Linux Fundamentals

Version :1.0

By: Nguyen Anh Tu

Course duration: 6 hours

Last update: Aug 2015

Global CyberSoft

A World of Difference

www.globalcybersoft.com

Content

I. Introduction to Linux

1. Linux history
2. Distributions

1. Linux history

- 1969
  - ✓ All modern operating systems have their roots in 1969 when the C language and the Unix operating system were developed.

- 1980s
  - ✓ Many companies started developing their own Unix:
    - IBM created AIX
    - Sun SunOS (later Solaris)
    - HP HP-UX and about a dozen other companies did the same.

1. Linux history (cont)

- 1990s
  - ✓ Linus Torvalds, buying a 386 computer and writing a brand new POSIX (Portable Operating System Interface) compliant kernel.

- 2015
  - ✓ More than 97 percent of the world's supercomputers,
  - ✓ More than 80 percent of all smartphones,
  - ✓ Many millions of desktop computers,
  - ✓ Around 70 percent of all web servers,
  - ✓ A large chunk of tablet computers,
  - ✓ And several appliances (dvd-players, washing machines, dsl modems, routers, self-driving cars, space station laptops...) run Linux.

2. Distributions

- Which to choose?

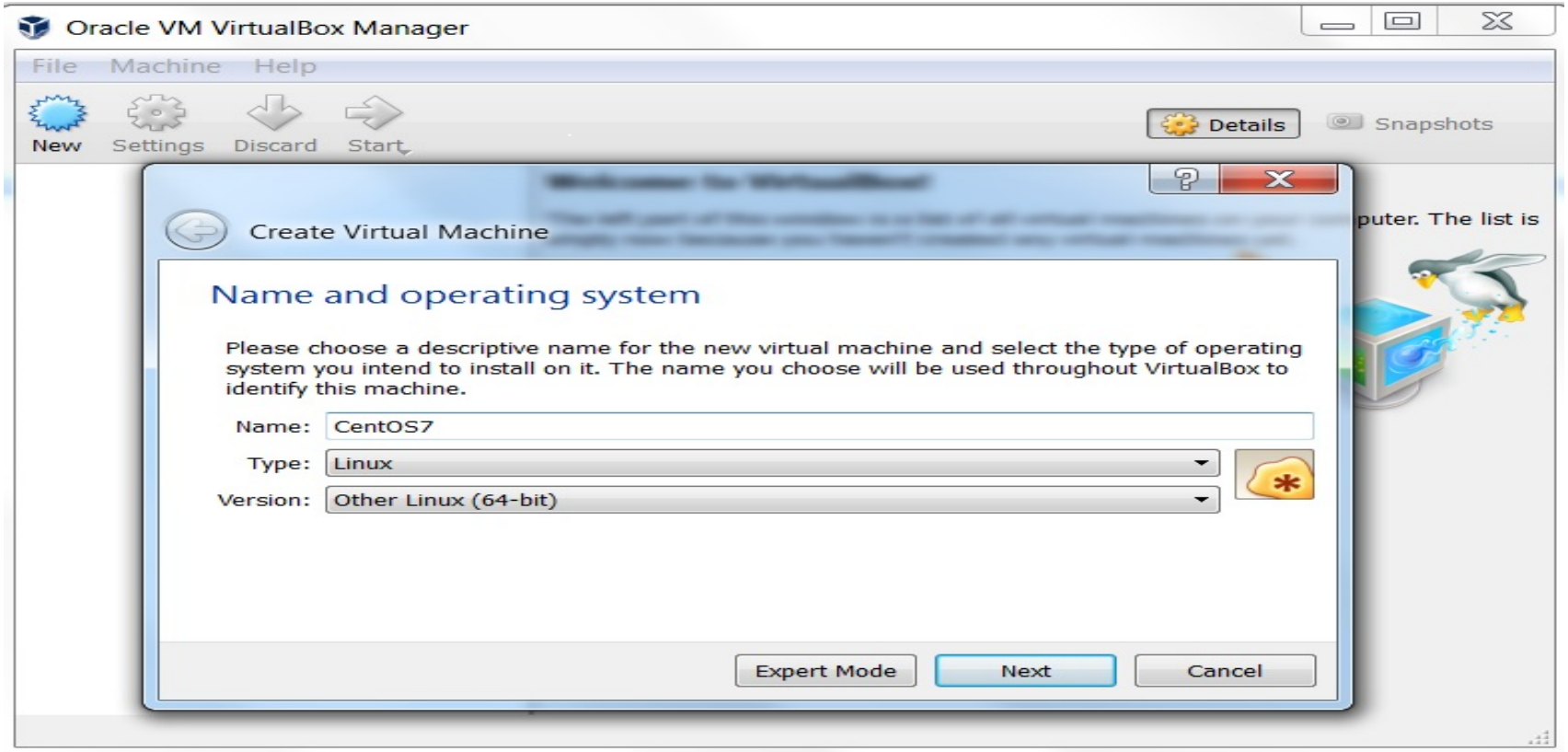
Distribution name	Reason(s) for using
Red Hat Enterprise (RHEL)	You are a manager and you want a good support contract.
CentOS	You want Red Hat without the support contract from Red Hat.
Fedora	You want Red Hat on your laptop/desktop.
Linux Mint	You want a personal graphical desktop to play movies, music and games.
Debian	Favorite for servers, laptops, and any other device.
Ubuntu	Very popular, based on Debian.
Kali	You want a pointy-clicky hacking interface.
Others	Advanced users may prefer Arch, Gentoo, OpenSUSE, Scientific, ...

II. Installing Linux

1. Virtualbox
2. Installing CentOS 7

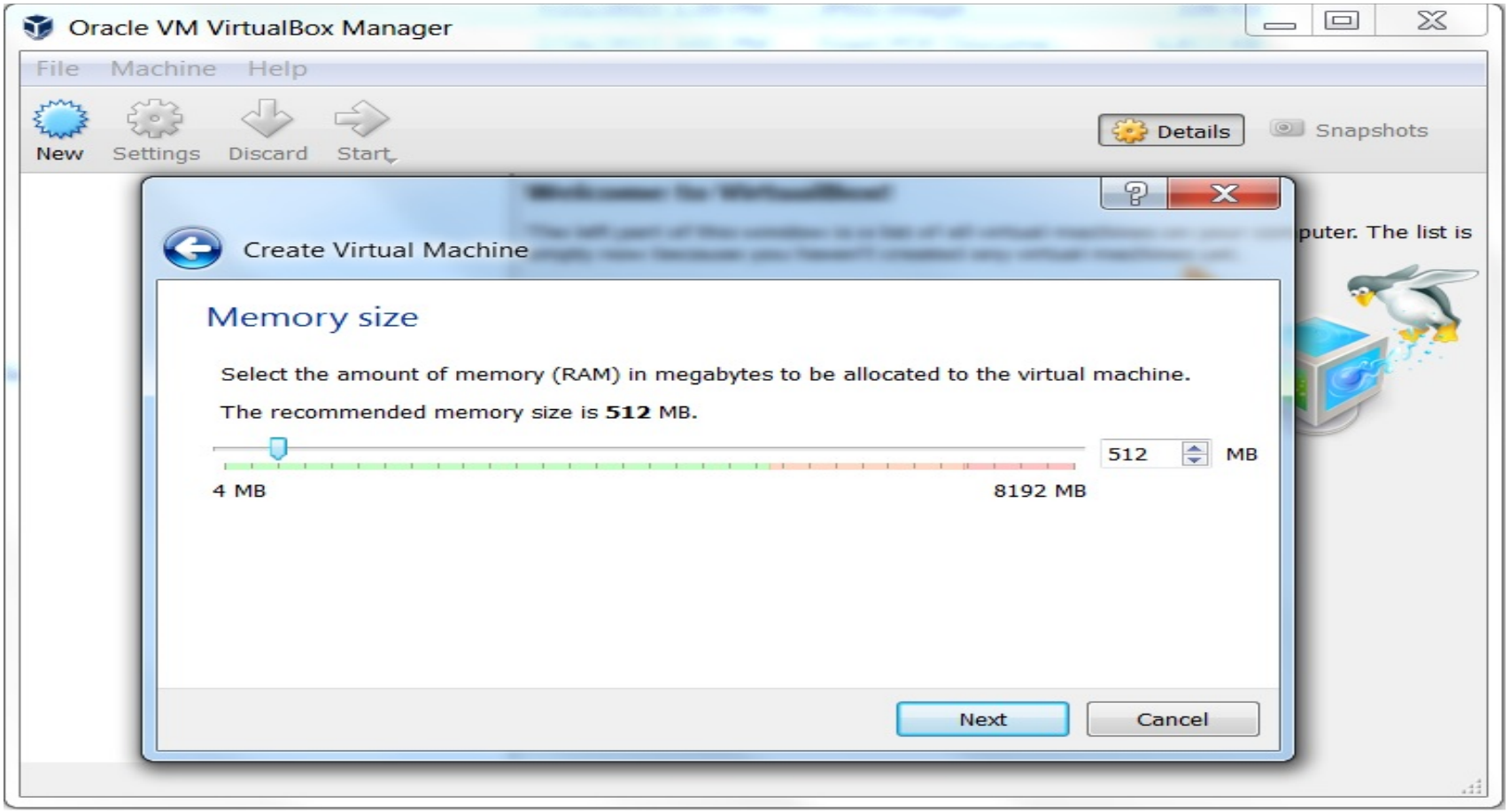
1. Virtualbox

- The Virtualbox website https://www.virtualbox.org/
- Start by clicking New and give a name (CentOS7). Click Next



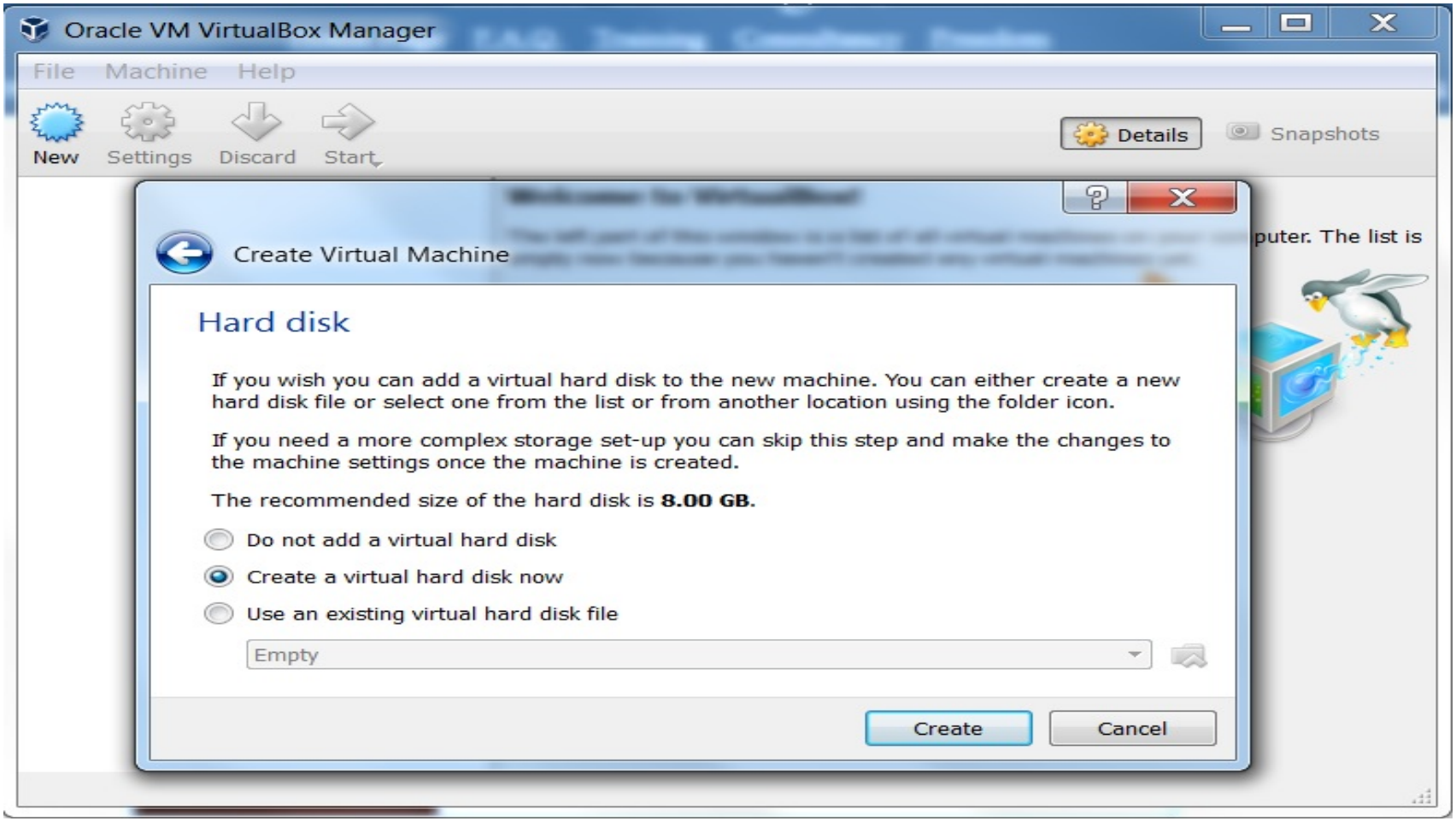
1. Virtualbox (cont)

- Linux without graphical interface will run fine on half a gigabyte of RAM.



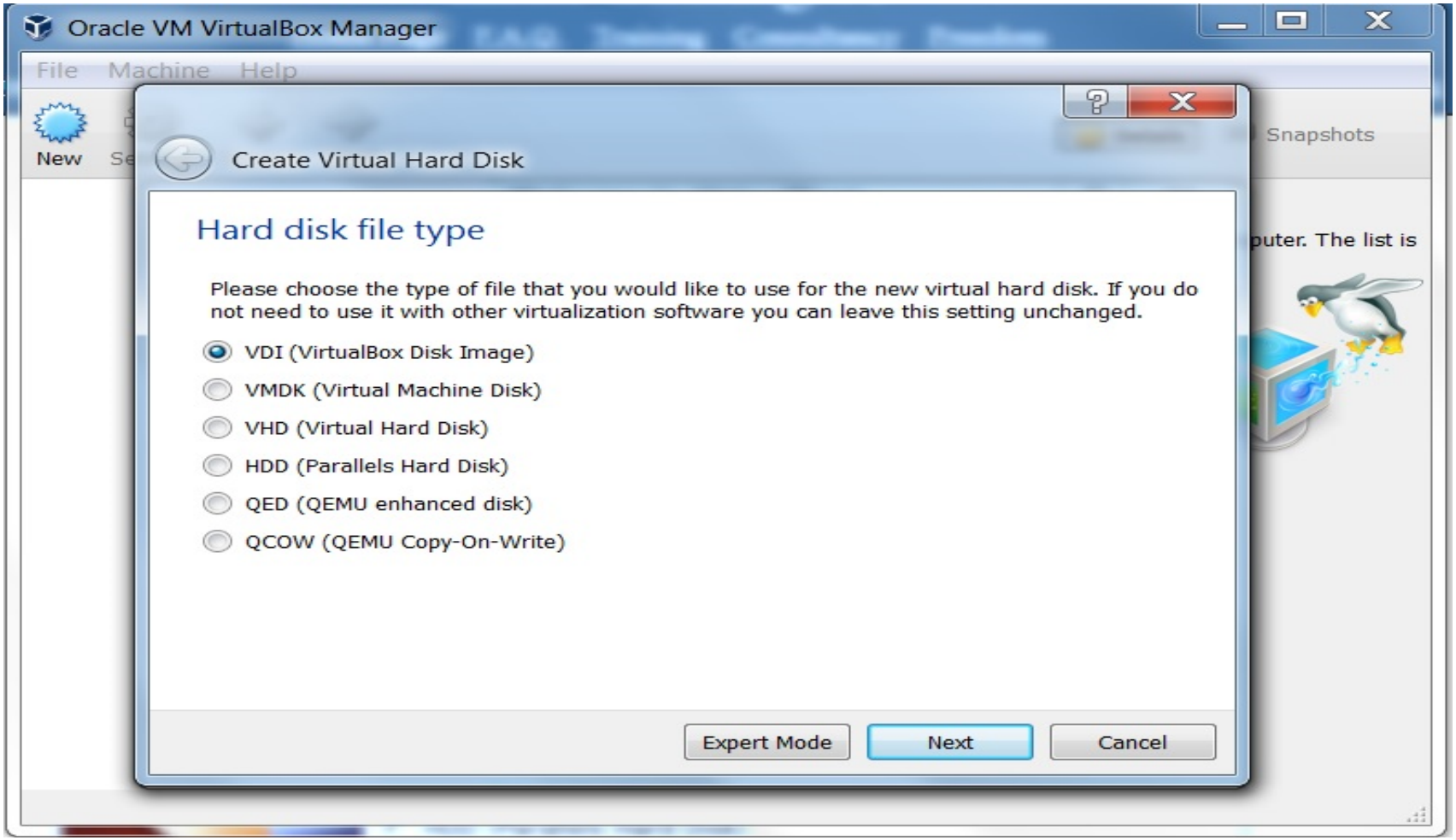
1. Virtualbox (cont)

- A Linux virtual machine will need a virtual hard drive.



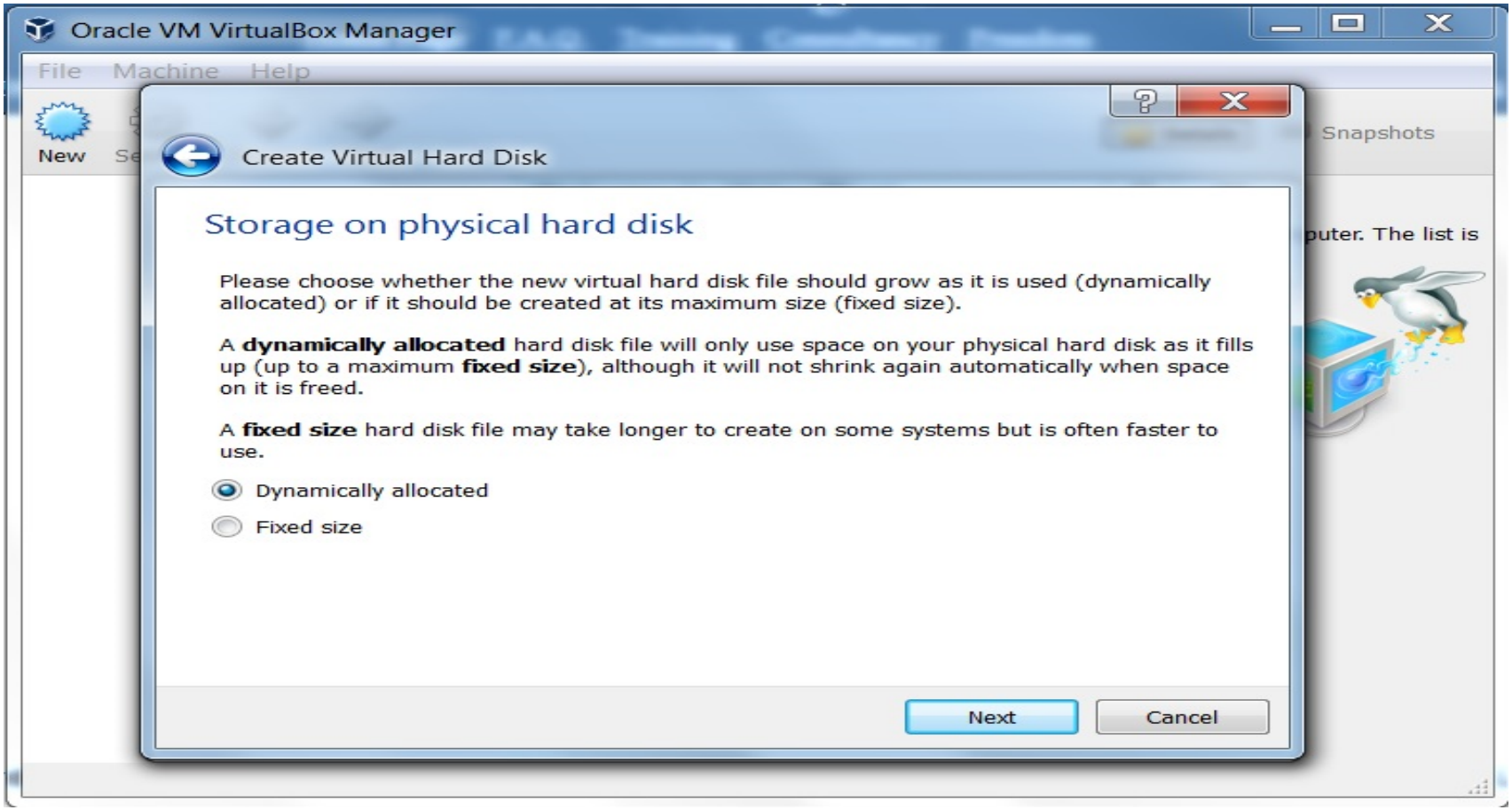
1. Virtualbox (cont)

- Left the default vdi for Hard disk file type.



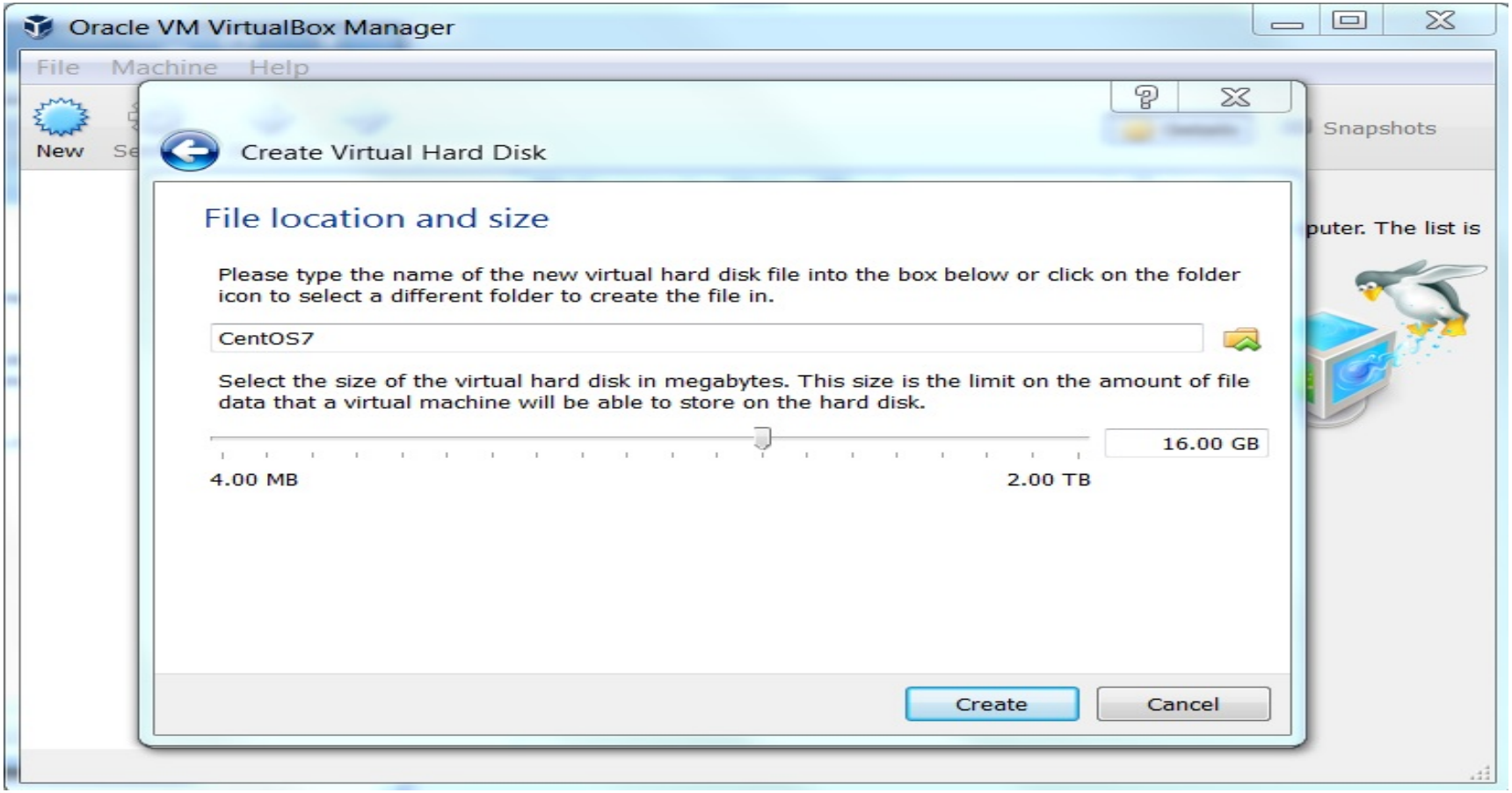
1. Virtualbox (cont)

- The default dynamically allocated type will save disk space.



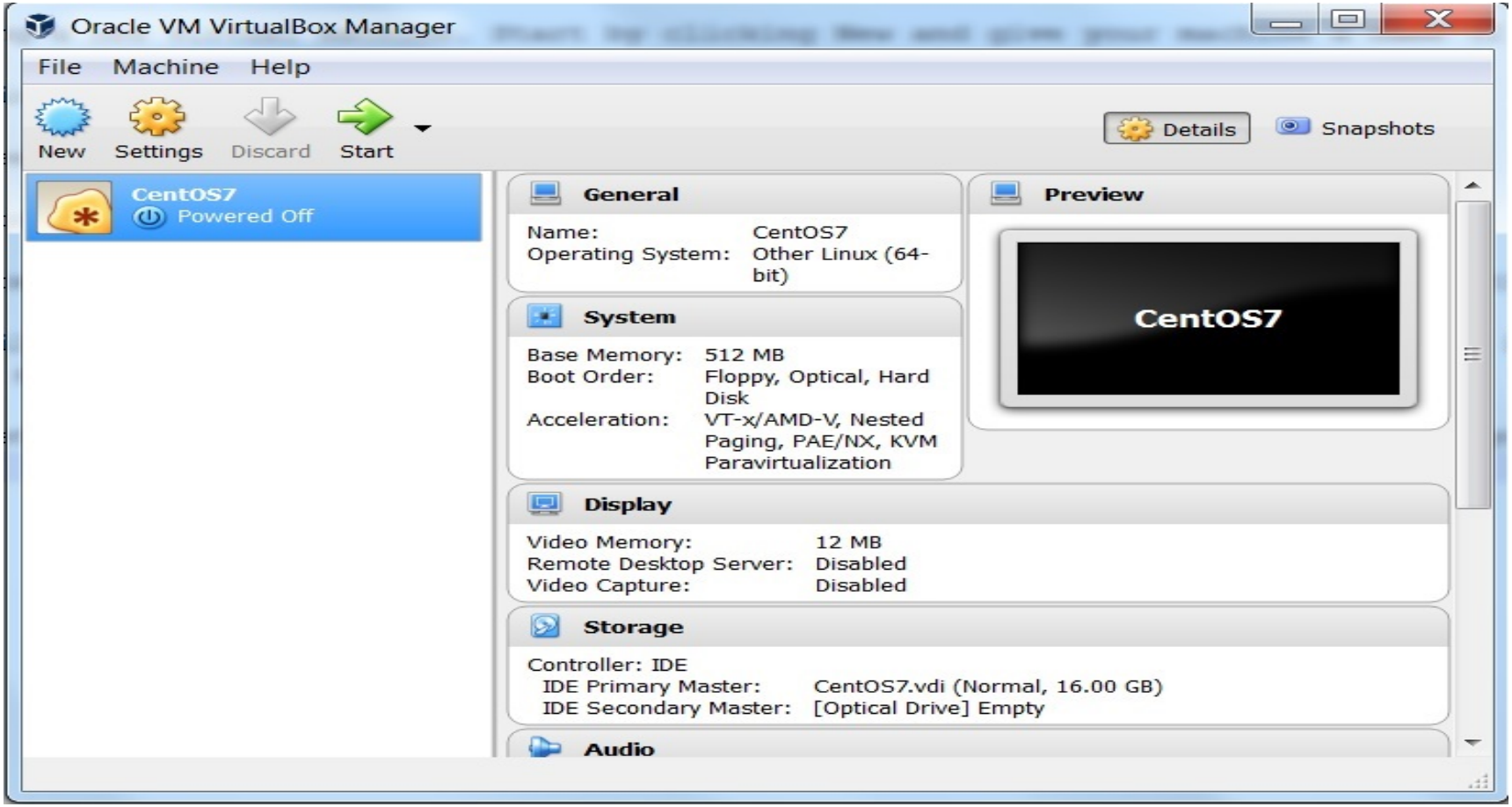
1. Virtualbox (cont)

- The name of the virtual disk file will be CentOS7.vdi.
- Also 16 GB should be enough to practice Linux.



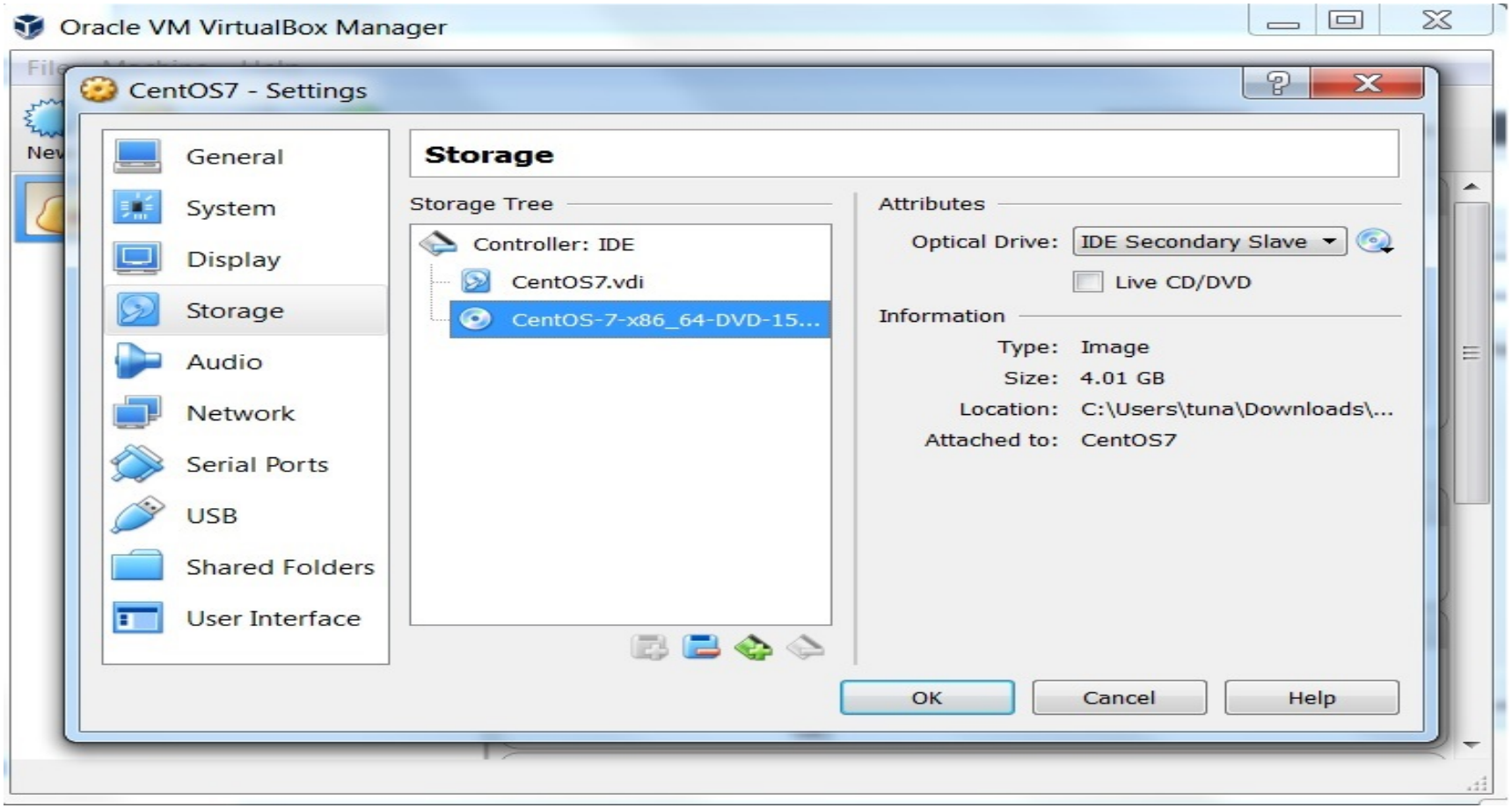
1. Virtualbox (cont)

- If all went well, then should see the machine just created in the list.



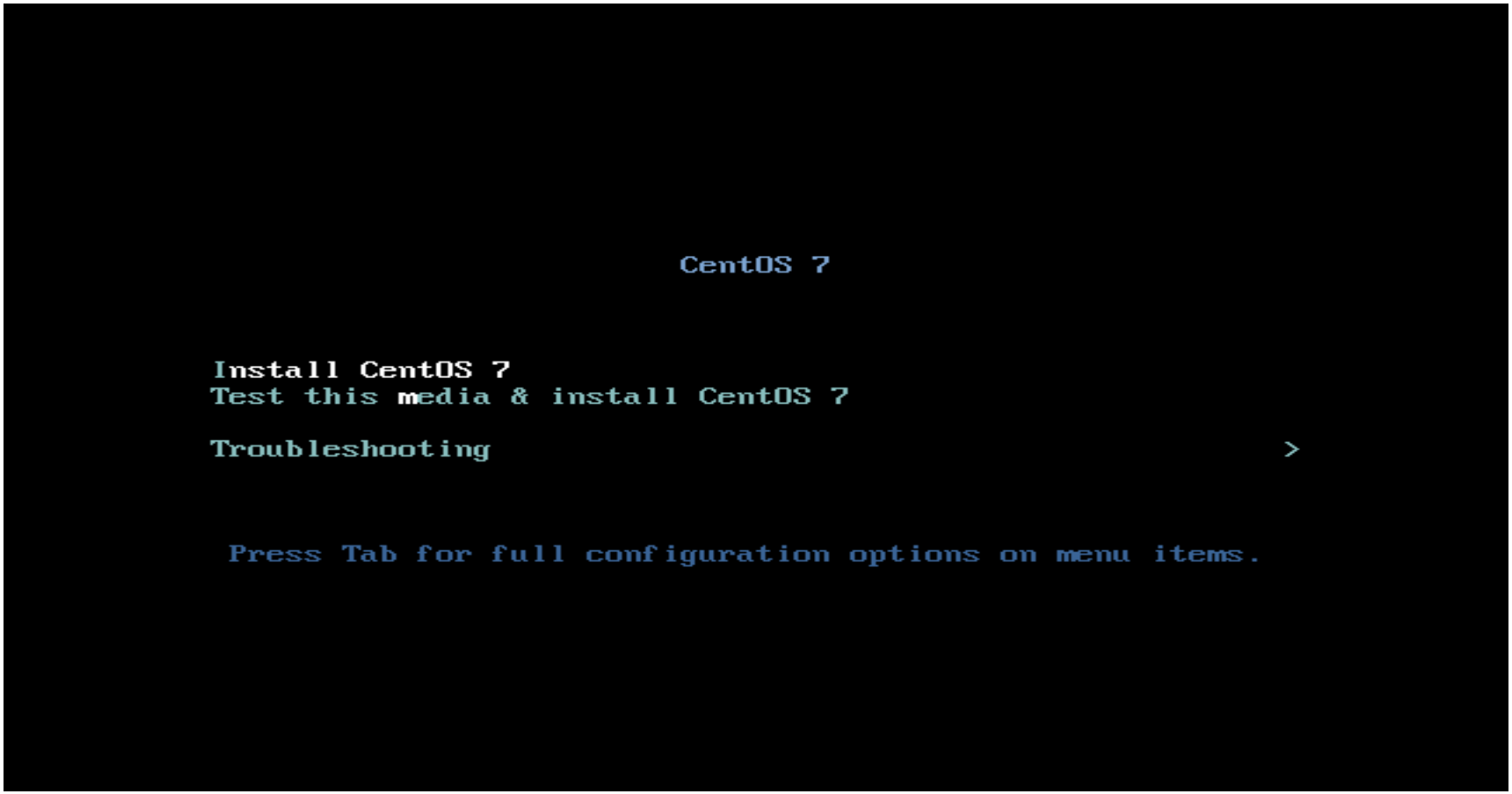
1. Virtualbox (cont)

- After finishing the setup, go into the Settings of the virtual machine and attach the .iso file.



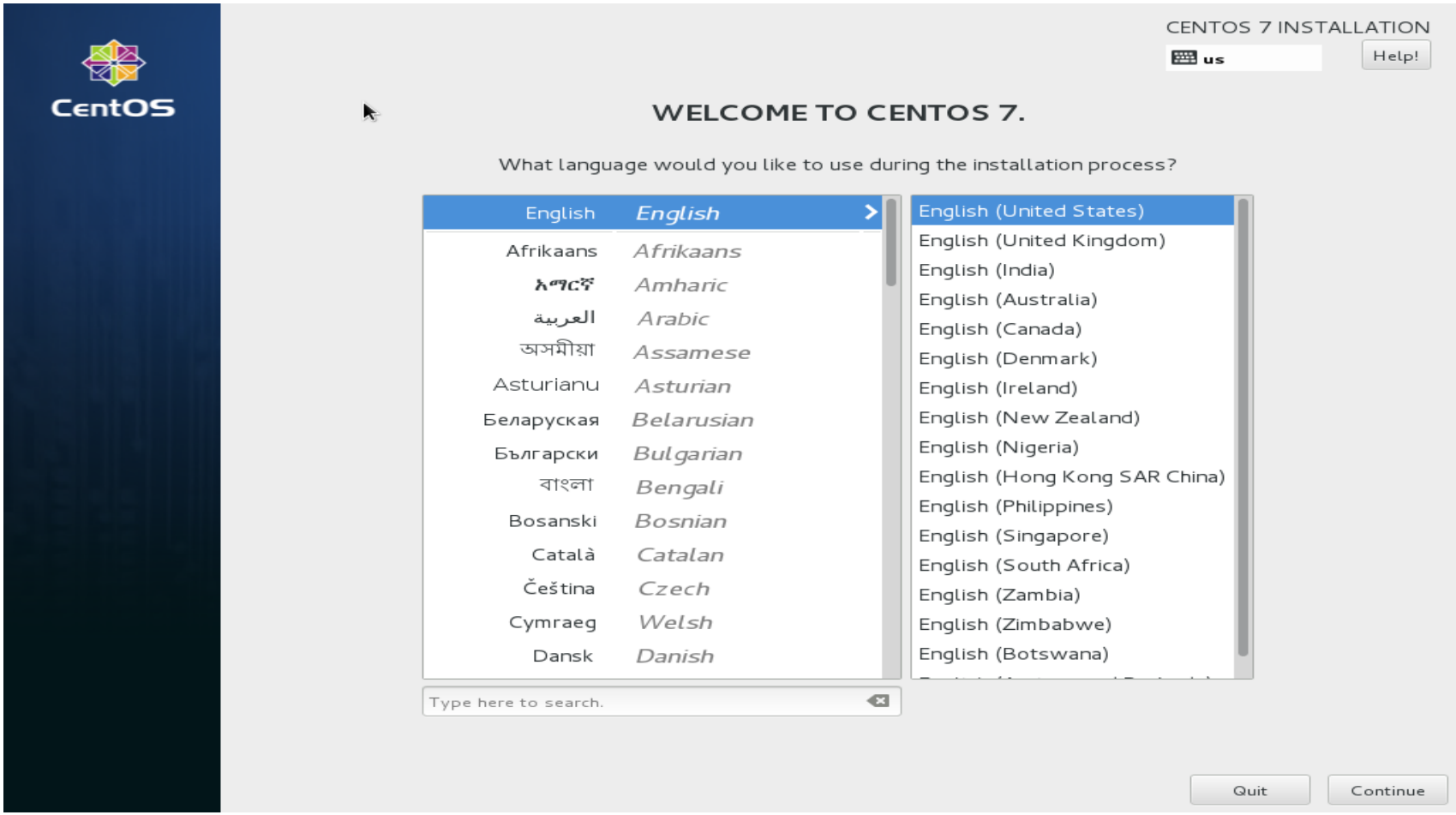
2. CentOS 7 installing

- The CentOS 7 website: http://www.centos.org/
- Starting the virtual machine for the first time.



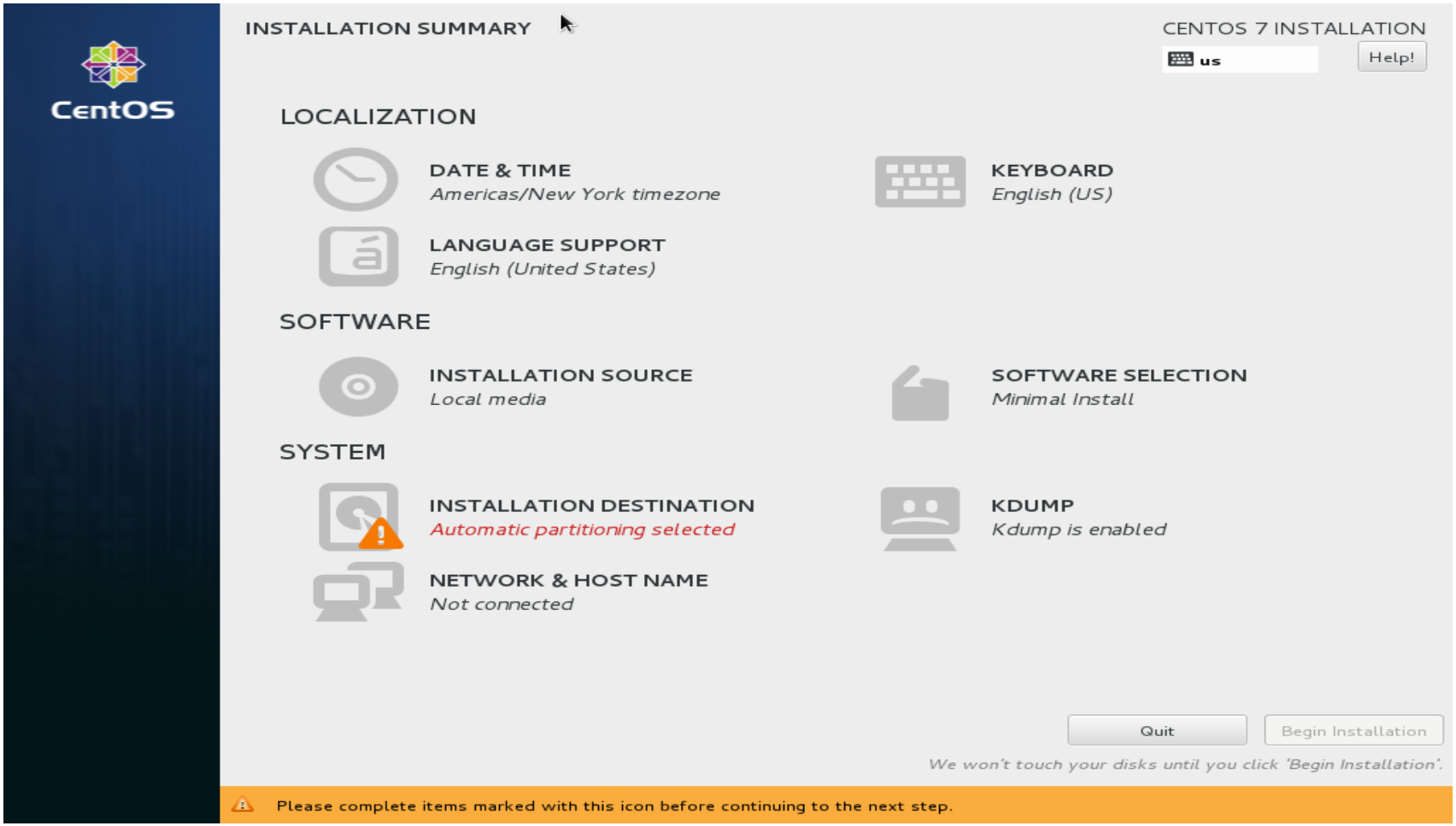
2. CentOS 7 installing (cont)

- Select the language and also select the keyboard.



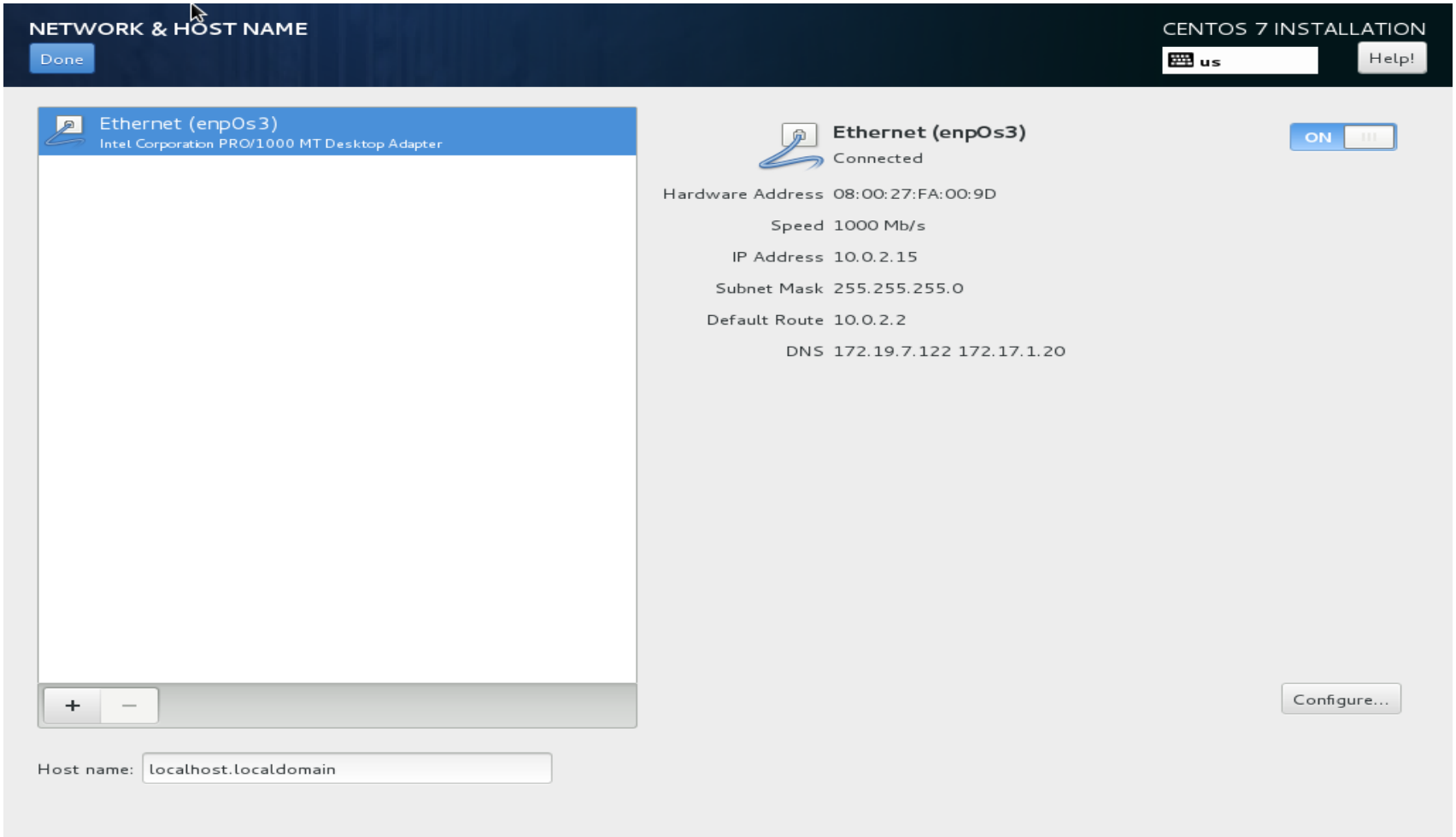
2. CentOS 7 installing (cont)

- At a summary page (with one or more warnings).



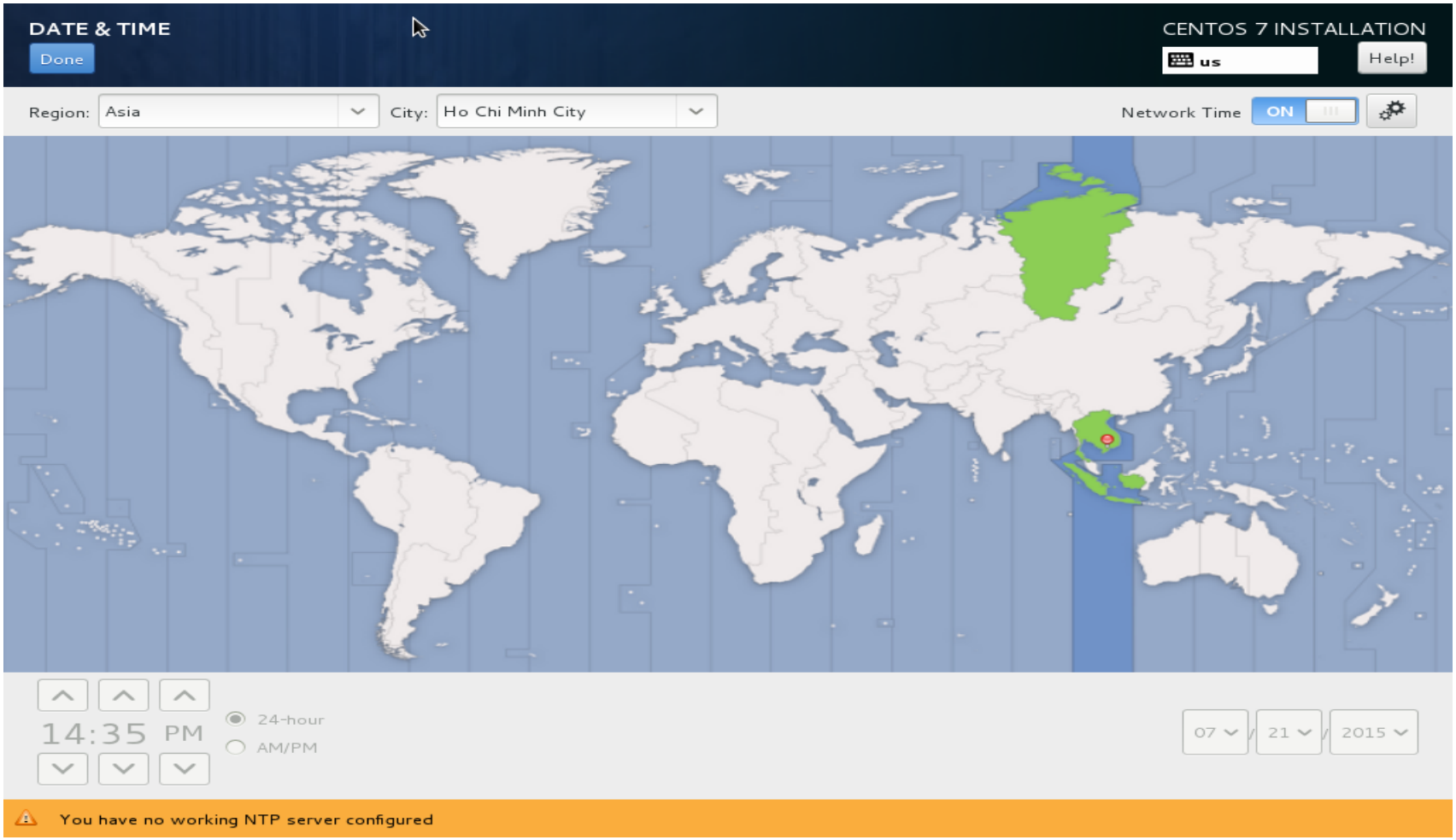
2. CentOS 7 installing (cont)

- Start by configuring the network.



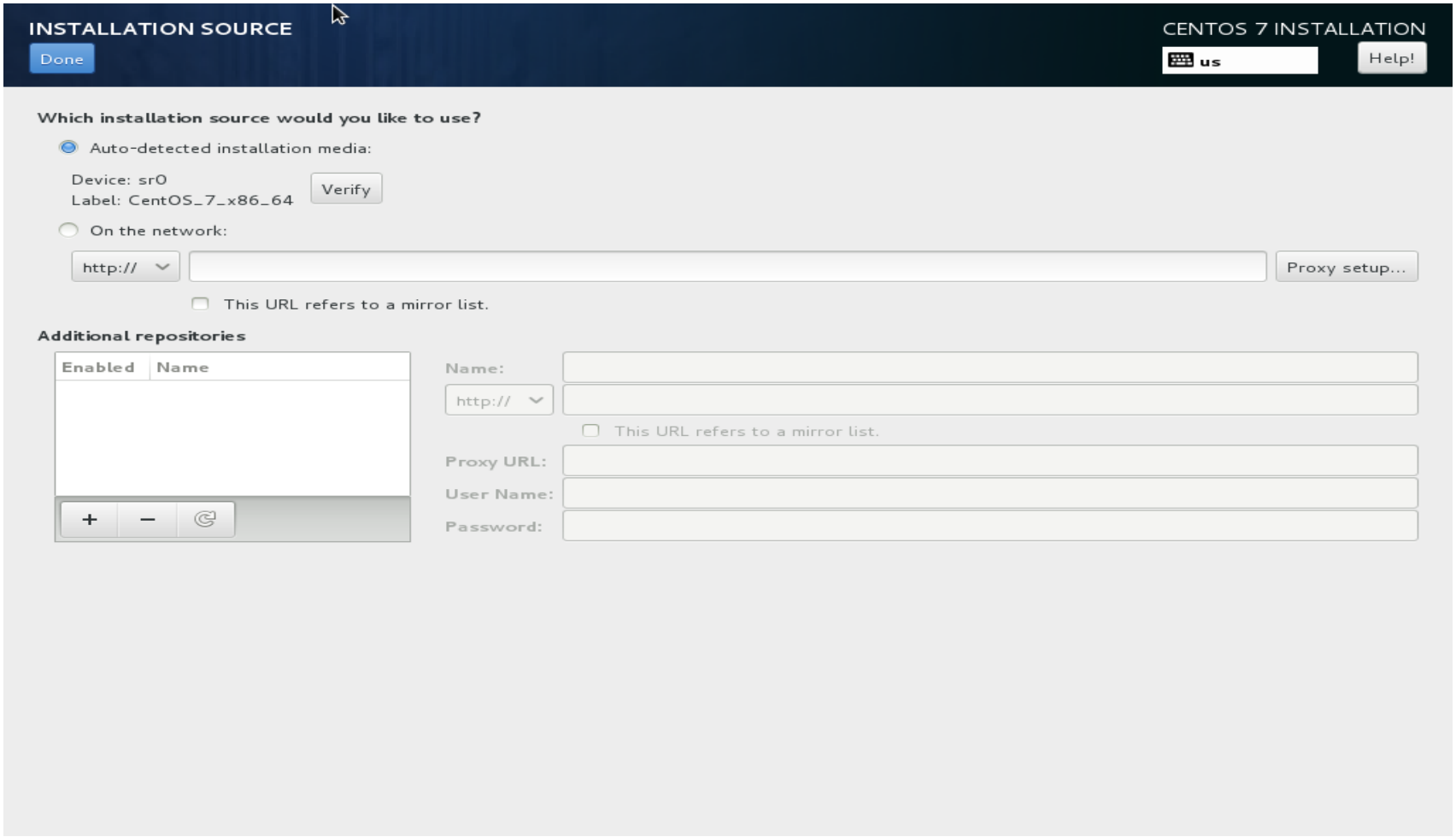
2. CentOS 7 installing (cont)

- Select your time zone, and activate ntp.



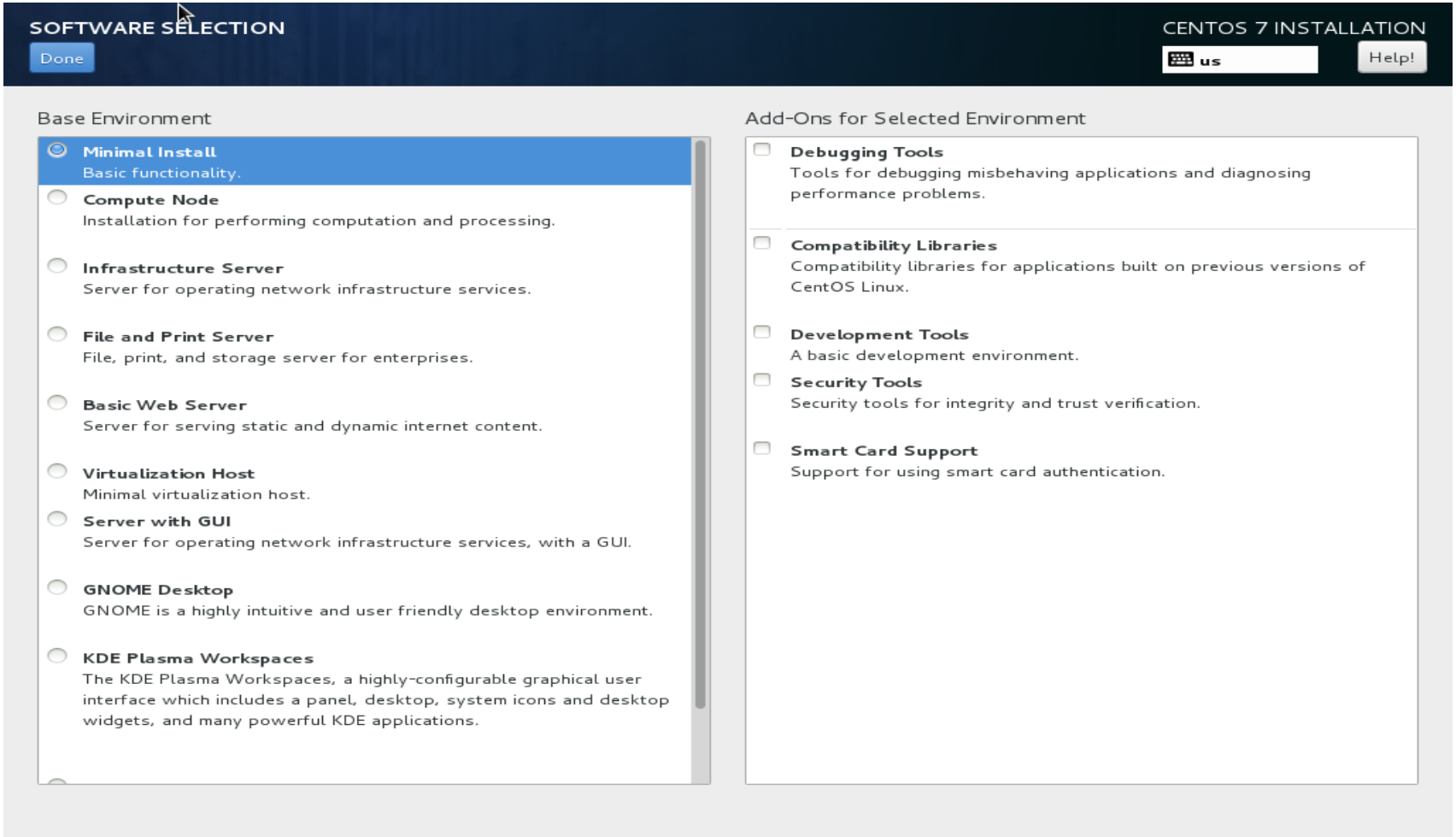
2. CentOS 7 installing (cont)

- Choose a mirror that is close to you.



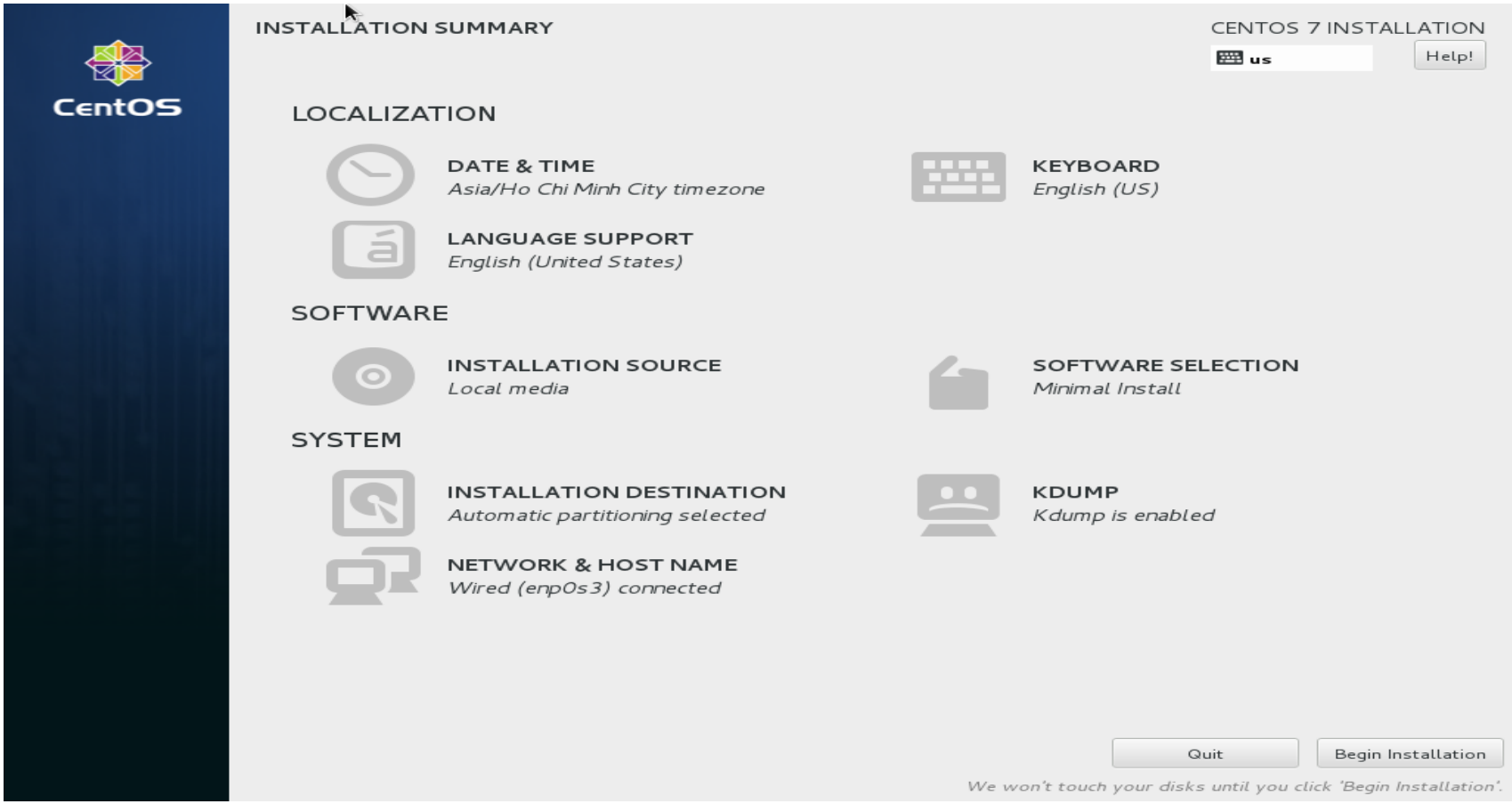
2. CentOS 7 installing (cont)

- Select softwares.



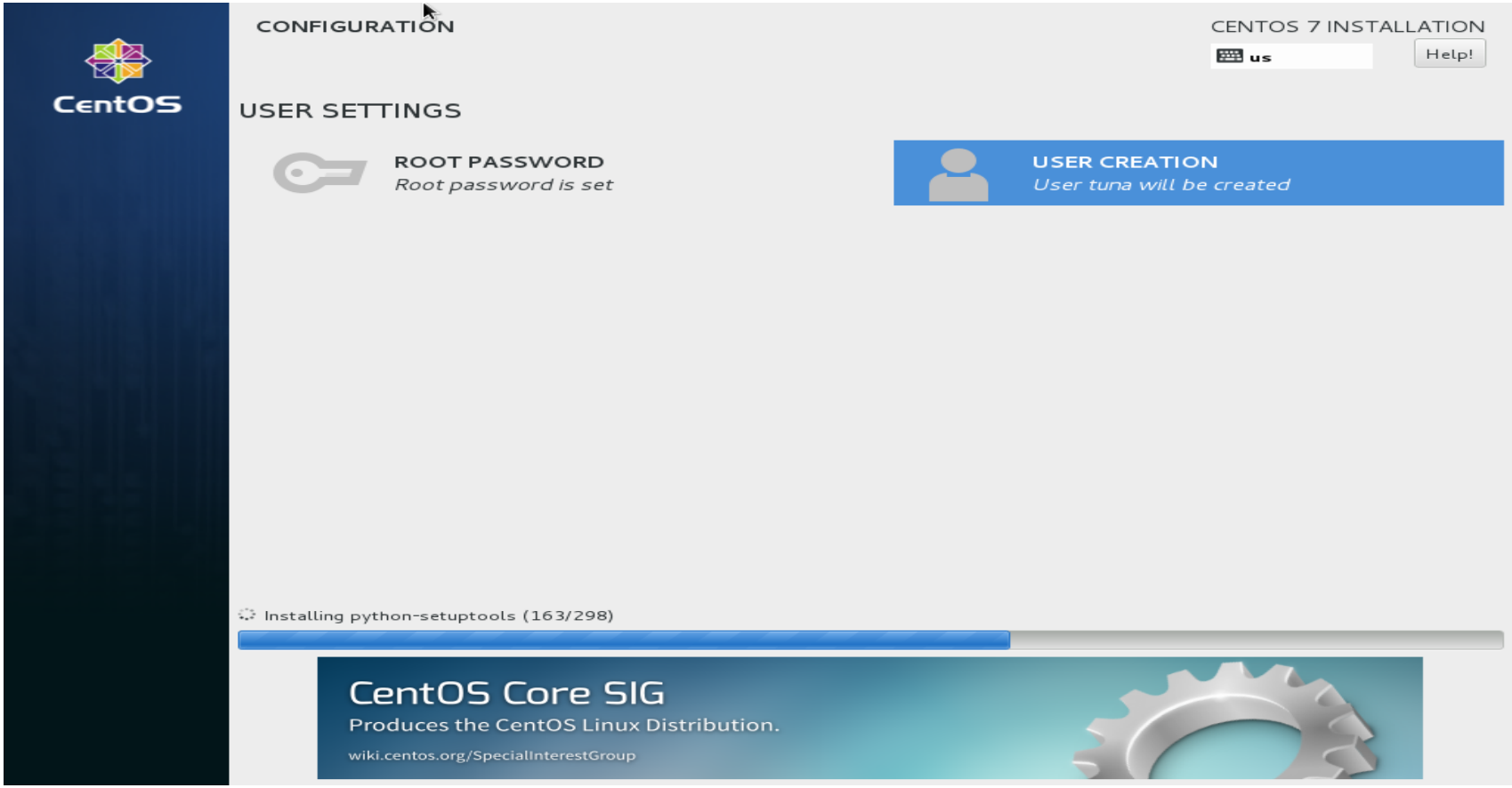
2. CentOS 7 installing (cont)

- After configuring network, location, software and all, should be back on this page.



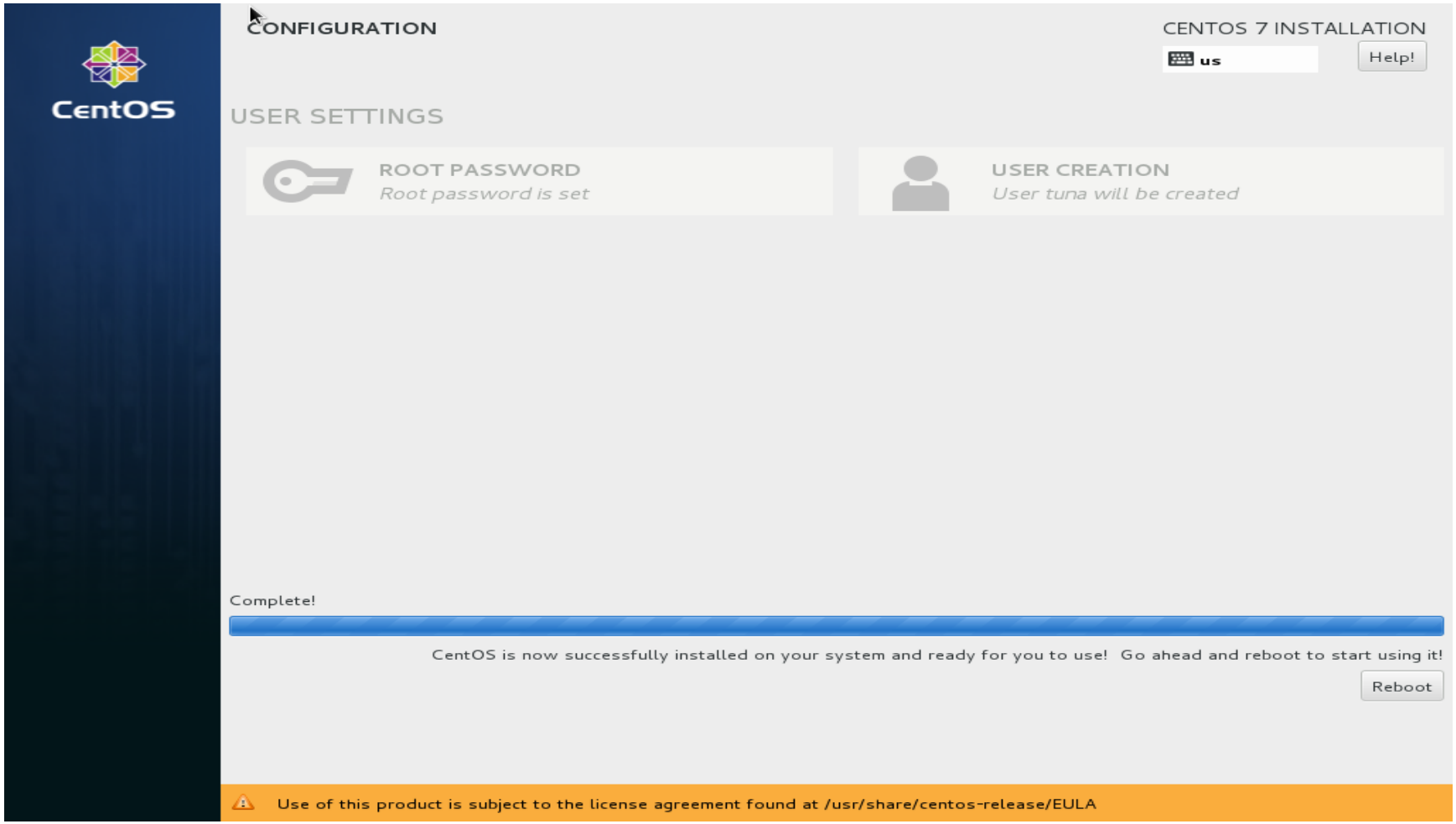
2. CentOS 7 installing (cont)

- Can enter a root password and create a user account while starting package installation process.



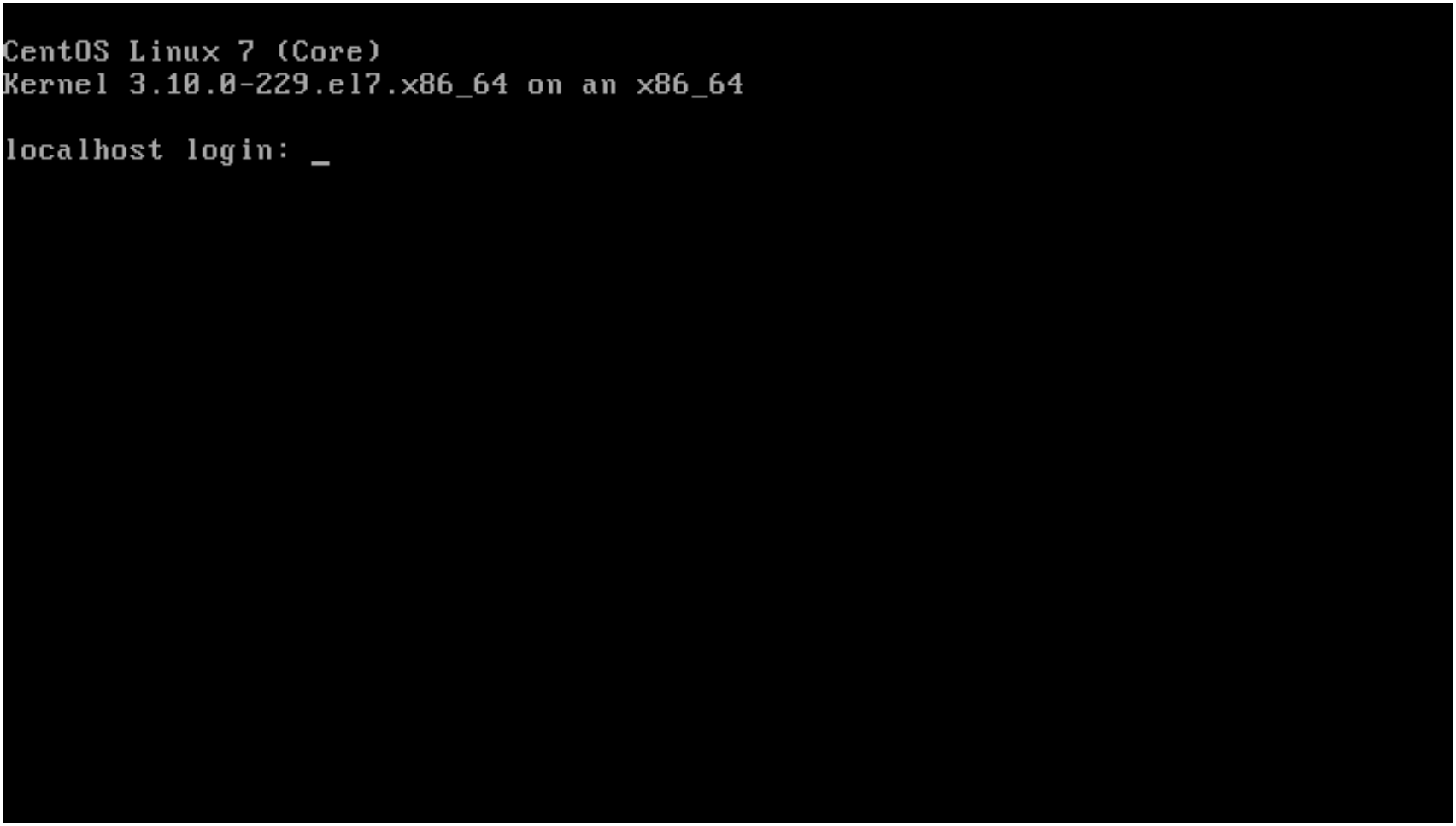
2. CentOS 7 installing (cont)

- The installation was successful.



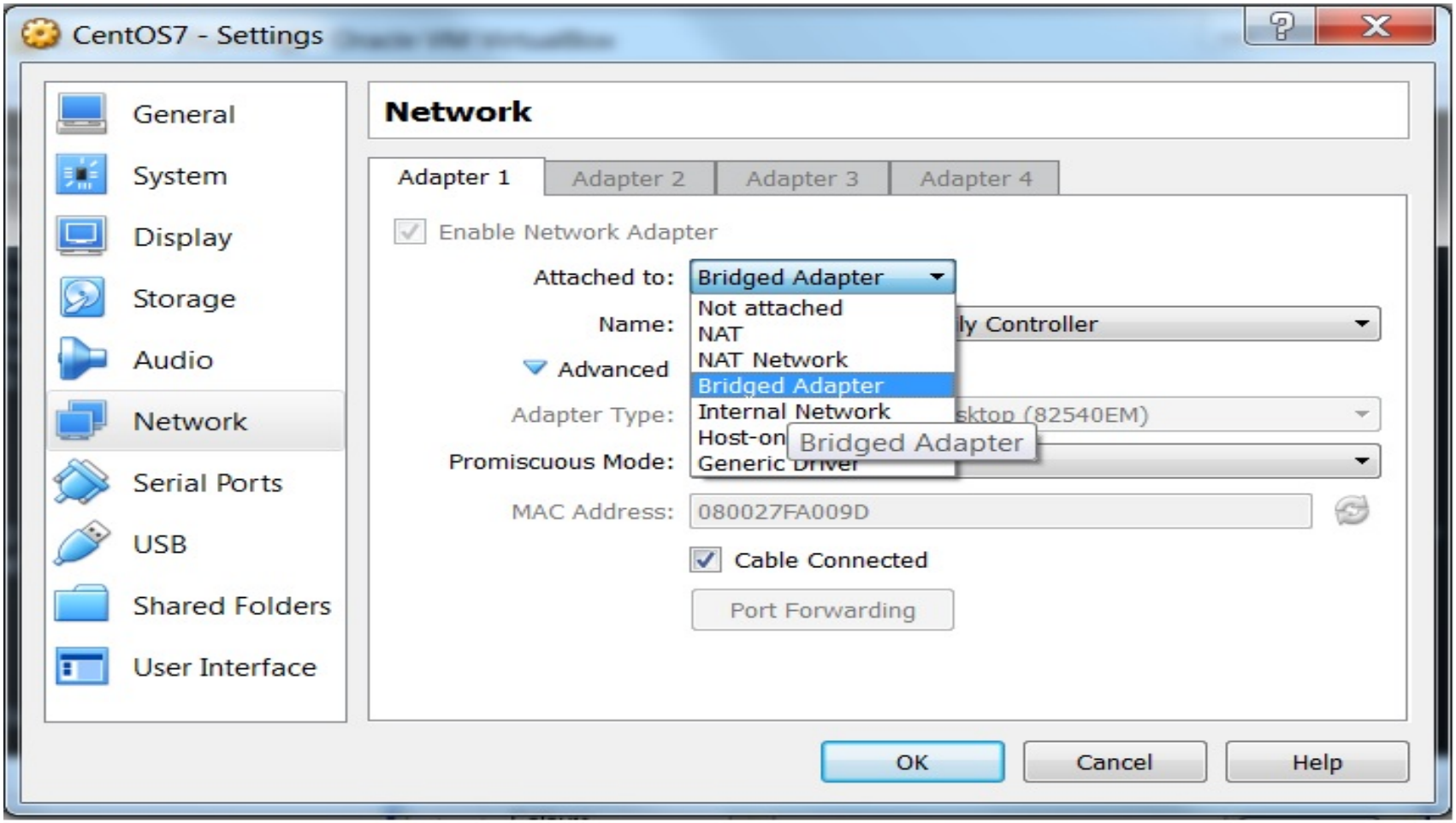
2. CentOS 7 installing (cont)

- CentOS 7 first logon.



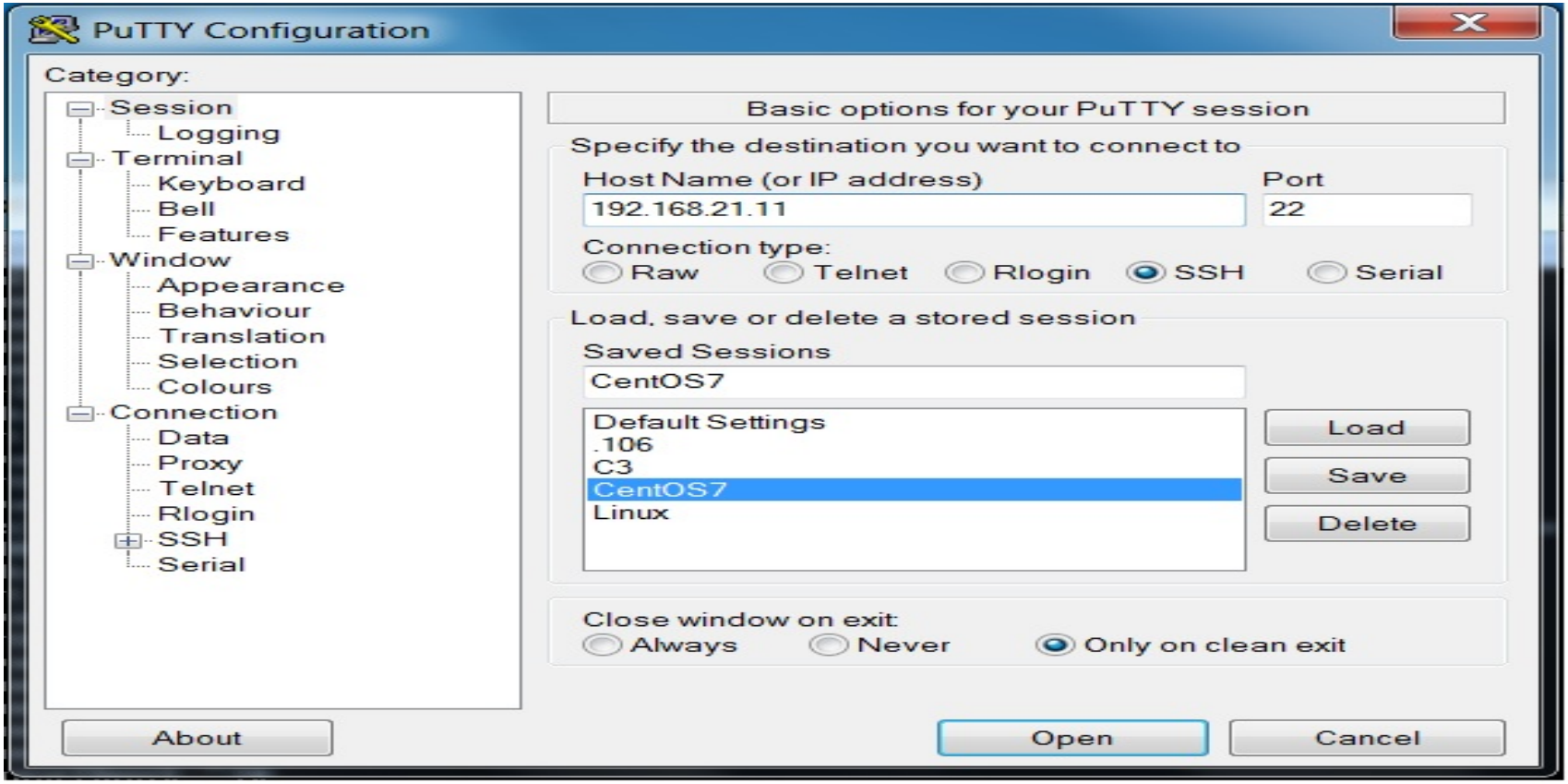
2. CentOS 7 installing (cont)

- Change Virtualbox network interface to bridge.



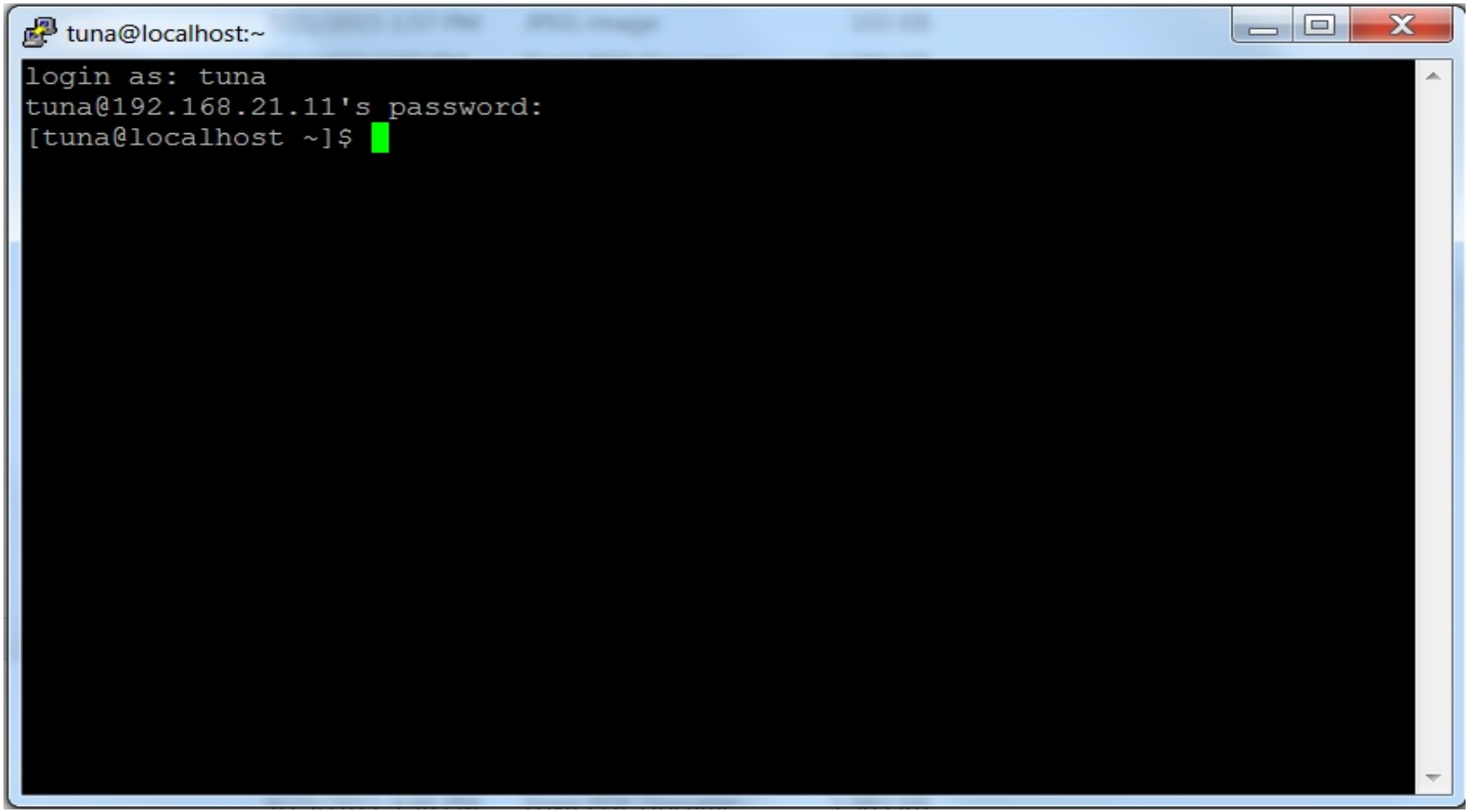
2. CentOS 7 installing (cont)

- Logon from MS Windows
  - Download putty.exe from http://www.putty.org/



2. CentOS 7 installing (cont)

- Enter your userid (or root) and the correct password.



III. First steps on the command line

1. man pages
2. Working with directories
3. Working with files
4. Working with file contents
5. The Linux file tree

1. man pages (also called manual pages)

- man $command
  - ✓ Type man followed by a command (for which you want to help) and start reading.
  - ✓ Press q to quit the manpage.

  [tuna@localhost ~]$ man ls

  [tuna@localhost ~]$ man pwd

- man $configfile
  - ✓ Most configuration files have their own manual.

  [tuna@localhost ~]$ man yum.conf

  [tuna@localhost ~]$ man rsyslog.conf

1. man pages (cont)

- whatis
  - ✓ To see just the description of a manual page, use whatis followed by a string.

    [tuna@localhost ~]$ whatis route

    [tuna@localhost ~]$ whatis ps

- whereis
  - ✓ The location of a manpage can be revealed with whereis.

    [tuna@localhost ~]$ whereis ps

    [tuna@localhost ~]$ whereis ls

2. Working with directories

- pwd
  - ✓ The “you are here sign” can be displayed with the pwd command (Print Working Directory).

    [tuna@localhost ~]$ pwd

- cd
  - ✓ Can change the current directory with the cd command (Change Directory).

    [tuna@localhost ~]$ cd /var/log/

  - ✓ cd ~

    The cd is also a shortcut to get back into your home directory, typing cd ~ has the same effect.

2. Working with directories (cont)

- cd (cont)
  - ✓ cd ..

    To go to the parent directory, type cd ..

    [tuna@localhost var]$ cd /var/log/

    [tuna@localhost log]$ pwd

    [tuna@localhost log]$ cd ..

    [tuna@localhost var]$ pwd

  - ✓ cd –

    Another useful shortcut with cd is to just type cd - to go to the previous directory.

    [tuna@localhost var]$ cd -

2. Working with directories (cont)

- Absolute and relative paths
  - ✓ When typing a path starting with a slash (/), then the root of the file tree is assumed.
  - ✓ If not start the path with a slash, then the current directory is the assumed starting point.

    [tuna@localhost log]$ pwd

    [tuna@localhost log]$ cd home

    [tuna@localhost log]$ cd /home/

    [tuna@localhost home]$ pwd

- path completion
  - ✓ The tab key can help typing a path without errors.

2. Working with directories (cont)

- ls
  - ✓ List the contents of a directory with ls.
  - ✓ ls –a

    A frequently used option with ls is -a to show all files.

    [tuna@localhost ~]$ ls –a

  - ✓ ls –l

    Typing ls -l gives you a long listing.

    [tuna@localhost ~]$ ls -l

  - ✓ ls –lh

    It shows the numbers (file sizes) in a more human readable format.

    [tuna@localhost ~]$ ls -lh

2. Working with directories (cont)

- mkdir
  - ✓ Create your own directories with mkdir.

[tuna@localhost ~]$ mkdir mydir

[tuna@localhost ~]$ cd mydir/

  - ✓ mkdir –p
    - The following command will fail, because the parent directory of threedirsdeep does not exist.

mkdir mydir2/mysubdir2/threedirsdeep

    - *When given the option -p, then mkdir will create parent directories as needed.*

mkdir -p mydir2/mysubdir2/threedirsdeep

2. Working with directories (cont)

- rmdir
  - ✓ When a directory is empty, can use rmdir to remove the directory.

    [tuna@localhost ~]$ rmdir mydir

  - ✓ rmdir –p

    And similar to the mkdir -p option, can also use rmdir to recursively remove directories.

    [tuna@localhost ~]$ mkdir -p test/subdir

    [tuna@localhost ~]$ rmdir -p test/subdir

3. Working with files

- All files are case sensitive
  - ✓ Files on Linux (or any Unix) are case sensitive

    [tuna@localhost ~]$ cat summer.txt

    [tuna@localhost ~]$ cat Summer.txt

- Everything is a file
  - ✓ A directory is a special kind of file, but it is still a (case sensitive!) file.
  - ✓ Each terminal window (for example /dev/pts/4), any hard disk or partition (for example /dev/sdb1) and any process are all represented somewhere in the file system as a file.

3. Working with files (cont)

- File
  - ✓ Linux does not use extensions to determine the file type. The file utility determines the file type.

    [tuna@localhost ~]$ file Downloads/3420.pdf

  - ✓ It is interesting to point out file -s for special files like those in /dev and /proc.

    [root@localhost ~]# file /dev/sda

    [root@localhost ~]# file -s /dev/sda

3. Working with files (cont)

- touch
  - ✓ Create an empty file.

    [tuna@localhost ~]$ touch test.txt temp.txt

  - ✓ touch –t
    - The touch command can set some properties while creating empty files.

    [tuna@localhost ~]$ touch -t 200505051200 oldfile.txt

    [tuna@localhost ~]$ ll

3. Working with files (cont)

- rm
  - ✓ Remove forever
    - When no longer need a file, use rm to remove it.
  - ✓ rm –i
    - To prevent yourself from accidentally removing a file, type rm -i.

      [tuna@localhost ~]$ rm -i temp.txt
  - ✓ rm –rf
    - By default, rm -r will not remove non-empty directories.
    - The rm -rf statement is famous because it will erase anything.

      [tuna@localhost ~]$ rm -rf test

3. Working with files (cont)

- cp
  - ✓ To copy a file, use cp with a source and a target argument.
    [tuna@localhost ~]$ cp temp.txt cptemp.txt
  - ✓ cp –r
    To copy complete directories, use *cp -r*.
    [tuna@localhost ~]$ cp Downloads/ newdir
    [tuna@localhost ~]$ cp -r Downloads/ newdir
  - ✓ Can also use cp to copy multiple files into a directory.
    [tuna@localhost ~]$ cp oldfile.txt temp.txt Summer.txt newdir/
  - ✓ cp –i
    To prevent cp from overwriting existing files, use the -i option.

3. Working with files (cont)

- mv
  - ✓ Rename files with mv

    Use mv to rename a file or to move the file to another directory.

    [tuna@localhost ~]$ mv oldfile.txt newfile.txt

  - ✓ Rename directories with mv

    The same mv command can be used to rename directories.

    [tuna@localhost ~]$ mv newdir olddir

  - ✓ mv –i

    The mv also has a -i switch similar to cp and rm.

    [tuna@localhost ~]$ mv -i temp.txt Summer.txt

4. Working with file contents

- head
  - ✓ Use head to display the first ten lines of a file.

    [tuna@localhost ~]$ head /etc/passwd

  - ✓ The head command can also display the first n lines of a file.

    [tuna@localhost ~]$ head -5 /etc/passwd

  - ✓ And head can also display the first n bytes.

    [tuna@localhost ~]$ head -c10 /etc/passwd

- tail
  - ✓ Similar to head, the tail command will display the last ten lines of a file.
  - ✓ And can give tail the number of lines that want to see.

4. Working with file contents (cont)

- cat
  - ✓ The cat command is one of the most universal tools, yet all it does is copy standard input to standard output.

    [tuna@localhost ~]$ cat /etc/resolv.conf

    [root@localhost ~]# cat /var/log/messages

  - ✓ Concatenate

    One of the basic uses of cat is to concatenate files into a bigger (or complete) file.

    [tuna@localhost ~]$ cat part1 part2 part3 > all

  - ✓ Create files

    [tuna@localhost ~]$ cat > winter.txt

    - *The Ctrl d key combination will send an EOF to the running process ending the cat command.*

4. Working with file contents (cont)

- cat (cont)
  - ✓ Custom end marker

    [tuna@localhost ~]$ cat > hot.txt <<stop

  - ✓ Copy files

    [tuna@localhost ~]$ cat hot.txt > summer.txt

- tac
  - ✓ tac (cat backwards)

    [tuna@localhost ~]$ cat count

    [tuna@localhost ~]$ tac count

- more and less
  - ✓ Displaying files that more than one screen.

5. The Linux file tree

- The root directory /
  - ✓ All Linux systems have a directory structure that starts at the root directory.
  - ✓ The root directory is represented by a forward slash, like this: */*.

    [tuna@localhost ~]$ ls /

    bin dev home lib64 mnt proc run srv tmp var

    boot etc lib media opt root sbin sys usr

  - Binary directories
  - ✓ Binaries are files that contain compiled source code (or machine code)
  - ✓ Binaries can be executed on the computer
  - ✓ Sometimes binaries are called executables.

5. The Linux file tree (cont)

- Binary directories (cont)

✓ /bin

- The /bin directory contains binaries for use by all users

[tuna@localhost ~]$ ls /bin

✓ Other /bin directories

- Can find a /bin subdirectory in many other directories. A user named tuna could put his own programs in /home/tuna/bin.

5. The Linux file tree (cont)

- Binary directories (cont)

✓ /sbin

- /sbin contains binaries to configure the operating system

[tuna@localhost ~]$ ls -l /sbin/ifconfig /sbin/fdisk

✓ /lib

- Binaries found in /bin and /sbin often use shared libraries located in /lib

✓ /lib32 and /lib64

- We are currently in a transition between 32-bit and 64-bit.

✓ /opt

- The purpose of /opt is to store optional software.

5. The Linux file tree (cont)

- Configuration directories
  - ✓ /boot
    - The /boot directory contains all files needed to boot the computer.
  - ✓ /etc
    - All of the machine-specific configuration files should be located in /etc (Editable Text Configuration).

      [tuna@localhost ~]$ ls /etc/*.conf
  - ✓ /etc/init.d/
    - A lot of Unix/Linux distributions have an /etc/init.d directory that contains scripts to start and stop daemons.

5. The Linux file tree (cont)

- Configuration directories (cont)
  - ✓ /etc/X11/
    - The graphical display (aka X Window System or just X) is driven by software from the X.org foundation.
  - ✓ /etc/skel/
    - The skeleton directory /etc/skel is copied to the home directory of a newly created user. It usually contains hidden files like a .bashrc script.
  - ✓ /etc/sysconfig/
    - This directory contains a lot of Red Hat Enterprise Linux configuration files.

5. The Linux file tree (cont)

- Data directories
  - ✓ /home
    - Users can store personal or project data under /home.
  - ✓ /root
    - On many systems /root is the default location for personal data and profile of the root user.
  - ✓ /srv
    - You may use /srv for data that is served by your system.

5. The Linux file tree (cont)

- Data directories (cont)
  - ✓ /media
    - The /media directory serves as a mount point for removable media devices such as CD-ROM's, digital cameras, and various usb-attached devices.
  - ✓ /mnt
    - The /mnt directory should be empty and should only be used for temporary mount points.
  - ✓ /tmp
    - Applications and users should use /tmp to store temporary data when needed.
    - Data stored in /tmp may use either disk space or RAM. Both of which are managed by the operating system.

5. The Linux file tree (cont)

- In memory directories
  - ✓ /dev
    - The /dev directory is populated with files as the kernel is recognizing hardware.
  - ✓ /proc conversation with the kernel
    - /proc is another special directory, appearing to be ordinary files, but not taking up disk space. It is actually a view of the kernel, or better, what the kernel manages, and is a means to interact with it directly. /proc is a proc filesystem.
  - ✓ /sys Linux 2.6 hot plugging
    - Linux uses sysfs to support usb and IEEE 1394 (FireWire) hot plug devices.
    - Basically the /sys directory contains kernel information about hardware.

5. The Linux file tree (cont)

- /usr Unix System Resources
  - ✓ The /usr hierarchy should contain shareable, read only data.
  - ✓ /usr/bin
    - The /usr/bin directory contains a lot of commands.

      [tuna@localhost ~]$ ls /usr/bin | wc -l
  - ✓ /usr/include
    - The /usr/include directory contains general use include files for C.
  - ✓ /usr/lib
    - The /usr/lib directory contains libraries that are not directly executed by users or scripts.

5. The Linux file tree (cont)

- /usr Unix System Resources (cont)
  - ✓ /usr/local
    - The /usr/local directory can be used by an administrator to install software locally.
  - ✓ /usr/share
    - The /usr/share directory contains architecture independent data. As you can see, this is a fairly large directory.
  - ✓ /usr/src
    - The /usr/src directory is the recommended location for kernel source files.

5. The Linux file tree (cont)

- /var variable data
  - ✓ /var/log
    - The /var/log directory serves as a central point to contain all log files.
  - ✓ /var/log/messages
    - A typical first file to check when troubleshooting on Red Hat (and derivatives) is the /var/log/messages file. By default this file will contain information on what just happened to the system.
    - The file is called /var/log/syslog on Debian and Ubuntu.

5. The Linux file tree (cont)

- /var variable data (cont)
  - ✓ /var/cache
    - The /var/cache directory can contain cache data for several applications.
  - ✓ /var/spool
    - The /var/spool directory typically contains spool directories for mail and cron, but also serves as a parent directory for other spool files (for example print spool files).
  - ✓ /var/lib
    - The /var/lib directory contains application state information.

IV. Shell expansion

1. Commands and arguments
2. Control operators
3. Shell variables
4. Shell history
5. File globing

1. Commands and arguments

- Arguments
  - ✓ One of the primary features of a shell is to perform a command line scan, cutting it up in arguments.

- White space removal
  - ✓ Parts that are separated by one or more consecutive white spaces (or tabs) are considered separate arguments
  - ✓ Any white space is removed.

[tuna@localhost ~]$ echo Hello World

[tuna@localhost ~]$ echo Hello World

- Single quotes
  - ✓ Prevent the removal of white spaces by quoting the spaces.

1. Commands and arguments (cont)

- Double quotes
  - ✓ Prevent the removal of white spaces by double quoting the spaces.

    [tuna@localhost ~]$ echo 'A line with single quotes'

    [tuna@localhost ~]$ echo "A line with double quotes"

- echo and quotes
  - ✓ Quoted lines can include special escaped characters recognised by the echo command (when using echo -e).

    [tuna@localhost ~]$ echo -e 'A line with \na newline'

    [tuna@localhost ~]$ echo -e "A line with \ta tab"

1. Commands and arguments (cont)

- Commands
  - ✓ External or built-in commands?
    - Not all commands are external to the shell, some are builtin.
  - ✓ type
    - To find out whether an external command or a built-in command.

      [tuna@localhost ~]$ type cd ps ls
  - ✓ which
    - The which command will search for binaries in the $PATH environment variable.

      [tuna@localhost ~]$ which cp ls cd mkdir pwd

1. Commands and arguments (cont)

- Aliases
  - ✓ Used to create an easier to remember name for an existing command or to easily supply parameters.
  - ✓ Create an alias

    [tuna@localhost ~]$ alias dog=tac

    [tuna@localhost ~]$ dog count.txt

  - ✓ Default options

    [tuna@localhost ~]$ alias rm='rm -i'

    [tuna@localhost ~]$ rm winter.txt

  - ✓ Viewing aliases

    [tuna@localhost ~]$ alias

    [tuna@localhost ~]$ alias dog rm

1. Commands and arguments (cont)

- Unalias
  - ✓ Undo an alias with the unalias command.

    [tuna@localhost ~]$ type rm

    [tuna@localhost ~]$ unalias rm

    [tuna@localhost ~]$ type rm

2. Control operators

- ; semicolon
  - ✓ Put two or more commands on the same line separated by a semicolon ;

    [tuna@localhost ~]$ echo Hello ; echo World

- & ampersand
  - ✓ When a line ends with an ampersand &, the command is executed in background

    [tuna@localhost ~]$ sleep 20 &

2. Control operators (cont)

- $? dollar question mark
  - ✓ The exit code of the previous command is stored in the shell variable $?.

```
[tuna@localhost ~]$ touch file
[tuna@localhost ~]$ echo $?
0
[tuna@localhost ~]$ rm file
[tuna@localhost ~]$ echo $?
0
[tuna@localhost ~]$ rm file
rm: cannot remove ‘file’: No such file or directory
[tuna@localhost ~]$ echo $?
1
```

2. Control operators (cont)

- && double ampersand
  - ✓ The shell will interpret && as a logical AND
  - ✓ When using && the second command is executed only if the first one succeeds.

    [tuna@localhost ~]$ echo first && echo second

    [tuna@localhost ~]$ zecho first && echo second

- || double vertical bar
  - ✓ The || represents a logical OR
  - ✓ The second command is executed only when the first command fails

    [tuna@localhost ~]$ echo first || echo second ; echo third

    [tuna@localhost ~]$ zecho first || echo second ; echo third

2. Control operators (cont)

- combining && and ||
  - ✓ Use this logical AND and logical OR to write an if-then-else structure on the command line.

    [tuna@localhost ~]$ touch file

    [tuna@localhost ~]$ rm file && echo It worked! || echo It
    [tuna@localhost ~]$ rm file && echo It worked! || echo It It failed!

- # pound sign

  Everything written after a pound sign (#) is ignored by the shell.

  [tuna@localhost ~]$ mkdir test *# we create a directory*

2. Control operators (cont)

- \ escaping special characters
  - ✓ The backslash \ character enables the use of control characters, but without the shell interpreting it, this is called escaping characters.

    [tuna@localhost ~]$ echo escaping \\\ \#\ \&\ \"\ \'

    - End of line backslash
  - ✓ Lines ending in a backslash are continued on the next line.

    ***[tuna@localhost ~]$ echo This command line ***

    ***> is splitted in three ***

    > parts

3. shell variables

- $ dollar sign
  - ✓ Another important character interpreted by the shell is the dollar sign $.
  - ✓ The shell will look for an environment variable named like the string following the dollar sign and replace it with the value of the variable.
  - ✓ Some examples using $HOSTNAME, $USER, $UID, $SHELL, and $HOME.

    [tuna@localhost~]$ echo This is $SHELL on computer $HOSTNAME

Creating variables

- ✓ This example creates the variable $MyVar and sets its value.

  [tuna@localhost ~]$ MyVar=555

  [tuna@localhost ~]$ echo $MyVar

3. Shell variables (cont)

- Quotes
  - ✓ Notice that double quotes still allow the parsing of variables, whereas single quotes prevent this.

    [tuna@localhost ~]$ city=Hochiminh

    [tuna@localhost ~]$ echo "I live in $city now"

    [tuna@localhost ~]$ echo 'I live in $city now'

- set
  - ✓ Use the set command to display a list of environment variables.
- unset
  - ✓ Use the unset command to remove a variable from your shell environment.

    [tuna@localhost ~]$ unset MyVar

3. Shell variables (cont)

- $PATH
  - ✓ The $PATH variable is determines where the shell is looking for commands to execute (unless the command is builtin or aliased).
  - ✓ This variable contains a list of directories, separated by colons.

    [tuna@localhost ~]$ echo $PATH

  - ✓ The shell will not look in the current directory for commands to execute!
  - ✓ If you want the shell to look in the current directory, then add a . at the end of your $PATH.

    [tuna@localhost ~]$ PATH=$PATH:.

4. Shell history

- Repeating the last command
  - ✓ To repeat the last command in bash, type !!. This is pronounced as bang bang.

    [tuna@localhost ~]$ ls -a

    [tuna@localhost ~]$!!

- Repeating other commands
  - ✓ You can repeat other commands using one bang followed by one or more characters.

    [tuna@localhost ~]$!l

    ls -a

4. Shell history (cont)

- History
  - ✓ To see older commands, use history to display the shell command history (or use history n to see the last n commands).

    [tuna@localhost ~]$ history 10

- !n
  - ✓ When typing ! followed by the number preceding the command you want repeated, then the shell will echo the command and execute it.

    [tuna@localhost ~]$!15

- Ctrl-r
  - ✓ Another option is to use ctrl-r to search in the history.

4. Shell history (cont)

- $HISTSIZE
  - ✓ The $HISTSIZE variable determines the number of commands that will be remembered in your current environment.
  - ✓ Most distributions default this variable to 500 or 1000.

    [tuna@localhost ~]$ echo $HISTSIZE

  - ✓ You can change it to any value you like.

    [tuna@localhost ~]$ HISTSIZE=15000

    [tuna@localhost ~]$ echo $HISTSIZE

5. File globing

- * asterisk
  - ✓ The asterisk * is interpreted by the shell as a sign to generate filenames, matching the asterisk to any combination of characters (even none).
  - ✓ When no path is given, the shell will use filenames in the current directory.

    [tuna@localhost ~]$ touch fileA fileB fileC fileD fileE fileAB

    [tuna@localhost ~]$ ls file*

- ? question mark
  - ✓ Similar to the asterisk, the question mark ? is interpreted by the shell as a sign to generate filenames, matching the question mark with exactly one character.

    [tuna@localhost ~]$ ls file?

5. File globing (cont)

- [] square brackets
  - ✓ The [] is interpreted by the shell as a sign to generate filenames, matching any of the characters between [and the first subsequent]
  - ✓ The order in this list between the brackets is not important
  - ✓ Each pair of brackets is replaced by exactly one character.

    [tuna@localhost ~]$ touch file1 file2 file3 File4 File55 FileA fileab Fileab FileAB fileabc

    [tuna@localhost ~]$ ls File[5A]

    [tuna@localhost ~]$ ls File[A5]

    [tuna@localhost ~]$ ls File[A5][5b]
  - ✓ Can also exclude characters from a list between square brackets with the exclamation mark !. And you are allowed to make combinations of these wild cards.

    [tuna@localhost ~]$ ls file[!5]*

5. File globing (cont)

- a-z and 0-9 ranges
  - ✓ The bash shell will also understand ranges of characters between brackets.

    [tuna@localhost ~]$ ls file[a-z]*

    [tuna@localhost ~]$ ls file[0-9]

- preventing file globing
  - ✓ Globbing can be prevented using quotes or by escaping the special characters

    [tuna@localhost ~]$ echo *

    [tuna@localhost ~]$ echo *

    [tuna@localhost ~]$ echo '*'

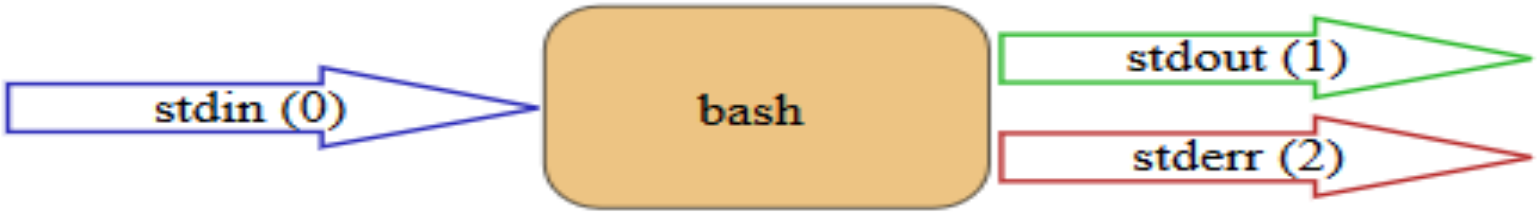
    [tuna@localhost ~]$ echo "*"

V. Pipes and commands

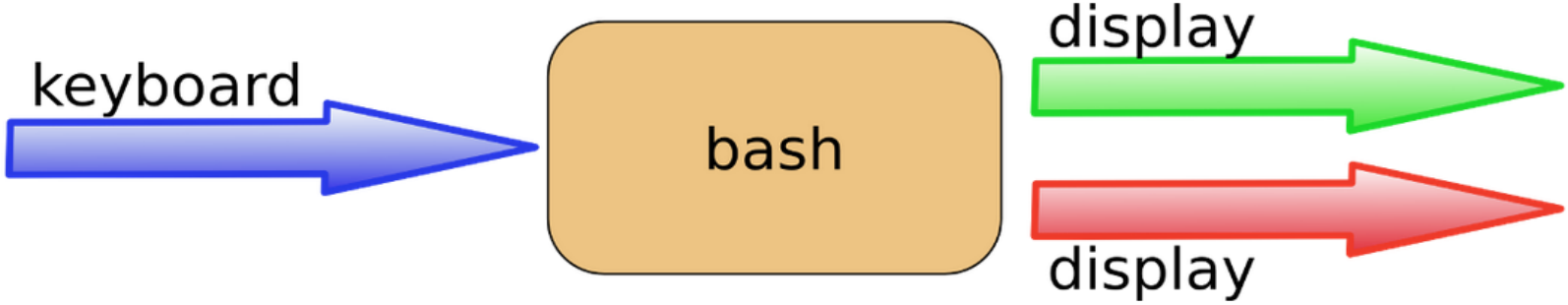
1. I/O redirection
2. Filters
3. Basic Unix tools
4. Regular expressions

1. I/O redirection

- stdin, stdout, and stderr
  - ✓ The bash shell has three basic streams; it takes input from stdin (stream 0), it sends output to stdout (stream 1) and it sends error messages to stderr (stream 2).

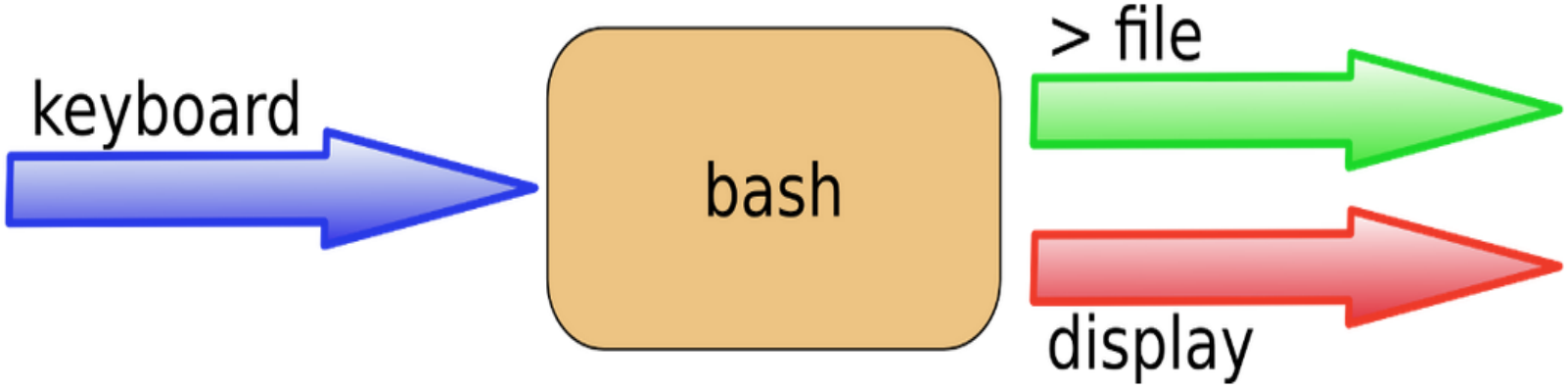


  - ✓ The keyboard often serves as stdin, whereas stdout and stderr both go to the display.



1. I/O redirection (cont)

- Output redirection
  - ✓ > stdout
    - stdout can be redirected with a greater than sign.



- The > notation is in fact the abbreviation of 1> (stdout being referred to as stream 1).

  [tuna@localhost ~]$ echo It is hot today > summer.txt

1. I/O redirection (cont)

- Output redirection (cont)
  - ✓ Output file is erased
    - While scanning the line, the shell will see the > sign and will clear the file!

[tuna@localhost ~]$ cat summer.txt

[tuna@localhost ~]$ zecho It is hot today > summer.txt

[tuna@localhost ~]$ cat summer.txt

  - ✓ noclobber
    - Erasing a file while using > can be prevented by setting the noclobber option.

[tuna@localhost ~]$ set -o noclobber

[tuna@localhost ~]$ echo It is hot today > summer.txt

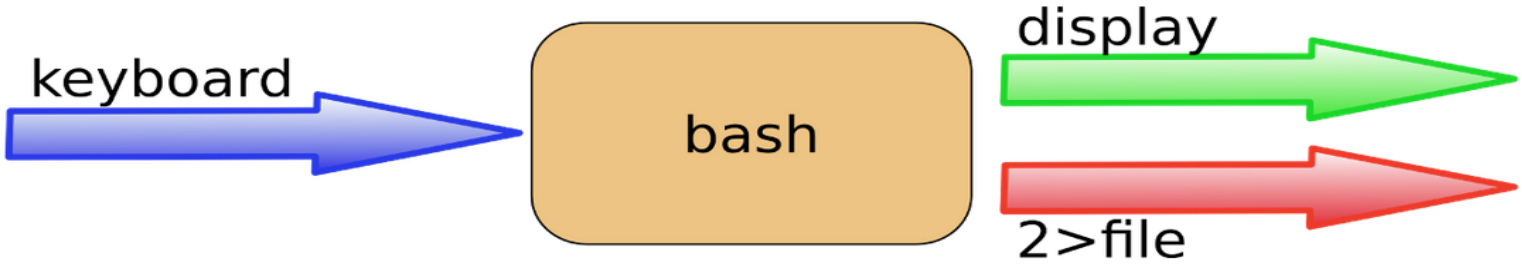
    - The noclobber can be overruled with >|.

1. I/O redirection (cont)

- Output redirection (cont)
  - ✓ >> append
    - Use >> to append output to a file.

[tuna@localhost ~]$ echo Where is the autumn? >> summer.txt

- Error redirection
  - ✓2> stderr
    - Redirecting stderr is done with 2>



[tuna@localhost ~]$ zecho error >> summer.txt 2> error.txt

2. Filters

- grep
  - ✓ The grep filter is famous among Unix users.
  - ✓ The most common use of grep is to filter lines of text containing (or not containing) a certain string.

    [tuna@localhost ~]$ cat tennis.txt

    Amelie Mauresmo, Fra

    Kim Clijsters, BEL

    Justine Henin, bel

    Serena Williams, usa

    Venus Williams, USA

    [tuna@localhost ~]$ cat tennis.txt | grep Williams

  - ✓ You can write this without the cat.

    [tuna@localhost ~]$ grep Williams tennis.txt

2. Filters (cont)

- grep (cont)
  - ✓ One of the most useful options of grep is grep -i which filters in a case insensitive way.

    [tuna@localhost ~]$ grep usa tennis.txt

    [tuna@localhost ~]$ grep -i usa tennis.txt

  - ✓ Another very useful option is grep -v which outputs lines not matching the string.

    [tuna@localhost ~]$ grep -v Fra tennis.txt

  - ✓ And of course, both options can be combined to filter all lines not containing a case insensitive string.

    [tuna@localhost ~]$ grep -vi usa tennis.txt

2. Filters (cont)

- cut
  - ✓ The cut filter can select columns from files, depending on a delimiter or a count of bytes.

    [tuna@localhost ~]$ cut -d: -f1,3 /etc/passwd | tail -5

  - ✓ This example uses cut to display the second to the seventh character of /etc/passwd.

    [tuna@localhost ~]$ cut -c2-7 /etc/passwd | tail -5

- tr
  - ✓ You can translate characters with tr.

    [tuna@localhost ~]$ cat tennis.txt | tr 'e' 'E'

    [tuna@localhost ~]$ cat tennis.txt | tr 'a-z' 'A-Z'

2. Filters (cont)

- wc
  - ✓ Counting lines, words and characters is easy with wc.

    [tuna@localhost ~]$ wc tennis.txt

    [tuna@localhost ~]$ wc -l tennis.txt

- sort
  - ✓ The sort filter will default to an alphabetical sort.

    [tuna@localhost ~]$ cat music.txt

    Queen

    Brel

    Queen

    Abba

    [tuna@localhost ~]$ sort music.txt
  - ✓ With **uniq** you can remove duplicates from a sorted list.

3. Basic Unix tools

- find
  - ✓ The find command can be very useful at the start of a pipe to search for files. Here are some examples.
    - Find all files in /etc and put the list in etcfiles.txt

      [root@localhost ~]# find /etc > etcfiles.txt

    - Find all files of the entire system and put the list in allfiles.txt

      [root@localhost ~]# find / > allfiles.txt

    - Find files that end in .conf in the current and subdirs.

      [root@localhost etc]# find . -name "*.conf"

    - Find files that are newer than file42.txt

      [tuna@localhost ~]$ find . -newer File55

3. Basic Unix tools (cont)

- locate
  - ✓ The locate tool is very different from find in that it uses an index to locate files.
  - ✓ This is a lot faster than traversing all the directories, but it also means that it is always outdated.
  - ✓ If the index does not exist yet, then have to create it (as root) with the updatedb command.

    [tuna@localhost ~]$ locate test_locate

    [root@localhost ~]# updatedb

    [tuna@localhost ~]$ locate test_locate
- date
  - ✓ The date command can display the date, time, time zone and more.

    [tuna@localhost ~]$ date

3. Basic Unix tools (cont)

- cal
  - ✓ The cal command displays the current month, with the current day highlighted.

    [tuna@localhost ~]$ cal

  - ✓ can select any month in the past or the future.

    [tuna@localhost ~]$ cal 1 1900

- sleep
  - ✓ The sleep command is sometimes used in scripts to wait a number of seconds.

    [tuna@localhost ~]$ sleep 5

3. Basic Unix tools (cont)

- gzip – gunzip
  - The gzip command can make files take up less space.

    [tuna@localhost ~]$ ls -lh tennis.txt

    [tuna@localhost ~]$ gzip tennis.txt

    [tuna@localhost ~]$ ls -lh tennis.txt.gz

  - You can get the original back with gunzip.

    [tuna@localhost ~]$ gunzip tennis.txt.gz

- zcat – zmore
  - Text files that are compressed with gzip can be viewed with zcat and zmore.

    [tuna@localhost ~]$ head -3 tennis.txt

    [tuna@localhost ~]$ zcat tennis.txt.gz | head -3

3. Basic Unix tools (cont)

- bzip2 - bunzip2
  - Files can also be compressed with bzip2 (better than gzip)

    [tuna@localhost ~]$ ls -lh tennis.txt

    [tuna@localhost ~]$ bzip2 tennis.txt

    [tuna@localhost ~]$ ls -lh tennis.txt.bz2

  - Files can be uncompressed again with bunzip2.

    [tuna@localhost ~]$ bunzip2 tennis.txt.bz2

- bzcat - bzmore
  - And in the same way bzcat and bzmore can display files compressed with bzip2.

    [tuna@localhost ~]$ bzcat tennis.txt.bz2 | head -3

4. Regular expressions

- Regular expressions are a very powerful tool in Linux.
- They can be used with a variety of programs like bash, vi, rename, grep, sed, and more.
- grep
  - ✓ grep is a popular Linux tool to search for lines that match a certain pattern.
  - ✓ Print lines matching a pattern

[tuna@localhost ~]$ cat names

Tania

Laura

Valentina

[tuna@localhost ~]$ grep u names

[tuna@localhost ~]$ grep e names

[tuna@localhost ~]$ grep i names

4. Regular expressions (cont)

- grep (cont)
  - ✓ Concatenating characters
    - Two concatenated characters will have to be concatenated in the same way to have a match.

      [tuna@localhost ~]$ grep ia names

      [tuna@localhost ~]$ grep in names

  - ✓ Match the end of a string
    - Use the dollar character to match the end of a string.

      [tuna@localhost ~]$ grep ra$ names

  - ✓ Match the start of a string
    - The caret character (^) to match the start of a line.

      [tuna@localhost ~]$ grep ^T names

4. Regular expressions (cont)

- Separating words
  - ✓ Regular expressions use a \b sequence to reference a word separator.

    [tuna@localhost ~]$ cat text

    The governer is governing.

    The winter is over.

    Can you get over there?

  - ✓ Simply grepping for over will give too many results.

    [tuna@localhost ~]$ grep over text

  - ✓ Use \b to find only the searched word:

    [tuna@localhost ~]$ grep '\bover\b' text

  - ✓ grep also has a -w option to grep for words.

VI. vi

Introduction to vi(m)

Introduction to vi(m)

- Command mode and insert mode
  - ✓ The vi editor starts in command mode, In command mode, can type commands. Some commands will bring us into insert mode.
  - ✓ In insert mode, can type text. The escape key will return you to command mode.

Key	Action
Esc	set vi(m) in command mode.

Introduction to vi (cont)

- Start typing (a A i I o O)
  - ✓ The difference between a A i I o and O is the location where you can start typing.

Command	Action
a	start typing after the current character
A	start typing at the end of the current line
i	start typing before the current character
I	start typing at the start of the current line
o	start typing on a new line after the current line
O	start typing on a new line before the current line

Introduction to vi (cont)

- Replace and delete a character (r x X)

Command	Action
x	delete the character below the cursor
X	delete the character before the cursor
r	replace the character below the cursor
p	paste after the cursor (here the last deleted character)
xp	switch two characters

- Undo and repeat (u .)

Command	Action
u	undo the last action
.	repeat the last action

Introduction to vi (cont)

- Cut, copy and paste a line (dd yy p P)

Command	Action
dd	cut the current line
yy	(yank yank) copy the current line
p	paste after the current line
P	paste before the current line

- cut, copy and paste lines (3dd 2yy)

Command	Action
3dd	cut three lines
2yy	copy four lines

Introduction to vi (cont)

- Start and end of a line (0 or ^ and $)

Command	Action
0	jump to start of current line
^	jump to start of current line
$	jump to end of current line
d0	delete until start of line
d$	delete until end of line

- Join two lines (J) and more

Command	Action
J	join two lines
yyp	duplicate a line
ddp	switch two lines

Introduction to vi (cont)

- Words (w b)

Command	Action
w	forward one word
b	back one word
3w	forward three words
dw	delete one word
yw	yank (copy) one word
5yb	yank five words back
7dw	delete seven words

Introduction to vi (cont)

- Save (or not) and exit (:w :q :q!)

command	action
:w	save (write)
:w fname	save as fname
:q	quit
:wq	save and quit
ZZ	save and quit
:q!	quit (discarding your changes)
:w!	save (and write to non-writable file!)

Introduction to vi (cont)

- Searching (/ ?)

Command	Action
/string	forward search for string
?string	backward search for string
n	go to next occurrence of search string
/^string	forward search string at beginning of line
/string$	forward search string at end of line
/br[aeio]l	search for bral brel bril and brol
/\<he\>	search for the word he (and not for here or the)

Introduction to vi (cont)

- Replace all (:1,$ s/foo/bar/g)

Command	Action
:4,8 s/foo/bar/g	replace foo with bar on lines 4 to 8
:1,$ s/foo/bar/g	replace foo with bar on all lines

- Reading files (:r :r !cmd)

Command	Action
:r fname	(read) file fname and paste contents
:r !cmd	execute cmd and paste its output

- Text buffers

Command	Action
"add	delete current line and put text in buffer a
"g7yy	copy seven lines into buffer g
"ap	paste from buffer a

Introduction to vi (cont)

- Multiple files

Command	Action
vi file1 file2 file3	start editing three files
:args	lists files and marks active file
:n	start editing the next file
:e	toggle with last edited file
:rew	rewind file pointer to first file

- Abbreviations

Command	Action
:ab str long string	abbreviate str to be 'long string'
:una str	un-abbreviate str

Introduction to vi (cont)

- Setting options
  - ✓ Some options that you can set in vim.

    :set number (also try :se nu)

    :set nonumber

    :syntax on

    :syntax off

    :set all (list all options)

    :set tabstop=8

    :set tx (CR/LF style endings)

    :set notx

  - ✓ You can set these options (and much more) in ~/.vimrc for vim or in ~/.exrc for standard vi.

VII. Scripting

1. Scripting introduction
2. Scripting loops

1. Scripting introduction

- Hello World
  - ✓ Just like in every programming course, we start with a simple hello_world script.

    [tuna@localhost ~]$ echo echo Hello World > hello_world

    [tuna@localhost ~]$ chmod +x hello_world

    [tuna@localhost ~]$ ./hello_world

- she-bang
  - ✓ A script that works flawlessly in bash might not work in ksh, csh, or dash.
  - ✓ To run the script in a certain shell, start with a she-bang.

    [tuna@localhost ~]$ cat hello_world

    #!/bin/bash

    echo Hello World

1. Sripting introduction (cont)

- Comment
  - ✓ Let's expand our example a little further by adding comment lines.

```
[tuna@localhost ~]$ cat hello_world
#!/bin/bash
#
# Hello World Script
#
echo Hello World
```

1. Scripting introduction (cont)

- Variables
  - ✓ Scripts can contain variables, but since scripts are run in their own shell, the variables do not survive the end of the script.

```
[tuna@localhost ~]$ cat var
#! /bin/bash
#
#simple variable in script
#
var1=5
echo var1 = $var1
[tuna@localhost ~]$ ./var
[tuna@localhost ~]$ echo $var1
```

1. Scripting introduction (cont)

- Troubleshooting a script
  - ✓ Another way to run a script in a separate shell is by typing bash with the name of the script as a parameter.

    [tuna@localhost ~]$ bash var

  - ✓ Expanding this to bash -x allows you to see the commands that the shell is executing (after shell expansion).

    [tuna@localhost ~]$ bash -x var

    + var1=5

    + echo var1 = 5

    var1 = 5

2. Scripting loops

- test [ ]
  - ✓ The test command can test whether something is true or false.

    [tuna@localhost ~]$ test 10 -gt 55 ; echo $?

    1

  - ✓ The test command returns 1 if the test fails. Test returns 0 when a test succeeds.

    [tuna@localhost ~]$ test 56 -gt 55 ; echo $?

    0

  - ✓ If prefer true and false, then write the test like this.

    [tuna@localhost ~]$ test 56 -gt 55 && echo true || echo false

    [tuna@localhost ~]$ test 6 -gt 55 && echo true || echo false

2. Scripting loops (cont)

- test [ ] (cont)
  - ✓ The test command can also be written as square brackets.

    [tuna@localhost ~]$ [56 -gt 55] && echo true || echo false

    [tuna@localhost ~]$ [6 -gt 55] && echo true || echo false

  - ✓ Tests can be combined with logical AND and OR.

    [tuna@localhost ~]$ [66 -gt 55 -a 66 -lt 500] && echo true || echo false

    [tuna@localhost ~]$ [66 -gt 55 -a 660 -lt 500] && echo true || echo false

    [tuna@localhost ~]$ [66 -gt 55 -o 660 -lt 500] && echo true || echo false

2. Scripting loops (cont)

- if then else
  - ✓ The if then else construction is about choice. If a certain condition is met, then execute something, else execute something else.

```
[tuna@localhost ~]$ cat choice
#! /bin/bash
if [ -f tennis.txt ]
then echo tennis.txt exists!
else echo tennis.txt not found!
fi
[tuna@localhost ~]$ ./choice
```

2. Scripting loops (cont)

- if then elif
  - ✓ Can nest a new if inside an else with elif.

```
[tuna@localhost ~]$ cat elif
#! /bin/bash
count=10
if [ $count -eq 20 ]
then
 echo "$count is correct."
elif [ $count -gt 20 ]
then
 echo "Too much."
else
 echo "Not enough."
fi
```

2. Scripting loops (cont)

- for loop
  - ✓ An example using the bash {from..to} shorthand.

```
[tuna@localhost ~]$ cat forloop
#! /bin/bash
for counter in {1..20}
do
 echo counting from 1 to 20, now at $counter
  sleep 1
done
[tuna@localhost ~]$
```

2. Scripting loops (cont)

- while loop
  - ✓ Below is a simple example of a while loop.

```
[tuna@localhost ~]$ cat whileloop
#! /bin/bash
i=10;
while [ $i -ge 0 ] ;
do
  echo Counting down, from 10 to 0, now at $i;
  let i--;
  sleep 1
done
[tuna@localhost ~]$
```

  - ✓ Endless loops can be made with while true or while :

2. Scripting loops (cont)

- until loop
  - ✓ Below is a simple example of an until loop.

```
[tuna@localhost ~]$ cat untilloop
#! /bin/bash
let i=10;
until [ $i -le 0 ] ;
do
  echo Counting down, from 10 to 1, now at $i;
  let i--;
  sleep 1
done
[tuna@localhost ~]$
```

VIII. Local user management

1. Introduction to users
2. User management
3. User passwords
4. Groups

1. Introduction to users

- whoami
  - ✓ The whoami command tells your username.

    [tuna@localhost ~]$ whoami

- who
  - ✓ The who command will give you information about who is logged on the system.

    [tuna@localhost ~]$ who

- who am i
  - ✓ With who am i the who command will display only the line pointing to your current session.

    [tuna@localhost ~]$ who am i

1. Introduction to users (cont)

- w
  - ✓ The w command shows you who is logged on and what they are doing.

    [tuna@localhost ~]$ w

- id
  - ✓ The id command will give you your user id, primary group id, and a list of the groups that you belong to.

    [tuna@localhost ~]$ id

- su to another user
  - ✓ The su command allows a user to run a shell as another user.

    [tuna@localhost ~]$ su tania

1. Introduction to users (cont)

- su to root
  - ✓ Yes you can also su to become root, when you know the root password.

    [tuna@localhost ~]$ su root

- su as root
  - ✓ The root user can become any existing user without knowing that user's password.

    [tuna@localhost ~]$ su tania

- su - $username
  - ✓ To become another user and also get the target user's environment, issue the su - command followed by the target username.

1. Introduction to users (cont)

- su –
  - ✓ When no username is provided to su or su -, the command will assume root is the target.

    [tuna@localhost ~]$ su -

- Run a program as another user
  - ✓ The sudo program allows a user to start a program with the credentials of another user. This can be useful to delegate administrative tasks to another user (without giving the root password).
    - First the command fails for tuna
      [tuna@localhost ~]$ useradd -m valentina
    - But with sudo it works.
      [tuna@localhost ~]$ sudo useradd -m valentina

2. User management

- /etc/passwd
  - The local user database on Linux (and on most Unixes) is /etc/passwd.

    [tuna@localhost ~]$ tail /etc/passwd
    tuna:x:1000:1000:tuna:/home/tuna:/bin/bash

  - The columns contain the username, an x, the user id, the primary group id, a description, the name of the home directory, and the login shell.

- root
  - The root user also called the superuser is the most powerful account.
    [tuna@localhost ~]$ head -1 /etc/passwd

2. User management (cont)

- useradd
  - You can add users with the useradd command.

    [root@localhost ~]# useradd -m -d /home/yanina -c "yanina wickmayer" yanina

    [root@localhost ~]# tail -1 /etc/passwd

- userdel
  - You can delete the user yanina with userdel. The -r option of userdel will also remove the home directory.

    [root@localhost ~]# userdel -r yanina

2. User management (cont)

- Usermod
  - ✓ You can modify the properties of a user with the usermod command.

    [root@localhost ~]# usermod -c 'Nguyen Anh Tu' tuna

    [root@localhost ~]# tail /etc/passwd | grep tuna

    tuna:x:1000:1000:Nguyen Anh Tu:/home/tuna:/bin/bash

- */etc/skel/*
  - ✓ The /etc/skel/ directory contains some (usually hidden) files that contain profile settings and default values for applications. In this way /etc/skel/ serves as a default home directory and as a default user profile.
    - ***[root@localhost ~]# ls -la /etc/skel/***

3. User passwords

- passwd
  - ✓ Passwords of users can be set with the passwd command.

    [tuna@localhost ~]$ passwd

  - ✓ The passwd tool will do some basic verification to prevent users from using too simple passwords.
  - ✓ The root user does not have to follow these rules.

- shadow file
  - ✓ User passwords are encrypted and kept in /etc/shadow, the /etc/shadow file is read only and can only be read by root.

    [root@localhost ~]# tail -5 /etc/shadow

3. User passwords (cont)

- chage
  - ✓ The chage command can be used to set an expiration date for a user account (-E), set a minimum (-m) and maximum (-M) password age, a password expiration date, and set the number of warning days before the password expiration date.

    [root@localhost ~]# chage -l tuna

- disabling a password
  - ✓ When the second field in /etc/passwd starts with an exclamation mark, then the password can not be used.

    [root@localhost ~]# grep kelly /etc/shadow | cut -c1-70

    [root@localhost ~]# usermod -L kelly

    [root@localhost ~]# grep kelly /etc/shadow | cut -c1-70

  - ✓ You can unlock the account again with usermod -U.

4. Groups

- groupadd
  - ✓ Groups can be created with the groupadd command.

    [root@localhost ~]# groupadd tennis

    [root@localhost ~]# groupadd football

    [root@localhost ~]# groupadd snooker

    [root@localhost ~]# groupadd formula1

    [root@localhost ~]# groupadd salsa

- group file
  - ✓ Users can be a member of several groups. Group membership is defined by the /etc/group file.

    [root@localhost ~]# tail -5 /etc/group

4. Groups (cont)

- groups
  - ✓ A user can type the groups command to see a list of groups where the user belongs to.

    [tuna@localhost ~]$ groups

- usermod
  - ✓ Group membership can be modified with the useradd or usermod command.

    [root@localhost ~]# usermod -a -G tennis tuna

    [root@localhost ~]# tail /etc/group | grep tennis

- groupmod
  - ✓ Change the group name with the groupmod command.

    [root@localhost ~]# groupmod -n cafe snooker

4. Groups (cont)

- groupdel
  - ✓ You can permanently remove a group with the groupdel command.

    [root@localhost ~]# groupdel salsa

    [root@localhost ~]# tail -5 /etc/group

- gpasswd
  - ✓ You can delegate control of group membership to another user with the gpasswd command.

    [root@localhost ~]# gpasswd -A tuna tennis

    [root@localhost ~]# su - tuna

    [tuna@localhost ~]$ gpasswd -a kelly tennis

IX. File security

1. Standard file permissions
2. File links

1. Standard file permissions

- file ownership
  - ✓ User owner and group owner
    - The users and groups can own files. Actually, every file has a user owner and a group owner

    [tuna@localhost ~]$ ls –lh

  - ✓ Listing user accounts
    - Use the following command to list all local user accounts.

    [tuna@localhost ~]$ cut -d: -f1 /etc/passwd | column

1. Standard file permissions (cont)

- file ownership (cont)
  - ✓ chgrp
    - You can change the group owner of a file using the chgrp command.

    [root@localhost ~]# chgrp cafe /home/tuna/tennis.txt

    [root@localhost ~]# ls -l /home/tuna/ | grep tennis

  - ✓ chown
    - You can also use chown to change both the user owner and the group owner.

    [root@localhost ~]# chown root:tennis /home/tuna/tennis.txt

    [root@localhost ~]# ls -l /home/tuna/ | grep tennis

1. Standard file permissions (cont)

- List of special files
  - ✓ When you use **ls -l**, for each file you can see ten characters before the user and group owner.
  - ✓ The first character tells us the type of file.

First character	File type
-	normal file
d	directory
l	symbolic link
p	named pipe
b	block device
c	character device
s	socket

[tuna@localhost ~]$ ls -ld /dev/console /dev/sda

1. Standard file permissions (cont)

- Permissions
  - ✓ rwx
    - The nine characters following the file type denote the permissions in three triplets.

Permission	On a file	On a directory
r (read)	read file contents (cat)	read directory contents (ls)
w (write)	change file contents (vi)	create files in (touch)
x (execute)	execute the file	enter the directory (cd)

1. Standard file permissions (cont)

- Permissions (cont)
  - ✓ Three sets of rwx

    [tuna@localhost ~]$ ls -l temp.txt

    -rw-rw-r--. 1 tuna tuna 0 Jul 31 10:21 temp.txt

Position	Characters	Function
1	-	This is a regular file
2-4	rw-	permissions for the user owner
5-7	rw-	permissions for the group owner
8-10	r--	permissions for others

1. Standard file permissions (cont)

- Permissions (cont)
  - ✓ Setting permissions (chmod)
    - Permissions can be changed with chmod.
      [tuna@localhost ~]$ touch permissions.txt
      [tuna@localhost ~]$ ls -l permissions.txt
    - Give the user owner execute permission
      [tuna@localhost ~]$ chmod u+x permissions.txt
    - Remove the group owners read permission
      [tuna@localhost ~]$ chmod g-r permissions.txt
    - Removes the others read permission.
      [tuna@localhost ~]$ chmod o-r permissions.txt

1. Standard file permissions (cont)

- Permissions (cont)
  - ✓ Setting permissions (chmod)
    - Give all of them the write permission.

      [tuna@localhost ~]$ chmod a+w permissions.txt

      [tuna@localhost ~]$ chmod +w permissions.txt

    - Set explicit permission.

      [tuna@localhost ~]$ chmod u=rw permissions.txt

    - Make any kind of combination.

[tuna@localhost ~]$ chmod u=rw,g=rw,o=r permissions.txt

[tuna@localhost ~]$ chmod u=rwx,ug+rw,o=r permissions.txt

1. Standard file permissions (cont)

- Setting octal permissions

Binary	Octal	Permission
000	0	-
001	1	--x
010	2	-w-
011	3	-wx
100	4	r--
101	5	r-x
110	6	rw-
111	7	rwx

1. Standard file permissions (cont)

- umask
  - ✓ When creating a file or directory, a set of default permissions are applied.
  - ✓ These default permissions are determined by the umask.

    [tuna@localhost ~]$ umask

    [tuna@localhost ~]$ touch test.txt

    [tuna@localhost ~]$ ls -l test.txt

- mkdir –m
  - ✓ When creating directories with mkdir you can use the -m option to set the mode

    [tuna@localhost ~]$ mkdir -m 700 MyDir

    [tuna@localhost ~]$ mkdir -m 777 Public

    [tuna@localhost ~]$ ls -dl MyDir/ Public/

2. File links

- Hard links
  - ✓ Creating hard links
    - When creating a hard link to a file with ln, a new file name is mapped to an existing inode.

      [tuna@localhost ~]$ ls -li bin/choise
      [tuna@localhost ~]$ ln bin/choise hardln_choise
      [tuna@localhost ~]$ ls -li hardln_choise

  - ✓ Finding hard links
    - Can use the find command to look for files with a certain inode.

      [tuna@localhost ~]$ find / -inum 401951 2> /dev/null

2. File links (cont)

- Symbolic links
  - ✓ Symbolic links (soft links) do not link to inodes, but create a name to name mapping

    [tuna@localhost ~]$ ln -s bin/choise symln_choice

    [tuna@localhost ~]$ ls -li symln_choice

- Removing links
  - ✓ Links can be removed with rm.

    [tuna@localhost ~]$ rm hardln_choise

    [tuna@localhost ~]$ rm symln_choice

X. System management

1. Package management

1. Package management

- Package terminology
  - ✓ Repository
    - A lot of software and documentation for the Linux distribution is available as packages in one or more centrally distributed repositories.
    - These packages in such a repository are tested and very easy to install (or remove) with a graphical or command line installer.
  - ✓ .deb packages
    - Debian, Ubuntu, Mint and all derivatives from Debian and Ubuntu use .deb packages.
    - To manage software on these systems, you can use aptitude or apt-get, both these tools are a front end for dpkg.

1. Package management (cont)

- Package terminology (cont)
  - ✓ .rpm packages
    - Red Hat, Fedora, CentOS, OpenSUSE, Mandriva, Red Flag and others use .rpm packages.
    - The tools to manage software packages on these systems are yum and rpm.
  - ✓ Dependency
    - Some packages need other packages to function.
    - Tools like apt-get, aptitude and yum will install all dependencies you need.
    - When using dpkg or rpm, or when building from source, you will need to install dependencies yourself.

1. Package management (cont)

- rpm
  - ✓ About rpm
    - The Yellowdog Updater, Modified (yum) is an easier command to work with rpm packages.
    - It is installed by default on Fedora and Red Hat Enterprise Linux since version 5.2.
  - ✓ yum list
    - Issue yum list available to see a list of available packages. The available parameter is optional.

      [root@localhost ~]# yum list | wc –l
    - Issue yum list $package to get all versions (in different repositories) of one package.

      [root@localhost ~]# yum list firefox

1. Package management (cont)

- rpm (cont)
  - ✓ yum search
    - To search for a package containing a certain string in the description or name use yum search $string.

      [root@localhost ~]# yum search ifconfig

  - ✓ yum install
    - To install an application, use yum install $package. Naturally yum will install all the necessary dependencies.

      [root@localhost ~]# yum install firefox

    - *You can add more than one parameter here.*

      [root@localhost ~]# yum install mlocate net-tools

1. Package management (cont)

- rpm (cont)
  - ✓ yum update
    - To bring all applications up to date, by downloading and installing them, issue yum update.

      [root@localhost ~]# yum update

    - If you only want to update one package, use yum update $package.

      [root@localhost ~]# yum update firefox

  - ✓ yum remove/erase
    - You can remove one or more applications by appending their name. behind yum remove/erase

      [root@localhost ~]# yum remove firefox

XI. Network management

1. Interface configuration
2. ssh client and server

1. Interface configuration

- RHEL nic configuration
  - ✓ /etc/sysconfig/network
    - The /etc/sysconfig/network file is a global configuration file. It allows us to define whether we want networking (NETWORKING=yes|no), what the hostname should be (HOSTNAME=) and which gateway to use (GATEWAY=).

      [tuna@els-e82050 ~]$ cat /etc/sysconfig/network

    - There are a dozen more options settable in this file, details can be found in /usr/share/doc/initscripts-*/sysconfig.txt.

    - Note that this file contains no settings at all in a default RHEL7 install (with networking enabled).

      [root@localhost ~]# cat /etc/sysconfig/network
      # Created by anaconda

1. Interface configuration (cont)

- RHEL nic configuration (cont)
  - ✓ /etc/sysconfig/network-scripts/ifcfg-
    - Each network card can be configured individually using the /etc/sysconfig/network-scripts/ifcfg-* files.
    - dhcp client
      - Below is an example of ifcfg-eth0 configured for dhcp

[tuna@els-e82050 ~]$ cat /etc/sysconfig/network-scripts/ifcfg-eth0

DEVICE=eth0

HWADDR=b8:ac:6f:7d:13:88

NM_CONTROLLED=no

BOOTPROTO="dhcp"

ONBOOT=yes

1. Interface configuration (cont)

- RHEL nic configuration (cont)
  - ✓ /etc/sysconfig/network-scripts/ifcfg-
    - dhcp client
      - RHEL7 adds ipv6 variables to this file.

root@localhost ~]# cat /etc/sysconfig/network-scripts/ifcfg-enp0s3

IPV6INIT="yes"
IPV6_AUTOCONF="yes"
IPV6_DEFROUTE="yes"
IPV6_PEERDNS="yes"
IPV6_PEERROUTES="yes"
IPV6_FAILURE_FATAL="no"
NAME="enp0s3"
UUID="ec9f675f-ad1d-4754-8983-9080f3880f76"
DEVICE="enp0s3"

1. Interface configuration (cont)

- RHEL nic configuration (cont)
  - ✓ /etc/sysconfig/network-scripts/ifcfg-
    - fixed ip
      - Below an example of a fixed ip configuration in /etc/sysconfig/network-scripts/ifcfg-eth0.

```
tuna@els-e82050 ~]$ cat /etc/sysconfig/network-scripts/ifcfg-eth0
HWADDR=00:1b:21:58:c5:17
BOOTPROTO=none
IPADDR=192.168.21.2
NETMASK=255.255.255.0
GATEWAY=192.168.21.254
DNS1=172.19.7.122
DNS2=172.17.1.20
```

1. Interface configuration (cont)

- RHEL nic configuration (cont)
  - ✓ /etc/sysconfig/network-scripts/ifcfg-
    - nmcli
      - On RHEL7 you should run nmcli connection reload if you changed configuration files in /etc/sysconfig/ to enable your changes.
      - The nmcli tool has many options to configure networking on the command line in RHEL7/CentOS7
        [root@localhost ~]# man nmcli
    - nmtui
      - Another recommendation for RHEL7/CentOS7 is to use nmtui. This tool will use a 'windowed' interface in command line to manage network interfaces.
        [root@localhost ~]# man nmtui

1. Interface configuration (cont)

- RHEL nic configuration (cont)
  - ✓ /etc/sysconfig/network-scripts/ifcfg-
    - /sbin/ifup and /sbin/ifdown
      - The ifup and ifdown commands will set an interface up or down, using the configuration discussed above.

[root@localhost ~]# ifdown enp0s3 && ifup enp0s3

Device 'enp0s3' successfully disconnected.

Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/2)

[root@localhost ~]# ifconfig

1. Interface configuration (cont)

- ifconfig
  - ✓ The use of /sbin/ifconfig without any arguments will present you with a list of all active network interface cards, including wireless and the loopback interface.

    [root@localhost ~]# ifconfig

  - ✓ You can also use ifconfig to obtain information about just one network card.

    [root@localhost ~]# ifconfig lo

    [root@localhost ~]# ifconfig enp0s3

1. Interface configuration (cont)

- Ifconfig (cont)
  - ✓ up and down
  - – You can also use ifconfig to bring an interface up or down.

[root@localhost ~]# ifconfig enp0s3 down
[root@localhost ~]# ifconfig enp0s3 up
[root@localhost ~]# ifconfig enp0s3

  - ✓ Setting ip address
    - You can temporary set an ip address with ifconfig.
    - This ip address is only valid until the next ifup/ifdown cycle or until the next reboot.

[root@localhost ~]# ifconfig enp0s3 192.168.21.15 netmask 255.255.255.0 broadcast 192.168.21.255
[root@localhost ~]# ifdown enp0s3 && ifup enp0s3

1. Interface configuration (cont)

- ip
  - ✓ The ifconfig tool is deprecated on some systems. Use the ip tool instead.
  - ✓ To see ip addresses on RHEL7 for example, use this command:

    [root@localhost ~]# ip a

- dhclient
  - ✓ Home and client Linux desktops often have /sbin/dhclient running.
  - ✓ This is a daemon that enables a network interface to lease an ip configuration from a dhcp server.
  - ✓ When your adapter is configured for dhcp or bootp, then /sbin/ifup will start the dhclient daemon.

1. Interface configuration (cont)

- route
  - ✓ You can see the computer's local routing table with the /sbin/route command (and also with netstat -r).

    [root@#localhost ~]# netstat –r

  - ✓ If the computer does not have a gateway configured, so we use route add default gw to add a default gateway

    [root@#localhost ~]# route add default gw 192.168.21.254

- ping
  - ✓ If you can ping to another host, then tcp/ip is configured.

    [root@#localhost ~]# ping 42.117.10.84

1. Interface configuration (cont)

- ethtool
  - ✓ To display or change network card settings, use ethtool.
  - ✓ The results depend on the capabilities of your network card.
  - ✓ The example shows a network that auto-negotiates it's bandwidth.

    [root@#localhost ~]# ethtool enp0s3

2. ssh client and server

- About ssh
  - ✓ secure shell
    - The secure shell or ssh is a collection of tools using a secure protocol for communications with remote Linux computers.
  - ✓ Log on to a remote server
    - The following example shows how to use ssh to log on to a remote computer running Linux.

      [root@#localhost ~]# ssh *tuna@192.168.21.2*

    - The user can log out of the remote server by typing exit or by using Ctrl-d.

2. ssh client and server (cont)

- scp
  - ✓ The scp command works just like cp, but allows the source and destination of the copy to be behind ssh.
  - ✓ Here is an example where we copy the CAPTURE.sh file from the remote server to the /home/tuna/Downloads/ directory of user tuna.

    [root@#localhost ~]# scp tuna@192.168.21.2:CAPTURE.sh /home/tuna/Downloads/

  - ✓ Here is an example of the reverse, copying a local file to a remote server.

    [root@#localhost ~]# scp firefox-38.1.0-1.el7.centos.x86_64.rpm tuna@192.168.21.2:Downloads

2. ssh client and server (cont)

- Setting up passwordless ssh
  - ✓ ssh authentication through public/private keys, use **ssh-keygen** to generate a key pair without a passphrase, and then copy the public key to the destination server.
  - ✓ Let's do this step by step.
    - We will set up ssh without password between tuna and tania.

    1. ssh-keygen

       [tuna@localhost ~]$ ssh-keygen -t rsa
       Enter file in which to save the key (/home/tuna/.ssh/id_rsa):
       Enter passphrase (empty for no passphrase):
       Enter same passphrase again:

2. ssh client and server (cont)

- Setting up passwordless ssh (cont)
    1. ~/.ssh
        - While ssh-keygen generates a public and a private key, it will also create a hidden .ssh directory with proper permissions.
        - If you create the .ssh directory manually, then you need to chmod 700 it! Otherwise ssh will refuse to use the keys.
        - As you can see, the **.ssh** directory is secure in tuna's home directory.

[tuna@localhost ~]$ ls -ld .ssh/

drwx------. 2 tuna tuna 66 Aug 9 11:34 .ssh/

2. ssh client and server (cont)

- Setting up passwordless ssh (cont)
  1. ~/.ssh (cont)
     - Manually create the **.ssh** directory for tania, so it needs to be manually secured.

```
[tania@localhost ~]$ mkdir .ssh
[tania@localhost ~]$ ls -ld .ssh
drwxrwxr-x. 2 tania tania 6 Aug  9 11:57 .ssh
[tania@localhost ~]$ chmod 700 .ssh/
[tania@localhost ~]$ ls -ld .ssh
drwx------. 2 tania tania 6 Aug  9 11:57 .ssh
```

2. ssh client and server (cont)

- Setting up passwordless ssh (cont)
  1. Copy the public key to the other computer
     - ***[tuna@localhost ~]$ scp .ssh/id_rsa.pub tania@192.168.0.111:~/.ssh/authorized_keys***
     - Tania could also have used **ssh-copy-id** like in this example.
     - ***[tania@localhost ~]$ ssh-copy-id .ssh/id_rsa.pub tuna@192.168.0.109***

✓ Now try logging into the machine, with:
"ssh 'tuna@192.168.0.109'"

Q&A

Thank you !