2 - Operating Systems & Linux Basics

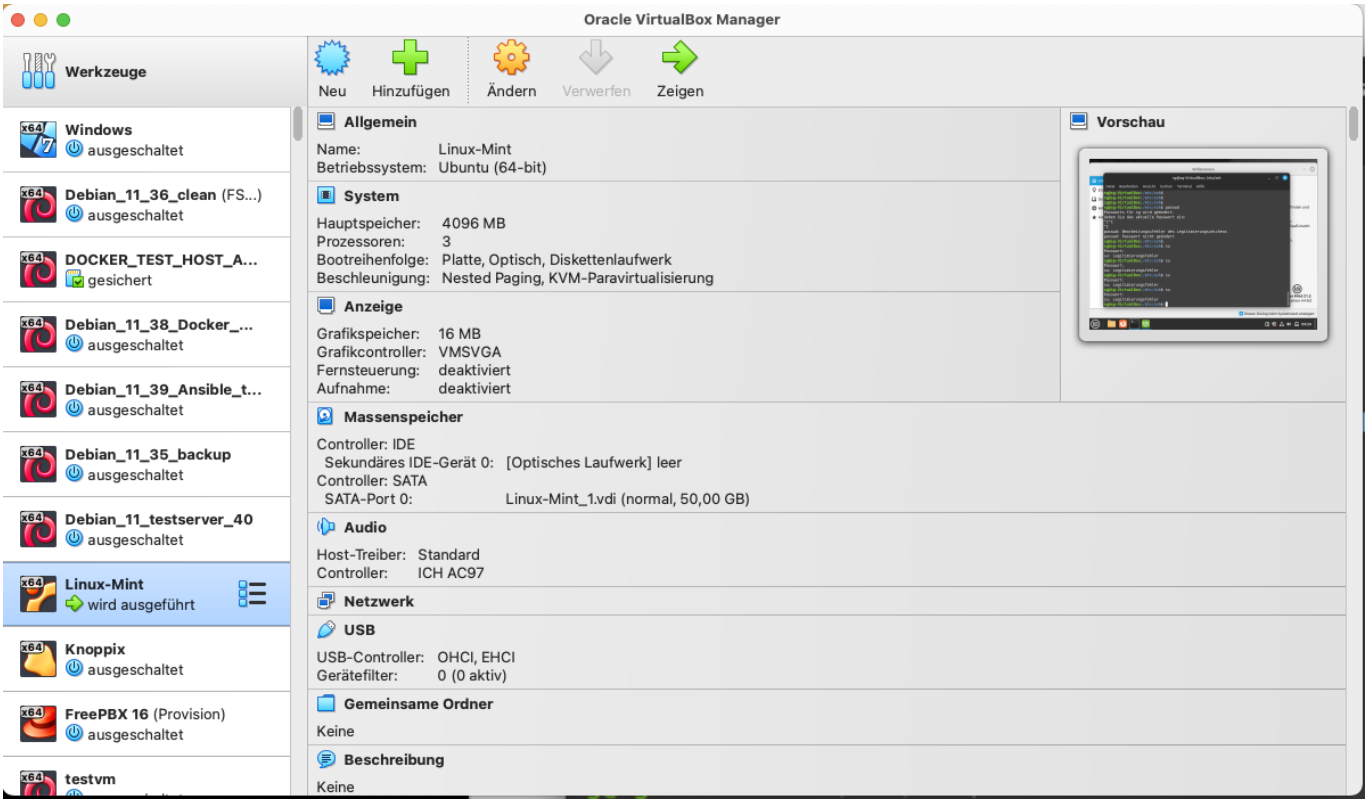
Solutions for Module "Linux"

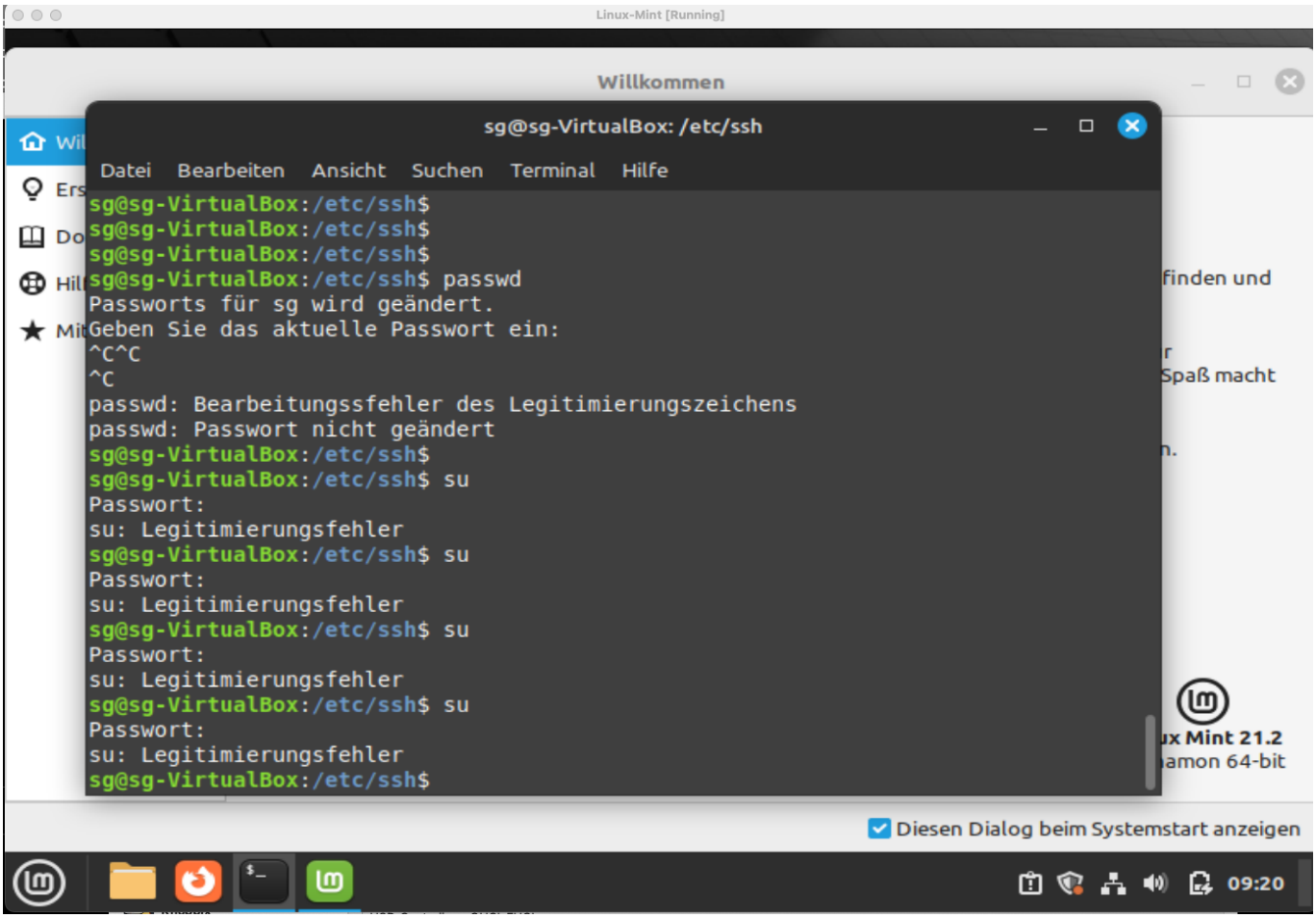
▶ Solution 1: Linux Mint Virtual Machine

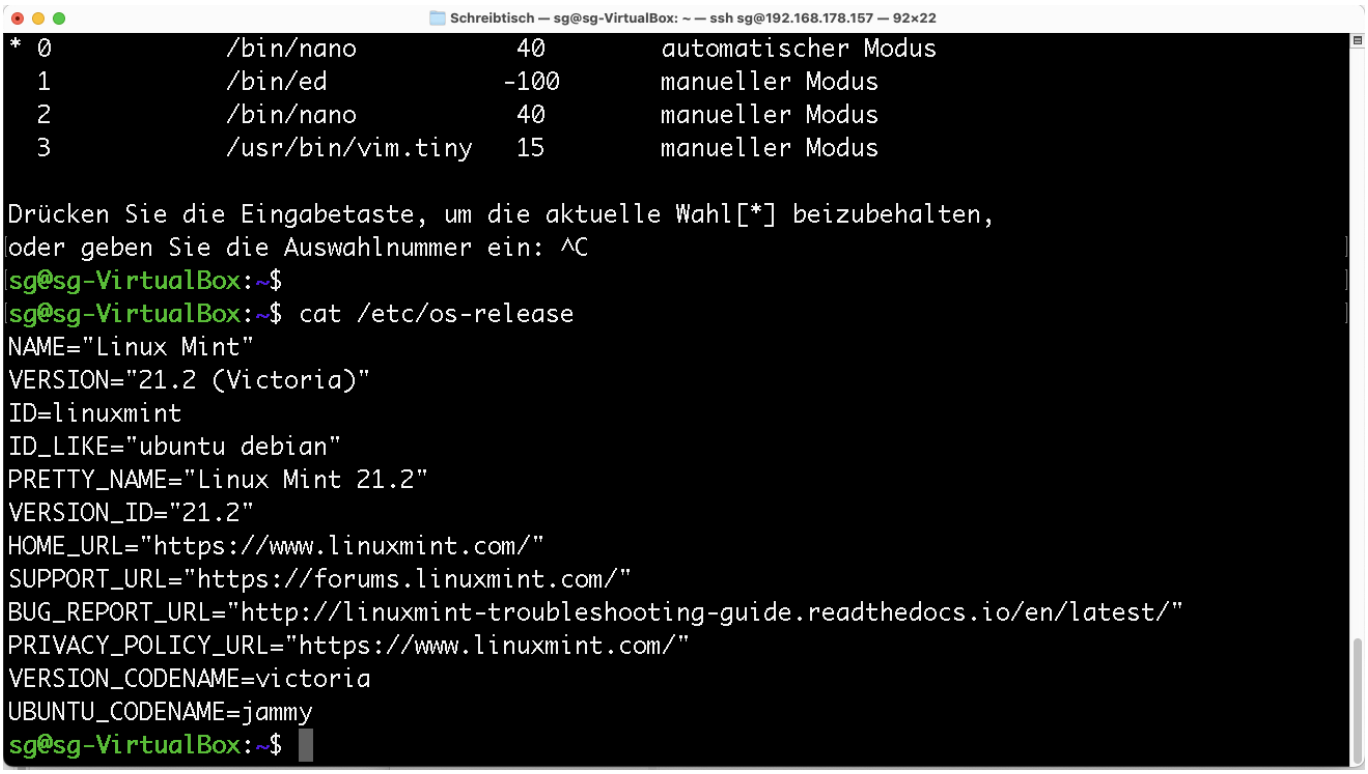
> Exercise 1: Linux Mint Virtual Machine

- Create a Linux Mint Virtual Machine on your computer.
- Check the distribution, which package manager it uses (yum, apt, apt-get).
- Which CLI editor is configured (Nano, Vi, Vim).
- What software center/software manager it uses.
- Which shell is configured for your user.

I have installed Linux Mint from this source Linux Mint Source

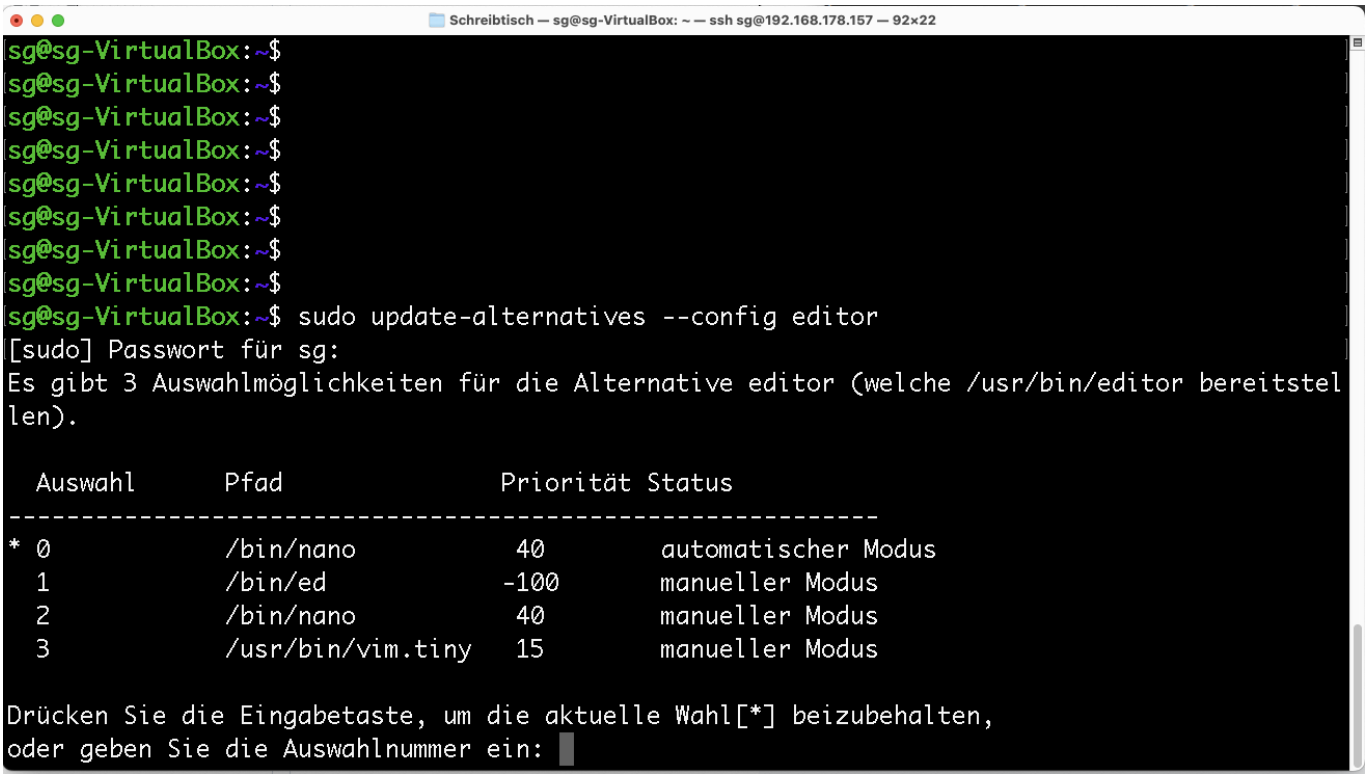






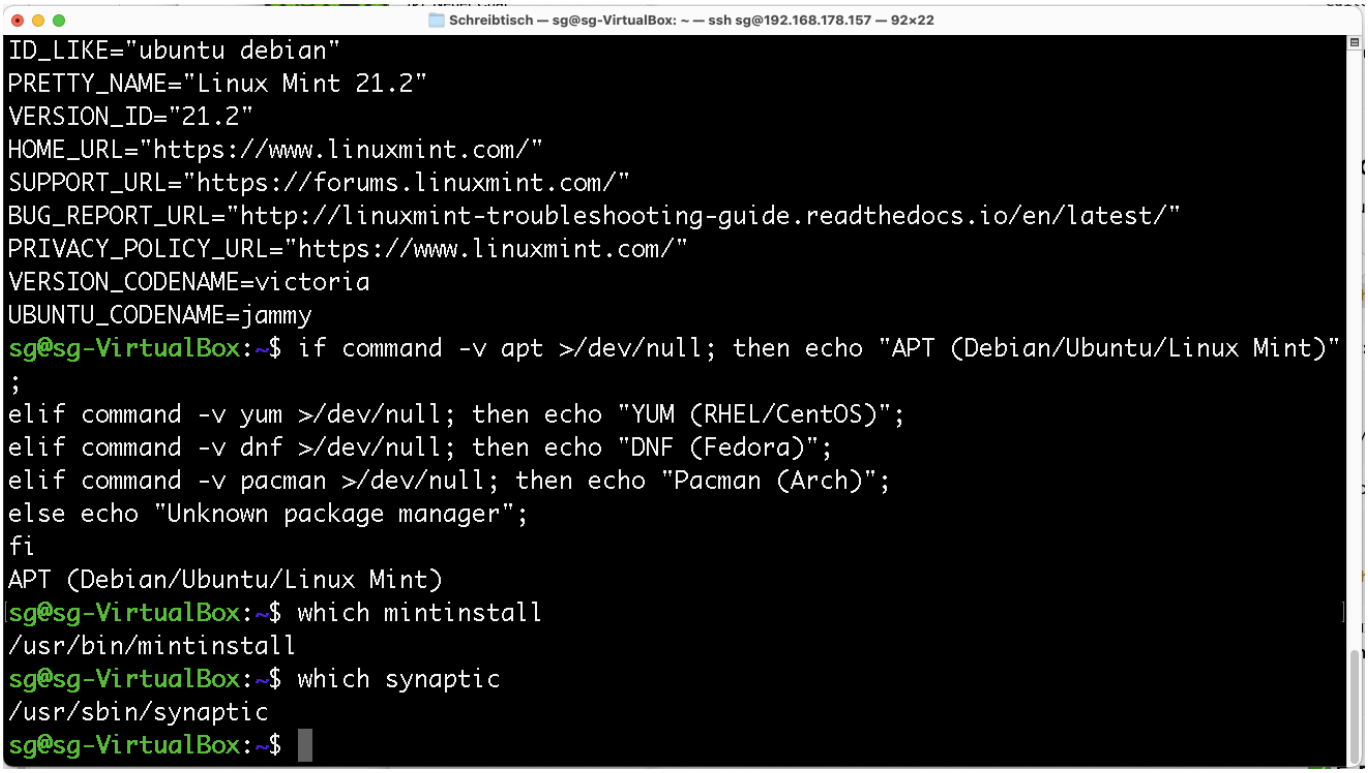
Nano was my default editor:

```
sudo update-alternatives --config editor
```



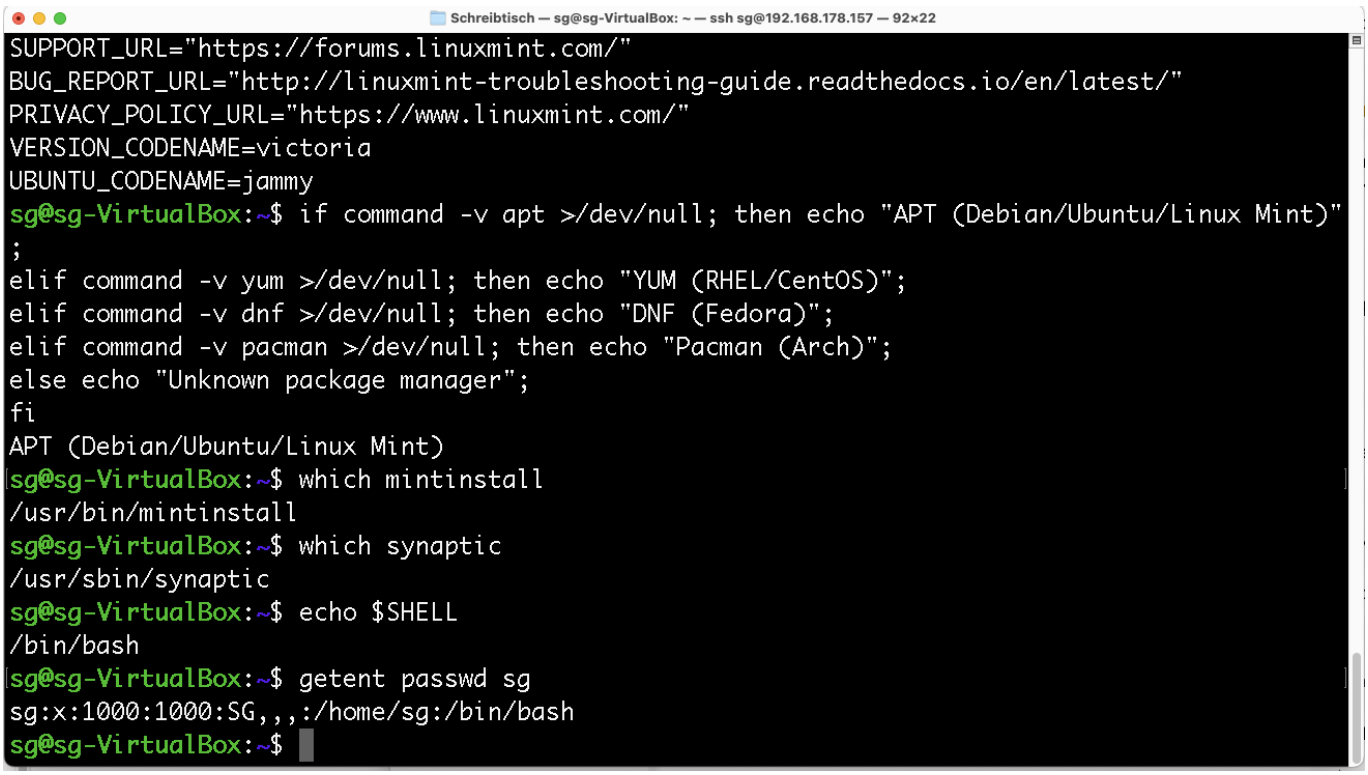
It uses as default the apt packet manager.

It uses the software manager:



I have used the following methods to identify the shell, it uses the bash shell:

```
echo $SHELL
ps -p $$
echo $0
echo $BASH
bash
```



▶ Solution 2: Bash script to install java and check the java version

> EXERCISE 2: Bash Script - Install Java

- Write a bash script using Vim editor that installs the latest java version and checks whether java was installed successfully by executing a java -version command. After installation command, it checks 3 conditions:
- whether java is installed at all
- whether an older Java version is installed (java version lower than 11)
- whether a java version of 11 or higher was installed

It prints relevant informative messages for all 3 conditions. Installation was successful if the 3rd condition is met and you have Java version 11 or higher available.

```
#!/bin/bash

apt update

package_name="default-jre"
apt_policy_output=$(apt policy $package_name)

installed_version=$(echo "$apt_policy_output" | awk '/Installiert:/ {print
$2}')

echo "$installed_version"

if [ "$installed_version" == "(keine)" ]
then
 echo "Java isn't installed, i will install java"
 install_result=$(apt install default-jre)
else
 echo "Java is installed, i will do nothing"
fi

java_version=$(java --version)
version=$(echo "$java_version" | head -n 1 | awk '{print $2}' | awk -F'.'
'{print $1}')
echo "$version"
version_int=$(expr $version + 0)

if [ "$version" == "" ]
then
     echo "The java installation has failed"
elif [ $version_int -lt 11 ]
then
     echo "An older java version is installed"
elif [ $version_int -ge 11 ]
then
     echo "The java version 11 or greater installed sucessfully installed"
fi
```

▶ Solutions 3-4-5: Bash script to sort the specific user processes

> EXERCISE 3: Bash Script - User Processes Write a bash script using Vim editor that checks all the processes running for the current user (USER env var) and prints out the processes in console.

- Hint: use ps aux command and grep for the user.

> EXERCISE 4: Bash Script - User Processes Sorted

- Extend the previous script to ask for a user input for sorting the processes output either by memory or CPU consumption, and print the sorted list.

> EXERCISE 5: Bash Script - Number of User Processes Sorted

- Extend the previous script to ask additionally for user input about how many processes to print. Hint: use head program to limit the number of outputs.

```
# Solution 3
#! /bin/bash

CURRENT_USER=$(printenv | grep USER | awk -F'=' '{print $2}')

echo "The current is: $CURRENT_USER"

USER_PROCESSES=$(ps aux | grep $CURRENT_USER)

echo -e "The processes from the user: $CURRENT_USER are :
\n$USER_PROCESSES"
```

```
# Solution 4
#! /bin/bash

read -p "Please enter CPU or MEM for user sorted processes list:"
MEM_CPU_ENUM

CURRENT_USER=$(printenv | grep USER | awk -F'=' '{print $2}')
echo "The current is: $CURRENT_USER"
echo "your entered input is: $MEM_CPU_ENUM"

if [ "$MEM_CPU_ENUM" == "CPU" ]
then
   USER_PROCESSES=$(ps aux --sort -pcpu | grep $CURRENT_USER)
   echo -e "The processes sorted by CPU from the user: $CURRENT_USER are :
\n$USER_PROCESSES"

elif [ "$MEM_CPU_ENUM" == "MEM" ]
then
```

```
    USER_PROCESSES=$(ps aux --sort -pmem | grep $CURRENT_USER)
    echo -e "The processes sorted by memory from the user: $CURRENT_USER
are : \n$USER_PROCESSES"
else
    echo "No valid input was entered! "

fi
```

```
# Solution 5
#! /bin/bash

read -p "Please enter CPU or MEM for user sorted processes list:"
MEM_CPU_ENUM
read -p "Please enter the line count you want to show :" LINE_COUNT

CURRENT_USER=$(printenv | grep USER | awk -F'=' '{print $2}')

echo "The current is: $CURRENT_USER"
echo "your entered input is: $MEM_CPU_ENUM"


if [ "$MEM_CPU_ENUM" == "CPU" ]
then
    USER_PROCESSES=$(ps aux --sort -pcpu | grep $CURRENT_USER | head -n
$LINE_COUNT)
    echo -e "The processes sorted by CPU from the user: $CURRENT_USER are :
\n$USER_PROCESSES"

elif [ "$MEM_CPU_ENUM" == "MEM" ]
then
    USER_PROCESSES=$(ps aux --sort -pmem | grep $CURRENT_USER | head -n
$LINE_COUNT)
    echo -e "The processes sorted by memory from the user: $CURRENT_USER
are : \n$USER_PROCESSES"
else
    echo "No valid input was entered! "

fi
```

▶ Solutions 6-7: Bash script to start a nodejs app

Context: We have a ready NodeJS application that needs to run on a server. The app is already configured to read in environment variables.

> EXERCISE 6: Bash Script - Start Node App

Write a bash script with following logic:

- Install NodeJS and NPM and print out which versions were installed
- Download an artifact file from the URL: https://node-envvars-artifact.s3.eu-west-2.amazonaws.com/bootcamp-node-envvars-project-1.0.0.tgz. >
- Hint: use curl or wget
- Unzip the downloaded file
- Set the following needed environment variables: APP_ENV=dev, DB_USER=myuser, DB_PWD=mysecret
- Change into the unzipped package directory
- Run the NodeJS application by executing the following commands: npm install and node server.js

> Notes: Make sure to run the application in background so that it doesn't block the terminal session where you execute the shell script If any of the variables is not set, the node app will print error message that env vars is not set and exit It will give you a warning about LOG_DIR variable not set. You can ignore it for now.

> EXERCISE 7: Bash Script - Node App Check Status Extend the script to check after running the application that the application has successfully started and prints out the application's > > running process and the port where it's listening.

```
#!/bin/bash

# prepare environment, install all tools
apt update

echo "install tree, node, npm, curl, wget, net-tools"
apt install -y tree nodejs npm curl net-tools

working_dir=./node_js_app

if [ -d "$working_dir" ]; then
  echo "this working directory exists, remove everthing in this directory"
  rm -rf $working_dir/*
  cd $working_dir
else
echo "this directory doesn't exist, create the directory"
mkdir $working_dir
cd $working_dir
fi

echo "the working directory is: $(pwd)"

# fetch NodeJS project archive from s3 bucket
wget https://node-envvars-artifact.s3.eu-west-2.amazonaws.com/bootcamp-node-envvars-project-1.0.0.tgz

# extract the project archive to ./package folder
tar -zxf ./bootcamp-node-envvars-project-1.0.0.tgz

echo "working tree view: $(tree)"

NODEJS_VERSION=$(node --version)
```

```
NPM_VERSION=$(npm --version)

echo "the installed NPM version is: $NPM_VERSION"
echo "the installed NODEJS version is: $NODEJS_VERSION"

#export the env variables
export APP_ENV=dev;export DB_PWD=mysecret;export DB_USER=myuser


cd ./package
echo "$(pwd)"

npm install

nohup node server.js &

ps aux | grep "node server.js" | grep -v grep

ss -tulpen | grep :3000
```

▶ Solutions 8-9: Bash script to start a nodejs app with a specific user and a specific log directory

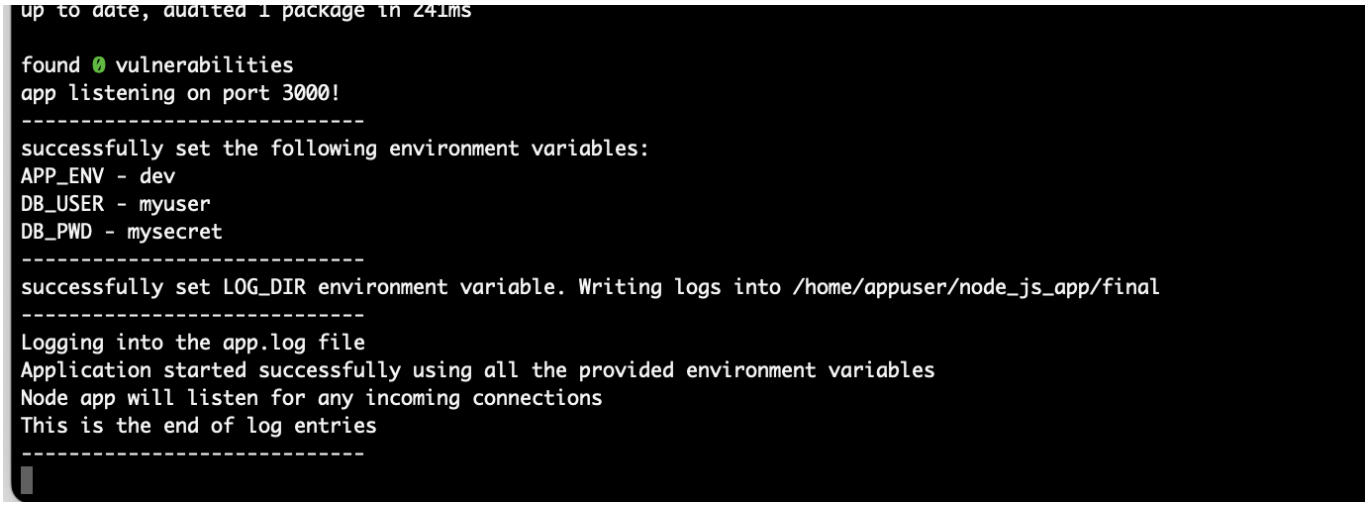
> EXERCISE 8: Bash Script - Node App with Log Directory

- Extend the script to accept a parameter input log_directory: a directory where application will write logs.
- The script will check whether the parameter value is a directory name that doesn't exist and will create the directory, if it does exist,
- it sets the env var LOG_DIR to the directory's absolute path before running the application, so the application can read the LOG_DIR environment variable and write its logs there.

> Note: Check the app.log file in the provided LOG_DIR directory. This is what the output of running the application must look like: node-app-output.png

> EXERCISE 9: Bash Script - Node App with Service user

- You've been running the application with your user. But we need to adjust that and create own service user: myapp for the application to run. So extend the script to create the user and then run the application with the service user.




```
#!/bin/bash

# create the new app user
new_user=appuser
useradd appuser -m

# prepare environment, install all tools
apt update

echo "install tree, node, npm, curl, wget, net-tools"
apt install -y tree nodejs npm curl net-tools

read  -p "please enter the log directory in which the node-js app should
log: " log_dir
echo "entered log directory is: $log_dir"

working_dir="/home/$new_user/node_js_app"

log_dir_app="$working_dir/$log_dir"
echo "LOG_DIR: $log_dir_app"


if [ -d "$working_dir" ]; then
  echo "this working directory exists, do nothing"
else
  echo "this directory doesn't exist, create the directory"
  mkdir $working_dir
fi


if [ -d "$log_dir_app" ]; then
  echo "this log directory exists"
else
  echo "this directory doesn't exist, create the directory"
  mkdir -p $log_dir_app
fi

touch $log_dir_app/app.log
```

```
chown  $new_user:$new_user -R $log_dir_app
chown  $new_user:$new_user -R $working_dir
cd $working_dir
echo "the working directory is: $(pwd)"

# fetch NodeJS project archive from s3 bucket
runuser -l $new_user -c "wget https://node-envvars-artifact.s3.eu-west-
2.amazonaws.com/bootcamp-node-envvars-project-1.0.0.tgz"

# extract the project archive to ./package folder
runuser -l $new_user -c "tar -zxf ./bootcamp-node-envvars-project-
1.0.0.tgz"

echo "working tree view: $(tree)"

NODEJS_VERSION=$(node --version)
NPM_VERSION=$(npm --version)

echo "the installed NPM version is: $NPM_VERSION"
echo "the installed NODEJS version is: $NODEJS_VERSION"


cd /home/$new_user/package
echo "$(pwd)"

runuser -l $new_user -c "printenv"
runuser -l $new_user -c "export APP_ENV=dev &&
                         export DB_PWD=mysecret &&
                         export DB_USER=myuser &&
                         export LOG_DIR=$log_dir_app &&
                         npm install &&
                    node package/server.js &"

ps aux | grep "node server.js" | grep -v grep

ss -tulpen | grep :3000
```