

Faculty of Mathematics and Physics CHARLES UNIVERSITY

**NOFY077** 



《口》 《國》 《臣》 《臣》

æ

# Introduction to the Linux OS

#### Peter Huszár

KFA: DEPARTMENT OF ATMOSPHERIC PHYSICS

### Pavel Řezníček

 $\acute{\mathrm{U}}\check{\mathrm{C}}\mathrm{J}\mathrm{F}\mathrm{:}$  Institute of particle and nuclear physics

November 15, 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

### Syllabus

- UNIX systems, history, installation, basic applications
- Structure of the Linux OS, file systems, hierarchy of the file system
- Ommand line, shells, remote access (ssh, ftp)
- Processes and their administration, basic system commands, packages, printing
- Users, file and directory permissions
- Work with files and directories, file compression, links, partition
- Text-file processing commands, redirection, pipeline
- Regular expressions
- Ommand line based text editors
- User and system variables, output processing
- Oscripts: basic construction, conditionals, loops, functions, automation
- Wetworking, server-client services: http, (s)ftp, scp, ssh, sshfs, nfs
- Programming in Linux (examples of Fortran, C/C++, Python), version control systems, documents in Latex



# Text manipulation



MFF UK

Huszár, Řezníček

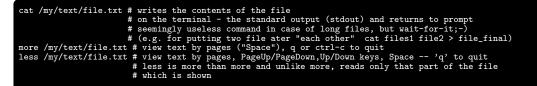


ÚČJF MFF UK

#### 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!)



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

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

MFF UK

### 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)

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

• 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



**MFF UK** 

### 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

 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

MFF UK

## Input/output redirection and command chaining



Huszár, Řezníček

Linux: Introduction NOFY077

November 15, 2024 8 / 13

## Input/output redirection

standard output, standard error output, standard input

When a linux command is launched, three data streams are relevant/created: stdout, stderr, stdin, the Standard output (what we saw so far in the terminal, the standard error, what we also saw in the terminal if an error occured and standard input what is read in.

• Redirect the stdout and stderr to a file

#### • Redirecting from the standard input

MEET

## Linux command chaining

How to write a sequence of commands - oneliners

• ; (semicolon) - execute commands in chain independently to the previous

command1 ; command2 ; command3 # first, command1 is executed and then command2, ...

 && (double AND) - logical "and" between commands. From now on, we will regard commands as logical expression which are evaluated TRUE, if executed without errors (i.e. is successfull), or FALSE, when executed with error.

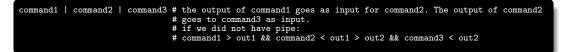
command1 && command2 && command3 # if command1 is successfull (TRUE), command2 will be executed # if command2 is successfull too, command3 is executed # if e.g. command1 is FALSE, than nor command2 neither command3 is executed # Why? False && anything && anything will be always false, and the Linux # command line environment tries to evaluated all the commands as True # but here it "gives up" after the first FALSE.

 || (double vertical bar) - logical "or" between commands. Commands in a || sequence will be executed until the last successfull

How to write a sequence of commands

• & (single AND) - execute the command before in the background

- Note: ctrl-z stops executing the running command, bg resumes it and sends it to background, i.e. command & == command + ctrl-z + bg
- command == ctrl-z + fg
- | the pipe, one of the most important way combining commands. It is pipe because it serves as a "medium" for data stream. In particular, it takes the output from the command before the pipe and provides it as input for the command that follows the pipe.



• By default, stderr is not "piped" and is written out to the terminal.



More advanced sequencing of commands

• () (paranthesis) – execute the command(s) in the paranthesis in a separate sub-shell. Way how to combine all the methods for command chaining.

( command1 && command2 ) || ( command3 || command4 )
# If command1 is TRUE and command2 is TRUE, the other commands are not executed
# if command1 is TRUE but command2 is FALSE, the command line tries to execute one of commands3/4
# if command1 is FALSE, does not execute command2, and tries to execute one of command3/4
# More complex, but in theory working construction as an example:
(( command1 && command2 ) &> log\_c1+2.log || ( command3 || ! command4 2> error\_c4.log )& )

• { }(braces) – Placing a list of commands between curly braces causes the list to be executed in the current shell context. No subshell is created. The semicolon (or newline) following list is required.

{ command1 && command2 ; } || command3



## Excercises - Linux command chaining

Redirection and pipe-ing

- Excercise 1: How to redirect the stdout to a file, the stderr to a different file and send the stderr to the terminal too (hint: 2>&1 > /dev/null)
- Excercise 2: Write out the second largest file from directory /usr/bin (hint: ls, head or tail. and pipe)
- Excercise 3: Count the number of letters in the name of the first user sorted alphabetically (hint: cut, sort, tail or head, wc)
- Excercise 4: What will be the output of: true || echo aaa && echo bbb
- Excercise 5: what will be the standard output of the following crazy command and the content of the out.log file? Without tying to write to the prompt...
- (! date > /dev/null && cat /etc/passwd ) || ( ( ! ls -l ~ || head -n 1 /etc/passwd | cut -d: -f1 ) ... ... ||( df -h &> df.log) ) | tail -n 1 > out.log

