CENG 334
Computer Networks

Laboratory I
Linux Tutorial

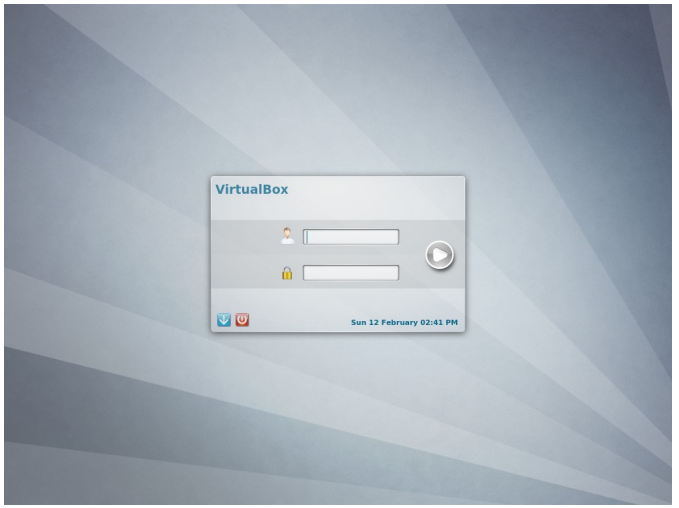
Çankaya University
Department of Computer Engineering
2013

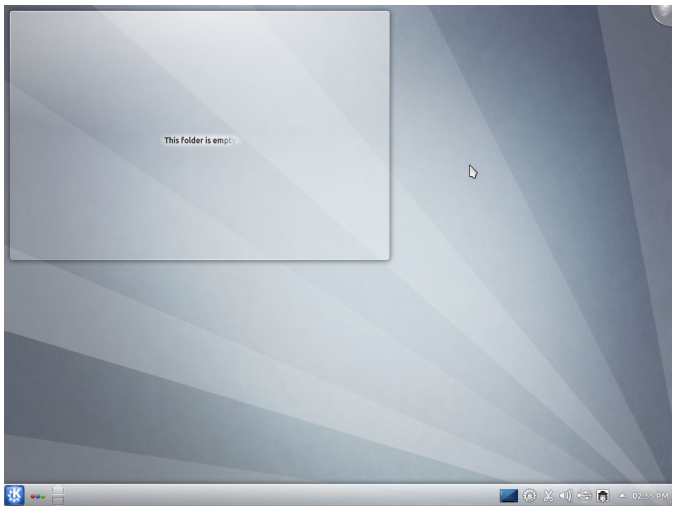
Contents

1. Logging In and Starting Session
2. Using Commands
    1. Basic Commands
    2. Working With Files and Directories
    3. Permission Bits
3. Introduction to Shell Scripting

1. Logging In and Starting Session

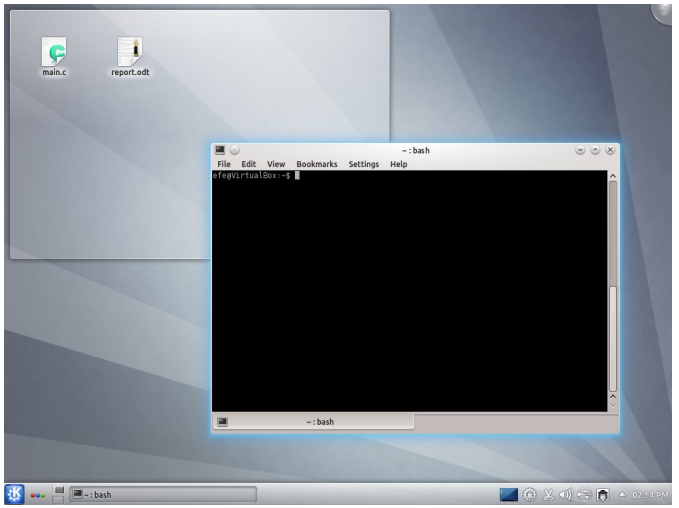
- Logging in to a Linux based operating system requires a username and a password.
- Both the username and the password are case sensitive so pay attention whether Caps Lock is on or not.
- If you have entered your information correctly, your desktop will be shown.





1. Logging In and Starting Session

- Each Linux based operating system (Debian, Ubuntu, Kubuntu, Pardus, Fedora, ...) may present different user interfaces and applications but common operations can be performed in a similar manner on each system.



2. Using Commands

- This laboratory manual will help you learn the basic Unix commands. You are expected to exercise each command on the following slides.
- For your own benefit, do not copy and paste the commands.
- You must first open a Konsole application to try the commands. In order to do so, click on the menu button on the bottom left corner of the screen and start typing "konsole". When you see the application, click on its icon.
- As with your username and password, commands are case sensitive too, so type them exactly as you see.

2.1. Basic Commands

- **date** – displays the current date and time.
- **whoami** – displays the login name of the current user.
- **echo** – tells the computer to retype the string after echo. Try the following:
  - **echo This is a test**
  - **echo $USER**
    What is the output now?
    What is $USER?
  - You can combine variables with strings: **echo Hello $USER**
  - **echo 2 + 2 = $((2+2))**
    What is the output of $((2+2))?

2.2. Working With Files and Directories

- **ls -l** – lists the files and directories in the working directory.
- **cd *<directory name>*** - used for changing the working directory.
- **pwd** – (print working directory) tells in which directory the user currently is.
- Try **cd Desktop**. Then try **pwd**. What is different now?
- **cat > services.txt**
  ftp 21 tcp
  ssh 22 tcp
  http 80 tcp
  http 80 udp
  https 443 tcp
  https 443 udp
  <ctrl – d>
- By **<ctrl – d>** we mean: hold the **Ctrl key** down and while it is down press **d**. We have just used cat to create a short list of common TCP/UDP ports.

2.2. Working With Files and Directories

- **cat services.txt** – shows us what is in the file dict.txt.
- **wc services.txt** – counts words, lines and characters (letters) in a given file.
- **grep tcp services.txt** – looks for the word tcp in the file services.txt and displays the lines in which this word appears. It gives us a way to search through files.
- **sort services.txt** – command does just what it says.
- **sort services.txt > services2.txt** – Note the use of the "into" symbol ">". In our example it had the effect of directing the output of the sort command from screen to the file services2.txt. Use **ls** to check if both files are available
- **cat services2.txt** – be sure that the content is correct.

2.2. Working With Files and Directories

- **mkdir services** – (**m**ake **dir**ectory), create a new directory named letters.
- **mv services.txt services2.txt services** – move both files into the services directory.
- **cd letters** – work inside the letters directory.
- **pwd**
- **ls**
- **rm services.txt services2.txt** – remove both files.
- **cd ..** – to go to the upper directory.
- **pwd**
- **rmdir services** – remove services directory. You can delete only empty directories or rmdir will fail.

2.2. Working With Files and Directories

- **man** – manual/help, to read detailed information about commands. Try:
  - **man *<commandname>***
  - **man ls** – Displays detailed information about ls command. Find which parameter is used for sorting by modification time.
  - To quit man simply type the letter **q**.

2.3. Permission Bits

- **ls -l *<filename>*** - will list the long directory list entry (which includes owner and permission bits) and the group of a file. The output looks something like:

```
permission   owner   group    size  date       time  filename
-rw-rw-r-- 1 student students 65536 2012-02-12 17:39 commands.html
```

- The Permission Bits;
  - The first position (which is not set) specifies what type of file this is. If it were set, it would probably be a d (for directory) or l (for link).
  - The next nine positions are divided into three sets of binary numbers and determine permissions for three different sets of people.

2.3. Permission Bits

```
u   g   o
421 421 421
rw- r-- ---
6   4   0
```

- The file has "mode" 640.
- The first bits, set to "r + w" (4+2=6) in our example, specify the permissions for the user who owns the files (u).
- The user who owns the file can read or modify (including deleting) the file.
- The next trio of bits, set to "r" (4) in our example, specify access to the file for other users in the same group (g) as the group of the file.
- In this case the group is students – all members of the students group can read the file (print it, copy it, or display it).
- Finally, all other users (o) are given no access (0) to the file.

2.3. Permission Bits

- ```
  u   g   o
  421 421 421
  rw- r-- ---
  6   4   0
  ```
  - One form of access which no one is given, even the owner, is "x" (for execute).
  - This is because the file is not a program to be executed.
  - It is probably a text file which would have no meaning to the computer. The x would appear in the third position if it was an executable file.
- If you wanted to make the file readible to all other users, you could type:

  chmod 644 *<filename>* or **chmod o+r *<filename>***.
  ```

3. Introduction to Shell Scripting

- Instead of typing commands into shell one by one, you can also create text files which contain these commands. Type the following:

```
cat > hello.sh
#!/bin/bash
# this is a simple shell script which greets the user
# and displays date
echo "Hello! The date and time is:"
date
<ctrl-d>
```

- Check the file you just have created with **ls -l**. You should note that your file does not have execution permissions, so you should fix it with **chmod +x hello.sh**. Now you can execute this simple script with **./hello.sh**.
- The first line is a must for all shell scripts, It defines which shell interpreter will execute this script. In this case it is **/bin/bash**.
- The next two lines are comments, they will never be executed and their only purpose is to inform the user.

3. Introduction to Shell Scripting

- Below is an another script which demonstrates usage of parameters:

```
#!/bin/bash
COURSE="CENG 334"
echo "Hello $1, welcome to $COURSE."
echo "Your username is $USER and $2 is your student ID."
```

- Save this script to **hello2.sh**. Then run it as:

```
./hello2.sh "your name" number
```

- In this example, we have used three different variables:
  - **COURSE**: a local variable which exists as long as the script runs.
  - **USER**: An environmental variable which is defined by the system.
  - **$1**, **$2**...: Command line arguments.