

Linux and Bash Command Cheat Sheet: The Basics

Getting information

return your user name

whoami

return your user and group id

id

return operating system name, username, and other info

uname -a

display reference manual for a command

man top

get help on a command

curl --help

return the current date and time

date

Monitoring performance and status

list selection of or all running processes and their PIDs

ps

ps -e

display resource usage

top

list mounted file systems and usage

df

Working with files

copy a file

```
cp file.txt new_path/new_name.txt
```

change file name or path

```
mv this_file.txt that_path/that_file.txt
```

remove a file verbosely

```
rm this_old_file.txt -v
```

create an empty file, or update existing file's timestamp

```
touch a_new_file.txt
```

change/modify file permissions to 'execute' for all users

```
chmod +x my_script.sh
```

get count of lines, words, or characters in file

```
wc -l table_of_data.csv
```

```
wc -w my_essay.txt
```

```
wc -m some_document.txt
```

return lines matching a pattern from files matching a filename pattern - case insensitive and whole words only

```
grep -iw hello \*.txt
```

return file names with lines matching the pattern 'hello' from files matching a filename pattern

```
grep -l hello \*.txt
```

Navigating and working with directories

list files and directories by date, newest last

```
ls -lrt
```

find files in directory tree with suffix 'sh'

```
find -name '*.sh'
```

return present working directory

```
pwd
```

make a new directory

```
mkdir new_folder
```

change the current directory: up one level, home, or some other path

```
cd ../  
cd ~ or cd  
cd another_directory  
  
# remove directory, verbosely  
rmdir temp_directory -v
```

Printing file and string contents

```
# print file contents  
cat my_shell_script.sh  
  
# print file contents page-by-page  
more ReadMe.txt  
  
# print first N lines of file  
head -10 data_table.csv  
  
# print last N lines of file  
tail -10 data_table.csv  
  
# print string or variable value  
echo "I am not a robot"  
echo "I am $USERNAME"
```

Compression and archiving

```
# archive a set of files  
tar -cvf my_archive.tar.gz file1 file2 file3  
  
# compress a set of files  
zip my_zipped_files.zip file1 file2  
zip my_zipped_folders.zip directory1 directory2  
  
# extract files from a compressed zip archive  
unzip my_zipped_file.zip  
unzip my_zipped_file.zip -d extract_to_this_directory
```

Performing network operations

print hostname

hostname

send packets to URL and print response

ping www.google.com

display or configure system network interfaces

ifconfig

ip

display contents of file at a URL

curl <url>

download file from a URL

wget <url>

Bash shebang

#!/bin/bash

Pipes and Filters

chain filter commands using the pipe operator

ls | sort -r

pipe the output of manual page for ls to head to display the first 20 lines

man ls | head -20

Shell and Environment Variables

list all shell variables

set

define a shell variable called my_planet and assign value Earth to it

my_planet=Earth

display shell variable

echo \$my_planet

list all environment variables

```
env
```

environment vars: define/extend variable scope to child processes

```
export my_planet
```

```
export my_galaxy='Milky Way'
```

Metacharacters

comments

```
# The shell will not respond to this message
```

command separator

```
echo 'here are some files and folders'; ls
```

file name expansion wildcard

```
ls *.json
```

single character wildcard

```
ls file_2021-06-???.json
```

Quoting

single quotes - interpret literally

```
echo 'My home directory can be accessed by entering: echo $HOME'
```

double quotes - interpret literally, but evaluate metacharacters

```
echo "My home directory is $HOME"
```

backslash - escape metacharacter interpretation

```
echo "This dollar sign should render: \$"
```

I/O Redirection

redirect output to file

```
echo 'Write this text to file x' > x
```

append output to file

```
echo 'Add this line to file x' >> x

# redirect standard error to file
bad_command_1 2> error.log

# append standard error to file
bad_command_2 2>> error.log

# redirect file contents to standard input
$ tr "[a-z]" "[A-Z]" < a_text_file.txt

# the input redirection above is equivalent to
$cat a_text_file.txt | tr "[a-z]" "[A-Z]"
```

Command Substitution

```
# capture output of a command and echo its value
THE_PRESENT=$(date)
echo "There is no time like $THE_PRESENT"
```

Command line arguments

```
./My_Bash_Script.sh arg1 arg2 arg3
```

Batch vs. concurrent modes

```
# run commands sequentially
start=$(date); ./MyBigScript.sh ; end=$(date)

# run commands in parallel
./ETL_chunk_one_on_these_nodes.sh & ./ETL_chunk_two_on_those_nodes.sh
```

Scheduling jobs with Cron

```
# open crontab editor
crontab -e
```

job scheduling syntax

m h dom mon dow command

minute, hour, day of month, month, day of week

* means any

append the date/time to file every Sunday at 6:15 pm

15 18 * * 0 date >> sundays.txt

run a shell script on the first minute of the first day of each month

1 0 1 * * ./My_Shell_Script.sh

back up your home directory every Monday at 3 am

0 3 * * 1 tar -cvf my_backup_path\my_archive.tar.gz \$HOME

deploy your cron job

Close the crontab editor and save the file

list all cron jobs

crontab -l

© Copyright IBM Corporation 2021. All rights reserved.