Computer Networks - laboratory exercise:

UNIX/Linux operating system basics

Basic commands cheat sheet

symbol or command	meaning
	using the terminal
`man`	**man**ual pages for a command, e.g.: • `man pwd` – display help for command pwd
`clear`	**clear** the terminal screen
`Ctrl+L`	clear the terminal screen
[tab key]	auto-completion of the name that you started to enter
*	wildcard matching zero or more characters (most commands allow using wildcards in their arguments), e.g.: • `file*` – all of these are a match: *file, file.txt, file1, file_ABC.jpg*, and so on • `m*e` – all of these are a match: *me, mae, mbe, mce, mAe, m5e, mike, mouse, my_program.exe*, and so on
?	wildcard matching exactly one character (most commands allow using wildcards in their arguments), e.g.: • `file?` – all of these are a match: *file1, file2, filea, fileA*, and so on • `m?e` – all of these are a match: *mae, mbe, mAe, m1e, m5e, m.e,* and so on
	directories
`.`	current directory (working directory)
`..`	parent directory of the current directory
`/`	root directory in the system; also, separator in paths
`~`	home directory of the current user (usually in the form /home/name_of_user)
`pwd`	**p**rint **w**orking **d**irectory (display the path to the current directory)
`cd`	**c**hange **d**irectory (change working directory to some other one), e.g.: • `cd ~` – go to your home directory • `cd ..` – go to the parent directory of the current directory • `cd dir` – go to directory dir
`ls`	**l**i**s**t the contents of a directory, e.g.: • `ls` – display the content (files and subfolders) of the current directory • `ls dir` – display the content of directory dir
`ls -l`	**l**i**s**t directory contents using a **l**ong listing format (it shows, e.g. file permissions, owner, size in bytes, modification date)
`mkdir`	**m**a**k**e (create) a **dir**ectory, e.g.: • `mkdir my_directory`
`rmdir`	**r**e**m**ove an empty **dir**ectory, e.g.: • `rmdir my_directory`

files	
`touch`	create an empty file (or change modification date of existing file), e.g.: • `touch file1.txt` – create a file with .txt extension • `touch my_file` – create a file with no extension
`cp`	**co**py a file, e.g.: • `cp file1 file2` – copy file1 to a file called file2 (both in current directory) • `cp dir1/file dir2/copied_file` – copy a file from directory dir1 to directory dir2 and give a new name to the copied file • `cp dir1/file dir2` – copy a file from directory dir1 to directory dir2 and keep the same name of the file
`cp -r`	**co**py files **r**ecursively (including directories), e.g. • `cp -r dir1 dir2` – copy directory dir1 with all its files and subfolders to directory dir2
`mv`	**mo**ve a file to different location or rename it, e.g.: • `mv file1 shopping_list.txt` – rename file1 to shopping_list.txt • `mv file1 dir1` – move file1 from current directory to directory dir1 and keep the name of the file unchanged • `mv file1 dir1/new_file` – move file1 from current directory to directory dir1 and change the name of the file at the same time
`rm`	**rm**ove a file, e.g.: • `rm file1`
`rm -r`	**rm**ove files **r**ecursively (remove nonempty directories), e.g.: • `rm -r my_dir_with_stuff_inside`
`chown`	**ch**ange file **own**er and/or group
`chgrp`	**ch**ange **gr**ou**p** ownership of a file
`chmod`	**ch**ange file **mod**e bits (change file permissions) – more on permissions under the link "Linux: rwx permissions" provided on the course webpage; e.g.: • `chmod 777 file` – everyone may read, write or execute the file (please first consider if this is a safe setting) • `chmod 700 file` – the owner may read, write or execute the file; other user cannot do any of these operations
`find`	search for files matching the given criteria in a directory tree – more details under the link "Linux: 'find' command" provided on the course webpage; e.g.: • `find dir` – like a recursive version of `ls`: search for all files and subdirectories (and their files and subdirectories, and so on) inside directory dir • `find dir -name "test.txt"` – search for all files called test.txt in all subdirectories of dir • `find dir -name "test*"` – search for all files whose names begin with "test" in all subdirectories of dir (so, all of these match: *test*, *test1*, *test24.txt*, *testing_my_program.cpp* and so on) • `find dir -iname "*.txt"` – search for all files with .txt extension in all subdirectories of dir, but **i**gnore the case (so, all of these match: *.txt*, *.TXT*, *.TxT*, *.txT* and so on)

links and aliases	
ln	make hard **lin**ks between files – more about hard links under the link "Linux: hard and soft links" provided on the course webpage; e.g.: • `ln file1 my_link` – creates a hard link called my_link which points to file file1
ln -s	make **s**ymbolic (soft) **lin**ks between files – more about soft links under the link "Linux: hard and soft links" provided on the course webpage; e.g.: • `ln -s file1 my_link` – creates a symbolic link called my_link which points to file file1
alias	create an **alias** (alternative name) for a command, e.g.: • `alias cdd='cd ~/Desktop'` – after that, you can execute a new "command" `cdd` instead of `cd ~/Desktop` to go to the desktop of the current user • `alias` – display all existing aliases • `alias cdd` – display the command that has alias cdd
unalias	remove an alias of a command, e.g. • `unalias cdd` – after that, cdd will no longer work
text printing (displaying) and filtering	
echo	print a text, e.g.: • echo something – print the text *something* on the screen • echo -e "Text with\tbackslash escape codes\ninterpretation" – print the text *Text with*[tab character]*backslash escape codes*[newline]*interpretation* • echo $var – print the value of variable var on the screen
cat	con**cat**enate files and print them on the screen, e.g.: • cat file1 – print the content of file1 • cat file1 file6 – print the content of file1, then of file6
more	print a file on the screen, one screenful at a time (press enter to show next line, press space bar to show next page; no scrolling up), e.g.: • `more long_text.txt`
less	print a file on the screen with up-down scrolling (use arrow keys for it) and searching options (type `/pattern` to search for `pattern`), e.g.: • `less long_text.txt`
head	print only a few first lines of a file, e.g.: • `head long_text.txt` – print the first 10 lines of the file (10 is the default) • `head -n 3 long_text.txt` – print the first 3 lines of the file
tail	print only a few last lines of a file, e.g.: • `tail long_text.txt` – print the last 10 lines of the file (10 is the default) • `tail -n 3 long_text.txt` – print the last 3 lines of the file
sort	alphabetically **sort** the lines of a text file, e.g.: • `sort favourite_movies.txt` – print the lines of the file favourite_movies.txt in alphabetical order

grep	(**g**lobally search a **r**egular **e**xpression and **p**rint) print lines that match a pattern, e.g.: • `grep cookie file1` – print all the lines from file1 that contain the word *cookie* • `grep -i cookie file1` – print all the lines from file1 that contain the word *cookie* and ignore the case (so, all of these match: *cookie*, *COOKIE*, *Cookie* and so on)
cut	print selected parts of lines of a file
strings	print **strings** of printable characters in a file (especially useful on binary files)
wc	print line, **w**ord and byte **c**ount for a file
file	determine **file** type
I/O redirection more on input/output redirection under the link 'Linux: I/O redirection' provided on the course webpage	
>	redirect standard output of a **command**/program to a **file** (instead of to the screen) with overwriting the file contents, e.g.: • `echo My text > file.txt` – execute command echo and print its output to file.txt instead of displaying it on the screen; remove the previous content of the file if it existed
>>	redirect standard output of a **command**/program to a **file** (instead of to the screen) with appending to the file, e.g.: • `echo My text >> file.txt` – execute command echo and print its output to file.txt instead of displaying it on the screen; add the new content at the end of the file (do not remove existing content if it existed)
<	redirect standard input of a **command**/program from a **file** (instead of from the keyboard), e.g.: • `my_program xyz 1 2` – run my_program with command line arguments given from the keyboard, whereas: • `my program < file` – run my_program with command line arguments read from a file
|	pipe - redirect standard output of a **command** to standard input of another **command**, e.g.: • `cat file1 file2 | sort | head -n 3` – read the two files, then sort their lines alphabetically, then display the first 3 of the sorted lines
disk usage	
df	(**d**isk **f**ree) disk space usage (free and used) for mounted file systems
du	**d**isk **u**sage (size) of directories and files
processes	
ps	report **p**rocess **s**tatus (a snapshot)
top	display information about processes (with real-time update)
kill	send a signal to a process (usually a termination signal)