Interview Questions: The Four Stages of NTFS File Growth

Easy Questions (Direct Recall)

#	Question	Marking Criteria (Scale of 1-10)
E1	According to the article, what is the initial size of the file record segment that NTFS creates for every file?	• 1-4: Incorrect size or unable to recall. • 5-7: States 1KB but may be hesitant or slightly unclear. • 8-10: Correctly and confidently states 1KB, as mentioned for the base record segment.
E2	The article mentions several common attributes found within a file record. Name three of these attributes explicitly listed in the text.	• 1-4: Names incorrect attributes or less than two correct attributes. Confuses with OS flags (Read-only, Hidden). • 5-7: Names two of the three explicitly mentioned attributes ($STANDARD_INFORMATION, $FILE_NAME, $DATA). • 8-10: Correctly names all three explicitly mentioned attributes: $STANDARD_INFORMATION, $FILE_NAME, and $DATA.
E3	What term does the article use to describe Stage one of file growth, where all parts of the file fit into its base record?	• 1-4: Incorrect term or unable to recall. • 5-7: Uses a similar but incorrect term (e.g., "internal", "local") or recalls "resident" but is unsure. • 8-10: Correctly states "Completely resident" or simply "resident" as used in the Stage one description.
E4	In Stage two (Nonresident Data), the article states the data is shipped outside the base record. Where is this data stored?	• 1-4: Incorrect location or unable to recall. • 5-7: Mentions it's stored "elsewhere" or "on the disk" but misses the specific term used. • 8-10: Correctly states it is stored elsewhere on the disk in an "allocated range of clusters", as per the article's description.
E5	What is the name of the new structure created in the base record during Stage three (Nonresident Attribute) to track attributes housed in child records?	• 1-4: Incorrect name, names a structure from another stage (e.g., mapping pair), or unable to recall. • 5-7: Recalls part of the name (e.g., "list", "attribute structure") or gives one of the two names. • 8-10: Correctly identifies the structure as an "attribute list" or "$ATTRIBUTE_LIST" as stated in the article.

Medium Questions (Interpretation & Connection)

#	Question	Marking Criteria (Scale of 1-10)
M1	Based on the article's explanation, what is the relationship between a file's data and its attributes? Is data the file itself?	• 1-4: Incorrectly states data is the file, or cannot explain the relationship based on the article. • 5-7: Explains that data is related to the file but struggles to articulate that it's *one specific attribute* according to the text. May mention other attributes exist. • 8-10: Clearly explains that according to the article, data is just one attribute (specifically $DATA) of a file, and the file itself encompasses multiple attributes like $STANDARD_INFORMATION and $FILE_NAME.
M2	The article differentiates between NTFS attributes (like $DATA) and file attributes like Read-only or Hidden. According to the text, what *are* Read-only or Hidden attributes?	• 1-4: Incorrectly describes them as the same as NTFS attributes or provides a definition based on external knowledge. • 5-7: Correctly identifies they are different but struggles to recall how the article defines them. May guess they are stored differently. • 8-10: Accurately recalls or infers from the text that the article states Read-only, Hidden, or System attributes are actually just "flags".
M3	According to the article, what specifically triggers the transition from Stage one (Completely resident) to Stage two (Nonresident Data)?	• 1-4: Incorrect trigger (e.g., file renaming, adding attributes) or vague answer like "file gets bigger". • 5-7: Correctly identifies file data growth as the trigger but fails to mention the specific constraint related to the base record segment size (1KB). • 8-10: Accurately explains the trigger is when the file data becomes too large to fit into the 1KB base record segment.
M4	In Stage two, the article notes that the file data is nonresident, but the *attribute record* is still in the base record segment. Which attribute record is being referred to here, and what information does it now contain instead of the full data?	• 1-4: Incorrectly identifies the attribute or the information it contains. • 5-7: Correctly identifies the $DATA attribute but is unclear or incorrect about what it now holds (mentions pointers generally, but not mapping pairs). • 8-10: Correctly identifies the $DATA attribute record remains in the base record and explains it now contains mapping pair(s) that track the location and length of the external data clusters.

#	Question	Marking Criteria (Scale of 1-10)
M5	What is the purpose of a "child record" as introduced in Stage three of the article? What kind of information does it hold?	• 1-4: Incorrect purpose or content, confuses child record with base record or data clusters. • 5-7: Partially correct; understands it holds something that overflowed from the base record, but unclear on specifics (e.g., says it holds "data" or just "attributes"). • 8-10: Accurately explains that a child record is created to house attribute information (specifically, the list of mapping pairs for a nonresident attribute like $DATA) that no longer fits in the base record. Mentions it contains elements like attribute records and mapping pairs.

Hard Questions (Synthesis & Application within Article Context)

#	Question	Marking Criteria (Scale of 1-10)
H1	Based *only* on the article's descriptions and diagrams, contrast how NTFS handles the overflow in Stage two (Nonresident Data) versus Stage three (Nonresident Attribute). What moves out, and what structure remains in the base record to track it in each stage?	• 1-4: Cannot differentiate the stages or provides incorrect information about what moves or what tracks it. • 5-7: Correctly identifies that data moves in Stage 2 and attribute info (mapping pairs) moves in Stage 3, but is unclear or incorrect about the tracking structures left in the base record (mapping pair vs. attribute list entry). • 8-10: Accurately contrasts: Stage 2: File *data* moves to clusters, tracked by *mapping pair(s)* within the $DATA attribute in the base record. Stage 3: *Mapping pairs* move to a child record, tracked by an *attribute list entry* (within the $ATTRIBUTE_LIST) in the base record pointing to the child record.
H2	Trace the path described in the article to locate the actual file data (the stream) for a file that has reached Stage four (Nonresident Attribute List). Start from the base record.	• 1-4: Incorrect path, misses key steps or structures mentioned in the article. • 5-7: Provides a partially correct path, perhaps missing the attribute list record step or confusing child records and data clusters, but understands multiple indirections are involved. • 8-10: Accurately traces the path based on the article: 1. Read base record to find the *attribute list record*. 2. Follow the pointer in the attribute list record to read the *attribute list* (from clusters). 3. Find the relevant *list entry* for $DATA within the attribute list. 4. Follow the pointer in the list entry to read the correct *child record*. 5. Read the *$DATA attribute* within the child record to find the *mapping pair(s)*. 6. Follow the mapping pair(s) to locate and read the actual file *data* from clusters on disk.

#	Question	Marking Criteria (Scale of 1-10)
H3	According to the article, what key characteristic distinguishes a Stage four file (Nonresident Attribute List) from a Stage three file (Nonresident Attribute)? Specifically, what becomes nonresident in Stage four that was resident in Stage three, and what *must* remain in the base record in Stage four?	• 1-4: Incorrect distinction, confuses the stages, or identifies wrong components becoming nonresident. • 5-7: Correctly identifies that the attribute list moves out in Stage four, but is unclear or incorrect about what remains in the base record or misremembers the state of the list in Stage three. • 8-10: Accurately explains: In Stage four, the *attribute list itself* becomes nonresident and moves to clusters. This list was resident (inside the base record) in Stage three. In Stage four, an *attribute list record* must remain resident in the base record to track the location of the external attribute list.
H4	The article states, "The greater the complexity used to store the file, the greater the performance hit will be". Using *only* the descriptions of the four stages provided in the article, explain why accessing data from a Stage three file would likely incur more overhead than accessing data from a Stage one file.	• 1-4: Vague answer about complexity or performance without relating it to the stages described. Uses external knowledge not in the article. • 5-7: Correctly identifies Stage three is more complex but provides a limited explanation, perhaps mentioning child records but not the sequence of reads required compared to Stage one. • 8-10: Clearly explains based on the article: Stage one requires only reading the base record (MFT) as all data is resident. Stage three requires reading the base record (for the attribute list entry), then reading the child record (to get mapping pairs), and finally reading the data clusters. This involves multiple lookups/reads compared to the single read for Stage one.
H5	A comment included with the article mentions using `FORMAT /L` to create 4KB file records instead of the 1KB default discussed in the main text. Based *solely* on the mechanisms described in the article's four stages, how might using a 4KB base record potentially affect when a file transitions between these stages?	• 1-4: Cannot relate the 4KB size to the stages or provides an explanation based on external knowledge (e.g., general performance). • 5-7: Suggests that 4KB records would delay transitions but struggles to connect it clearly to the specific contents of the base record described in the stages (data, mapping pairs, attribute list). • 8-10: Logically concludes based on the article's content: A larger 4KB base record could hold more resident data, delaying Stage two. It could also hold more mapping pairs within the $DATA attribute, delaying Stage three. Furthermore, it could hold a larger attribute list, delaying Stage four. Thus, transitions would likely happen for larger/more complex files compared to the 1KB default.