Exercises 2

Contents

Preparation 1

Exercise 2.1 – Files and File info 1

Exercise 2.2 – Redirection 1

Exercise 2.3 – Shell script(s) that uses data from the command line 2

Exercise 2.4 Using WSL with Java development in Windows 2

2.4.1 Create a WSL link and add a folder inside WSL 2

2.4.2 Git Branching and Merging 3

Exercise 2.5 – Extracting the project's Git log 4

In this set of exercises, we'll practice with various Linux tools and concepts using the WSL. In the course of the exercises, we'll try to work in both Windows and Linux seamlessly.

Preparation

Note the directory where you placed the gitlog.txt file created in Exercise 1.4b

Exercise 2.1 – Files and File info

For this exercise, you'll investigate the types of information kept about a file.

a) Look at each of the three dates for your file gitlog.txt using `ls -l` (`-l` by default gives modification date. Run the ls man page with `man ls` and read under the options -c and -u to see how to get the access date.
b) Which of these dates above will change when you do some operations for instance micro on the file? Now run the command on the file using `micro gitlog.txt` (just do some minor change on the file and save it again) and verify your answer.

Exercise 2.2 – Redirection

For this exercise, you'll run a Linux command and send the result to a file.

a) Use the output redirect operator (`>`) to send the contents of your home directory to file `myhome.out`. **Hint:** ensure that you are in the home directory. Use `pwd` to check the current working directory. Then issue the command `ls > myhome.out`.
b) Use the `cat` or `less` command to see the contents of `myhome.out`
c) Use VSCode to see the output: `$ code myhome.out`

Exercise 2.3 – Shell script(s) that uses data from the command line

Imagine you need to maintain a file of SEP4 tasks in your home directory. A task file consists of one line per team member with three fields separated by tabs: Team member's name, their email address and assigned task id. An example of a line in the file could be:

```
Mikael          123456@via.dk   task_234
```

All inside WSL (doing all stuff from the command line and using the `micro` editor), write a bash script called addtask.sh to add a task to a file. We will create a script that can be run with 4 command line arguments as follows:

```
$ ./addtask.sh filename <name> <email> <taskname>
```

Where

filename:	is the name of a task file to update – ex: `ourtasks.txt`
<name>:	is the first name of the person that should be assigned a task – ex: `Jonas`
<email>:	is the email of the person to be assigned a task – ex: `654321@via.dk`
<taskname>:	is the description of the task being assigned – ex: `"Do the dishes"`[1]

Your script should read the name of the task file from the command line argument and then ask the user for the task details (name, email, id), and then append the task to the file.

Hints:

- Use `echo` with `>>` to append to a file.
- Remember to make your script executable with the command `chmod`
- Look at https://ryanstutorials.net/bash-scripting-tutorial/bash-variables.php#arguments for info on getting data into your script via command line arguments.

Exercise 2.4 Using WSL with Java development in Windows

2.4.1 Create a WSL link and add a folder inside WSL

If you have not already done so, you should create a link to your WSL installation in Explorer:

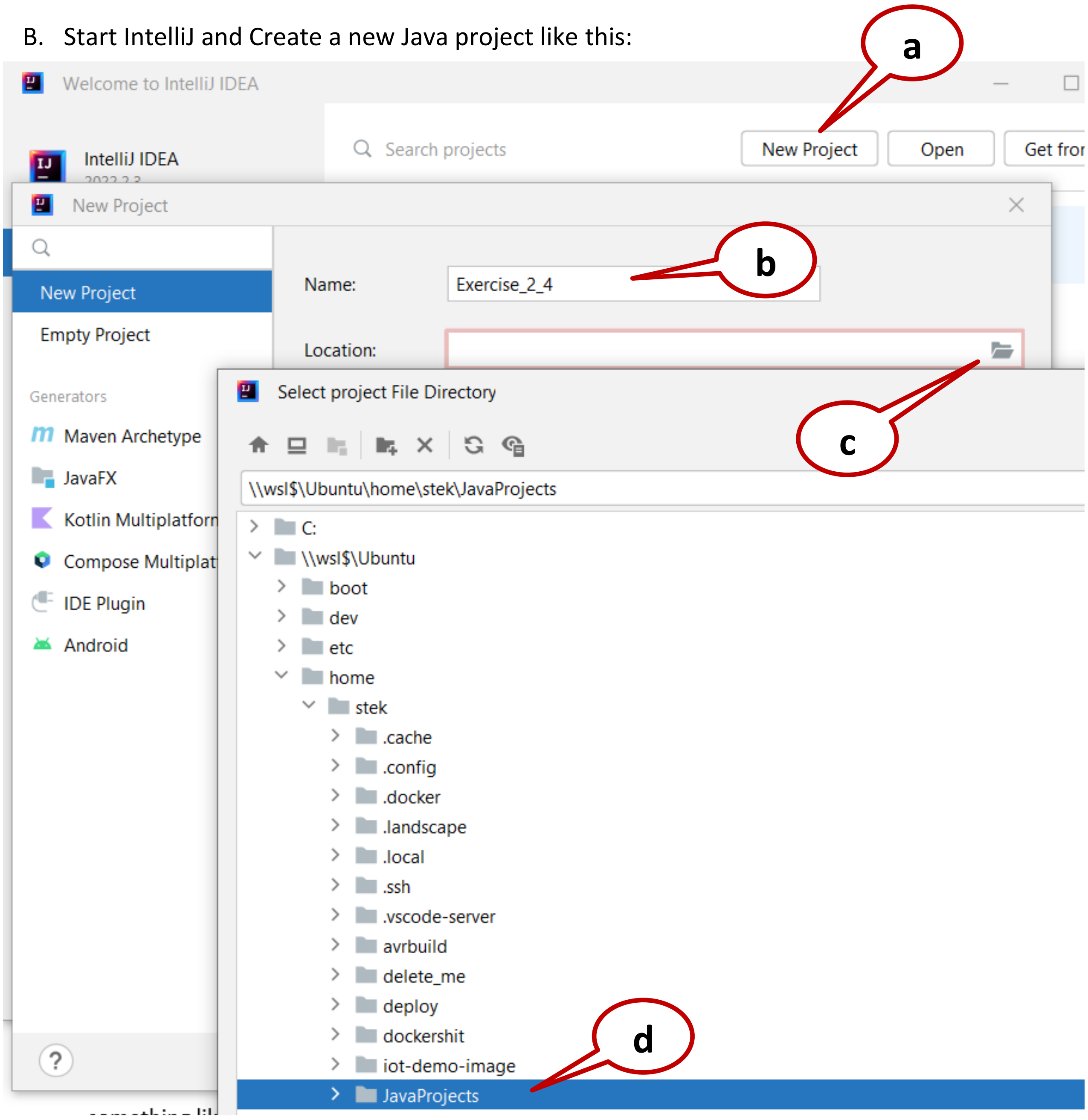
1) Start an WSL shell
2) In the shell, type: `explorer.exe .`
3) A Windows Explorer opens showing the contents of your WSL home folder.
4) Pull the path shown in Explorer to the *Quick Access* (or *Hurtig Adgang* on Danish Windows) list on the left in your Explorer.
5) That's it – you can now easily access your WSL installation. It actually works as a network drive but that doesn't matter, it works fine.

Now we will redo parts of exercise 1.3 from last time, but using a Java project placed inside your WSL home folder and using the WSL shell for all command line stuff. To get started quickly do like this:

A. Start a WSL shell, and make a new folder called JavaProjects: `$ mkdir JavaProjects`
(see next page for more steps)

[1] If we put the task details inside a set of "...." we will be allowed to have spaces in the description, and it will still just be seen as a single argument.

B. Start IntelliJ and Create a new Java project like this:



Name: Exercise_2_4

Location:

```
\\wsl$\Ubuntu\home\stek\JavaProjects
> C:
v \\wsl$\Ubuntu
    > boot
    > dev
    > etc
    v home
        v stek
            > .cache
            > .config
            > .docker
            > .landscape
            > .local
            > .ssh
            > .vscode-server
            > avrbuild
            > delete_me
            > deploy
            > dockershit
            > iot-demo-image
            > JavaProjects
```

a. Click on New Project as usual.
b. Give your new project a suitable name.
c. Click on the small folder icon to choose a location and select the folder called something like \\wsl$\Ubuntu\home\your_wsl_username
d. Make sure you select the *JavaProjects* folder we create just before.

That's it, you can now work with the Java project as you normally do, even though it's actually located inside the WSL file system.

At this point, we'll simply redo exercise 1.3 (more or less) from session 1, only this time *do all your command line stuff* inside WSL using a WSL shell:

2.4.2 Git Branching and Merging

a) Create a new Java project in IntelliJ (it could be a Spring boot Project or just a simple project with a single Main class containing psvm) and place the new project under version control.

b) Create 3 small classes (feel free to create 3 controller classes or whatever you please). For each class:
    i. Create a new dev branch
    ii. Create a single class
    iii. Make a happy test of the class (in psvm)
    iv. Add/commit changes to current branch (both via IntelliJ and WSL commandline)
    v. Do some poking around via an WSL shell to see Git status, view git log
    vi. Merge new branch into main
    vii. Delete dev branch

Exercise 2.5 – Extracting the project's Git log

You need to export the Git log for your project and submit it to the itslearning assignment called "*Upload the Git log for exercise 2*". Follow these steps:

a) Open WSL Bash and `cd` to the root folder of your project.
b) On the command line write: $ `git log > gitlog.txt`
c) Inspect, using VS Code, the file (gitlog.txt) in 2 ways:
    - From Linux using the WSL shell: `code gitlog.txt`
    - From Windows using Explorer: Start Explorer and, using the link we created earlier, drill down to the folder where the gitlog.txt resides and right-click the file and choose "Open with Code"
d) Upload the file.