جامعة القاهرة الجديدة
التكنولوجية

Fall 2023

(3)

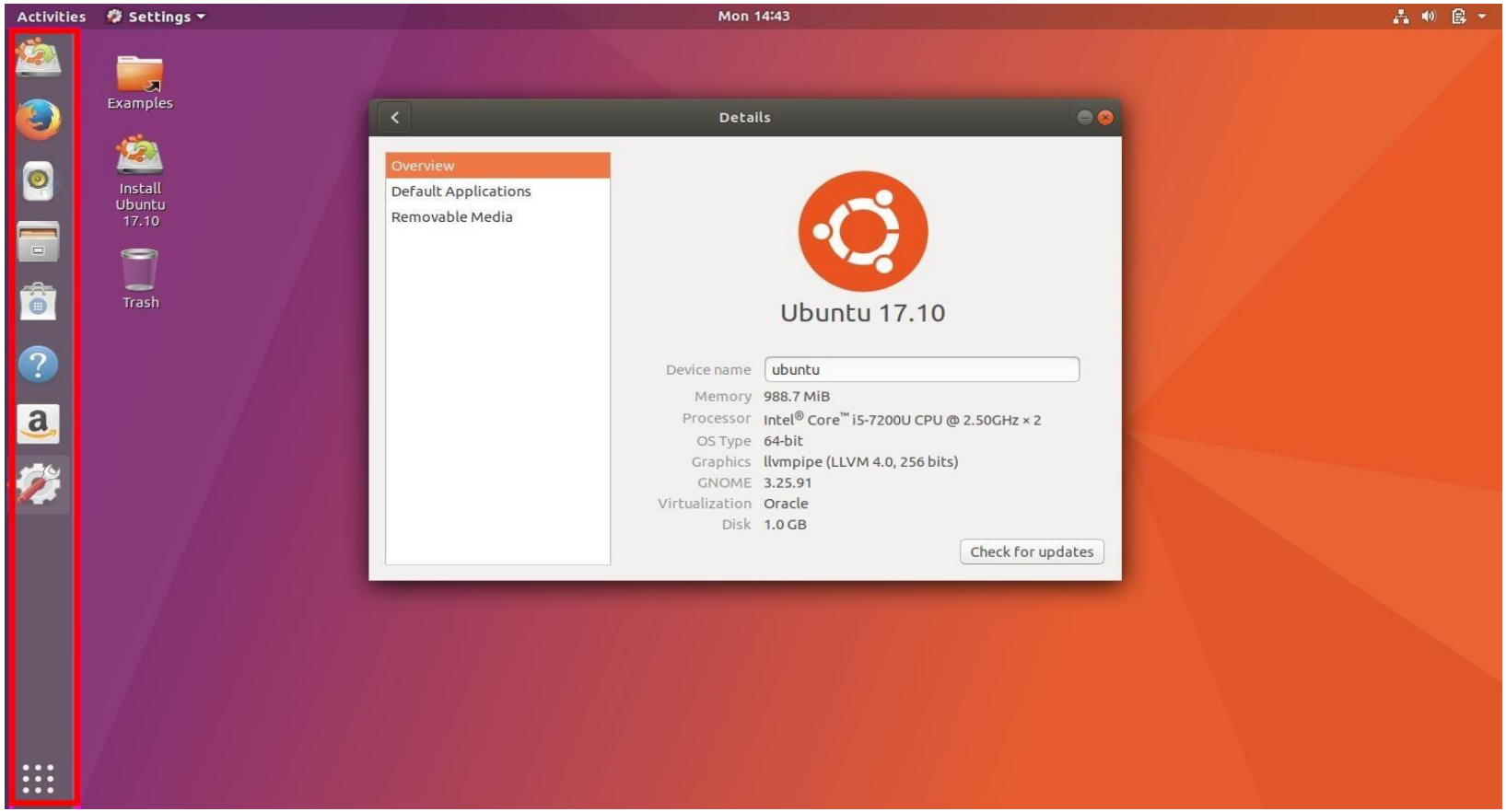
Linux Essentials

Dr. Hatem Yousry

Agenda

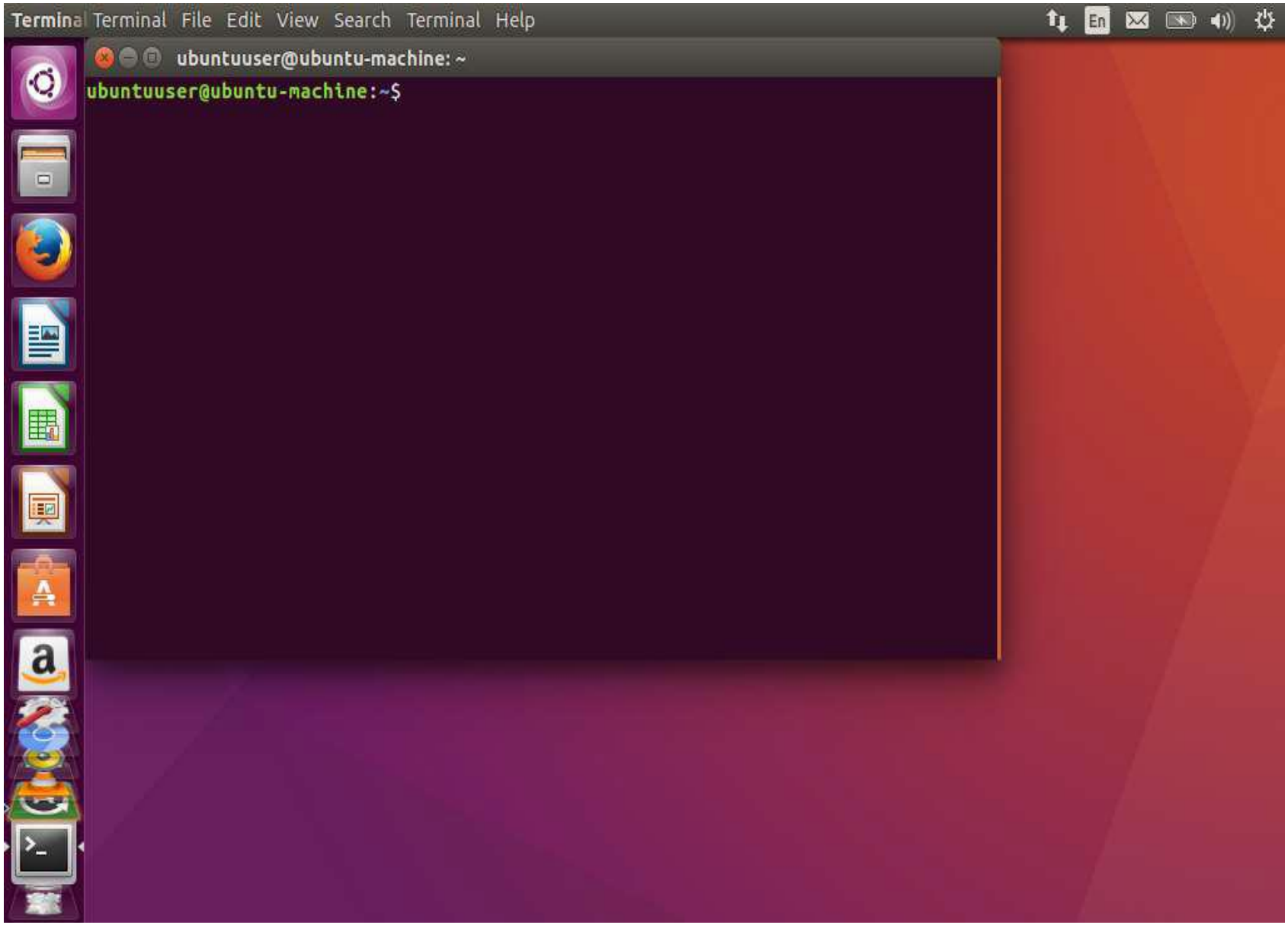
- **Ubuntu.**
- **Terminals and Shells.**
- **Commands.**

Ubuntu


Activities
Settings
Mon 14:43
Examples
Install Ubuntu 17.10
Trash
Details
Overview
Default Applications
Removable Media
Ubuntu 17.10
Device name ubuntu
Memory 988.7 MiB
Processor Intel® Core™ i5-7200U CPU @ 2.50GHz × 2
OS Type 64-bit
Graphics llvmpipe (LLVM 4.0, 256 bits)
GNOME 3.25.91
Virtualization Oracle
Disk 1.0 GB
Check for updates

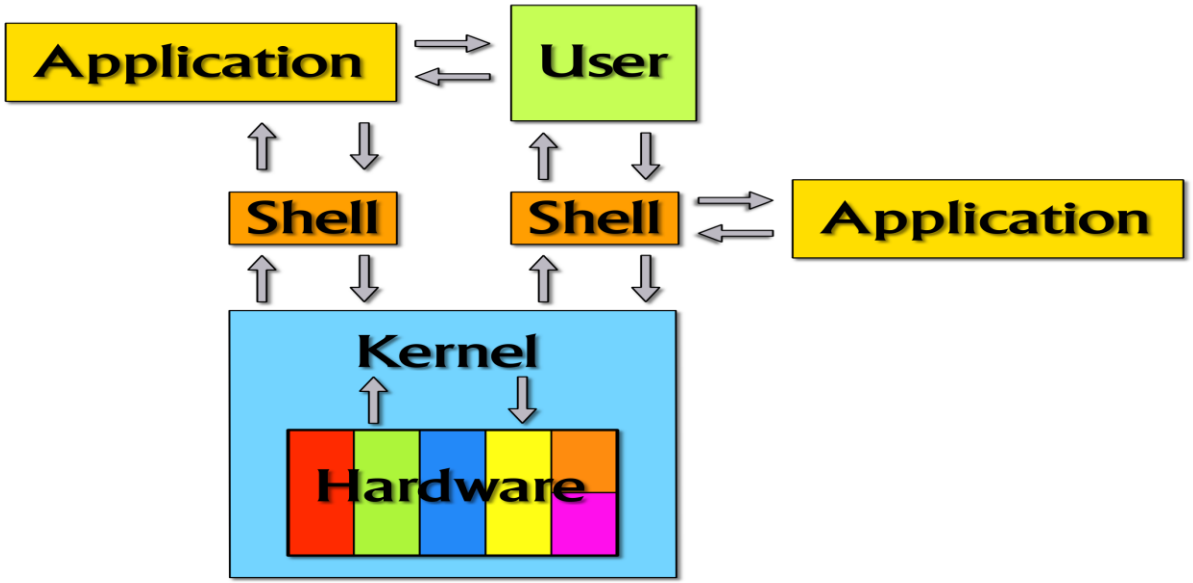
Terminals and Shells

- Even within a graphical Linux environment it is often convenient to access a **“terminal window”** where you can enter textual commands in a **“shell”** (the remainder of this manual mostly talks about shell commands, so you are likely to need this).
- Fortunately, on most Linux desktop environments a terminal window is only a few mouse clicks away. **In KDE on Debian GNU/Linux, for example, there is an entry called “Konsole (Terminal)” within the start menu under “System”,** which will open a convenient program running a shell that will accept and execute textual commands. Similar methods are available on other desktop environments and distributions.


Terminal Terminal File Edit View Search Terminal Help
En
ubuntuuser@ubuntu-machine: ~
ubuntuuser@ubuntu-machine:~$

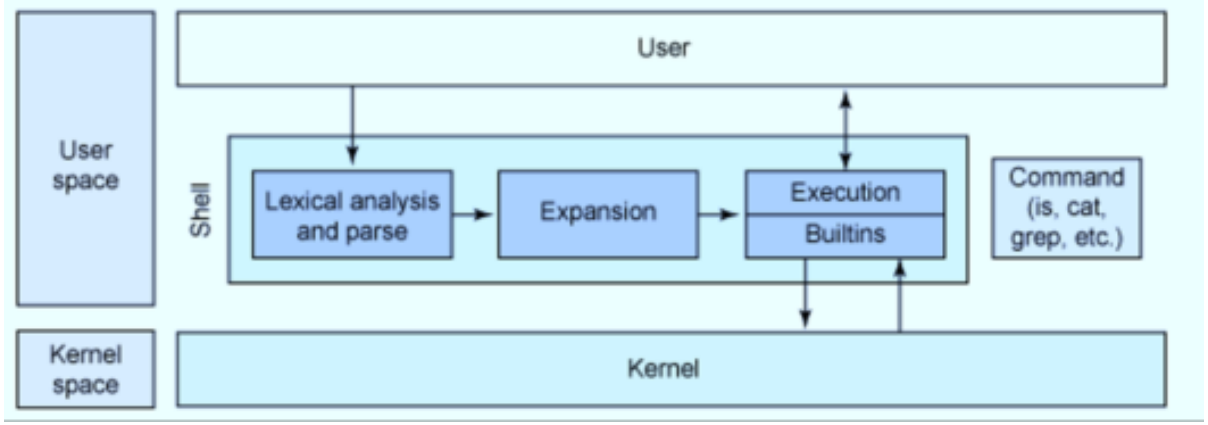
What Is The Shell?

- **Shell** is an interface between a user and OS to access to an operating system's services. It can be either GUI or CLI (Command Line interface).



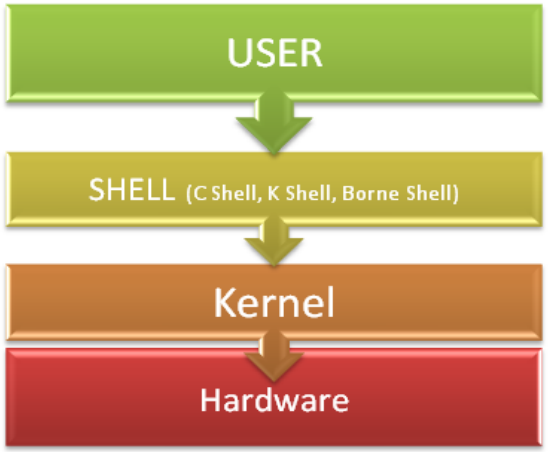
What Is The Shell?

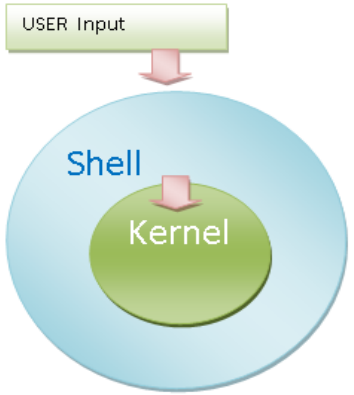
- Users cannot communicate directly with the operating system kernel. This is only possible through programs accessing it via **“system calls”.**
- However, you must be able to start such programs in some way. This is the task of the shell, a special user program that (usually) reads commands from the keyboard and interprets them (for example) as commands to be executed.
- Accordingly, the shell serves as an “interface” to the computer that encloses the actual operating system like a shell (as in “shellfish”—hence the name) and hides it from view.
- Of course the shell is only one program among many that access the operating system.



Shell Types

- The Bourne Shell (sh) Developed at AT&T Bell Labs by Steve Bourne, the Bourne shell is regarded as the first UNIX shell ever. ...
- **The GNU Bourne-Again Shell (bash).**
- **The C Shell (csh).**
- **The Korn Shell (ksh).**
- **The Z Shell (zsh).**





Shell Types

- **sh (Bourne shell)** is a shell command-line interpreter, for Unix/Unix-like operating systems. It provides some built-in commands. In scripting language we denote interpreter as **#!/bin/sh.** It was one most widely supported by other shells like bash (free/open), kash (not free).
- **Bash (Bourne again shell)** is a shell replacement for the Bourne shell. **Bash is superset of sh.**
- Bash supports sh. POSIX (Portable Operating System Interface) is a set of standards defining how POSIX-compliant systems should work. Bash is not actually a POSIX compliant shell.
- In a scripting language we denote the interpreter as #!/bin/bash.

Shell Types

- **Shell is like an interface or specifications or API.**
- **sh is a class which implements the Shell interface.**
- **Bash is a subclass of the sh.**

Unix Shell application comparison table

Application	sh	csh	ksh	bash	tcsh
Job control	N	Y	Y	Y	Y
Aliases	N	Y	Y	Y	Y
Input/Output redirection	Y	N	Y	Y	N
Command history	N	Y	Y	Y	Y
Command line editing	N	N	Y	Y	Y
Vi Command line editing	N	N	Y	Y	Y
Underlying Syntax	sh	csh	ksh	sh	csh

Shell Prompt

- **The prompt, $,** which is called the command prompt, is issued by the shell. While the prompt is displayed, you can type a command.
- Shell reads your input after you press Enter. It determines the command you want executed by looking at the first word of your input. A word is an unbroken set of characters. Spaces and tabs separate words.
- Following is a simple example of the date command, which displays the current date and time −
- **$date**
- **Thu Jun 25 08:30:19 MST 2009**

Shell Prompt Example

- Code segments and script output will be displayed as monospaced text. Command-line entries will be preceded by the **Dollar sign ($).** If your prompt is different, enter the command:
- **PS1="$ " ; export PS1**
- Then your interactions should match the examples given (such as ./my-script.sh below).
- Script output (such as **"Hello World"** below) is displayed at the start of the line.
- **$ echo '#!/bin/sh' > my-script.sh**
- **$ echo 'echo Hello World' >> my-script.sh**
- **$ chmod 755 my-script.sh**
- **$ ./my-script.sh**
- **Hello World**
- **$**

Shell Prompt Example

- **my-script.sh**
- **#!/bin/sh**
- **# This is a comment!**
- **echo Hello World # This is a comment, too!**

- Note that to make a file executable, you must set the **eXecutable bit**, and for a shell script, the Readable bit must also be set:
- **$ chmod a+rx my-script.sh**
- **$ ./my-script.sh**

Shell Scripts

- Shells may be invoked interactively to read user commands (normally on a “terminal” of some sort).
- Most shells can also read commands from files containing pre-cooked command sequences. Such files are called **“shell scripts”.**
- A shell performs the following steps:

1. **Read a command from the terminal (or the file)**
2. **Validate the command**
3. **Run the command directly or start the corresponding program**
4. **Output the result to the screen (or elsewhere)**
5. **Continue at step 1.**

Commands

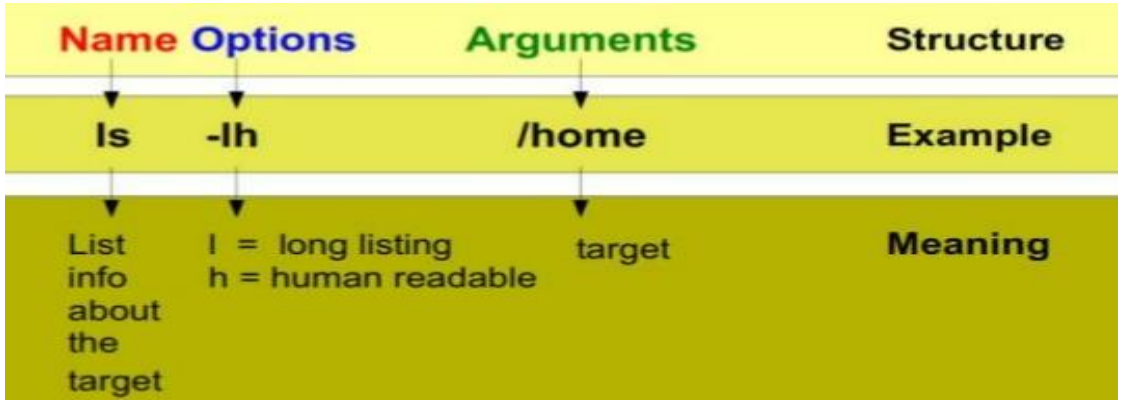
- A computer's operation, no matter which operating system it is running, can be loosely described in three steps:

1. The computer waits for user input
2. The user selects a command and enters it via the keyboard or mouse
3. The computer executes the command

- In a Linux system, the shell displays a **"prompt",** meaning that commands can be entered. **This prompt usually consists of a user and host (computer) name, the current directory, and a final character:**
- User:/home > _

Command Structure

- A command is essentially a sequence of characters which is ends with a press of the ↩ key and is subsequently evaluated by the shell. A command's parameters can be roughly divided into two types:
  - Options
  - Arguments
- The general command structure can be displayed as follows:
  - **Command—"What to do?"**
  - **Options—"How to do it?"**
  - **Arguments—"What to do it with?"**



Options and Arguments

- Parameters starting with a **dash (“-”) are called options**.
- These are usually, er, optional—the details depend on the command in question. Figuratively spoken they are **“switches”** that allow certain aspects of the command to be switched on or off. If you want to pass several options to a command, they can (often) be accumulated behind a **single dash**, i. e., the options sequence **“-a -l -F” corresponds to “-alF”.**
- Many programs have more options than can be conveniently mapped to single characters, or support **“long options”** for readability (frequently in addition to equivalent single-character options). Long options most often start with two dashes and cannot be accumulated: **“foo --bar --baz”.**
- Parameters with no leading dash are called arguments. These are often the names of files that the command should process.

Command Types

- In shells, there are essentially two kinds of commands:
- **Internal commands :**These commands are made available by the shell itself.
- The Bourne-again shell contains approximately **30 such commands**, which can be executed very quickly. Some commands **(such as exit or cd)** alter the state of the shell itself and thus cannot be provided externally.
- **External commands:** The shell does not execute these commands by itself but launches **executable files**, which within the file system are usually found in directories **like /bin or /usr/bin**. As **a user, you can provide your own programs**, which the shell will execute like all other external commands.
- $ **type echo**
- echo is a shell builtin
- $ **type date**
- date is /bin/date

Command Types

- **echo** is an interesting command which simply outputs its parameters:
- $ **echo Thou has it now, king, Cawdor, Glamis, all**
- Thou has it now, king, Cawdor, Glamis, all
- date displays the current date and time, possibly adjusted to the current time zone and language setup:
- $ **date**
- Mon May 7 15:32:03 CEST 2020

Linux Command

For this Purpose	Use this Command Syntax	Example (In front of $ Prompt)
To see date	date	**$ date**
To see who's using system.	who	**$ who**
Print working directory	pwd	**$ pwd**
List name of files in current directory	ls or dirs	**$ ls**
To create text file NOTE: Press and hold **CTRL** key and press **D** to stop or to end file (**CTRL+D**)	cat > { file name }	**$ cat > myfile** *type your text when done press ^D*
To text see files	cat {file name }	**$ cat myfile**
To display file one full screen at a time	more {file name }	**$ more myfile**
To move or rename file/directory	mv {file1} {file2}	**$ mv sales sales.99**
To create multiple file copies with various link. After this both oldfile newfile refers to same name	ln {oldfile} {newfile}	**$ ln Page1 Book1**
To remove file	rm file1	**$ rm myfile**

Background Process & Some Operators

Redirect : Redirection is a feature in Linux such that when executing a command, you can change the standard input/output devices.

> **Input**

< **Output**

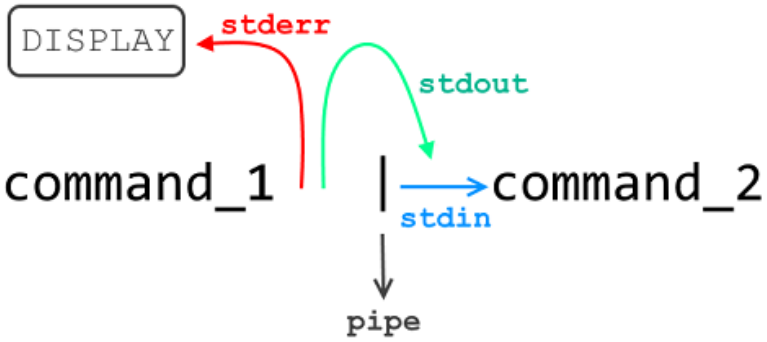
| **Pipe**

Symbol	Description
>	Directs the standard output of a command to a file. If the file exists, it is overwritten.
>>	Directs the output to a file, adding the output to the end of the existing file.
2>	Directs standard error to the file.
2>>	Directs the standard error to a file, adding the output to the end of the existing file.
&>	Directs standard output and standard error to the file.
<	Directs the contents of a file to the command.
<<	Accepts text on the following lines as standard input.
<>	The specified file is used for both standard input and standard output.

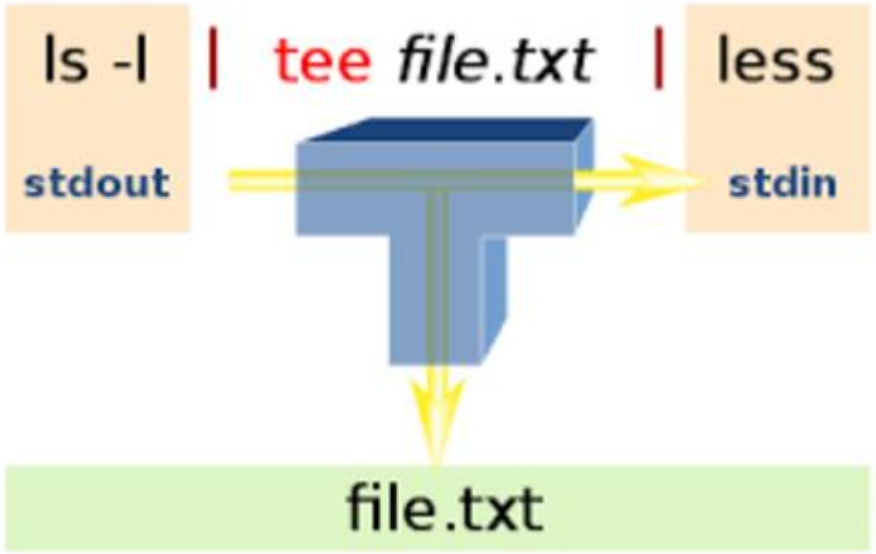
- The pipe command lets you sends the output of one command to another.

ex:

ls -l 1>allfile &



Example



$ls | more

$ls > dir_listing.txt

$cat < file.sh

append:

$ ls >> dir_listing.txt

The following adds the contents of File1 at the end of File2:

$ cat File1 >> File2

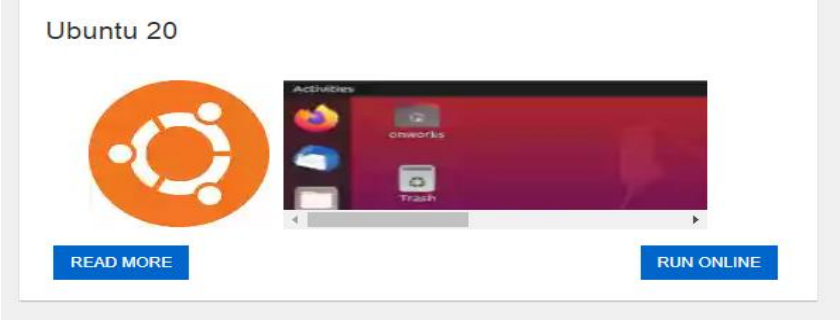
Disk related commands

- **du -** Summarize disk usage of each FILE, recursively for directories.
- **df -** report filesystem disk space usage
- Directory Related Commands:
- **cd**
- **mkdir**
- **Pwd:** stands for Print Working Directory. It prints the path of the working directory, starting from the root.

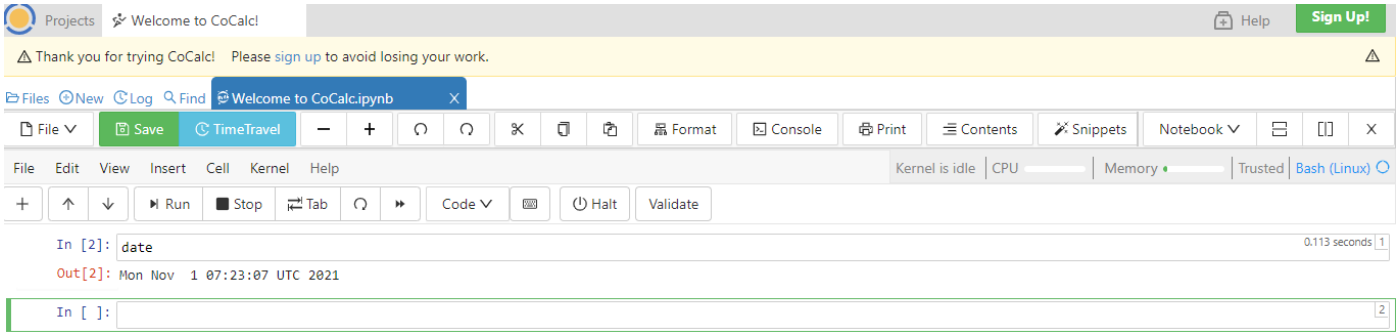
Process Related Commands

- **Ps:** is used to list the currently running processes and their PIDs along with some other information
- **Top:** shows the summary information of the system and the list of **processes** or threads which are currently managed by the **Linux Kernel**.
- **Netstat:** The **netstat command** generates displays that show network status and protocol statistics.
- **Pstree:** pstree is a **Linux** command that shows the running processes as a tree. It is used as a more visual alternative to the ps command. The root of the tree is either init or the process with the given pid.
- **Kill:** which is used to terminate processes manually. **kill command** sends a signal to a process which terminates the process.

Linux Ubuntu Online Terminal



- https://cocalc.com/projects/3bbb1fa0-33ed-45b1-8f56-5c41b8ac7da0/files/Welcome%20to%20CoCalc.ipynb?session=default
- https://chrome.google.com/webstore/detail/ubuntu-free-online-linux/pmaonbjcobmgkemldgcedmpbmmncpbgi?hl=en
- https://www.onworks.net/os-distributions/ubuntu-based



Thank You

Linux

Dr. Hatem Yousry
E-mail: Hyousry@nctu.edu.eg