SY1-THREADS: 211-230

211 [T / F] In the concurrent webserver code, the parent process sets up the file descriptors (the FD table) and simply calls fork(), which by implication makes the child process has its own FD table that inherits the same content (pointers to FD structures) in the parent's FD table.

212 [T / F] If you're lazy to close file descriptors, the OS will clean them up upon exit().

213 [T / F] In a long/infinitely-running code, it is important to close() files that you no longer use, otherwise the corresponding FD entries (within the infinitely-running process) cannot be reused.

214 [T / F] Thus, in general, long-running programs are more prone to memory leaks (*e.g.*, web/gaming servers).

215 [T / F] pthread_create() will create a new thread that runs starting from the main() function of the process.

216 [T / F] If a peer thread calls exit(), only the peer thread will be killed by the OS.

217 [T / F] If the main thread calls exit(), all the threads of the same process will be killed.

218 [T / F] Killing a process essentially means that the OS never gives the CPU back to the process; the OS never tells the CPU to return to the process' program counter (next instruction address) and the process' process control block (PCB) is destroyed.

219 [T / F] The exit() system call will wait until the peer threads finish before shutting down the process.

220 [T / F] When the execution of the main thread reaches the end of the main() function, all the peer threadds are killed even though they are still running.

221 [T / F] If your process hits a race condition, you can guarantee that the bug can be reproduced in one run (in just one test).

222 [T / F] Race condition/non-determinism will never happen if you don't use threads.

223 [T / F] In a multi-threaded process, threads share the same stack area (*i.e.*, one stack base address) in the process address space.

224 [T / F] Every thread within a process has its own segment table. So, three threads in a process means three segment tables.

225 [T / F] In many-thread programming, a 4-bit segment indexing does not make sense because a program might create dozens of threads but the segment table does not have enough entries for all the stacks.

226 [T / F] A bug in one thread (*e.g.*, divide by zero, dangling pointer) can potentially impact other threads of the same process.

227 [T / F] —

228 [T / F] —

229 [T / F] —

Scanned with CamScanner

Scanned with CamScanner