ORACLE

Srcipting: BASH

Introduction and Deep Dive

Georg Völl
Cloud Architect
Technical Cloud Engineering Germany

Agenda

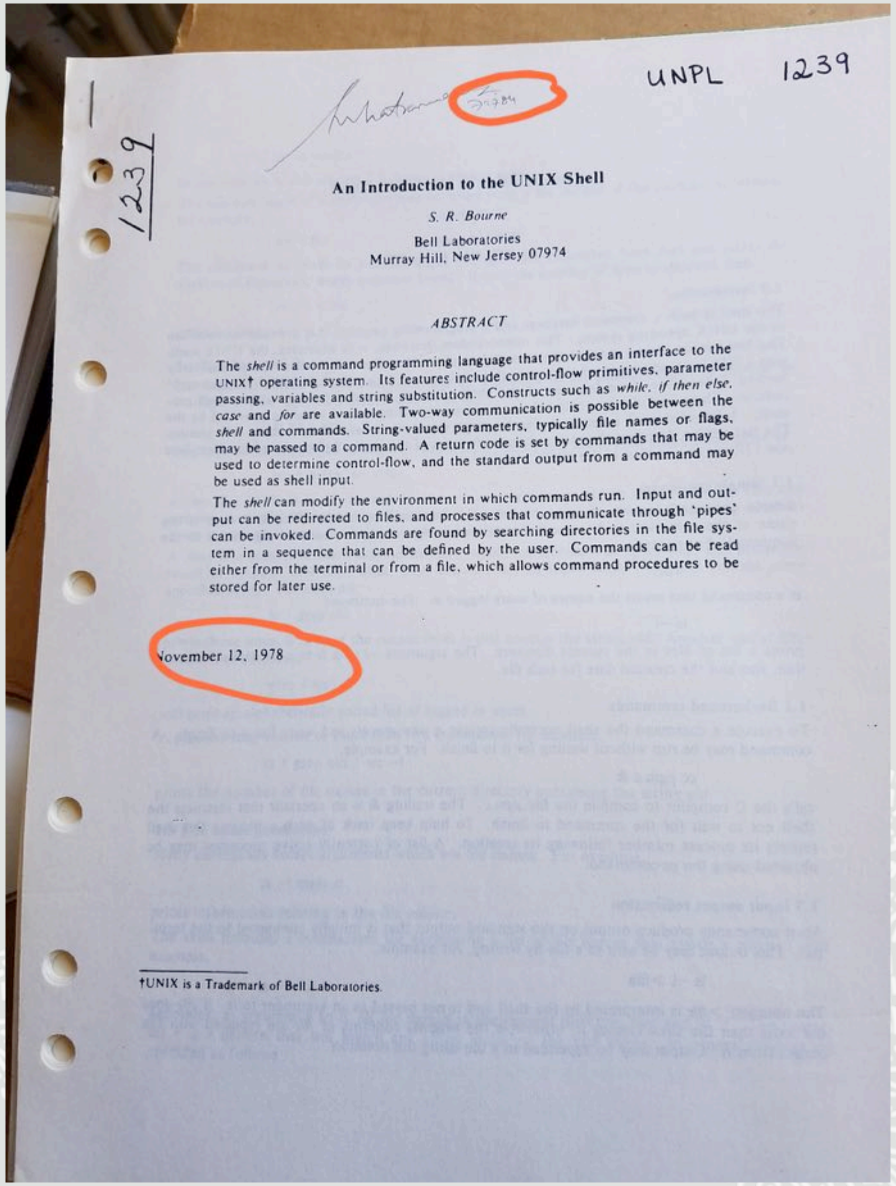
1) What is BASH and when do I use it?
2) Workshop: Generate ssh keys
3) Workshop: Installing Python and Java
4) Workshop: Process each line in file
5) Workshop: Using Admin Scripts Framework
6) Admin Scripts Framework Demo: disk-management

What is a Unix Shell?

https://en.wikipedia.org/wiki/Unix_shell

A Unix shell is a command-line interpreter or shell that provides a command line user interface for Unix-like operating systems. The shell is both an interactive command language and a scripting language, and is used by the operating system to control the execution of the system using shell scripts.

The Bourne shell, sh, was a new Unix shell by Stephen Bourne at Bell Labs. Distributed as the shell for UNIX Version 7 in 1979, it introduced the rest of the basic features considered common to all the Unix shells, including here documents, command substitution, more generic variables and more extensive builtin control structures.



UNPL 1239

1239

[illegible] [illegible]

An Introduction to the UNIX Shell

S. R. Bourne

Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

The *shell* is a command programming language that provides an interface to the UNIX† operating system. Its features include control-flow primitives, parameter passing, variables and string substitution. Constructs such as *while*, *if then else*, *case* and *for* are available. Two-way communication is possible between the *shell* and commands. String-valued parameters, typically file names or flags, may be passed to a command. A return code is set by commands that may be used to determine control-flow, and the standard output from a command may be used as shell input.

The *shell* can modify the environment in which commands run. Input and output can be redirected to files, and processes that communicate through 'pipes' can be invoked. Commands are found by searching directories in the file system in a sequence that can be defined by the user. Commands can be read either from the terminal or from a file, which allows command procedures to be stored for later use.

November 12, 1978

†UNIX is a Trademark of Bell Laboratories.

What is BASH?

https://en.wikipedia.org/wiki/Bash_(Unix_shell)

GNU Bash or simply Bash is a Unix shell and command language written by Brian Fox for the GNU Project as a free software replacement for the Bourne shell. First released in 1989, it has been used as the default login shell for most Linux distributions and all releases of Apple's macOS prior to macOS Catalina. A version is also available for Windows 10 via the Windows Subsystem for Linux. It is also the default user shell in Solaris 11.

Bash is a command processor that typically runs in a text window where the user types commands that cause actions. Bash can also read and execute commands from a file, called a shell script. Like all Unix shells, it supports filename globbing (wildcard matching), piping, here documents, command substitution, variables, and control structures for condition-testing and iteration. The keywords, syntax, dynamically scoped variables and other basic features of the language are all copied from sh. Other features, e.g., history, are copied from csh and ksh. Bash is a POSIX-compliant shell, but with a number of extensions.

The shell's name is an acronym for Bourne Again Shell, a pun on the name of the Bourne shell that it replaces and the notion of being "born again".

When do I use BASH?

BASH	Python / Java / …
Control the execution of the system	Control the execution of an Application
OS oriented Admin Tasks	App oriented Admin Tasks
Needs “helper tools“ like grep, awk, sed, head, tail, find, …	Everything inside the scripting language
Porting and reusability is challenging	Porting and reusability is easier (version specific)
Known by most of Unix Admins	Known by Developers
OCI CLI can be used (slow)	Easy to use OCI Python SDK (fast)

Example: Auto-Install

- Complex setup of different applications
- Repetitive activities
- Activities that build on one another
- Review of the requirements
- Check whether previous activity has been carried out correctly
- Automation and avoidance of unnecessary interactions
- Reproducible and reliable results

Sun ONE
Communication Services
Test & Evaluation Software
13.03.2003
Sun microsystems
Überreicht durch:
Georg.Voell@Sun.com
Sun ONE
Open Net Environment
OS: Solaris
Kein Produktionseinsatz!
Messaging Server 5.2
Calendar Server 5.1
Directory Server 5.1
NameFinder 3.0
Web Server 6.0
DA4C&C 1.2
©2003 Sun Microsystems, Inc.
Sun, Sun Microsystems, iPlanet und alle Warenzeichen und Logos, basierend auf Sun und iPlanet, sind Warenzeichen und eingetragene Warenzeichen von Sun Microsystems Inc. in den USA und anderen Ländern.

bash Cookbook, 2nd Edition

- https://learning.oreilly.com/library/view/bash-cookbook-2nd/9781491975329/

- **Google is your friend!**
  - Example: Look for „bash while“

- All script examples and this PPT can be found here:
  - https://standby.cloud/download/training/bash/

Workshop

1) Generate ssh keys
2) Installing Python and Java
3) Process each line in file
4) Using Admin Scripts Framework

1) Generate ssh keys

Generate all needed keys for Cloud

- Web UI: https://standby.cloud/ssh-keygen/
- Source Code: https://standby.cloud/ssh-keygen/createkeys
- API Usage:
  - -Setting environment variables:

```
username="opc"
passphrase="mysecret"
```

  - -Printing environment variables:

```
echo $username
printf "Username: %s\n" "$username"
```

  - -Get the keys:

```
curl -L -s "https://standby.cloud/ssh-keygen/keygen.pl?\
username=${username}&passphrase=${passphrase}" -o ${username}.zip
```

2) Installing Python and Java

One-Off Install (Oracle Linux on OCI)

- Choose an Editor (syntax highlighting, setting break-points availabilities preferred)
- Create a file „install-tools“ and make it executable (chmod 755)

```
sudo yum –y install python3 python3-libs python3-setuptools python3-pip
sudo yum –y install java-1.8.0-openjdk
```

- Execute „./install-tools“
- We are done.

Tricks with BASH (1)

- Create a first line starting with #! and the interpreter to use (Shebang or Hash-Bang)

```
#!/bin/bash
```

- If you are not shure where your interpreter is installed, you can use „env“

```
#!/bin/env tcsh
```

- Best practice is to set a default PATH to ensure that used tools are found

```
PATH="/usr/local/sbin:/usr/local/bin:/usr/gnu/bin:/sbin:/bin:/usr/sbin:/usr/bin"
```

- We could check if tools are in PATH with „which“

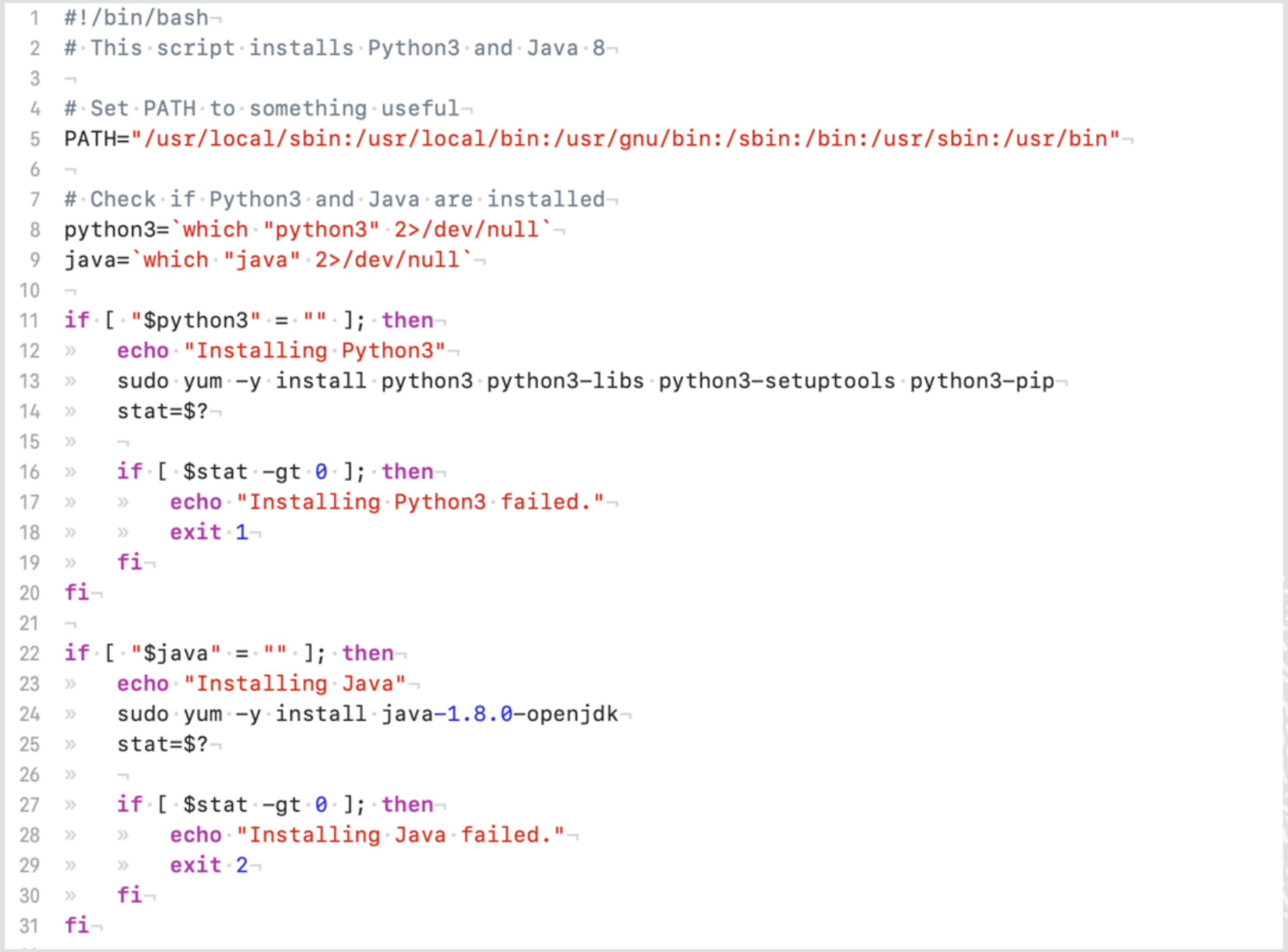
```
yum=`which "yum" 2>/dev/null`
```

- Exitcodes can be used to check if operation was successful

```
stat=$?
```

Check if we need to install tools and if installation was successful

example1



```bash
#!/bin/bash
# This script installs Python3 and Java 8

# Set PATH to something useful
PATH="/usr/local/sbin:/usr/local/bin:/usr/gnu/bin:/sbin:/bin:/usr/sbin:/usr/bin"

# Check if Python3 and Java are installed
python3=`which "python3" 2>/dev/null`
java=`which "java" 2>/dev/null`

if [ "$python3" = "" ]; then
	echo "Installing Python3"
	sudo yum -y install python3 python3-libs python3-setuptools python3-pip
	stat=$?

	if [ $stat -gt 0 ]; then
		echo "Installing Python3 failed."
		exit 1
	fi
fi

if [ "$java" = "" ]; then
	echo "Installing Java"
	sudo yum -y install java-1.8.0-openjdk
	stat=$?

	if [ $stat -gt 0 ]; then
		echo "Installing Java failed."
		exit 2
	fi
fi
```

Download examples

```
curl -s https://standby.cloud/download/training/bash/example1 -o example1
chmod 755 example1
```

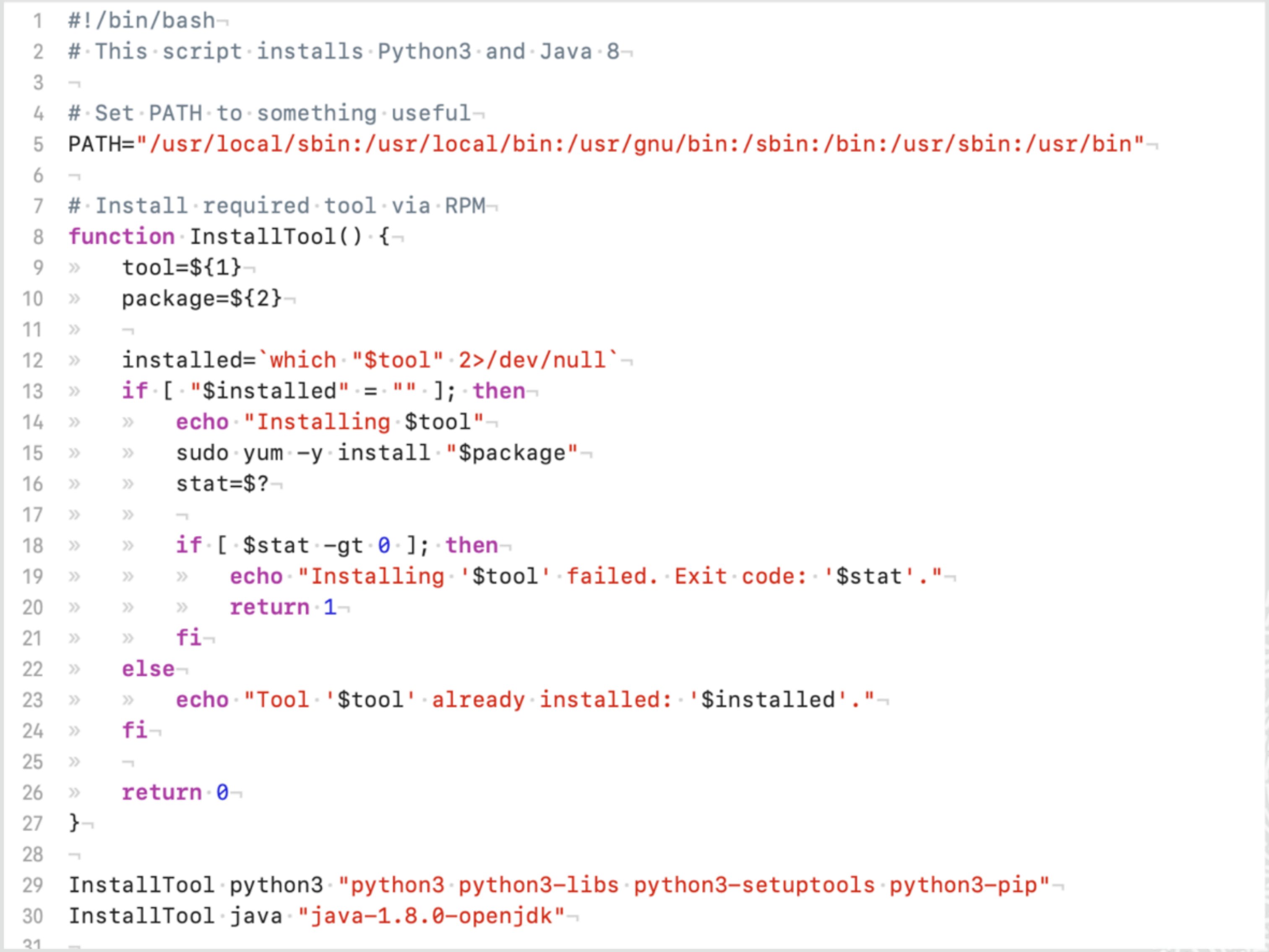
Repeat this 6 times (replacing example1 with exampleX) where X is the number of repetitions.

Alternatively, you can also load the examples via browser:
https://standby.cloud/download/training/bash/

2) Installing Python and Java

Using a function

example2



```bash
#!/bin/bash
# This script installs Python3 and Java 8

# Set PATH to something useful
PATH="/usr/local/sbin:/usr/local/bin:/usr/gnu/bin:/sbin:/bin:/usr/sbin:/usr/bin"

# Install required tool via RPM
function InstallTool() {
	tool=${1}
	package=${2}

	installed=`which "$tool" 2>/dev/null`
	if [ "$installed" = "" ]; then
		echo "Installing $tool"
		sudo yum -y install "$package"
		stat=$?

		if [ $stat -gt 0 ]; then
			echo "Installing '$tool' failed. Exit code: '$stat'."
			return 1
		fi
	else
		echo "Tool '$tool' already installed: '$installed'."
	fi

	return 0
}

InstallTool python3 "python3 python3-libs python3-setuptools python3-pip"
InstallTool java "java-1.8.0-openjdk"

```

Tricks with BASH (2)

- **OS can be checked by using „uname“**

```
os=`uname -s`
```

- **User can be checked by using „whoami“**

```
user=`whoami`
```

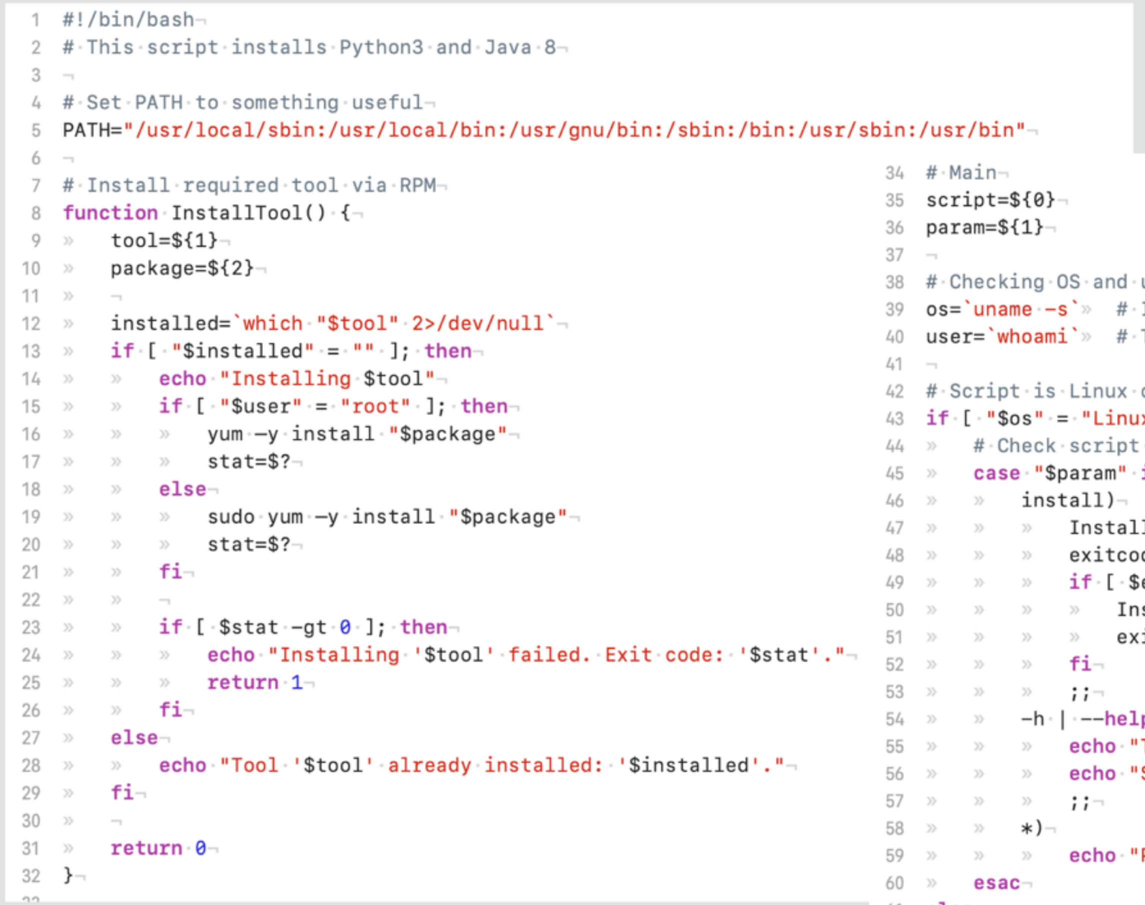
- **Script parameter can be checked by using ${0}.. ${x}**

```
script=${0}
param=${1}
```

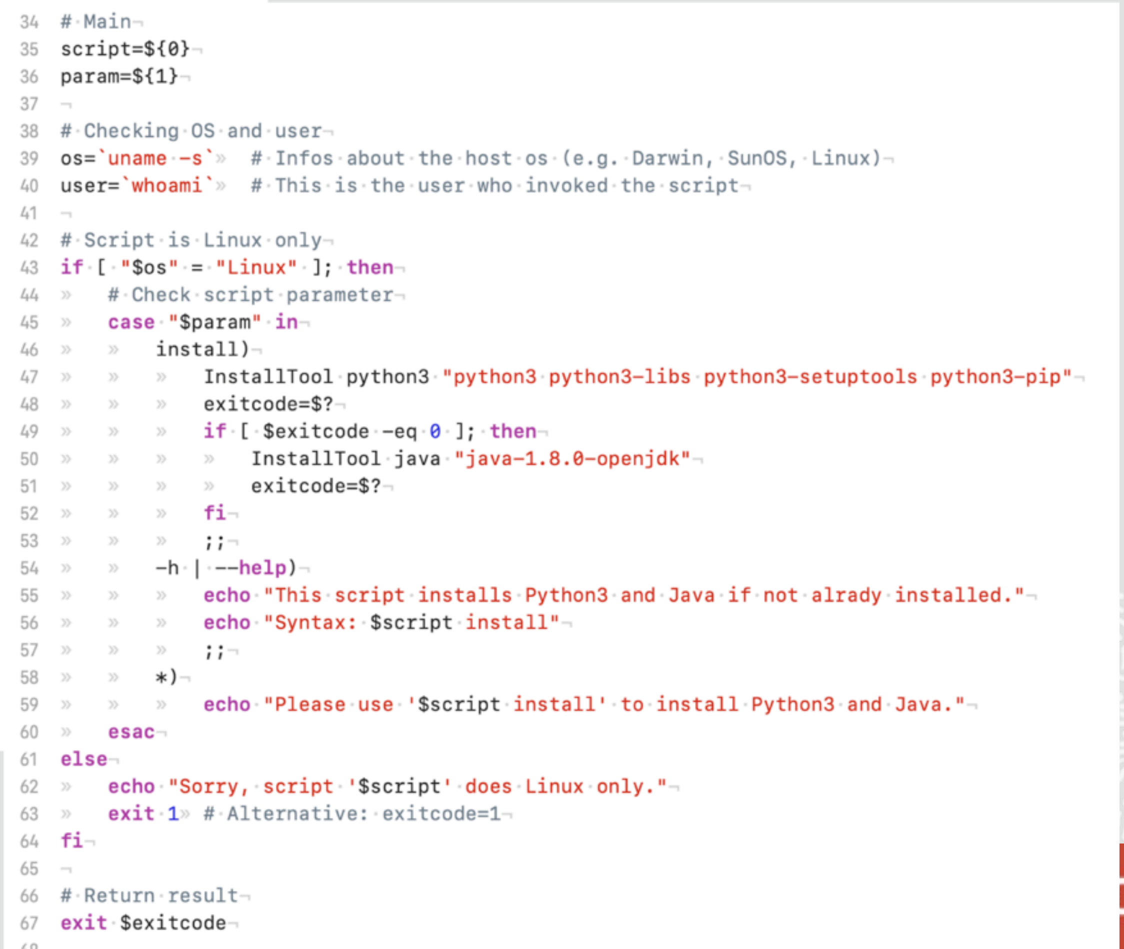
2) Installing Python and Java

More checks

example3



```bash
#!/bin/bash
# This script installs Python3 and Java 8

# Set PATH to something useful
PATH="/usr/local/sbin:/usr/local/bin:/usr/gnu/bin:/sbin:/bin:/usr/sbin:/usr/bin"

# Install required tool via RPM
function InstallTool() {
	tool=${1}
	package=${2}
	
	installed=`which "$tool" 2>/dev/null`
	if [ "$installed" = "" ]; then
		echo "Installing $tool"
		if [ "$user" = "root" ]; then
			yum -y install "$package"
			stat=$?
		else
			sudo yum -y install "$package"
			stat=$?
		fi
		
		if [ $stat -gt 0 ]; then
			echo "Installing '$tool' failed. Exit code: '$stat'."
			return 1
		fi
	else
		echo "Tool '$tool' already installed: '$installed'."
	fi
	
	return 0
}
```



```bash
# Main
script=${0}
param=${1}

# Checking OS and user
os=`uname -s`	# Infos about the host os (e.g. Darwin, SunOS, Linux)
user=`whoami`	# This is the user who invoked the script

# Script is Linux only
if [ "$os" = "Linux" ]; then
	# Check script parameter
	case "$param" in
		install)
			InstallTool python3 "python3 python3-libs python3-setuptools python3-pip"
			exitcode=$?
			if [ $exitcode -eq 0 ]; then
				InstallTool java "java-1.8.0-openjdk"
				exitcode=$?
			fi
			;;
		-h | --help)
			echo "This script installs Python3 and Java if not alrady installed."
			echo "Syntax: $script install"
			;;
		*)
			echo "Please use '$script install' to install Python3 and Java."
	esac
else
	echo "Sorry, script '$script' does Linux only."
	exit 1	# Alternative: exitcode=1
fi

# Return result
exit $exitcode
```

3) Process each line in file

List all users in /etc/passwd

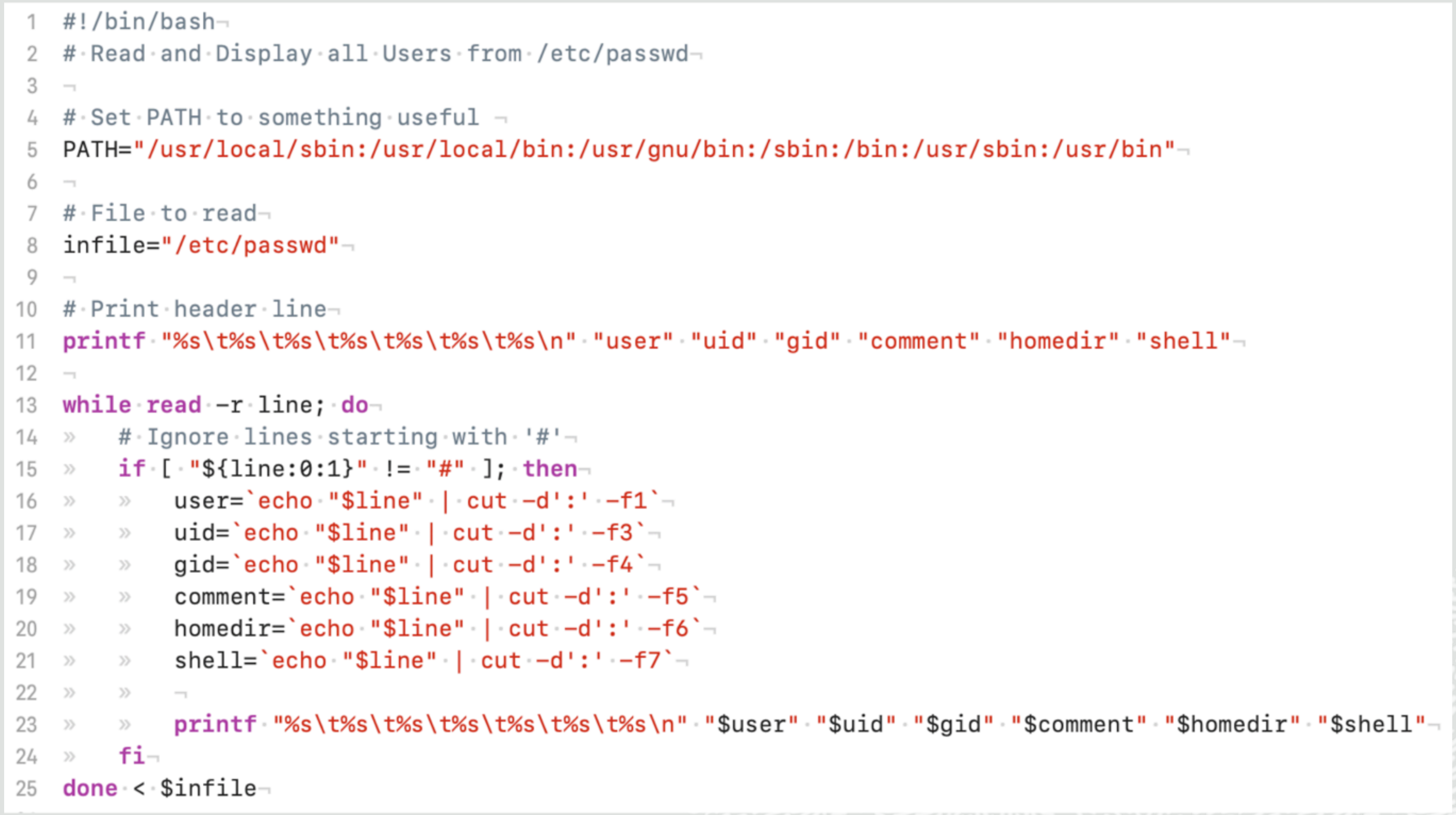
- /etc/password contains all users in system
- Items are seperated by colon e.g.

```
jsmith:x:1001:1000:Joe Smith,Room 1007,(234)555-8910,(234)555-0044,email:/home/jsmith:/bin/sh
```

3) Process each line in file

List all users in /etc/passwd

example4



```bash
#!/bin/bash
# Read and Display all Users from /etc/passwd

# Set PATH to something useful
PATH="/usr/local/sbin:/usr/local/bin:/usr/gnu/bin:/sbin:/bin:/usr/sbin:/usr/bin"

# File to read
infile="/etc/passwd"

# Print header line
printf "%s\t%s\t%s\t%s\t%s\t%s\t%s\n" "user" "uid" "gid" "comment" "homedir" "shell"

while read -r line; do
	# Ignore lines starting with '#'
	if [ "${line:0:1}" != "#" ]; then
		user=`echo "$line" | cut -d':' -f1`
		uid=`echo "$line" | cut -d':' -f3`
		gid=`echo "$line" | cut -d':' -f4`
		comment=`echo "$line" | cut -d':' -f5`
		homedir=`echo "$line" | cut -d':' -f6`
		shell=`echo "$line" | cut -d':' -f7`

		printf "%s\t%s\t%s\t%s\t%s\t%s\t%s\n" "$user" "$uid" "$gid" "$comment" "$homedir" "$shell"
	fi
done < $infile
```

3) Process each line in file

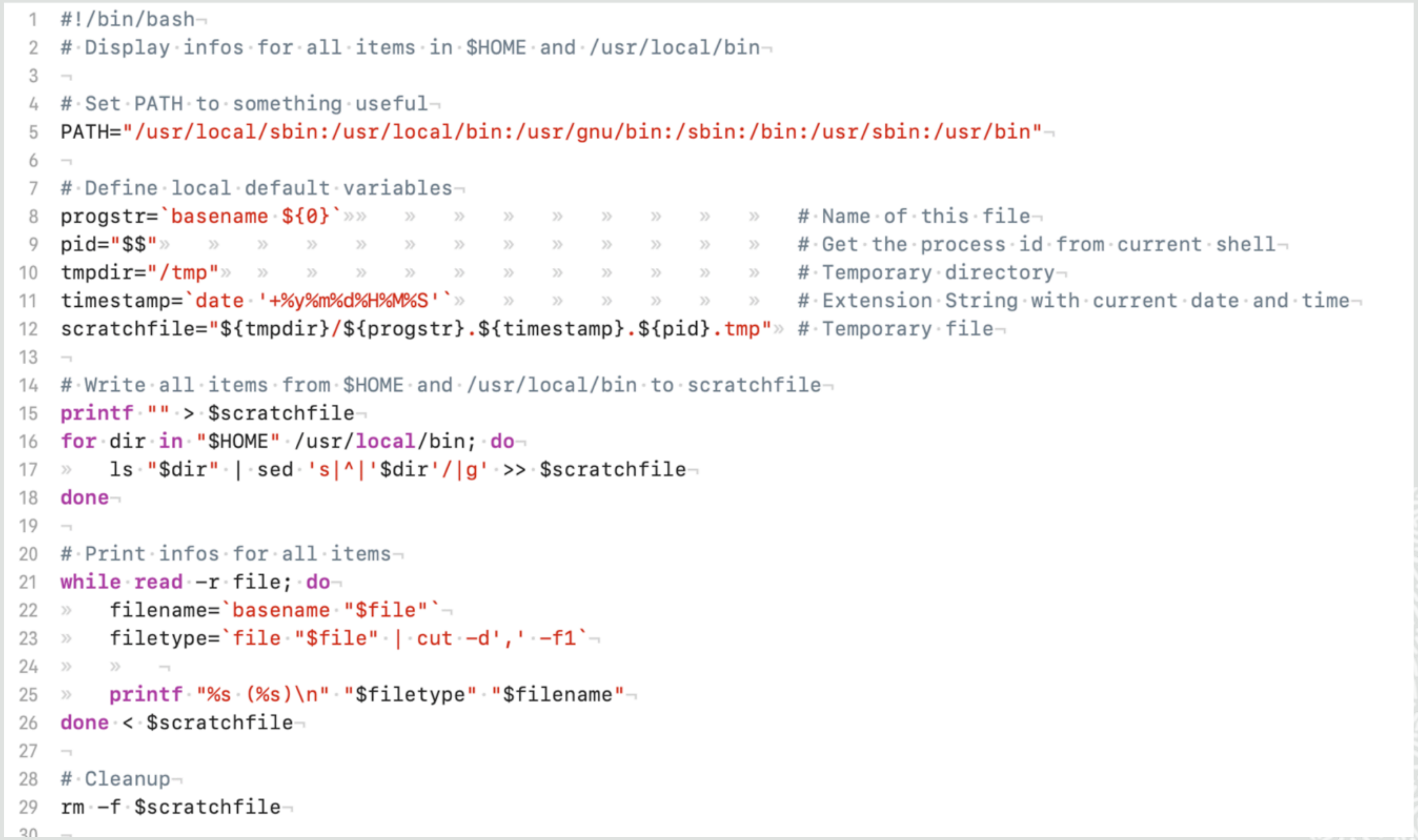
Display infos for all items in $HOME and /usr/local/bin

- List all items in User Home Dirctory and /usr/local/bin and write them to a temporary file
- Check filetypes for all items in temporary file

3) Process each line in file

Display infos for all items in $HOME and /usr/local/bin

example5



```bash
#!/bin/bash
# Display infos for all items in $HOME and /usr/local/bin

# Set PATH to something useful
PATH="/usr/local/sbin:/usr/local/bin:/usr/gnu/bin:/sbin:/bin:/usr/sbin:/usr/bin"

# Define local default variables
progstr=`basename ${0}`                                    # Name of this file
pid="$$"                                                   # Get the process id from current shell
tmpdir="/tmp"                                              # Temporary directory
timestamp=`date '+%y%m%d%H%M%S'`                           # Extension String with current date and time
scratchfile="${tmpdir}/${progstr}.${timestamp}.${pid}.tmp" # Temporary file

# Write all items from $HOME and /usr/local/bin to scratchfile
printf "" > $scratchfile
for dir in "$HOME" /usr/local/bin; do
	ls "$dir" | sed 's|^|'$dir'/|g' >> $scratchfile
done

# Print infos for all items
while read -r file; do
	filename=`basename "$file"`
	filetype=`file "$file" | cut -d',' -f1`

	printf "%s (%s)\n" "$filetype" "$filename"
done < $scratchfile

# Cleanup
rm -f $scratchfile
```

What is „Admin Scripts Framework“?

- Set of helpful scripts which can be used and modified without any license costs
- Licensed under the Universal Permissive License v 1.0 as shown at https://oss.oracle.com/licenses/upl/
- Maintained and updated (limited support)
- Mainly based on BASH (other scripting languages may also been used)
- Integrated with Cloud Tools like OCI CLI, PSM, OPC, RCLONE and others
- Full source code available
- All scripts and tools can be downloaded or pre-checked here:
    - https://standby.cloud/download/

4) Using Admin Scripts Framework

Installing Admin Scripts Framework

- **If „sudo“ is available (Scripts are installed in /usr/local/bin):**

```
curl --silent https://standby.cloud/download/latest/install-scripts | sudo bash
```

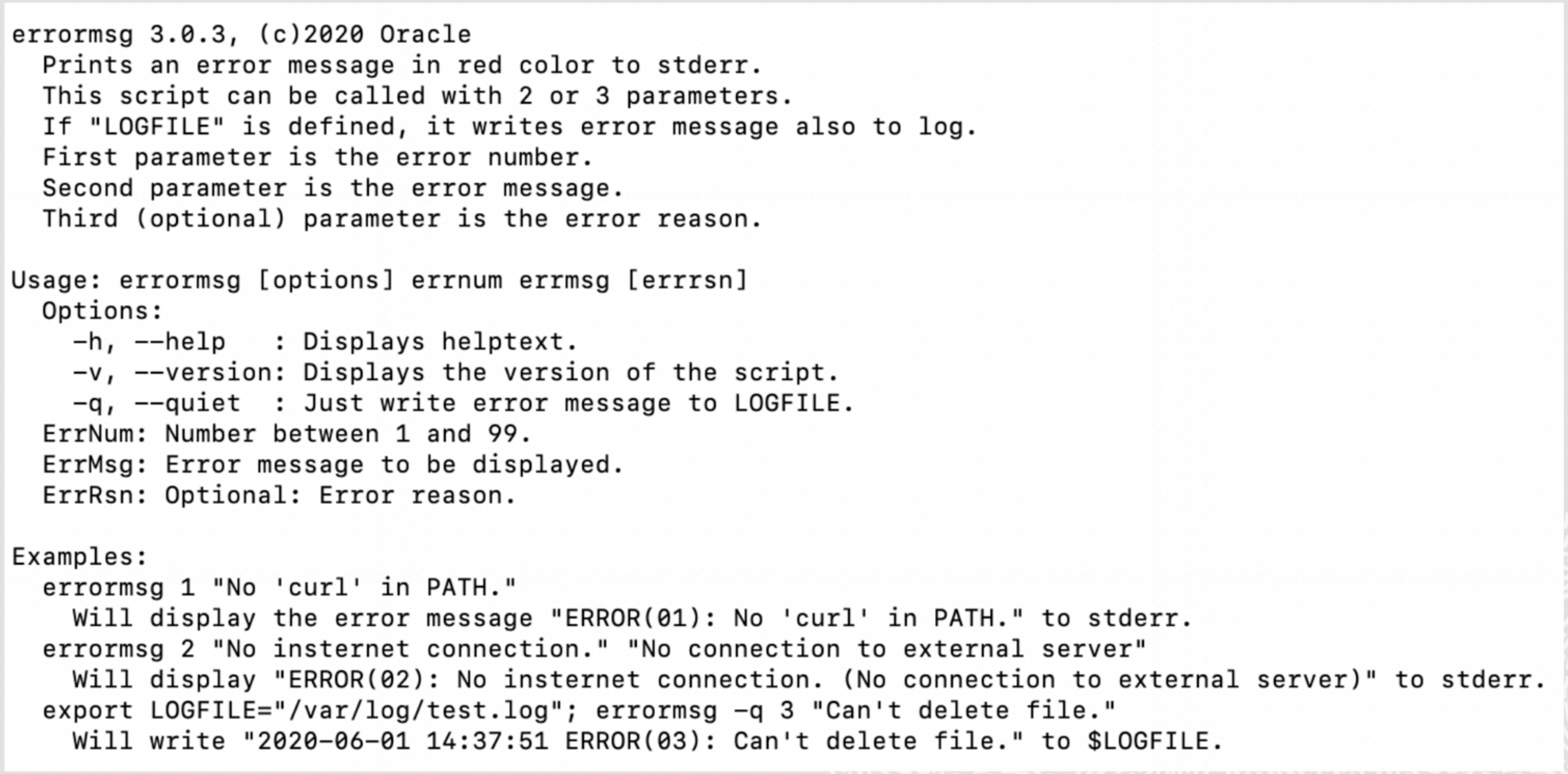
- **In OCI Shell (no „sudo“ – Scripts are installed in $HOME/.local/bin):**

```
curl --silent https://standby.cloud/download/latest/install-scripts | bash
```

- **To update the scripts or to install or update Java / Python3 and other helpful tools and update the OS (optional) after installation of Framework:**

```
setup-tools update
```

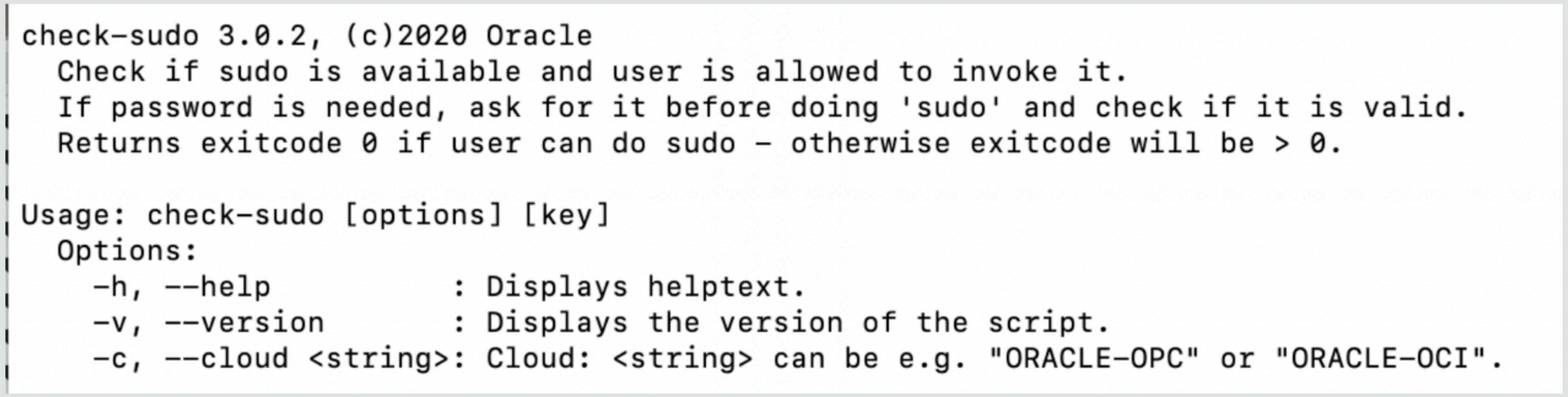
Helper Scripts: errormsg



```
errormsg 3.0.3, (c)2020 Oracle
  Prints an error message in red color to stderr.
  This script can be called with 2 or 3 parameters.
  If "LOGFILE" is defined, it writes error message also to log.
  First parameter is the error number.
  Second parameter is the error message.
  Third (optional) parameter is the error reason.

Usage: errormsg [options] errnum errmsg [errrsn]
  Options:
    -h, --help    : Displays helptext.
    -v, --version: Displays the version of the script.
    -q, --quiet   : Just write error message to LOGFILE.
  ErrNum: Number between 1 and 99.
  ErrMsg: Error message to be displayed.
  ErrRsn: Optional: Error reason.

Examples:
  errormsg 1 "No 'curl' in PATH."
    Will display the error message "ERROR(01): No 'curl' in PATH." to stderr.
  errormsg 2 "No insternet connection." "No connection to external server"
    Will display "ERROR(02): No insternet connection. (No connection to external server)" to stderr.
  export LOGFILE="/var/log/test.log"; errormsg -q 3 "Can't delete file."
    Will write "2020-06-01 14:37:51 ERROR(03): Can't delete file." to $LOGFILE.
```

4) Using Admin Scripts Framework

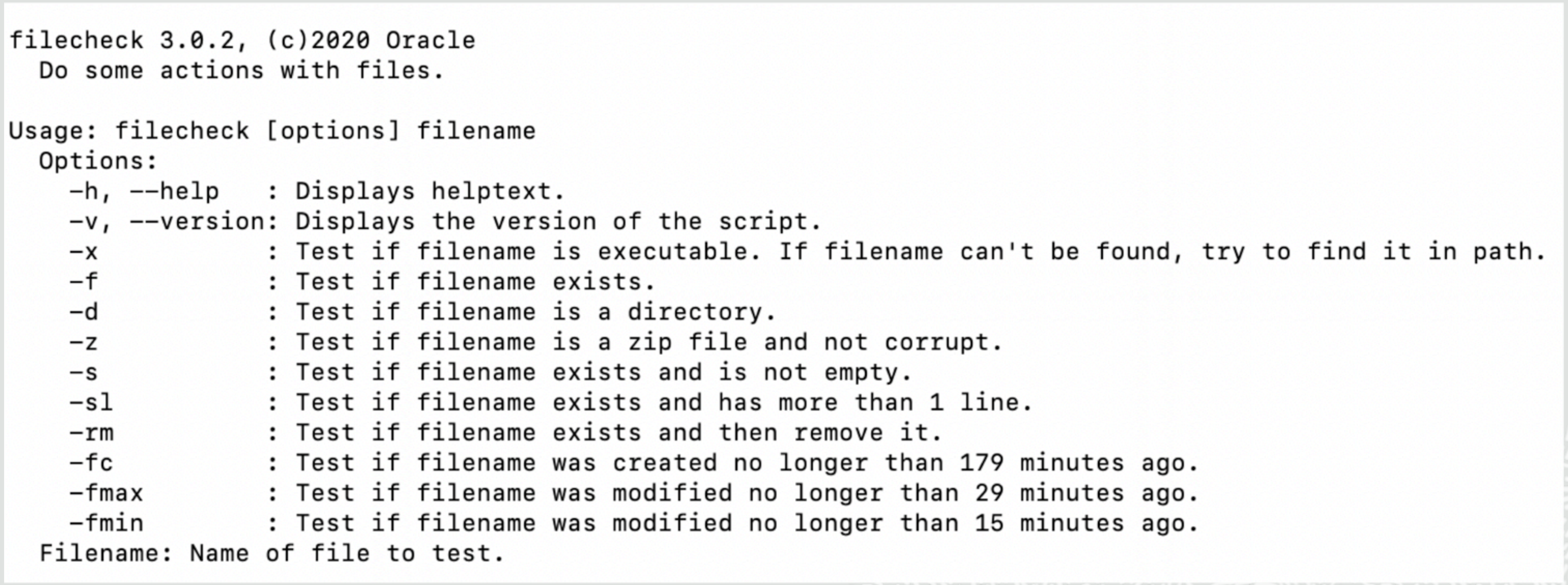
Helper Scripts: check-sudo



```
check-sudo 3.0.2, (c)2020 Oracle
  Check if sudo is available and user is allowed to invoke it.
  If password is needed, ask for it before doing 'sudo' and check if it is valid.
  Returns exitcode 0 if user can do sudo - otherwise exitcode will be > 0.

Usage: check-sudo [options] [key]
  Options:
    -h, --help          : Displays helptext.
    -v, --version       : Displays the version of the script.
    -c, --cloud <string>: Cloud: <string> can be e.g. "ORACLE-OPC" or "ORACLE-OCI".
```

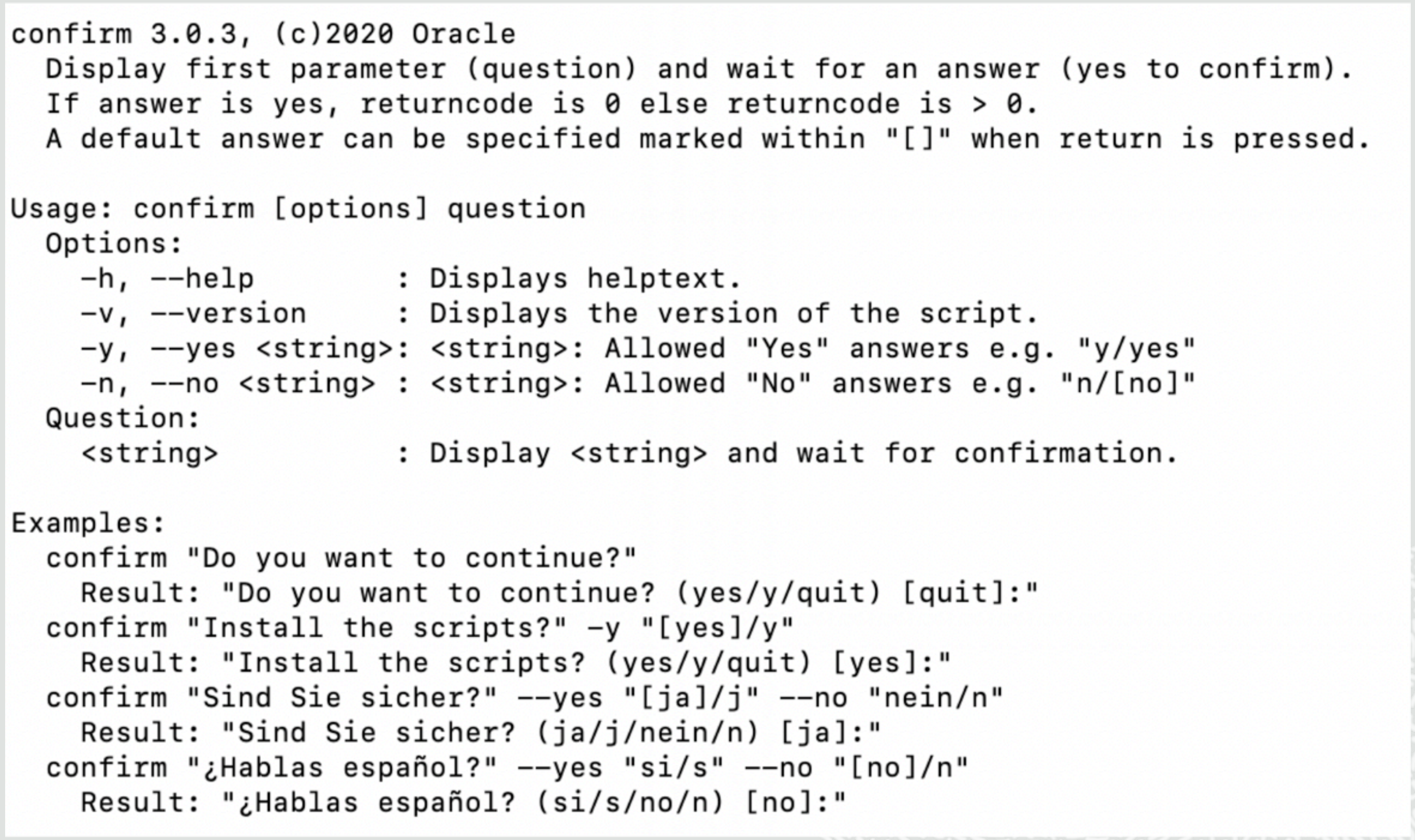
4) Using Admin Scripts Framework

Helper Scripts: filecheck



```
filecheck 3.0.2, (c)2020 Oracle
  Do some actions with files.

Usage: filecheck [options] filename
  Options:
    -h, --help    : Displays helptext.
    -v, --version: Displays the version of the script.
    -x            : Test if filename is executable. If filename can't be found, try to find it in path.
    -f            : Test if filename exists.
    -d            : Test if filename is a directory.
    -z            : Test if filename is a zip file and not corrupt.
    -s            : Test if filename exists and is not empty.
    -sl           : Test if filename exists and has more than 1 line.
    -rm           : Test if filename exists and then remove it.
    -fc           : Test if filename was created no longer than 179 minutes ago.
    -fmax         : Test if filename was modified no longer than 29 minutes ago.
    -fmin         : Test if filename was modified no longer than 15 minutes ago.
  Filename: Name of file to test.
```

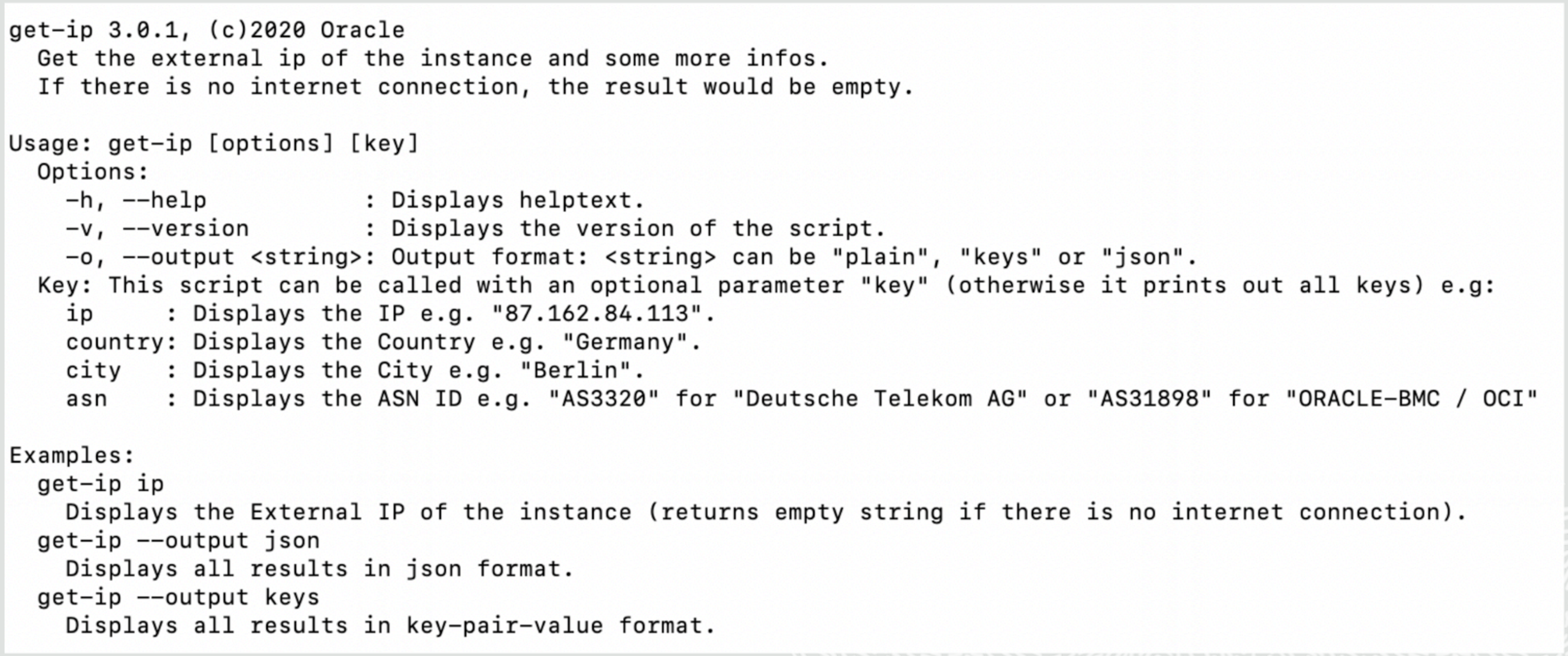
Helper Scripts: confirm



```
confirm 3.0.3, (c)2020 Oracle
  Display first parameter (question) and wait for an answer (yes to confirm).
  If answer is yes, returncode is 0 else returncode is > 0.
  A default answer can be specified marked within "[]" when return is pressed.

Usage: confirm [options] question
  Options:
    -h, --help          : Displays helptext.
    -v, --version       : Displays the version of the script.
    -y, --yes <string>: <string>: Allowed "Yes" answers e.g. "y/yes"
    -n, --no <string> : <string>: Allowed "No" answers e.g. "n/[no]"
  Question:
    <string>            : Display <string> and wait for confirmation.

Examples:
  confirm "Do you want to continue?"
    Result: "Do you want to continue? (yes/y/quit) [quit]:"
  confirm "Install the scripts?" -y "[yes]/y"
    Result: "Install the scripts? (yes/y/quit) [yes]:"
  confirm "Sind Sie sicher?" --yes "[ja]/j" --no "nein/n"
    Result: "Sind Sie sicher? (ja/j/nein/n) [ja]:"
  confirm "¿Hablas español?" --yes "si/s" --no "[no]/n"
    Result: "¿Hablas español? (si/s/no/n) [no]:"
```

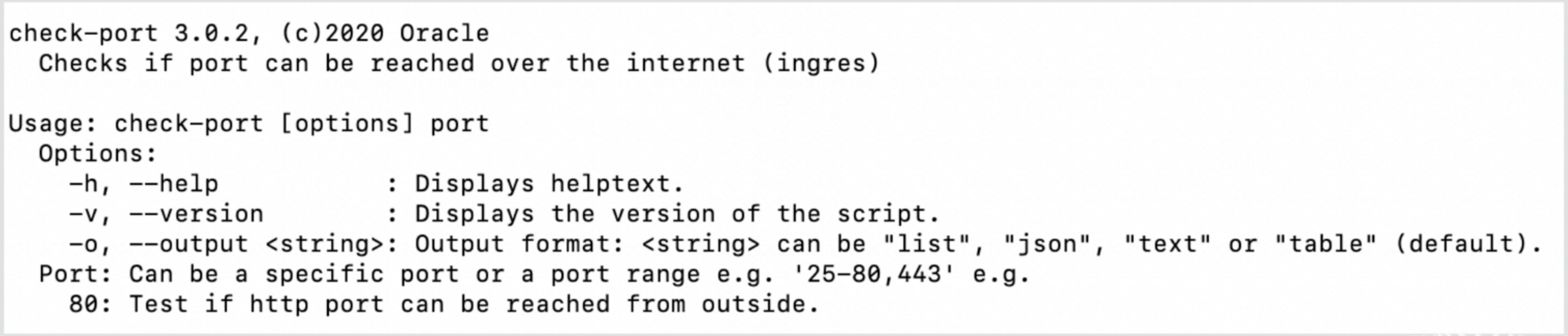
4) Using Admin Scripts Framework

Helper Scripts: get-ip



```
get-ip 3.0.1, (c)2020 Oracle
  Get the external ip of the instance and some more infos.
  If there is no internet connection, the result would be empty.

Usage: get-ip [options] [key]
  Options:
    -h, --help          : Displays helptext.
    -v, --version       : Displays the version of the script.
    -o, --output <string>: Output format: <string> can be "plain", "keys" or "json".
  Key: This script can be called with an optional parameter "key" (otherwise it prints out all keys) e.g:
    ip     : Displays the IP e.g. "87.162.84.113".
    country: Displays the Country e.g. "Germany".
    city   : Displays the City e.g. "Berlin".
    asn    : Displays the ASN ID e.g. "AS3320" for "Deutsche Telekom AG" or "AS31898" for "ORACLE-BMC / OCI"

Examples:
  get-ip ip
    Displays the External IP of the instance (returns empty string if there is no internet connection).
  get-ip --output json
    Displays all results in json format.
  get-ip --output keys
    Displays all results in key-pair-value format.
```

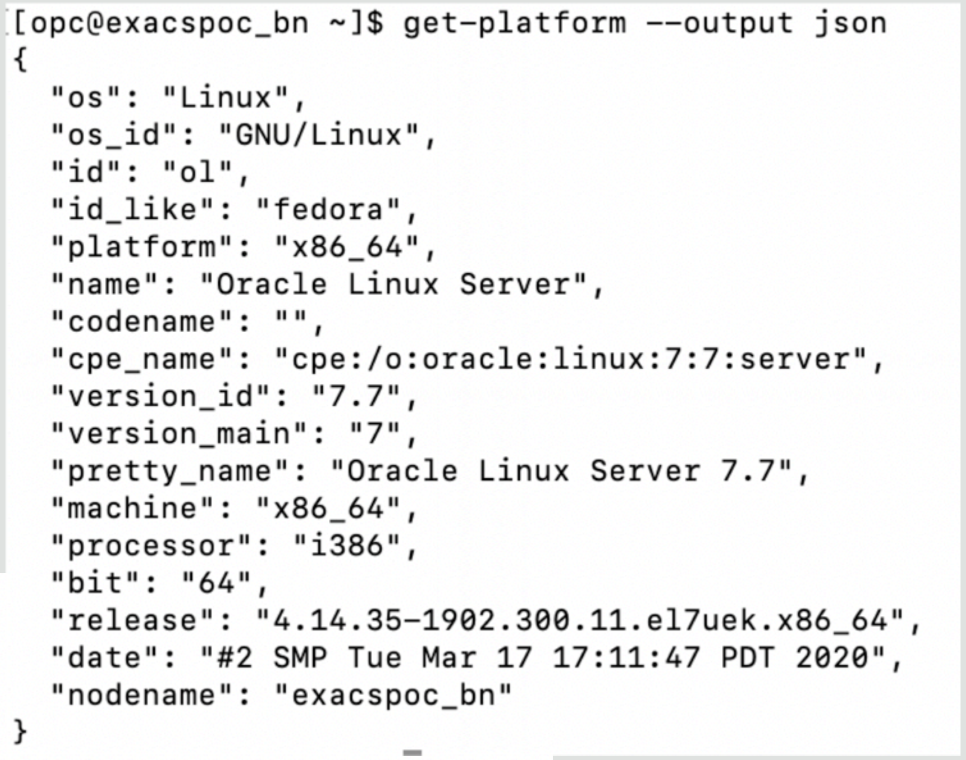
Helper Scripts: check-port



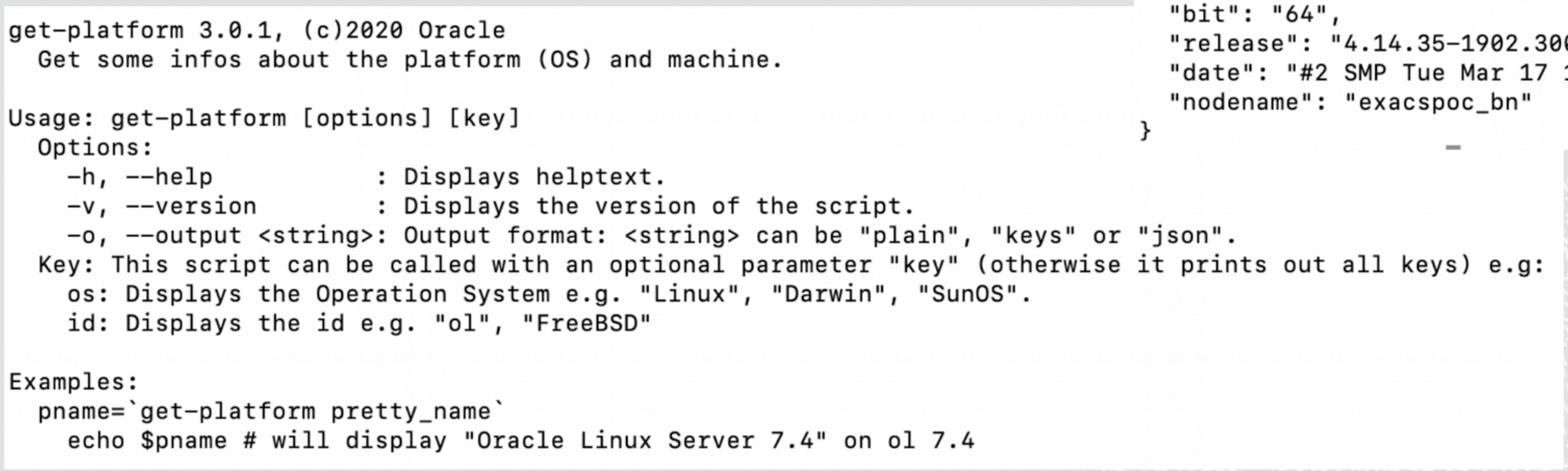
```
check-port 3.0.2, (c)2020 Oracle
  Checks if port can be reached over the internet (ingres)

Usage: check-port [options] port
  Options:
    -h, --help            : Displays helptext.
    -v, --version         : Displays the version of the script.
    -o, --output <string>: Output format: <string> can be "list", "json", "text" or "table" (default).
  Port: Can be a specific port or a port range e.g. '25-80,443' e.g.
    80: Test if http port can be reached from outside.
```

4) Using Admin Scripts Framework

Helper Scripts: get-platform



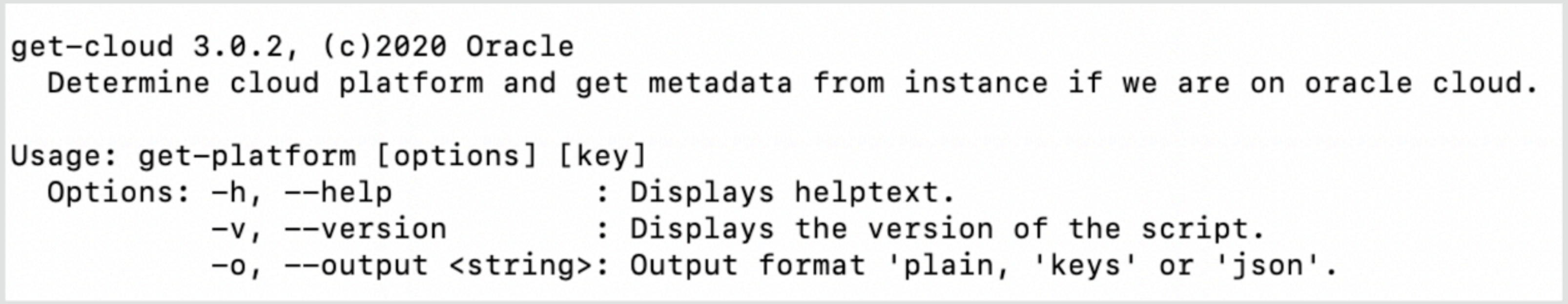
```
[opc@exacspoc_bn ~]$ get-platform --output json
{
  "os": "Linux",
  "os_id": "GNU/Linux",
  "id": "ol",
  "id_like": "fedora",
  "platform": "x86_64",
  "name": "Oracle Linux Server",
  "codename": "",
  "cpe_name": "cpe:/o:oracle:linux:7:7:server",
  "version_id": "7.7",
  "version_main": "7",
  "pretty_name": "Oracle Linux Server 7.7",
  "machine": "x86_64",
  "processor": "i386",
  "bit": "64",
  "release": "4.14.35-1902.300.11.el7uek.x86_64",
  "date": "#2 SMP Tue Mar 17 17:11:47 PDT 2020",
  "nodename": "exacspoc_bn"
}
```



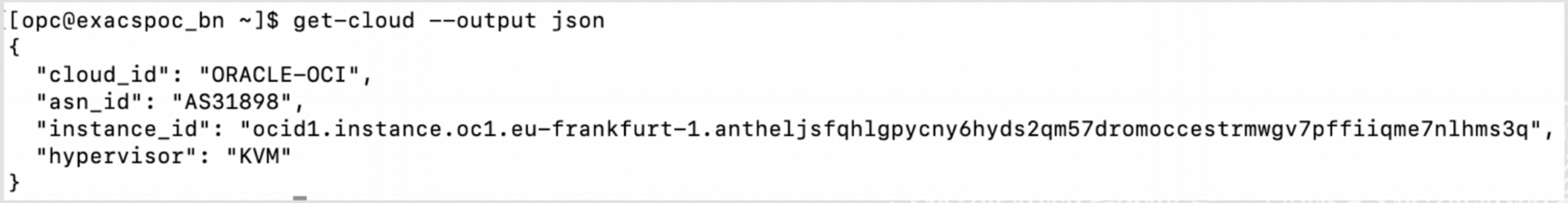
```
get-platform 3.0.1, (c)2020 Oracle
  Get some infos about the platform (OS) and machine.

Usage: get-platform [options] [key]
  Options:
    -h, --help            : Displays helptext.
    -v, --version         : Displays the version of the script.
    -o, --output <string>: Output format: <string> can be "plain", "keys" or "json".
  Key: This script can be called with an optional parameter "key" (otherwise it prints out all keys) e.g:
    os: Displays the Operation System e.g. "Linux", "Darwin", "SunOS".
    id: Displays the id e.g. "ol", "FreeBSD"

Examples:
  pname=`get-platform pretty_name`
    echo $pname # will display "Oracle Linux Server 7.4" on ol 7.4
```

4) Using Admin Scripts Framework

Helper Scripts: get-cloud



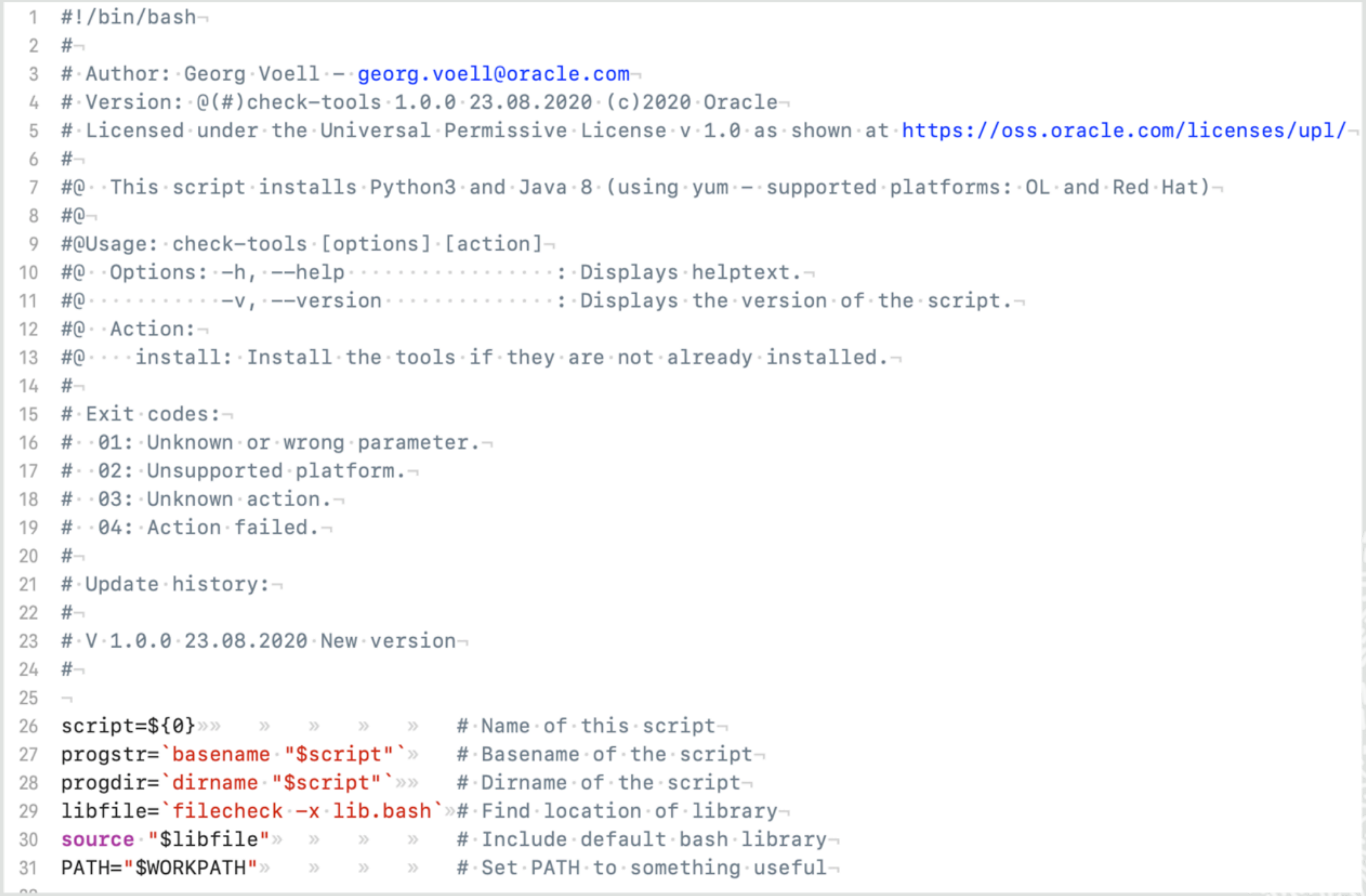
```
get-cloud 3.0.2, (c)2020 Oracle
  Determine cloud platform and get metadata from instance if we are on oracle cloud.

Usage: get-platform [options] [key]
  Options: -h, --help            : Displays helptext.
           -v, --version         : Displays the version of the script.
           -o, --output <string>: Output format 'plain, 'keys' or 'json'.
```



```
[opc@exacspoc_bn ~]$ get-cloud --output json
{
  "cloud_id": "ORACLE-OCI",
  "asn_id": "AS31898",
  "instance_id": "ocid1.instance.oc1.eu-frankfurt-1.antheljsfqhlgpycny6hyds2qm57dromoccestrmwgv7pffiiqme7nlhms3q",
  "hypervisor": "KVM"
}
```

4) Using Admin Scripts Framework

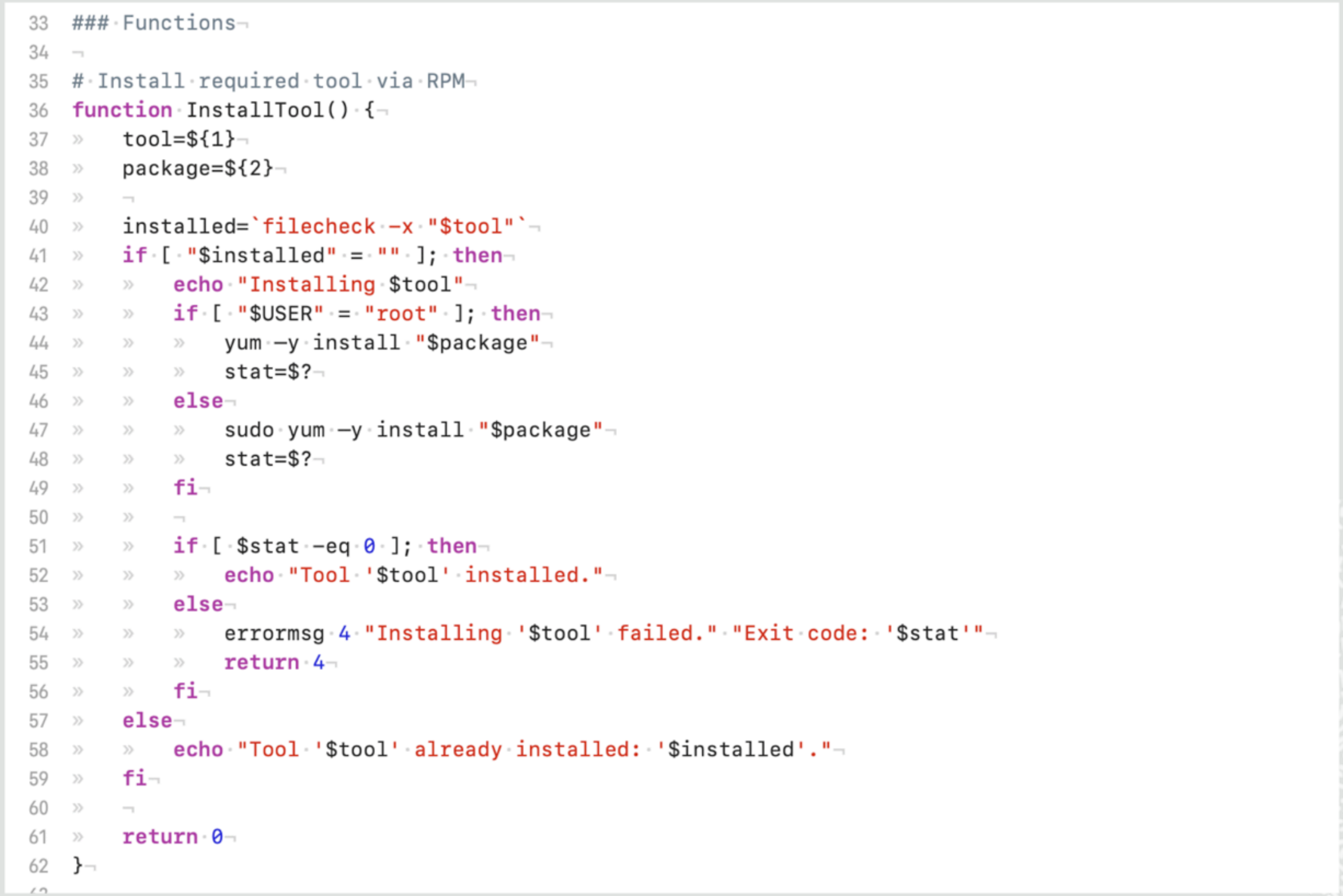
Putting it all together (1)



```
#!/bin/bash
#
# Author: Georg Voell - georg.voell@oracle.com
# Version: @(#)check-tools 1.0.0 23.08.2020 (c)2020 Oracle
# Licensed under the Universal Permissive License v 1.0 as shown at https://oss.oracle.com/licenses/upl/
#
#@  This script installs Python3 and Java 8 (using yum - supported platforms: OL and Red Hat)
#@
#@Usage: check-tools [options] [action]
#@  Options: -h, --help                 : Displays helptext.
#@           -v, --version              : Displays the version of the script.
#@  Action:
#@    install: Install the tools if they are not already installed.
#
# Exit codes:
#  01: Unknown or wrong parameter.
#  02: Unsupported platform.
#  03: Unknown action.
#  04: Action failed.
#
# Update history:
#
# V 1.0.0 23.08.2020 New version
#

script=${0}                             # Name of this script
progstr=`basename "$script"`            # Basename of the script
progdir=`dirname "$script"`             # Dirname of the script
libfile=`filecheck -x lib.bash`         # Find location of library
source "$libfile"                       # Include default bash library
PATH="$WORKPATH"                        # Set PATH to something useful
```

4) Using Admin Scripts Framework

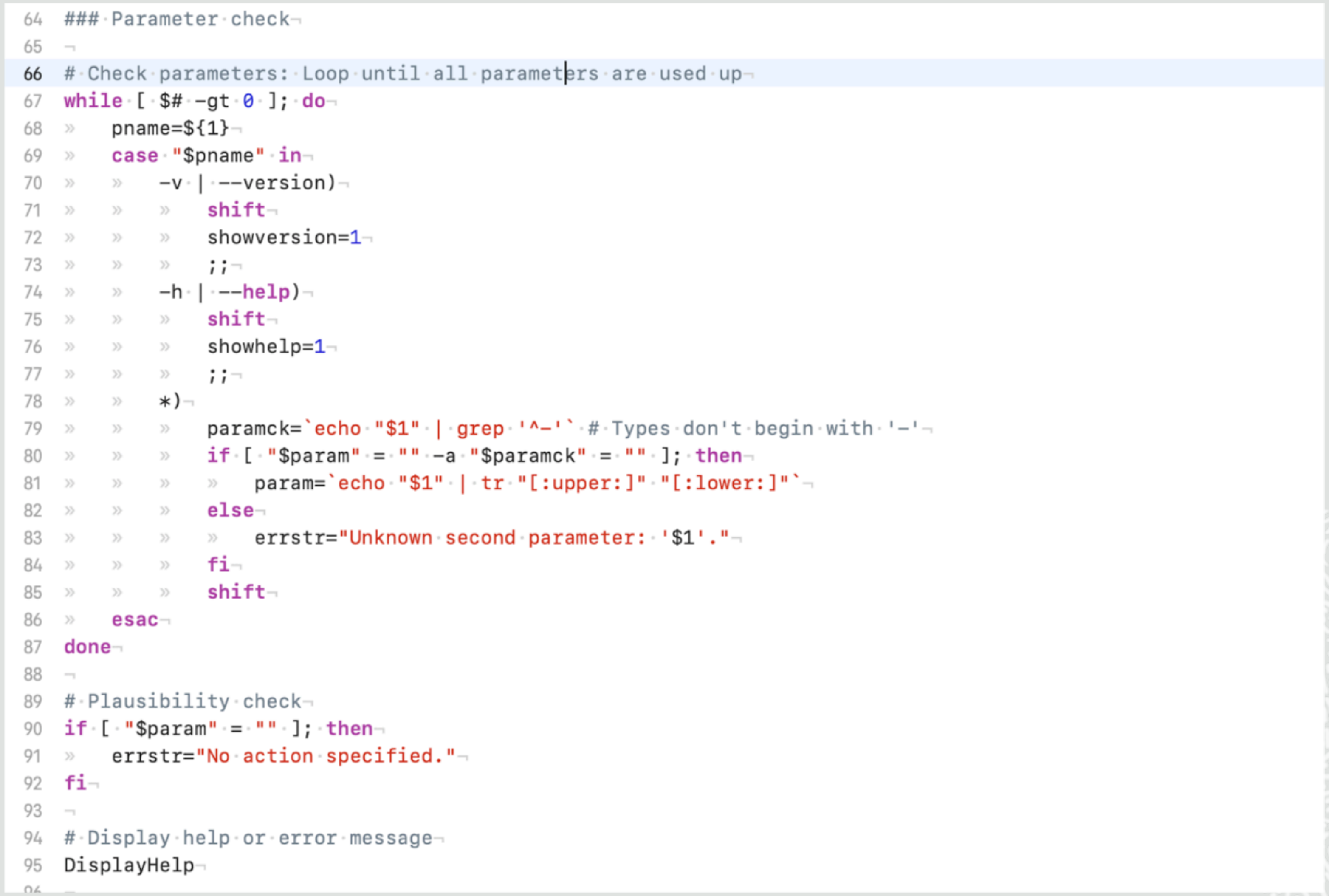
Putting it all together (2)



```
### Functions

# Install required tool via RPM
function InstallTool() {
	tool=${1}
	package=${2}
	
	installed=`filecheck -x "$tool"`
	if [ "$installed" = "" ]; then
		echo "Installing $tool"
		if [ "$USER" = "root" ]; then
			yum -y install "$package"
			stat=$?
		else
			sudo yum -y install "$package"
			stat=$?
		fi
		
		if [ $stat -eq 0 ]; then
			echo "Tool '$tool' installed."
		else
			errormsg 4 "Installing '$tool' failed." "Exit code: '$stat'"
			return 4
		fi
	else
		echo "Tool '$tool' already installed: '$installed'."
	fi
	
	return 0
}
```

4) Using Admin Scripts Framework

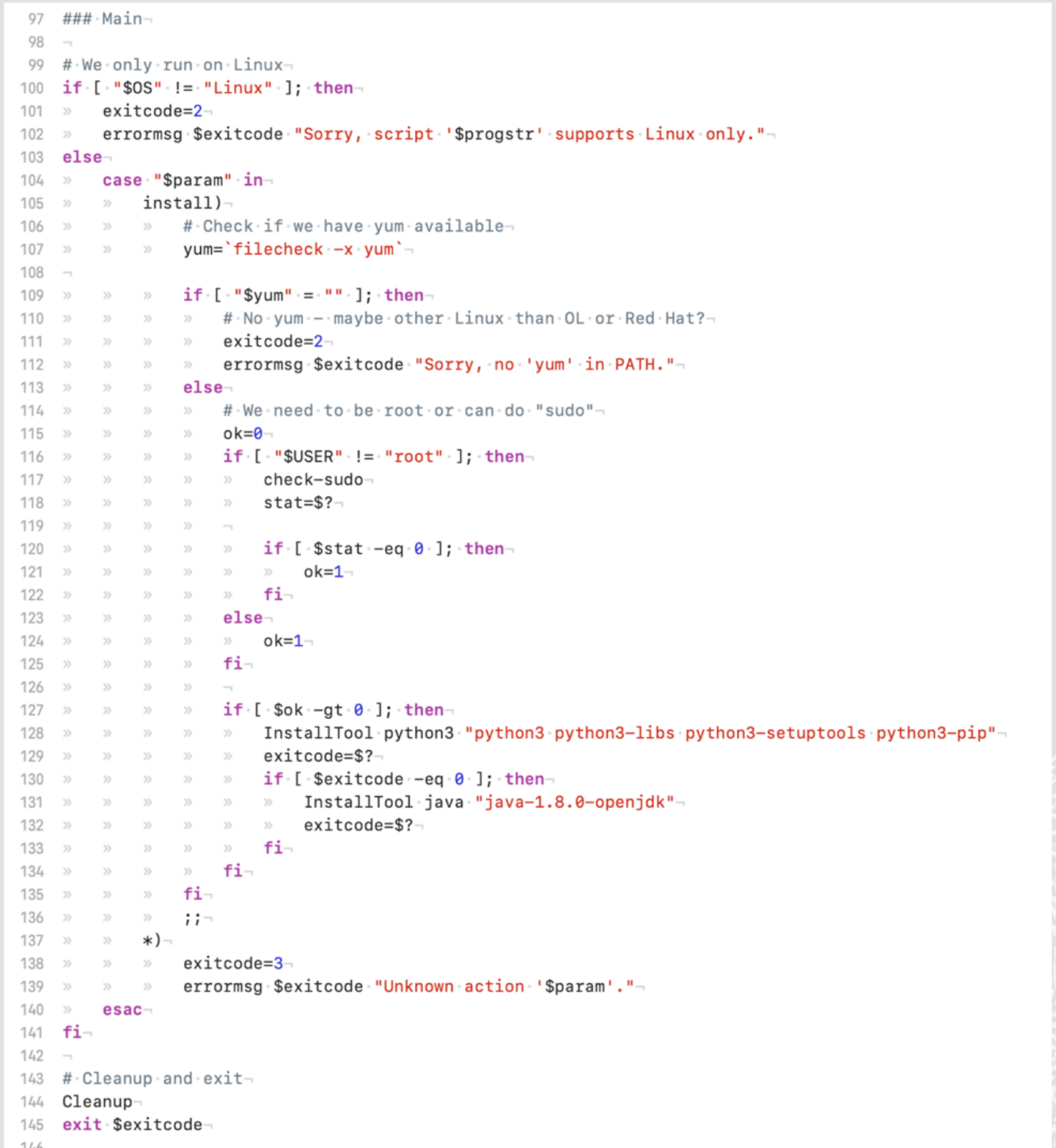
Putting it all together (3)



```
### Parameter check

# Check parameters: Loop until all parameters are used up
while [ $# -gt 0 ]; do
	pname=${1}
	case "$pname" in
		-v | --version)
			shift
			showversion=1
			;;
		-h | --help)
			shift
			showhelp=1
			;;
		*)
			paramck=`echo "$1" | grep '^-'` # Types don't begin with '-'
			if [ "$param" = "" -a "$paramck" = "" ]; then
				param=`echo "$1" | tr "[:upper:]" "[:lower:]"`
			else
				errstr="Unknown second parameter: '$1'."
			fi
			shift
	esac
done

# Plausibility check
if [ "$param" = "" ]; then
	errstr="No action specified."
fi

# Display help or error message
DisplayHelp
```

4) Using Admin Scripts Framework

Putting it all together (4)



```
### Main

# We only run on Linux
if [ "$OS" != "Linux" ]; then
    exitcode=2
    errormsg $exitcode "Sorry, script '$progstr' supports Linux only."
else
    case "$param" in
        install)
            # Check if we have yum available
            yum=`filecheck -x yum`

            if [ "$yum" = "" ]; then
                # No yum - maybe other Linux than OL or Red Hat?
                exitcode=2
                errormsg $exitcode "Sorry, no 'yum' in PATH."
            else
                # We need to be root or can do "sudo"
                ok=0
                if [ "$USER" != "root" ]; then
                    check-sudo
                    stat=$?

                    if [ $stat -eq 0 ]; then
                        ok=1
                    fi
                else
                    ok=1
                fi

                if [ $ok -gt 0 ]; then
                    InstallTool python3 "python3 python3-libs python3-setuptools python3-pip"
                    exitcode=$?
                    if [ $exitcode -eq 0 ]; then
                        InstallTool java "java-1.8.0-openjdk"
                        exitcode=$?
                    fi
                fi
            fi
            ;;
        *)
            exitcode=3
            errormsg $exitcode "Unknown action '$param'."
    esac
fi

# Cleanup and exit
Cleanup
exit $exitcode
```

Admin Script Framework - Demo

1) Script: disk-management

Copy private keys to Cloud Shell (optional)

- **Start the Cloud Shell and copy your Private SSH Key to $HOME/.ssh/id_rsa**

```
vi $HOME/.ssh/id_rsa
```

- **Enter i / Paste private key / Enter :x to save the key**

```
chmod 600 $HOME/.ssh/id_rsa
```

- **Copy your Private API Key to $HOME/.oci/oci_api_key.pem**
- **Doc: https://docs.cloud.oracle.com/en-us/iaas/Content/API/Concepts/apisigningkey.htm**

```
mkdir $HOME/.oci
vi $HOME/.oci/oci_api_key.pem
```

- **Enter i / Paste private key / Enter :x to save the key**

```
chmod 600 $HOME/.oci/oci_api_key.pem
```

4) Using Admin Scripts Framework

Initializing tools

- **Follow these instructions:**

  https://gitlab.com/georg.voell/Multitier/-/blob/master/docs/setup.md

- **Push the configuration to your instance:**

```
setup-tools push
```

- **Use your public ip (of your instance) when asked and select a Hostname (e.g. linux)**

```
ssh linux
```

4) Using Admin Scripts Framework

Using Script „disk-management“

- **Add Blockstorage to your Instance**
- **On your instance type**

```
sudo /usr/local/bin/disk-management
```

- **Create Directory on new disk, change filerights, create symbolic link to new storage e.g.**

```
newstorage=`ls /mnt`
sudo mkdir /mnt/${newstorage}/data
sudo chown opc:opc /mnt/${newstorage}/data
sudo ln -s /mnt/${newstorage}/data /data
```

- **Resize Blockstorage and try again**