System Administration for Beginners

Week 4 Laboratory

March 1, 2010

Due to security concerns, most of the commands you'll need to perform this laboratory have been removed from the servers in Soda Hall. Thankfully, the OCF provides access to all the commands you need. Part of Homework #1 was to create an OCF account, so we will be using your OCF account to perform this lab.

At the top of each submission, please provide the assignment name, your full name, `inst` login (`cs198-XX`), and your email address. Answer the following questions below. If no specific information is being asked, include any output or answers that you think would help show us that you understand the material (text only). Turn in your *paper* submission at the start of class next week.

NOTE It is advisable that you first compose your lab submission in some text editor (this would be a good time to practice a text editor, like `nano`, `vim`, or `emacs`) and then print using the `lpr` command.

NOTE There are a few questions marked **EXTRA**; these are *optional*, but for those so inclined, may present new and perhaps more interesting material. Most of these are not covered in the lecture notes in great detail and require some research using other resources (the facilitators being one of them!).

1 Logging into the OCF

1. Open a terminal window. Enter `ssh YOURLOGIN@ocf.berkeley.edu`.
2. You will be asked for your current OCF password. Enter it. If you get a message asking you if you want to continue connecting, please enter `yes`.

From now on, you will do everything in this terminal window (or if you're handy with `screen`, multiple windows in one). If for some reason you close the terminal window or need to use another one, you will have to log into your OCF account in a new window.

2 IP Addresses

Every computer connected to the Internet has a unique address called an *IP address*. IP addresses are a set of 4 numbers separated by periods. Each number can be between 0 and 255. For example, most IP addresses owned by Berkeley have 169 and 229 as their first two numbers; that is, they are in the form of `169.229.x.x` where `x` is a number between 0 and 255.

3. Given the above information, how many IP addresses can Berkeley have in that *address space* – that is, theoretically, how many unique IP addresses can Berkeley have in the form of `169.229.x.x`?

4. In total, how many unique IP addresses are possible?

5. The current world population is just under 6.5 billion. Are there enough IP addresses for every single person in the world? While this may sound like a stupid question, note that corporations and organizations (like Berkeley) often own large blocks of IP addresses, reducing the number of publicly available IP addresses.

6. **EXTRA:** What IP addresses are reserved for private networks? What set of IP addresses are restricted for any kind of use? What is the *broadcast address* and why is it used? Could you assign a computer with a broadcast address? What kinds of effects would happen?

7. **EXTRA:** What's the difference between IPv4 and IPv6 addresses? For the following questions, do each of the IPv4 components of what's being asked have an associated IPv6 component as well (*e.g.*, if asked to perform a `traceroute`, see if you can do the same with IPv6)? You might have some trouble with getting an IPv6 address, but some areas of AirBears and Soda (2nd floor labs) are assigning public IPv6 addresses.

3 DNS

8. *host* is a command to translate a hostname into the IP address to which it is linked. Read the man page for *host* and determine the syntax for translating `nova.cs.berkeley.edu` into its IP address. Please note that when man pages specify a command's syntax, anything in []'s (brackets) are optional parameters that you probably don't need.

9. As explained above, IP address translation can also work in the reverse direction. Run *host* on the IP address you obtained for `nova.cs.berkeley.edu`. This is called looking up the *reverse DNS*. Please note that the reverse DNS does not necessarily have to match the forward DNS or even exist.

10. Now look up the IP address for `www.berkeley.edu`. Look at the IP address for `berkeley.edu` (without the `www`). Are the two the same? Try looking up the IPs for `ocf.berkeley.edu` and `www.ocf.berkeley.edu`.

The output should look different from what you got when you looked up the IP address for `nova.cs.berkeley.edu`. This is because hostnames can actually be linked to other hostnames. There are different classes of hostnames (more generally called *records*), and they do not necessarily have to be linked directly to an IP address. Hostnames that are linked directly to IP addresses are called *A records*. Hostnames that are linked to another hostname are called *CNAME records*.

11. Open a web browser and use a search engine to lookup two more types of DNS records. Write them down.

12. Let's find more about `www.berkeley.edu`. *dig* is a command to query a DNS server to find out more about its DNS records. Use the command *dig* on `www.berkeley.edu` and see if you can locate the A and CNAME records, as well as any records of the type you looked up in the previous exercise.

13. *dig* a few more of your favorite websites. If you notice a strange record, ask an instructor to explain it to you.

4 Ping

As a systems administrator, you will want to make sure your servers are always up and running. Unfortunately, you may not always be physically near your servers to check on them, and even if you were, just seeing the on light wouldn't tell you if your server was still connected to the Internet. A tool to check the connectivity of a server is *ping*.

14. *ping* `solar.cs.berkeley.edu` to make sure it is connected to the Internet.

Sometimes systems administrators block ping requests with a firewall. This is because ping can be used by malicious people to attack your server and flood it with requests.

15. *ping* `www.cnn.com`. Usually when you get no response from the 'ping' command, it means that the server is either down or has blocked your request. Recall that you can press Ctrl-C to abort a command.

5 Traceroute

traceroute is a tool to follow the path data will take between you and a recipient servre. It will list the routers it passes through along the way, as well as the round-trip time it takes to reach a certain router. By using *traceroute*, you can determine if a particular Internet link is responsible for any slowness in your connection.

16. *traceroute* the path to `fallingrocks.ocf.berkeley.edu`.

17. *traceroute* the path to `www.stanford.edu`.

18. *traceroute* the path to `www.mit.edu`.

19. Besides the different names of routers, what is the difference between the outputs you received for the last three exercises? Why is this so? Can you guess where any peering points may be in the routes you obtained?

20. *traceroute* the path of your favorite websites. Occasionally *traceroute* will fail after a number of *hops*. You will have to abort the command or wait for it to finish. Do you remember how to abort a command?

21. **EXTRA:** What is the information being given by *traceroute*? Why is it that sometimes you will come across an entry that are all asterisks (*)? Try running traceroute on the same domain twice; is it possible to get two different routes?

6 Whois

22. To register a top-level domain, ICANN regulations state that you must provide contact information for your domain to the public. Usually this involves your name, address, and phone number. *whois* is a command to look-up this information. Run *whois* on `berkeley.edu`.

23. While corporations and organizations can provide a business contact, most individuals can not. If you own a domain or know somebody that does, run *whois* on the domain.