Introduction to the Linux OS

Peter Huszár
KFA: Department of Atmospheric Physics

Pavel Řezníček
ÚČJF: Institute of particle and nuclear physics

November 10, 2025

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

File and Directory Manipulation

File and directory manipulation

In theory, files and directories (folders) can be:

- **Deleted** - remove file(s) and/or directory(-ies) from the filesystem
- **Copied** - copy file(s) and/or directory(-ies) to different part of the filesystem
- **Moved** - move file(s) and/or directory(-ies) to different part of the filesystem = copy and delete the original
- **Linked** - link file(s) and/or directory(-ies) to different part of the filesystem
- **Created** - create a new file(s) and/or directory(-ies)

i.e. operations on whole files, not on part of the data in the files (next lesson)
Wildcards – useful constructions to perform actions on more than one file/directory

Removes files and directories from the FS.

- For files

```
rm [OPTION]... [FILE]...
rm /my/file # one or many files
rm -i files/in/my/dir/file1 files/in/my/dir/file2 # interactive
rm -f /my/files # force removal
```

- For directories

```
rm -r [OPTION]... [DIRECTORY]...
rm -r /my/directory # one or many directories
rm -ri dirs/in/my/dir/directory1 dirs/in/my/dir/directory1 # interactive
rm -rf /my/dir # force removal
# combination of file(s) and directory(-ies)
rm -r files/in/my/dir/file1 dirs/in/my/dir/directory1
rmdir dir # for empty directories only
```

There is no undelete!!!

Once you delete with rm, your data are (mostly) gone! rm -rf / when logged as 'root' will delete everything ;-) For some filesystems, the rm keeps the content only delete the record.

Copy file(s) and/or directory(-ies) to different part of the filesystem or to different name

- For files

```
cp [OPTION]... [-T] SOURCE DEST
cp /my/file /another/file # DEST is a file
cp /my/file /another/directory # DEST is a directory
cp /my/file1 /my/file2 /another/directory # copy many file to destination directory
cp -i file1 file2 # interactive, if file2 already exists, it asks if to overwrite
```

- For directories

```
cp -r /my/dir1 /another/dir # if dir exists, it will contain dir1
 # otherwise dir is creared and will have the same content as dir1
# combination
cp -r /my/file /my/dir1 /dest/dir # dir has to exists
# if the DEST dir is the actual working directory (‘pwd‘) then
cp -r /my/file /my/dir1 .
cp -r /my/file /my/dir1 /my/dir2 # if dir2 does not exists, error is raised
```

- Does not show progress - use rsync for that purpose

Moves file(s) and/or directory(-ies) to different part of the filesystem or to different name

- For files

```
mv [OPTION]... SOURCE... DEST
mv /my/file /another/file # DEST is a file
mv /my/file /another/directory # DEST is a directory
mv /my/file1 /my/file2 /another/directory # move many file to destination directory
mv -i file1 file2 # interactive, if file2 already exists, it asks if to overwrite
```

- For directories

```
mv /my/dir1 /another/dir
# combination
mv /my/file /my/dir1 /dest/dir
# if the DEST dir is the actual working directory (‘pwd‘) then
mv /my/file /my/dir1 .
mv /my/file /my/dir1 /my/dir2 # if dir2 does not exists, it raises an error
```

Hard links

- hard links reference a physical(!!!) file location. Each file is actually a hardlink.
- ls -l 2nd column shows the number of links (at least 1!!!) (number of subdirs in case of dir)
- Links have actual file contents
- Removing any link, just reduces the link count, but doesn't affect other links.
- We cannot create a hard link for a directory to avoid recursive loops.
- If original file is removed then the link will still show the content of the file.
- Command to create a hard link is:

```
ln  [original filename] [link name]
ln  /my/file1 /other/file2
```

Soft links

- A soft link or symlink is similar to the file shortcut feature which is used in Windows Operating systems. Each soft points to the original file but not to the physical localtion of the data. As similar to hard links, any changes to the data in either file is reflected in the other. Soft links can be linked across different file systems, although if the original file is deleted or moved, the soft linked file will not work correctly (called hanging link).
- ls -l command shows all links with first column value 'l' and the link points to original file.
- Soft Link contains the path for original file and not the contents.
- Removing soft link doesn't affect anything but removing original file, the link becomes 'dangling' link which points to nonexistent file.
- A soft link can link to a directory.
- If you want to link files across the filesystems, you can only use soft links.

```
      ln -s [original filename] [link name]
      ln -s /my/file1 /other/file2 # link file to different filename
      ln -s /my/file /my/dir # link file to different directory
      ln -s /my/dir /my/other/dir # linking directories
ln -s /my/dir/or/files . # linking to the actual dir
```

Files

- A new file is created by many ways, some of them were seen on previous slides, others will be shown during next lessons
- Commands cp, mv, ln create new files (a symbolic link is a file too)
- Using touch - a new empty file is created:

```
touch myfile
```

- Redirecting the standard output (stdout) and standard error output (stderr) to a file (see next lesson)

```
command > /my/file
ls -l /etc > etc.content # the long listing of the
                         # /etc directory is saved in file etc.content
```

- Created by running programs (Text Editors, Photoeditors, Games etc.)

Directories

```
mkdir /my/dir # the dir /my MUST exist
mkdir -p /my/dir1/dir2/dir3/dir4 # if dir1 (and so on) does not exists,
                                 # it will create parent dirs (-p)
```

Wildcards

Standard wildcards (also known as globbing patterns) are used by various command-line utilities to work with multiple files, especially usefull with file/directory manipulation.

- ? (question mark) – this can represent any single character (i.e. only one character). If you specified something at the command line like "hd?" GNU/Linux would look for hda, hdb, hdc and every other letter/number between a-z, 0-9 etc.
- * (asterisk) – this can represent any (including zero) number of characters. If you specified a "cd*" it would use "cda", "cdrom", "cdrecord" and anything that starts with "cd" also including "cd" itself.
- [](square brackets) – specifies a range a[a-c]d = aad,abd, acd, same as a[abc]d, [0-9] all numbers between 0 and 9. [A-Z]. etc.
- { } (curly brackets) – terms are separated by commas and each term must be the name of something or a wildcard, {abc, def, ghi} mean one of the strings inside
- [!]– [!0km] = any other character than 0, k or m
- Wildcards can be combined

```
ls -l /etc/??? # list all files/dirs in /etc with three character long names
cp -r ~/* /tmp # copy all files/dirs from the home directory to /tmp
               # (~ is a shortcut for the home directory)
rm  /etc/[nm]* # removes all config files from /etc starting with n or m
cp /tmp/[ab]{*.txt,*.doc,*.pdf} ~ # copy or files starting with a or b with
                                   # txt, doc and pdf extension from /tmp directory to home
```

File Manipulation/Wildcards - Excercises

- Excercise 1: create directories ~/cviceni/{dnesni datum} and copy all files starting with b or c from /usr
- Excercise 2: make symlink of all file ending with "t" in ~/cviceni/{dnesni datum} in a new directory ~/cviceni/{dnesni datum}/t
- Excercise 3: move all files from ~/cviceni/{dnesni datum} to ~/tmp and check the content of dir ~/cviceni/{dnesni datum}/t
- Excercise 4: remove dir ~/cviceni/{dnesni datum} and everything from ~/tmp

File and directory (de)compression

File and directory (de)compression

Prupose: creating one file to replace many files and to reduce the size of files/directories Commands to be learnt: zip, tar, gzip, bzip2, 7z ...

- ZIP - similar to WinZIP

```
# Compression
zip [options] zipfile files_list
zip myfile.zip  my_files
zip -r student_home.zip /home/student
zip -P password zipfile files_list # zipping with "password"
# Decompression
unzip myfiles.zip
unzip -l myfiles.zip # shows the zip content without unzipping
```

- TAR - create a tar archive (originally tape archive = tar). Merges whole directories into one file called 'tarball'. No compression!!!

```
# "Compression" - no compression in fact
tar -cvf myfilesdirs.tar /my/dirs /and/files # make a tarball from dirs and files
                                             # -c means compress
# Uncompress - untar
tar -xvf myfilesdirs.tar # -x means eXtract
# does not compresses
tar -cvf downloads.tar Downloads/
ls -l downloads.tar
976005120
du -cb Downloads
975353493
```

Tar can be combined with compression to reduce the size of the tarball file. Using compression algorithms from GZIP and BZIP2 utilities

- TAR with GZIP

```
# Compression
tar -zcvf myfiles.tar.gz /my/files /and/my/dirs # the extension is .tar.gz or .tgz
# Decompression
tar -zxvf  myfiles.tar.gz
tar -tvf myfiles.tar.gz # view contents
```

- TAR with BZIP2

```
# Compression
tar -jcvf myfiles.tar.bz2 /my/files /and/my/dirs # the extension is .tar.bz2 or .tbz2
# Decompression
tar -jxvf  myfiles.tar.bz2
tar -tvf myfiles.tar.bz2 # view contents
```

- gzip and bzip2 can be used directly for files

```
# Compression
gzip /my/file # creates file.gz
bzip2 /my/file # creates file.bz2
# Decompression
gunzip file.gz
bunzip2 file.bz2
```

- 7z - From man pages "7-Zip is a file archiver with the highest compression ratio. Supports LZMA, LZMA2, XZ, ZIP, Zip64, CAB, RAR (if the non-free p7zip-rar package is installed), ARJ, GZIP, BZIP2, TAR, CPIO, RPM, ISO, most filesystem images and DEB formats. Compression ratio in the new 7z format is 30-50% better than ratio in ZIP format."

```
# Compression
7z a dir.7z  /my/dir
# Decompression
7z e dir.7z
```

- RAR and UNRAR

```
# Compression
rar a myfiles.rar /my/files /my/dirs
# Decompression
unrar x myfiles.rar
```

- XZ - general-purpose data compression tool with command line syntax similar to gzip and bzip2, contained by 7z

```
# Compression
xz /my/files
# Decompression
xz -dk files.xz
```

- Excercise 1: Archive all files and directories in your home folder starting with . (dot) with tar+gzip
- Excercise 2: make a tarball from /etc directory - will it work for all the files
- Excercise 3: Compare the sizes of gzip, bzip2, zip, 7z, rar and xz for this file http://meop3.troja.mff.cuni.cz:8010/linux/netcdf/PHA03_STS.2007-2011DJF.nc
- Excercise 4: How to use 7z with password? Is it safe?

Homework 2

1. Create the "homeworks/HW2/myetc" directory(-ies) in your home directory with one(!) command and copy all files from /etc ending with "f" and containing a dot ".", or starting with "u" and the second character being "p" to this newly created directory.
2. Compress this newly created directory with zip and uncompress it again into a NEW directory homework/HW2/myetc_tmp (see man unzip)
3. List all files in /usr/bin that does not(!) start with "c" or "d", but they can start with "cd" or "dc". The command should have only one argument, so a smart combination of wildcards is needed.
4. Link (soft/sym link) all files starting with "d" and the second letter NOT BEING "x" from both /usr/bin and /sbin to homework/bin (using one command)
5. What happens if you move a softlink that is created using absolute and relative path to a different directory. Will it point to the right file in both cases?
6. It was not mentioned on the lecture, but what happens to input file if gzip (or bzip2) is used and how to change this behavior? (try it out and then check the man pages)