`sudo -su -`
`sudo -s`
`su -`
`cd /` - root

`uptime`
`hostname`
`uname` --> `uname -a`
`ps ... ps -fu .... ps -f ... ps -ef .... ps awx ... ps u`
`top`
`kill -PID`
`kill -9` (No Question)
`ls -l .... ls -ltr`
`whoami`
`wim` -- > exit :wq!
`mkdir`
`| pipe`
`ls -ltr | more` (lapozás - space)
d -directory, l -link, - -file
rwx - read, write, execute
ugo - user, group, other

---

chown "new-user" file
chgrp "new-group" file
`chmod g+w file` (group enable write permission)
`chmod a+r file` (all enable read permission)
`chmod u+w file` (user enable write permission)

`rm` - remove

text `echo "something" > filename`
`echo "something" >> filename` - append
`touch` - create a file

`whatis command`
`command --help`
`command man`

Maintenance Commands

`cp` - copy
`rm` - remove
`mv` - move
`mkdir` - make directory
`rmdir` or `rm -r` ---- remove directory

`chgrp` -
`chown` -
`rm - Rf` - force full remove
`chown root:root file`

Filters/Text Processors Commands

- `out`
- `awk`
- `grep`
- `sort`
- `uniq`
- `wc` (word count)

`cut -c1 filename` (give you back first letters)
`awk` separate each columns
`awk '(print $1)' filename` (first column)
`grep` ---- search grep mit miben
`sort` - sorba rendezés
`sort filename`
`sort -r` fordított sorrend
`uniq` - removes all
`sort | uniq` együtt
`wc` - word count
`wc filename` (-l, lines)

Finding System Informations

- `cat`
- `uname -a`
- `dmidecode`

User Account Management

1. `useradd`
2. `groupadd`
3. `userdel`
4. `groupdel`
5. `usermod`

Switch Users and sudo access

- `su -username`
- `sudo command`
- `visudo`

`ifconfig`
`dmidecode`
`fdisk -l`

System Utility Commands

1. date
2. uptime
3. hostname
4. uname
5. which
6. cal
7. bc

`main hier` (könyvtárszerkezet)
`shutdown -t 300` (300sec)
`shutdown -21:00` (konkrét időpontban)

`wget link`

CTRL + C prompt back

Könyvtárszerkezet

- /BOOT Contains file that is used by the boot loader (grub.cfg)
- /ROOT Root user home directory - it is not same as /
- /DEV System devices (disk, cdrom, speakers etc.)
- /ETC Configuration files
- /BIN --/USR/BIN Everyday user commands
- /SBIN -- /USR/SBIN System, filesystem commands
- /OPT Optional add-on application (NOT part of OS apps)
- /PROC Running processes (Only exist in memory)
- /LIB -- /USR/LIB C programming library files needed by commands and apps.
- /TMP Directory for temporary files
- /HOME Directory for users
- /VAR System logs
- /RUN System deamons that start very early to store temporary rundtime files like PID files
- /MNT To mount external filesystem
- /MEDIA For CD-rom mounts

`cd` - change directory
`pwd` - print working directory
`ls` - listing
`find . -name filename`

`locate filename`
`updatedb`

`passwd userid`
Old password: ----
New password: ----

Wildcards

- * zero or more characters
- ? single characters
- [] range of characters

Create 9 file:
`touch filename{1..9}`
`touch Csaba{1..9}`

List filename file
`ls -l Csaba*`
Több file törlése
`rm Csaba*`

\ = slash (escape character)
^ = caret (the beginning of the line)
$ = dollar sign (the end of the line)

Soft and Hardlink

- inode (pointer or number of a file on the hard disk)
- soft link (link will be remover if file is removed)
- hard link (deleting, renaming or moving the original file will not affect the hard link)

`ln -s file` -- softlink
`ln new file original file`

Commands Syntax

Command options and arguments
Options:

- Modify the way that a commands works
- hyphen (kötőjel)
- dash (gondolatjel - followed by a single letter.)

Some commands accept multiple options.

Arguments:

- Most commands are used together wieht one or more arguments.
- Some commands assume a default argument if none is supplied.
- Arguments are optional for some commands and required by others.

`ls -l bart` (ls - command, l - options, bart - argument)

File Permission

3 type of permission r-w-x

Each permission can be controlled at 3 levels

- u (user)
- g (group)
- o (other)

Command : chmod
`chmod g-w filename` - (remove group write permission)
`chmod a-r filename` - (a -- every level remove read permission)

```
setfacl - m u:user:rwx 'path'
setfacl - m g:group:rw 'path'
setfacl - Rm "entry" 'path'
setfacl - x u:user 'path'
setfacl - b 'path'
```

Help Commands

- Whatis command
- command --help
- mand command

TAB completion and Up arrow

Adding text to Files (Redirects)

- `vi` (vi editor)
- Redirect command output > or >>
- echo > or >>

`cat` - what inside in the file

Standard Output to a File (tee)

```
echo "szöveg" | tee filename
```

append
`echo "szöveg" | tee -a filename`

How many characters --- `wc -c`
word -- `wc -w`

`ls -l | tee listdir` same `cat listdir`

Pipes

`ls -ltr | more`
`ls -l | tail -1` - last line

File Display Commands

- `cat`
- `more`
- `less`
- `head -2 filename` - first 2 line
- `tail -2 filename` - last 2 line

Filter/Text Processor Commands

- `cut`
- `awk`
- `grep` and `egrep`
- `sort`
- `uniq`
- `wc` (word count)

---

cut commands

`cut -c1 filename` - first character
`cut -c1,2,3 filename` - picked characters
`cut -c1-3 filename` - range of characters
`cut -b1-3 filename` - by bite size

---

awk commands

`awk '{print $1}' filename` - print 1st field from a file

`ls -l | awk '{print $1, $3}'`
`ls -l | awk '{print $NF}' filename` - last column
`awk '/jerry/ {print}' filename` - search command

Replace Word

`echo "Hello Tom" | awk '{$2="Adam"; print $0}'`

Get line that have more than 15 byte size

`awk 'length($0) > 15' filename`

---

grep and egrep

`grep --version` or `grep --help`
`grep keyword file` - search for a keyword from a file

> > `grep Seinfeld seinfeld-characters` - example

`grep -c keyword file` - search for a keyword and count

> > `grep -c Seinfeld seinfeld-characters` - example

`grep -i KEYword file` - search for a keyword ignore case-sensitive

> > `grep -i seinfeld seinfeld-characters` - example

`grep -n keyword file` - Display the matched lines and their line numbers

`grep -v keyword file` - Display everything but keyword

> > `grep -vi seinfeld seinfeld-characters` - example

`grep keyword file | awk '{print $1}'` - Search for a keyword and then only give the 1st field

`ls -l | grep Desktop` - Search for a keyword and then only give the 1st field

`egrep -i "keyword|keyword2" file` - Search for 2 keyword

> > `egrep -i "Seinfeld|Costanza" seinfeld-characters` - example

---

sort/ uniq - Text processors commands

Sort command sorts in alphabetical order.
Uniq command filters out the repeated or duplicate lines.

`sort --version` or `sort --help` - Check version or help
`sort file` - Sorts file in alphabetical order
`sort -r file` - Sorts in reverse alphabetical order
`sort -k2 file` - Sort by field number
`ls -l | sort file` - List sort by alphabetical order

`uniq file` - Removes duplicates
`sort file | uniq` - Always `sort` before using `uniq` their line numbers
`sort file | uniq -c` - Sort first then uniq and list count
`sort file | uniq -d` - Only show repeated line

\`wc\` - Text processors commands

The command reads either standard input or a list of files and generates:
newline count, word count, and byte count.

`wc file` - Check file line count, word count, and byte count
`wc -l file` - Get the number of lines in a file
`wc -w file` - Get the number of words in a file
`wc -c file` - Get the number of byte in a file

`ls -l | wc -l` - Number of files
`ls -l | grep drw` - Get the Directories
`ls -l | grep drw | wc -l` - Get the line of Directories
`grep keyword | wc -l` - Number of keywords line

Compare Files

- `diff` - Line by line
- `cmp` - Byte by byte

Compress and uncompress file

- `tar`
- `gzip`
- `gzip - d` or `gunzip`

`tar cvf file.tar file` - Compress
`tar xvf file.tar` - Uncompress
`tar czvf`
`tar xzvf`

`gzip file.tar`
`gzip -d file.tar.gz`

`rm -rf`

Truncate File Size

The linux `truncate` command is often used to shring or extend the size of a file to the specified size.

`truncate -s 10 filename`

Combining and Splitting Files

- Multiple files can be combined into one and
- One file can be split into multiple files

> > `cat file1 file2 file3 > file 4split file4`

> > example: `split -l 300 file.txt childfile`

Split file.txt into 300 lines per file and output childfileaa, childfileab, childfileac

`cat filename | wc -l` - how many lines have

Linux file editor

- A text editor is a program which enables you to create and manipulate data (text) in a Linux file.
- There are several standard text editors available on most Linux sytems: ------- vi - Visual editor ------- ed - Standard line editor ------- ex - Extended line editor ------- emacs - A full screen editor ------- pico - Beginner's editor ------- vim - Advance version of vi

Introduction to vi editor

- **vi supplies commands for:**
  - inserting and deleting text
  - replacing text
  - moving around the file
  - finding and substitutings strings
  - cutting and pasting text
- **Most common keys:**
  - i - insert
  - Esc - Escape out of any mode
  - r - replace
  - d - delete
  - :q! - quit without saving
  - :wq! - quit and save

`sed` command

- Replace a string in a file with a newstring
- Find and delete a line
- Remove empty lines
- Remove the first or n lines in a file
- To replace tabs with spaces
- Show defined lines from a file
- Substitute within vi editor
- And much more ....

example:

- `sed 's/Kenny/Lenny/g' filename` - only change display not a file
- `sed -i 's/Kenny/Lenny/g' filename` - change file
- `sed 's/Costanza// filename` - only remove on the screen

- `sed -i 's/Costanza// filename` - remove in the file
- `sed '/Seinfeld/d filename` - delete line where is e.g. Seinfeld
- `sed '/^$/d' filename` - delete empty lines only a screen
- `sed -i '/^$/d' filename` - delete empty lines in the file
- `sed '1d' filename` - delete the first line only a screen
- `sed -i '1d' filename` - delete the first line in the file
- `sed '1,2d' filename` - delete the first 2 line on the screeen
- `sed -i '1,2d' filename` - delete the first 2 line in the file
- `sed 's/\t/ /g' filename` - replace tab to space on the screen
- `sed -i 's/\t/ /g' filename` - replace tab to space in the file
- `sed 's/ /\t/g' filename` - replace space to tab on the screen
- `sed -i 's/ /\t/g' filename` - replace space to tab in the file
- `sed -n 12,18p filename` - show defined lines from a file
- `sed 12,18d filename` - shows outside the specified lines
- `sed G filename` - put under each line an empty line on the screen
- `sed -i G filename` - put under each line an empty line in the file

User Account Management

commands:

- `useradd`
- `groupadd`
- `userdel`
- `groupdel`
- `usermod`

files:

- /etc/passwd
- /etc/group
- /etc/shadow

Example: `useradd -m superheroes -s /bin/bash -c "user description" -m -d /home/spiderman spiderman`

`useradd -m newusername`
`useradd - g newusername` - add new user a group
`userpasswd newusername`
`userdel newusername`

userupdate: `sudo usermod -a -G sudo newusername`

Switch Users and Sudo Access

Commands

- `su - username`

- `sudo command`
- `visudo`

File

- /etc/sudoers

Monitor Users

- `who`
- `last`
- `w`
- `finger`
- `id.`

`last | awk '{print $1}' | sort | uniq` - only first column without duplicate

Talking to Users

- `users`
- `wall`
- `write`

Linux Account Authentication

- Types of Accounts
  - Local accounts
  - Domain/Directory accounts

System Utility Commands

- `date`
- `uptime`
- `hostname`
- `uname`
- `which`
- `cal`
- `bc`

Processes and Jobs

- Application = Service
- Script
- Process
- Daemon
- Threads

- Job

Process/Services Commands

- `systemctl` or `service`
- `ps`
- `top`
- `kill`
- `crontab`
- `at`

Process Management

- Background = CTRL-z, jobs and `bg`
- Foreground = `fg`
- Run process even after exit = `nohup process &`
  - OR = `nohup porcess > /dev/null 2>&1 &`
- Kill a process by name = `pkill`
- Process priority = `nice` (e.g. `nice -n 5 process`)
- Process monitoring = `top`
- List process = `ps`

System Monitoring

- `top`
- `df`
- `dmesg`
- `iostat 1`
- `netstat`
- `free`
- `cat /proc/cpuinfo`
- `cat /proc/meminfo`

Log Monitoring

Log Directory = /var/log

- boot
- chronyd = NTP
- cron
- maillog
- secure
- messages
- httpd

System Maintenance Commands

- shutdown
- init 0-6
- reboot
- halt

Changing System Hostname

- `hostnamectl - set-hostname newhostname`

Finding System Information

- `cat` /etc/redhat-release
- `uname -a`
- `dmidecode`

Terminal Control Keys

- CTRL-u - erase everything you've typed on the command line
- CTRL-c - stop/kill a command
- CTRL-z - suspend a command
- CTRL-d - exit from an interactive program (signals end of data)

Terminal Commands

- `clear` - clear your screen
- `exit` - exit out of the shell, terminal or a user session
- `script` - The script command stores terminal activities in a log file that can be named by a user, when a name is not provided by a user, the default filename, typescript is used

---

SOS report

- What is SOS report?
  - Collect and package diagnostic and support data
- Package name
  - `sos-version`
- Command
  - `sos report`

Environment variables

What are environment variables?

- An environment variable is a dynamic-named value that can effect the way running processes will behave on a computer. They are part of the environment in which a process runs.
- In simple words: set of defined rules and values to build an environment

To view all environment variables

- `printevn` OR `env`

To view ONE environment variable

- echo $SHELL

To set the environment variables

- `export TEST=1`
- `echo $TEST`

To set environment variable permanently

- `vi .bashrc`
- `TEST='123'`
- `export TEST`

To set global environment permanently

- `vi /etc/profile` OR `vi /etc/bashrc`
- `TEST='123'`
- `export TEST`

Linux kernel

What is a Kernel?

- Interface between hardware and software

Introduction to Shell

What is Shell?

- Its like a container
- Interface between Users and Kernel/OS
- CLI is a Shell

Find your Shell

- `echo $0`
- Available Shells `cat /etc/shells`
- Your Shells? /etc/passwd

Types of Linux Shells

- Gnome
- KDE
- sh
- bash
- csh and tcsh
- ksh

cat/etc/shells

Shell Scripting

What is a Shell script? A shell script is an executable file containing multiple shell commands that are executed sequentially. The file can contain: - Shell (#! /bin/bash) - Comments (# comments) - Commands (echo,cp,grep etc.) - Statements (if,while,for etc.)

Shell script should have executable permission (e.g -rwx r-x r-x)

Shell script has to be called from absolute path (e.g /home/userdir/script.bash)

If called from current location then ./script.bash

Basic scripts/Shell scripts

- Output to screen using "echo"
- Creating tasks
  - Telling your id, current location, your files/directories, system info
  - Creating files or directories
  - Output to a file ">"
- Filters/Text processors through scripts (cut,awk,grep,sort,uniq,wc)

Input and Output

Create script to take input from the user - `read` - `echo`

---

if-then scripts

- If then statement
  - **If this happens = do this**
  - **Otherwise = do that**

For Loop Scripts

- For loops

- **Keep running until specified number of variable**
- **e.g: variable=10 then run the script 10 times** OR
- **variable=green,blue,red (then run the script 3 times for each colors)**

#!/bin/bash

# For loop to create 5 files named 1-5

for i in {1..5} do touch $i done

#!/bin/bash

# Example of defining variables

a=Csaba b=Bajzáth c="Linux class"

echo "My first name is $a"
echo "My surname is $b"
echo "My class name is $c"

#!/bin/bash

# Simple for loop output

for i in 1 2 3 4 5 do echo "Welcome $i times" done

#!/bin/bash

# Check the variable

count=100 if [$count -eq 100] then echo Count is 100 else echo Count is not 100 fi

---

#!/bin/bash # Author # Date # Desc

echo Hello, my name Csaba Bajzáth echo echo What is your name? read namecontainer echo echo Hello $namecontainer echo

---

#!/bin/bash

# List all users one by one from /etc/passwd file

i=1 for username in `` `awk -F: '{print $1}' /etc/passwd` `` do echo "Username $((i++)) : $username" done

---

#!/bin/bash

# Specify days in for loop

i=1 for day in Mon Tue Wed Thu Fri do echo "weekday $((i++)) : $day" done

---

#!/bin/bash

# Check if a variable value is met

a=`date | awk '{print $1}'`

if ["$a" == Mon]

```
then
echo Today is $a
else
echo Today is not Monday
```

fi

#!/bin/bash

# Check if a file exist

clear if [-e /home/lin6echo/error.txt]

```
    then
    echo "File exist"
    else
    echo "File does not exist"
```

fi

---

do-while scripts

do while

- The while statement continually executes a block of statements while a particular condition is true or met
- e.g: Run script until 2pm

  while [condition] do

  ```
          command1
          command2
          commandN
  ```

done

#!/bin/bash

# Script to run for a number of seconds

count=0 num=10 while [$count -lt 10] do

```
echo
echo $num seconds left to stop this process $1
echo
sleep 1
```

num=`expr $num - 1` count=`expr $count + 1` done echo echo $1 process is stopped!!! echo

---

```
#!/bin/bash

# Script to run for a number of times

c=1
while [ $c -le 5 ]
do
        echo "Welcome $c times"
        (( c++ ))
done
```

97