Introduction to the Linux OS

Peter Huszár
KFA: Department of Atmospheric Physics

Pavel Řezníček
ÚČJF: Institute of particle and nuclear physics

December 2, 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

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]`	any except mFf2019
		`[^A-Z]`	any character except capital letters
Quantificators/repetition			
`?`	occurs 0x or 1x	`ab?c`	ac, abc
		`0[0-9]?1`	01, 011, 021, 031 ..
`*`	occurs arbitrary times (0-inf)	`ab*c`	ac, abbc, abbbbbbbc
		`0[0-9]*1`	01, 091, 011535451 ..
		`x.*x`	"xx", "x13 +-*x", "x 34-+ x 123 x"
`+`	occurs at least once	`ab+c`	abbc, abbbbbbbc
		`0[0-9]+1`	091, 011535451 ..
		`x.+x`	"x13 +-*x", "x 34-+ x 123 x"
`{n}`	occurs n-times	`ab{2}c`	abbc
		`0[0-9]{2}1`	0991, 0181 ..
		`x.{2}x`	"x13x", "x zx", "xxxx"
`{n,m}`	occurs n-m times	`ab{2,4}c`	abbc, abbbc
		`0[0-9]{2,4}1`	0991, 018231
		`x.{2,4}x`	"x13x", "x zx", "xxxxx"
		`x.{2,}x`	two or more occurences...

KFA MFF UK

ÚČJF MFF UK

Anchor characters

pattern	meaning	example regexp	example matches
`\<`	beginning of word	`\<[A-Z][a-z]+`	Paul, Judit, but not 09Tom
`\>`	end of word	`a\>`	all words ending on "a"
`^`	beginning of the line	`^[0-9].*`	all lines starting with a digit
`$`	end of the line	`*.[0-9]$`	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 occurence
  - E.g.: `((r1)+(r2){2}){3}` – grouping regexps
  - E.g.: `([A-Z](\.|[a-z]+)){2,}` – (maybe) abbrevated 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, xyzzyx)

Regular expressions - Excercises

- 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 '.?!')

sed - stream editor

- SED is the ultimate stream editor. It can substitue strings, add parts of text and delete part of text according to rules given by the users.
- SED can be used in two basic ways:

```
# 1
cat /my/input/file.txt | sed -e  'sed-script' # this generate output on stdout
# 2
sed -i 'sed-script' /my/input/file.txt # this will change the input file inline.
```

- The most used functionality of `sed` is string substitution

```
# on each line finds the first match for regex and replaces with 'string'
cat /my/file.txt | sed -e 's/regex/string/'
      # "/" here serves as a command delimiter
      # one can use different one too
cat /my/file.txt | sed -e 's:regex:string:'
cat /my/file.txt | sed -e 's/regex/string/2' # replace the 2nd occurence on line
cat /my/file.txt | sed -e 's/regex/string/g' # replace all occurence on the line
```

- In subtitution, one can remember the substituted string

```
echo "123 abc" | sed 's/[0-9][0-9]*/& &/'   # & will stand for the match (123)
                                             # -> 123 123 abc
sed -re 's/([a-z]*) ([a-z]*)/\2 \1/' # \1 and \2 memorize the searched string
                                     # and replace them with \2 \1
                                     # -r extended regex !!!
```

- By default, sed prints all lines in the output, not only those where the replacement occured

```
cat /my/file.txt | sed -n -re 's/regex/string/g p'
# p - print, sed will print the matched lines (not only replacing)
# -n, suppress printing lines, overwritten by "p" (effective for matched lines)
cat /my/file.txt | sed  -re 's/regex/string/g p' # will print matched lines 2x
```

- So far we have learn substitution and printing to all lines
- However, in sed, you can specify, for which line (or line range) to do it. We call it restriction.

- Specifying lines and line range - *restrictions*. The sed command than look like *sed restriction command*

```
cat /my/file.txt | sed -e '5 s/[0-9]/x/' # do the substitution on the 5th
cat /my/file.txt | sed -e '$ s/[0-9]/x/' # do the substitution on the last line ($)
...  | sed -e '10,30 p' # print lines 10-30 twice (with -n only those lines)
...  | sed -e '10,$ any-sed-command' # perform the command from the 10line till end
...  | sed -e '/pattern/ any-sed-command' #  perform the command on lines that match pattern
...  | sed -e '10,/pattern/ any-sed-command' # from line 10 to the 1st matched line (including)
...  | sed -e '/pattern1/,/pattern2/ any-sed-command' #
              # pattern1 switches the 'any-sed-command' on which is switched off by pattern2
```

- i - insert, before the restricted lines (if no restriction is present insert before each line)
- a - append, after the restricted lines (if no restriction is present appned after each line)

```
cat /my/file.txt | sed -e '10,30 i ???' # insert ??? before lines 10-30
cat /my/file.txt | sed -e '/pattern/ a ###' # append ### after lines matching the 'pattern'
```

- A superb SED howto: http://www.grymoire.com/Unix/Sed.html#uh-0