Lab #14: Miscellaneous

Introduction to Linux (NSWI177)

Table of contents

The last lab is a mix of miscellaneous things that did not fit into any of the preceding labs and yet we consider them important (or interesting) enough to be mentioned.

We will be looking at printing support, running scripts periodically, downloading videos or restoring data from broken disks.

Printing with CUPS

Printing in Linux is handled by the **CUPS** subsystem that works out-of-the box with virtually every printer supporting IPP (internet printing protocol) and supports also many legacy printers.

Simple `sudo dnf install cups` installs the basic subsystem, extra drivers might be needed for specific models. **OpenPrinting.org** contains a searchable database to determine which (if any) drivers are needed. For example, for most HP printers you would need to install `hplip` package.

You typically want CUPS up and running on your system all the time, hence you need to enable it.

```
sudo systemctl start cups
sudo systemctl enable cups
```

CUPS has a nice web interface that you can use to configure your printers. For many modern network-connected printers, even that is often unnecessary as they will be auto-discovered correctly.

If you have started CUPS already, try visiting **http://localhost:631/**. Under the *Administration* tab, you can add new printers. Selecting the right model helps CUPS decide which options to show in the printing dialog and enables proper functioning of grayscale printing and similar features.

Scanning with Sane

Scanner support on Linux is handled with **SANE** (Scanner Access Now Easy). As with printing, most scanners will be autodetected and if you already know **GIMP**, it has SANE support. Add it with `sudo dnf install xsane-gimp`.

Actual scanning of the image can be done from *File -> Create -> XSane dialog* where you select your device, scanning properties (e.g., resolution or colors) and then you can start the actual scan.

Periodically running tasks with Cron

There are many tasks in your system that needs to be executed periodically. Many of them are related to system maintenance, such as log rotation but even normal users may want to perform regular tasks.

A typical example might be a backup of your `$HOME` or a day-to-day change of your desktop wallpaper.

From the administrator point of view, you need to install the cron daemon and start it. On Fedora, the actual package is called `cronie` but you still start and enable the `crond` service.

System-wide jobs (tasks) are defined `/etc/cron.*/` where you can directly place your scripts. For example, periodic backup of your machine would typically go as a script `backup.sh` into `/etc/cron.daily`.

If you want more fine-grained specification than the one offered by the `cron.daily` or `cron.hourly` directories, you can specify it in a special file inside `/etc/cron.d`.

There, each line specifies which cron job (i.e. which command) and when to be executed (and also under which user to execute, typically it will be `root`). It is possible to specify minute (0-59), hour (0-23), day of month (1-31), month (1-12) and day of week (0-6, 0 is Sunday) or use `*` for any.

Therefore, the following will execute `/usr/local/bin/backup.sh` every day 85 minutes after midnight (i.e., 1:25 am). The second line will call `big-backup.sh` every Sunday morning.

```
25 1 * * * root /usr/local/bin/backup.sh
0 8 * * 0 root /usr/local/bin/big-backup.sh
```

Note that `cron.d` will typically contain a special call of the following form that ensures that `cron.hourly` scripts are executed (i.e., the `cronie` deamon itself looks only inside `/etc/cron.d`, the use of `cron.daily` or `cron.monthly` is handled by extra jobs).

```
01 * * * * root run-parts /etc/cron.hourly
```

Running as a normal user

Normal (i.e., non superuser) users cannot edit files under `/etc/cron.d`. Instead, they have command called `crontab` that can be used to edit their personal cron table (i.e., list of cron jobs).

Calling `crontab -l` will list current content of your cron table. It will probably print nothing.

To edit the cron table, execute it as `crontab -e`. It will launch your favourite editor where you can add lines in the above mentioned format, this time without the user specification.

For example, adding the following entry will change your desktop background every day.

```
1 1 * * * /home/intro/bin/change_desktop_background.sh
```

Of course, assuming you have such script in the given location. If you really want to try it, the following script works for Xfce and uses **Lorem Picsum**.

```
#!/bin/bash

# Update to your hardware configuration
screen_width=1920
screen_height=1080


wallpaper_path="$HOME/.wallpaper.jpg"


curl -L --silent "https://picsum.photos/$screen_width/$screen_height" >"$wallpaper_path"

# Xfce
# Select the right path from xfconf-query -lvc xfce4-desktop
xfconf-query -c xfce4-desktop -p /backdrop/screen0/monitor0/workspace0/last-image -s "$wallpap
er_path"

# LXDE
pcmanfm -w "$wallpaper_path"

# Sway
# For more details see `man 5 sway-output`
# You can also set a different wallpaper for each output (display)
# Run `swaymsg -t get_outputs` for getting specific output name
swaymsg output '*' bg "$wallpaper_path" fill
```

Nvidia drivers

We received several requests to describe installation of Nvidia drivers and provide some guidance. Unfortunately, it seems that none of the teachers actually have an Nvidia card at the moment to provide first-hand experience.

Therefore, we must redirect you to other sources on the internet. The following articles are reasonably new (written less than a year ago) to provide up-to-date information and instructions.

- **How to install the NVIDIA drivers on Fedora 32** (LinuxConfig.org)
- **How to install Nvidia Drivers on Fedora Workstation** (FOSSLinux.com)

Multimedia

Just a few bits of interesting possibilities, mostly related to control from the command line.

VLC

VLC is a popular multimedia player available for many platforms. We will not bore you with GUI but instead show several interesting command-line options. They might be useful in specific setups such as kiosk-mode (e.g., on an exhibition) or similar.

Following command-line switches will play the given video but split it into four different windows. Useful if you have multiple monitors with very thin edges, perhaps.

```
vlc --video-splitter wall --wall-cols 2 --wall-rows 2 --wall-element-aspect 4:3 video.mpg
```

VLC can be also used to stream video over the network. Assuming you have files `1.mp4`, `2.mp4` and `3.mp4` in your current directory, prepare the following file (name it `vod.vlc`):

```
new channel1 vod
setup channel1 input 1.mp4
setup channel1 enabled

new channel2 vod
setup channel2 input 2.mp4
setup channel2 enabled

new channel3 vod
setup channel3 input 3.mp4
setup channel3 enabled
```

Each block setups one video-on-demand configuration (i.e., the client asks for a particular video and the VLC server will provide it).

Next, we will start command-line VLC in server mode to listen for RTSP connections on port 5554.

```
cvlc --vlm-conf vod.vlc  --rtsp-tcp --rtsp-port 5554
```

We can now play the selected video with the following command.

```
vlc rtsp://127.0.0.1:5554/channel3
```

The above assumes you play the video on the same machine. The first command can be extended with `--rtsp-host` to listen on another interface to stream the video over network (and run the second command on a different machine).

`youtube-dl`

youtube-dl is a video downloader that is able to find and download videos from various websites. As a command-line utility, it can be used in scripts or for downloading videos for later viewing (i.e., download with fast connectivity and view later).

As an example, we will use it to download videos from **Pexels** that is hosting videos and photos that are **free to use**.

```
youtube-dl "https://www.pexels.com/video/penguins-at-the-zoo-1528489/"
```

After the download completes, you shall see `Pexels Videos 1528489-296271282.mp4` file with the video.

Another supported site is **OpenClassroom** of the **Standford University**. Here, some of the videos are in FLV format but `youtube-dl` can convert the video by simply adding `--recode-video mp4`:

```
youtube-dl --recode-video mp4 "http://openclassroom.stanford.edu/MainFolder/VideoPage.php?cour
se=IntroToAlgorithms&video=CS161L1P1&speed=100"
```

`youtube-dl` supports many other sites: downloading from many of them is explicitly prohibited by their terms of use, for many using similar tools is bordering on the concept of *fair use* policy.

`ffmpeg`

Another interesting tool is `ffmpeg` that is a general converter of video formats. Apart from trivial conversion between various formats it can do a plethorea of extra effects.

As usual: the advantage of command-line interface is substantial for mass conversions where no user interactivity is needed.

For example, the following command converts the audio to AAC while keeping video without any modification. We use it for the lab videos so they work in Firefox too.

```
ffmpeg -i "input_file.mp4" -c:v copy -c:a aac "output_file.mp4"
```

But it can do much more than that. Let's download the following clips first.

```
youtube-dl -o 1.mp4 "https://www.pexels.com/video/snow-removal-on-an-airport-runway-3657191/"
youtube-dl -o 2.mp4 "https://www.pexels.com/video/passenger-airplanes-of-different-airlines-ta
xiing-on-the-airport-ground-3678399/"
youtube-dl -o 3.mp4 "https://www.pexels.com/video/flying-above-the-clouds-4070515/"
```

The following command then puts these clips next to each other into a single video. It uses multiple `-i` to actually load multiple input streams and then it uses a complex filter to stack the videos next to each other. The `-t` is used to limit the conversion to first 10 seconds only.

The variables are used only to convey the whole pipeline. In `f_rescale` we rescale all the videos. The `[0]` denotes the first input file (first in the array of input files), the `[v0]` is a user identifier to name the result of the filter. In `f_stack`, we use the `xstack` filter with the given layout specification – see **this page** if you are interested in details about positioning with `0_0|w0_0|w0_h1`.

```
f_rescale='[0]scale=-1:360[v0];[1]scale=-1:180[v1];[2]scale=-1:180[v2]'
f_stack='[v0][v1][v2]xstack=inputs=3:layout=0_0|w0_0|w0_h1[v]'

ffmpeg -i 1.mp4 -i 2.mp4 -i 3.mp4 -filter_complex "$f_rescale;$f_stack" -map "[v]" -t 10 outpu
t.mp4
```

If you need to perform the above for a single video, usinging an interactive editor would be certainly easier. But if you need to process multiple videos with same or similar configuration, `ffmpeg` might be a better choice.

Restoring broken file-system

The following text (and the utilities mentioned) may come handy if you hard-drive (or USB stick or camera SD card) became corrupted – either because of mechanical failure or software damage.

In that case, it is needed to copy the data as soon as possible to some reliable storage and recover them. Note that many failures are not fatal: you can still copy data from the disk but it is not possible to mount the partition. If the disk fails completely (because of hardware issues), you may need to find a company that specializes in data recovery to fix the hardware first.

Copying a disk image with dd

First, we will copy the disk to a disk image. We will not try to interpret it at all: we will copy it as a stream of bytes without any structure. The best command for that is `dd`: it has a non-standard syntax but works very well. Assuming your drive appeared as `/dev/sdb`, you need to run the following.

```
dd if=/dev/sdb of=$HOME/disk.img bs=1024
```

`if` and `of` refer to input and output file, `bs` is block size and sets the size of an internal buffer. Changing this value can affect performance but it depends on many factors. Note how we easily leverage the fact that the drive can be accessed as a file.

If the drive has some bad sectors, `dd` will either hang or end with error. In that case, it may help to use `ddrescue` (`sudo dnf install ddrescue`) that is able to skip the bad sectors and copy only data that can be copied.

```
ddrescue /dev/sdb $HOME/disk.img $HOME/disk.map
```

The `.map` file is a auxiliary file where `ddrescue` stores information about skipped blocks etc. and can be used to restart `ddrescue`.

Restoring the files

Once the disk is safely copied, we can try to restore the files. It is highly recommended to always work on a copy of the disk image in case we manage to break it even more. This can be quite demanding in terms of a disk space: in the end it all comes down to money – are the data worth more than buying an extra disk or even bringing it completely to a professional company focusing on this sort of work. Note that unless the drive is physically broken, we have a very good chance of recovering the data by ourselves with very little work.

The recovery programs are based on a very simple concept: corrupted disk is rarely corrupted fully. Often, the mounting fails only because few (but important) blocks were corrupted. So, instead of trying to mount it, the programs scans the whole content and try to find sequences of bytes that look like a file signature (recall, for example, how **GIF format looks like**). Thus, typically these programs are able to recover the content of the files but you loose their original filenames and directory hierarchy.

The first program we will show is **photorec** (`sudo dnf install testdisk`). Before starting it, prepare an empty directory where to store the results.

It takes a single argument: the file image to scan. It then starts an interactive mode where you select where to store the recovered files and also guess on file system type (for most cases, it will be FAT or NTFS). Then it tries to recover the files. Nothing more, nothing less.

`photorec` is able to recover **plenty of file formats** including JPEG, MP3, ZIP files (this includes also ODT and DOCX) or even RTF files.

Another tool is **recoverjpeg** that focuses on photo recovery. Unlike `photorec`, `recoverjpeg` runs completely non-interactively and offers some extra parameters that allow you to fine-tune the recovery process.

`recoverjpeg` is not packaged for Fedora: you can try installing it manually or play with `photorec` only (and hope you will never need it).

Exercise

As an exercise, try to recover files from **this file image**.

First check that you cannot mount it manually and then run `photorec` or `recoverjpeg` to recover the files.

You shall recover 5 photos, one PDF and one ODT file. As a practical note, most JPEGs have some kind of EXIF information so it is possible to sort them semiautomatically (e.g., after recovering of several GBs of photos from your camera).

Solution.

Graded tasks

Note that only the first task is submitted through GitLab. Also, only that task has an automated tests in the pipeline.

`14/recovery.txt` (25 points)

File `linux.ms.mff.cuni.cz:~/lab14.img` is a disk image with several JPEG photographs. Unfortunately, the disk image is broken and cannot be mounted directly. Try to restore its content (i.e., the files on this disk).

You shall see 9 JPEG files after restoration. One of them contains a text: copy the text to the file `14/recovery.txt` (to GitLab).

Note that we can create the source file `~/lab14.img` only after you login to the remote machine for the first time.

If the file is not there, wait for the next work day for the file to appear.

The text is part of the photograph, you will need to go through the restored files manually.

Do not leave this task for the last minute and contact us if the file has not appeared as explained in the previous paragraph.

Cron at `linux.ms.mff.cuni.cz` (25 points)

Setup a cron job at `linux.ms.mff.cuni.cz`. The job shall run some time between 1AM and 2AM (using local time on the machine, i.e. do not recompute for your time zone and simply use `1`) every day and must copy file `/srv/nswi177/cron.txt` to `$HOME/LAB14_CRON.txt`.

Note that we will verify functionality by modifying the source file so you really need to setup a periodic execution of your script.

Important: do not create this file on GitLab but configure it on `linux.ms.mff.cuni.cz`.

Survey (50 points)

Survey text will be available next week (i.e. around Monday 31st).

We would like to collect your thoughts and comments about this course. We already had some discussions but we would like to hear from a broader audience than those few of you that participated in the discussions.

Therefore, we have created a survey with several questions regarding this course and what you liked and disliked about it.

Because completing the survey can take some time, we want to award points for submission. But we also want to preserve anonymity as not everyone might be comfortable with a survey that can be tracked back to the author of the answers.

Therefore, you will fill-in a Google Form survey (it will be available both in English and in Czech) and once you complete the survey (anonymously), you will create file `14/SURVEY` so we can assign you the points.

Update: Links to the survey are available at `linux.ms.mff.cuni.cz:/srv/nswi177/14_survey`.

Deadline: June 21, AoE

Solutions submitted after the deadline will not be accepted.

Note that at the time of the deadline we will download the contents of your project and start the evaluation. Anything uploaded/modified later on will not be taken into account!

Note that we will be looking *only at your master branch* (unless *explicitly specified otherwise*), do not forget to merge from other branches if you are using them.

Changelog

2021-06-01: Link to survey published.