Operating Systems

2015 – Exam 3 Review

Paul Krzyzanowski
Rutgers University
Spring 2015

Question 1

Explain the major difference between a file system that supports journaling (e.g., Linux ext4) versus a log-structured file system (e.g., YAFFS2).

The question does not ask you to explain YAFFS or ext4 – just the big conceptual differences between journaling and a log structured FS.

Both use logs

- A **journaling file system** uses a log to record transactions *only until* they are committed to the file system – this is for fault tolerance to ensure the file system can always be brought to a consistent state
- With a **log-structured file system**, the log ***is*** the file system.

The answer has nothing to do with checkpointing

That is just an optimization in YAFFS to avoid having to scan the entire log to reconstruct the file system

Question 2

How does Linux minimize the number of times data is copied as it moves through the various protocol layers of the network stack from the network controller to the application's socket?

- All data sits in allocated memory that is pointed to by a **socket buffer** (sk_buff) struct
  - Pointers to the struct are passed between layers
  - Pointers in the sk_buff struct identify the start and end of useful packet data to handle encapsulation

The packet is copied to kernel memory when it is received from the controller and then copied to user memory on a read from the socket

Bad answer: *data is moved, not copied*

Question 3

How do you validate that a signature S associated with a block of data M is valid if you are given the author's certificate? Assume that you know the author's certificate is authentic and you do not need to validate it.

1. Create a hash of the data
   h = hash(M)
2. Decrypt the signature using the author's public key in the certificate
   $h' = D_K(S)$
3. Compare the results. If h = h' then the signature is valid
   $h \equiv h'$?

Question 4

The purpose of a flash translation layer (FTL) is to:
a) Extend the life of flash memory.
b) Make flash memory look like a block device.
c) Convert generic flash memory operations to manufacturer-specific ones.
d) Provide a file system interface to flash storage.

A Flash Translation Layer was designed to be a software solution at the OS level to provide wear leveling, error detection & correction, and block remapping

(a) Yes. The primary purpose of an FTL is to extend the life of flash memory.

(b) No. NAND flash already looks like a block device

(c) No. That would be a device driver.

(d) No. The FTL still provides a block interface.

Flash wear leveling

- Assume a NAND flash device with 4K total blocks
  - 2.5% allowable bad blocks
  - System updates 3 files that each take up 50 blocks
    - 6 file updates per hour
- No wear leveling
  - Assume the same 200 physical blocks are reused for updates
  - NAND flash device will wear out in under 1 year
  - 95% of memory remains unused
- Wear leveling
  - Even distribution of all 4,096 blocks in the device
  - Useful life of the device > 15 years

Question 5

A checkpoint in a log-structured file system is:
a) A temporary area to store file transactions until they are committed to the file system.
b) A security filter to control what data may be written to the file system.
c) A snapshot of the current state of the file system to speed up future mounts.
d) An error correcting code written along with the data to detect bad blocks

To mount a log-structured file system, you go through the log of operations and construct the directory tree in memory, adding and deleting files as they get created.

(a) That's journaling.

(b) No. That doesn't exist. The closest is a virus scanner.

(c) Yes. A checkpoint creates a point-in-time version of that tree. You only need to scan log entries news than the checkpoint.

(d) No. That happens at a lower level (usually handled by the controller).

Question 6

A *loop* device:
a) Is a pseudo device that echoes whatever data is written to it.
b) Allows a normal file to be accessed as a block device.
c) Is a pseudo network driver that loops outgoing network packets back to the receiver.
d) Allows a block device to be accessed as a normal file.

(a) No.

(b) Yes. It enables a file to be accessed via block operations. The file can now hold a mountable file system.

(c) No. That's a network *loopback* driver.

(d) No. That's the opposite of what it does.

Question 7

IP (Internet Protocol) operates at this layer of the OSI network stack:
a) Data Link (2).
b) Network (3).
c) Transport (4).
d) Session (5).

(a) No. The data link layer is responsible for delivering packets within a LAN. Example: Ethernet

(b) Yes. The network layer handles the routing of packets via multiple networks.

(c) No. The transport layer handles application-to-application communication and may provide virtual connections, reliability, etc. Examples: TCP, UDP

(d) No. The session layer manages sessions on a connection.

Question 8

Which statement is ***true*** about UDP ?
a) UDP does not have any form of error detection.
b) UDP retransmits lost packets.
c) UDP detects network congestion and lowers its rate of transmission.
d) UDP data may arrive out of order

(a) UDP does no error ***correction***. It does detect errors. Packets with errors in the data are dropped.

(b) No. There is no keeping track of packets and no retransmission.

(c) No. TCP assumes that lost packets mean congestion and drops its rate of transmission. UDP does not.

(d) Yes. UDP does not add sequence numbers to packets.

Question 9

Sockets were designed to be compatible with files in that:
a) They implement some of VFS's inode ops and provide a file system namespace for network resources.
b) They implement some of VFS's file ops and enable the use of file read/write operations for network connections.
c) They use the buffer cache to hold network packets.
d) All of the above.

(a) No. Sockets do not provide a file system namespace.

(b) Yes. Sockets provide file-compatible read/write operations.

(c) No. The system buffer cache is used to hold frequently used data from block devices. Network messages are not cached – they are ephemeral.

Question 10

With Linux's NAPI ("new" API), the kernel :
a) Switches to polling a network controller when network traffic rates are high.
b) Switches to using interrupts from a network controller when network traffic rates are high.
c) Can use either interrupts or polling, depending on what the network controller can support.
d) Separates interrupt processing into two parts: a top half and a bottom half.

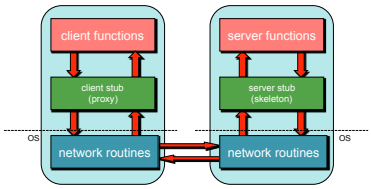
To avoid the system being paralyzed by huge interrupt volumes, NAPI disables network interrupts and switches to polling.

If polling detects no data, that means network activity diminished, so interrupts are enabled again.

Question 11

A server-side stub (or skeleton) in remote procedure calls:

a) Unmarshals incoming parameters and calls the requested procedure locally.
b) Contains the implementation of the user's remote procedure.
c) Is a placeholder function that is replaced with the user's function.
d) Is an interface layer that bridges between generic and device-specific communication functions



Question 12

What contributes most to the fault tolerant design of NFS?

a) The separation of the mounting protocol from the directory and file access protocol.
b) Caching on the client so cached data may be accessed even if the server is unreachable.
c) The absence of server state.
d) Its use of remote procedure calls to request operations on the server.

(a) Access control. Nothing to do with fault tolerance.

(b) No. NFS uses short-term caching of frequently-used blocks.

(c) Yes.
- Nothing to clean up when the server restarts or if a client disappears.
- Server can resume accepting client requests when it restarts.

(d) No. RPC requests don't imply fault tolerance.

Question 13

AFS makes *long-term client caching* work well primarily because of:

a) Its use of a connection-oriented protocol.
b) The whole-file download model.
c) The server's callback promise.
d) Client-based validation of cached contents.

(a) Nothing to do with long-term caching.

(b) Whole-file downloads make the cached content more useful but this is not what makes it work well.

(c) Yes. The fact that the server will send invalidation messages to each client that has cached data allows the client to hold it for as long as it wants to.

(d) No. AFS does not do this.

Question 14

The *principle of least privilege* states that:

a) Ordinary users should be granted the lowest-available privilege levels.
b) Each entity should be able to access only the resources it needs to perform its task.
c) High priority tasks will be granted higher privilege levels to ensure they can complete their task.
d) Only the system administrator should have access to all files and devices on a computer.

A task should have the *minimum* privileges it needs to do its job.

Question 15

A *capability list*:

a) Associates with a file a list of users and the allowable access permissions for that file.
b) Lists the complete set of operations that may be performed on files; users may be granted rights to a subset of these.
c) Is a matrix that identifies the permissible operations of each user on any file.
d) Associates with each user a list of files and access permissions for each one of them.

(a) That's an ACL.

(b) No.

(c) That's an access matrix

(d) Yes.

objects / *domains of protection*

	F_0	F_1	Printer	D_0	D_1	D_2	D_3	D_4
D_0	read owner	read-write	print	–	switch	swtich		
D_1	read-write-execute	read*			–			
D_2	read-execute				swtich	–		
D_3		read	print					
D_4			print					

Question 16

With *Mandatory Access Control*:

a) Only an administrator can define a user's access permissions for files.
b) The operating system ensures that a user has full access to all the necessary files.
c) Every file must have a default set of access permissions.
d) A user may grant access rights for a set of files that he or she owns to another user.

With MAC, the OS (via the admin) restricts the ability of a user to access or grant permissions to objects

(b) No. The OS doesn't know a what "necessary" files are

(c) No.

(d) No. The user can to that with Discretionary Access Control (DAC) – which we normally use.

Question 17

For Bob to send a message securely to Alice, he encrypts the message with:

a) Bob's private key.
b) Bob's public key.
c) Alice's private key.
d) Alice's public key.

He needs to encrypt the message in a way that only Alice can decrypt it.

The only thing Alice – and nobody else – has is Alice's private key.

If Bob encrypts a message with Alice's public key, she can decrypt the message with her private key.

(a) Anyone can decrypt this using Bob's public key.

(b) Only Bob will be able to decrypt this message.

(c) Anyone can decrypt this using Alice's private key.

Question 18

An example of a hybrid cryptosystem in use is:

a) A message encrypted with a public key algorithm and the result encrypted with a symmetric algorithm.
b) A message encrypted with a symmetric algorithm and the result encrypted with a public key algorithm.
c) A message encrypted with a symmetric algorithm and the symmetric key encrypted with a public key algorithm.
d) All of the above.

(a, b) A hybrid cryptosystem is NOT two levels of encryption

(c) Hybrid cryptosystem =

1. Encrypt a random symmetric key using a public key cryptosystem. This makes key distribution easy.
2. Encrypt/decrypt message contents using the symmetric key.

Question 19

Bob wants to authenticate himself to Alice. Alice sends him a nonce (random bits). He encrypts it with:

a) Bob's private key.
b) Bob's public key.
c) Alice's private key.
d) Alice's public key.

For authentication, Bob has to perform an operation that nobody else can perform.

The information that only Bob has is his private key.

Question 20

A symmetric cipher is a cipher:

a) That can encrypt an arbitrary number of bytes rather than being limited to multiples of blocks.
b) Where encrypting data twice with the same key results in the original data.
c) That uses a related pair of keys: one to encrypt data and the other to decrypt.
d) Where the same key is used to encrypt the data as to decrypt the data.

Question 21

Salt in a hashed password:

a) Makes a brute force attack on the password much harder.
b) Makes it difficult to use precomputed hashes to find the password.
c) Encrypts the password so that it is not readable in the password file.
d) All of the above.

Salt is a non-secret random value that is hashed with the password:

password file contains: { salt, *hash*(salt, password) }

(a) No. A brute force attack is a targeted attack on a password where you try all combinations. Salt is not part of the secret. It does not make a brute force attack harder – you still try all combinations

(b) Yes. Precomputed hashes store hashes of possible passwords (e.g., *hash("monkey")*). Salt makes this impractical. You'll need to store
hash("monkey" + "QxLUF1bgIAdeQX")
hash("monkey" + "bv5PehSMfV11Cd")
hash("monkey" + "YYLmfY6IehjZMQ"), etc. for thousands more entries.

(c) No. Passwords are usually hashed in a password file – nothing to do with salt.

Question 22

Which statement about Password Authentication Protocol (PAP) is *TRUE*?

a) PAP transmits encrypted passwords.
b) PAP transmits hashed passwords.
c) PAP requires that both the client and the server know the password.
d) PAP transmits unencrypted passwords.

(a) No. PAP uses plain text password in its protocol

(b) No. See (a)

(c) Not necessarily. If the server stores hashed passwords, it can still authenticate a user's password by comparing its hashed value.

(d) Yes.

Question 23

Which technique does **not** help handle a buffer overflow attack?

a) Address space layout randomization by the operating system's program loader.
b) Executable address space protection in the memory management unit (MMU).
c) s
d) Stack canaries in the compiler

(a) Helps avoid buffer overflow attacks because it makes return oriented programming (ROP) difficult – you can't count on code being in a predetermined place.

(b) Helps avoid buffer overflow attacks because you cannot inject executable code into the buffer.

(c) This validates that the code has not been tampered but does not validate that the code has no security holes.

(d) Helps avoid buffer overflow attacks by allowing a function to check if the return value on the stack has been corrupted.

Question 24

A *chroot jail* improves security by:

a) Restricting the system calls that a process can execute.
b) Limiting the parts of the file system that a process can see.
c) Hiding the existence of a process from other processes.
d) Not allowing the process to run with administrative privileges.

Chroot changes the root of the file system for a process and its children.

The End