

Ethiopian Free and Open Source Software Network
(EFOSSNet)

Basic Linux Administration Training

Yonas Hailu

HiLCoE School of Computer Science and Technology
yyonashailu@yahoo.com

December 2005

Outline

- Linux Network configuration
- Print configuration
- Job Scheduling
- Package Management
- Monitoring the system
- Advanced Topics
 - Shell Scripting
 - Programming in Linux
 - Some server utilities

Part 0

**Text Editor and File
Permission Administration**

Part I

Linux Networking

Linux Network

- **Modern paradigm of computing:-Computer Networking**
 - Merging of computer and communications technologies
 - Connection of computing systems
- **Application:**
 - resource sharing programs, equipment, data
 - communication medium
 - videoconferencing
 - electronic business

Network Protocols

■ TCP/IP

- The reason for the development of TCP/IP was to provide a reliable data transport system over an unreliable network.
- Very simply put, IP provides a solution for sending packets of information from one machine to another, while TCP ensures that the packets are arranged in streams, so that packets from different applications don't get mixed up, and that the packets are sent and received in the correct order.
- TCP/IP networking has been present in Linux since its beginnings.
- It has been implemented from scratch.
- It is one of the most robust, fast and reliable implementations and is one of the key factors of the success of Linux.
- Linux and networking are made for each other, in so much that not connecting your Linux system to the network may result in slow startup and other troubles. Even if you don't use any network connections to other computers, networking protocols are used for internal system and application communications.
- Linux expects to be networked.

Protocols (cont`d)

- A good starting point for learning more about TCP and IP is in the following documents:
 - **man 7 *ip***: Describes the IPv4 protocol implementation on Linux (version 4 currently being the most wide-spread edition of the IP protocol).
 - **man 7 *tcp***: Implementation of the TCP protocol.
 - RFCs
 - RFC793, RFC1122, RFC2001 for TCP, and RFC791, RFC1122 and RFC1112 for IP.
 - The Request For Comments documents contain the descriptions of networking standards, protocols, applications and implementation. These documents are managed by the Internet Engineering Task Force, an international community concerned with the smooth operation of the Internet and the evolution and development of the Internet architecture.
 - Your ISP usually has an RFC archive available, or you can browse the RFCs via <http://www.ietf.org/rfc.html>.

Protocols (cont`d)

■ TCP/IP_{v6}

- Nobody expected the Internet to grow as fast as it does.
- IP proved to have quite some disadvantages when a really large number of computers is in a network, the most important being the availability of unique addresses to assign to each machine participating.
- IP version 6 was devised to meet the needs of today's Internet.
- Unfortunately, not all applications and services support IPv6, yet.
- A migration is currently being set in motion in many environments that can benefit from an upgrade to IPv6. For some applications, the old protocol is still used, for applications that have been reworked the new version is already active.
- So when checking your network configuration, sometimes it might be a bit confusing since all kinds of measures can be taken to hide one protocol from the other so as the two don't mix up connections.
- More information can be found in the following documents:
 - **man 7 *ipv6***: the Linux IPv6 protocol implementation.
 - RFC1883 describing the IPv6 protocol.

Protocols (cont`d)

■ PPP, SLIP, PLIP, PPPOE

- The Linux kernel has built-in support for:
 - PPP (Point-to-Point-Protocol),
 - SLIP (Serial Line IP) and
 - PLIP (Parallel Line IP).
- PPP is the most popular way individual users access their ISP, although in densely populated areas it is often being replaced by PPPOE, PPP over Ethernet, the protocol used in cable modem connections.
- Most Linux distributions provide easy-to-use tools for setting up an Internet connection.
- The only thing you basically need is a username and password to connect to your Internet Service Provider (ISP), and a telephone number in the case of PPP. These data are entered in the graphical configuration tool, which will likely also allow for starting and stopping the connection to your provider.

Protocols (cont`d)

■ ISDN

- The Linux kernel has built-in ISDN capabilities. Isdn4linux controls ISDN PC cards and can emulate a modem with the Hayes command set. The possibilities range from simply using a terminal program to full connection to the Internet.

■ AppleTalk

- Appletalk is the name of Apple's internetworking stack. It allows a peer-to-peer network model which provides basic functionality such as file and printer sharing. Each machine can simultaneously act as a client and a server, and the software and hardware necessary are included with every Apple computer.
- Linux provides full AppleTalk networking. Netatalk is a kernel-level implementation of the AppleTalk Protocol Suite, originally for BSD-derived systems.
- It includes support for routing AppleTalk, serving UNIX and AFS file systems using AppleShare and serving UNIX printers and accessing AppleTalk printers.

Protocols (cont`d)

■ SMB/NMB

- For compatibility with MS Windows environments, the Samba suite, including support for the NMB and SMB protocols, can be installed on any UNIX-like system.
- The Server Message Block protocol (also called Session Message Block, NetBIOS or LanManager protocol) is used on MS Windows 3.11, NT, 95/98, 2K and XP to share disks and printers.
- The basic functions of the Samba suite are sharing Linux drives with Windows machines, accessing SMB shares from Linux machines, sharing Linux printers with Windows machines and sharing Windows printers with Linux machines.
- Most Linux distributions provide a *samba* package, which does most of the server setup and starts up **smbd**, the Samba server, and **nmbd**, the netbios name server, at boot time by default. Samba can be configured graphically, via a web interface or via the command line and text configuration files.

Protocols (cont`d)

■ Miscellaneous protocols

- Linux also has support for:
 - WAN internetworking (X25, Frame Relay, ATM),
 - InfraRed and other wireless connections,
 - etc

Network configuration

■ Configuration of local network interfaces

- All the big, user-friendly Linux distributions come with various graphical tools, allowing for easy setup of the computer in a local network or for connecting it to an Internet Service Provider. These tools can be started up from the command line or from a menu:
- RedHat Linux comes with **redhat-config-network**, which has both a graphical and a text mode interface.
- Suse's YAST or YAST2 is an all-in-one configuration tool.
- Mandrake comes with a Network and Internet Configuration Wizard, which is preferably started up from Mandrake's Control Center.
- Information you'll need to provide
 - for connecting to the local network, for instance with your home computers, or at work:
 - Hostname
 - domainname and
 - IP address.
 - For connecting to the Internet:
 - Username and password for your ISP, telephone number when using a modem.

Configuration (cont`d)

■ Network configuration files

- The graphical helper tools edit a specific set of network configuration files, using a couple of basic commands.
- The exact names of the configuration files and their location in the file system are largely dependent on your Linux distribution and version.
 - `/etc/hosts`
 - `/etc/resolv.conf`
 - `/etc/nsswitch.conf`

Configuration (cont`d)

■ Network interface names

- On a Linux machine, the device name `lo` or the local loop is linked with the internal `127.0.0.1` address.
- The first ethernet device, `eth0` in the case of a standard network interface card, points to your local LAN IP address.
- Normal client machines only have one network interface card. Routers, connecting networks together, have one network device for each network they serve.
- If you use a modem to connect to the Internet, your network device will probably be named `ppp0`. This is normally also the case for connections using a cable modem.

Network configuration commands

- **arp**:- prints the entire system ARP table, listing IP and MAC for each host.

- **hostname**:- to set and prints host name

- **domainname**:- similar to hostname.

Shows the network information service (NIS) domain used by the host.

- **dnsdomainname**:- similar to hostname.

Sets and Prints the DNS domain name used by the host.

- **nisdomainname**:-show or set system's NIS/YP domain

- **ypdomainname** :- show or set system's NIS/YP domain

Commands (cont`d)

- **ifconfig** :- to configure the kernel's networking configuration. TCP/IP parameters

synopsis:

```
# ifconfig [interface] options | address
```

Options:- up, down,[-]arp etc

- **route** :- shows and manipulates the IP routing table. Used to set static route to specific host or network via an interface after it ahs been configured with ifconfig program. With add, and del route modifies the route table else displays the current containte of the routing table.
- **netstat** :- prints Linux network information. Use the -i option to display information about network interfaces used to display routing information with the -nr option to the command

Commands (cont`d)

- **host**:- dns lookup program that display information on hosts or domains. usually used for IP-Name mapping.
- **dig** :-dns lookup. Gives additional information about how records are stored in the name server.
- **ping**:- To check if a host is alive, use. If your system is configured to send more than one packet, interrupt **ping** with the **Ctrl+C** key combination
- **whois**:- Specific domain name information can be queried using the command, as is explained by many **whois** servers, like the one below. **whois [domain.be@whois.dns.be](#)**
- **telnet**:- remote login that is not
- **traceroute** :- prints the path taken to reach specific host including hop in the network path . Usually used in place of path.

The Domain Name System

- The domain name system (DNS) is a worldwide server network used to help translate easy to remember domain names like www.efossnet.org into an IP address that can be used behind the scenes by your computer.
- Here step by step description of what happens with a DNS lookup.
 - Most home computers will get the IP address of their DNS server via DHCP from their router.
 - Home router providing DHCP services often provides its own IP address as the DNS name server address for home computers.
 - The router will then redirect the DNS queries from your computer to the DNS name server of your Internet service provider (ISP).
 - Your ISP's DNS server will then probably redirect your query to one of the 13 "root" name servers.
 - The root server will then redirect your query to one of the Internet's ".org" DNS name servers which will then redirect the query to the "efossnet.org" domain's name server.

DNS (cont`d)

■ Domain

- Domains can be simply described as the name given to a family of websites.
- For example the domain linuxhomenetworking.com has a number of children such as `www.linuxhomenetworking.com` and `mail.linuxhomenetworking.com`.

■ BIND

- is an acronym for the "Berkeley Internet Name Domain" project which maintains the DNS related software suite that runs under Linux.
- The most well known program in BIND is "named", the daemon that responds to DNS queries from remote machines.

■ DNS Client

- A DNS client doesn't store DNS information; it always has to refer to a DNS server to get it..
- The only DNS configuration file for a DNS client is the `/etc/resolv.conf` file which defines the IP address of the DNS server it should use.

■ Authoritative DNS Server

- These are the servers that provide the definitive information for your DNS domain such as the names of servers and websites in it.

DNS (cont`d)

- These are the servers that provide the definitive information for your DNS domain such as the names of servers and websites in it.
- There are thirteen "root" authoritative DNS servers which all DNS servers query first.
- These servers know all the authoritative DNS servers for all the main domains such as ".com", ".net" etc.
- These servers keep track of all the sub domains beneath them.
- When you register a domain such as "my-site.com" you are actually inserting a record on the ".com" DNS servers that points to the authoritative DNS servers for your domain.

DNS (cont`d)

- Most servers don't ask authoritative servers for DNS directly, they usually ask a caching DNS server to do it on their behalf.
- The caching DNS servers then store (or cache), the most frequently requested information to reduce the lookup overhead of subsequent queries.
- Once you have set up your caching DNS server you will then have to configure each of your home network PCs to use it as their DNS server. If your home PCs get their IP addresses using DHCP, then you will have to configure your DHCP server to make it aware of the IP address of your new DNS server so that it can advertise it to its PC clients.
- Off the shelf router/firewall appliances used in most home networks will usually act as both the caching DNS and DHCP server. In this case a separate DNS server is unnecessary.

DNS (cont`d)

■ Downloading and Install the BIND Packages

- Most RedHat and Fedora Linux software products are available in the RPM format. Downloading and installing RPMs isn't hard.
- When searching for the file, remember that the BIND RPM's filename usually starts with the word "bind" followed by a version number like this: **bind-9.2.2.P3-9.i386.rpm**.
- You can use the **chkconfig** command, which updates and queries runlevel information for system services, to get BIND configured to start at boot:

```
[root@efossnet /]# chkconfig named on
```

- To start/stop/restart BIND after booting

```
[root@efossnet /]# /etc/init.d/named start
```

```
[root@efossnet /]# /etc/init.d/named stop
```

```
[root@efossnet /]# /etc/init.d/named restart
```

Note: Remember to restart the BIND process every time you make a change to the configuration file for the changes to take effect on the running process.

DNS (cont`d)

■ The /etc/resolv.conf File

- This file is used by DNS clients (servers not running BIND) to determine both the location of their DNS server and the domains to which they belong.
- It generally has two columns, the first contains a keyword and the second contains the desired value(s) separated by commas.
- Here is a list of keywords:

Keyword	Value
Nameserver	IP address of your DNS nameserver. There should be only one entry per "nameserver" keyword. If there is more than one nameserver, you'll need to have multiple "nameserver" lines.
Domain	The local domain name to be used by default. If the server is efoosNet.my-site.com, then the entry would just be my-site.com
Search	If you refer to another server just by its name without the domain added on, DNS on your client will append the server name to each domain in this list and do an nslookup on each to get the remote servers' IP address. This is a handy time saving feature to have so that you can refer to servers in the same domain by only their servername without having to specify the domain. The domains in this list must be separated by spaces.

DNS (cont`d)

- Example: a sample configuration in which:
 - The client server's main domain is my-site.com, but it also is a member of domains my-site.net and my-site.org which should be searched for short hand references to other servers.
 - Two nameservers, 192.168.1.100 and 192.168.1.102 provide DNS name resolution.

```
[root@efossnet /]# domainname my-site.com
```

```
[root@efossnet /]# search my-site.com my-site.net my-site.org
```

```
[root@efossnet /]# nameserver 192.168.1.100
```

```
[root@efossnet /]# nameserver 192.168.1.102
```

DNS (cont`d)

- The Red Hat / Fedora default installation of BIND is configured to convert your Linux box into a caching name server.
- The only file you have to edit is `/etc/resolv.conf` in which you'll have to comment out the reference to your previous DNS server (most likely your router) with a "#" or make it point to the server itself using the universal localhost IP address of 127.0.0.1
 - Old Entry

```
#nameserver 192.168.1.1
```
 - New Entry

```
# nameserver 192.168.1.1
```

or:

```
#nameserver 127.0.0.1
```

DNS (cont`d)

- Fedora versions of BIND file locations:

File	Purpose	Location Fedora Core
named.conf	Tells the names of the zone files to be used for each of your website domains.	/var/named/chroot/etc
rndc.key rndc.conf	Files used in named authentication	/var/named/chroot/etc
zone files	Links all the IP addresses in your domain to their corresponding server	/var/named/chroot/var/named

DNS (cont`d)

- Fedora takes the added precaution of using Linux's **chroot** feature to not only run **named** as user **named** but also to limit the files **named** can see.
- In Fedora, **named** is fooled into thinking that the directory `/var/named/chroot` is actually the root or `"/"` directory. Therefore **named** files normally found in the `/etc` directory are found in `/var/named/chroot/etc` directory instead, and those you'd expect to find in `/var/named` are actually located in `/var/named/chroot/var/named`.
- Fedora BIND adds to the confusion by correctly installing the files in their non **chroot** locations but never uses them.
- Before starting Fedora BIND, copy the configuration files to their **chroot** locations like this:

```
[root@efossnet /]# cp -f /etc/named.conf /var/named/chroot/etc/  
[root@efossnet /]# cp -f /etc/rndc.* /var/named/chroot/etc/
```

DNS (cont`d)

■ Configuring a Regular Nameserver

- For the purposes of this discussion, the subnet that has been assigned to us by the ISP is 192.168.1.0 with a subnet mask of 255.255.255.0
- **Configuring resolv.conf**
 - Make your DNS server refer to it self for all DNS queries by configuring the `/etc/resolv.conf` file to only reference localhost.
 - `nameserver 127.0.0.1`
- **Configuring named.conf**
 - The main DNS configuration is kept in the `named.conf` file which is used to tell BIND where to find the configuration files for each domain you own.

DNS (cont`d)

- There are usually two zone areas in this file:
 - Forward zone file definitions which list files to map domains to IP addresses
 - Reverse zone file definitions which list files to map IP addresses to domains
 - In this example the forward zone for `www.my-site.com` is being set up by placing the following entries at the bottom of the `named.conf` file.
 - The zone file is named `my-site.zone` and, though not explicitly stated, the file `my-site.zone` should be located in the default directory of `/var/named/chroot/var/named`

Dynamic Host Configuration Protocol

- Dynamic Host Configuration Protocol (DHCP), which is gradually replacing good old **bootp** in larger environments. It is used to control vital networking parameters such as IP addresses and name servers of hosts.
- DHCP is backward compatible with bootp.
- DHCP client machines will usually be configured using a GUI that configures the `dhcpcd`, the DHCP client daemon.

DHCP (cont`d)

■ The `/etc/dhcpd.conf` File

- When DHCP starts it reads the file `/etc/dhcpd.conf`.
- It uses the commands here to configure your network.
- Many RPM packages don't automatically install a `/etc/dhcpd.conf` file, but you can find a sample copy of `dhcpd.conf` in the following directory which you can always use as a guide.

`/usr/share/doc/dhcp-<version-number>/dhcpd.conf.sample`

- You will have to copy the sample `dhcpd.conf` file to the `/etc` directory and then you'll have to edit it. Here is the command to do the copying for the version 3.0p11 RPM file:

```
[root@efossnet ~]# cp /usr/share/doc/dhcp-3.0p11/dhcpd.conf.sample /etc/dhcpd.conf
```

DHCP (cont`d)

■ How to get DHCP started

- Some older Fedora / RedHat versions of the DHCP server will fail unless there is an existing `dhcpd.leases` file. Use the command "`touch /var/lib/dhcp/dhcpd.leases`" to create the file if it does not exist.

```
[root@efossnet ~]# touch /var/lib/dhcp/dhcpd.leases
```

- Use the `chkconfig` command to get DHCP configured to start at boot:

```
[root@bigboy tmp]# chkconfig dhcpd on
```

- Use the `/etc/init.d/dhcpd` script to start/stop/restart DHCP after booting

```
[root@efossnet ~]# /etc/init.d/dhcpd start
```

```
[root@efossnet ~]# /etc/init.d/dhcpd stop
```

```
[root@efossnet ~]# /etc/init.d/dhcpd restart
```

- Remember to restart the DHCP process every time you make a change to the conf file for the changes to take effect on the running process.
- You also can test whether the DHCP process is running with the following command, you should get a response of plain old process ID numbers:

```
[root@efossnet ~]# pgrep dhcpd
```

- Finally, always remember to set your PC to get its IP address via DHCP.

DHCP (cont`d)

- **Configuring Linux clients to use DHCP**
 - Remember to have your Linux based clients configured to have DHCP obtained IP addresses

Samba

- Samba is a suite of utilities that allows your Linux box to share files and other resources such as printers with Windows boxes.
- We will discuss how can we make our Linux box into a Windows Primary Domain Controller (PDC) or a server for a Windows Workgroup.
- Either configuration will allow everyone at home to have:
 - their own logins on all the home windows boxes while having their files on the Linux box appear to be located on a new Windows drive
 - Shared access to printers on the Linux box.
 - Shared files accessible only to members of their Linux user group.

Samba (cont`d)

- Samba Domains and Linux share the same usernames so you can log into the Samba based Windows domain using your Linux password and immediately gain access to files in your Linux user's home directory. For added security you can make your Samba and Linux passwords different.
- When Samba starts up it reads the configuration file `/etc/samba/smb.conf` to determine its various modes of operation. You can create your own `smb.conf` using a text editor or using the easier web based SWAT utility.
- Keep in mind that you will lose all your comments inserted in `/etc/samba/smb.conf` with a text editor if you subsequently use SWAT to edit it.

Samba (cont`d)

■ Getting SAMBA Started

- You can configure Samba to start at boot time using the `chkconfig` command:

```
[root@efossnet /]# chkconfig smb on
```

- You can start/stop/restart Samba after boot time using the `smb` initialization script as in the examples below:

```
[root@efossnet /]# /etc/init.d/smb start
```

```
[root@efossnet /]# /etc/init.d/smb stop
```

```
[root@efossnet /]# /etc/init.d/smb restart
```

- Remember to restart the `smb` process every time you make a change to the `smb.conf` file for the changes to take effect on the running process.
- You can test whether the `smb` process is running with the `pgrep` command, you should get a response of plain old process ID numbers:

```
[root@bigboy tmp]# pgrep smb
```

Samba (cont`d)

■ The Samba Configuration File

- The `/etc/samba/smb.conf` file is the main configuration file you'll need to edit. It is split into five main sections. These are:

Section	Description
[global]	General Samba configuration parameters
[printers]	Used for configuring printers
[homes]	Defines treatment of user logins
[netlogon]	A share for storing logon scripts. (Not created by default.)
[profile]	A share for storing domain logon information such as "favorites" and desktop icons. (Not created by default.)

Part II

Linux Printer Configuration

Basics

- Printer configuration that can be sued over network some times is more complex and time taking when compared with network configuration.
- Linux is more straight forward in approach and configuration of printers.
- The printing services are provided through **lpd** – a line printer daemon
- The initial service to submit print jop to lpd is the lp command. Whoever lpr is the most commonly used today.

Syntax

```
# lpr {filename}
```

- The pr command (/bin/pr)is used to perform many of the page and text formatting required by lpr

Print configuration files

- Three principal files are required for a print service to work:
 - `lpd.conf`
 - `lpd.perms`
 - `/etc/printcap`
- When `lpd` starts it read **printcap** to determine what printers are available and then processes the printer queues looking for unfinished print jobs to complete.
- The purpose of **printcap** is to hold the definition for connected printer.
- Permissions for various printer services, including `lpr`, `lpq`, and basic spooling operations are found in `/etc/lpd.perms`
- **`/etc/lpd.conf`**
 - The third file is extremely lengthy, and all of the configuration options are disabled by default. Each entry in the file takes two lines. The first line contains comments describing the parameters. The second line is command itself.
 - There are approximately about 180 configuration parameters available to optimize the `lpd` service.

Printer (cont`d)

- Controlling print service:
 - lpc: line printer control program
 - cancel active jobs: **abort**
 - enable/disable printers: **down**
 - enable/disable spools :**disable/enable**
 - hold/release print jobs: **hold/release**
 - reprint a job: **redo**
 - etc
- Graphical utility that can be used to add a printer to your system.
 - **printtool**

Part III

Job scheduling

Job scheduling

- Linux has a facility to run regularly scheduled jobs on your behalf, regardless of where you are at work or home
- If a job is to be run once the **at** command is used .
- For running jobs that run more than once on a regular basis require the use of **cron**
- Add shell script to have run hourly, daily, weekly or monthly into the appropriate directory:
 - /etc/cron.hourly/
 - /etc/cron.daily/
 - /etc/cron.weekly/
 - /etc/cron.monthly/
- These are preconfigured schedules.

at command

■ at command

- Enables you to run command at a specific time.
- The job run only once

- Syntax:

`#at {time}`

- Time is specified before the date:

- at HH:MM month-name day with an optional year
- at midnight MMDDYY
- at HH PM today
- at noon DD.MM.YY
- at 14:30 19.03.06
- at noon tomorrow
- now : runs the job now
- now + 5 days

at (cont`d)

- The daemon `/usr/sbin/atd` will run jobs scheduled with the `at` command.
- The system initially configured to only enable the root user to use the `at` command.
- Access control to the command is controlled using the files:
 - `/etc/at.allow` :-list of user id's permitted to use the `at` command and
 - `/etc/at.deny`:- if the deny file exist all users except those in `at.deny` file are permitted to use `at`.
- After entering the time specification at the command prompt `at` command will respond with PS2 prompt (it's "`at>`" prompt) upon which you enter the command you wish to execute folowed by "Enter".
- More commands may be entered. When done enter "control-d".
- The command `atq` or `at-l` enables the user to see his or her own queued jobs.
- Each spooled job is saved as a text file in the directory `/var/spool/atjobs`
- It is possible to cancel scheduled jobs prior to execution time by using `at -d` or `atrm` commands.
- `at` command used `-m` options to tell the system to mail to the user when the job is done.
- `-f` option instructs `at` to read the command list from a file.

at (cont`d)

- There is a similar command called **batch** which is almost similar with at command except it runs the commands based on the system utilization.
- If the system load is too high, batch will not run the command.

Example

```
# cat < atdemo
```

```
hostname
```

```
date
```

```
w
```

```
who
```

```
df -v
```

```
netstat -i
```

```
{ ctr +c}
```

```
# at now
```

```
at>./atdemo
```

```
at>
```

```
#atq
```

```
1 2005-12-25 10:30 tstdemon //Jobnumber date time jobs
```

Working with Cron

- cron command runs a job at a regular interval.-
Scheduling a re-occurring task
 - cron reads a file with command and times named crontab as in cron table
 - each user has his/her own crontab file.
 - Scheduling access and control:
 - The administrator can allow users to use this facility with specific control by using the `/etc/cron.deny` and `/etc/cron.allow` files.
 - The at facility may be controlled with the `/etc/at.deny` and `/etc/at.allow` files.

Cron (cont`d)

- A user can list the content of his current crontab using the command:
`#crontab -l`
- To remove crontab use:
`#Crontab -r` To edit the crontabs entries:
`#Crontab -e`
- A user can see only his/her own crontab file. but the root can see an crontab file using a command:
`#Crontab -u "username"`
- The six fields in the crontab entry are:
 - Minutes(0-59)
 - Hours(0-23)
 - Day of Month(1-31)
 - Month(1 -12)
 - Day of week (0=Sunday , 6=Saturday)

Man pages:

- You can consult the documentation page of the following commands:
 - `at` - schedule job at a specified time
 - `atq` - lists the user's pending jobs or all if root
 - `atrm` - deletes jobs, identified by their job number
 - `batch` - executes commands when system load levels permit (based on a specified system load)
 - `atrun` - for backward compatibility. Functionality built into `at`
 - `cron`
 - `crontab`
 - `ntpd` - Network Time Protocol (NTP) daemon
 - `ntpdate` - Set the date and time via NTP
 - `ntpq` - Standard NTP query program
 - `ntpc` - Special NTP query program
 - `ntpstat` - Show network time synchronisation status
 - `ntptime` - Read kernel time variables
 - `ntptrace` - Trace a chain of NTP servers back to the primary source
- Links:
 - [NTP.org](http://ntp.org)
 - [Network Time Protocol \(NTP\)](http://www.ntp.org/)
 - `/usr/share/doc/ntp-*/ntpd.html`

Thank You