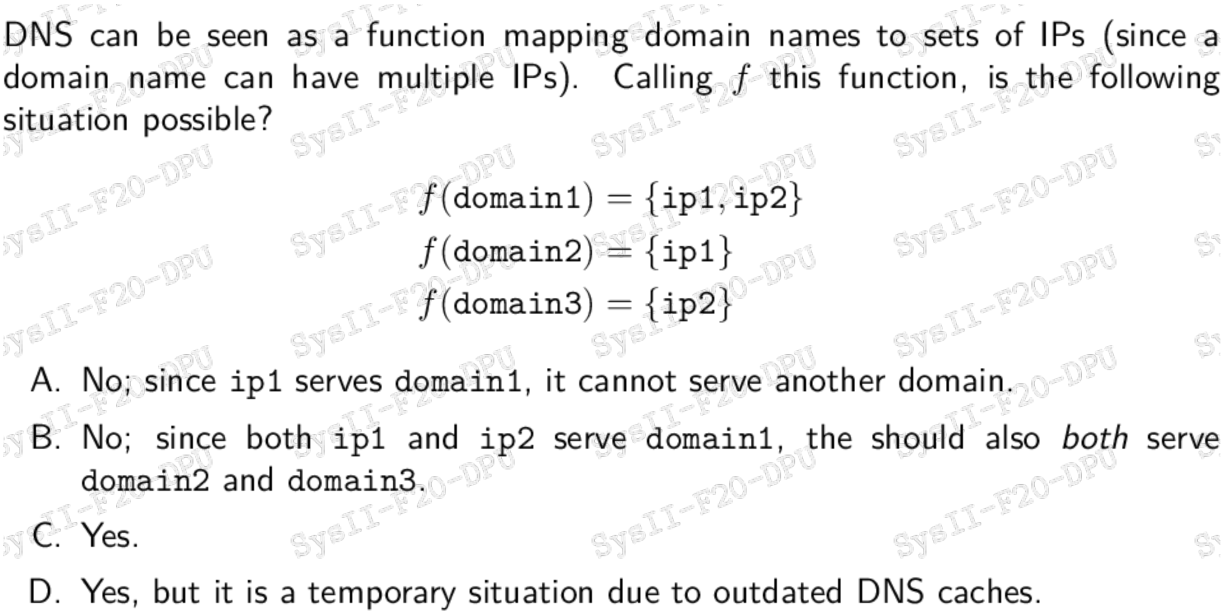
✓ ○ Answer B.

○ Answer C.

○ Answer D.

▷ View Feedback

Question 5 1 / 1 point



DNS can be seen as a function mapping domain names to sets of IPs (since a domain name can have multiple IPs). Calling f this function, is the following situation possible?

$$f(\texttt{domain1}) = \{\texttt{ip1}, \texttt{ip2}\}$$
$$f(\texttt{domain2}) = \{\texttt{ip1}\}$$
$$f(\texttt{domain3}) = \{\texttt{ip2}\}$$

A. No; since `ip1` serves `domain1`, it cannot serve another domain.

B. No; since both `ip1` and `ip2` serve `domain1`, the should also *both* serve `domain2` and `domain3`.

C. Yes.

D. Yes, but it is a temporary situation due to outdated DNS caches.

○ Answer A.

○ Answer B.

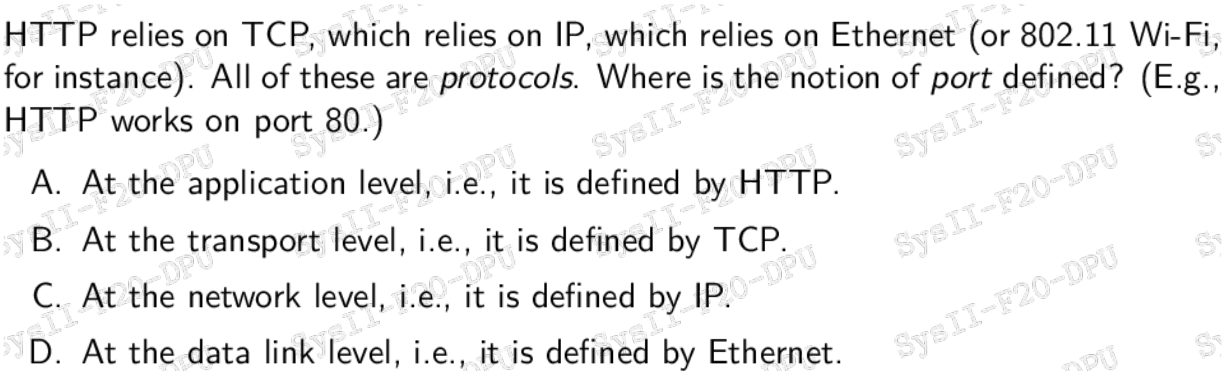
✓ ○ Answer C.

○ Answer D.

▷ View Feedback

Question 6

1 / 1 point



HTTP relies on TCP, which relies on IP, which relies on Ethernet (or 802.11 Wi-Fi, for instance). All of these are *protocols*. Where is the notion of *port* defined? (E.g., HTTP works on port 80.)

A. At the application level, i.e., it is defined by HTTP.

B. At the transport level, i.e., it is defined by TCP.

C. At the network level, i.e., it is defined by IP.

D. At the data link level, i.e., it is defined by Ethernet.

○ Answer A.

✓ ○ Answer B.

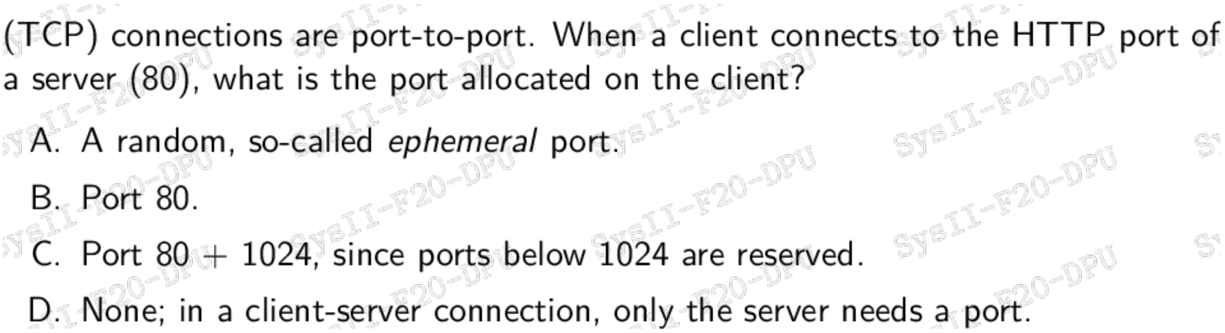
○ Answer C.

○ Answer D.

▷ View Feedback

Question 7

1 / 1 point



(TCP) connections are port-to-port. When a client connects to the HTTP port of a server (80), what is the port allocated on the client?

A. A random, so-called *ephemeral* port.

B. Port 80.

C. Port 80 + 1024, since ports below 1024 are reserved.

D. None; in a client-server connection, only the server needs a port.

✓ ○ Answer A.

○ Answer B.

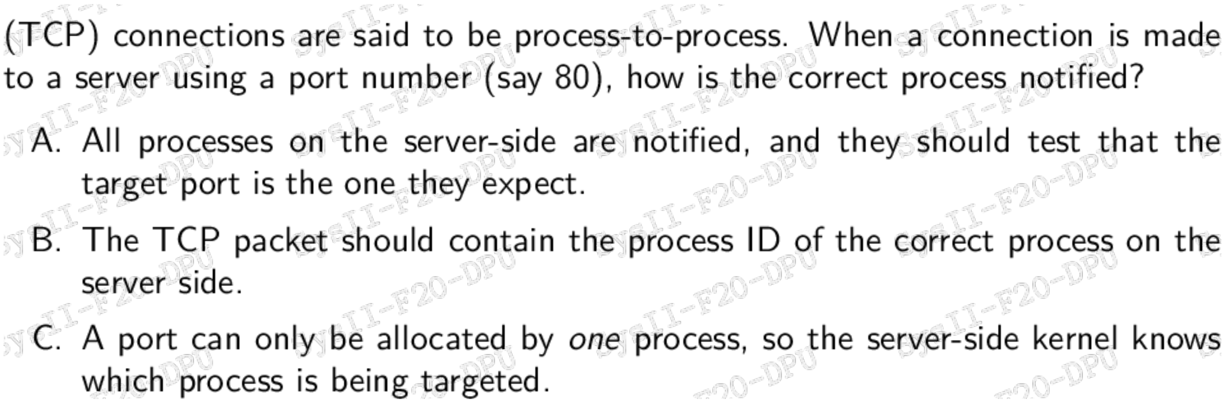
○ Answer C.

○ Answer D.

▷ View Feedback

Question 8

1 / 1 point



(TCP) connections are said to be process-to-process. When a connection is made to a server using a port number (say 80), how is the correct process notified?

A. All processes on the server-side are notified, and they should test that the target port is the one they expect.

B. The TCP packet should contain the process ID of the correct process on the server side.

C. A port can only be allocated by *one* process, so the server-side kernel knows which process is being targeted.

○ Answer A.

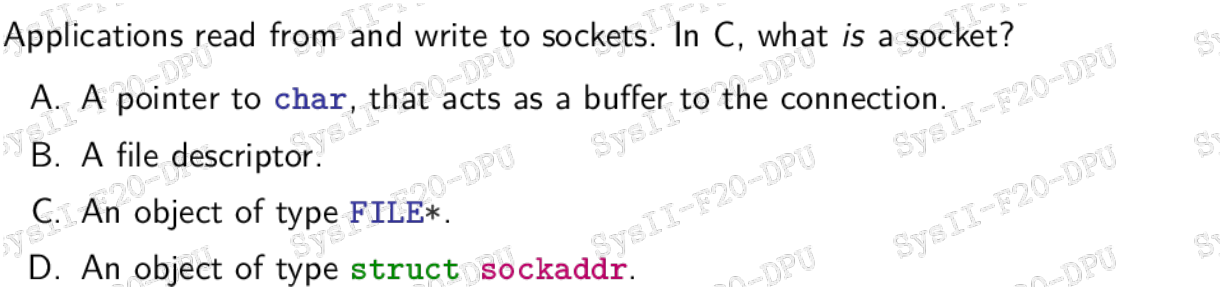
○ Answer B.

✓ ○ Answer C.

▷ View Feedback

Question 9

1 / 1 point



Applications read from and write to sockets. In C, what *is* a socket?

A. A pointer to `char`, that acts as a buffer to the connection.

B. A file descriptor.

C. An object of type `FILE*`.

D. An object of type `struct sockaddr`.

○ Answer A.

✓ ○ Answer B.

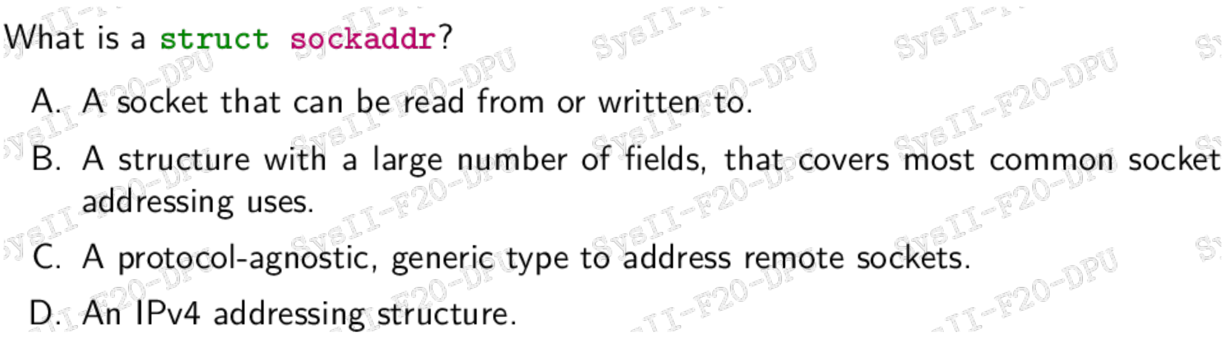
○ Answer C.

○ Answer D.

▷ View Feedback

Question 10

1 / 1 point



What is a `struct sockaddr`?

A. A socket that can be read from or written to.

B. A structure with a large number of fields, that covers most common socket addressing uses.

C. A protocol-agnostic, generic type to address remote sockets.

D. An IPv4 addressing structure.

○ Answer A.

○ Answer B.

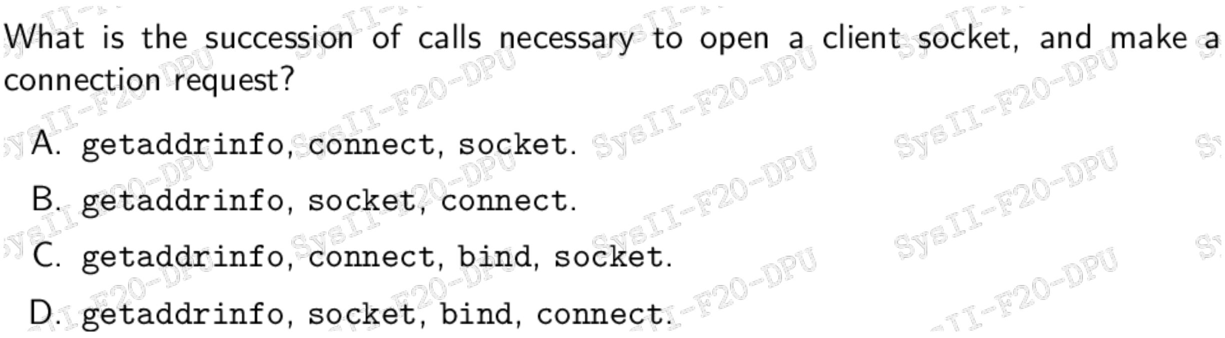
✓ ○ Answer C.

○ Answer D.

▷ View Feedback

Question 11

1 / 1 point



What is the succession of calls necessary to open a client socket, and make a connection request?

A. `getaddrinfo`, `connect`, `socket`.

B. `getaddrinfo`, `socket`, `connect`.

C. `getaddrinfo`, `connect`, `bind`, `socket`.

D. `getaddrinfo`, `socket`, `bind`, `connect`.

○ Answer A.

✓ ○ Answer B.

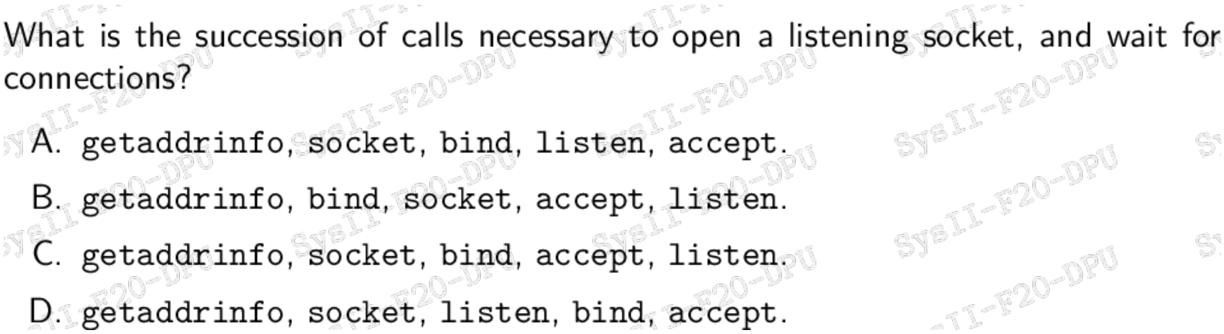
○ Answer C.

○ Answer D.

▷ View Feedback

Question 12

1 / 1 point



What is the succession of calls necessary to open a listening socket, and wait for connections?

A. `getaddrinfo, socket, bind, listen, accept.`

B. `getaddrinfo, bind, socket, accept, listen.`

C. `getaddrinfo, socket, bind, accept, listen.`

D. `getaddrinfo, socket, listen, bind, accept.`

✓ ○ Answer A.

○ Answer B.

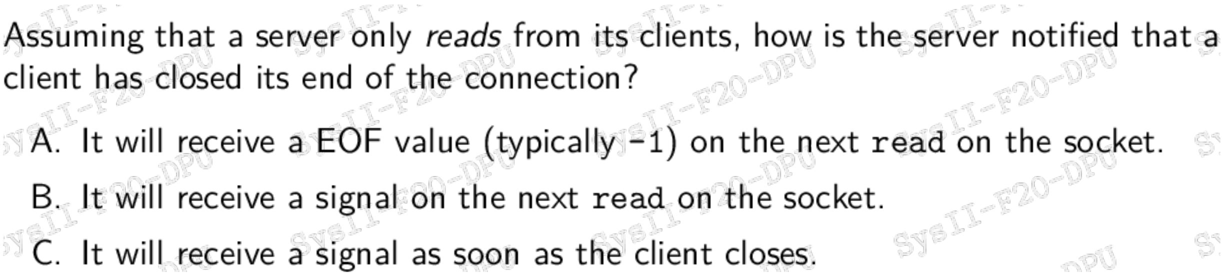
○ Answer C.

○ Answer D.

▷ View Feedback

Question 13

1 / 1 point



Assuming that a server only *reads* from its clients, how is the server notified that a client has closed its end of the connection?

A. It will receive a EOF value (typically `-1`) on the next `read` on the socket.

B. It will receive a signal on the next `read` on the socket.

C. It will receive a signal as soon as the client closes.

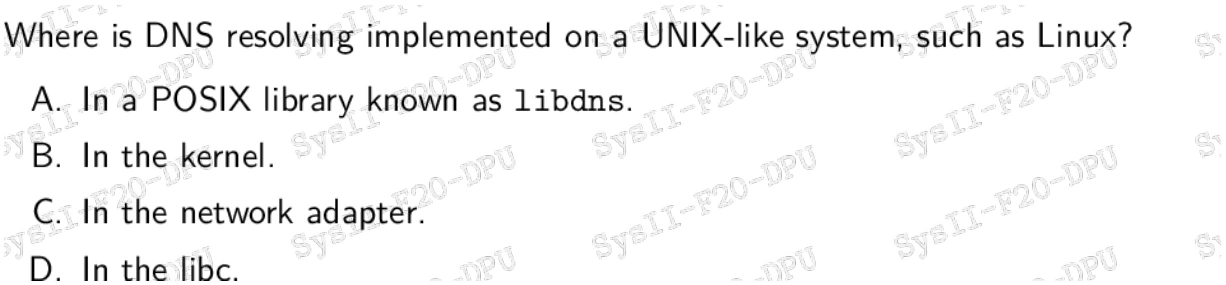
✓ ○ Answer A.

○ Answer B.

○ Answer C.

▷ View Feedback

Question 14 1 / 1 point



Where is DNS resolving implemented on a UNIX-like system, such as Linux?

A. In a POSIX library known as `libdns`.

B. In the kernel.

C. In the network adapter.

D. In the libc.

○ Answer A.

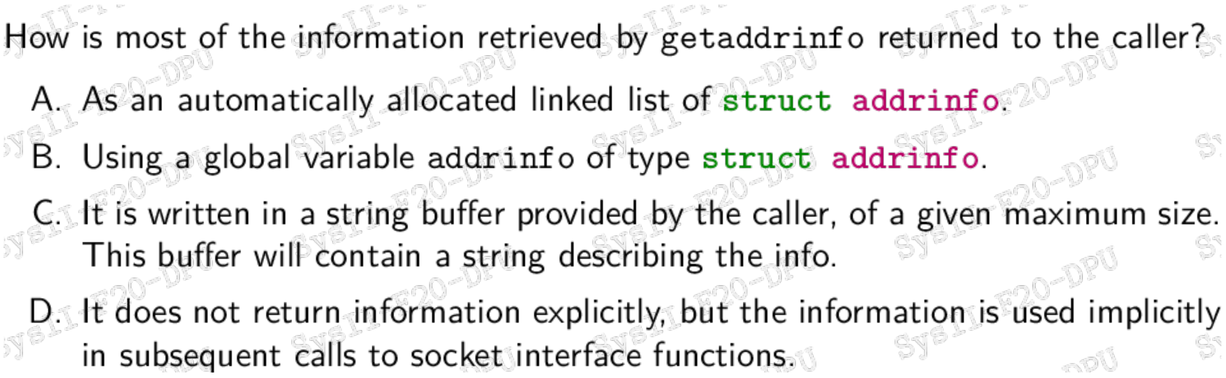
○ Answer B.

○ Answer C.

✓ ○ Answer D.

▷ View Feedback

Question 15 1 / 1 point



How is most of the information retrieved by `getaddrinfo` returned to the caller?

A. As an automatically allocated linked list of `struct addrinfo`.

B. Using a global variable `addrinfo` of type `struct addrinfo`.

C. It is written in a string buffer provided by the caller, of a given maximum size. This buffer will contain a string describing the info.

D. It does not return information explicitly, but the information is used implicitly in subsequent calls to socket interface functions.

✓ ○ Answer A.

○ Answer B.

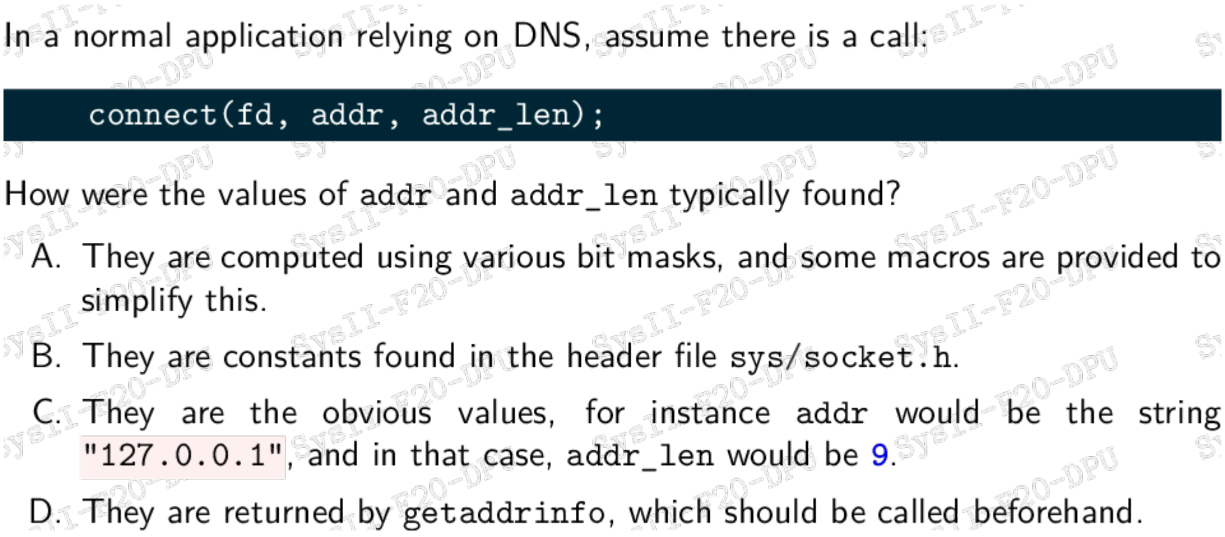
○ Answer C.

○ Answer D.

▷ View Feedback

Question 16

1 / 1 point



In a normal application relying on DNS, assume there is a call:

```
    connect(fd, addr, addr_len);
```

How were the values of `addr` and `addr_len` typically found?

A. They are computed using various bit masks, and some macros are provided to simplify this.

B. They are constants found in the header file `sys/socket.h`.

C. They are the obvious values, for instance `addr` would be the string `"127.0.0.1"`, and in that case, `addr_len` would be `9`.

D. They are returned by `getaddrinfo`, which should be called beforehand.

○ Answer A.

○ Answer B.

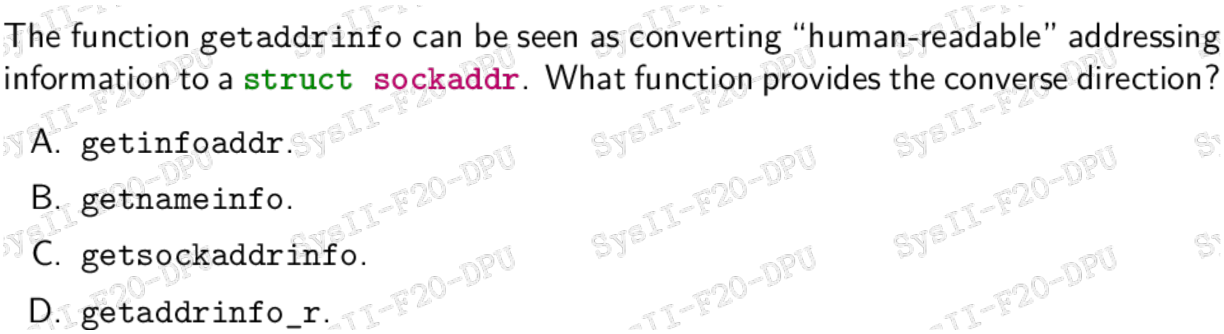
○ Answer C.

✓ ○ Answer D.

▷ View Feedback

Question 17

1 / 1 point



The function `getaddrinfo` can be seen as converting “human-readable” addressing information to a `struct sockaddr`. What function provides the converse direction?

A. `getinfoaddr`.

B. `getnameinfo`.

C. `getsockaddrinfo`.

D. `getaddrinfo_r`.

○ Answer A.

✓ ○ Answer B.

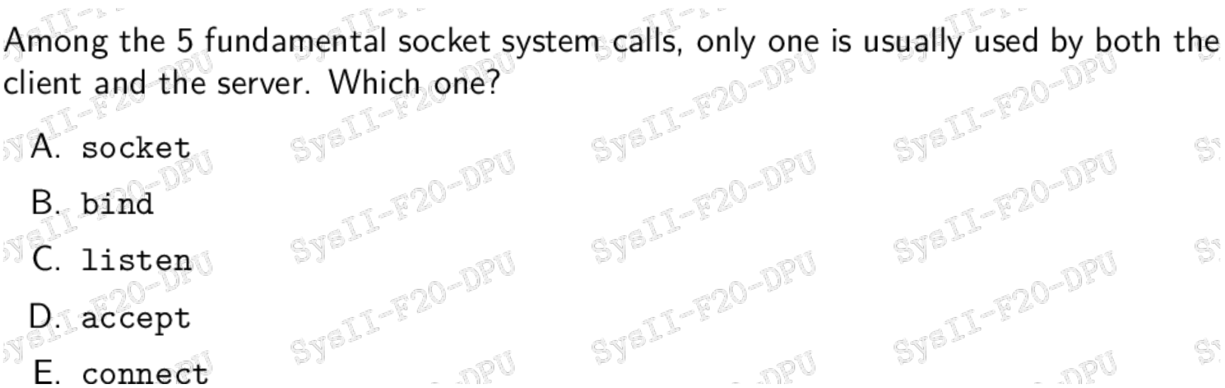
○ Answer C.

○ Answer D.

▷ View Feedback

Question 18

1 / 1 point



Among the 5 fundamental socket system calls, only one is usually used by both the client and the server. Which one?

A. `socket`
B. `bind`
C. `listen`
D. `accept`
E. `connect`

✓ ○ Answer A.

○ Answer B.

○ Answer C.

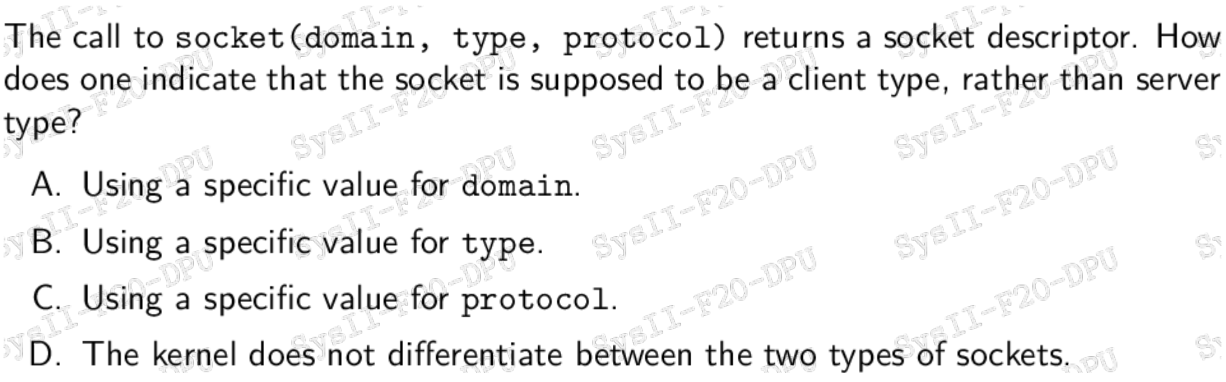
○ Answer D.

○ Answer E.

▷ View Feedback

Question 19

1 / 1 point



The call to `socket(domain, type, protocol)` returns a socket descriptor. How does one indicate that the socket is supposed to be a client type, rather than server type?

A. Using a specific value for `domain`.
B. Using a specific value for `type`.
C. Using a specific value for `protocol`.
D. The kernel does not differentiate between the two types of sockets.

○ Answer A.

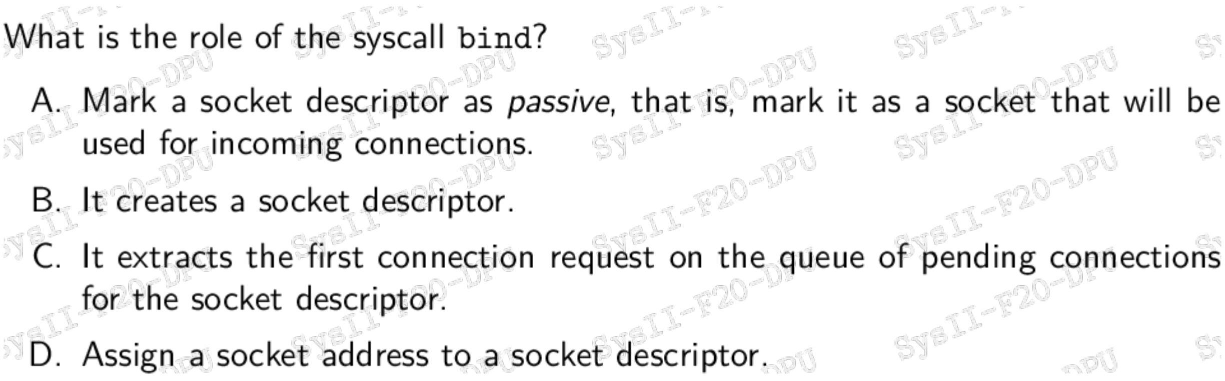
○ Answer B.

○ Answer C.

✓ ○ Answer D.

▷ View Feedback

Question 20 1 / 1 point



What is the role of the syscall `bind`?

A. Mark a socket descriptor as *passive*, that is, mark it as a socket that will be used for incoming connections.

B. It creates a socket descriptor.

C. It extracts the first connection request on the queue of pending connections for the socket descriptor.

D. Assign a socket address to a socket descriptor.

○ Answer A.

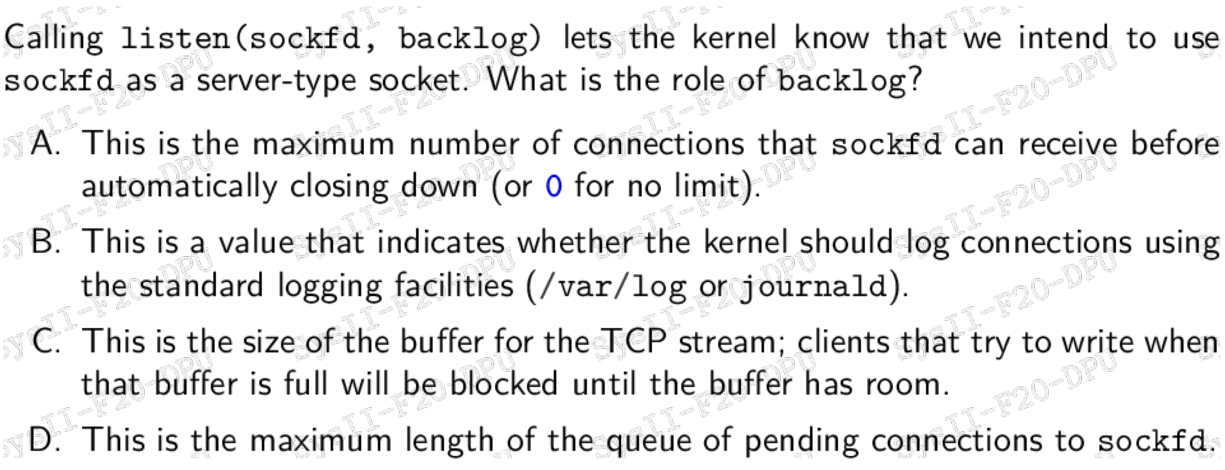
○ Answer B.

○ Answer C.

✓ ○ Answer D.

▷ View Feedback

Question 21 1 / 1 point



Calling `listen(sockfd, backlog)` lets the kernel know that we intend to use `sockfd` as a server-type socket. What is the role of `backlog`?

A. This is the maximum number of connections that `sockfd` can receive before automatically closing down (or `0` for no limit).

B. This is a value that indicates whether the kernel should log connections using the standard logging facilities (`/var/log` or `journald`).

C. This is the size of the buffer for the TCP stream; clients that try to write when that buffer is full will be blocked until the buffer has room.

D. This is the maximum length of the queue of pending connections to `sockfd`.

○ Answer A.

○ Answer B.

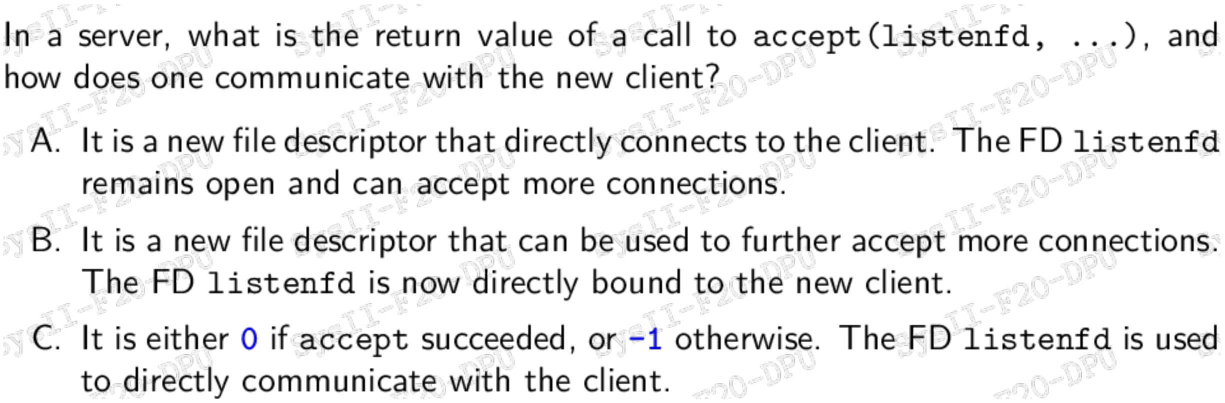
○ Answer C.

✓ ○ Answer D.

▷ View Feedback

Question 22

1 / 1 point



In a server, what is the return value of a call to `accept(listenfd, ...)`, and how does one communicate with the new client?

A. It is a new file descriptor that directly connects to the client. The FD `listenfd` remains open and can accept more connections.

B. It is a new file descriptor that can be used to further accept more connections. The FD `listenfd` is now directly bound to the new client.

C. It is either `0` if `accept` succeeded, or `-1` otherwise. The FD `listenfd` is used to directly communicate with the client.

✓ ○ Answer A.

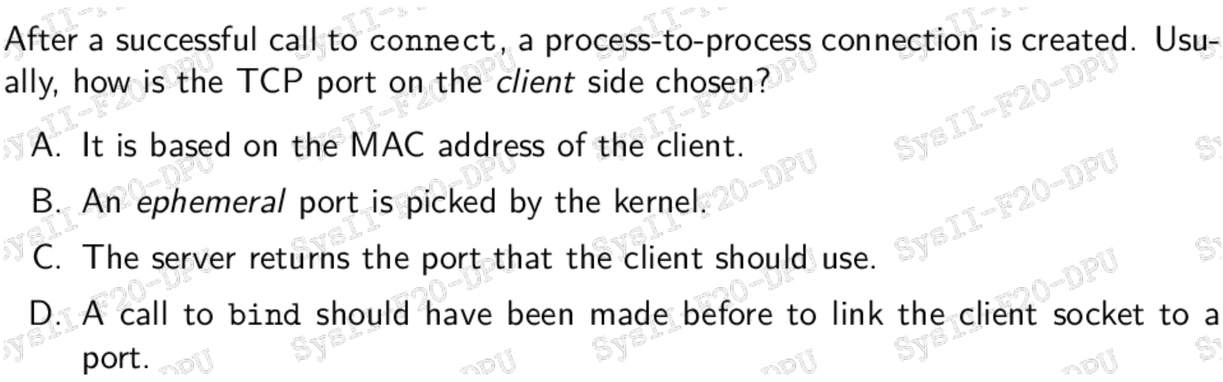
○ Answer B.

○ Answer C.

▷ View Feedback

Question 23

1 / 1 point


After a successful call to `connect`, a process-to-process connection is created. Usually, how is the TCP port on the *client* side chosen?

A. It is based on the MAC address of the client.

B. An *ephemeral* port is picked by the kernel.

C. The server returns the port that the client should use.

D. A call to `bind` should have been made before to link the client socket to a port.

○ Answer A.

✓ ○ Answer B.

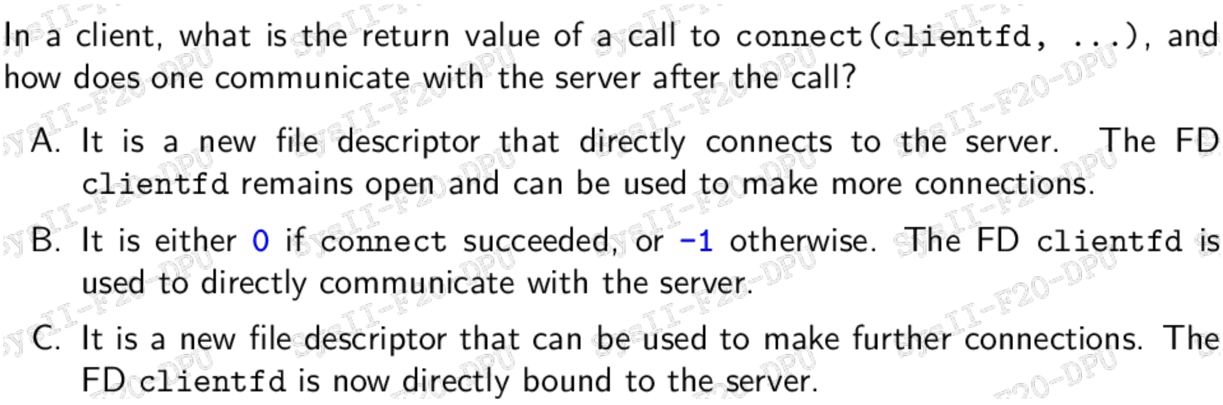
○ Answer C.

○ Answer D.

▷ View Feedback

Question 24

1 / 1 point


In a client, what is the return value of a call to `connect(clientfd, ...)`, and how does one communicate with the server after the call?

A. It is a new file descriptor that directly connects to the server. The FD `clientfd` remains open and can be used to make more connections.

B. It is either `0` if `connect` succeeded, or `-1` otherwise. The FD `clientfd` is used to directly communicate with the server.

C. It is a new file descriptor that can be used to make further connections. The FD `clientfd` is now directly bound to the server.

○ Answer A.

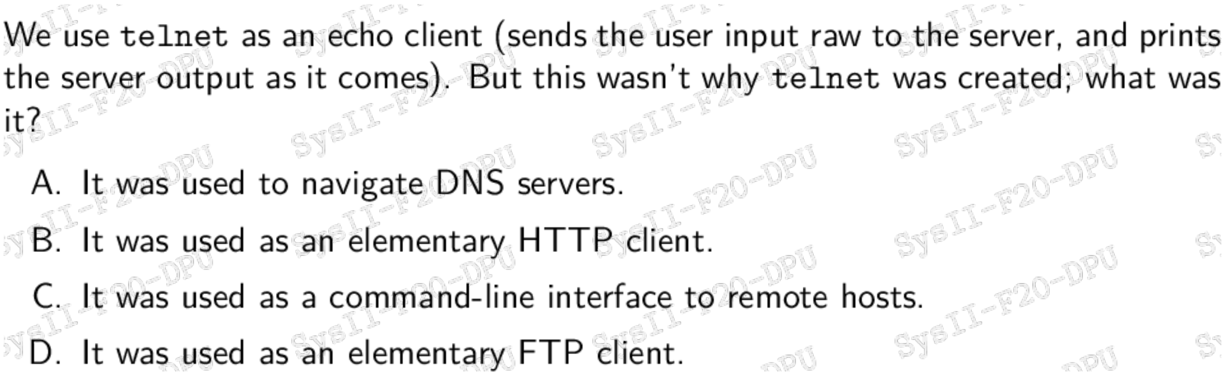
✓ ○ Answer B.

○ Answer C.

▷ View Feedback

Question 25

1 / 1 point



We use `telnet` as an echo client (sends the user input raw to the server, and prints the server output as it comes). But this wasn't why `telnet` was created; what was it?

A. It was used to navigate DNS servers.

B. It was used as an elementary HTTP client.

C. It was used as a command-line interface to remote hosts.

D. It was used as an elementary FTP client.

○ Answer A.

○ Answer B.

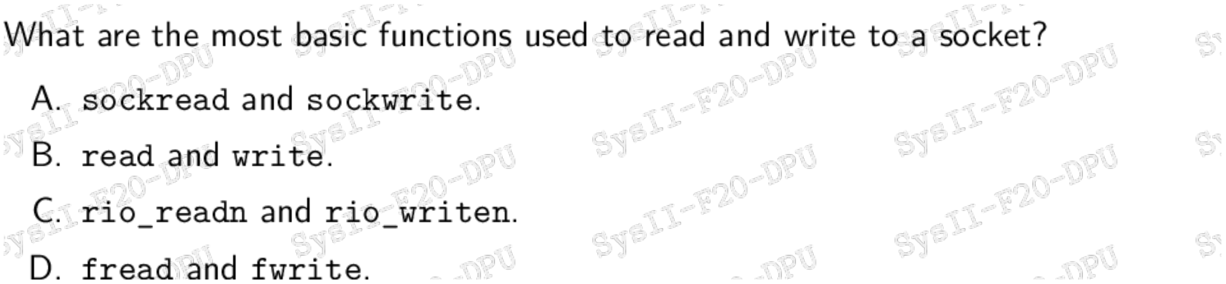
✓ ○ Answer C.

○ Answer D.

▷ View Feedback

Question 26

1 / 1 point



What are the most basic functions used to read and write to a socket?

A. `sockread` and `sockwrite`.

B. `read` and `write`.

C. `rio_readn` and `rio_writen`.

D. `fread` and `fwrite`.

○ Answer A.

✓ ○ Answer B.

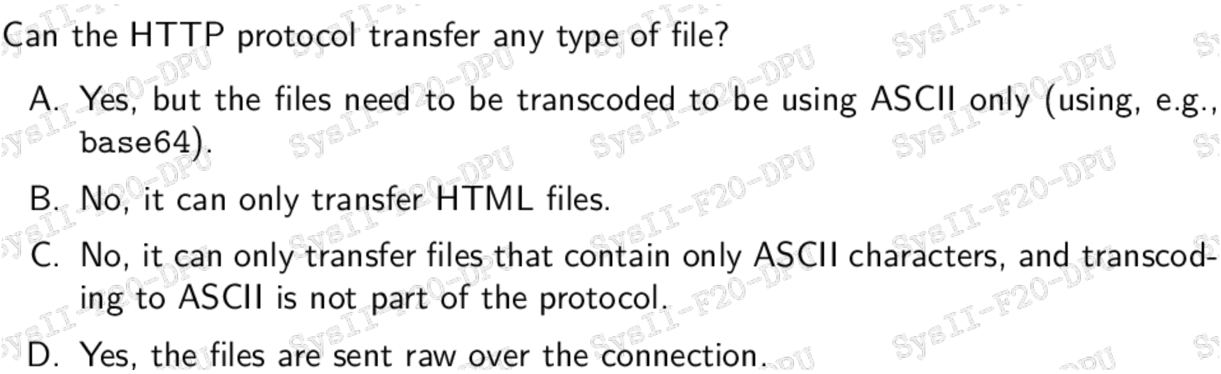
○ Answer C.

○ Answer D.

▷ View Feedback

Question 27

1 / 1 point



Can the HTTP protocol transfer any type of file?

A. Yes, but the files need to be transcoded to be using ASCII only (using, e.g., `base64`).

B. No, it can only transfer HTML files.

C. No, it can only transfer files that contain only ASCII characters, and transcoding to ASCII is not part of the protocol.

D. Yes, the files are sent raw over the connection.

○ Answer A.

○ Answer B.

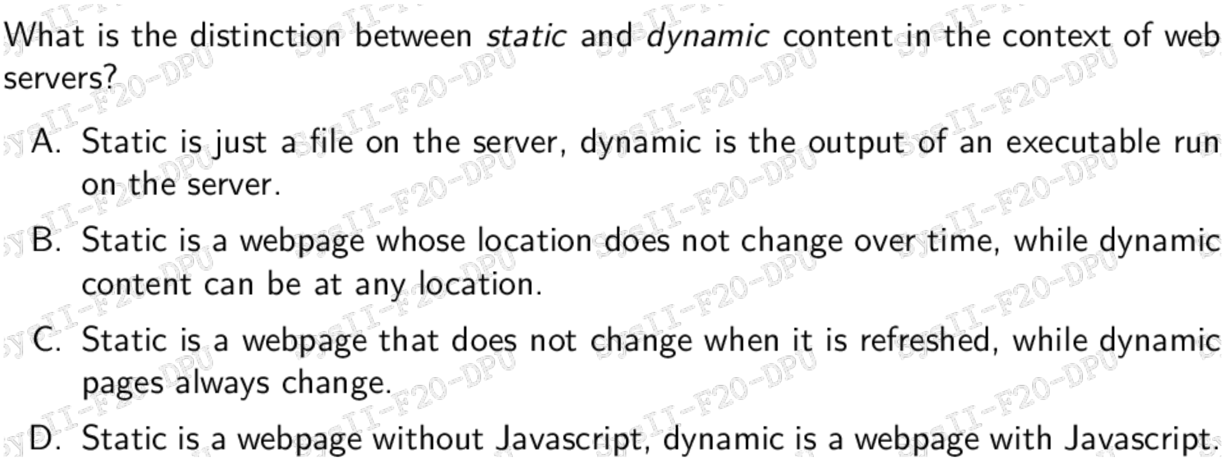
○ Answer C.

✓ ○ Answer D.

▷ View Feedback

Question 28

1 / 1 point



What is the distinction between *static* and *dynamic* content in the context of web servers?

A. Static is just a file on the server, dynamic is the output of an executable run on the server.

B. Static is a webpage whose location does not change over time, while dynamic content can be at any location.

C. Static is a webpage that does not change when it is refreshed, while dynamic pages always change.

D. Static is a webpage without Javascript, dynamic is a webpage with Javascript.

✓ ○ Answer A.

○ Answer B.

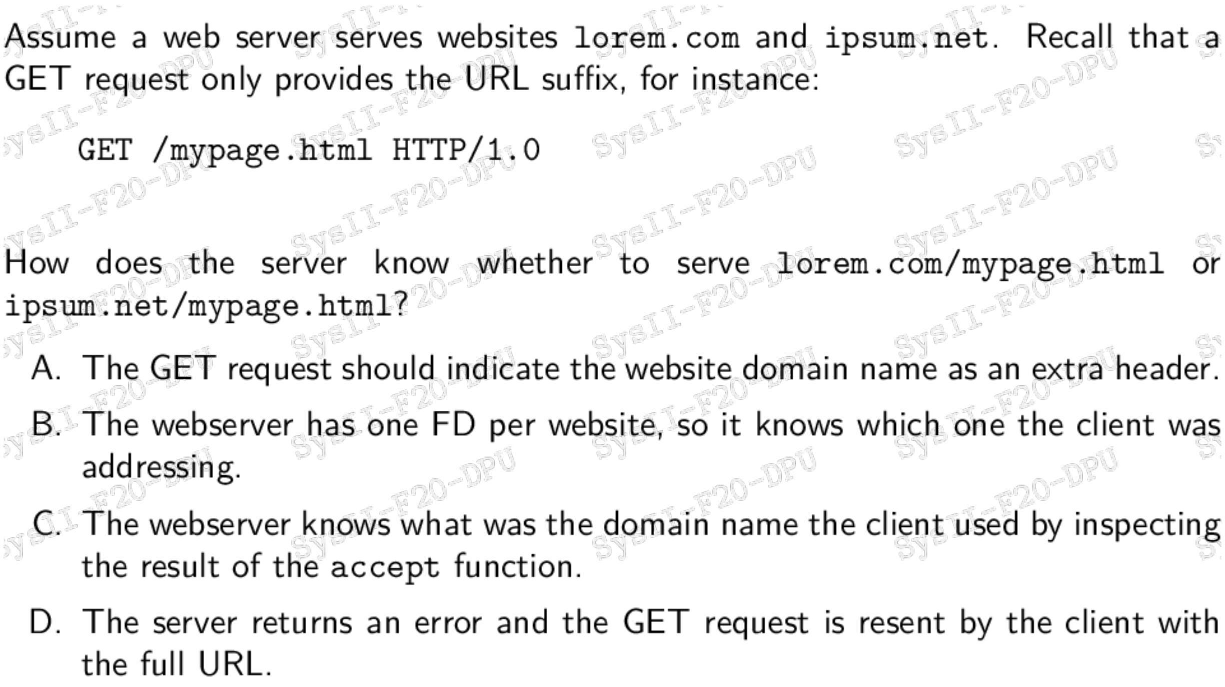
○ Answer C.

○ Answer D.

▷ View Feedback

Question 29

1 / 1 point



Assume a web server serves websites `lorem.com` and `ipsum.net`. Recall that a GET request only provides the URL suffix, for instance:

```
    GET /mypage.html HTTP/1.0
```

How does the server know whether to serve `lorem.com/mypage.html` or `ipsum.net/mypage.html`?

A. The GET request should indicate the website domain name as an extra header.

B. The webserver has one FD per website, so it knows which one the client was addressing.

C. The webserver knows what was the domain name the client used by inspecting the result of the `accept` function.

D. The server returns an error and the GET request is resent by the client with the full URL.

✓ ○ Answer A.

○ Answer B.

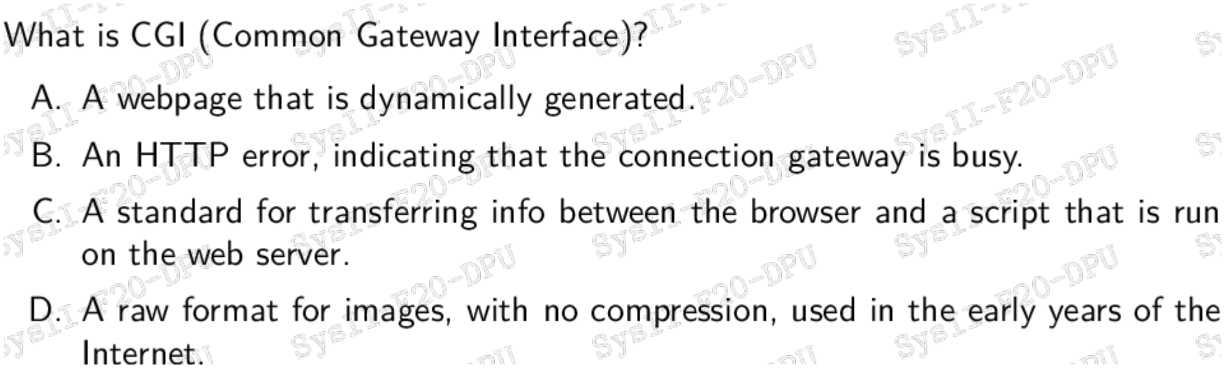
○ Answer C.

○ Answer D.

▷ View Feedback

Question 30

1 / 1 point



What is CGI (Common Gateway Interface)?

A. A webpage that is dynamically generated.

B. An HTTP error, indicating that the connection gateway is busy.

C. A standard for transferring info between the browser and a script that is run on the web server.

D. A raw format for images, with no compression, used in the early years of the Internet.

○ Answer A.

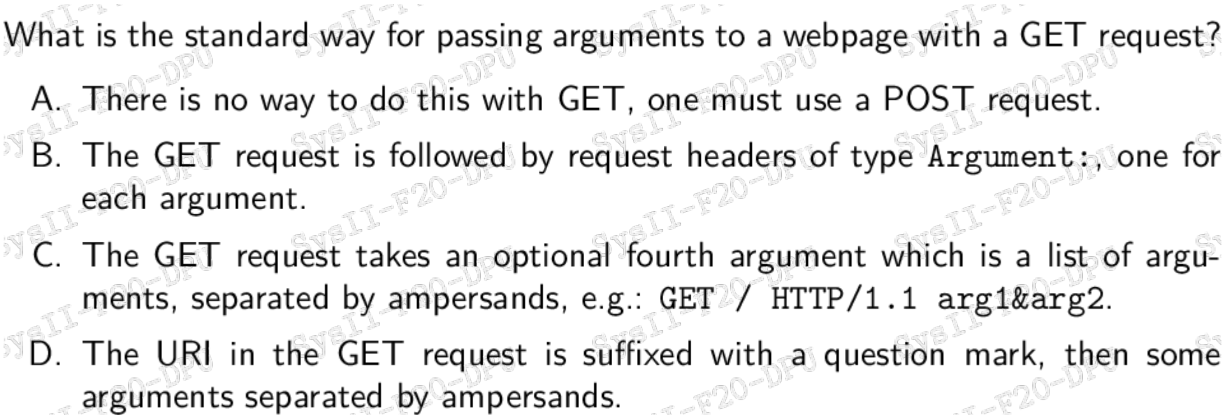
○ Answer B.

✓ ○ Answer C.

○ Answer D.

▷ View Feedback

Question 31 1 / 1 point



What is the standard way for passing arguments to a webpage with a GET request?

A. There is no way to do this with GET, one must use a POST request.

B. The GET request is followed by request headers of type `Argument:`, one for each argument.

C. The GET request takes an optional fourth argument which is a list of arguments, separated by ampersands, e.g.: `GET / HTTP/1.1 arg1&arg2`.

D. The URI in the GET request is suffixed with a question mark, then some arguments separated by ampersands.

○ Answer A.

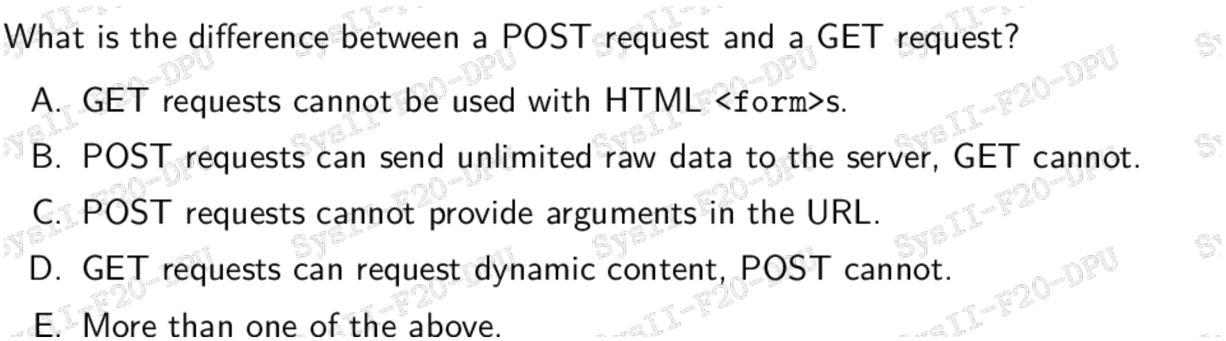
○ Answer B.

○ Answer C.

✓ ○ Answer D.

▷ View Feedback

Question 32 1 / 1 point



What is the difference between a POST request and a GET request?

A. GET requests cannot be used with HTML `<form>`s.

B. POST requests can send unlimited raw data to the server, GET cannot.

C. POST requests cannot provide arguments in the URL.

D. GET requests can request dynamic content, POST cannot.

E. More than one of the above.

○ Answer A.

✓ ○ Answer B.

○ Answer C.

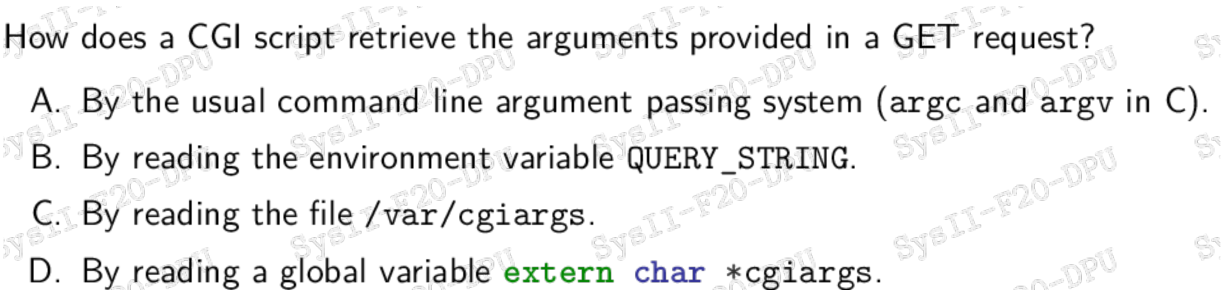
○ Answer D.

○ Answer E.

▷ View Feedback

Question 33

1 / 1 point



How does a CGI script retrieve the arguments provided in a GET request?

A. By the usual command line argument passing system (`argc` and `argv` in C).

B. By reading the environment variable `QUERY_STRING`.

C. By reading the file `/var/cgiargs`.

D. By reading a global variable `extern char *cgiargs`.

○ Answer A.

✓ ○ Answer B.

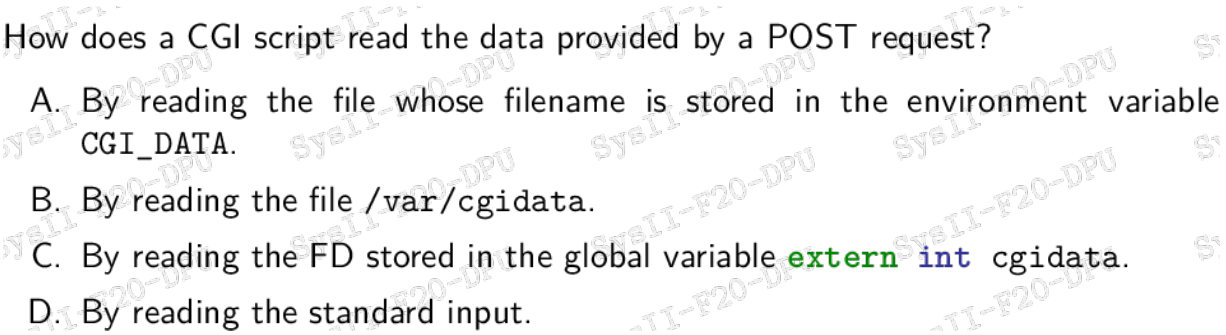
○ Answer C.

○ Answer D.

▷ View Feedback

Question 34

1 / 1 point



How does a CGI script read the data provided by a POST request?

A. By reading the file whose filename is stored in the environment variable `CGI_DATA`.

B. By reading the file `/var/cgidata`.

C. By reading the FD stored in the global variable `extern int cgidata`.

D. By reading the standard input.

○ Answer A.

○ Answer B.

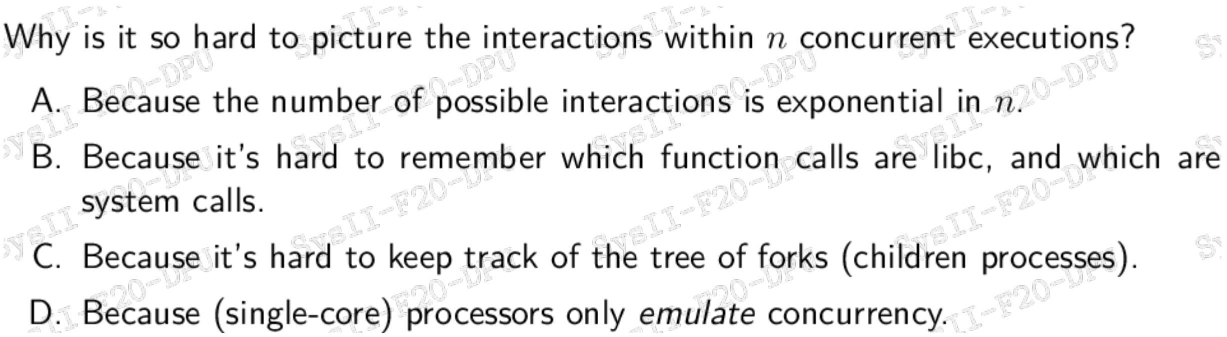
○ Answer C.

✓ ○ Answer D.

▷ View Feedback

Concurrent Programming

Question 35 1 / 1 point



Why is it so hard to picture the interactions within n concurrent executions?

A. Because the number of possible interactions is exponential in n.

B. Because it's hard to remember which function calls are libc, and which are system calls.

C. Because it's hard to keep track of the tree of forks (children processes).

D. Because (single-core) processors only *emulate* concurrency.

✓ ○ Answer A.

○ Answer B.

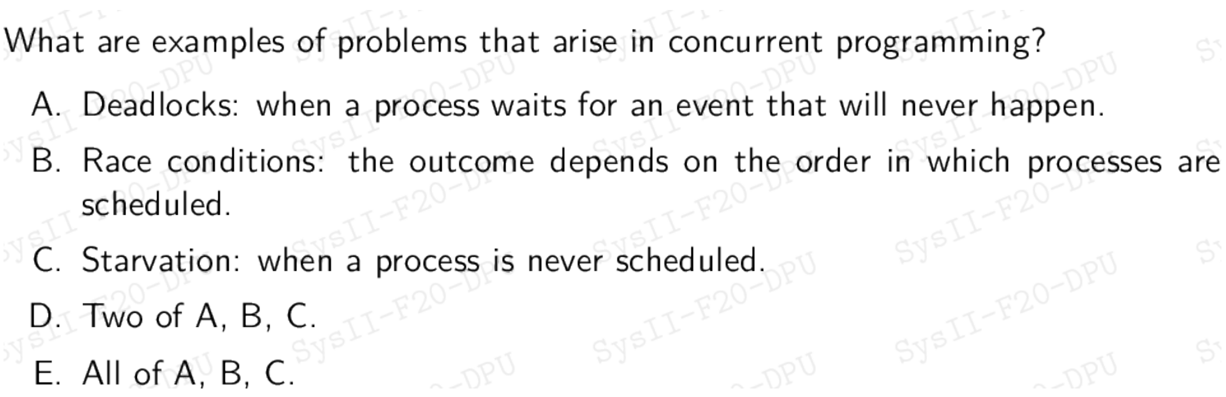
○ Answer C.

○ Answer D.

▷ View Feedback

Question 36

1 / 1 point



What are examples of problems that arise in concurrent programming?

A. Deadlocks: when a process waits for an event that will never happen.

B. Race conditions: the outcome depends on the order in which processes are scheduled.

C. Starvation: when a process is never scheduled.

D. Two of A, B, C.

E. All of A, B, C.

○ Answer A.

○ Answer B.

○ Answer C.

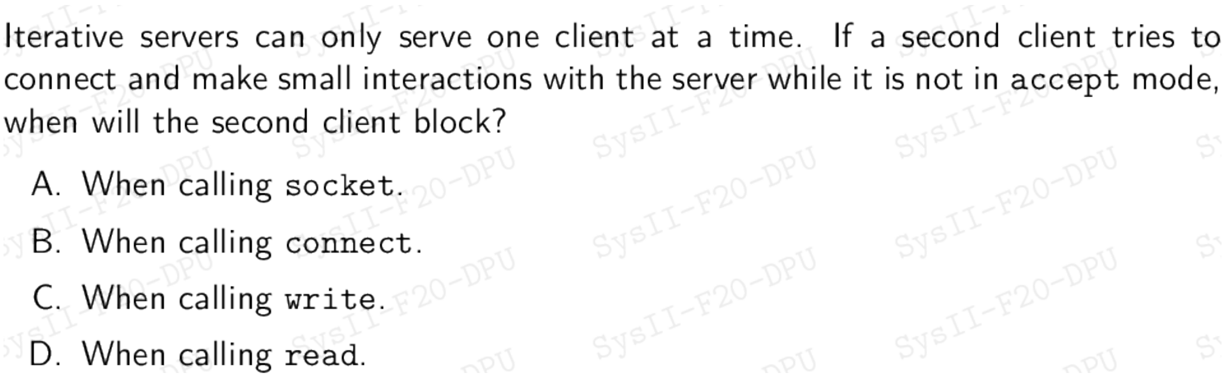
○ Answer D.

✓ ○ Answer E.

▷ View Feedback

Question 37

1 / 1 point



Iterative servers can only serve one client at a time. If a second client tries to connect and make small interactions with the server while it is not in `accept` mode, when will the second client block?

A. When calling `socket`.

B. When calling `connect`.

C. When calling `write`.

D. When calling `read`.

○ Answer A.

○ Answer B.

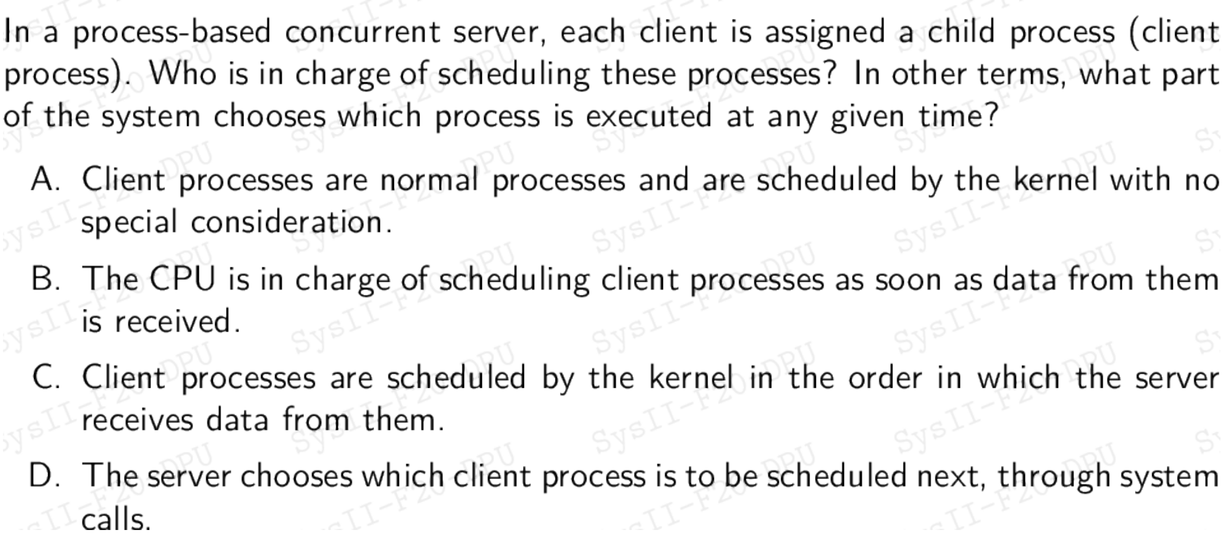
○ Answer C.

✓ ○ Answer D.

▷ View Feedback

Question 38

1 / 1 point



In a process-based concurrent server, each client is assigned a child process (client process). Who is in charge of scheduling these processes? In other terms, what part of the system chooses which process is executed at any given time?

A. Client processes are normal processes and are scheduled by the kernel with no special consideration.

B. The CPU is in charge of scheduling client processes as soon as data from them is received.

C. Client processes are scheduled by the kernel in the order in which the server receives data from them.

D. The server chooses which client process is to be scheduled next, through system calls.

✓ ○ Answer A.

○ Answer B.

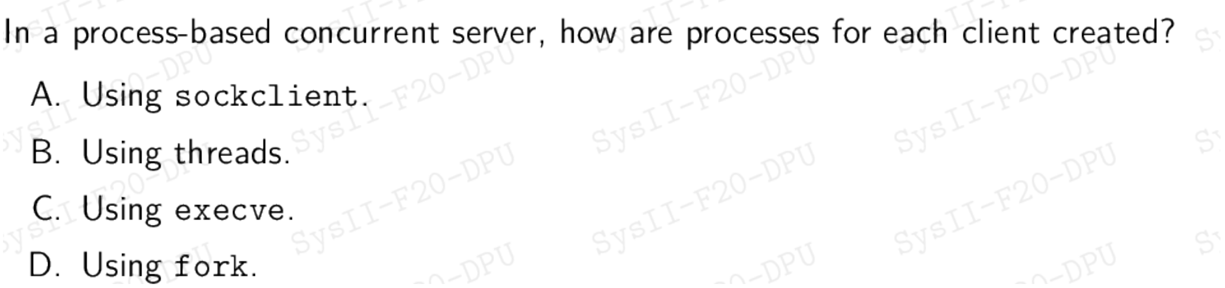
○ Answer C.

○ Answer D.

▷ View Feedback

Question 39

1 / 1 point



In a process-based concurrent server, how are processes for each client created?

A. Using `sockclient`.

B. Using threads.

C. Using `execve`.

D. Using `fork`.

○ Answer A.

○ Answer B.

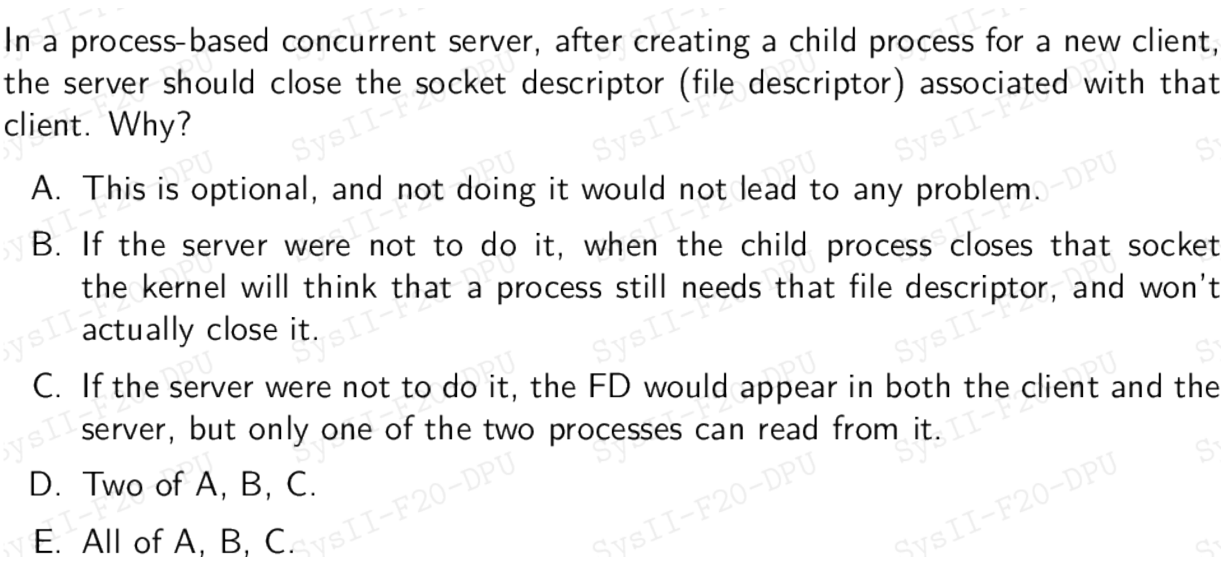
○ Answer C.

✓ ○ Answer D.

▷ View Feedback

Question 40

1 / 1 point



In a process-based concurrent server, after creating a child process for a new client, the server should close the socket descriptor (file descriptor) associated with that client. Why?

A. This is optional, and not doing it would not lead to any problem.

B. If the server were not to do it, when the child process closes that socket the kernel will think that a process still needs that file descriptor, and won't actually close it.

C. If the server were not to do it, the FD would appear in both the client and the server, but only one of the two processes can read from it.

D. Two of A, B, C.

E. All of A, B, C.

○ Answer A.

✓ ○ Answer B.

○ Answer C.

○ Answer D.

○ Answer E.

▷ View Feedback

Attempt Score:40 / 40

Overall Grade (highest attempt)**:**40 / 40

Done