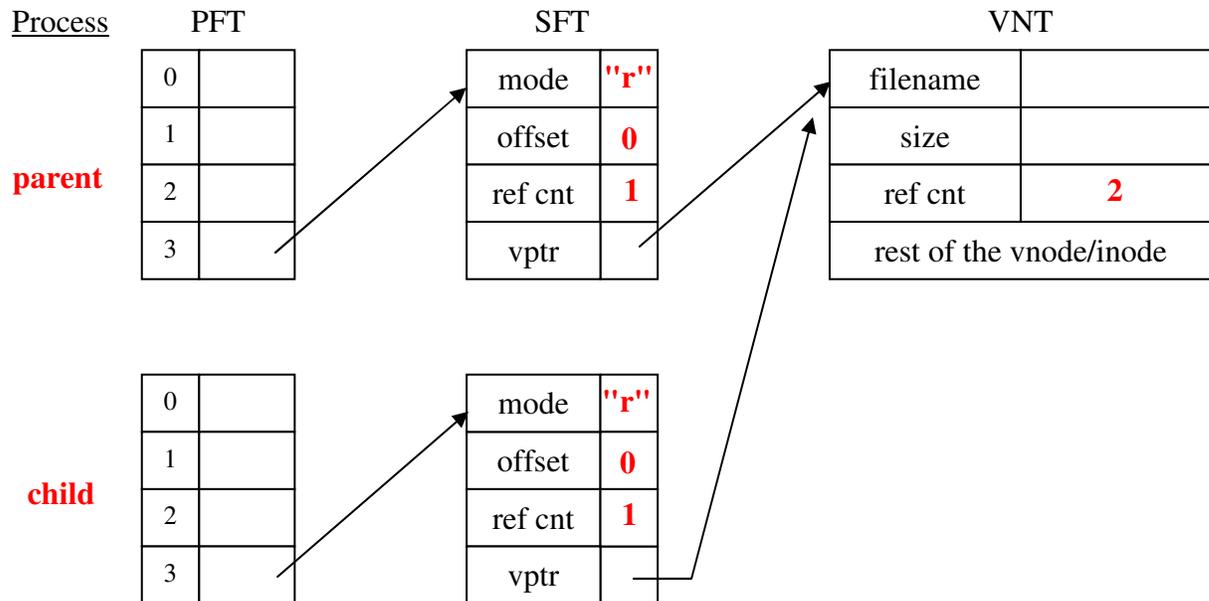


1a. Draw the per-process, system open file, and vnode tables after the following sequence occurs: (1) A process forks, (2) the parent opens a file for read (3) the child opens the same file for read



1b. Parent and child are both reading from the same file. Are we concerned about the order in which these reads occur. Explain your answer.

**No. Both processes maintain their own file offset.**

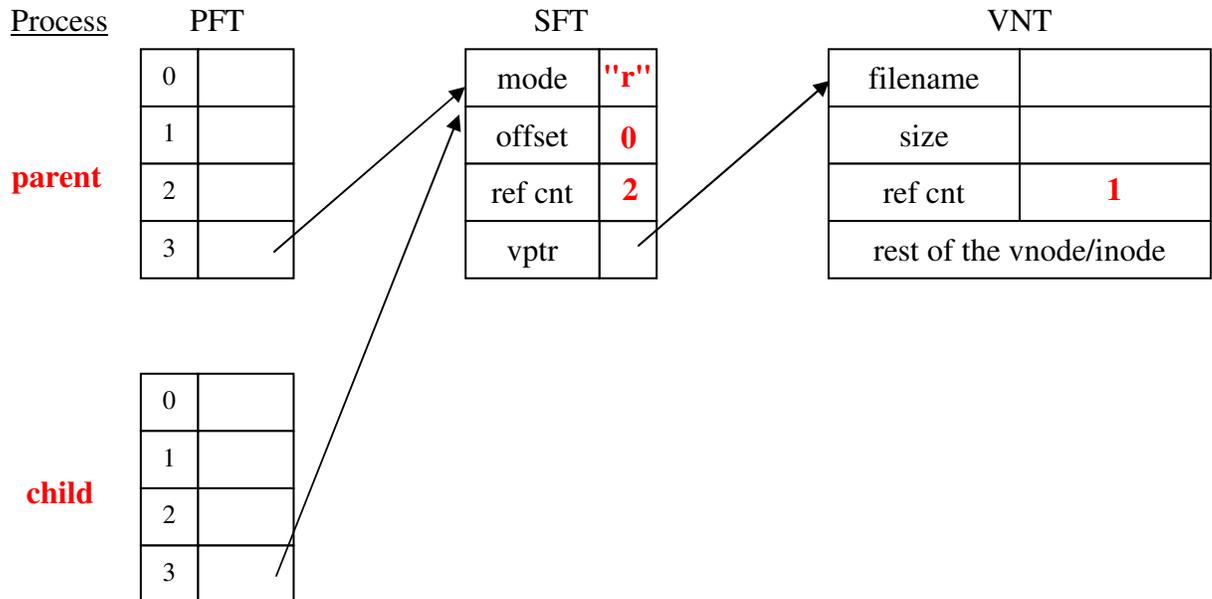
1c. After the sequence in part a. the child reads 4 bytes from and then closes the file. Does this have an effect on the parent? Explain your answer.

**No. The vnode table maintains a reference count of processes that have the file open. The file will not actually be closed until the reference count reaches zero.**

1d. Do parent and child share file descriptors for files that are opened after a fork?

**No.**

2a. Draw the per-process, system open file, and vnode tables after the following sequence occurs: (1) A process opens a file for read, then forks.



2b. Parent and child are both reading from the same file. Are we concerned about the order in which these reads occur. Explain your answer.

**Yes. Both processes share the same file offset.**

2c. After the sequence in part a. the child reads 4 bytes from and then closes the file. Does this have an effect on the parent? Explain your answer.

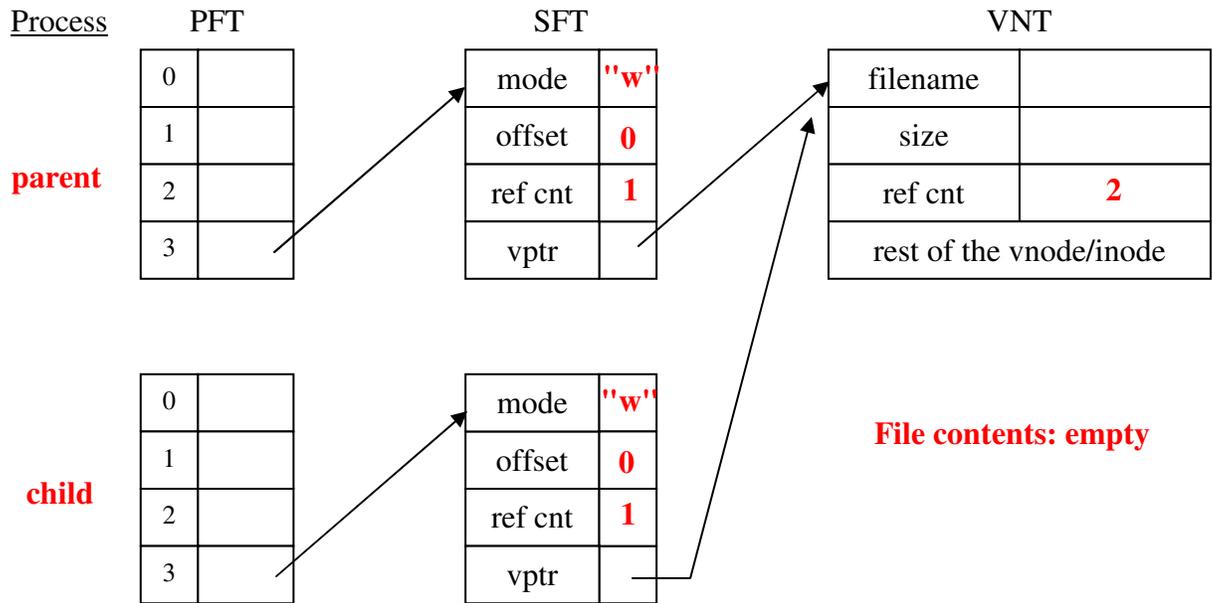
**No. The system open file table maintains a reference count of processes that have the file open. The file will not actually be closed until the reference count reaches zero.**

2d. Do parent and child share file descriptors for files that are opened before a fork?

**Yes.**

For each part below, write the file contents that exists after each step.

3a. Draw the per-process, system open file, and vnode tables after the following sequence occurs: (1) A process forks, (2) the parent opens a file for write (3) the child opens the same file for write.



3b. Parent and child are both writing to the same file. Are we concerned about the order in which these writes occur. Explain your answer.

**Yes. Although they maintain separate file offsets, since they are writing to the same file, writes by one process can overwrite what is already in the file.**

3c. After the sequence in part a. the child writes 4 bytes to and then *closes the file*. Does this have an effect on the parent? Explain your answer.

**No. The vnode table maintains a reference count of processes that have the file open. The file will not actually be closed until the reference count reaches zero.**

**File contents: ABCD**

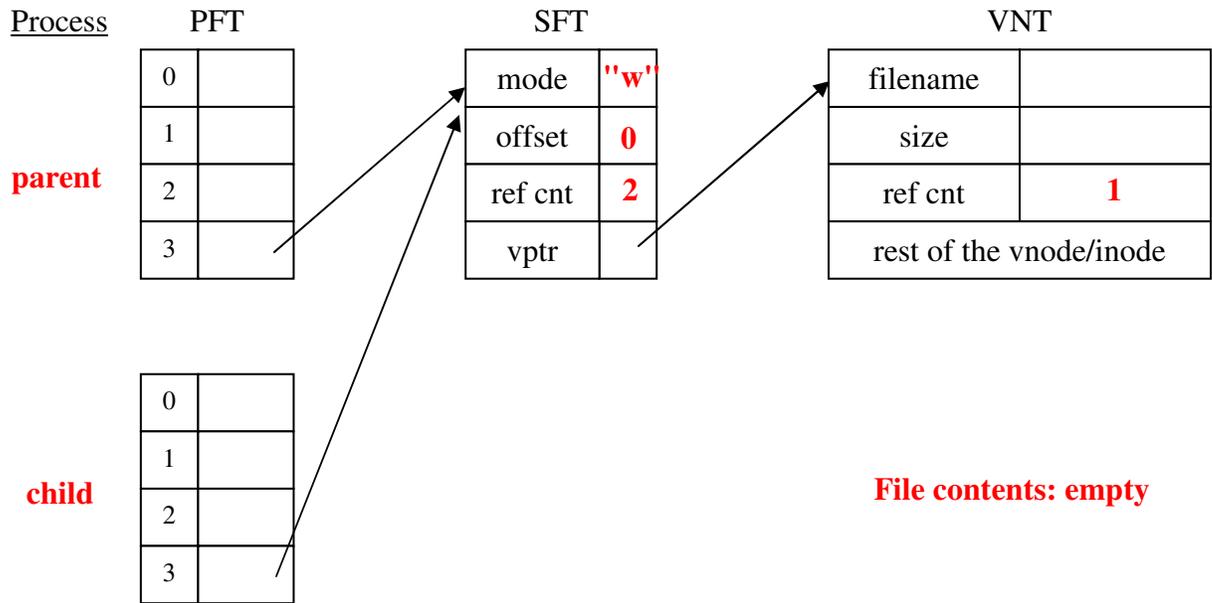
3d. After the sequence in part c. the parent writes 4 bytes to the file. Does this have an effect on what the child wrote? Explain your answer.

**Yes. The parent overwrites what is already in the file.**

**File contents: abcd**

For each part below, write the file contents that exists after each step.

4a. Draw the per-process, system open file, and vnode tables after the following sequence occurs: (1) A process opens a file for write, then forks.



4b. Parent and child are both writing to the same file. Are we concerned about the order in which these writes occur. Explain your answer.

**Yes. Since they share the file offset and writing to the same file, writes by one process will be appended to writes by the other process.**

4c. After the sequence in part a. the child writes 4 bytes to and then closes the file. Does this have an effect on the parent? Explain your answer.

**No. The vnode table maintains a reference count of processes that have the file open. The file will not actually be closed until the reference count reaches zero.**

**File contents: ABCD**

4d. After the sequence in part c. the parent writes 4 bytes to the file. Does this have an effect on what the child wrote? Explain your answer.

**No. The parent appends to the end of the file.**

**File contents: ABCDabcd**