GRIG̃ALE

Linux User Pocket guide (ver 1.1) **www.grigale.com©**

Legend

[ ] - an optional syntax within brackets, for a command.
[a|b] - an optional syntax within brackets, you can use syntax – a or syntax – b, for a command.
< > - a must have syntax within brackets, for a command.

User commands

id – returns the user identity including login name and user id.
who [-a] – lists who is in the system. -a lists also process information.

File and directory operations

cd – changes the current directory to be the home directory.
cd - – changes to the previous directory.
file f1 – displays the file type of file: f1.
cat f1 – displays the content of ascii file: f1(on binary file this may damage your terminal).
mv f1 f2 – renames file: f1 to be file: f2
mv d1 d2 – renames directory: d1 to be directory: d2
mv f1 d1 – moves file: f1 to directory: d1.
cp f1 f2 – copies file: f1 to file f2.(overwrite!).
cp f1 f2 d3 – copies files: f1 and f2 to directory: d3.
rm f1 f2 – deletes files: f1 and f2
rm -rf d1 – removes the whole content of directory: d1

ln f1 f2 – creates file f2 to be a hard link of file f1. (can not cross filesystem boundaries)
ln -s f1 f2 – creates file f2 to be a soft link of file f1.
touch f1 – creates an empty file f1.
mkdir d2 – creates directory: d2

mkdir -p /d1/d2/d3/d4 – creates the directory d4 with all of its sub-directories. (if not exist).

rmdir dir1 – removes dir1 only if empty.

ls -l – shows only the visible files, including list of available attributes of the files.
ls -a – shows files, including hidden files(files that begin with a dot).
ls -altr – shows files with sorting them by modification time.

File content commands

file f1 – displays the type of the file f1.
more f1 – displays content of a file in pages. allows back (b) or forward (SPACE) of a pages.
cat f1 – displays the entire content of the file f1.

Filesystem commands

df -h – shows all file systems and their Usage.
du -sh /dir1 – displays a human readable summary size of /dir1.

File permission operations

chown [-R] user1[:group1] <f1|dir1> – changes the ownership of the file f1 or directory dir1 to be with user ownership – user1 and group ownership – group1. -R – will perform recursive operation on all the content of dir1 and it's sub-directories.
chgrp [-R] group1 <f1|dir1> – changes the ownership of the file f1 or directory dir1 to be with group ownership – group1. -R – will perform recursive operation on all the content of dir1 and it's sub-directories.
chmod [-R] u+x <f1|dir1> – allows only the owner of the file to execute file **f1**. (if you want all to execute use **a** instead of **u**) . -R – will perform recursive operation on all the content of dir1 and it's sub-directories.
chmod [-R] g-w <f1 | dir1> – don't allow someone in my group to modify file **f1**. -R – will perform recursive operation on all the content of dir1 and it's sub-directories.
chmod o+r f1 – allows other users or groups to read file **f1.**
chmod 755 d1 – allows the following permissions on directory **dir1: rwxr-xr-x**
chmod 664 f1 – allows the following permissions on file **f1: rw-rw--r--**
chmod 644 f1 – allows the following permissions on file **f1: rw-r—r--**
umask 022 – allows to create directories or files with permissions of 777-022=**755** for a directory or 666-022=**644** for a file
umask 077 - allows to create directories or files with permissions of 777-077=**700** for a directory, or 666-077=**600** for a file.

Searching files and directories

find . |grep hello – the fastest way to find the path of the file hello, if it exists.
find /etc -name "ifcfg*" - finds the files that begin with **ifcfg**.
find /etc -name core -exec rm {} \; - finds the files named core in the /etc directory and remove it.
find /tmp -name core -ok rm {} \; - finds the files named core in the directory: **/tmp** and remove them. (ask before any removal).
grep hello f1 - prints all lines in the file: f1, that contain the pattern: **hello**
grep -n hello f1 - do the same as above and also print the line number.
grep -v hello f1 - prints all the lines the that do not contain the word hello.

Handling Jobs and Processes

top – shows online process table with CPU and Memory usage. **q** – to quit, l – shows load, t – shows tasks/process, m – shows memory consumption.
ps | jobs – shows all the running processes / jobs ,

which started from the current shell
ps -ef – shows a all the processes in the system.
ps -ef | grep xclock – shows all the processes in the system that have xclock in their name or as a parameter.
pgrep -lf xclock – the same output as the command above.
kill -l – lists available signals.
kill -9 1101- kills a process which pid number is: **1101**.
kill -9 %1 – kills job number 1.
xkill – kills a hanged GUI.

Printer commands

lp file1 – prints **file1** to the default printer.
lp -d printerA file1 – prints **file1** to printer **printerA**.
lpstat -d – shows the defualt printer on the system with it's current status.
lpstat printerA – shows all available jobs on **printerA**.
cancel printerA-1 – cancels job request **printerA-1**.
enable printerA – enables **printerA** queue.
pstat printerA – disables **printerA** queue.

Pipes and redirections

echo hello – prints hello to the screen.
echo $path – prints the value of the variable path.
alias h = "echo hello;date" - creates an alias command named h that will do all the commands written in the right.
history – lists all the commands that have already been executed in the current shell.
date > current.txt – redirects all the output of the command: date to be saved in the file: **current.txt** . If the file exist it will be overwritten.
date >> current.txt – this will do the same as above except it will append the output and will not overwrite the file if it exist.
echo "Hello there" > f1 – redirects the output of the command: echo to the file: **f1** it will overwrite f1 if exists !

ls -l >> f2 - appends the ouput of the command: ls to the file **f2**.

mail test@mail.com < f3 - redirects the file: **f3** as a message body for the mail command.

find . -name hello > f1 2> /dev/null - redirects the output of the command find to file: **f1**. All the errors of this command will be sent to the trash.

find . -name hello > f1 2>&1 - redirects all the output of the command find to file: **f1**. All the errors of this command will be also sent to the same place where the output goes.

ps -ef |grep ^Xvnc | grep -v root – lists all processes starting with the word Xvnc but that not contain the word root.

Handling Archives

zip -r d1.zip /home/haim/d1 - creates the archive file: **d1.zip** of the directory: **d1** with all of it contents.
unzip -l d1.zip - this will only show the content of the archive file: **d1.zip** but will not extract the archive.
unzip d1.zip – extracts the archive of the file: **d1.zip** to the current directroy.

Using tar

tar is a command for creating an archive of directories with out compression.
tar cvf d1.tar \ /home/haim/d1/* - creates and archive file: **d1.tar** of the directory **d1**, with all of it contents.
tar tvf mydir.tar – lists the content of the archive: **d1.tar** without extracting it.
tar xvf mydir.tar – extracts the archive: **d1.tar** .

Using gzip/gunzip

gzip and gunzip are commands for compressing files.
gzip d1.tar – compresses the file: **d1.tar** and will create the file: **d1.tar.gz** .
gunzip d1.tar.gz – uncompresses the file: **d1.tar.gz** .
*you can also use a combined command by typing:
gtar xzvf d1.tar.gz- uncompresses and extract the file **d1.tar.gz** .

SED and AWK

sed 1,5d f1 – displays the file: **f1** without lines **1-5**
sed -n 5,10p f1 – displays only lines **5-10** of the file: **f1**.
sed s/Install/Uninstall/g readme.txt – replaces the word **Install** with the word Uninstall in the file: readme.txt , the output will be generated to the standard output.
sed s?/home?/soft?g f1 – replaces the strings: /home to be /soft in the file: **f1** .(note that now **?** - is the delimiter between the strings.)
ls -l |awk '{print $5,$9}' – displays the **5th field** and **9th field** of the output of **ls -l** command.
awk '{print NF ":" $0 }' – displays a number representing the number of field and the whole line.
awk 'BEGIN {FS=":"} {print "Field1:" , $1 , "Field3:" $3}' f1 – displays the **1st** and the **3rd** fileds with the respective lables.
ls -l | sed 1d | |awk '{print $5,$9}' – displays the **5th field** and **9th field** of the output of **ls -l** command (after deleting the fist line which is not in the needed format).