**Common Linux Shell Commands**  
**ls** : list files/directories in a directory, comparable to dir in windows/dos.  
*ls -al* : shows all files (including ones that start with a period), directories, and details attributes for each file.   
  
**cd** : change directory ·· cd /usr/local/apache : go to /usr/local/apache/ directory   
*cd ~* : go to your home directory   
*cd -* : go to the last directory you were in  
*cd ..* : go up a directory

**cat** : print file contents to the screen   
*cat filename.txt* : cat the contents of filename.txt to your screen   
  
**chmod:** changes file access permissions  
The set of 3 go in this order from left to right:  
USER - GROUP - EVERONE  
  
0 = --- No permission  
1 = --X Execute only  
2 = -W- Write only  
3 = -WX Write and execute  
4 = R-- Read only  
5 = R-X Read and execute  
6 = RW- Read and write  
7 = RWX Read, write and execute  
  
Usage:   
chmod numberpermissions filename  
  
chmod 000 : No one can access   
chmod 644: Usually for HTML pages  
chmod 755: Usually for CGI scripts  
  
  
**chown**: changes file ownership permissions  
The set of 2 go in this order from left to right:  
USER - GROUP  
  
chown root myfile.txt : Changes the owner of the file to root  
chown root.root myfile.txt : Changes the owner and group of the file to root  
  
  
**tail** : like cat, but only reads the end of the file  
*tail /var/log/messages* : see the last 20 (by default) lines of /var/log/messages   
*tail -f /var/log/messages* : watch the file continuously, while it's being updated   
*tail -200 /var/log/messages* : print the last 200 lines of the file to the screen  
  
**more** : like cat, but opens the file one screen at a time rather than all at once   
*more /etc/userdomains* : browse through the userdomains file. hit *Space* to go to the next page, *q* to quit   
  
**pico** : friendly, easy to use file editor   
*pico /home/burst/public\_html/index.html* : edit the index page for the user's website.   
  
  
**File Editing with VI commands**  
**vi** : another editor, tons of features, harder to use at first than pico   
*vi /home/burst/public\_html/index.html* : edit the index page for the user's website.   
Whie in the vi program you can use the following useful commands, you will need to hit SHIFT + : to go into command mode  
  
:q! : This force quits the file without saving and exits vi  
:w : This writes the file to disk, saves it  
:wq : This saves the file to disk and exists vi  
:LINENUMBER : EG :25 : Takes you to line 25 within the file  
:$ : Takes you to the last line of the file  
:0 : Takes you to the first line of the file  
  
**grep** : looks for patterns in files   
*grep root /etc/passwd* : shows all matches of root in /etc/passwd  
*grep -v root /etc/passwd* : shows all lines that do not match root   
  
**ln** : create's "links" between files and directories  
*ln -s /usr/local/apache/conf/httpd.conf /etc/httpd.conf* : Now you can edit /etc/httpd.conf rather than the original. changes will affect the orginal, however you can delete the link and it will not delete the original.   
  
**last** : shows who logged in and when  
*last -20* : shows only the last 20 logins   
*last -20 -a* : shows last 20 logins, with the hostname in the last field   
  
**w** : shows who is currently logged in and where they are logged in from.  
who : This also shows who is on the server in an shell.  
  
**netstat** : shows all current network connections.  
*netstat -an* : shows all connections to the server, the source and destination ips and ports.  
*netstat -rn* : shows routing table for all ips bound to the server.  
  
**top** : shows live system processes in a nice table, memory information, uptime and other useful info. This is excellent for managing your system processes, resources and ensure everything is working fine and your server isn't bogged down.  
*top* then type *Shift + M* to sort by memory usage or *Shift + P* to sort by CPU usage  
  
**ps:** ps is short for process status, which is similar to the *top* command. It's used to show currently running processes and their PID.  
A process ID is a unique number that identifies a process, with that you can kill or terminate a running program on your server (see kill command).  
*ps U username* *:* shows processes for a certain user  
*ps aux* : shows all system processes  
*ps aux --forest* : shows all system processes like the above but organizes in a hierarchy that's very useful!  
  
**touch** : create an empty file   
*touch /home/burst/public\_html/404.html* : create an empty file called 404.html in the directory /home/burst/public\_html/   
  
**file** : attempts to guess what type of file a file is by looking at it's content.   
*file \** : prints out a list of all files/directories in a directory   
  
**du** : shows disk usage.   
*du -sh* : shows a summary, in human-readble form, of total disk space used in the current directory, including subdirectories.  
*du -sh \** : same thing, but for each file and directory. helpful when finding large files taking up space.   
  
**wc :** word count  
*wc -l filename.txt* : tells how many lines are in filename.txt   
  
**cp** : copy a file   
*cp filename filename.backup* : copies filename to filename.backup  
*cp -a /home/burst/new\_design/\* /home/burst/public\_html/* : copies all files, retaining permissions form one directory to another.   
cp -av \* ../newdir : Copies all files and directories recurrsively in the current directory INTO newdir  
  
**mv** : Move a file command  
mv oldfilename newfilename : Move a file or directory from oldfilename to newfilename  
  
**rm** : delete a file  
*rm filename.txt* : deletes filename.txt, will more than likely ask if you really want to delete it  
*rm -f filename.txt* : deletes filename.txt, will not ask for confirmation before deleting.  
*rm -rf tmp/* : recursively deletes the directory tmp, and all files in it, including subdirectories. BE VERY CAREFULL WITH THIS COMMAND!!!   
 **TAR**: Creating and Extracting .tar.gz and .tar files  
tar -zxvf file.tar.gz : Extracts the file  
tar -xvf file.tar : Extracts the file  
tar -cf archive.tar contents/ : Takes everything from contents/ and puts it into archive.tar  
gzip -d filename.gz : Decompress the file, extract it  
  
**ZIP Files**: Extracting .zip files shell command  
unzip file.zip  
  
  
**Firewall - iptables commands**iptables -I INPUT -s IPADDRESSHERE -j DROP : This command stops any connections from the IP address  
iptables -L : List all rules in iptables  
iptables -F : Flushes all iptables rules (clears the firewall)  
iptables --save : Saves the currenty ruleset in memory to disk  
service iptables restart : Restarts iptables   
  
**Apache Shell Commands**  
httpd -v : Outputs the build date and version of the Apache server.   
httpd -l : Lists compiled in Apache modules  
httpd status : Only works if mod\_status is enabled and shows a page of active connections   
service httpd restart : Restarted Apache web server  
  
**MySQL Shell Commands**  
mysqladmin processlist : Shows active mysql connections and queries  
mysqladmin drop databasenamehere : Drops/deletes the selected database  
mysqladmin create databasenamehere : Creates a mysql database  
  
Restore MySQL Database Shell Command  
mysql -u username -p password databasename < databasefile.sql : Restores a MySQL database from databasefile.sql   
  
Backup MySQL Database Shell Command  
mysqldump -u username -p password databasename > databasefile.sql : Backup MySQL database to databasefile.sql  
  
**kill:** terminate a system process  
*kill -9 PID* EG: *kill -9 431  
kill PID* EG: *kill 10550*  
Use *top* or *ps ux* to get system PIDs (Process IDs)   
  
EG:

|  |  |  |  |
| --- | --- | --- | --- |
| PID | TTY | TIME | COMMAND |
| 10550 | pts/3 | 0:01 | /bin/csh |
| 10574 | pts/4 | 0:02 | /bin/csh |
| 10590 | pts/4 | 0:09 | APP |

Each line represents one process, with a process being loosely defined as a running instance of a program. The column headed PID (process ID) shows the assigned process numbers of the processes. The heading COMMAND shows the location of the executed process.   
  
**Putting commands together**  
Often you will find you need to use different commands on the same line. Here are some examples. Note that the | character is called a pipe, it takes date from one program and pipes it to another.  
*>* means create a new file, overwriting any content already there.   
*>>* means tp append data to a file, creating a newone if it doesn not already exist.   
*<* send input from a file back into a command.   
  
*grep User /usr/local/apache/conf/httpd.conf |more*This will dump all lines that match User from the httpd.conf, then print the results to your screen one page at a time.   
  
*last -a > /root/lastlogins.tmp*   
This will print all the current login history to a file called lastlogins.tmp in /root/   
  
*tail -10000 /var/log/exim\_mainlog |grep domain.com |more*This will grab the last 10,000 lines from /var/log/exim\_mainlog, find all occurances of domain.com (the period represents 'anything',   
-- comment it out with a so it will be interpretted literally), then send it to your screen page by page.   
  
*netstat -an |grep :80 |wc -l*Show how many active connections there are to apache (httpd runs on port 80)   
  
*mysqladmin processlist |wc -l*Show how many current open connections there are to mysql

**ip addr show  OR  /sbin/ifconfig**

To find out IP address of Linux/UNIX/BSD/Unixish system you need to use command called ifconfig. It is used to configure the kernel-resident network interfaces. It is used at boot time to set up interfaces as necessary. After that, it is usually only needed when debugging or when system tuning is needed. If no arguments are given to ifconfig command it displays the status of the current active interfaces. It displays Ethernet IP address, Mac address, subnet mask and other information.

**Note:**

### **1) Get Single IP Address by Interface**

**return the plain IP Address**

**ifconfig** $1 **|** **grep** "inet addr" **|** **awk** -F: '{print $2}' **|** **awk** '{print $1}'

### **2 Get Every Interfaces IP Address**

Returns every interface and IP address pairs.

**ifconfig** **|grep** -B1 "inet addr" **|awk** '{ if ( $1 == "inet" ) { print $2 } else if ( $2 == "Link" ) { printf "%s:" ,$1 } }' **|awk** -F: '{ print $1 ": " $3 }'

Create simple bash function (example *int-ips*) with following command.

**function** int-ips **{** **ifconfig** **|grep** -B1 "inet addr" **|awk** '{ if ( $1 == "inet" ) { print $2 } else if ( $2 == "Link" ) { printf "%s:" ,$1 } }' **|awk** -F: '{ print $1 ": " $3 }'; **}**