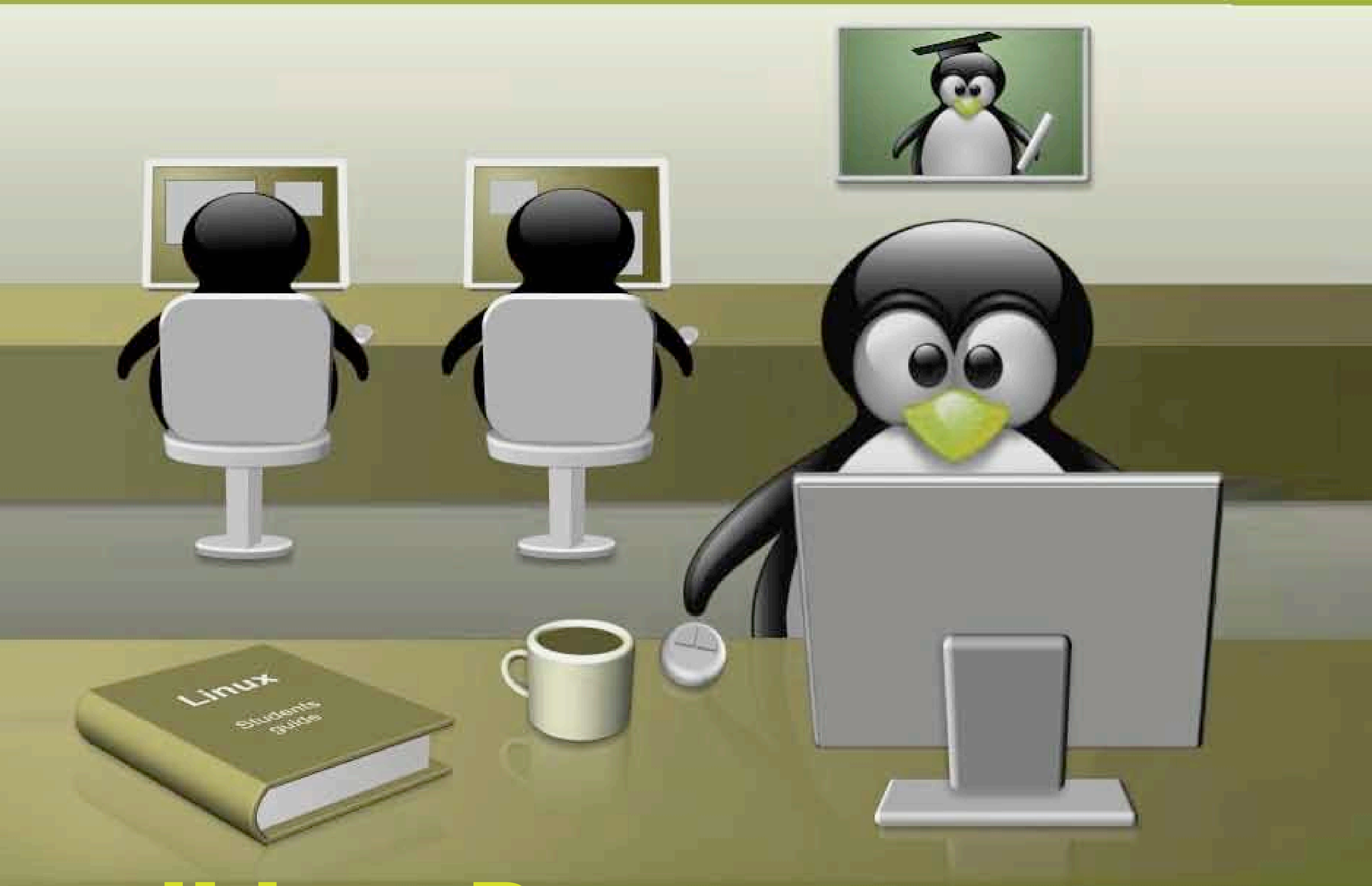


iLinuxBot

This article is to encourage newbies to use a little creativity to solve real-world problems. It shows you how to manage a Linux lab, or a lot of Linux PCs, from one machine.

I will consider you comfortable with client-server programming in C, using Berkeley UNIX standards. If you are not, please refer to LFY's earlier series on, 'How to make your own server' by Pankaj Tanwar.

The idea of my management system was inspired by botnets (you're right, the very technology used by crackers to DDoS websites). A botnet is nothing but a group of infected computers controlled by the cracker using a command-and-control channel to perform various tasks, which may be to DDoS a website or to click advertisements for the cracker's profit. For more information on botnets, please refer to the various DDoS articles published in previous issues of *LFY*.

Let us now design and develop a system that works like a botnet (a bot or client, and a server). Our server will be responsible for managing clients and issuing commands to clients or bots, whereas the bot will receive commands and execute them on each host. We need the following tools for development:

- GNU C compiler
- Any text editor, i.e., *gedit*
- Any Linux distribution

Code

Now, let us code a chat program based on the connection-oriented TCP protocol (*server.c* and *client.c*). For details on this, please refer to the above mentioned series 'How to make your own server' by Pankaj Tanwar. We will alter the code of the chat client and insert the following instructions below the point where the message is received—code that will execute the following:

```
system (command_name_and _arguments);
```

Note: The source code files *server.c* and *client.c* can be downloaded from *http://linuxforu.com/article_source_code/feb12/source_code.zip*.

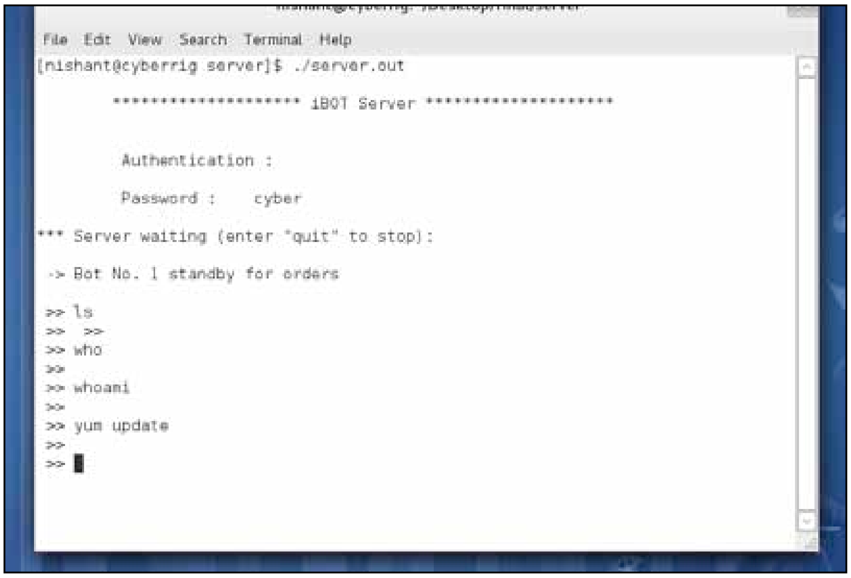
Because our client code takes the port number and IP address of the server machine as arguments, for simplicity, let us create a shell script file named *start.sh* with the following code:

```
. /client -p port_no ip_of_server
```

Deploying

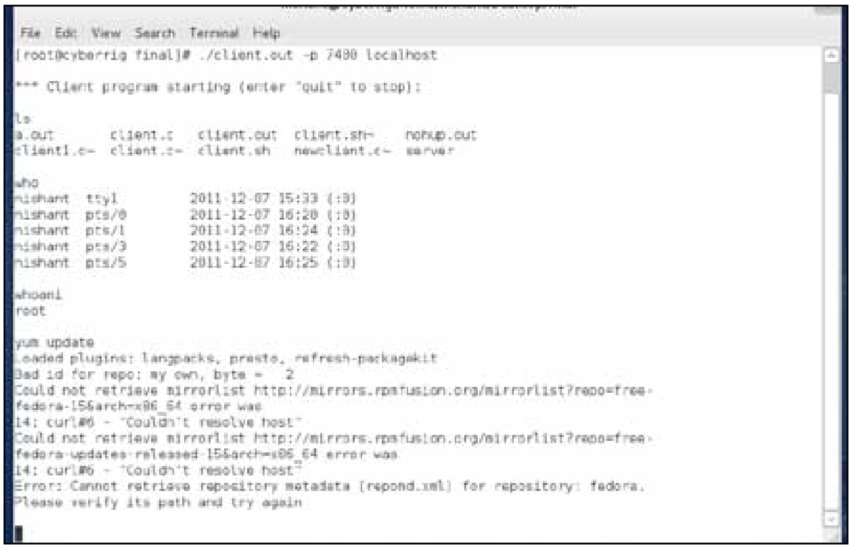
First, let us run the server program, which will listen on port 7400 (you can change this in the code). Now, when all the other systems boot, they will automatically run the client program with *root* privileges (since we have added the client initiator script *start.sh* into start-up applications). As we are demonstrating the system, let the bot run in the foreground—but in real deployment scenarios, it will be run as a daemon process.

As the client executes a run, it will automatically connect to the server and wait for commands. When it receives commands, it will execute them. Figure 1 shows the server start up and issue commands (here, I named it iLinuxBot). Figure 2 shows the client receiving and executing commands.



```
[nishant@cyberrig server]$ ./server.out

        ******************** iBOT Server ********************


         Authentication :

         Password :    cyber

*** Server waiting (enter "quit" to stop):

 -> Bot No. 1 standby for orders

 >> ls
 >>  >>
 >> who
 >>
 >> whoami
 >>
 >> yum update
 >>
 >> █
```

Figure 1: Server screenshot



```
[root@cyberrig final]# ./client.out -p 7400 localhost

*** Client program starting (enter "quit" to stop):

ls
a.out       client.c   client.out  client.sh~    nohup.out
client1.c~  client.c~  client.sh   newclient.c~  server

who
nishant  tty1         2011-12-07 15:33 (:0)
nishant  pts/0        2011-12-07 16:20 (:0)
nishant  pts/1        2011-12-07 16:24 (:0)
nishant  pts/3        2011-12-07 16:22 (:0)
nishant  pts/5        2011-12-07 16:25 (:0)

whoami
root

yum update
Loaded plugins: langpacks, presto, refresh-packagekit
Bad id for repo: my own, byte =   2
Could not retrieve mirrorlist http://mirrors.rpmfusion.org/mirrorlist?repo=free-
fedora-15&arch=x86_64 error was
14: curl#6 - "Couldn't resolve host"
Could not retrieve mirrorlist http://mirrors.rpmfusion.org/mirrorlist?repo=free-
fedora-updates-released-15&arch=x86_64 error was
14: curl#6 - "Couldn't resolve host"
Error: Cannot retrieve repository metadata (repomd.xml) for repository: fedora.
Please verify its path and try again

█
```

Figure 2: Client screenshot

Potential

This program can cut down on manpower and time by installing or changing settings, by changing configuration files, or by rebooting all systems at the same time. In this version, the ability to interact with a specific single computer is not implemented, but it can be overcome by installing and running the SSH daemon on all machines.

All commands that don't need iterative user interaction, such as *yum* or *reboot*, can be executed successfully from this system. We will extend support to other commands and also create a GUI interface in the next article.

As this is my first article, I would also like to hear your views on it. Do send me a mail in case of any queries or suggestions. (Here, I would like to express my gratitude to Ankit Sharma for his contribution to this article.)

Keep thinking in open and odd ways—that's what a techno-freak is meant to do. Long live FOSS… END

By: Nishant Sharma aka cyberpirate

The author is a FOSS lover, a mid-level gamer and an avid fan of the *Call of Duty Modern Warfare* series. He can be reached at *sharmanishant@aol.in*.