

Introduction to the Linux OS

Peter Huszár

KFA: DEPARTMENT OF ATMOSPHERIC PHYSICS

Pavel Řezníček

ÚČJF: INSTITUTE OF PARTICLE AND NUCLEAR PHYSICS

December 5, 2024

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>



- 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/directory search

Search commands

Commands to search files and directories

- locate - search files based on a pre-built database by updatedb. Locate searches the whole directory tree but search in a database of files which is updated e.g. once-per-week.

```
locate name # locate all files with 'name' in their name ()
locate -c name # prints the number of found files
locate -e name # prints only those files which really exists in the moment of search.
```

- find - a powerful search engine for files and directories with the possibility to logically combine search criteria

```
# basic usage
find /my/dir "search criteria" # search files/directories in /my/dir (and deeper)
                                # based on the search criteria
find /my/dir -name '*.jpg'      # search based on name, files ending with jpg.
find /my/dir -type d -name 'a*' # search directories starting with "a" (or files only -type f)
find /my/dir -empty # find all empty files/directories
find /my/dir -perm 664 # find files/directories with permissions 664 (-perm u+rw)
find /my/dir -user student # find files/directories with "student" as owner
find /my/dir -size 5M # files larger than 5Megabytes
# you can combine search criteria
find /my/dir -size 5M -and -perm 664 -or -empty -and -not -name '*.jpg' # -or,-and,-not
```

Further actions on found items

```
find /my/dir "search criteria" -exec rm -rf {} \; # this will find files/dirs and applies
                                                # the command after the -exec for each file found
                                                # (one-by-one!!)
find /my/dir -iname "*dvorak*mp3" -exec mplayer {} \; # listen to all dvorak mp3-s, mplayer is started
                                                # separately for each file found
find /my/dir "search criteria" -exec command {} + # start the command only once for all files as parameters
find /my/dir -iname "*dvorak*mp3" -exec mplayer {} + # mplayer is started once and plays all files found
```

Search commands

Commands to search files and directories (cont'd)

- `whereis/which` - find the whole path to the called binary/command. Can be useful if one has multiple instalations of a program and the specific binary can be executed from different directory. In this case, it is good to know which one is executed.

```
whereis python3 # this will show the full path for the ls command
which python3 # shows exactly which binary is executed if more "python3"-s are installed
```

Pattern search in texts - regular expressions

Searching in texts - regular expressions

Finding parts of text according to a specific pattern

- `grep` - One of the most useful and versatile commands in a Linux terminal environment is the "`grep`" command. The name "`grep`" stands for "global regular expression print". This means that `grep` can be used to see if the input it receives matches a specified pattern.

```
cat /my/input/file(s) | grep "pattern" # this will print all lines with the word 'pattern'
# This is of course equivalent to grep "pattern" /my/input/file(s)
cat /my/input/file(s) | grep --color "pattern" # occurrences are 'colored'
cat /my/input/file(s) | grep -o "pattern" # print only matches (-c will print the count)
# useful options
# -i - case insensitive, -v invert search; -l -- prints only files with matches
# -L - print files without match
```

- However, the real power of `grep` comes with the introduction of regular expressions!!!

Regular expressions - Regexp

Sequence of characters that define a search pattern

We see that with `grep`, we can search for some **characters, words**, but what about more complicated patterns???

For example:

- words that start to/end/contain a specific set of letters
- words starting with capitals or having certain number of characters
- email addresses
- IP address
- special numbers (e.g. real numbers)
- specific parts of computer code
- webpage address ...
- The above examples cannot be searched with simple `grep "word" /my/file`
- **The solution is "regular expressions"**
- A quick example: regular expression and search for a valid email address within a textfile

```
grep -E ^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$ /my/file
```

Regular expressions - Regexp

Sequence of characters that define a search pattern

Single character

pattern	meaning	example regexp	example matches
.	any (!!!) single character	a.c	aac akc aZc a?c a+c ...
\	turns off special character	\.	. (dot)
[]	any of the characters in brackets	[+mFf2019!]	any of m,F,f,2,1,0,9,+,!]
-	any character within the range	[a-zA-Z3-6]	any of a-z, A-Z, 3-6
[^]	negation of the above	[^mFf2019] [^A-Z]	any except mFf2019 any character except capital letters

Quantifiers/repetition

?	occurs 0x or 1x	ab?c 0[0-9]?1	ac, abc 01, 011, 021, 031 ..
*	occurs arbitrary times (0-inf)	ab*c 0[0-9]*1 x.*x	ac, abbc, abbbbbbbc 01, 091, 011535451 .. "xx", "x13 +-*x", "x 34-+ x 123 x"
+	occurs at least once	ab+c 0[0-9]+1 x.+x	abbc, abbbbbbbc 091, 011535451 .. "x13 +-*x", "x 34-+ x 123 x"
{n}	occurs n-times	ab{2}c 0[0-9]{2}1 x.{2}x	abbc 0991, 0181 .. "x13x", "x zx", "xxxx"
{n,m}	occurs n-m times	ab{2,4}c 0[0-9]{2,4}1 x.{2,4}x x.{2,}x	abbc, abbbc 0991, 018231 "x13x", "x zx", "xxxxx" two or more occurrences...

Regular expressions - Regexp (cont'd)

Sequence of characters that define a search pattern

Anchor characters

pattern	meaning	example regexp	example matches
<code>\<</code>	beginning of word	<code>\<[A-Z][a-z]+</code>	Paul, Judit, but not 09Tom
<code>\></code>	end of word	<code>a\></code>	all words ending on "a"
<code>^</code>	beginning of the line	<code>^[0-9].*</code>	all lines starting with a digit
<code>\$</code>	end of the line	<code>*.[0-9]\$</code>	all lines ending with a digit

- Selection
 - `(r1|r2|r3)` – any of the regex `r1,2` or `3`
 - E.g.: `([0-9] | [a-b] | xyz)` – 0,8,a,xyz
- Grouping
 - `(r1)+` – group with `regex1` with at least 1 occurrence
 - E.g.: `((r1)+(r2){2}){3}` – grouping regexps
 - E.g.: `([A-Z](\ . | [a-z]+)){2,}` – (maybe) abbreviated names
- Remembering
 - `(r1)r2\1` – the match for the first regex will be saved and revoced by `\1`
 - E.g.: `([a-z])([a-z])([a-z])\3\2\1` – this finds all palindroms of length 6 (abccba, xzyzyx)

- Find all users with names starting with "r" (/etc/passwd)
- Find all latitude/longitude definition in AirBase-CZ-v8-stations.csv (regex for real numbers)
- Find all "acid" names in 'chemicals'
- In further, just use the echo "any_string the-test-string any_string2" — grep -color -Eo 'regex' to test, if the regex is correct
- Construct a regex for valid date in YYYYMMDD format (expect Feb has 28 days)
- Find regex for email address
- Find regex for whole sentences (Starts with capital letter, ends with one of '?!')