Program: BE

|  |  |  |  |
| --- | --- | --- | --- |
| Date | 29 April 2025 | Maximum Marks | 50 |
| Course Code | **CS362AI** | Duration | 90 min |
| 6th Sem | VI Semester | CIE-I | |
| **Network Programming and Security** | | | |

|  |  |  |
| --- | --- | --- |
|  | **QUIZ** |  |
| 1 | FIN\_WAIT\_2 | 1 |
| 2 | TIME\_WAIT | 1 |
| 3 | ESTABLISHED | 1 |
| 4 | No, unless TCP keep-alive is enabled and expires | 1 |
| 5 | Because the local machine (server) was the one to send the last ACK after the client's FIN. | 1 |
| 6 | No. | 1 |
| 7 | **Parent closes** the connected socket; **child closes** the listening socket after fork() | 2 |
| 8 | getpeername() returns the server’s address info after a successful connect() | 2 |
| **Sl. No.** | **Test Questions** | **M** |
| 1a | TCP Connection Termination | GeeksforGeeks | **5** |
| 1b | The 32-bit TCP sequence number field allows tracking up to 4 GB of data, but it’s still sufficient for large data transfers because TCP handles **wrap-around** using modulo 2³² arithmetic. Even if the **initial sequence number (ISN)** is randomly chosen near 2³² - 1, and wrap-around happens quickly, | **5** |
| 2a | * **TIME\_WAIT ensures proper connection termination** by waiting for any delayed packets to arrive. * **2 RTT timeout accounts for delayed packets**, ensuring the final ACK is received before termination. * **TIME\_WAIT prevents reincarnation issues** by ensuring old packets don’t interfere with new connections. * **TIME\_WAIT guarantees reliable cleanup**, clearing any remaining connection state and sequence numbers. | **5** |
| 2b | The **SYN** or **FIN** flag in TCP doesn't carry actual data but is treated as occupying one byte in the sequence number space. This ensures that **SYN** or **FIN** consumes a sequence number, making it clear when an acknowledgment is for the control flag (SYN/FIN) versus actual data. For example, if the **SYN** is sent with **Sequence Number = x**, the first data byte will start at **x + 1**, and the corresponding acknowledgment will be **x + 1**. This prevents ambiguity, ensuring that the receiver can distinguish between acknowledging the control flag and actual data. | **5** |
| 3a | * socket() * bind() * listen() * accept() * read() / recv() * write() / send() * close() |  |
| 3b | Program to be executed  X1.c  #include <stdio.h>  int main(int argc, char \*argv[])  {  printf("Inside child program:\n");  printf("Number of arguments: %d\n", argc);  for (int i = 0; i < argc; i++) {  printf("arg[%d]: %s\n", i, argv[i]);  }  return 0;  }  $ cc -o x1 x1.c  In main program create a child and execute x1 using exec  execl("x1", "x1", "Hello", "from", "exec", NULL); |  |
| 4a | * getsockname() returns the local IP address and port of the socket. * getpeername() returns the remote (peer) IP address and port of the socket. * They might return the same IP and port in a loopback (localhost) connection where both client and server run on the same machine, and the OS assigns the same port (rare but possible under specific conditions like binding explicitly to the same port). * Usually, local and peer ports differ even on localhost unless explicitly controlled. | **06** |
| 4b | In a concurrent server, the parent closes connectfd and the child closes listenfd after fork(), but the socket remains active as long as one process holds it open. TCP connection is terminated (FIN sent) only when all references to the socket are closed across all processes. | **05** |
| 5a | TCP Echo client and TCP Echo Server programs | **10** |

**Course Outcomes**

|  |  |
| --- | --- |
| CO1 | Explore the variety of network programming concepts and protocols. |
| CO2 | Analyse the interoperability of networking protocols and its usage. |
| CO3 | Design the client/server communication on Unix platforms. |
| CO4 | Investigate & Design the cryptographic algorithms to ensure secure transfer of secret keys and encryption/decryption of messages. |
| CO5 | Demonstrate Network Programming and Cryptographic algorithms to solve real-world problems. |

**Blooms’ taxonomy**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| L1 | L2 | L3 | L4 | L5 | L6 | CO1 | CO2 | CO3 | CO4 | CO5 | CO6 |
| 14 | 20 | 16 | - | - | - | 19 | 16 | 15 | - | - | - |