# Module 3 Cheat Sheet - Introduction to Shell Scripting

Module 3 Cheat Sheet - Introduction to Shen Scripting		
Bash shebang		
#!/bin/bash		
Get the path to a command		
which bash		
Pipes, filters, and chaining		
Chain filter commands together using the pipe operator:		
ls   sort -r		
Pipe the output of manual page for 1s to head to display the first 20 lines:		
man ls   head -20		
Use a pipeline to extract a column of names from a csv and drop duplicate names:		
cut -d "," -f1 names.csv   sort   uniq		

List all shell variables:

Working with shell and environment variables:

set

Define a shell variable called my_planet and assign value Earth to it:
my_planet=Earth
Display value of a shell variable:
echo \$my_planet
Reading user input into a shell variable at the command line:
read first_name
Tip: Whatever text string you enter after running this command gets stored as the value of the variable first_name.
List all environment variables:
env
Environment vars: define/extend variable scope to child processes:
export my_planet export my_galaxy='Milky Way'
export my_galaxy='Milky Way'
Metacharacters
Wetacharacters

#### 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'

# $\label{lem:conditional} \textbf{Double quotes} \ "" - \textbf{interpret literally, but evaluate metacharacters:}$

echo "My home directory is \$HOME"

about:blank 3/10

Backslash \ - escape metacharacter interpretation:

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

#### I/O Redirection

Redirect output to file and overwrite any existing content:

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:

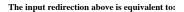
bad\_command\_1 2> error.log

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</pre>



\$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"

## Capture output of a command and echo its value:

echo "There is no time like \$(date)"

# **Command line arguments**

./My\_Bash\_Script.sh arg1 arg2 arg3

#### Batch vs. concurrent modes

## Run commands sequentially:

start=\$(date); ./MyBigScript.sh ; end=\$(date)

#### $\label{lem:commands} \textbf{Run commands in parallel:}$

./ETL\_chunk\_one\_on\_these\_nodes.sh & ./ETL\_chunk\_two\_on\_those\_nodes.sh

about:blank 5/10

8/11/25, 9:37 PM

about:blank

# 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)

Tip: You can use the \* wildcard to mean "any".

Append the date/time to a 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 3:00 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
```

## Conditionals

if-then-else syntax:

```
if [[ $# == 2 ]]
then
    echo "number of arguments is equal to 2"
else
    echo "number of arguments is not equal to 2"
fi
```

'and' operator &&:

```
if [ condition1 ] && [ condition2 ]
```

```
'or' operator ||:
```

```
if [ condition1 ] || [ condition2 ]
```

# **Logical operators**

Operator	Definition
==	is equal to
!=	is not equal to
<	is less than
>	is greater than
<=	is less than or equal to
>=	is greater than or equal to

## **Arithmetic calculations**

Integer arithmetic notation:

about:blank 7/10

## Basic arithmetic operators:

Symbol	Operation
+	addition
-	subtraction
*	multiplication
/	division

Display the result of adding 3 and 2:

echo \$((3+2))

Negate a number:

echo \$((-1\*-2))

# Arrays

Declare an array that contains items 1, 2, "three", "four", and 5:

my\_array=(1 2 "three" "four" 5)

Add an item to your array:

my\_array+="six"
my\_array+=7

```
my_array=($(echo $(cat column.txt)))
```

## for loops

Use a for loop to iterate over values from 1 to 5:

```
for i in {0..5}; do
    echo "this is iteration number $i"
done
```

Use a for loop to print all items in an array:

```
for item in ${my_array[@]}; do
  echo $item
done
```

Use array indexing within a for loop, assuming the array has seven elements:

```
for i in {0..6}; do
    echo ${my_array[$i]}
done
```

# **Authors**

Jeff Grossman Sam Propupchuk

#### **Other Contributors**

Rav Ahuja

about:blank 9/10



about:blank 10/10