

# 3. SOFTWARE & OPERATING SYSTEMS



# Copyright Notice

2

- This presentation is presented as is. This presentation was assembled using information from various websites or sources across the web.
- This presentation uses Creative Commons Attribution 4.0 International (CC BY 4.0). © 2021 BilimEdtech



# 3.1: System Software (Part 1)

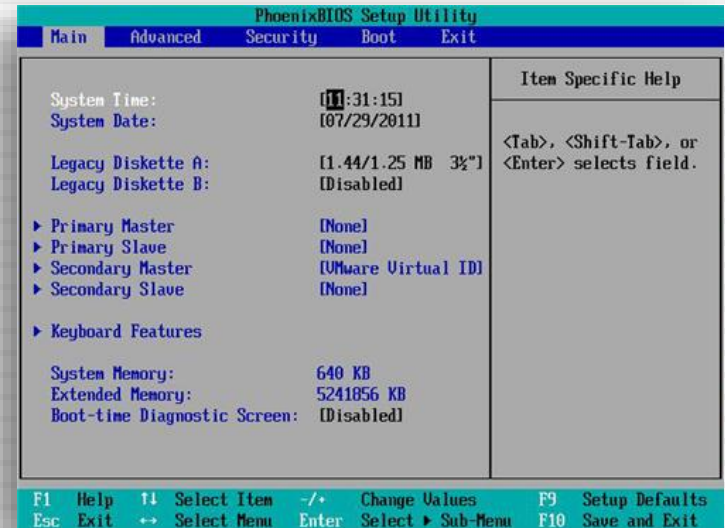
## 3.1: System Software (Part 1)

- ▶ Programming Languages
- ▶ Compilers and Interpreters

## 3.2: System Software (Part 2)

- ▶ Operating Systems

## 3.3: Application Software



# Learning Objectives

- ❑ List the different categories of computer software
- ❑ Describe the purpose of system software
- ❑ Explain the difference between a high-level and low-level programming language
- ❑ Describe the purpose of compilers

# Introduction

5

- **Program**

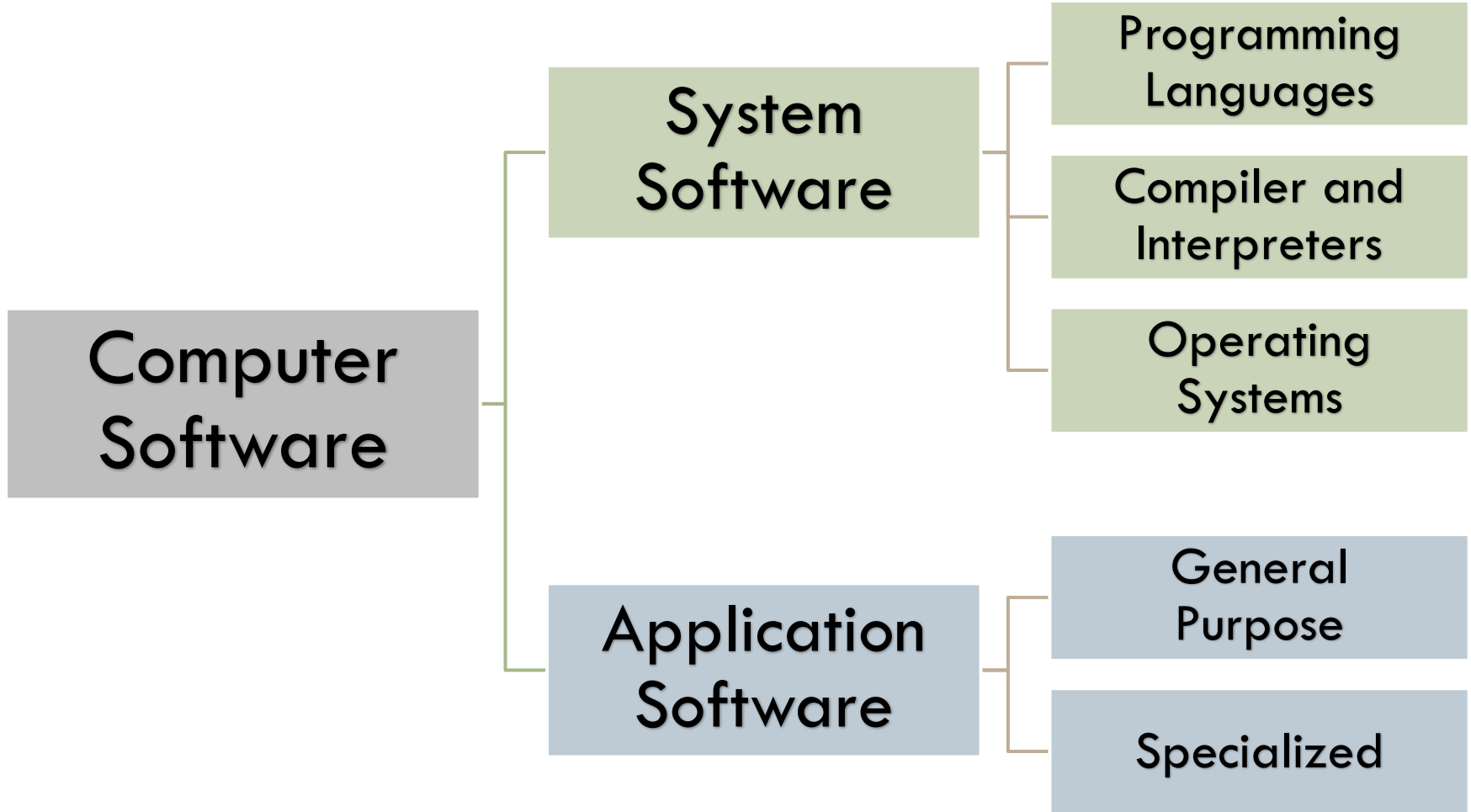
- A set of sequential instructions that tells the computer what to do

- **Software**

- A collection of programs, data, and information

- **Programmer**

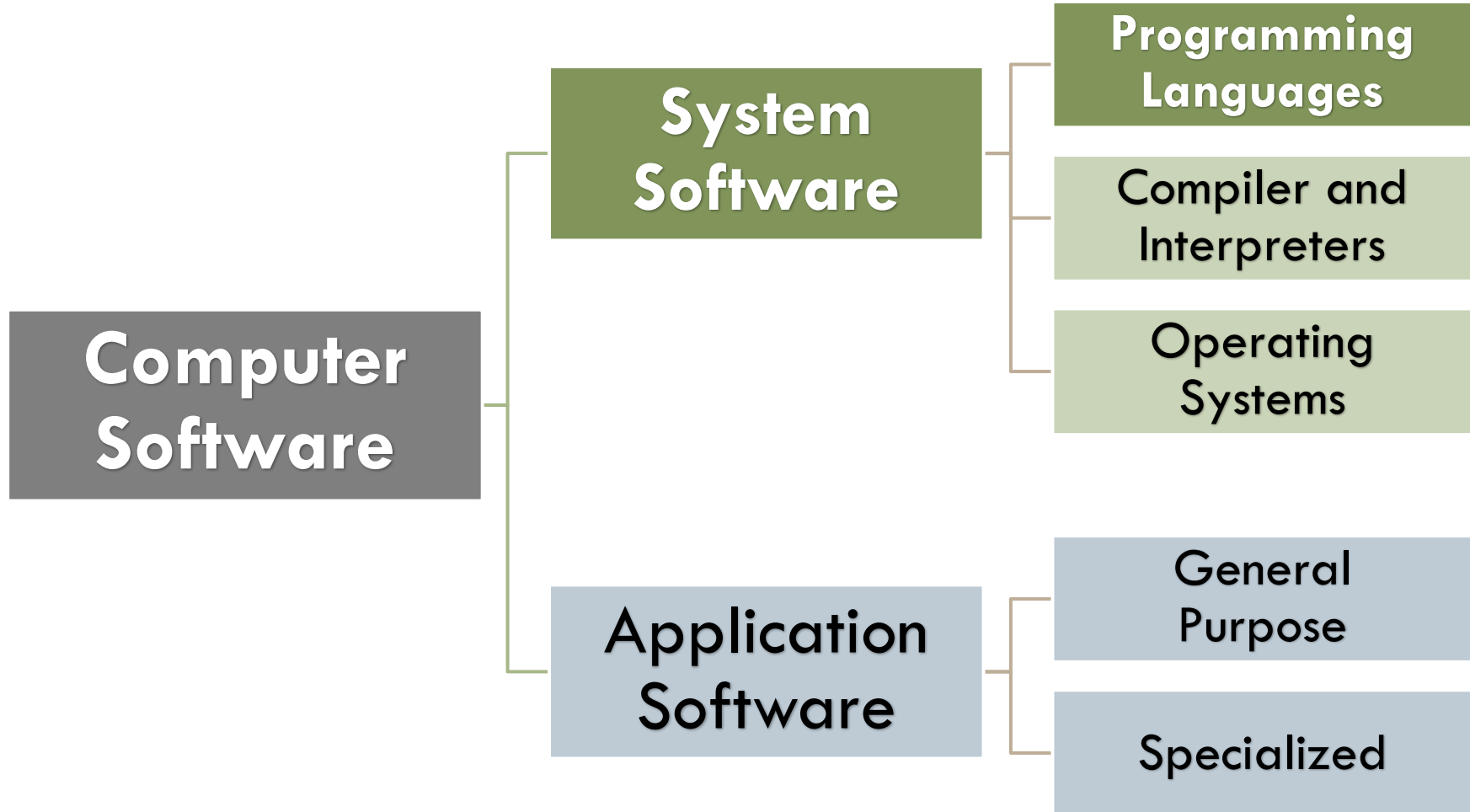
- The person who makes a program



# System Software

7

- Software that is
  - ▣ The computer uses
  - ▣ That operates a computer system
- Included with the computer
  - ▣ Embedded software, such as firmware or the BIOS
- Installed or loaded later
  - ▣ Operating systems and drivers





# Why Programming Languages?

9

- To understand where software comes from, we need to describe how a programmer writes programs
  - ▣ Software controls hardware
  - ▣ Programmers need a way to create the software
- Programming languages allow the programmer to create programs to control hardware

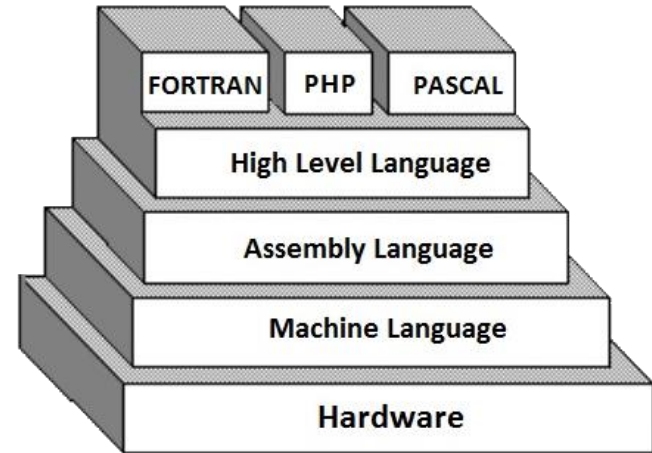
# Programming language

- Programming languages allow programmers to create programs
- It is a set of words, rules, syntax
- There are many programming languages, each designed to solve specific kinds of problems
  - ▣ Ex. C, C#, Java, PHP, ASP, Python, etc.

# Programming language (2)

11

- Levels of Programming Languages
  - ▣ Machine Language
  - ▣ Assembly Language
  - ▣ High-Level Language
  - ▣ Application Languages



# Machine Language

12

- They are written in computer languages
  - ▣ binary (0s and 1s), Octal (base 8), Hex (base 16)
- Changed on the computer architecture
- Programmers rarely write machine language
  - ▣ They write programs to translate code into machine language

# Machine Language (2)

13

- Both produce the same result

- ▣ Machine language

```
169 1 160 0 153 0 128 153 0 129 153 130  
153 0 131 200 208 241 96
```

- ▣ BASIC language

```
5 FOR I=1 TO 1000: PRINT "A";: NEXT I
```

# Assembly Language

14

- Depends on using some of the mnemonic symbols, like
  - ▣ ADD, SUB, MUL, DIV, MOV
- Easier than the machine language
- Depends on the computer architecture
- Needs an 'assembler' to translate the instructions to machine code

# Assembly Language (2)

15

## □ Assembly version of the machine code

```
1 REM PET VERSION
800 FOR AD=864TO883:READ DA:POKE AD
    ,DA:NEXT AD
810 PRINT"SYS 864 TO ACTIVATE"
820 DATA169,01,160,0,153,0
830 DATA128,153,0,129,153,0
840 DATA130,153,0,131,200,208
850 DATA241,96
```

# High-Level Language

16

- Use statements that users can understand
- It can be used very easily to solve complex problems
- They do not depend on the computer's architecture
- Ex. C#, Python, Java, PHP, and JavaScript
- It needs translators to generate machine code



# High-Level Language (2)

17

- Code like this gets translated to machine code



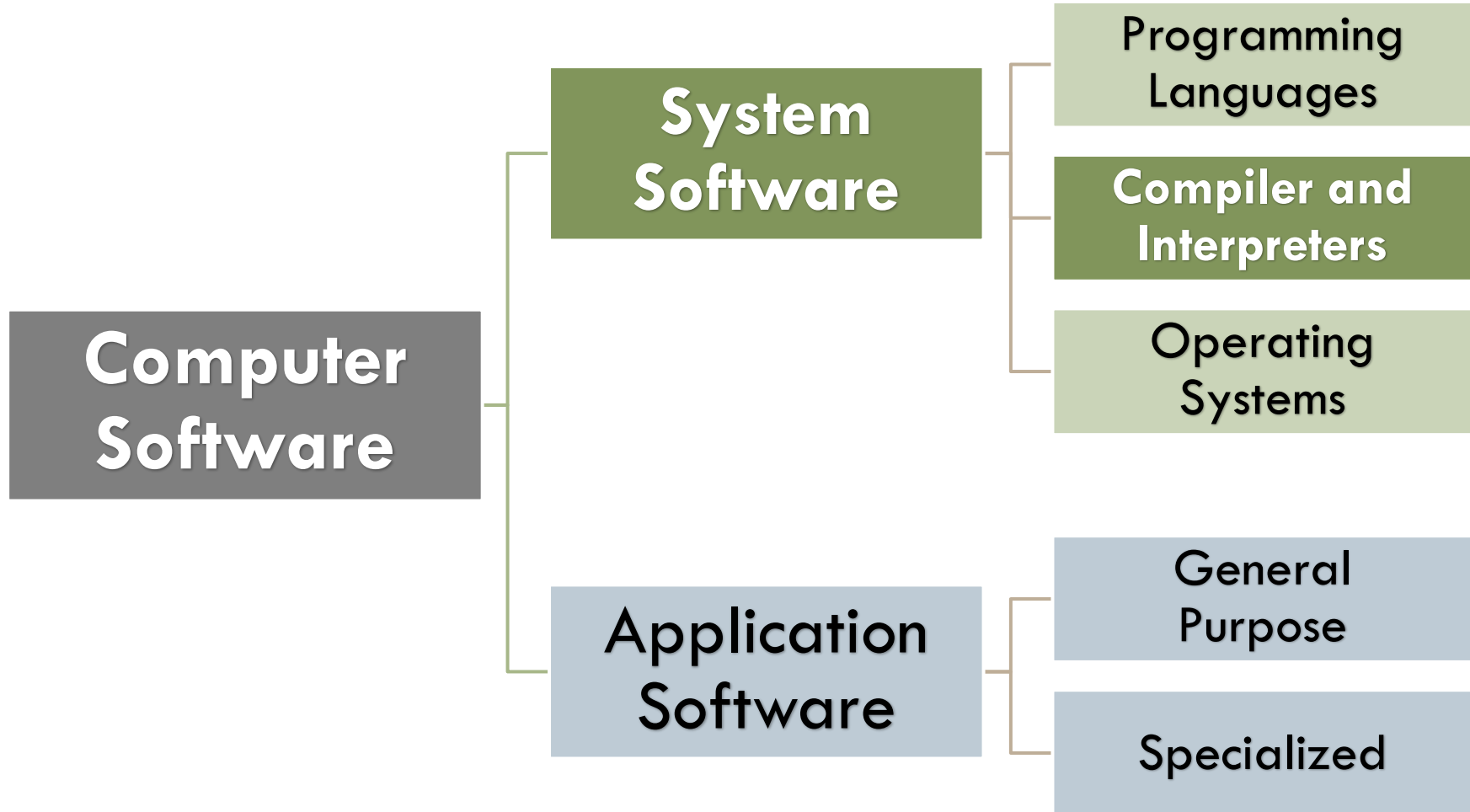
```
- DATA investCalc;
  value = 0;
  DO WHILE (value < 50000);
    value + 1200;
    value + value * 0.05;
    year + 1;
    OUTPUT;
  END;
RUN;
```

```
00000000 fc 31 c0 8e c0 8e d8 8e d0 bc 00 7c 89 e6 bf 00 |.1.....|...|
00000010 06 b9 00 01 f3 a5 89 fd b1 08 f3 ab fe 45 f2 e9 |.....E...|...|
00000020 00 8a f6 46 bb 20 75 08 84 d2 78 07 80 4e bb 40 |...F.u...x...N.@|
00000030 8a 56 ba 88 56 00 e8 fc 00 52 bb c2 07 31 d2 88 |.V.V...R...1...|
00000040 6f fc 0f a3 56 bb 73 19 8a 07 bf 87 07 b1 03 f2 |o...V.s.....|...|
00000050 ae 74 0e b1 0b f2 ae 83 c7 09 8a 0d 01 cf e8 c5 |.t.....|...|
00000060 00 42 80 c3 10 73 d8 58 2c 7f 3a 06 75 04 72 05 |.B...s.X...;u.r.|
00000070 48 74 0d 30 c0 04 b0 88 46 b8 bf b2 07 e8 a6 00 |Ht.0...F.....|...|
00000080 be 7b 07 e8 b2 00 8a 56 b9 4e e8 8e 00 eb 05 b0 |.f.....V.N.....|
00000090 07 e8 b0 00 30 e4 cd 1a 89 d7 03 7e bc b4 01 cd |...0.....~...|...|
000000a0 16 75 0d 30 e4 cd 1a 39 fa 72 f2 8a 46 b9 eb 16 |.u.0...9.r..F...|
000000b0 30 e4 cd 16 88 e0 3c 1c 74 f1 2c 3b 3c 04 76 0e |0...<.t...<.v...|
000000c0 2c c7 3c 04 77 c9 98 0f a3 46 0c 73 c2 88 46 b9 |.<.w...F.s..F...|
000000d0 be 00 08 8a 14 89 f3 3c 04 9c 74 0a c0 e0 04 05 |.....<.t...|...|
000000e0 be 07 93 c6 07 80 53 f6 46 bb 40 75 08 bb 00 06 |.....S.F.@....|
000000f0 b4 03 e8 59 00 5e 9d 75 06 8a 56 b8 80 ea 30 bb |...Y.^u..V...0...|
00000100 00 7c b4 02 e8 47 00 72 86 81 bf fe 01 55 aa 0f |.|...G.r...F...U...|
00000110 85 7c ff be 85 07 e8 19 00 ff e3 b0 46 e8 24 00 |.|.....F.$...|...|
00000120 b0 31 00 d0 eb 17 0f ab 56 0c be 78 07 e8 eb ff |.1.....V.x...|...|
00000130 89 fe e8 03 00 be 85 07 ac a8 80 75 05 e8 04 00 |.....U...|...|
00000140 eb f6 24 7f 53 bb 07 00 b4 0e cd 10 5b c3 8a 74 |...$.S.....[.t...|
00000150 01 8b 4c 02 b0 01 56 89 e7 f6 46 bb 80 74 13 66 |...L...V...F...t.f|
00000160 6a 00 66 ff 74 08 06 53 6a 01 6a 10 89 e6 48 80 |j.f.t..Sj.j...H...|
00000170 cc 40 cd 13 89 fc 5e c3 20 20 a0 0a 44 65 66 61 |@...^...Defa|...|
00000180 75 6c 74 3a a0 0d 8a 00 05 0f 01 06 07 0b 0c 0e |ult;.....|...|
00000190 83 a5 a6 a9 0d 0c 0b 0a 09 08 0a 0e 11 10 01 3f |.....?|...|
000001a0 bf 44 4f d3 4c 69 6e 75 f8 46 72 65 65 42 53 c4 |.DO.Linu.FreeBS...|
000001b0 66 bb 44 72 69 76 65 20 00 00 80 8f b6 00 00 00 |f.Drive .....|
```

# Application Languages

18

- Programming languages built into applications
  - VBScript in Microsoft Office
- Database language
  - Create files, forms, queries, and reports without the writing of any programs
  - Oracle, Access, SQL



# Compilers and Interpreters

- **Compilers and Interpreters** are programs that translate the source code into computer code
  - ▣ Human write source code using our language (a high-level language)
  - ▣ Compilers and interpreters turn the source code into machine language (a low-level language)
    - Code that a computer processor uses

# Compilers and Interpreters Differences

21

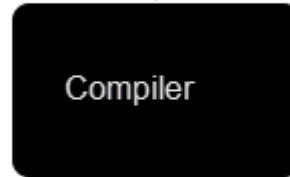
- An interpreter **directly executes** the instructions one line at a time in the source programming language
- A compiler **translates** all the instructions into efficient machine code

# Compilers

22

- Translate the **entire source code** into an executable or compiled file
- Languages that use a compiler
  - ▣ C#, C++, Java

```
func greet() = {  
    Console.println("Hello, World!")  
}
```



The magic happens here!

```
10100111100  
11110011001  
10010010010  
10110111001  
11101111011
```

# Interpreters

- Translate and execute one instruction at a time
  - ▣ Slower than the compiled code
  - ▣ Uses a program to interpret the single line of code
- Interpreted language examples
  - ▣ Scripting languages, such as JavaScript and VBScript
  - ▣ Web languages, such as PHP and Perl

# Summary

24

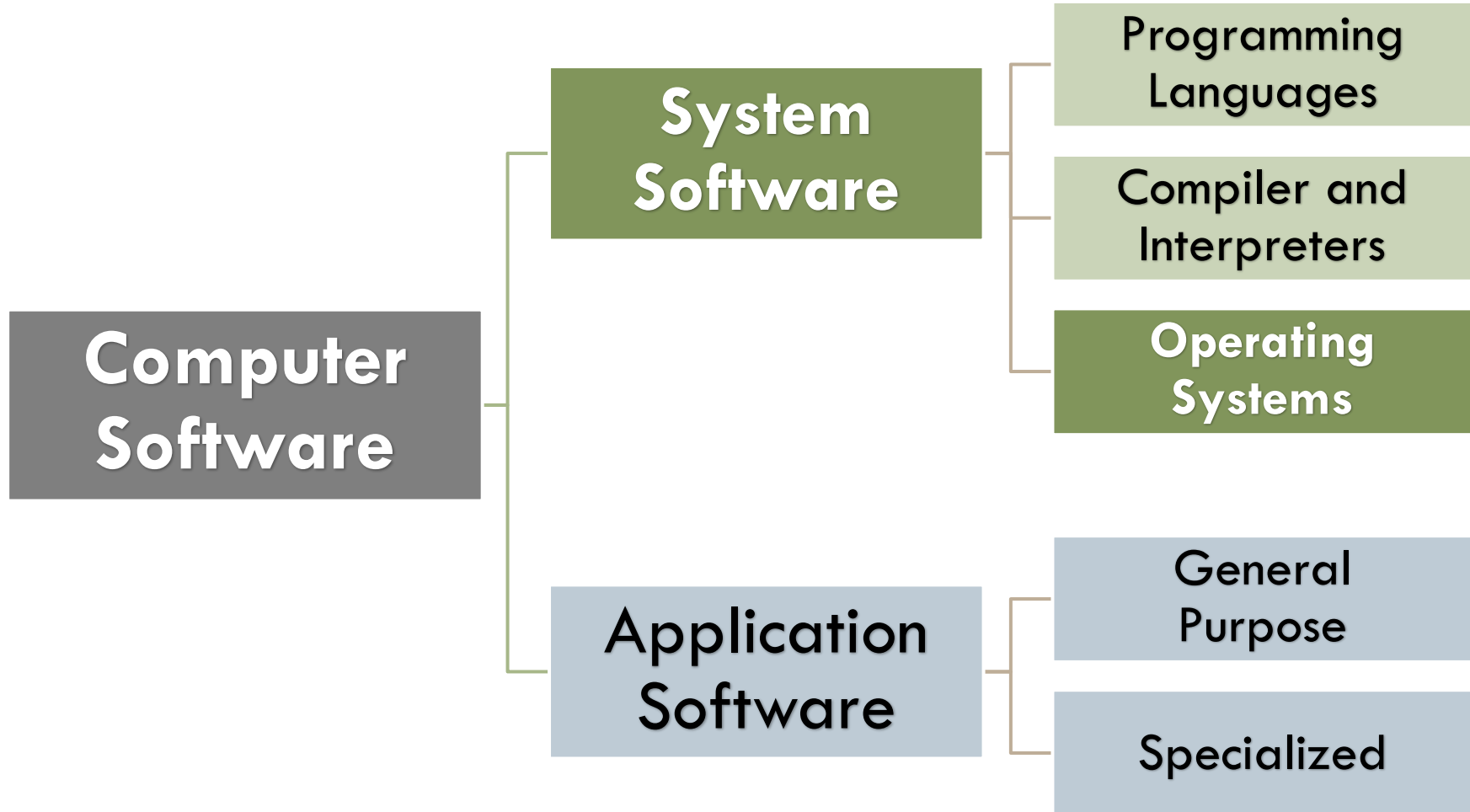
- Computer software has two major subcategories
  - ▣ **System software** and **application software**
- System software operates the hardware and allows the system to run (not for the user)
- Programming languages allow programmers to write software
  - ▣ High-level languages are closer to how humans talk
  - ▣ Low-level languages are closer to how computers talk
- Compilers and interpreters convert human instructions to machine language





# Learning Objectives

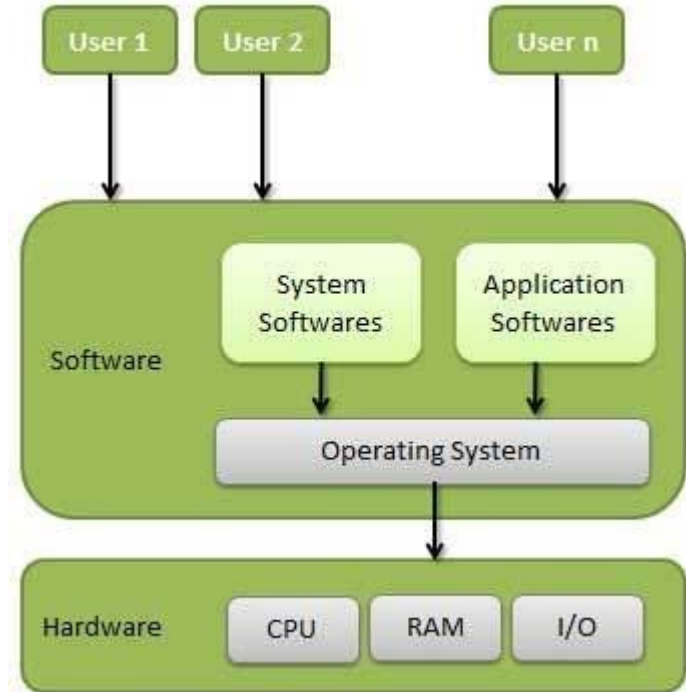
- Define an operating system
- State the primary functions of the OSs
- List the primary types of OSs
- Describe the function of each type of OS
- List the three main types of OS families
- Give an example of a brand in each OS family



# An Operating System

28

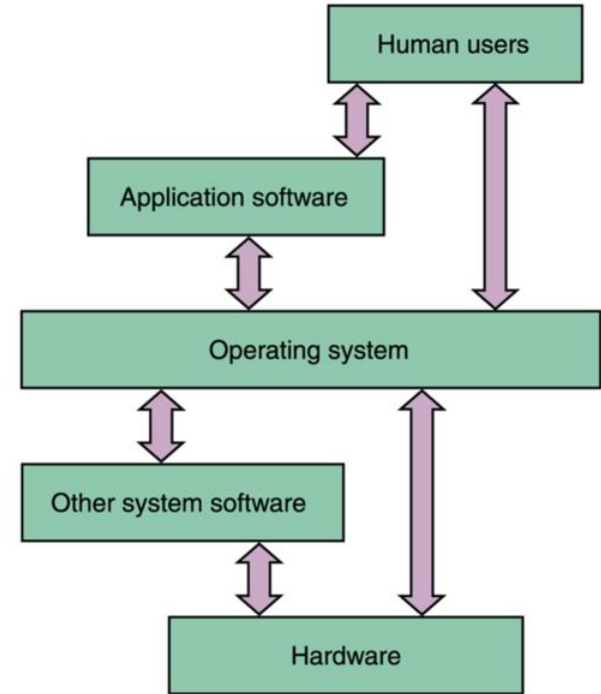
Is system software that acts as an **interface** between the **user** and the **computer hardware** and provides common services for computer programs



# An Operating System

29

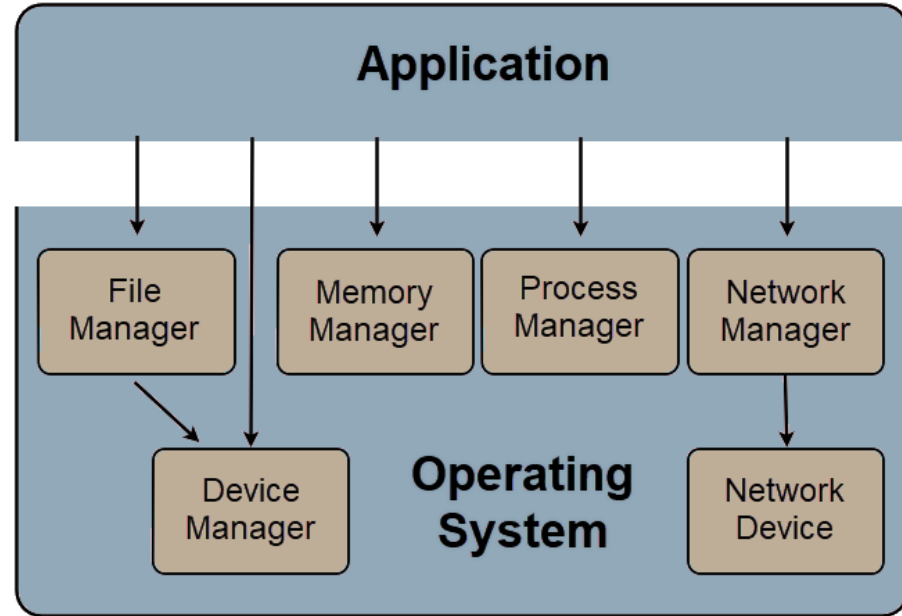
- Users and applications software **talk to the OS**
- The OS **talks to the hardware** and other system software
- The OS **informs the users** of changes



# Important Functions

30

- ❑ Memory Management
- ❑ Processor Management
- ❑ Device Management
- ❑ File Management



# Memory Management

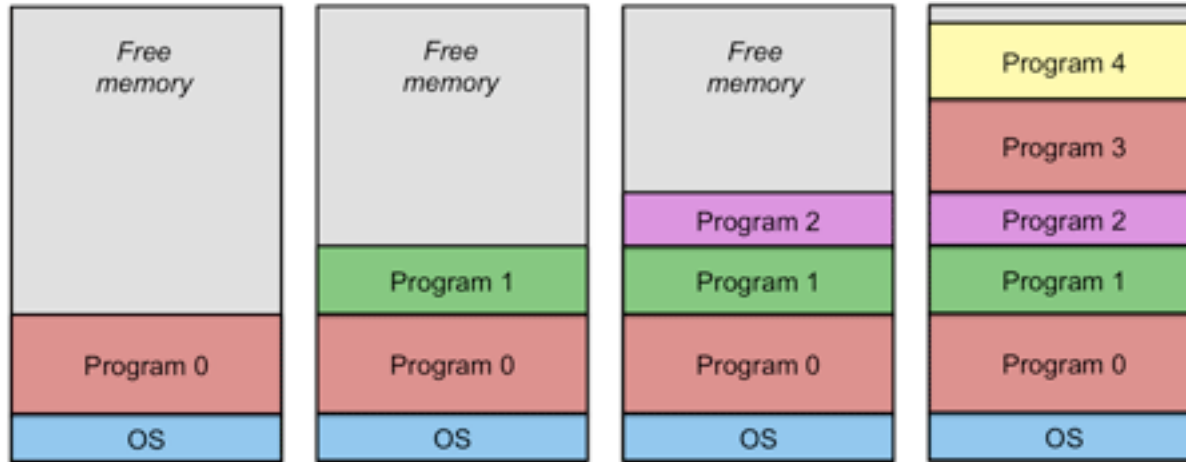
31

- Keeps tracks of primary memory
  - ▣ i.e., what parts of it are used by which application, and what parts are free
- When multiple programs are running, the OS decides which process will get memory, when, and how much

# Memory Management (2)

32

- Allocates memory when a process makes a request

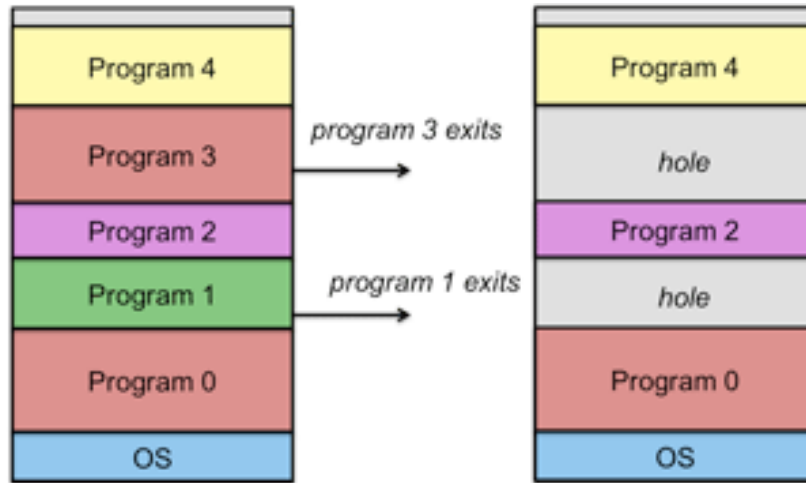




# Memory Management (3)

33

- Deallocates the memory when a process no longer needs it or the process exits or terminates



# Processor Management

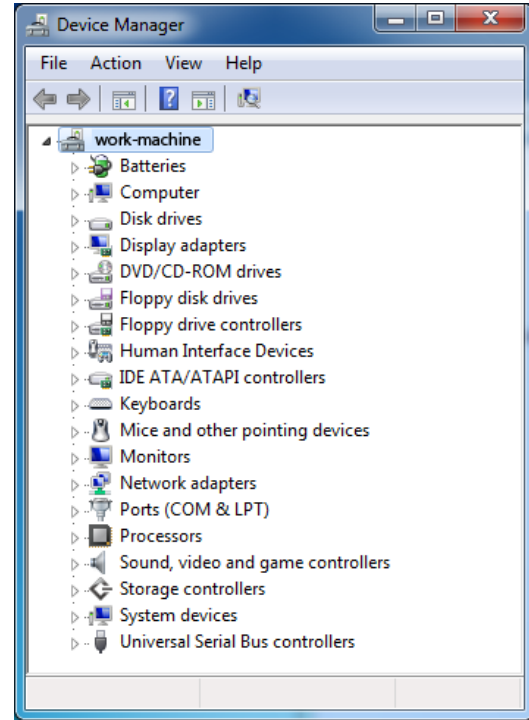
34

- Keeps track of the processor(s) and the status of all processes
- Allocates the processor (CPU) to a process
- Deallocates the processor when a process no longer needs it

# Device Management

35

- ❑ Keeps track of all devices
- ❑ Decides which process gets the device when and for how much time
- ❑ Allocates the device to a process
- ❑ Deallocates devices



# File Management

- Keeps track of information, location, uses, status, etc., on the storage device
  - ▣ The OS manages the file system
- Decides who gets the resources (files)
- Allocates the resources
- Deallocates the resources

# Other Important Activities

37

- **Security** – Prevents unauthorized access to programs and data
- **Control over system performance** – Recording delays between request for a service and response from the system
- **Job accounting** – Keeping track of time and resources used by various jobs and users

# Other Important Activities (2)

- **Error detecting aids** – Production of dumps, traces, error messages, and other debugging and error detecting aids
- **Coordination between other software and users** – Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems

# Primary Types of Operating Systems

39

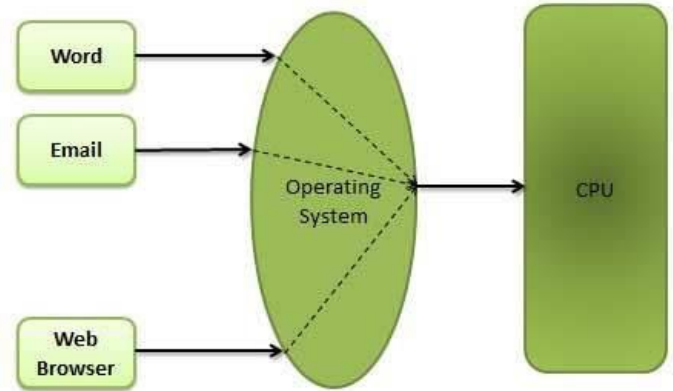
- ❑ Single-tasking and Multi-tasking
- ❑ Single-user and Multi-user
- ❑ Multi-processing
- ❑ Distributed
- ❑ Embedded
- ❑ Real-Time



# Single-tasking and Multi-tasking OS

40

- A single-tasking system can only run **one program** at a time
- A multi-tasking system run **multiple programs** at the same time
  - ▣ Most modern OSs are multitasking
  - ▣ Windows, Android, Linux, macOS

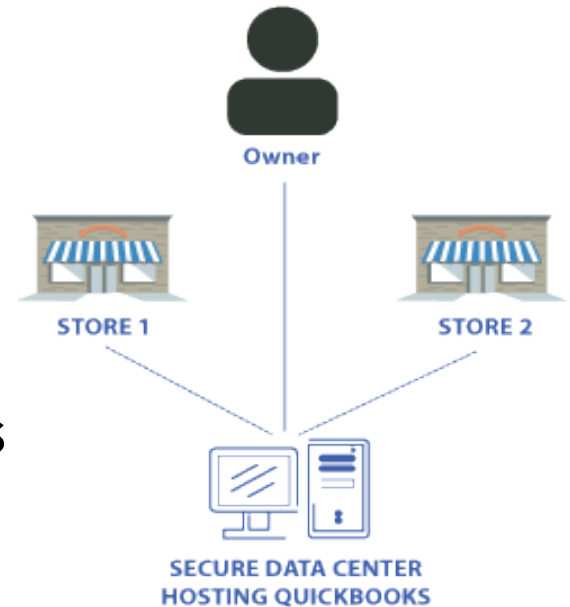




# Single-user and Multi-user OS

41

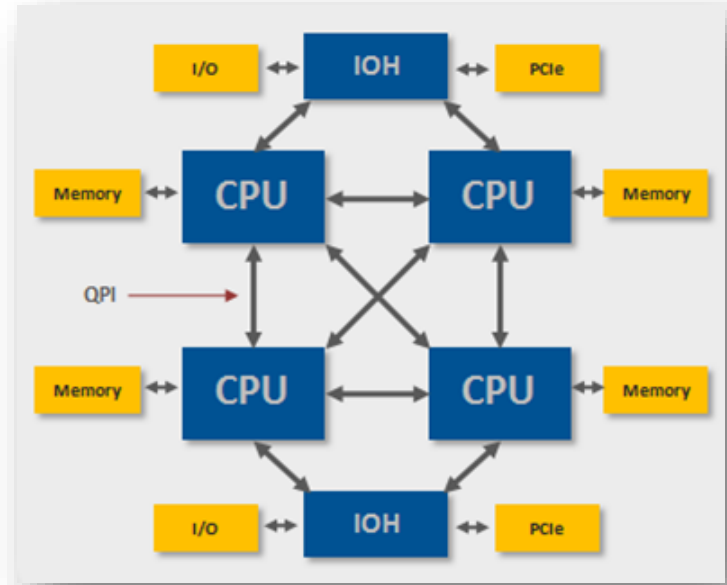
- A single-user OS only allows **one user** to use the interface at a time
  - ▣ The OS and resources **are dedicated** to that user
- A multi-user OS shares the resources among all logged in users



# Multiprocessing Operating System

42

- Here, a single process runs on two or more processors
- All the processing and their management happens at the same time
  - ▣ Called as parallel processing



# Distributed Operating System

43

- This type of OS distributes the processing power to many computers
  - ▣ Similar to how a computer can have multiple processors
- Each computer processes data as part of larger network



# Embedded Operating System

44

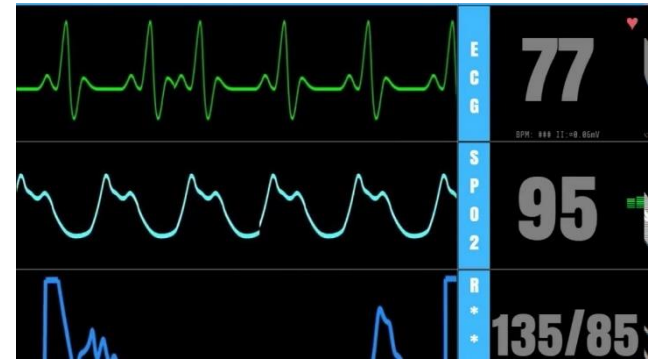
- Embedded in a chip on the device
- Performs specific functions for the device
  - ▣ Most smart devices have an embedded OS
- Terminal, copy machine, camera, washing machine, TV



# Real-Time Operating System (RTOS)

45

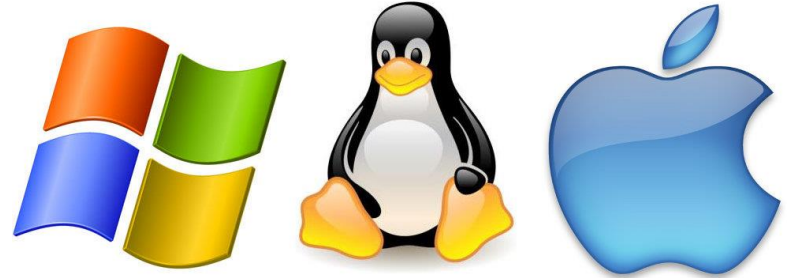
- Input and output operations are happening immediately, without delay
- Time-critical devices:
  - ▣ medical devices
  - ▣ scientific experiments
  - ▣ industrial systems, transportation control systems



# OS Families

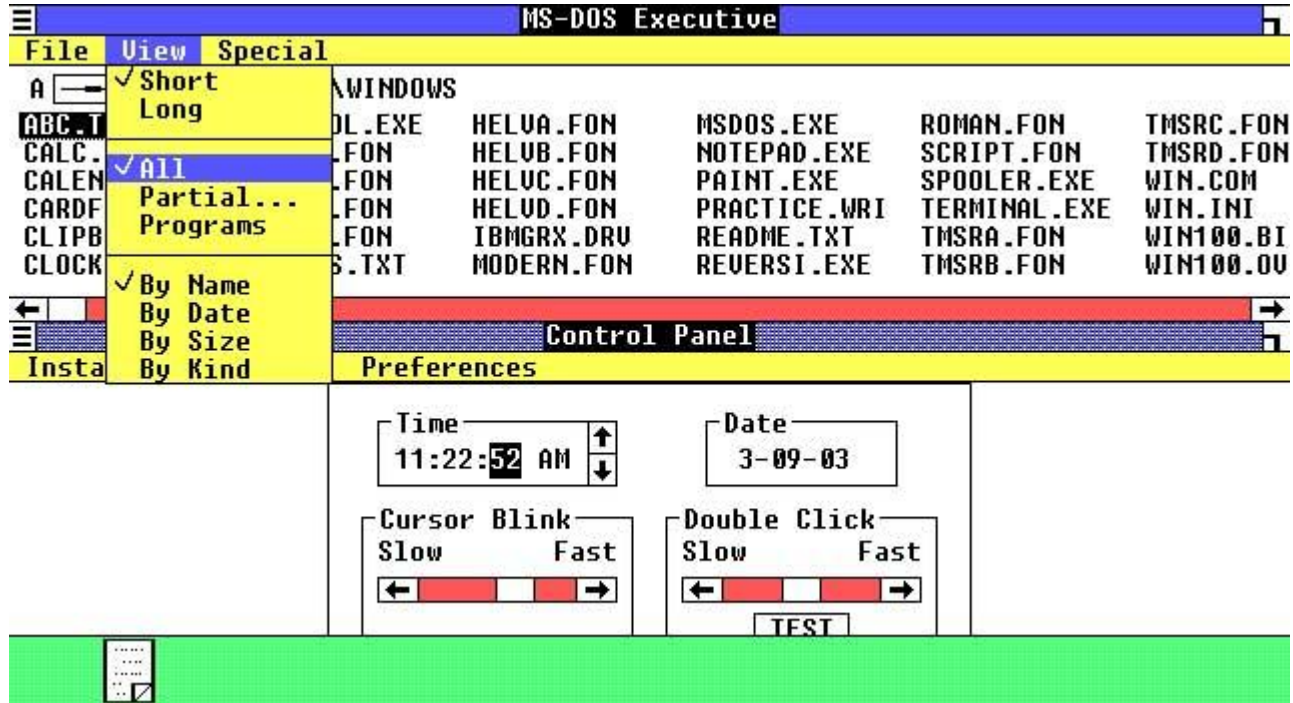
46

- Windows
  - ▣ Desktop
- Macintosh
  - ▣ Desktop; Mobile (iOS)
- Unix-like
  - ▣ Servers; Mobile (Android)



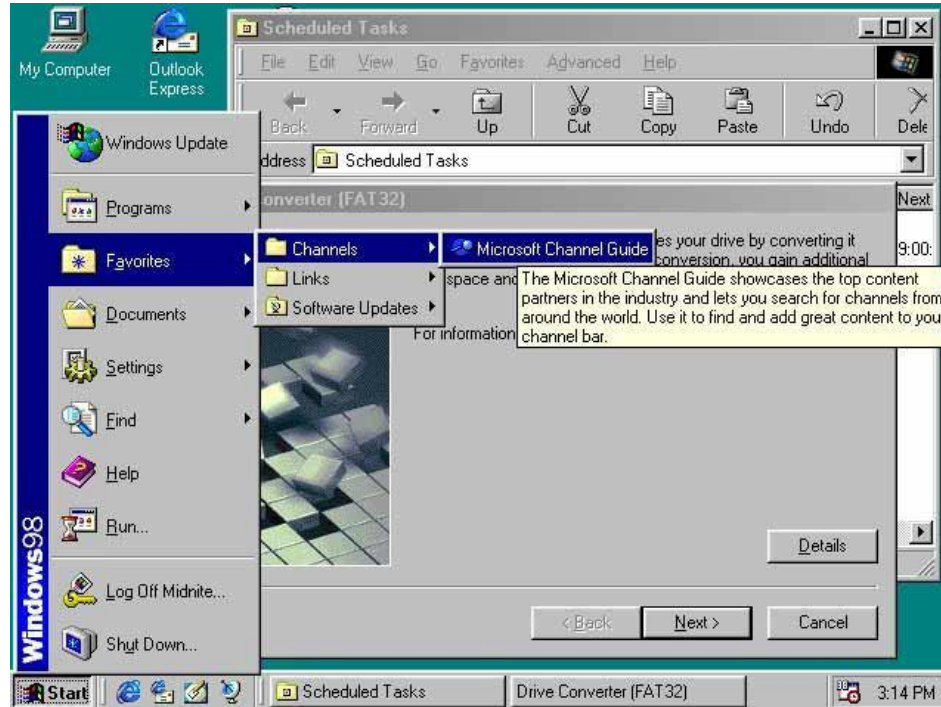
# MS Windows 1.x

47



# MS Windows 98

48





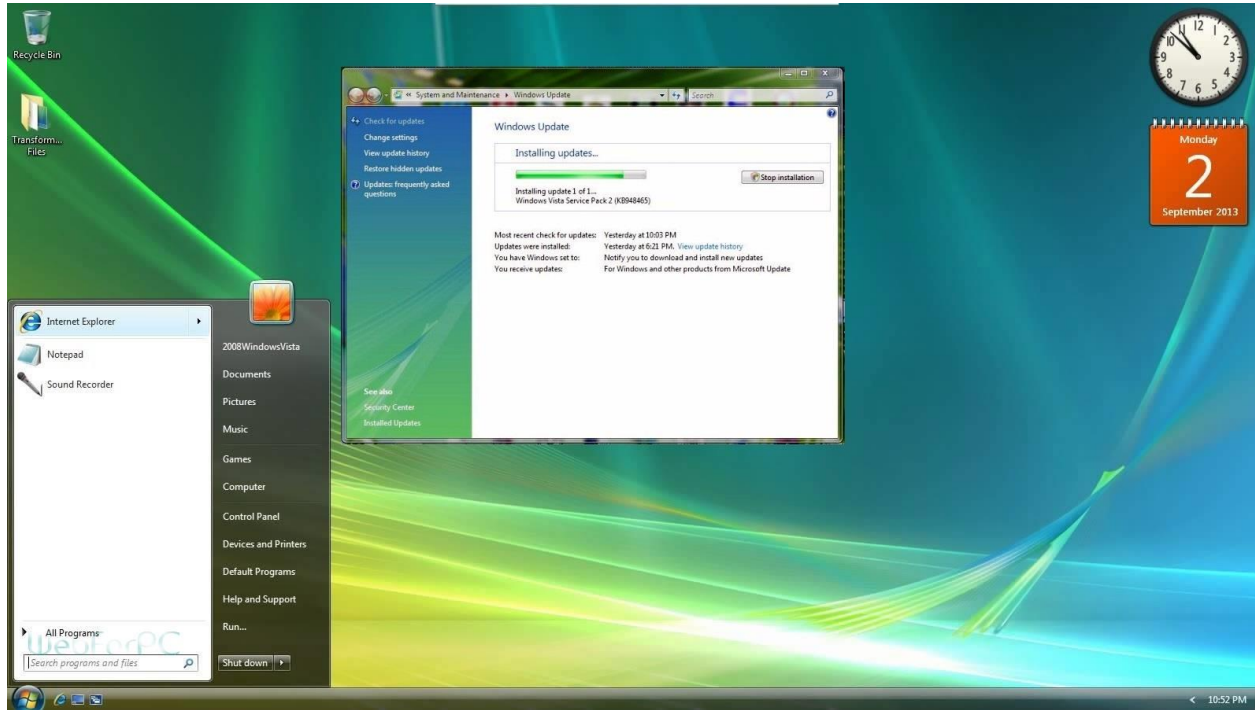
# Windows XP

49



# Windows Vista

50



# Windows 7

51



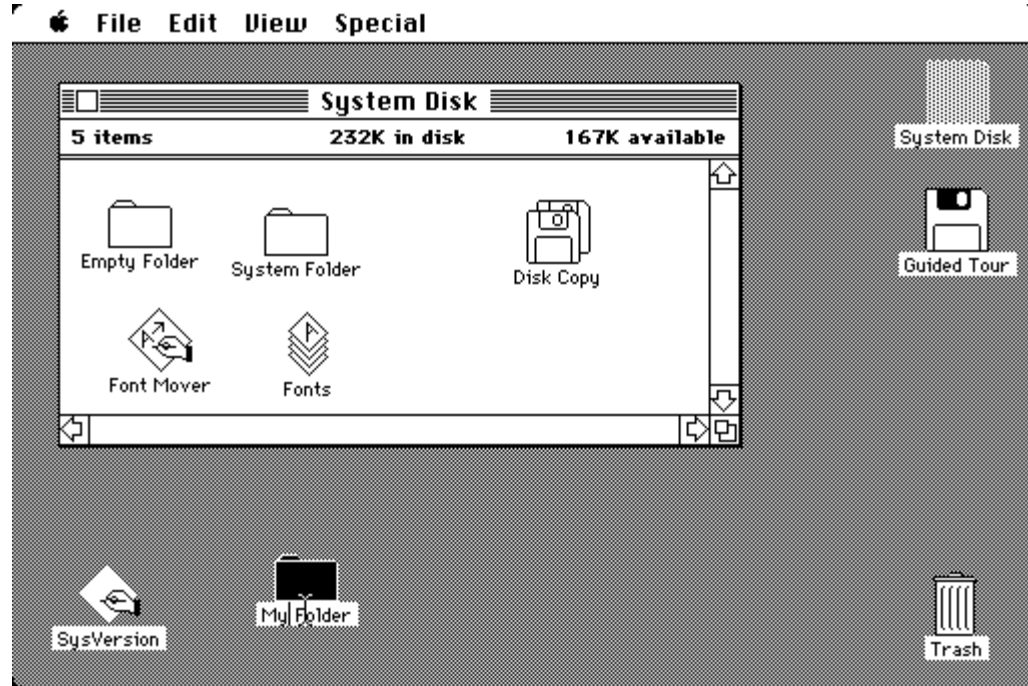
# Windows 8/10

52



# System 1, 2, 3, 4 (Macintosh)

53



# Mac OS 8

54



# Mac OS X

55

- Created incremental builds called OS X using the name of cats for each build



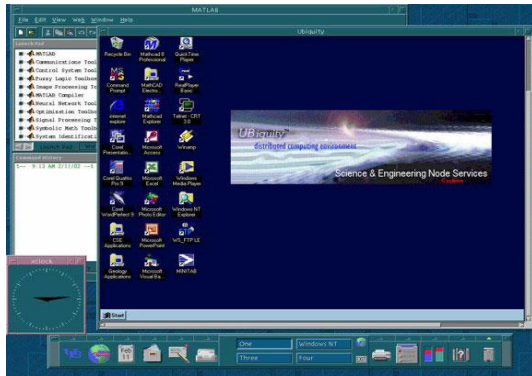
Box/Mac App Store artwork for OS X. Left to right: Cheetah/Puma (1), Jaguar (2), Panther (3), Tiger (4), Leopard (5), Snow Leopard (6), Lion (7), Mountain Lion (8).



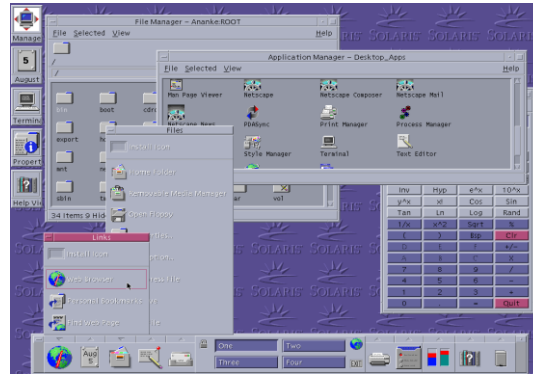
# Unix-like Systems

56

## Unix



## Solaris



## Linux





# Linux

57

## Command Line

```
adarsh@adarsh-fosbytes: /tmp
File Edit View Search Terminal Help
eb'

linux-image-4.12.0- 100%[=====>] 48.83M  2.30MB/s   in 20s

2017-07-07 13:33:11 (2.40 MB/s) - 'linux-image-4.12.0-041200-generic_4.12.0-041200.201707022031_amd64.deb' saved [51203494/51203494]

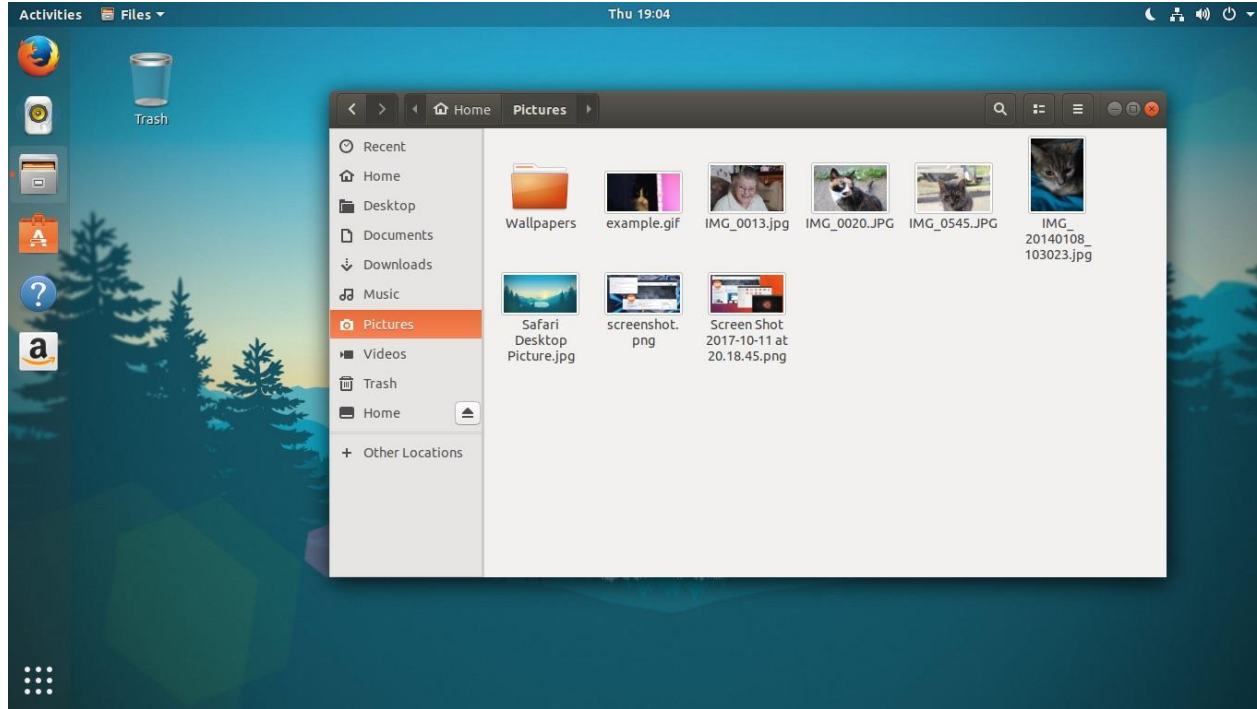
adarsh@adarsh-fosbytes: /tmp$ sudo dpkg -i *.deb
[sudo] password for adarsh:
Selecting previously unselected package linux-headers-4.12.0-041200.
(Reading database ... 138317 files and directories currently installed.)
Preparing to unpack linux-headers-4.12.0-041200_4.12.0-041200.201707022031_all.d
eb ...
Unpacking linux-headers-4.12.0-041200 (4.12.0-041200.201707022031) ...
Selecting previously unselected package linux-headers-4.12.0-041200-generic.
Preparing to unpack linux-headers-4.12.0-041200-generic_4.12.0-041200.2017070220
31_amd64.deb ...
Unpacking linux-headers-4.12.0-041200-generic (4.12.0-041200.201707022031) ...
Selecting previously unselected package linux-image-4.12.0-041200-generic.
Preparing to unpack linux-image-4.12.0-041200-generic_4.12.0-041200.201707022031
_amd64.deb ...
Done.
Unpacking linux-image-4.12.0-041200-generic (4.12.0-041200.201707022031) ...
█
```

## GUI



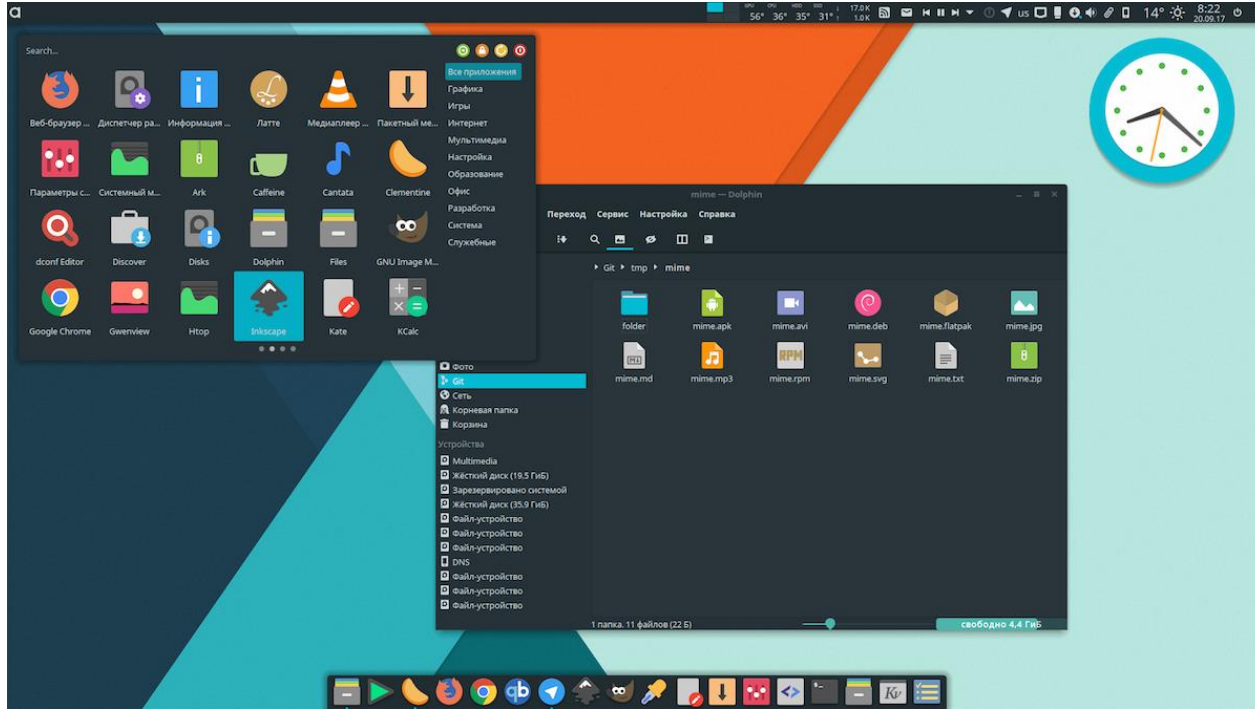
# Gnome Desktop

58



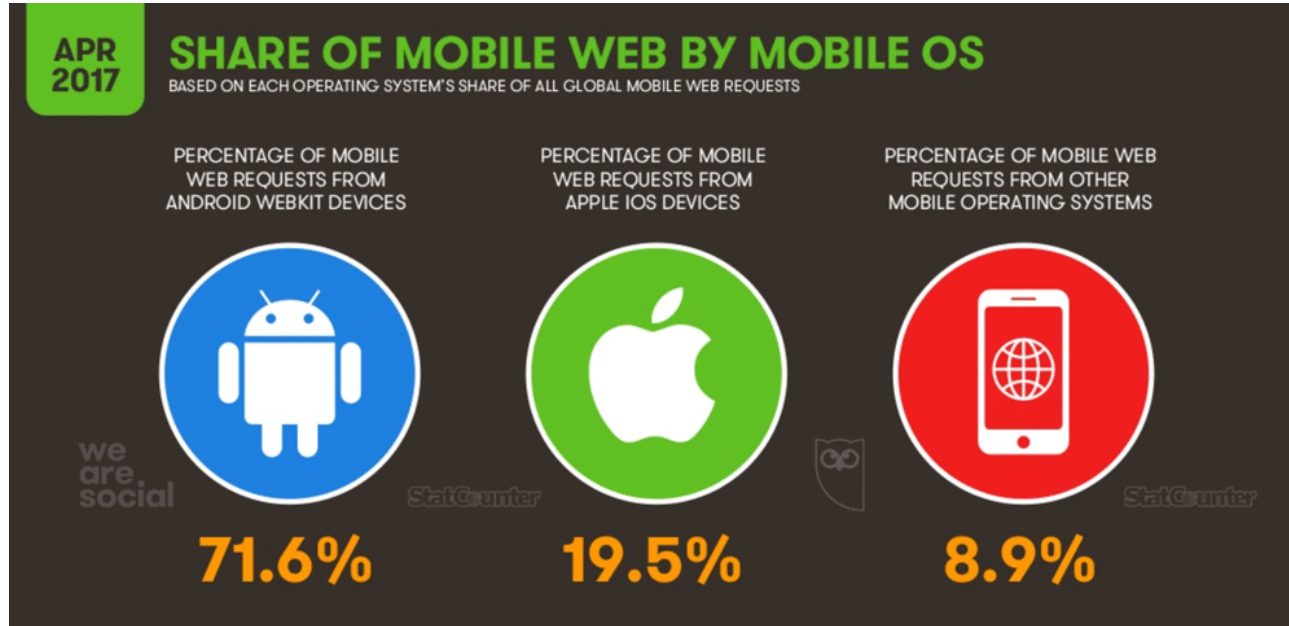
# KDE Desktop

59



# Mobile Operating Systems

60



# Summary

- An OS is system software that acts as an **interface** between the **user** and the **computer hardware** and provides common services for computer programs
- An OS performs the **primary system functions** of
  - ▣ Memory, Processor, Device, and File Management
- Types of operating systems
  - ▣ Single-tasking and Multi-tasking, Single-user and Multi-user, Multiprocessing, Distributed, Embedded, and Real-Time
- Primary OS Families
  - ▣ Windows, Macintosh, Unix-like

## 3.3: Application Software

### 3.1: System Software (Part 1)

- ▶ Programming Languages
- ▶ Compilers and Interpreters

### 3.2: System Software (Part 2):

- ▶ Operating Systems

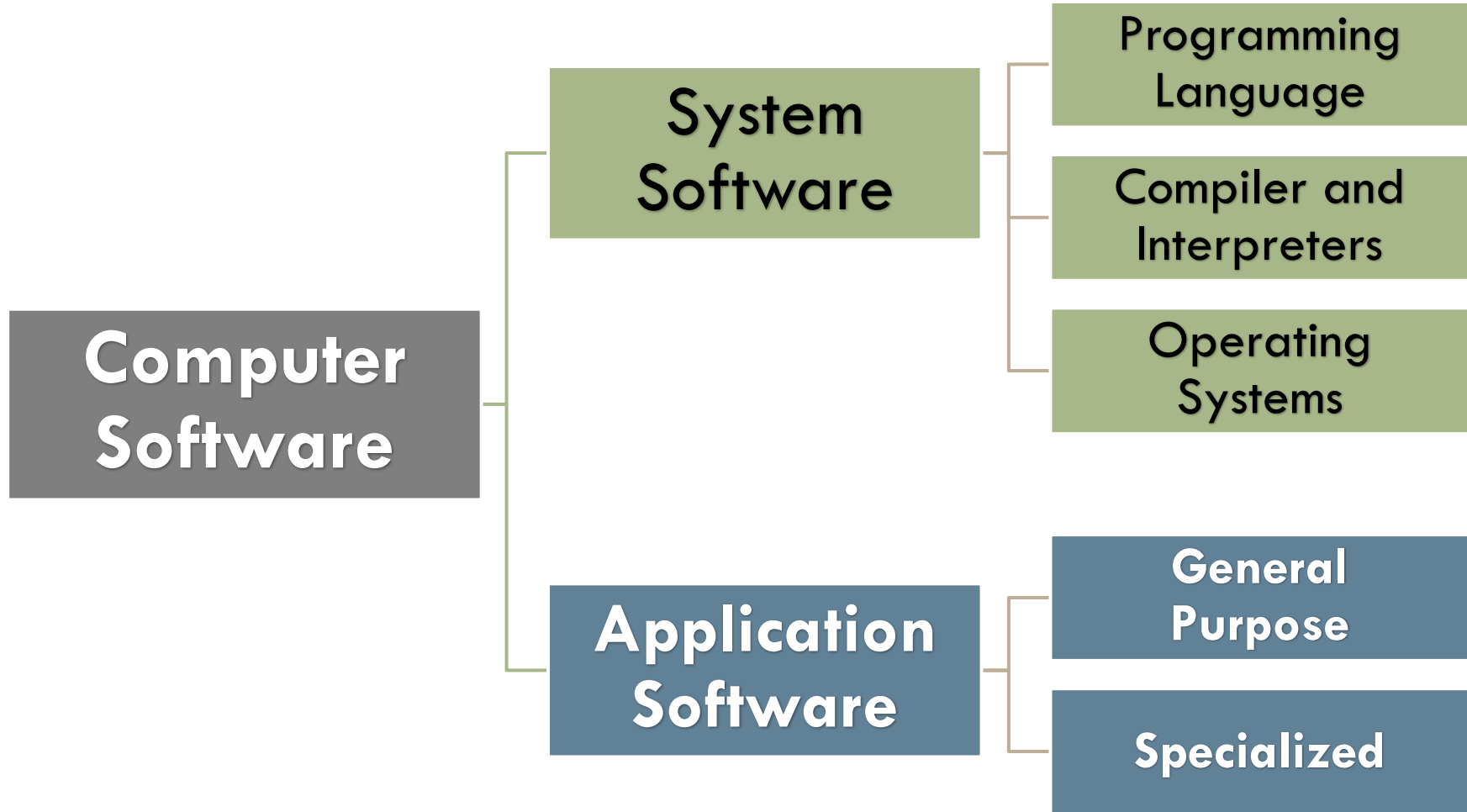
### 3.3: Application Software



# Learning Objectives

63

- Describe the difference between **application software** and **system software**
- List the two categories of application software
- Describe the two types of application interfaces

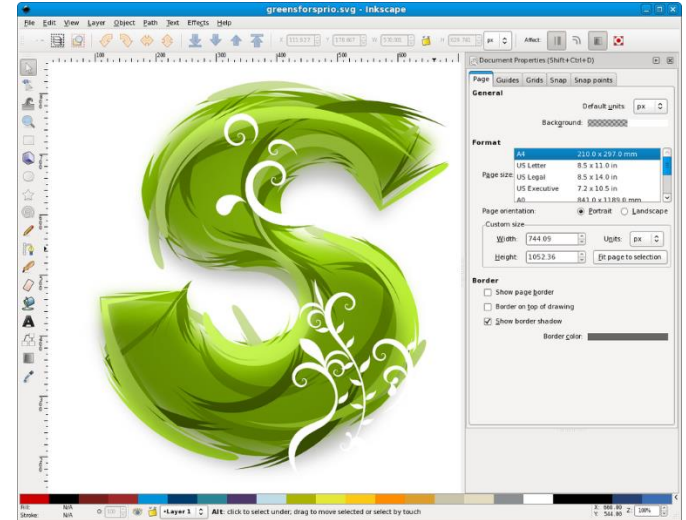




# Application Software






65

- Application software is software used by user
- Two categories
  - Basic applications
    - general-purpose
  - Specialized applications
    - For specific, advanced tasks



# Basic Applications

66

<b>Examples of Basic Application software</b>		
<b>Word Processor</b> E.g. Microsoft Word	Used to write and format texts, insert tables and pictures	
<b>Presentation</b> E.g. Microsoft PowerPoint	Used to design slides for business and education	
<b>Instant Message</b> E.g. WhatsApp	Used to interact with others using short text messages, audio, or video	
<b>Web Browser</b> E.g. Google Chrome	Used to open websites on the Internet	
<b>Compression</b> E.g. 7-zip	Used to compress or extract files	

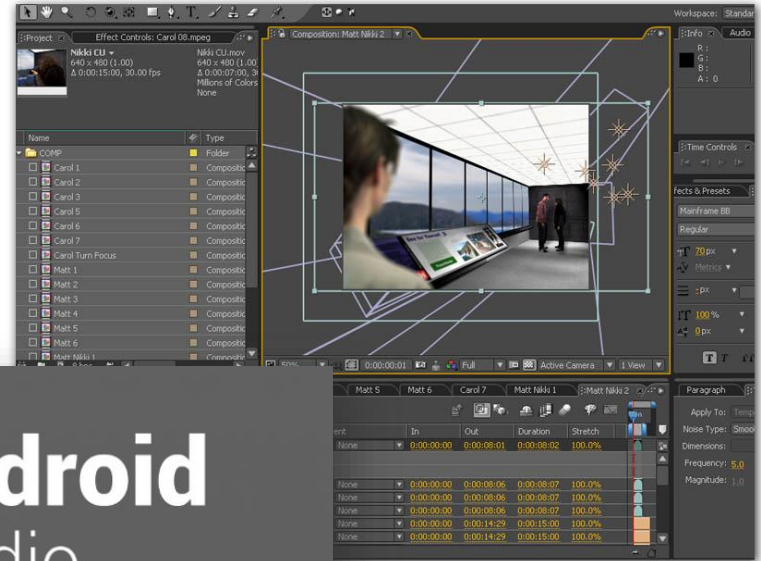
# Specialized Applications

67

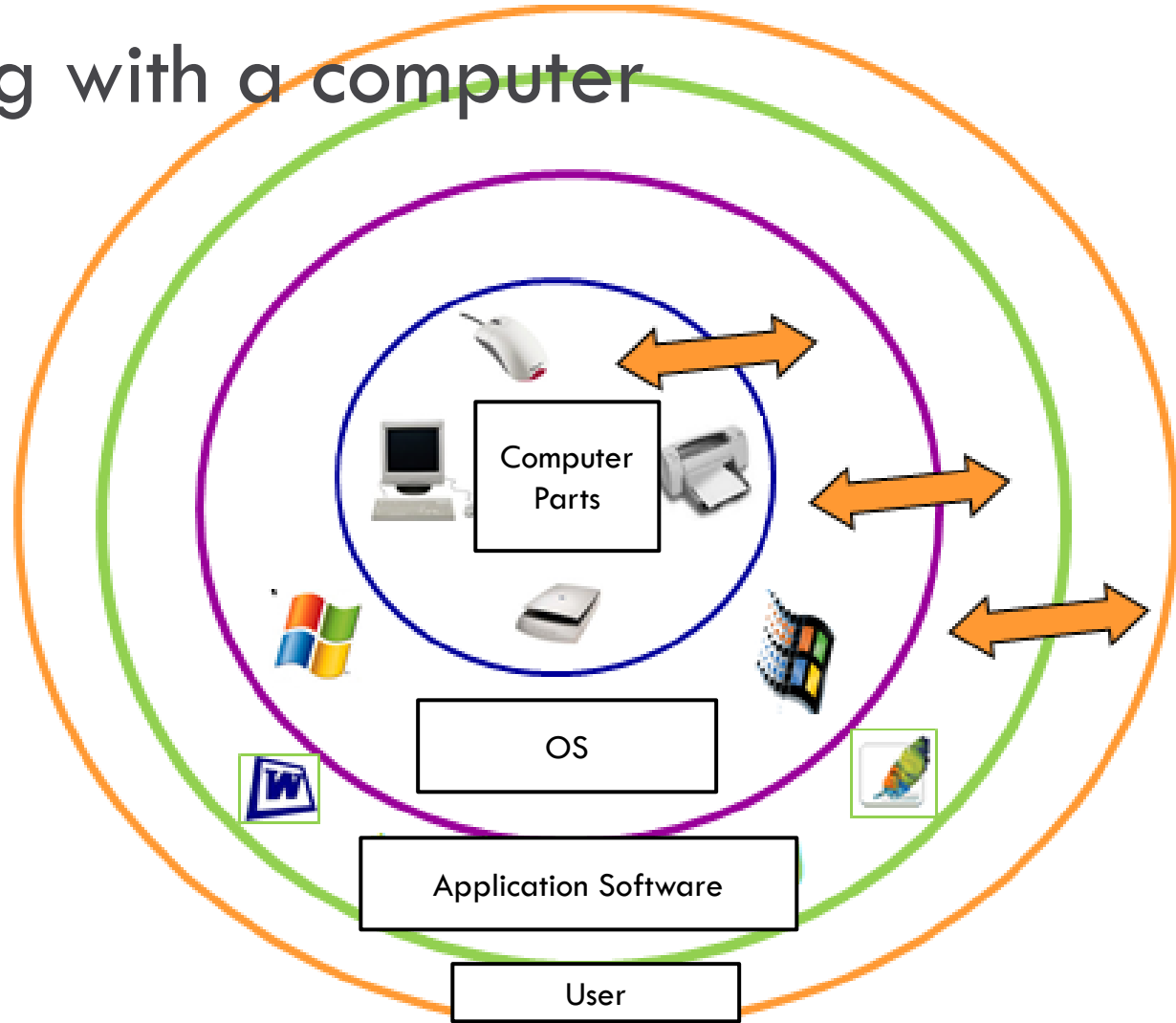
- Focus on specific disciplines and occupations
- Usually requires a high degree of skill
- Examples:
  - ▣ Graphics Programs (Adobe Photoshop)
  - ▣ Audio and Video Editing Software (Adobe Premiere)
  - ▣ Programming Platforms (Visual Studio; Android Studio)
  - ▣ Engineering Tools (AutoCAD; 3ds Max)

# Examples

68



# Interacting with a computer



# Application Interfaces

70

- **Interface:** It's the way that the user communicates with the computer
  - Command Line Interface (CLI)
  - Graphical User Interface (GUI)

```
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface gigabitethernet 0/0
Router(config-if)#no ip address
Router(config-if)#interface gigabitethernet 0/0.10
Router(config-subif)#encapsulation dot1q 10
Router(config-subif)#ip address 10.1.10.1 255.255.255.0
Router(config-subif)#interface gigabitethernet 0/0.20
Router(config-subif)#encapsulation dot1q 20
Router(config-subif)#ip address 10.1.20.1 255.255.255.0
Router(config-subif)#exit
Router(config)#interface service-engine 0/1
Router(config-if)#ip unnumbered gigabitethernet 0/0.20
Router(config-if)#exit
Router(config)#exit
Router#
Jun 29 10:51:58.211: %SYS-5-CONFIG_I: Configured from console
Building configuration...
[OK]
Router#
```

391, 8

VT400-7 -- COM1 at 9600 baud

# Command Line Interface (CLI)

71

- ❑ Commands must be typed on the keyboard
- ❑ Requires a high degree of computer knowledge
- ❑ Slow process

```
-DPIC -o .libs/libhx509_la-file.o
/bin/sh ../../libtool --tag=CC --mode=compile x86_64-pc-linux-gnu-gcc -DH
AVE_CONFIG_H -I. -I. -I../include -I../include -I./ref -I/usr/include
/et -pipe -O2 -march=native -D_LARGE_FILES= -Wall -Wmissing-prototypes -Wpoi
nter-arith -Wbad-function-cast -Wmissing-declarations -Wnested-externs -pipe
-O2 -march=native -c -o libhx509_la-sel.lo `test -f 'sel.c' || echo './'`sel
.c
libtool: compile: x86_64-pc-linux-gnu-gcc -DHAVE_CONFIG_H -I. -I. -I../in
clude -I../include -I./ref -I/usr/include/et -pipe -O2 -march=native -D_LA
RGE_FILES= -Wall -Wmissing-prototypes -Wpointer-arith -Wbad-function-cast -Wm
issing-declarations -Wnested-externs -pipe -O2 -march=native -c sel.c -fPIC
-DPIC -o .libs/libhx509_la-sel.o
/bin/sh ../../libtool --tag=CC --mode=compile x86_64-pc-linux-gnu-gcc -DH
AVE_CONFIG_H -I. -I. -I../include -I../include -I./ref -I/usr/include
/et -pipe -O2 -march=native -D_LARGE_FILES= -Wall -Wmissing-prototypes -Wpoi
nter-arith -Wbad-function-cast -Wmissing-declarations -Wnested-externs -pipe
-O2 -march=native -c -o libhx509_la-sel-gram.lo `test -f 'sel-gram.c' || ech
o './'`sel-gram.c
libtool: compile: x86_64-pc-linux-gnu-gcc -DHAVE_CONFIG_H -I. -I. -I../in
clude -I../include -I./ref -I/usr/include/et -pipe -O2 -march=native -D_LA
RGE_FILES= -Wall -Wmissing-prototypes -Wpointer-arith -Wbad-function-cast -Wm
issing-declarations -Wnested-externs -pipe -O2 -march=native -c sel-gram.c -
fPIC -DPIC -o .libs/libhx509_la-sel-gram.o

HOST="x86_64-pc-linux-gnu"
x86_64_pc_linux_gnu_CFLAGS="-pipe -O2 -march=native"
i686_pc_linux_gnu_CFLAGS="-pipe -O2 -march=native"

case "$CATEGORY" in
sys-apps/paludis)
NORMAL >> /etc/paludis/bashrc < sh << 9% : 1: 1

[exbull:0] [1:vim] | 2:zsh |
```

# Graphical User Interface (GUI)

72

- Uses pictures, windows, menus, icons to represent object and operation
- User can select any object by pointing the mouse at it and click on it
- Ease, enjoyable





# Summary

73

- System software manages system functions and interfaces with the hardware
- Application software is for the user
  - Interfaces with system software
  - Basic applications – For everyday use (web browsers, chat applications, etc.)
  - Specialized applications – Tools for professionals
- User interfaces
  - Command Line Interface (CLI) – Type in commands, work only in text
  - Graphical User Interface (GUI) – Interact with icons and visuals

# Attribution Notice

This presentation was assembled using information found online and referenced materials. Attribution is provided of borrowed materials if the author is clearly identified. Unreferenced materials are common knowledge or commonly found online without knowing the original author.