1. Basics

You can select which desktop environment (DE) you want to use in the login screen. The default choice in the computers of classroom 6218 is Unity, which is quite different compared to traditional desktops (for example Windows 7, 8, 10). A more traditional choice is KDE - experiment, and find the one you like. The exercises in this course are mostly done with a terminal and a text editor, so try to get them running in your choice of DE.

For terminal (text mode) use make sure you are familiar with the following set of shell commands and utilities:

- Getting help for various commands
  - man (see ~> `man man`, Use '/'sign to open search prompt)
- Listing and copying/moving/removing files or directories
  - cd, pwd
  - ls
  - cp, mv, rm, mkdir, rmdir
  - chmod (permissions)
- Handling processes
  - ps
  - kill
  - fg / bg / jobs (CTRL+Z to suspend a process)
- File contents
  - cat / more / less
  - grep
- Controls
  - CTRL+C
  - CTRL+D
  - CTRL+Z

If you start a graphical program from terminal remember to use `'&'` after the command to run it in background and keep the shell window operational. If you forget to do so you can still release the shell by pressing CTRL+Z (suspend), and use command "bg" to move the process into the background (or "fg" to bring it back to the foreground).

Check some of the following editors and pick one to use in programming assignments. Text mode editors are useful if you have to use them remotely, otherwise graphical editors are probably more comfortable.

- DE editor is the preferred choice for most users. Which editor is available in your environment? How to use that editor?
- nano is a lightweight text mode editor, which does not hold much special features.
- vim or emacs - if you already know what you are doing.

Remember, you may have multiple windows open, e.g. one for an editor, one for a compiler and one window to run the programs.

Vastaus

Vim oli ubuntun ehdottama ohjelma koodien muokkaamista varten, mutta se vaikutti itselle hieman hämmentävältä eli ymmärtämiseen olisi tarvinnut googlea, joten päädyin käyttämään tuttua ja turvallista Text Editoria, joka ainakin vielä ajaa asiansa vallan mainiosti.

2. (Secure) Remote use

You can operate *nix machines remotely using the secure *ssh* connection. On most workstations there is an *ssh client* installed. If you are using another *nix machine you might want to enable the X windowing with an -X option so that you are able to use the programs that open new windows (~> `ssh <computername> -X`). To move files between the local computer and the remote, you'll need to use *scp*. Test out these commands, and try to use another computer inside your network from your other computer. If unable to do so, test out the ssh connection by setting up your own GitHub account and connecting to it with these instructions:

https://docs.github.com/en/authentication/connecting-to-github-with-ssh

- ssh: `ssh <remote name>`, log out with "exit" or "logout"
- scp: `scp somefile <remote site location>`

Vastaus

```
Hi markus-byte-ard! You've successfully authenticated, but GitHub does not provi
de shell access.
```

3. Special characters

Test the following commands to see how the special meaning of a certain character behave. \ normally supresses the special meaning of any character that follows it (you can use any environment variable instead of SHELL)

```
% echo $SHELL
Output: /bin/bash

% echo \$SHELL
Output: $SHELL

% echo \\$SHELL
Output: \/bin/bash

% echo \\\$SHELL
Output: \$SHELL

% echo '\$'
Output: \$

% echo \\
```

```
Output: \

% echo \\\\
Output: \\

% echo "\$"
Output: $

% echo '"$'
Output: "$

% echo "$"
Output: $
```

4. Shell scripting and parameters

Experiment with (some of) the shell script examples given in the slide set (see exercise page), (starting from page 18).

Then write a shell script that prints its startup parameters (environment variables, and positional and special parameters), each parameter on a separate line.

```
1 #!/bin/bash
2 echo "$USER"
3 arvo="Tämä on arvo"
4 echo $arvo
5 for i in $*; do
6     echo $i
7 done
8 echo -n "Anna jokin argumentti: "
9 read argumentti
10 echo $argumentti
11 env
```

```
student@ubuntu:~/Desktop/Kurssimateriaali/Viikko_8$ ./4.bash Tässä muutama argumentti
student
Tämä on arvo
Tässä
muutama
argumentti
Anna jokin argumentti: jokin
jokin
SHELL=/bin/bash
SESSION_MANAGER=local/ubuntu:@/tmp/.ICE-unix/1614,unix/ubuntu:/tmp/.ICE-unix/1614
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
LC_ADDRESS=fi_FI.UTF-8
GNOME_SHELL_SESSION_MODE=ubuntu
LC_NAME=fi_FI.UTF-8
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
LC_MONETARY=fi_FI.UTF-8
SSH_AGENT_PID=1578
GTK_MODULES=gail:atk-bridge
DBUS_STARTER_BUS_TYPE=session
PWD=/home/student/Desktop/Kurssimateriaali/Viikko_8
XDG_SESSION_DESKTOP=ubuntu
```

5. File searching with a shell script

Write a shell script that can be used to search for some specified manual page from directories found under `/usr/share/man`, and print all matching ones. For example:

```
$ search_page
usage: search_page command
$ search_page printf
/usr/share/man/man1/printf.1.gz
/usr/share/man/man3/printf.3.gz
```

Notice that `/usr/share/man` is only the root directory of manual pages. The actual pages are located in subdirectories corresponding to sections. For instance, the manual pages of the user commands belong to section 1, so the full path to `printf.1.gz` is `/usr/share/man/man1/printf.1.gz`.

```
#!/bin/bash
if [  $# -le 0 ]; then
    echo "Usage: command"
    exit 1
fi
filelocation="/usr/share/man"
list=$(find $filelocation -name "$1*")
for i in $list; do
    echo $i
done
```

```
student@ubuntu:~/Desktop/Kurssimateriaali/Viikko_8$ ./5.bash printf
/usr/share/man/man1/printf.1.gz
/usr/share/man/man3/printf.3.gz
```

6. Printing file contents

Write a shell script that prints the 10 most frequent words found in an input. For example (using file /usr/share/common-licenses/GPL):

```
345 the
221 of
192 to
184 a
151 or
128 you
102 license
 98 and
 97 work
 91 that
```

You may want to experiment with commands

- `tr`
- `sort`
- `uniq`

- `tail`

and build an appropriate pipe for your script. Start by printing each word in the file on a separate line, and then proceed to counting (hint: the whole thing can be accomplished with one line of code, using pipes suitably). Also test your script with the attached text.

```
#!/bin/bash
#lukee tiedoston
cat midsummer_nights_dream.txt |
# muuttaa kaikki ei-alphanumeeriset merkit ja korvaa ne "\n":llä eli muuttaa sanojen välissä olevat välit rivinvaihdoiksi.
tr -cs "[:alnum:]" "\n"|
# muuttaa kaikki isot kirjaimet pieniksi
tr "[:upper:]" "[:lower:]" |
# järjestää
sort |
# Laskee kuinka monta kertaa sanat esiintyvät
uniq -c |
# järjestää numerot reverse järjestyksessä
sort -nr |
# näyttää rivinumeron
cat -n |
# lukee 10 ensimmäistä riviä
head -n 10
```

Ilman kommentteja koodi menee yhdelle riville.

```
student@ubuntu:~/Desktop/Kurssimateriaali/Viikko_8$ ./6.bash
     1      573 and
     2      562 the
     3      472 i
     4      337 to
     5      275 you
     6      267 of
     7      264 a
     8      240 in
     9      201 my
    10      195 is
```