Introduction to the Linux OS

Peter Huszár
KFA: Department of Atmospheric Physics

Pavel Řezníček
ÚČJF: Institute of particle and nuclear physics

November 9, 2026

Overview and Organization

Introduction to the Operation system Linux, focus on the command line, scripting, basic services and tools used in (not only) physics: tasks automation in data processing and modeling

Organization

- Graded Assessment (KZ): attendance to the lectures, worked out homeworks

Literature

- C. Herborth: Unix a Linux - Názorný průvodce, Computer Press, Praha, 2006
- D. J. Barrett: Linux - Kapesní přehled, Computer Press, Praha, 2006
- M. Sobell: Mistrovství v RedHat a Fedora Linux, Computer Press, Praha, 2006
- M. Sobell: Linux - praktický průvodce, Computer Press, Praha, 2002
- E. Siever: Linux v kostce, Computer Press, Praha, 1999
- Number of online sources...

Study materials and homeworks

- http://kfa.mff.cuni.cz/linux

KFA MFF UK

ÚČJF MFF UK

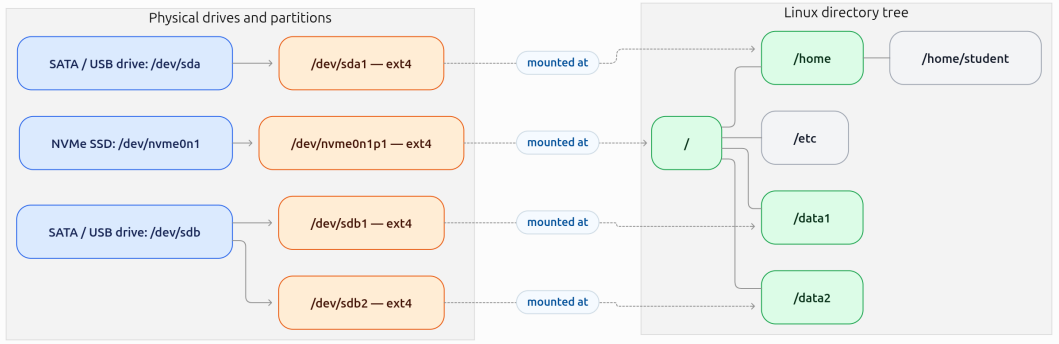
Syllabus

1. UNIX systems, history, installation, basic applications
2. Structure of the Linux OS, file systems, hierarchy of the file system
3. Command line, shells, remote access (ssh, ftp)
4. Processes and their administration, basic system commands, packages, printing
5. Users, file and directory permissions
6. Work with files and directories, file compression, links, partition
7. Text-file processing commands, redirection, pipeline
8. Regular expressions
9. Command line based text editors
10. User and system variables, output processing
11. Scripts: basic construction, conditionals, loops, functions, automation
12. Networking, server-client services: http, (s)ftp, scp, ssh, sshfs, nfs
13. Programming in Linux (examples of Fortran, C/C++, Python), version control systems, documents in Latex

Disk/partition/filesystem handling utilities

Basic disk/partition handling utilities

Physical drives (disks) vs. Linux directory structure



Basic disk/partition handling utilities

Files/directories are structures physically or virtually written into harddrives/flashdrives, cdrom/dvdroms, tapes, virtual memory etc. These are usually divided into partitions which enables to separate logically distinct parts of the filesystem hierarchy structure. For example /home will be placed on different partition than /.

- Each parition has its own filesystem (FS)
- Partitions are 'mounted' to directories
- disk → partition → filesystem → mount point
- df – report file system disk space usage. It shows which partition is 'mounted' to which directory.
  - /dev/sda, /dev/sdb, /dev/sdc are 'device' files corresponding to individual devices (SATA / SCSI Drives)
  - /dev/sda1, sda2 mean the partition number on the disk (i.e. each partition has a separate device file in /dev)
  - /dev/nvme0n1 and /dev/nvme0n1p1 /dev/nvme0n1p2 for newer solid state drives like SSDs
  - tmpfs is a temporal filesystem created in the RAM, so it behaves as a normal hraddisk, but when the computer is turned off, it is gone.
  - df -Th shows the type of the filesystems mounted (ext2, ext3, ext4, reiserfs, vfat, ntfs etc).
- lsblk – very useful utility to show the disk/partitions structure/filesystems and IDs of partitions
- hdparm – get/set hard diskparameters: tuning your hard disk or DVD drive, measuring read speed, deliver valuable information about the device, change important drive settings, and even erase disks securely.

- FDISK - manipulate disk partition table - "partitioning". A very powerful utility to create,modify and delete partitions on a disk. USE WITH CAUTION AS YOU CAN EASILY DAMAGE YOUR EXISTING FILES/DIRECTORIES!!! More info on: https://www.tldp.org/HOWTO/Partition/fdisk_partitioning.html or man fdisk.

```
fdisk -l # shows all disks connected to the computer (needs root privileges)
fdisk /dev/sdb # prompt to manipulate with disk /dev/sdb, press m to get all the commands
```

- Once partitions are created on a disk, we can create different filesystems (Linux File System list https://static.javatpoint.com/linux/images/linux-file-system2.png, https://en.wikipedia.org/wiki/List_of_file_systems mkfs.filesystem - "formatting".

```
mkfs.ext4 /dev/sdb1 # create Ext4 filesystem on disk /dev/sdb and partition 1.
```

- fsck - check and repair a Linux filesystem

```
fsck.ext4 /dev/sdb1 # check and repairs the ext4 filesystem on disk /dev/sdb and partition 1.
```

In order to access the files/directories on the partitions, this partition has to be 'mounted'.

- mount - mount a filesystem

```
mount -t type device dir # The standard form of the mount command
# type is the filesystem type, device is the device file in /dev and dir is the directory where to mount
mount /dev/sdb1 /data # -t type may be omitted as the system will find it out automatically
mount -o ro /dev/sdb1 /data # "-o" = adding mount options, ro = read-only
```

- mounting ISO filesystem (CD/DVD *.iso image)

```
mount /path/to/image.iso /media/iso -o loop
```

- Mounting NFS - network filesystem

```
mount -t nfs 10.76.120.40:/volume1/d01 /home/nas01
```

- tmpfs - temporal/virtual filesystem represented in the RAM

```
mount -t tmpfs -o size=1G tmpfs /mount/tmp # this takes 1G from the RAM and allocate it to /mnt/tmp
```

- unmounting filesystems

```
umount /mount/point
```

The `/etc/fstab` file is a system configuration file that contains all disks/partitions that are to be mounted automatically during boot. Each file system is described on a separate line. Each line contains six fields separated by one or more spaces or tabs.
The `mount` command reads each line at system boot and mounts the disks according to the options given on the lines.

```
# <file system> <mount point> <type> <options> <dump> <pass>
# filesystem root disk
UUID=f3b12014-faea-49c8-9e9a-f397b002454d / ext4 errors=remount-ro 0 1
# boot partitions
UUID=515B-6053     /boot/efi vfat umask=0077  0 1
UUID=141ad68b-b7f4-4074-a89d-43801222d5fb none      swap sw   0       0
# 2T data disk
UUID=6f3ffadf-418f-4e15-baaf-93ca06ffaf54 /data   ext4  errors=remount-ro 0       0
# NAS1 and NAS2
10.76.120.40:/volume1/d01   /home/nas01 nfs defaults  0 0
10.76.120.40:/volume2/d02   /home/nas02 nfs rw   0 0
# cesnet high capacity storage center Jihlava
sshfs#huszarpet@ssh.du4.cesnet.cz:/tape_tape/archive/VO_cuni_mff_meteo/home/huszarpet \
            /mnt/cesnet fuse defaults,idmap=user,IdentityFile=/home/huszi/.ssh/id_rsa,allow_other 0 0
```

- Disks can be specified either by the device file e.g. `/dev/sda2`, however, this is not uniq and can change if disk are connected to the computer in a different order.
- A much safer option is to specify the disks by their UUID number (`lsblk -f /dev/sda2`)
- Further options: type = FS type; options = mount options (like ro for read only); dump = backup operation with 'dump' (0/1); pass = FS check with fsck (root always 1, other 2, nocheck = 0)
- to mount the "lines" of fstab (when changes made) use `mount -a`.

Some more examples:

```
findmnt -T .            # Which filesystem contains the current directory?
df -h .                 # Space available on the filesystem containing .
du -sh ./results        # Disk space used by this directory
```

Suppose we got an old (or new) disk and want to format it as a single partition to the ext4 filesystem, then mount it to /mnt/mydisk.

- fdisk to repartition (delete old partitions and parition new ones)
- mkfs.ext4 to create a FS (formatting)
- mount it (for one time use) and permanently via fstab

Text manipulation

Basic text manipulation commands

Commands to view and transform text(files); and to extract information about the text

- cat, more and less - to view text file contents (read-only!)

```
cat /my/text/file.txt # writes the contents of the file
                      # on the terminal - the standard output (stdout) and returns to prompt
                      # seemingly useless command in case of long files, but wait-for-it;-)
                      # (e.g. for putting two file ater "each other"  cat files1 file2 > file_final)
more /my/text/file.txt # view text by pages ("Space"), q or ctrl-c to quit
less /my/text/file.txt # view text by pages, PageUp/PageDown,Up/Down keys, Space -- 'q' to quit
                       # less is more than more and unlike more, reads only that part of the file
                       # which is shown
```

- head and tail - we are oftne interested in what the begining/ending of text file contains

```
head -n 20 /my/text/file.txt # prints the first 20 lines of the textfile
head -n -20 my/text/file.txt # prints all but the last 20 lines
tail -n 20 /my/text/file.txt # prints the last 20 lines of the textfile
tail -n +20 /my/text/file.txt # prints the last lines starting with line number 20
                              # if -c is used instead -n in both cases
                              # printing applies for bytes (first N/last N bytes)
```

- wc - word count (counts bytes/characters/words/lines)

```
wc  /my/text/files.txt # Print newline, word, and byte counts for each file
                       # if more files provided, print info for each and the total
wc -l /my/text/files.txt # number of lines
wc -w /my/text/files.txt # number of words
wc -c /my/text/files.txt # number of bytes, -m number of characters (bytes and chars are not the same)
```

Basic text manipulation commands (cont'd)

Commands to view and transform text(files)

- cut - cuts parts of each line of a text ("vertical" version of head/tail). It uses "deliminator", default is the tabulator. Very useful to extract columns in a text file, where columns are separated by a certain character (e.g. comma)

```
cut -d"{deliminator}" -f1 /my/text/file.txt # for each line prints everything before the first appearance
                                            # of the {deliminato} when it is the tabulator, -d is not import
cut -d":" -f1,3,5,10-15 # takes the first column, the 3rd, the 5th and than the 10-15.
                        # --complement, select the rest
                        # -d".", -d";", -d" ", -d"-", -d"a" etc.
cut -c X-Y              # cut from Xth to the Yth character on each line of the input file
```

- paste - line-by-line merge files "next" to each other. This requires a delimiter, tabulator is again the default one

```
paste -d"{deliminator}" /my/text/file1.txt /my/text/file1.txt  # for each line1/2 of file1/2
                                                               # prints line1/2 next to each other
                                                               # separated by {deliminator}
```

- fmt - simple text formatter

```
fmt /my/file.txt # by default puts all words  in a single line and prints
fmt -w 10 /my/file.txt # print the text in the specified width (does not wrap words!!!)
fmt -t file.txt # add indentation for the first line different from others
fmt -u file.txt # uses one space between words and two spaces after sentences for formatting
```

Basic text manipulation commands (cont'd 2)

- sort - sort lines of file according to dictionary,numerical value etc.

```
sort -d /my/files.txt # dictionary sort
sort -b /my/files.txt # ignore leading blank characters
sort -n /my/file-with-numbers.txt # numeric sort
sort -r /my/file.txt # reverse the result
sort -u /my/file.txt # print only unique lines
sort -k 2 /my/file.txt # sort according to the second "column"
```

- uniq - report/omit repeated lines (related to sort -u)

```
uniq -c /my/file.txt # prefix lines by the number of occurrences
uniq -d/-u /my/file.txt # prints only duplicated lines/unique lines
```

- rev - a trivial utility that reverses each line characterwise

```
rev /my/file.txt # the output will be the file with same size but all lines in
                 # reversed order (you can doubt the usefullness :-))
```

- join - join lines of two files on a common field (joins two "column" files with different columns as fields delimited by a delimiter based on values in selected columns)

```
join -t, -1 FILE1COLUMN -2 FILE2COLUMN FILE1 FILE2 # joins lines from both file based on
                                                   # the value of FILE1COLUMN equals FILE2COLUMN
join -t, -1 1 -2 2 file1 file2 # join lines of file1 with file2 where the value of columns 1 from
                               # 1st file and column 2 from 2nd file equal;-t is the delimiter (e.g. -t,)
```

- For join to be successfull, files must be sorted, unless –nocheck-order is used
- –headers will skip the first line on each file as header line