CS 451/551 Quiz 3 Annotated Solution

Date: Mar 14, 2016 **Max Points**: 15

Important Reminder As per the course Academic Honesty Statement, cheating of any kind will minimally result in receiving an F letter grade for the entire course.

1. Which combination of `open()` flags ensures that the call to `open()` succeeds with the file being created but only if it does not already exist?

   a) `O_RDONLY|O_CREAT`

   b) `O_WRONLY|O_CREAT`

   c) `O_WRONLY|O_CREAT|O_EXCL`

   d) `O_WRONLY|O_CREAT|O_APPEND`

   e) `O_RDONLY|O_CREAT|O_APPEND`

   Answer: (c).

   `O_CREAT` is necessary to ensure that the file is created if it does not exist; `O_EXCL` is necessary to ensure that the `open()` fails if the file already exists. (c) is the only option which specifies both flags.

2. A **zombie** process is a process such that:

   a) It's parent has terminated.

   b) All of its child processes have terminated.

   c) It has terminated.

   d) It has terminated and its parent has terminated.

   e) It has terminated but its parent has not terminated and has not yet done a `wait()` to reap it's termination status.

   Answer: (e).

   A **zombie** process is one which is dead but not fully dead since the system cannot clean it up but must retain its termination status if its parent has not yet reap'd its termination status.

3. Which of the following octal `umask` values ensures that permissions for newly created files ensures that group and other do not have write access?

a) 0755

b) 022

c) 011

d) 044

e) 0700

Answer: (b).

The `umask` specifies the bits which must be turn'd off in the permissions for the newly created file. `022` specifies turning off `write` permissions for both group and other.

4. Which of the following statements is **clearly false**?

   a) The number of hard links to a directory is greater than or equal to 2.

   b) Hard links cannot cross file-systems.

   c) There is 1:1 mapping between Unix file-names and files.

   d) A symbolic link is a special kind of file.

   e) The `stat` structure for a file specifies the permissions for the file.

   Answer: (c).

   The mapping between file-names and files need not be 1:1 as the same file can be referenced using multiple file-names (*hard links*).

   Since a directory is always linked to by itself and by its parent, the number of hard links is at least 2. Since hard links map a name to an inode-number and inode-number's are local to a file-system, hard links cannot cross file systems. A symbolic link is indeed a special kind of file of type `S_ISLNK` and the `stat` structure contains the permissions of a file in the `st_mode` field.

5. A directory `d` and file `d/f` within it have the same user owner. The user owner cannot list the content of `d` but can read the contents of `d/f`. Which of the following `ls -l` permissions for `d` and `f` are consistent with this behavior?

   a) `d` has permissions `d-wxrwxrwx` and `f` has permissions `---xr-xr-x`.

   b) `d` has permissions `drwxrwxrwx` and `f` has permissions `-r-xr-xr-x`.

   c) `d` has permissions `drwxrwxrwx` and `f` has permissions `-rwxr-xr-x`.

   d) `d` has permissions `d--x--x--x` and `f` has permissions `---xr-xr-x`.

e) `d` has permissions `d-wxrwxrwx` and `f` has permissions `-r-xr-xr-x`.

Answer: (e).

Since the directory cannot be listed, its user read permissions must be off. Since the file can be read, the directory user execute permissions must be on and file user read permissions must be on. The only alternative with this combination is (e).

6. The `setuid` bit on a file affects the operation of:

   a) The `fork()` system call.

   b) The `exec()` family of system calls.

   c) The `wait()` family of system calls.

   d) The `stat()` family of system calls.

   e) The `open()` system call.

   Answer: (b).

   The `setuid` bit on a file controls the effective uid of a process which `exec()'`s that file.

7. Assume that a program performs the following steps:

```
FILE *f = fopen("file", "w");
int fd = fileno(f);
lseek(fd, 10, SEEK_CUR);
write(fd, "abc", 3);
FILE *f1 = fopen("file", "r");
int c = fgetc(f1);
```

   Assuming that none of the calls fail, the value of `c` will be:

   a) `'a'`

   b) `EOF`

   c) `'\0'`

   d) `'b'`

   e) `'c'`.

   Answer: (c).

   The `fopen()` will open the file for writing, truncating it if it already exists. The `lseek()` will insert a 10 byte hole into the file. Hence when the first byte is subsequently read, it will read as `'\0'`.