CS241 Spring 2010 Pop Quiz

Monday, April 19, 2010

Questions 1-4: 2 points each.

1. You use sigprocmask() to block the SIGKILL signal when your program starts running. What would happen if you send a SIGKILL signal to your running program by using "kill -9 <your program>"?
   (a) Your program continues running.
   (b) Your program terminates.
   (c) It depends on what happens in the signal handler.

2. Process A has no signals blocked. It installs a signal handler for SIGINT in a normal way, using sigaction() with no special options, like this:

   ```
   struct sigaction sa;
   sa.sa_handler = my_signal_handler;
   sa.sa_flags = 0;
   sigemptyset(&sa.sa_mask);
   sigaction(SIGINT, &sa, NULL);
   ```

   After that, Process B sends a SIGINT signal to Process A ten times. How many times is Process A's signal handler called?
   (a) Once: the first signal will be sent but the rest will definitely be lost.
   (b) Between 1 and 10 times, depending on the ordering of events.
   (c) Five times exactly in Linux, because Linux is based on System V UNIX.
   (d) 10 times: since A has a signal handler installed, every sent signal will be delivered.

3. A program on Alice's computer sends a series of network packets to a program on Bob's computer. Alice writes the packets to her socket in the order A, B, C, but Bob reads the packets from his socket in the order B, A, C. What must be true?
   (a) Alice and Bob's computers are in the same room.
   (b) Alice and Bob's computers are not connected via a TCP socket.
   (c) Alice and Bob's computers are running HTTP, which always swaps the first two messages.
   (d) Alice and Bob's computers have gone out of sync and will no longer be able to communicate without error recovery.
   (e) None of the above answers are true.

4. The correct order of calls for a TCP server is:
   (a) socket(); bind();
   (b) socket(); bind(); connect();
   (c) socket(); bind(); listen(); accept();
   (d) bind(); socket(); listen();

Questions 5-9: 1 point each. Recall that operating systems manage memory in chunks called pages. What happens when the OS uses a **smaller** page size? For each of the following, mark (a) for true or (b) for false.

5. There is less wasted memory.
6. The page table becomes larger.
7. The maximum amount of memory a program can allocate is reduced.
8. If the program accesses memory in a page that is swapped out, it can be swapped in more quickly.
9. If the page size is too small, pointers may not be large enough to address all the pages, especially on 32-bit architectures.

Solutions

1. (b). SIGKILL can't be blocked or handled by the program; the OS just kills the recipient.
2. (b). If a SIGINT arrives before the handler has been called for the previous SIGINT, then the signal is effectively lost. So the handler could be called once (if all the SIGINTs arrive before any handler is run), ten times (if the kernel chooses to call the handler after every SIGINT arrives), or anywhere in between.
3. (b). TCP ensures in-order delivery to the socket.
4. (c). As in the networking MP.
5. We accepted both True and False. The page table becomes larger, which uses more memory. However, less memory is wasted because there is less fragmentation: the OS can give programs an amount of memory closer to what they ask for. (For example, if the program asks for 1 more byte of heap space, then $pagesize - 1$ bytes could be wasted). Either of these two effects could dominate depending on the system.
6. True. There are more pages, and hence more virtual$\rightarrow$physical page mappings in the table.
7. False. The amount of memory a program can allocate is dependent on the amount of physical memory, not the number of pages.
8. True. In general, fewer disk accesses are necessary to retrieve the page off disk. (Once the page size drops below the disk's block size, this wouldn't make much difference.)
9. False. Pointers always have at least as many bits as are necessary to identify a page of memory.

Score statistics

- Median: 10
- Mean: 9.5
- Max: 13
- Min: 4