C File Access Answers

Sayak Haldar
IIEST, Shibpur

1)Answer is a) a) A character string containing the name of the file & the second argument is the mode.

2) Answer is b) "b"

Explanation: For binary files, "ab" mode is needed.

3) Answer is c) NULL

Explanation: Remember, open returns a file pointer. So, if there any error while opening to a file, the file pointer points to nothing i.e. the returned file pointer is NULL

4) Answer is c) Both a & b

Explanation: getc returns the next character from the stream referred to by file pointer if there is any character present in the stream otherwise it returns EOF when the end of file occurred or an error occurred.

5) Answer is d) all of the mentioned.

Answer of the 6th question:

6) Answer is c) struct type

FILE is of type struct.

The contents of the FILE struct are implementation defined. Meaning they can change any time the C maintainers or OS developers need to do that.

This is Solaris 9's idea of a FILE struct. I would dig thru you /usr/include file tree and find what your version looks like. Plus, consider chacking the internal file pointer and other struct members after you have done just a single read operation - like fgets. If you wait until EOF, you get NULL pointers.

```
struct __FILE_TAG       /* needs to be binary-compatible with old versions */
{
#ifdef _STDIO_REVERSE
      unsigned char   *_ptr;  /* next character from/to here in buffer */
      ssize_t         _cnt;   /* number of available characters in buffer */
#else
      ssize_t         _cnt;   /* number of available characters in buffer */
      unsigned char   *_ptr;  /* next character from/to here in buffer */
#endif
      unsigned char   *_base; /* the buffer */
      unsigned char   _flag;  /* the state of the stream */
      unsigned char   _file;  /* UNIX System file descriptor */
      unsigned        __orientation:2; /* the orientation of the stream */
      unsigned        __ionolock:1;   /* turn off implicit locking */
```

```
        unsigned        __seekable:1;   /* is file seekable? */
        unsigned        __filler:4;
};
```

7) Answer is b) Append

8) Answer is c) w

Explanation: “w” mode is for truncating. i.e. if there is any previously written contents in the file, that will be erased

9) Answer is d) None of the mentioned.

10) Answer is d) None of the mentioned.

11) Answer is a) It writes “Copying!” into the file pointed by fp

12) Answer is d) d) It is a type name defined in stdio.h

13) Answer is c) Nothing

Explanation: Nothing will be printed.
Since, file pointer fp is initialized to stdin, fprintf(fp,”%d”,45) will write the value 45 in stdin. But, we can only see the output screen which is attached to stdout and stderr. So, nothing will be printed or written in the output terminal.

14) Answer is b) 45

15) Answer is a) File pointers

Explanation: stdin, stdout, stderr are File pointers.

16) Answer is c) Both connected to screen by default

Explanation: Though, more than options seem to be correct
c) Both connected to screen by default
and
d) stdout is line buffered but stderr is unbuffered

17) Answer b) 65 45

Explanation: In case of stdout the output is buffered, (not line buffered) but in case of stderr, it is not buffered.

So,

```
 printf("before");
 fprintf(stderr,"%s","Slight problem here");
 printf("after");
```

would print:

Slight problem herebeforeafter

Whereas,

```
printf("before\n");//added a new line character here
fprintf(stderr,"%s","Slight problem here\n");
printf("after\n");
```

would print

before
Slight problem here
after

18) Answer is a) 45 65

19) Answer is a) 45 65

References:

1) http://www.sanfoundry.com/c-interview-questions-answers/