Internet Programming Exam

Monday, October 21st, 2013

This is a closed book exam: no documentation is allowed

WRITE CLEARLY!!!

---

1 Program Output (1 point)

What will be the output of the following program? If there are multiple possible outputs, show them *all*.

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main() {
  int i, x;

  x = 0;
  if (fork()==0) {
      x = x+1;
  }

  if (fork()==0) {
      x = x+1;
  }
  else {
    execl("/bin/echo","echo","BAR",NULL);
    execl("/bin/echo","echo","BAZ",NULL);
  }

  printf("%d\n",x);
  execl("/bin/echo","echo","echo","BAT",NULL);
  return 0;
}
```

Assumptions:

- Assume that the call to `/bin/echo` succeeds, and (as expected) outputs all parameters passed to it, plus a new line at the end.
- Assume that printing of `"A\n"` is atomic, that is, no other process can be swapped in after printing the character but before printing the new line.

2 Questions for Brief Answers (5 points)

SAMPLE QUESTION: What does the `listen()` system call do?
SAMPLE ANSWER: It is relevant only in the context of TCP. It promotes a client socket to a server socket, and sets the *backlog*, that is, the size of the queue holding incoming clients that have not yet been `accept()`ed.

1. What does the `signal()` system call do?
2. Can you use pipes[1] for communication between processes running on:
   (a) A single machine? If yes, under what conditions?
   (b) Different machines? If yes, under what conditions?
3. In the following Java program, which of `a.one()`, `a.two()`, `a.three()`, `b.one()`, `b.two()`, and `b.three()`, may **not** be executed concurrently by different threads?

```
public class MyNiceSyncClass {
  synchronized public void one() { ... }
  synchronized public void two() { ... }
  public void three() { ... }
}

MyNiceSyncClass a = new MyNiceSyncClass();
MyNiceSyncClass b = new MyNiceSyncClass();
```

4. What functionality does PHP provides for substantially facilitating personalization of sessions? Explain briefly what exactly it does (the principle, not particular keywords, variable names, syntax, etc.).
5. When is preforking preferable over one process per client?
6. Explain the role of the portmapper in Sun RPC.
7. In RPC, when would you prefer UDP as the transport protocol and when TCP?
8. How does a web server handle a new request to a servlet-based page? Does it start a new JVM (Java Virtual Machine)? Does it create a new thread in the same JVM? Does it queue the request to be processed by a single thread? Does it create a new instance of the servlet class? Does it operate on the same instance of the servlet class?
9. Explain the difference between a mutex and a condition variable. Do not focus on the API and its syntax. Only explain clearly the different functionality each of them provides.
10. In a preforking server where all children are accepting on the same socket (created by the parent), it is advisable to "protect" that `accept()` by a mutex around it. Why?

[1] The question refers to *unnamed* pipes (the ones taught in class). Do **not** consider *named* pipes (also known as *FIFO*s).

3 Meteorologic Sensor Network (4 points)

We are designing a Meteorologic Sensor Network to collect real time weather information across The Netherlands.

There are three entities in the context of this system: a centralized *Meteorology Server* (or just *server*), a large set of *Weather Sensors* (or just *sensors*), and *Users*. The Meteorology Server is expected to keep all weather information from Weather Sensors in local memory, and to reply to Users' queries about weather at any part of the country.

1. There is an extensive network of sensors (tens or hundreds of thousands).
2. A sensor provides weather information by means of a tuple `<x,y,rain,temp>`, where `x` and `y` are integers showing the coordinates of its location, `rain` is a boolean specifying whether it's currently raining at the sensor's location, and `temp` is a floating point specifying the current temperature.
3. Apparently, weather at a particular location does not change too frequently. E.g., the temperature or rain state could be changing in the order of hours, or (tens of) minutes, but definitely not every few seconds.
4. The server can be asked about weather information at some (rectangular) area, by specifying the area's coordinates $\texttt{<x}_{min}\texttt{,y}_{min}\texttt{,x}_{max}\texttt{,y}_{max}\texttt{>}$, and should return a list with all `<x,y,rain,temp>` tuples for coordinates contained in that area.
5. Such requests to the server can be happening very frequently, e.g., it may have to handle tens of such requests per second.

RMI interface

We decided to make the Meteorology Server based on RMI. That is, sensor information is transfered to the server via RMI, and requests for weather information by users are made through RMI too.

QUESTION A: Regarding the copying of weather data from the sensors to the server, which policy would you choose (*push* or *pull*), and why? By *push*, a sensor sends its weather information to the server when it wishes. By *pull*, the server asks a sensor its current weather state when it wishes. It is important to explain (briefly!!) why you choose one policy or another.

QUESTION B: Write the RMI interface(s) needed by the system, including RMI-specific statements[2], the defined methods, and the arguments of the methods. Most importantly, for the interface(s) you specify, don't forget to specify where (server? sensor? user?) its/their implementation(s) is/are supposed to run.

Web interface

After the RMI-based Meteorologic Sensor Network has been running smoothly for a few years, we decide to add Web access for users' requests. A user now has the option to log on to some web site, and navigate through the map of the Netherlands (e.g., with something like Google Maps) which should show a pin for each point where there is a sensor. The pin should display the temperature of the sensor and its color should denote the rain state (rain: blue, no-rain: orange).

[2] RMI interfaces should extent `java.rmi.Remote`, and remote methods should throw `java.rmi.RemoteException`

The user can scroll through the map and zoom in/out. Pins on the map should be updated real time, as weather conditions change at various sensors in the view, without having to reload the page.

QUESTION C: Explain in detail the architecture to support the web interface on top of the existing system. Note that the existing RMI server cannot be altered in any way at this point. Be explicit and detailed on what entities we have (e.g., Server, Sensors, Users, etc.), what technologies and protocols are used (and where), what information is sent between entities, how each individual type of communication is initiated (push? pull? who is the sender and who the receiver?), which calls are blocking, etc.

Note: Do not deal at all with how to interact with the map. All you need to know is that your browser automatically knows at any moment the coordinates of the currently visible portion of the map ($<x_{min}, y_{min}, x_{max}, y_{max}>$), and the map provides a trivial API for the visualization of pins and for updating their text and color, which is of no interest to this exam.

Security

On April Fool's Day, a talented group of VU students made a joke, hacking the system and reporting 35°C and sun all over the country!

QUESTION D: Suggest how you would alter the software running on the server and the sensors to secure the system against such attacks. You cannot use `stunnel` or `ssh`, think of a lower level solution. Which keys do you have? Who has which key? How are they being used?

Use the MINIMUM POSSIBLE techniques to ensure just that the server can trust that the received values come from a real sensor and are fresh (not replayed). I want to test that you know what is the role of each security principle, and how it is used. Including unnecessary security components "just in case", will cost you credits! :-)

— Good luck!! —