**Essential Commands**

Log in to Local and Remote Graphical and Text Mode

**Consoles**

Both Ubuntu/Debian and CentOS/Redhat

**VNC Viewer** is a tool used to connect to a VNC Server which allows for remote operation of a Linux distribution in graphics mode. It is similar to other technologies such as Remote Desktop for Windows. VNC ordinarily runs over tcp/ip ports starting at 5800, but can be configured.

**Secure Shell or ssh** is used to connect to remote servers at a command line,ordinarily over port 22. The connection is encrypted, and while it usually relies on username/password authentication, it can be configured instead to make use of user-generated keys for authentication purposes.

**Search for Files**

Linux is case sensitive! This means that “TEST”, “test”, “Test”, and “TesT” all register as completely different.

**find –help** for all of the **find** command options

Remember it also has a man entry:

**man find**

Examples:

Find all files in the entire system that are named test.txt:

find / -name “test.txt”

Find all files in the /etc directory not named test.txt:

find /etc/ -not -name “test.txt”

Find all character devices on the system:

find / -type c

Find all directories called log:

find / -type d -name “log”

Find all files in the /usr/bin directory or subdirecties that are larger than 27,000 bytes:

find /usr/bin -size +27000c or find /usr/bin -size +27k

Find all files in the /usr/bin directory or subdirectories that are larger than 1 megabyte:

find /usr/bin/ -size 1M

Find all files created more than a day ago:

find / -mtime 1

Find all files create less than a day ago:

find / -mtime -1

Find all files owned by the user named “chad”:

find / -user chad

Find all files in the /etc directory owned by the user root, and paginate the results:

find /etc/ -user root | more

Find all files in the /usr/bin directory with user permissions of 755:

find /usr/bin -perm 755

Find all files called test.txt and set their permissions to 700:

find / -name “test.txt” -exec chmod 700 {} ;

The **more** command will cause the output to pause after it fills a page and wait for you to hit the spacebar before continuing.

There is also a **less** command that will allow you to scroll back and forth through the output using arrow keys.

The **chmod** command changes a file’s permissions

The **which** command allows you to find the location of a command that will be **executed** if invoked.

This is useful for locating the **executable** file, especially if the behavior is not what you expected.

**which**

For instance, to find out which python executable would be invoked:

which python

The **locate** command can also help you find files but relies on a database rather than looking directly at the file system. The two can easily be out of sync. To synchronize locate’s database, run the command (as root) **updatedb.**

**locate**

**Compare and Manipulate File Content and Use Input-**

**Output Redirection**

The **cat** tool will read file contents to standard out, which is usually the screen.

The **more** command pauses output.

The **pipe** character takes the standard output from one command and passes it to another command. For example, to send the output from cat shoppinglist through more:

cat shoppinglist | more

or

more shoppinglist

The **less** command will paginate, similar to more, but it will also allow you to use your up and down arrow keys to scroll backward and forward through the file.

cat shoppinglist | less

or

less shoppinglist

The **sort** command sorts content passed to it on standard in and places the output on standard out.

sort shoppinglist

To sort in reverse order:

sort -r shoppinglist

The **cat(categories)** tool’s name is short for concatenate. You can combine the output of multiple files like this:

cat file1 file2

You can achieve complex results by combining commands:

cat file1 file2 | sort > file1and2

The **>** and **<** symbols allow you to send the standard output to a particular file on disk. The above example would have sorted the two files called file1 and file2 together and created a file in the current working directory called file1and2.

The **fmt** (format) command will clean up and format text.

The **nl** command counts the number of lines in a file, and also will number each line in a file for reference. ( satır numaralandırma)

The **cut** command will remove a specified delimiter and write out a part to standard output.

One;Two

Three]Four

If we execute the following on the above file called numbers,

cat numbers | grep “;” | cut -d “;” -f1

The output would be One. ( dosyadan ; 1 numara dısında herseyi keser, gösterir)

If we execute the following on the above file

cat numbers | grep “;” | cut -d “;” -f2,

then the output would be Two. ( dosyadan ;2 numara dısında herseyi keser,gösterir)

**Analyze text using basic regular expressions**

Some basic regular expression help:

|  |  |
| --- | --- |
| Token | **Meaning** |
| ^ | Start of line |
| $ | End of line |
| \< | Start of word |
| \> | End of word |
| \d | Digit |
| \D | Not a digit |
| [Aa1] | Either match on A,a or 1 |
| [A-Z] or [:upper:] | Matches on any capital letter |
| a-z] or [:lower:] | Matches on any lower case letter |
| [^lmno] | Matches on any charactr but l, m, n. or o. |
| \$ | Matches a literal dollar sign |

^A : line daki A ile başlayan satırlar

Z$ : line daki Z ile biten satırlar

grep ‘^The’ alice.txt : alice.txt deki THE ile başlayan satırları gösterir.

grep ‘^T[a-z][aeiou]’ alice.txt : alice.txt deki T ile başlayan,2.harfi a-z arasında olan ve 3.harfi seçilen(aeiou) harflerden olan satırları gösterir.

grep ‘^T[a-z][^e]’ alice.txt : alice.txt deki line’larda T ile başlayıp ikinci harfi bilmiyorsak ve 3.harfi “e” ile bitmeyen ( This, Those vs.) satırları göster.

grep ‘\<[tT]he\>’ alice.txt : alice.txt deki küçük ya da büyük t ile başlayan ve he ile devam eden kelimeleri gösterir.

**Archive, Backup, Compress, Unpack, and Uncompress**

**Files**

\*\***mkdir**: make a new directory ( mkdir data\_backup)\*\*

\*\***gzip**: dosya sıkıstırma komutu(gzip data\_backup.tar )\*\*

\*\***rm** : dosya silme (remove) komutu ( rm data\_backup.tar.gz)\*\*

\*\***clear ( Control+l)**: terminali temizler

The cp command allows you to directly copy files from one location to another.

cp (copy)

Such as:

cp /home/user/test.txt /opt/test.txt

The **scp** command allows you to copy files securely over the network to another host running an ssh server.

scp

Such as sending a file to another host:

scp /home/user/test.txt user@otherhost:/home/user/test.txt

Or retrieving a file from another host:

scp user@otherhost:/home/user/test2.txt /home/user/test2.txt

The **rm** command immediately deletes files.

rm unneededfile.txt

The **tar** (**t**ape\_**ar**chieve) command is the de facto standard for creating backups.

To create an uncompressed backup:

**tar cvf** backupfile.tar /path/to/backup

To create a compressed backup:

**tar cfzv** backupfile.tar.**gz /**path/to/backup

Mnemonics for commonly used tar flags:

|  |  |
| --- | --- |
| Flags | Means |
| c | Create |
| v | Verbous |
| f | File |
| z | Zip |
| x | Extract |
| t | List |

The **grep** tool allows you to search for a **particular** output. It accepts input on standard in and puts only lines containing the pattern you are looking for back out.

tar tvf databackup.tar | grep searchingfor.txt

The **gzip** tool is a compression utility that will compress and uncompress files. Text files compress very efficiently, so zipping them up will result in a far smaller file, with the disadvantage that it will have to be uncompressed before it can be read. By default, it will compress files “in place,” meaning it will take a file, compress it into the same name with a “.gz” appended to the end, and remove the original file.

gzip data\_backup.tar

Creates in a file in the directory called somefile.tar.gz, but the original somefile.tar is gone.

To create a compressed backup file, use the compression options in tar:

tar cvfz databackup.tar.gz /data

To extract the contents of a compressed tar file to the current directory:

tar xvfz databackup.tar.gz

**Create, Delete, Copy and Move Files and Directories**

The **touch** command quickly creates a zero-byte file. ( 0 byte lık bir dosya yaratır)

touch testfile/data

The **cp** command allows you to copy files from one location to another.

cp testfile testfile2

The **nano** command is a simple text editor. On some Ubuntu systems, there is an alias to this for users of the older, deprecated pico editor.

Pico testfile2

The **mkdir** command allows you to create directories.(Klasör yaratır)

mkdir secondfiles

You can create several at once using the -p flag: (alt klasör olusturma)

mkdir -p secondfiles/textfiles/otherfiles/resources

The **mv** command is used for moving files from one directory to another. It’s also used for renaming files. Think of renaming as just moving a file from one name to another:

mv testfile secondfiles/

The **rm** command allows you to delete or remove files (and sometimes directories with the right flags and wildcards).

rm testfile

or, beware, a command like this will remove all files and directories in /home/user that start with “unneeded” and, if a subdirectory, remove it completely:

rm -rf /home/user/unneeded\* ( klasör ya da altkalsör silme yaparken en sondan başlarız)

The **rmdir** command is for specifically removing empty directories. It will fail if there is anything in the directory. !!!! For your own safety, get in the habit of using rmdir to remove directories, and use those error messages to your advantage so you don’t accidentally wipe out half (or more) of your important files.

\*\*cd .. : Bir önceki dizini okuma komutu \*\*

**Create and Manage Hard and Soft Links**

The directory structure on your hard drive is basically a list of named pointers (directory file names) to locations on the disk. A **hard link** is a direct pointer to that location, and a soft link is a link to another file name in the directory structure. A file on disk can have multiple hard links pointing to it so that you can edit the file from multiple directory locations.

\*\* pico file1 yarattık

\*\* pico file2 yarattık ... Bunlar block device da hard link denir.

\*\* mkdir dir1

\*\* mkdir dir2 klasörleri olusturduk.

\*\*mv file1 dir1/ ( file1 dosyasını dir1 klasörüne gönderdik)

\*\*mv file2 dir2/ ( file2 dosyasını dir2 klasörüne gönderdik)

The **ln(link)** tool allows you to create both hard and soft links to files.

To create a hard link:

\*\*dir1 klasöründen file2 dosyasının içindeki dataya erişim için hard-link olusturuldu.(hard-line-to-file2 isminde)

ln ../dir2/file2 hard-line-to-file2

To create a soft link: -s (symbolic)

ln -s ../dir1/file1 file1

\*\*dosyadaki datalar herhangi bir klasörde editlense bile değiştirilen datalar diğer klasörden görülebilir.( hard-link)

\*\*aynı klasörde farklı sistemlerde bir klasörü spesifik olarak tanımlayıp erişme (soft-link..symbolic link..sim links..)

**List, Set, and Change Standard File Permissions**

\*ls -la = ll >> listeleme komutu

\*chmod >> change mode ( permission değiştirmede kullanılır)

The **chmod** tool allows you to set permissions on files.

Directories listed with the -la flag show privileges in three sets of characters like this:

drwxrwxrwx 24 owner group size date filename

The first character tells you whether the entry represents a directory or not:

d— — —

The next three characters tell you which of the read, write, and execute rights that

the file’s owner has:

-rwx— —

The next three characters tell which of the read, write, and execute rights that the file’s group has:

—-rwx—

The next three characters tell which of the read, write, and execute rights that everyone else has:

— —-rwx

To change file permissions, you can use chmod with the character method like below.

1) To add group read and write access to a file:

chmod g+rw file

( Not: eğer everyone(other) için yapsaydık; chmod o+rwx filename )

(Not: eğer herkes ( user,group,everyone) için yapsaydık ; chmod a+rwx filename )

2)To summarize the character method, a truncated usage statement is presented below:

chmod [ugoa]\*([-+=][rwx]\*[ugo]

4: read

2:write

1: execute

0: not permission

7: all(a) ( read+write+execute)

6: read+write

5: read+execute

3: write+execute

So if you want to give the owner full rights, the group read and write rights, and everyone else only read rights to a file, you’d use:

owner group everyone

chmod 7 6 4 filename

Justification:

Math Characters Meaning

7 = 4 + 2+ 1 rwx Full rights.

6 = 4 + 2 rw- Read/Write

4 = 4 r– Read only

**Manage Access to the root Account**

Protect your server. Don’t log in as root and don’t allow anyone else to, either. Don’t share a single password. Use the **sudo** command to escalate your privileges.

Change to the root user (**su or sudo su**) only when necessary, and log out as soon as you’ve completed your tasks.

The **sudo** command allows you to “do” things as the “super user” that you ordinarily wouldn’t have appropriate permissions to do.

sudo

sudo !!( önceki komutu çalıştırır.)

You’ll be asked for YOUR password to ensure that it’s you.

Use the **su** command to become another user, usually the root user.

su root (#root is implied if you only type “su”) : super user

You will be prompted for the password belonging to the user whose identity you are attempting to assume.

The **whoami** command will return the name of the Linux user you are currently logged in as.

root@somehost$ whoami

root

To provide a user with access to use the sudo command, use **visudo** to edit the sudoers file:

visudo

Add the user like others in the file.

Alternatively, you can set up a group that has sudo privileges in the sudoers file, and you can then assign users to the group to dynamically alter privileges

usermod -a -G wheel chad

su chad

**Operation of Running Systems**

**Boot, Reboot, and Shut Down a System Safely**

The **shutdown** command allows you to safely shut down the system while notifying users of what’s going on.

To power off a system now with a message of “System Maintenance – New Hard drive!

Back online in half an hour or so!”:

shutdown -h now “New Hard drive! Back online in half an hour or so!”

Common (but not complete) usage (Always see man pages for complete documentation)

shutdown -[h - halt | r - reboot ] <when?>

Shortcut commands reboot and poweroff do exactly what you’d expect them to do.

Note that changing the running status of the system requires root privileges.

NOTE:

1.Verify completion of scheduled jobs: cd /var/log -> cat syslog | grep CRON

2. reboot or shutdown a system : shutdown -r now,, shutdown -h now

3.cmd to see how long a computer has been running : uptime

**Install, Configure, and Troubleshoot Bootloaders**

**Grub 2** is the de facto entrenched bootloader in use by most Linux distributions as of the writing of this guide.

**/boot/grub/grub.cfg** is grub’s main configuration file. It is autogenerated and should not be manually edited. It is autogenerated by the **update-grub tool.**

**/etc/grub.d** is a configuration directory used by **the update-grub tool** to generate the grub.conf file and also specified menu entries.

The sample grub code I created in the video is:

#!/bin/sh -e

echo "displayed when update-grub is run"

cat << EOF

menuentry "Other Linux Partition" {

set root=(hd0,3)

linux /boot/vmlinuz

initrd /boot/initrd.img

}

grub-install /dev/hda

grub-install –-root-directory=/mnt /dev/sda

**Diagnose and Manage Processes**

The **top** command will show you a real-time, live updated console containing processes, their CPU utilization, and lots of other great information.

The **htop** command offers similar functionality, with a few more interactive options and major cosmetic differences.

The **ps** command will allow you to find processes running on the current machine to identify their owner or pid (process id).

Common ps invocation:

ps aux

The **kill** command is used to send termination signals to running applications.

Here is a brief list of kill signals commonly used (not complete). ( kill -l)

NOTE: List of all running process: ps aux

NOTE: Linux system priorities : 0 to 139. 0 to 99 for real time. 100 to 139 for users

NOTE: start process with lowest priority level: nice -n 20 <programme>

NOTE: how to see hierarchy of process : ps acjf

NOTE: anothr way to find specific process running using ps: ps aux | grep "whatever"

**Locate and Analyze System Log Files**

Usually in /var/log/messages for CentOs and /var/log/syslog for Ubuntu, most relevant logs are located there. For application files, consult their documentation. ( messages log file herseyi gösterir)

specific olarak Error loglarını görmek için;

grep “Error” messages

Shift + G : son satıra getirir

Cat, less, more, grep, and input-output redirection covered in a previous lesson are your best friends with log file analysis.

**Schedule Tasks to Run at a Set Date and Time**

A user’s **crontab** contains a list of commands a user wishes run at very particular times. Thesecommands are run by a daemon called cron. To view a user’s crontab, use this command (if you only want to see your own, you need not include the username):

crontab -u user -l

To delete a user’s entire crontab (all commands GONE!):

crontab -u saduser -r (remove)

To edit a user’s crontab:

crontab -u user -e (edit)

Here is an example of running a script called “/opt/do-the-things.sh” at 8:00 AM every weekday in March:

0 8 \* 3 1,2,3,4,5 /opt/do-the-things.sh

m h dom mon dow command

m: minute

h : hour

dom: day of month

mon: month

dow: day of week

**Position Meaning**

1 Minutes after the hour the scheduled command is to run.

2 Hour of the day

3 Day of the month (1 - 31)

4 Month (1 - 12)

5 Day of the week (0-6) (Sunday = 0)

6+ The command

**Update and Manage Software to Provide Required Functionality and Security**

**Part 1: Ubuntu/Debian**

On Debian/Ubuntu systems, the package management system is Aptitude and the two common tools used with it are **apt** and **apt-get**. Apt-get tends to receive preferential treatment because it doesn’t do fancy stuff to the screen, thus making it ideal for inclusion in scripts.

Often, you’ll need to update the repository cache by running an update:

apt **update**

or

apt-get update (update all of list or all of packages)

And to update all out-of-date packages:

apt **upgrade**

or

apt-get upgrade

On most systems, you will need escalated privileges, so remember to sudo where necessary.

To install a new package, use:

apt install or apt-get install

To remove a package, you will usually want to remove everything associated with it, like so:

apt purge

or

apt-get purge nmap = apt-get remove - - purge nmap

The **dpkg** tool is used when you already have a Debian package file (\*.deb) that you want to query or install.

To install a package file, use the command:

**dpkg** -i somepackage.deb \*\*-i ( install)

**Part 2: CentOS/Redhat**

On Redhat and CentOS, the Redhat Package Management system is in place. The packages are **.rpm** files and the package management command, the equivalent to Debian’s dpkg command is also called **rpm.** There is an apt type of system called **yum**.

To update the repository cache and packages on the system:

yum update

This lets us know if a package is installed:

yum list

To find out if a packagename exists (wildcards are okay): \*\*\*nmap: namepackages

yum search nmap

To install a package and any necessary dependencies:

yum install

The **-y** switch will answer yes to all questions so that we’re not prompted during an install:

yum install -y

The yum-utils package includes some excellent additional tools, such as yumdownloader which fetches the package but does not install it.

The **yum-list** command will list all packages available in the database (grep is helpful here).

Display enabled yum repositories:

yum repolist

Display ALL yum repositories:

yum repolist all

The rpm tool is useful when you have a .rpm file that you’d like to install or query. To install a package from an rpm:

rpm -i somepackage.rpm

List all files installed by a particular package:

rpm -ql

Examine docs inside a package:

rpm -qdf  
**Flashcards**

**"yum update"** to update packages  
**"yum search (package name)"** for information on package  
**"yum install (name of package)"** to install package  
**"yumdownloader (name of package)"** to download the file/package without installing  
**"yum list"** shows all package names  
**"yum search (name of package)"** to find a specific package  
**"yum list installed"** tells you the packages installed on the system  
**"yum list installed | grep cron"** will show you all the cron packages in the system  
**"yum grouplist"** shows groups available as some applications are grouped together for a node  
"**yum groupinstall 'name of group'"** for installing groups  
**"yum repolist"** shows all repos  
**"yum repolist all"** shows all the repos whether enabled or disabled  
**"yum --enablerepo=extras-source/7 pkgname"**  
**"yum clean all"** will clean or remove disabled or repo packages  
**"yum history"** to show the history of what's been done recently

**yum install xterm** looks for all dependencies and packages in order to install onto the system  
In order to remove packages and the dependences, use **"yum autoremove"** to remove dependencies

"**rpm -qpR (package)"** to query dependencies in a specific package (xterm for example)  
**"rpm -q (package name)"** to determine if the package is installed  
**"rpm -ql (package)"** to show all the files that were installed along with the specific package  
**"rpm -qa --last"** shows all the recently altered packages  
**"rpm -qdf /user/bin/vmstat"** shows everywhere vmstat is shown in the documentation dependency  
**rpm -qa gpg-pubkey\*** tells you the public keys that have been installed in order to access different repositories.  
**rpm --rebuilddb** to rebuild the database  
**rpm -i <pkg\_name>** \*\*\* install\*\*\*  
**rpm -r <pkg\_name>** \*\*\* remove \*\*\*  
**rpm -q <pkg\_name>** \*\*\* query \*\*\*

**Change kernel Runtime Parameters, Persistent and Non-Persistent**

The **sysctl** tool is made for examining and changing kernel parameters at runtime. It examines the virtual process file system in **/proc.**

To change a system parameter until the next boot:

sysctl parameter.name=value

To change a system parameter permanently, you’ll need to change the appropriate parameter by editing the appropriate file in /etc/sysctl.d/.

Flascards

**- sudo sysctl -a** (lists all of the different kernel runtime parameters)  
**- sudo sysctl -a | wc -l**  
**- /proc/sys** (location of kernel runtime parameters)  
  
Check value of kernel parameter:  
**- cat /proc/sys/net/ipv4/ip\_forward**  
OR  
**- sysctl net.ipv4.ip\_forward**  
  
Persisting changes:  
**- cd /etc/sysctl.d**  
  
Modifying .conf files (add comment(s) describing changes made and then add the changed parameter - Ex: **net.ipv4.ip\_forward=1)**

**sysctl -w** <param>=number -- will change for session  
**sysctl -p** will make persistant.  
----------------  
example  
  
sysctl -w vm.swappiness=10 will set swappiness to 10 for session.  
  
can also echo number > /proc/sys/vm .....  
  
sysctl -p will make persistant although may not survive reboot.  
  
**changes should be made to /etc/sysctl.conf files etc. !!!!!!**

**Use Scripting to Automate System Maintenance Tasks**

1. Always start with a **shebang**. That tells the system how to interpret the script.

#!/bin/bash

And point to an appropriate shell or interpreter.

2. Scripts are executed sequentially. The order of the commands matters.

3. Avoid relative paths like **../../home/**. Instead, use absolute paths like /home. This way, if you ever decide to move your script, you don’t have to go back in and fix it immediately just because it lives in a new directory.

4. Log it! Get in the habit of “echoing” important information to the screen or writing

results to either your own log file or appending it to the system log file. This can be very useful for debugging larger, more complex scripts later on.

Add to PATH : export PATH=$PATH:/home/chad/script

**Manage the Startup Process and Services (In Services**

**Configuration)**

In SystemD systems such as Ubuntu 14+, Centos, and RedHat systems, starting a service at boot is simply a matter of using the **systemctl** tool. Starting one of these services at boot is referred to as “enabling” it:

systemctl start/stop cron

To gain information about a particular service on a systemd system, you’d use:

systemctl status servicename

In older systems using **upstart** for services management, you’d create an appropriate configuration script in /etc/init.

To gain status information on upstart systems, you’d type:

status cron\*

**Flascards**

In Older System

**cd /etc/init** (contains configuration files for telling the OS how to get up and running)  
  
**- status cron  
- stop cron  
- start cron**

**-restart cron ((( her işlemde yeni pid<process id> verir)))**  
  
- **echo manual | sudo tee /etc/init/cron.override** (to manually start cron instead of it starting on system boot)

**- rm cron.override** (to reenable automatic startup of cron upon system boot)

In Newer System

**systemctl start <cron>**  
**systemctl status**  
**systemctl stop <cron>** (needs sudo)  
**systemctl enable/disable <cron>** (needs sudo) - **is persistent.**  
  
enable/disable basically creates/removes softlinks to the daemon files within /etc/systemd to service file.  
  
**crond for example**. symlink /etc/systemd/system/multi-user.target.wants/crond.service --> /usr/lib/systemd/systemd/crond.service

**List and Identify SELinux/AppArmor File and Process Contexts**

**SELinux (Redhat/Centos)** and **AppArmor (Ubuntu/Debian)** prevent applications on their respective systems from overstepping due to defects or malicious behavior.

**SELinux**

SELinux can be enabled/enforced, logged, or disabled.

To view all contexts on your system:

semanage fcontext -l

To find a specific context:

semanage fcontext -l | grep httpd

To view security context on a file:

ls -Z filename

ps auxZ

Used without a file name the command would show the security context of every file in a directory.

To show security context on processes:

ps auxZ

**AppArmor:**

The **aa-status** tool will show you similar security contexts on AppArmor systems.

Profiles are stored in the /etc/apparmor.d directory and you can view these profiles and see how they are constructed.

Regular expression experience is helpful, so you might want to review the basic regular expression lesson.

**ls -Z** is usually available on Ubuntu systems, but does not work for AppArmor profiles.

ps auxZ, however, does work

**Identify the Component of a Linux Distribution That a File Belongs To**

Specifically, you can find out which package provides a particular file on Redhat and

Centos systems:

rpm -qf

or

yum whatprovides

\*\*rpm -qf */bin/znew*

*\*\*yum whatprovides /bin/znew*

On Debian/Ubuntu machines, we’ll use **dpkg**:

dpkg -S

*\*\* dpkg -S /usr/bin/zdump*

*\*\*dpkg -L libc-bin*