System and Network Administration - Lab 5 - Bash scripting

Jaffar Totanji - j.totanji@innopolis.university

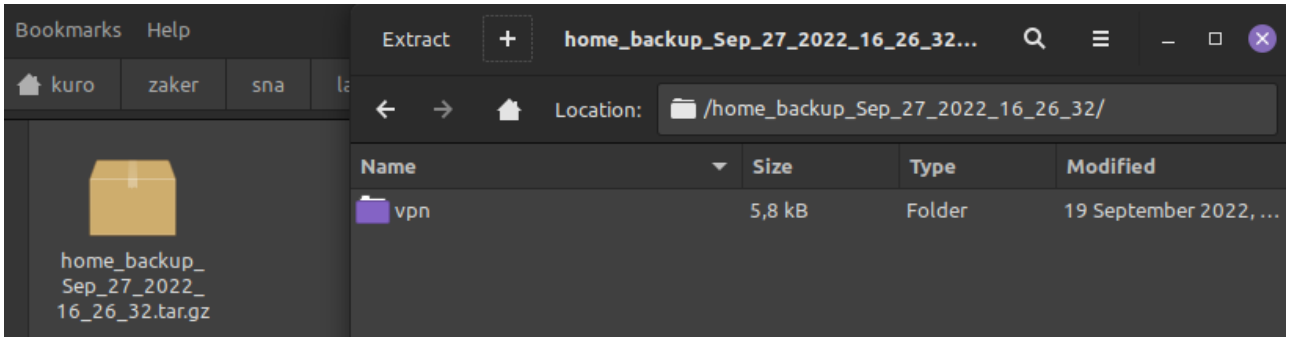
Questions to answer:

1. The environment variables in question are pretty well-known, in case on wanted to view them, this cane be done using `env` or `printenv`. The script itself is quite simple, it prints the a dscriptions along with its corresponding environment variable, with the exception of the IP Address which is extracted from `ip a` using `awk` then saved to `ipaddress` variable and then printed just like the others. Here's the result of running the script:

```
kuro@hp-pavilion:~/zaker/sna/lab_5$ ./scipt_1.sh
Username: kuro
Home Directory: /home/kuro
Shell: /bin/bash
Hostname: hp-pavilion
IP address: 127.0.0.1/8
```

2. The script is pretty straightforward, it first finds the current date in the required form using `date` and saves it to a variable `$cur_date`. Then creates a new directory `backups/home_backup_$cur_date/` under the current directory (not /backups/ in order not to mess with the file system on my main machine) if it doesnt already exist. The script then proceeds to copy the contents of the machine's home directory to the newly created directory using `cp -Rp` to that recursively and maintain all the file ownership/permissions while doing so. Finally, the script compresses to a `.tar.gz` archive with the same name and deletes the temporary folder. Here are screenshots of the script creating a backup of a simpler directory (my home directory is quite large):

```
kuro@hp-pavilion:~/zaker/sna/lab_5$ ./script_3.sh
home_backup_Sep_27_2022_16_26_32/
home_backup_Sep_27_2022_16_26_32/vpn/
home_backup_Sep_27_2022_16_26_32/vpn/us-atl.prod.surfshark.comsurfshark_openvpn_
udp.ovpn
home_backup_Sep_27_2022_16_26_32/vpn/us-atl.prod.surfshark.comsurfshark_openvpn_
tcp.ovpn
```



3. This can be achieved in one line using `find` to go through everything in the filesystem along with `-type f` flag to go through files only and the `-executable` flag to filter out files for which the current user has execute permissions. We then pass each result of that output to `grep` using `find`'s `-exec` flag which executes whatever command is given after it on every result of the search. The

`grep` command itself uses `-Rl` flags to execute the command recursively and show only the file names which contain the match `/bin/bash` instead of the match itself. Finally, `STDERR` is redirected to `/dev/null` to suppress errors caused by entries for which the current user has no read permission. Part of the output of the command looks like this:

```
kuro@hp-pavilion:~/zaker/sna/lab_5$ ./script_2.sh
/etc/gdm3/Xsession
/etc/openvpn/update-resolv-conf
/etc/X11/xinit/xinitrc.d/80xapp-gtk3-module.sh
/etc/cron.daily/mlocate
/etc/kernel/postinst.d/dkms
/etc/kernel/prerm.d/dkms
/etc/kernel/header_postinst.d/dkms
/etc/timeshift/restore-hooks.d/50_linuxmint
/usr/lib/grub/x86_64-efi/modinfo.sh
/usr/lib/grub/i386-pc/modinfo.sh
/usr/lib/grub/grub-multi-install
/usr/lib/dpkg/methods/apt/update
/usr/lib/dpkg/methods/apt/install
/usr/lib/gdm3/gdm-session-worker
/usr/lib/x86_64-linux-gnu/e2fsprogs/e2scrub_all_cron
/usr/lib/x86_64-linux-gnu/e2fsprogs/e2scrub_fail
/usr/lib/x86_64-linux-gnu/plymouth/plymouth-generate-initrd
/usr/lib/x86_64-linux-gnu/plymouth/plymouth-populate-initrd
/usr/lib/grub-legacy/update-grub
```

End of Exercises