Operating Systems

Week 13 Recitation:
Exam 3 Preview – Review of Exam 3, Spring 2014

Paul Krzyzanowski
Rutgers University
Spring 2015

Question 1

A weakness of using NAND flash memory for use as a file system is:

(a) Stored data wears out over time, requiring periodic refreshing.
(b) Its read speeds are unpredictable, making I/O scheduling difficult.
(c) It has very long seek times compared to those of a disk.
(d) It supports a limited number of writes per block.

(a) Flash memory wears out through erase/writes, not because of inactivity
(b) Read speeds are constant
(c) Seek time = 0 – there is no overhead for seeking
(d) Yes. Writing requires an erase/write operation. Flash page wears out after 100,000 … 1,000,000 of these.

Question 2

Checkpointing in a log-structured file system such as YAFFS:

(a) Periodically writes updates of the file system's superblock.
(b) Writes data blocks and their associated metadata as a single transaction.
(c) Replicates data in the log for reliability.
(d) Avoids the need to reconstruct the file system hierarchy by reading all logs.

(a) No.
(b) No. That's journaling.
(c) No data replication in a log-structured file system.
(d) Yes. The set of logs *is* the file system. A directory hierarchy is created in memory by scanning logs. *Checkpointing* allows for this hierarchy to be written to the file system.

Question 3

A *loop device*:

(a) Enables the output of one device to be fed to the input of another.
(b) Echoes back whatever data is sent to it.
(c) Enables two local processes to communicate.
(d) Converts a file to a block device.

(a) No.
(b) No. Don't confuse *loop* with network *loopback*.
A loopback device is a simple driver that echoes data.
(c) No. This has nothing to do with inter-process communication.
(d) Best answer. It allows a file to be accessed as a block device.

Question 4

The *Address Resolution Protocol* (ARP) is used to:

(a) Enable a computer to get a unique IP address.
(b) Find the IP address that corresponds to a system's domain name.
(c) Resolve the path that a packet must take to reach a given IP address.
(d) Convert an IP address to the corresponding ethernet address.

(a) No. That's DHCP – the *Dynamic Host Configuration Protocol*
(We did not cover DHCP)
(b) No. That's a DNS query – the *Domain Name System*
(c) No. IP does not resolve a full path – a route just takes you to the next hop
(d) Yes. ARP maintains a cache of recent IP→ethernet MAC address matches. If it needs an address that is not in the cache, it sends an *ARP Query* broadcast message asking any system on the local area network that owns the requested IP address to respond with its ethernet MAC address.

Question 5

The *network device driver* is responsible for which layer of the OSI stack?

(a) Data link.
(b) Network.
(c) Transport.
(d) Presentation.

(a) Yes. The network device driver interacts with the device
Data Link layer = layer 2. Transfers data between nodes in a LAN
(b) No. Network = layer 3 – manages journey of packets (IP)
(c) No. Transport = layer 4 – application endpoints (TCP, UDP: sockets)
(d) No. Presentation = layer 6 – data representation

Question 6

Sockets are said to be compatible with files because:

(a) They are implemented under the VFS layer to support all file system operations.

(b) You can use read/write system calls with them.

(c) They appear in the file system name space.

(d) File data can be directed to a socket and vice versa.

(a) No. Sockets are not implemented as a file system under the VFS layer although sockets do implement a subset of VFS's file_operations

(b) Yes – they implement a subset of VFS

(c) No. Sockets do not appear in the file system name space.

(d) Not automatically but you can read data from a file and write it to a socket. That says nothing about compatibility.

Question 7

A *socket buffer*:

(a) Is similar to a buffer cache and stores frequently used network messages.

(b) Is a block of memory in the kernel that holds all the sockets in the operating system.

(c) Avoids the need to copy data as it moves up or down the network stack.

(d) Identifies the set of operations that can be performed on each socket.

(a) No. There is no such thing as "frequently used network messages."

(b) No. There isn't a single block of memory for all sockets. A slab allocator is usually used to allocate socket buffers but it's the same memory allocator that is used for many other kernel structures.

(c) Yes. A packet's data sits in the socket buffer and only pointers to it are passed.

(d) No. Socket operations are associated with a socket.

Question 8

Linux's NAPI, the "New API," is best described as an enhancement that:

(a) Disables network device interrupts during times of high network traffic.

(b) Avoids the need to poll for incoming packets by taking advantage of interrupts.

(c) Integrates sockets into the file system API, providing a uniform access interface.

(d) Reduces the need to copy data between network layers.

(a) Yes. NAPI switches from interrupts to polling during high network traffic.

(b) Yes, but so did drivers in the past. This is the standard thing to do.

(c) No.

(d) No. Socket buffers existed long before NAPI.

Question 9

NFS does not improve performance via:

(a) Large reads.

(b) Use of a stateless protocol.

(c) Read-ahead.

(d) Client caching.

(a) NFS uses large reads (8K by default) regardless of the # bytes actually needed.

(b) *A stateless protocol does not improve performance*. It improves fault tolerance.

(c) NFS attempts to prefetch data if file access is sequential.

(d) NFS caches data at the client until it get old or a change is detected to the file's modification time.

Question 10

A *callback* in AFS:

(a) Tells a client that another client is modifying a file.

(b) Tells the server that a client has changed a file.

(c) Tells a client that a file was modified on the server.

(d) Tells a process that the requested download of a file is complete.

(a) No. Clients modify their files in isolation. AFS uses a file download model so files are changed locally and uploaded only after they are closed.

(b) No, but a file upload may cause callbacks to take place.

(c) Yes. When a file is uploaded to a server, it sends callbacks to all clients that downloaded that file in the past.

(d) No. A file download is not asynchronous.

Question 11

Which best describes Microsoft's *SMB remote file protocol*?

(a) A stateful file system with a download/upload model.

(b) A stateful file system with a remote access model.

(c) A stateless file system with a remote access model.

(d) A stateless file system with a download/upload model.

SMB servers are connection-oriented and keep state so they can provide full compatibility with local file systems, supporting operations such as locking.

SMB does not rely on file downloads and supports remote access operations (the SMB structure).

Question 12

Privilege separation is when:

(a) Users on a system get unique user IDs and privilege levels.

(b) A process cannot create data that processes running at lower privilege levels can access.

(c) A process is split into components with limited privileges.

(d) A process cannot interact with any other process or data at another privilege level.

Privilege separation is about dividing a program into multiple parts, with each part restricted to the minimum privileges it needs to do its job.

Question 13

Permissions stored in a file's metadata are an example of:

(a) An access matrix.

(b) An access control list.

(c) A capabilities list.

(d) A sandbox.

(a) No. The access matrix is the abstract structure that represents access permissions of domains on objects

(b) Yes. Domains are associated with objects. Files are objects. Users and groups of users are the domains.

(c) No. Here, objects are associated with domains.

(d) No. This provides a set of restrictions for a running process. Sandbox rules are generally not stored in a file's metadata.

Question 14

Multi-level security and the *Bell-LaPadula* model are an example of:

(a) Mandatory Access Control (MAC).

(b) Privilege separation.

(c) Access Matrix.

(d) Principle of least privilege.

(a) These are examples of OS restrictions on how a process can share data – the process cannot override this.

(b) No. This refers to splitting a program into multiple parts.

(c) No. This is a matrix specifying which domains can access which objects.

(d) No. This is a general rule that states that a process should have the least amount of privileges needed to do its work.

Question 15

Password files often store hashes of passwords:

(a) To make it difficult to steal a password even if an intruder gets the password file.

(b) To make looking up a password very quick: an O(1) operation.

(c) To be able to detect if somebody tampered with the password.

(d) To ensure that a given password is associated with only one user.

(a) Yes. Hashes are one-way functions. Given a hash, you cannot compute the password that created it.

(b) No. Password hashes have nothing to do with hashes used in lookup tables.

(c) No. Password hashes do not function as digital signatures.

(d) No.

Question 16

For Alice to send a signature of a message to Bob, she would:

(a) Hash the message and encrypt it with Bob's public key.

(b) Hash the message and encrypt it with Bob's private key.

(c) Hash the message and encrypt it with her public key.

(d) Hash the message and encrypt it with her private key.

A *signature* is a hash of a message encrypted with the signer's private key.

Nobody but the signer can do this.

Question 17

The *Diffie-Hellman algorithm*:

(a) Uses a one-way function to create cryptographic hashes of data.

(b) Provides users with a pair of keys for secure communication: encrypt data with one key; decrypt with the other.

(c) Enables a user to authenticate another user.

(d) Allows two parties to come up with a common key that they can use for encryption.

Ignore this question. We did not cover the Diffie-Hellman algorithm.

(a) Diffie-Hellman is *not* a hash function.

(b) Diffie-Hellman is *not* a public key encryption algorithm.

(c) Correct, but authentication is not automatic and Diffie-Hellman does not help you any more than other algorithms.

(d) It allows a user to generate a pair of "keys" – numbers that can be used by two parties to create a "common key" that can be used as an encryption key.

Question 18

A *hybrid cryptosystem*:

(a) Encrypts a message N times, once for each of N recipients.

(b) Encrypts a message with a symmetric algorithm and then encrypts the result again with a different key.

(c) Encrypts a message with a symmetric algorithm and then encrypts the result with a public key algorithm.

(d) Uses public key cryptography to encrypt a session key that will be used for symmetric cryptography.

(a – c) A *hybrid cryptosystem* is *not* multiple levels of encryption.

(d) Yes.

Question 19

The *Challenge Handshake Authentication Protocol* (CHAP) relies on:

(a) Encrypting the password with a key sent by the server.

(b) Encrypting the user's password with a shared key.

(c) Having the user authenticate the server before sending a password.

(d) Hashing a structure containing a user's password and a challenge sent by the server.

The client and server share a secret key.

1. Server sends a challenge (bunch of bits).
2. Client responds with *hash(challenge, key)*.
3. Server computes *hash(challenge, key)* and compares with data from client.

Question 20

A *digital certificate* cannot be modified because:

(a) It is encrypted by the certificate owner.

(b) Only the owner knows the corresponding private key to the public key that is present in the certificate.

(c) It contains a hash that was encrypted by the issuer.

(d) It is stored in a trusted database.

(a) It contains the owner's public key and identifying data. A certificate is *not* encrypted.

(b) Yes, but that's not what prevents the certificate from being modified. The owner cannot modify the certificate either.

(c) The certificate cannot be modified because **it contains the issuer's signature**. The signature is a hash of the certificate data that is encrypted with the issuer's private key. If you modify the data in the certificate, you will need to create a new signature but you need the issuer's private key.

(d) No. It can be passed around freely.

Question 21

A *rootkit* is:

(a) Software that makes itself undetectable on a system.

(b) A set of tools that allow a user to gain administrative access on a computer.

(c) Malicious software that can gain administrative privileges on a system.

(d) Software that replicates itself onto other systems.

(a) Yes.

(b) Maybe but not always. Some rootkits, such as Sony's, installed DRM (digital rights management) software.

(c) A rootkit is not always malicious and does not always give administrative privileges (see b).

(d) No. That's a virus.

Question 22

If it takes you one day to test all 7.2×10^{16} combinations of a 56-bit key, how long would it take with a 64-bit key?

(a) 8 days

(b) 256 days

(c) 1,024 days

(d) 1.8×10^{19} days (2^{64})

How many more keys are there in a 64-bit key than in a 56-bit key?
Each additional bit doubles the number of keys.
56 bits → 1 day
57 bits → 2^1 = 2× keys → 2 days
58 bits → 2^2 = 4× keys → 4 days
59 bits → 2^3 = 8× keys → 8 days

64 bits → 8 more bits → 2^8 more time → 256 days

Question 23

A *chroot* jail:

(a) Limits the parts of a file system that a process can access.

(b) Prevents a process from gaining administrative privileges.

(c) Ensures that a process cannot execute specific system calls.

(d) Runs a process in a secure environment to detect whether it has a virus.

chroot is a system call that changes a process' root file system to another directory node.

Question 24

An *interface definition language* (IDL) is:

(a) A machine-independent output language generated by the RPC precompiler.

(b) Processed by the RPC precompiler to generate stub functions.

(c) Used to implement remote procedures.

(d) Used by the operating system to provide user processes with an interface to remote procedure calls.

(a) IDL is *input*, not *output*. A user specifies interfaces in an IDL.

(b) Yes. The IDL is processed by an RPC compiler to create stub functions that do marshaling, unmarshaling, and data sending/receiving.

(c) No. It only defines the interfaces of remote procedures.

(d) The operating system has nothing to do with remote procedures except for providing sockets.

Question 25

Why did earlier versions of Linux, BSD, and UNIX not implement access control lists?

(a) They are redundant with protection mechanisms already in place.

(b) They opted to use capability lists instead.

(c) They don't fit within an inode.

(d) The operating system only supports discretionary access control.

(a) No.

(b) No.

(c) Yes. An ACL takes a variable amount of space, potentially large if there are a lot of users to be enumerated in the list.

(d) Most do, but this does not answer the question.

The End