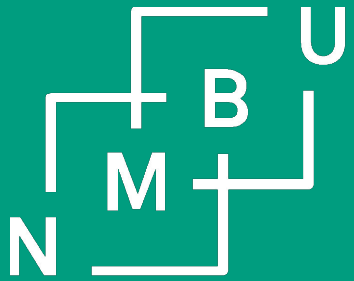


Norwegian University
of Life Sciences

Linux Fundamentals for Robotics

TEL211

Folder Hierarchy

- Open up a fresh Linux terminal window.
- Let's check out the directory (i.e. **folder**) structure of the **catkin_ws** folder, the workspace folder for ROS.
- Install the tree program.

```
sudo apt-get install tree
tree catkin_ws
```

- You should see a hierarchy of all the folders underneath the `catkin_ws` folder.
- If at anytime, you want to see the hierarchy of all folders underneath a specific folder, just type:

```
tree <path to the folder>
```

```
ros@ros-VirtualBox:~$ tree catkin_ws
catkin_ws
├── build
│   ├── atomic_configure
│   │   ├── env.sh
│   │   ├── local_setup.bash
│   │   ├── local_setup.sh
│   │   ├── local_setup.zsh
│   │   ├── setup.bash
│   │   ├── setup.sh
│   │   ├── _setup_util.py
│   │   └── setup.zsh
│   ├── catkin
│   │   └── catkin_generated
│   │       └── version
│   │           └── package.cmake
│   ├── catkin_generated
│   │   ├── env_cached.sh
│   │   ├── generate_cached_setup.py
│   │   ├── installspace
│   │   │   ├── env.sh
│   │   │   ├── local_setup.bash
│   │   │   ├── local_setup.sh
│   │   │   ├── local_setup.zsh
│   │   │   ├── setup.bash
│   │   │   ├── setup.sh
│   │   │   ├── _setup_util.py
│   │   │   └── setup.zsh
│   │   ├── metapackages
│   │   │   ├── turtlebot3
│   │   │   │   └── CMakeLists.txt
│   │   │   └── turtlebot3_simulations
│   │   │       └── CMakeLists.txt
│   │   ├── order_packages.cmake
│   │   ├── order_packages.py
│   │   ├── setup_cached.sh
│   │   └── stamps
│   │       └── Project
│   │           ├── interrogate_setup_dot_py.py.stamp
│   │           ├── order_packages.cmake.em.stamp
│   │           ├── package.xml.stamp
│   │           └── _setup_util.py.stamp
│   ├── CATKIN_IGNORE
│   ├── catkin_make.cache
│   ├── CMakeCache.txt
│   ├── CMakeFiles
│   │   ├── 3.10.2
```

Navigate Between Folders

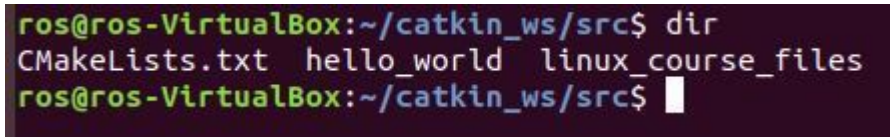
- move to our catkin_ws/src folder.

```
cd catkin_ws/src
```

Type the following command to see what files you have in there:

```
dir
```

E.g.



- Linux systems are made up of two main parts:

  Files: Used to store data

  Folders: Used to store files and other folders

Navigate Between Folders

- check the path to a directory.

```
pwd
```

- How do we get back to the home folder?

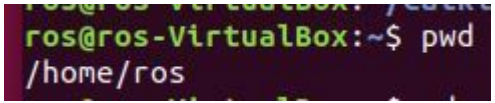
```
cd ~
```

- we could have also done:

```
cd /home/ros
```

- See what the official path to this directory is:

```
pwd
```



Listing Files in a Directory

- List all the folders in the home directory.

```
cd ~
ls
```

```
ros@ros-VirtualBox:~$ ls
Arduino         catkin_ws  Documents  examples.desktop  Pictures  Templates
arduino-1.8.10  Desktop    Downloads  Music             Public    Videos
ros@ros-VirtualBox:~$
```

- That `ls` command doesn't list the hidden files. We need to type this command to list **all** the files.

```
ls -la
ls --all, alternatively
```

- To get all the possibilities of commands you can type with ls, just type:

```
ls --help
```

- To get the manual, we could have alternatively typed:

```
man ls
```

- To get out of the manual, type **q**.

```
ros@ros-VirtualBox: ~
File  Edit  View  Search  Terminal  Help
drwxr-xr-x  3 ros  ros   4096 Oct 14 12:24 Documents
drwxr-xr-x  2 ros  ros   4096 Oct 21 18:01 Downloads
-rw-r--r--  1 ros  ros   8980 Oct  7 08:39 examples.desktop
drwxr-xr-x  5 ros  ros   4096 Oct 21 16:05 .gazebo
drwx------  3 ros  ros   4096 Oct 21 18:06 .gnome
drwx------  3 ros  ros   4096 Oct  7 08:51 .gnupg
-rw-------  1 ros  ros  10778 Nov 11 14:39 .ICEauthority
drwxr--r--  3 ros  ros   4096 Oct 21 06:23 .ignition
drwxr-xr-x  3 ros  ros   4096 Oct 21 18:11 .java
drwx------  3 ros  ros   4096 Oct  7 08:43 .local
drwx------  5 ros  ros   4096 Oct  7 08:45 .mozilla
drwxr-xr-x  2 ros  ros   4096 Oct  7 08:43 Music
drwxr-xr-x  2 ros  ros   4096 Oct 21 18:11 .oracle_jre_usage
drwxr-xr-x  2 ros  ros   4096 Oct  7 08:43 Pictures
-rw-r--r--  1 ros  ros    807 Oct  7 08:39 .profile
drwxr-xr-x  2 ros  ros   4096 Oct  7 08:43 Public
drwxr-xr-x  4 ros  ros   4096 Nov 11 15:15 .ros
drwxr-xr-x  2 ros  ros   4096 Oct 16 20:28 .rviz
drwxr-xr-x  2 ros  ros   4096 Oct 21 06:24 .sdformat
drwx------  2 ros  ros   4096 Oct  7 08:51 .ssh
-rw-r--r--  1 ros  ros      0 Oct  7 15:03 .sudo_as_admin_successful
drwxr-xr-x  2 ros  ros   4096 Oct  7 08:43 Templates
drwxr-xr-x  2 ros  ros   4096 Oct  7 08:43 Videos
ros@ros-VirtualBox:~$
```

Create New Folders and Files

- let's learn how to make a new folder. Type the following command:

```
cd ~

mkdir test_folder
```

- If you type **dir,** you should see a new folder in there named test_folder.

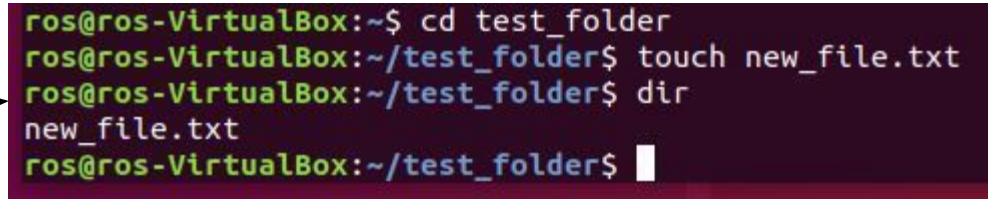
```
cd test_folder
```

- Create a new text file named **new_file.txt.**

```
touch new_file.txt
```

- Open the new file in a text editor.

```
gedit new_file.txt
```



Moving Folders and Files Between Directories

- syntax is for **moving**:

```
mv <file/folder we want to move> <destination>
```

- If you type `dir`, you should see a new folder in there named test_folder.

```
cd ~
mv test_folder ~/catkin_ws
```

- Lets see if we moved the `test_folder` to `~/catkin_ws` successfully

```
cd ~/catkin_ws
dir
```

- The syntax for the **copy** operation in Linux is:

```
cp <file we want to copy> <name of the new file>
```

- If you want to copy a folder, you can do the following command:

```
cp -r <folder we want to copy> <name of the new folder>
```

Moving Folders and Files Between Directories

- Let's make a copy of a file :

`cd ~/catkin_ws/test_folder` (remember we moved `test_folder` to `catkin_ws`)

```
cp new_file.txt new_file_copy.txt
```

- Let's create a folder and make a copy of it.

```
cd ~/catkin_ws/test_folder
mkdir temp_folder
cp -r temp_folder temp_folder_copy
```

- Get all options for `cp` command with

```
cp --help
```

- Remove files or folders with `rm`, you need to be careful with this command!

```
rm <file to remove>
rm -r <folder you want to remove>
```

- Lets clean mess we made

```
cd ~/catkin_ws
rm -r test_folder
dir
```

Permissions in Linux

- Let's make a copy of a file :

```
cd ~/catkin_ws/src/TEL211
ls -la
```

- In the beginning of the line, you see the following:

```
-rw-r--r-
```

- Linux has three permission types for files and directories:
  Read: Denoted by the letter `r`. This means that the user can read the file or directory.
  Write: Denoted by the letter `w`. This means that the user can write or modify the file or directory.
  Execute: Denoted by the letter `x`. This means that the user can execute the file.

- There are also three different user permission groups:
  Owner: That's me!
  Group: Whatever group the file or directory was assigned to.
  All Users: This permission group includes the rest of the users.

- For this case

  Owner has read and write privileges (i.e. rw-)
  The group has read privileges (i.e. r–)
  All other users have only read privileges (r–).

Bash Scripts in Linux

- Bash scripts are text files that can ***contain commands*** that we would normally type out manually in the Linux terminal.
- Then, when you want to run a set of commands, all you have to do is run the bash script

```
cd ~
mkdir test_bash
cd test_bash
touch  bash_script.sh
dir
gedit bash_script.sh
```

- Type these two lines of code inside that file and click Save.

```
#!/bin/bash
echo Hello from bash script
```

- First line of code tells Linux that this file is a **bash** script.
- **.sh** file extension is what you use for bash scripts
- **Execute** bash file with ?

```
./bash_script.sh
```

- ERROR , WHY ?

```
ls -la
```

Bash Scripts in Linux

- ERROR , WHY ?

```
ls -la
```

- `bash_script.sh` file only has read and write permissions because there is no `x`.

```
chmod +x bash_script.sh
```

```
ls -la
```

```
./bash_script.sh
```

- Got there yet ?
- Passing arguments to bash file

```
touch demo.sh
gedit demo.sh
```

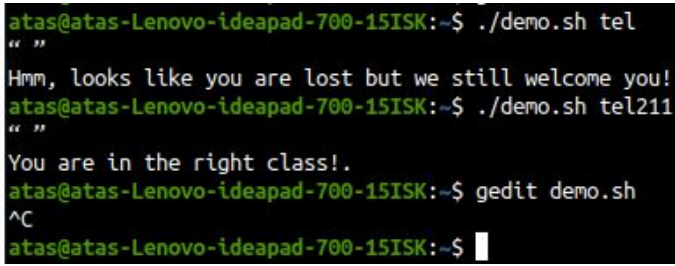
- Put The contents in next slide

Bash Scripts in Linux

- Put followin into `demo.sh`

```
#!/bin/bash

ARG1=$1

echo " "

if [ $ARG1 == "tel211" ]

then

  echo "You are in the right class!."

else

  echo "Hmm, looks like you are lost but we still welcome you!."

fi
```

- Make it executable with `chmod +x demo.sh` and execute with `./demo.sh`



The .bahsrc File and Environment Variables

- Let's explore `.bashrc` file

```
cd ~
gedit .bashrc
```

- .bashrc is a special bash script which is always located in your home directory.
- The .bashrc script runs automatically any time you open up a new terminal, window or pane in Linux.

- Open a fresh, new terminal window and type:

```
export
```

- You will see a list of all the **environment variables** in your system with their corresponding values
- The programs that run on your computer use environment variables to answer questions such as: What is the username of this computer? What version of ROS is installed? Where is ROS installed?, etc.
- Get environment variables with regex

```
export | grep ROS
```

- Only ROS related environment variables will be listed

The .bahsrc File and Environment Variables

- Let's explore `.bashrc` file

```
cd ~
gedit .bashrc
```

- .bashrc is a special bash script which is always located in your home directory.
- The .bashrc script runs automatically any time you open up a new terminal, window or pane in Linux.

- Open a fresh, new terminal window and type:

```
export
```

- You will see a list of all the **environment variables** in your system with their corresponding values
- The programs that run on your computer use environment variables to answer questions such as: What is the username of this computer? What version of ROS is installed? Where is ROS installed?, etc.
- Get environment variables with regex

```
export | grep ROS
```

- Only ROS related environment variables will be listed.
- Create a new environment variable

```
export NEW_ENV_VAR="I DEFINED THIS"
export | grep NEW
```

Processes in Linux

- Whenever you issue a command in Unix, it creates, or starts, a new **process**.
- A **process** is an instance of a running program

Foreground Processes

`command <args>` , by default every process is foreground

Background Processes

`command <args> &`, add `&` to end of command

- An example; launch a python program as a normal process

```
cd ~/catkin_ws/src/TEL211/scripts
python3 linux_basics_script_0.py
```
, in a separate terminal `ps -ef|grep script`

- Now you should see the “process” running with a PID, type `kill PID` in a terminal to terminate the running process Equivalent to Ctrl+C

Processes in Linux

- An example; launch a python program as a normal **background** process

```
cd ~/catkin_ws/src/TEL211/scripts
python3 linux_basics_script_0.py &
```

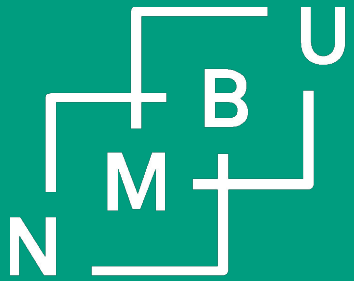
, in a separate terminal `ps -ef|grep script`

- You cannot stop a process running in background with Ctrl+C, you can also use `top` command to see CPU usages
- Only way to stop background process, is locate its PID with `ps -ef|grep script` and `kill PID`
- Good to know processes and their aims in linux programs

  You can debug programs to see their CPU consumption, memory footprint etc.

SSH in Linux

- Let's explore `.bashrc` fi



Norwegian University
of Life Sciences