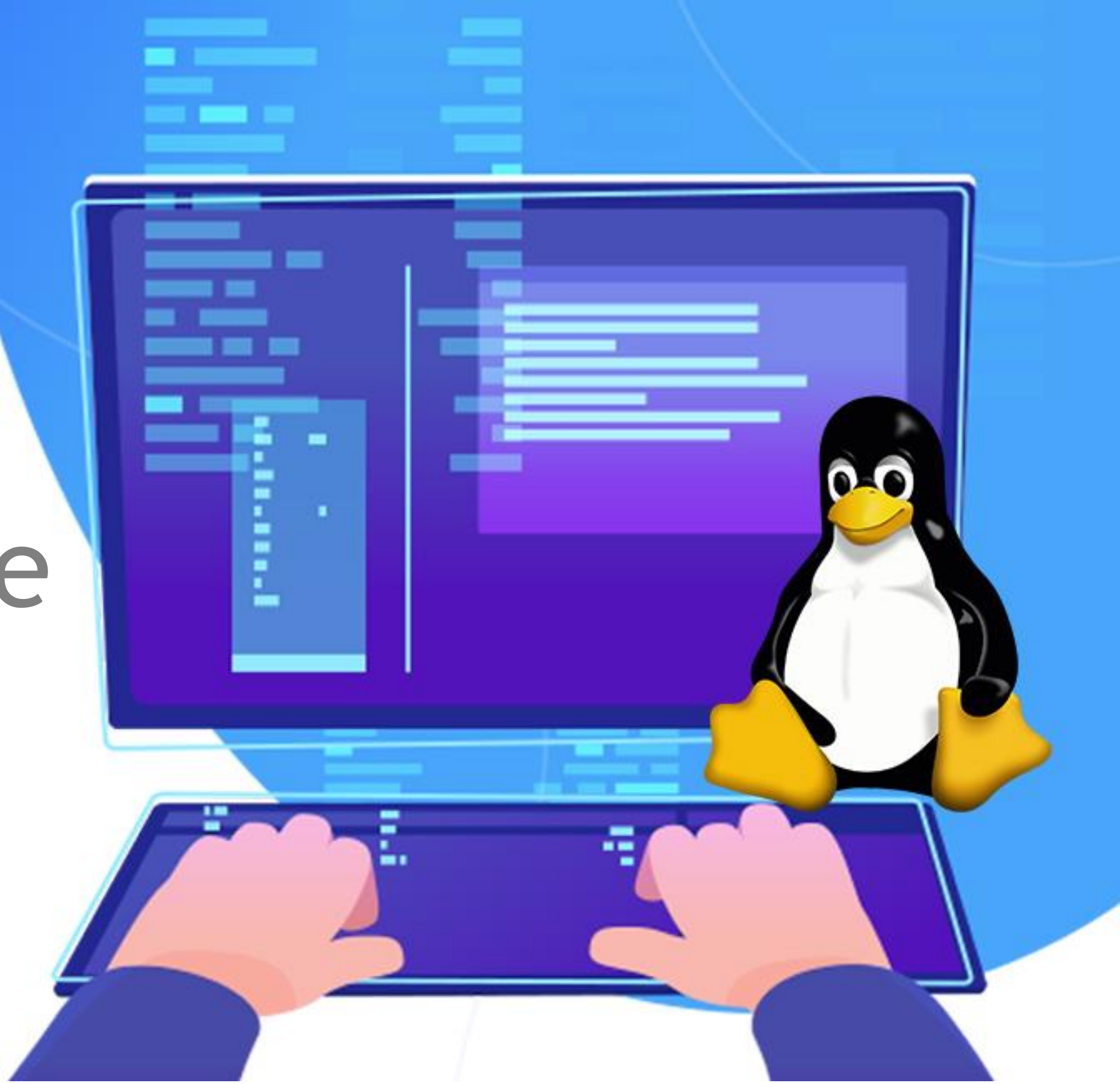
UNIX/Linux Course

Introduction



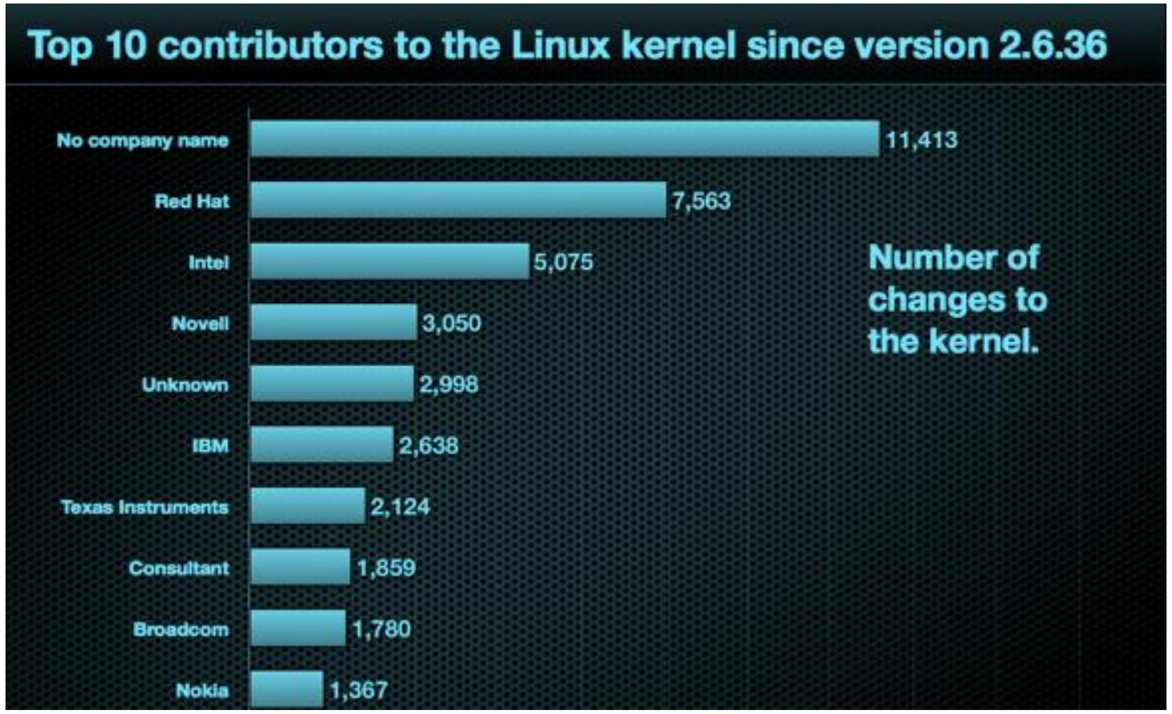
Agenda

Introduction to Linux

Introduction to Linux

Linux is a Unix-like OS developed by **Linus Torvalds** and thousands of opensource contributors

Linus Torvalds



Contributor	Number of changes
No company name	11,413
Red Hat	7,563
Intel	5,075
Novell	3,050
Unknown	2,998
IBM	2,638
Texas Instruments	2,124
Consultant	1,859
Broadcom	1,780
Nokia	1,367

Introduction to Linux

Linux is an Operating system. It is reliable and secure than others; Also it is completely opensource

Launch Date: 17 September 1991

Introduction to Linux

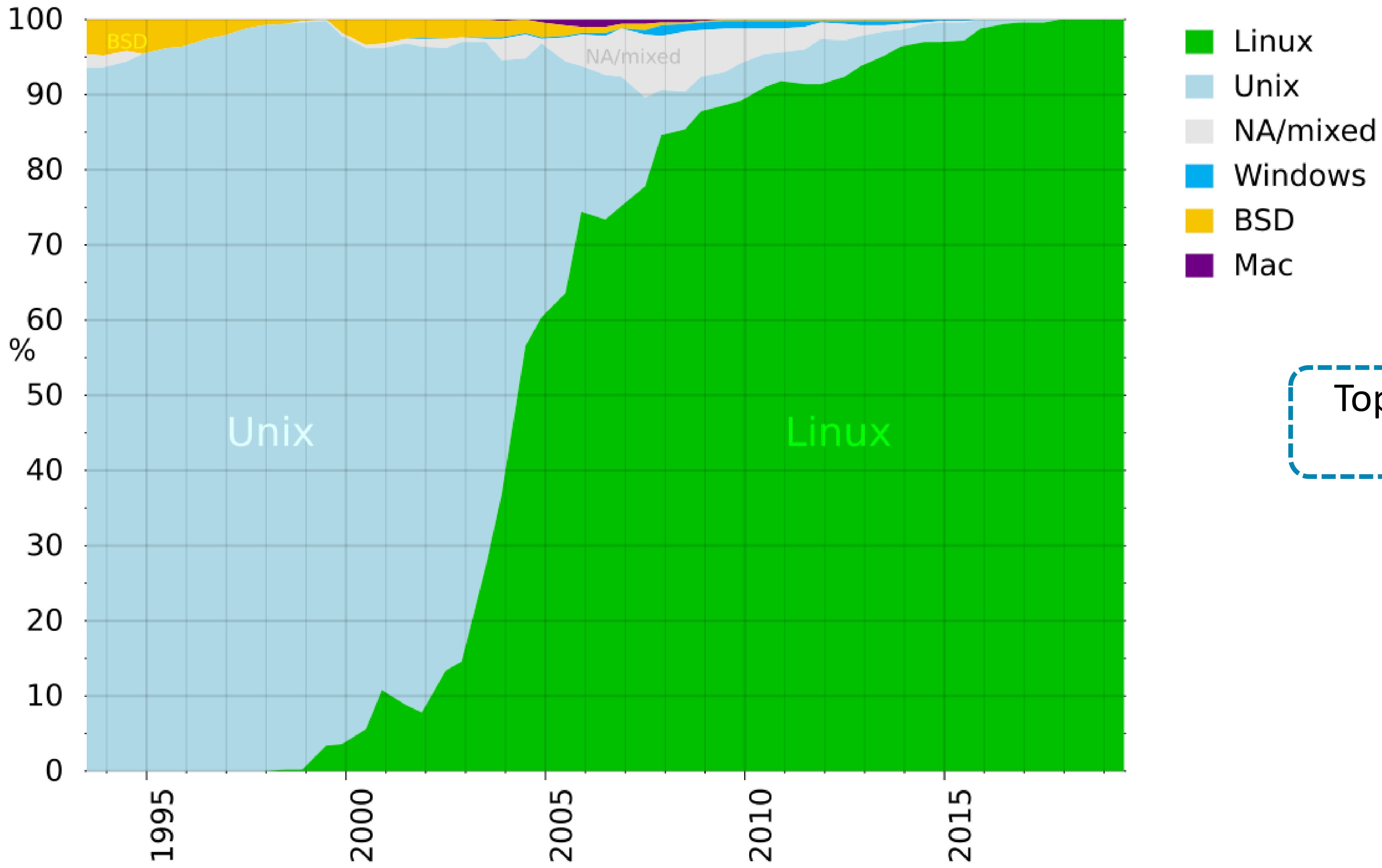
Linux is everywhere!

Linux is in your smartphones. 85% of all smartphones are based on Linux.

Your car uses Linux; Especially self-driving cars

Even your refrigerators need Linux to run

Introduction to Linux



Top 500 supercomputers run on Linux!

Introduction to Linux

Why use Linux?

Windows

Linux

- For a Windows server, you need to purchase the license and the expenditure increases according to the number of users
- Not many customization options available
- Windows is vulnerable to viruses and malware threats. A powerful anti-virus software is a need

- With Linux, license is free, installing software's is free and you can install Linux in any number of machines you want
- There are a lot of Linux distributions and you can choose one from it
- More secure than Windows and viruses can't easily break the kernel

Introduction to Linux

Linux Distributions

debian

fedora

ubuntu

CentOS

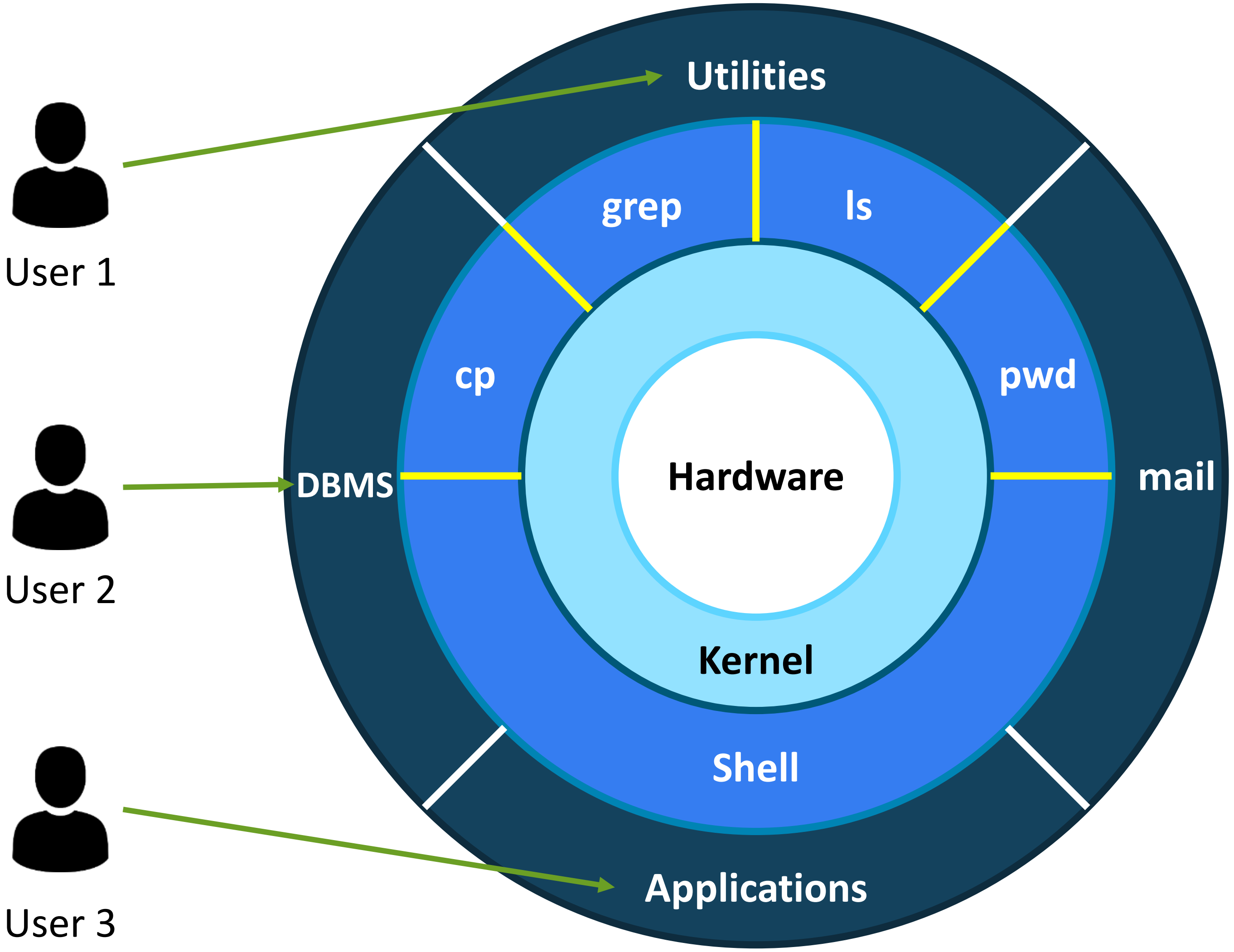
Red Hat Enterprise Linux

SUSE

Basics of Shell

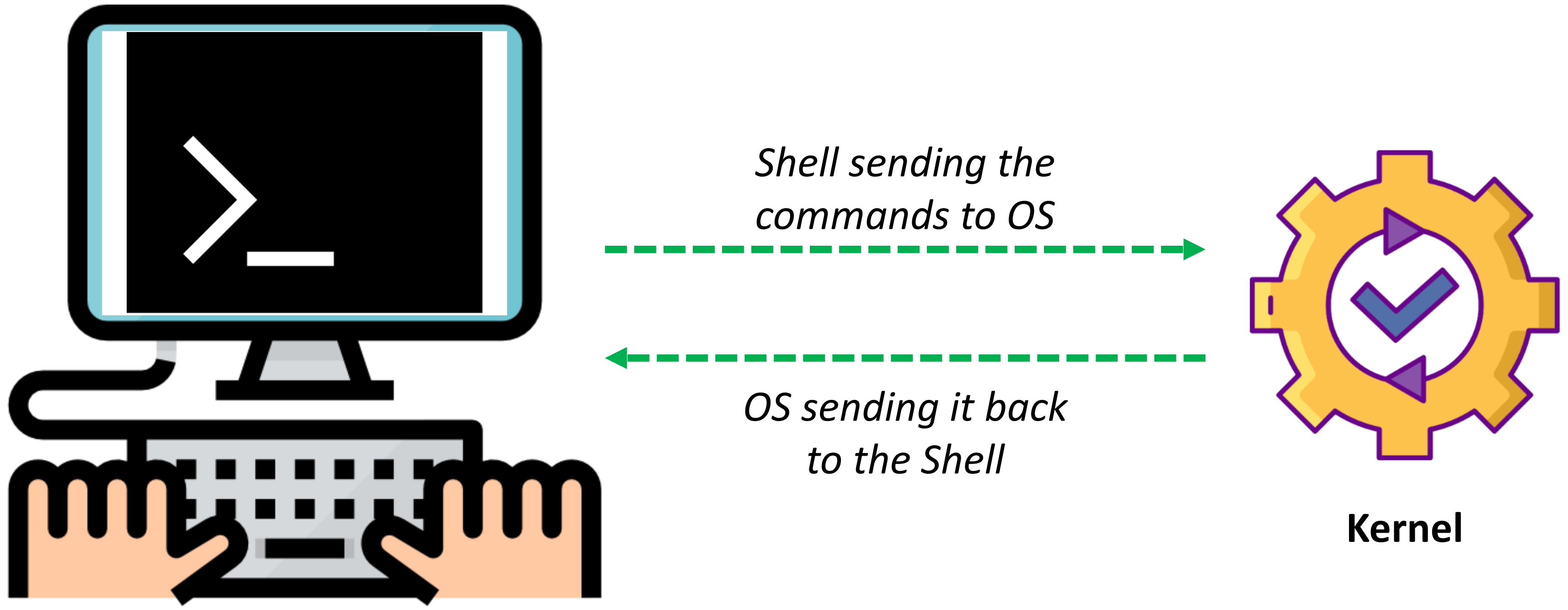
Basics of Shell

Linux Architecture



Basics of Shell

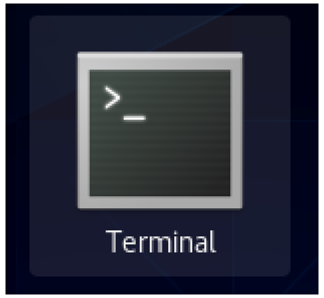
A Shell interprets the commands you have entered using a keyboard and sends it to the OS to perform them


Shell sending the commands to OS
OS sending it back to the Shell
Kernel

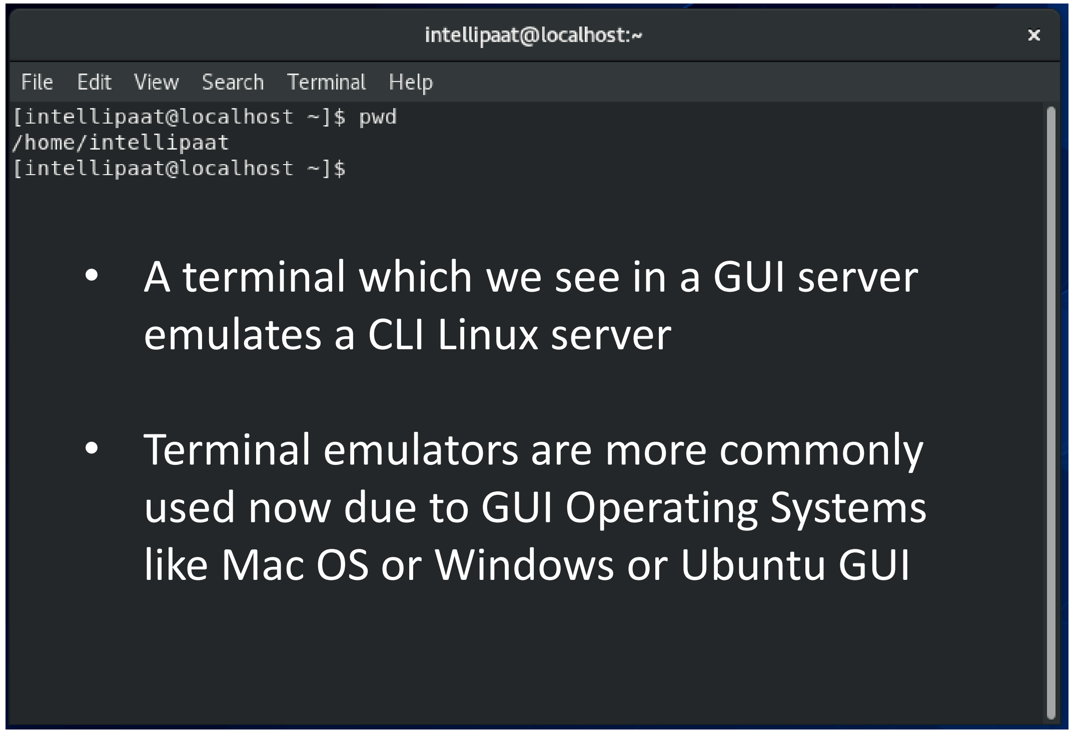
Keyboard inputs to the shell

Basics of Shell

Nowadays, we use a lot of GUI-based Linux distributions like CentOS. In these you have a terminal to contact the shell



Terminals are software's which emulate a CLI Linux system



```
[intellipaat@localhost ~]$ pwd
/home/intellipaat
[intellipaat@localhost ~]$
```

- A terminal which we see in a GUI server emulates a CLI Linux server
- Terminal emulators are more commonly used now due to GUI Operating Systems like Mac OS or Windows or Ubuntu GUI

Top Shells in Linux

bash

The Bourne Again Shell is the default shell in a lot of Linux distributions. It is the most portable shell available. Bash is the default shell of CentOS 8.

zsh

It is similar to bash or an extended version of it. It has a lot of useful features like sharing your command history across multiple terminals

fish

Friendly and Interactive shell is again an extended version of the common shell. It has great features like autocompletion of commands

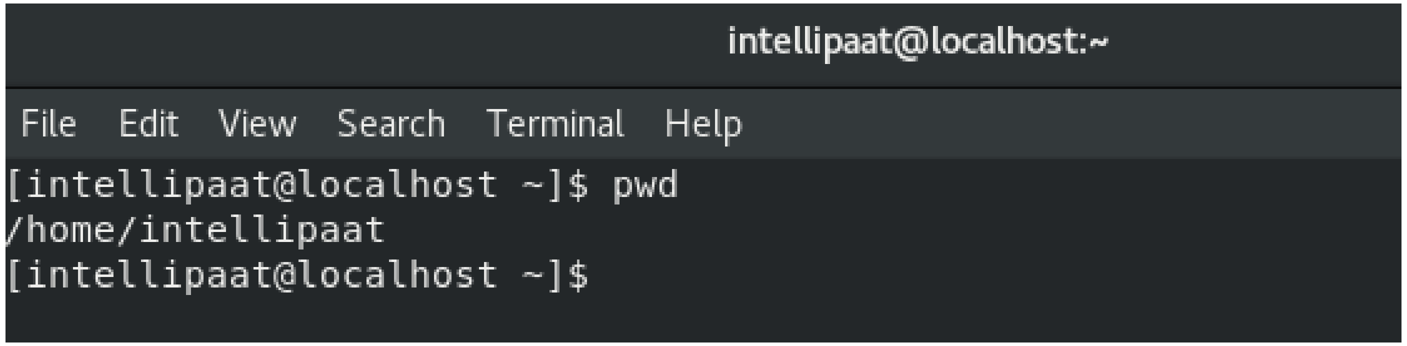
tcsh

Tenex C shell is an extended version of C shell. The plus of tcsh is its scripting language, because it will be similar for users with experience in C programming

Top Shells in Linux

A simple command to interact with the shell
This command below provides the present working directory

pwd

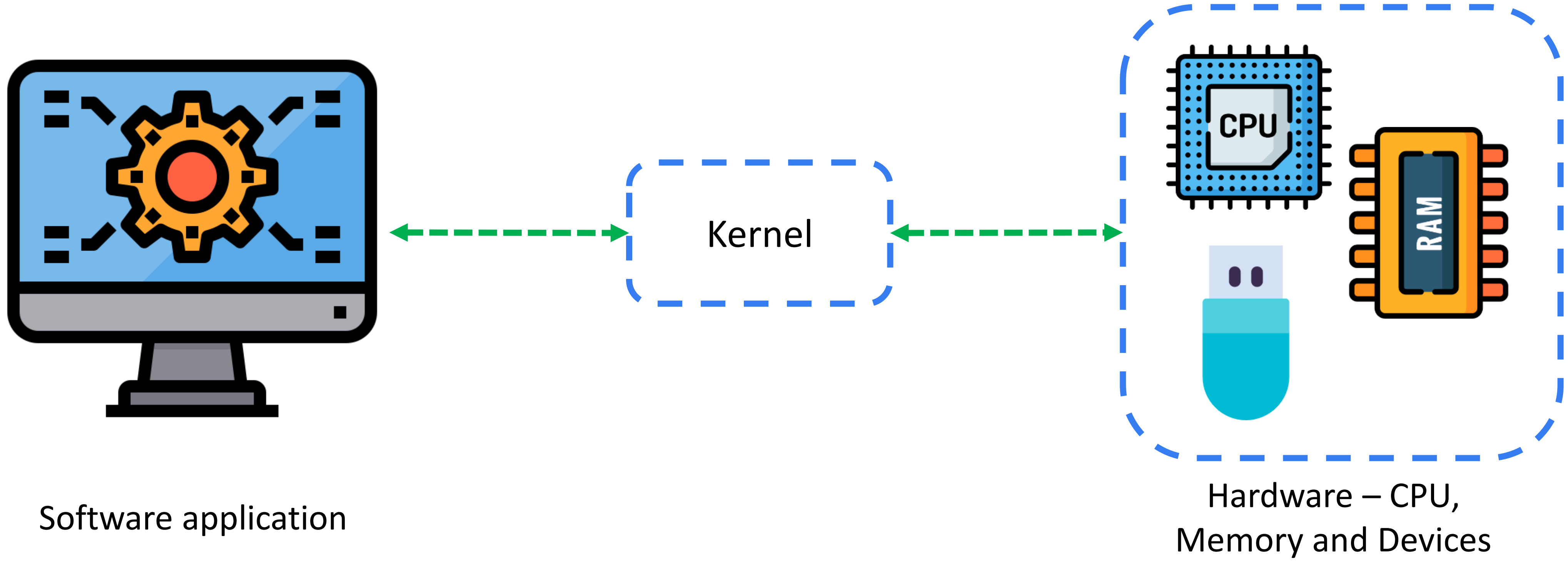


```
[intellipaat@localhost ~]$ pwd
/home/intellipaat
[intellipaat@localhost ~]$
```

Basics of Kernel

Basics of Kernel

A Linux Kernel is a UNIX-like OS kernel. It is a Computer Program which is the core interface which connects the Hardware components to the Software processes



Basics of Kernel

Top operations performed by a Kernel

1. **Resource Management** – Decides which process gets a resource for an operation

2. **Memory management** – Kernel has complete access to system memory and must efficiently manage it and allow memory access to processes

3. **Device management** – If we connect devices such as a printer or a pen drive, kernel detects it and helps the system establish connection with those peripherals

4. **System calls** – This is an interface between a process and the operating system. When the process does not have permissions to access a resource, a system call provides it without the process accessing the resource directly

Basic Linux Commands

Basic Linux Commands

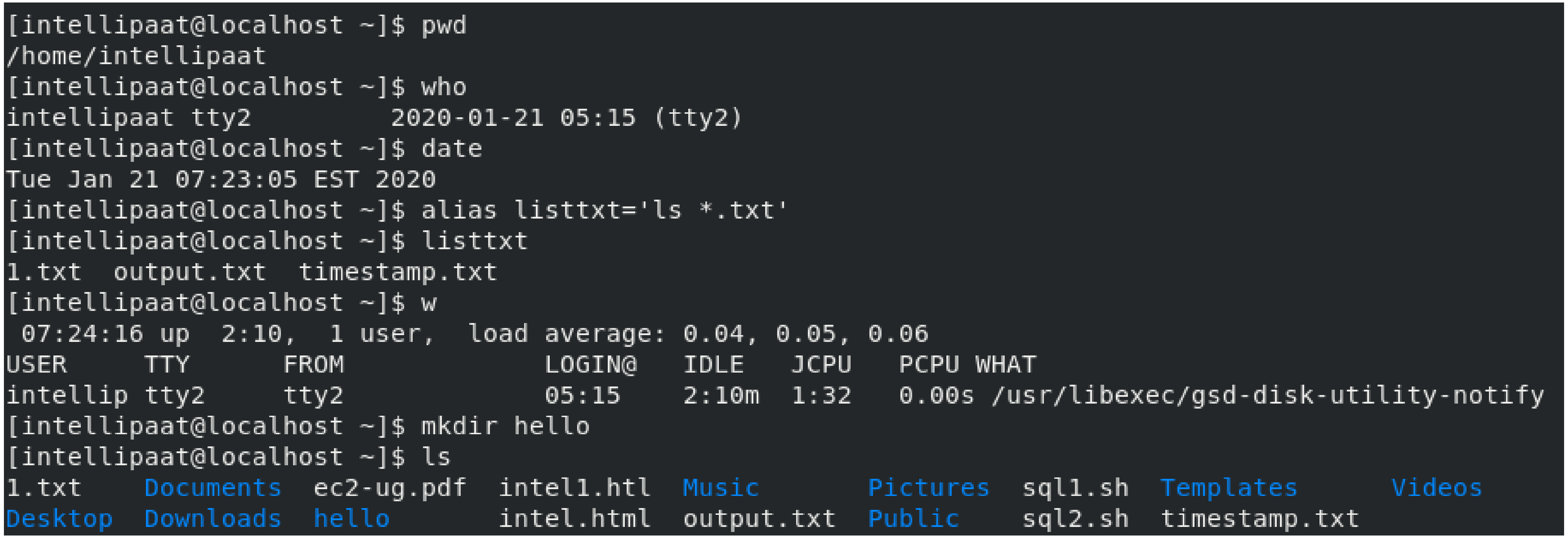
Command	Task
pwd	Shows the present working directory
whoami	Gives the current username
date	Gives the date and current time
history	Shows all the commands you have typed in recently
cp	Used to copy a file
rm	To delete a file
clear	Clears the entire terminals content
man	It is a guide to the commands
exit	Exits and closes the running terminal

Basic Linux Commands

Command	Task
who	Shows the logged in users of the system
w	Same as who but also shows the current process
mkdir	Used to make a directory
cat	Displays the contents of a file
mv	Used to move a file/folder from source to destination
alias	Give a name for a command and execute using it
echo	Prints text on the terminal
ls	Lists files and folders

Basic Linux Commands

Few examples of commands



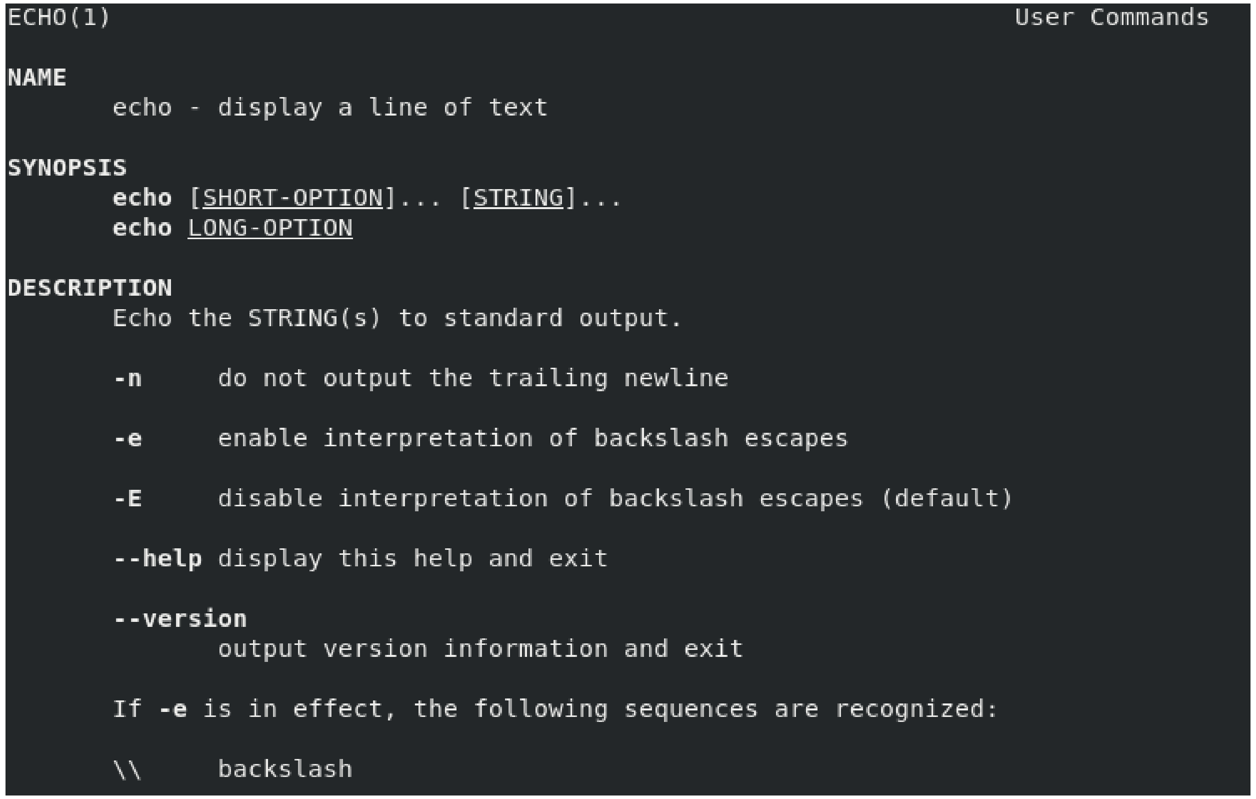
```
[intellipaat@localhost ~]$ pwd
/home/intellipaat
[intellipaat@localhost ~]$ who
intellipaat tty2         2020-01-21 05:15 (tty2)
[intellipaat@localhost ~]$ date
Tue Jan 21 07:23:05 EST 2020
[intellipaat@localhost ~]$ alias listtxt='ls *.txt'
[intellipaat@localhost ~]$ listtxt
1.txt  output.txt  timestamp.txt
[intellipaat@localhost ~]$ w
 07:24:16 up  2:10,  1 user,  load average: 0.04, 0.05, 0.06
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
intellip tty2     tty2             05:15    2:10m  1:32   0.00s /usr/libexec/gsd-disk-utility-notify
[intellipaat@localhost ~]$ mkdir hello
[intellipaat@localhost ~]$ ls
1.txt    Documents  ec2-ug.pdf  intel1.htl  Music       Pictures  sql1.sh  Templates      Videos
Desktop  Downloads  hello       intel.html  output.txt  Public    sql2.sh  timestamp.txt
```

Hands-on: Executing Linux Commands

Displaying – using echo

Displaying - using echo

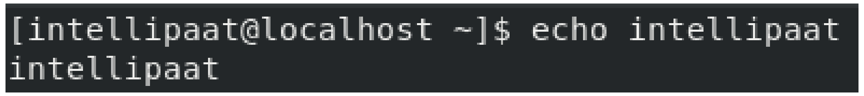
The command **echo** is used to display a line of text/string by passing it as an argument



```
ECHO(1)                                                    User Commands

NAME
       echo - display a line of text

SYNOPSIS
       echo [SHORT-OPTION]... [STRING]...
       echo LONG-OPTION

DESCRIPTION
       Echo the STRING(s) to standard output.

       -n     do not output the trailing newline

       -e     enable interpretation of backslash escapes

       -E     disable interpretation of backslash escapes (default)

       --help display this help and exit

       --version
              output version information and exit

       If -e is in effect, the following sequences are recognized:

       \\     backslash
```

Syntax

echo [options] [string]



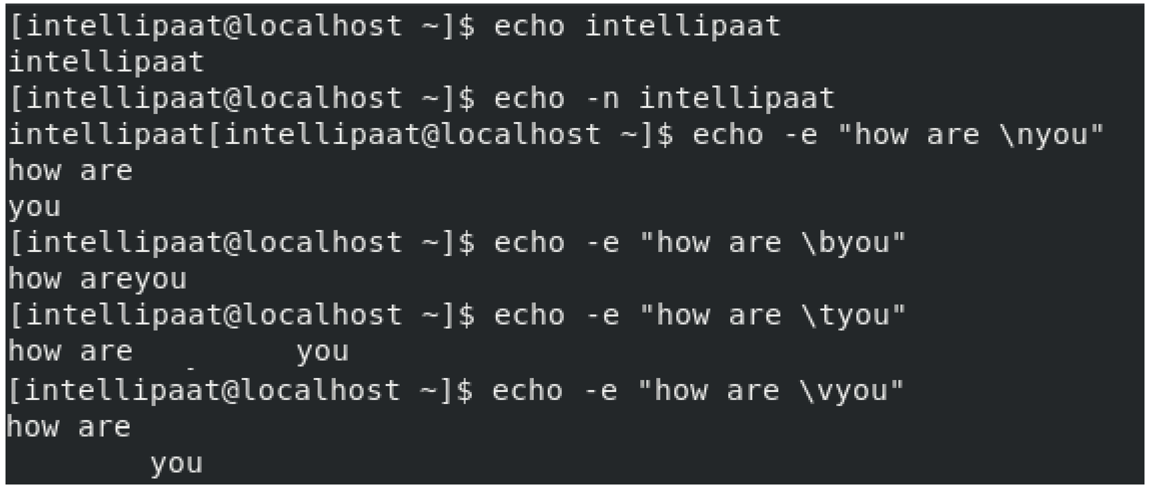
```
[intellipaat@localhost ~]$ echo intellipaat
intellipaat
```

Displaying - using echo

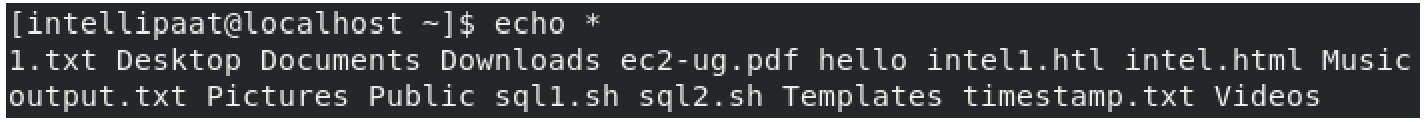
Options	Task
-n	Gives the output without a new line
-e	This will allow usage of backslash escapes
\b	Removes the space between text
\n	Prints the text in a new line
\t	Does a horizontal tab
\v	Does a vertical tab

Displaying - using echo

Few examples of echo

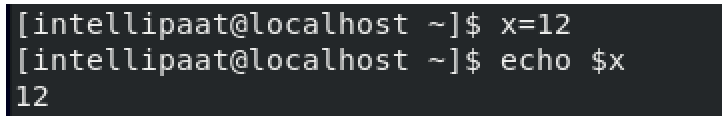


```
[intellipaat@localhost ~]$ echo intellipaat
intellipaat
[intellipaat@localhost ~]$ echo -n intellipaat
intellipaat[intellipaat@localhost ~]$ echo -e "how are \nyou"
how are
you
[intellipaat@localhost ~]$ echo -e "how are \byou"
how areyou
[intellipaat@localhost ~]$ echo -e "how are \tyou"
how are         you
[intellipaat@localhost ~]$ echo -e "how are \vyou"
how are
        you
```

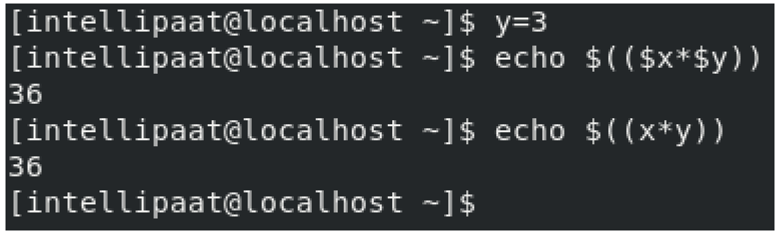


```
[intellipaat@localhost ~]$ echo *
1.txt Desktop Documents Downloads ec2-ug.pdf hello intel1.htl intel.html Music
output.txt Pictures Public sql1.sh sql2.sh Templates timestamp.txt Videos
```

Arithmetic operations



```
[intellipaat@localhost ~]$ x=12
[intellipaat@localhost ~]$ echo $x
12
```



```
[intellipaat@localhost ~]$ y=3
[intellipaat@localhost ~]$ echo $(($x*$y))
36
[intellipaat@localhost ~]$ echo $((x*y))
36
[intellipaat@localhost ~]$
```

Hands-on: Echo command

Set and Unset a variable

Set and Unset a variable

The **set** command is a built-in function in bash and few other cells which you can use to define the values of system variables. Set is not required to set a variable, there are various ways to do it

The **export** command is used to create Environment variables

The **unset** command is a built-in function in bash which you can use to remove a variable which is set

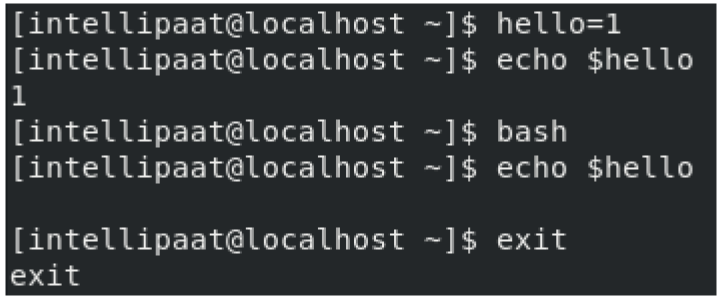
Set and Unset a variable

Options of set command

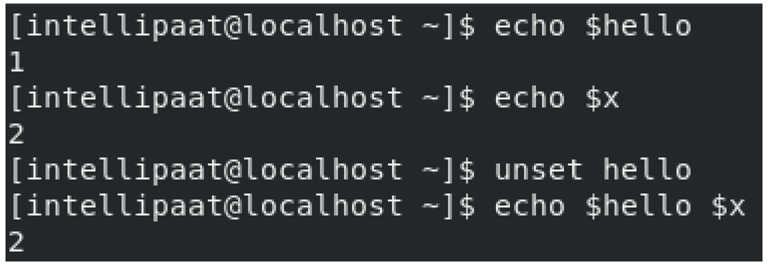
Options	Task
-b	Notify of job termination immediately
-e	Exit immediately if a command exits with a non-zero status
-m	Job control will be enabled
-o	option-name • allexport same as -a • braceexpand same as -B • errexit same as -e • errtrace same as -E • functrace same as –T

Set and Unset a variable

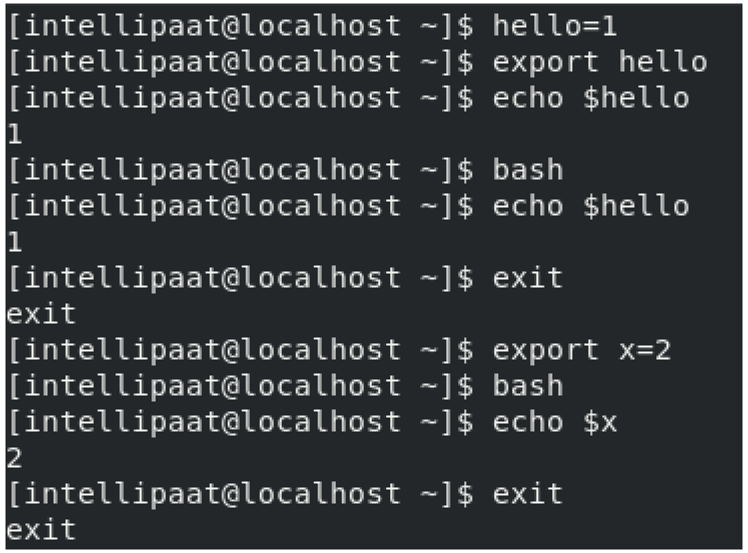
Few examples of set and unset



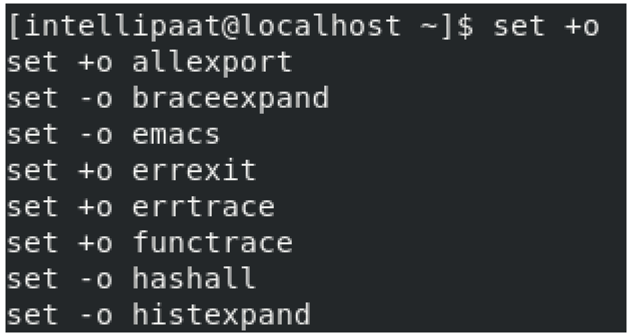
```
[intellipaat@localhost ~]$ hello=1
[intellipaat@localhost ~]$ echo $hello
1
[intellipaat@localhost ~]$ bash
[intellipaat@localhost ~]$ echo $hello

[intellipaat@localhost ~]$ exit
exit
```



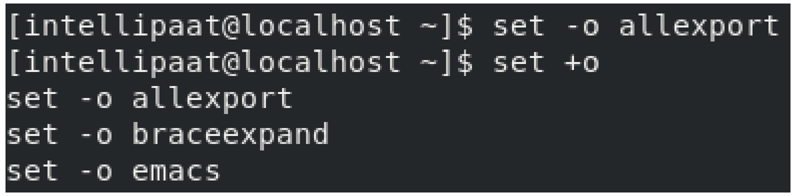
```
[intellipaat@localhost ~]$ echo $hello
1
[intellipaat@localhost ~]$ echo $x
2
[intellipaat@localhost ~]$ unset hello
[intellipaat@localhost ~]$ echo $hello $x
2
```



```
[intellipaat@localhost ~]$ hello=1
[intellipaat@localhost ~]$ export hello
[intellipaat@localhost ~]$ echo $hello
1
[intellipaat@localhost ~]$ bash
[intellipaat@localhost ~]$ echo $hello
1
[intellipaat@localhost ~]$ exit
exit
[intellipaat@localhost ~]$ export x=2
[intellipaat@localhost ~]$ bash
[intellipaat@localhost ~]$ echo $x
2
[intellipaat@localhost ~]$ exit
exit
```



```
[intellipaat@localhost ~]$ set +o
set +o allexport
set -o braceexpand
set -o emacs
set +o errexit
set +o errtrace
set +o functrace
set -o hashall
set -o histexpand
```



```
[intellipaat@localhost ~]$ set -o allexport
[intellipaat@localhost ~]$ set +o
set -o allexport
set -o braceexpand
set -o emacs
```

Using Expr

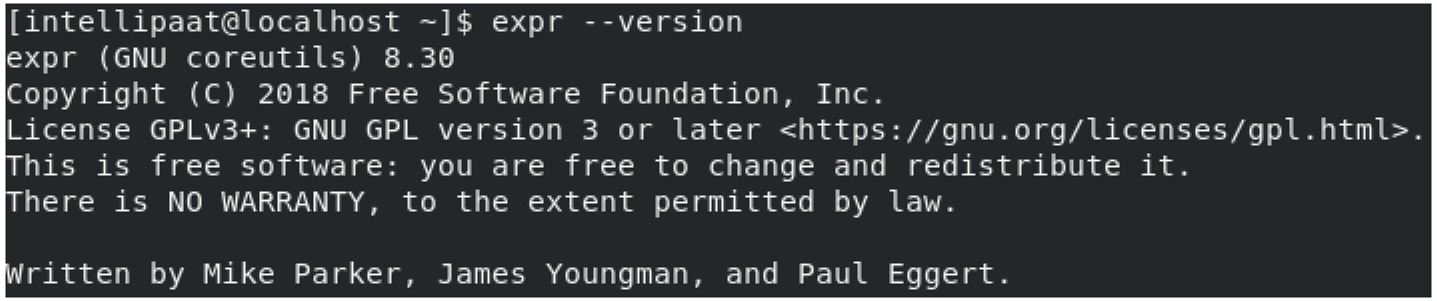
Using Expr

The command **expr** computes a given expression and displays the output

Syntax

$ expr expression

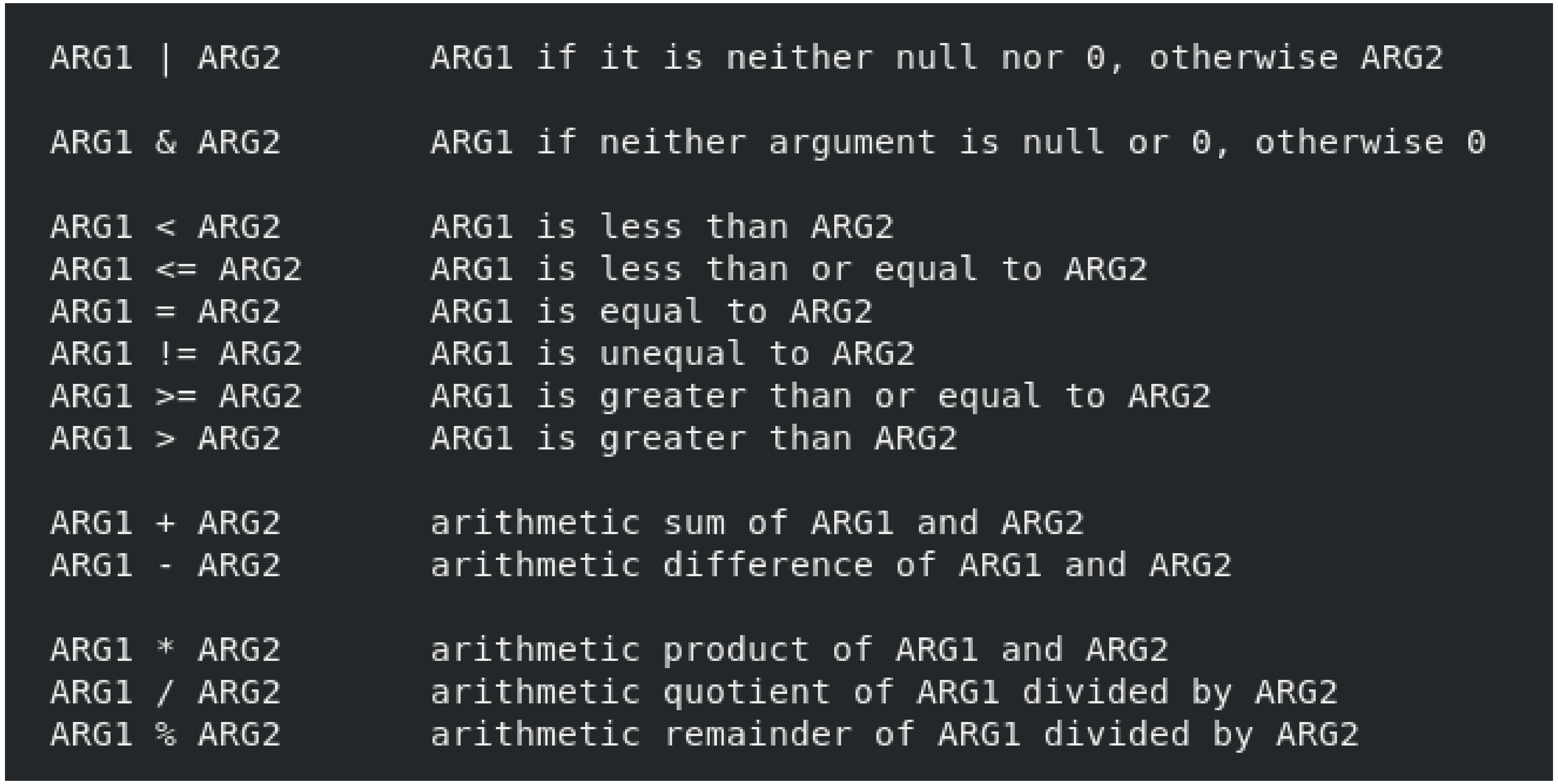
Checking whether expr is available



```
[intellipaat@localhost ~]$ expr --version
expr (GNU coreutils) 8.30
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Written by Mike Parker, James Youngman, and Paul Eggert.
```

Using Expr

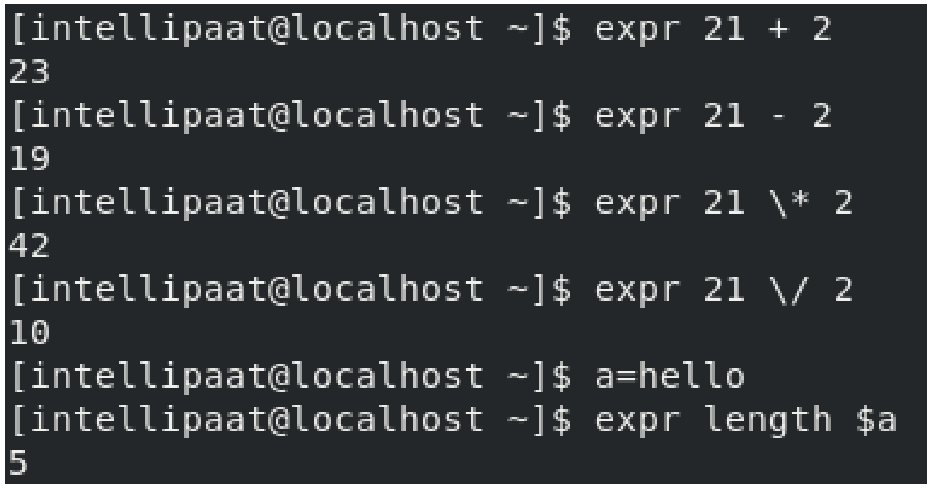
Arithmetic and logical operations



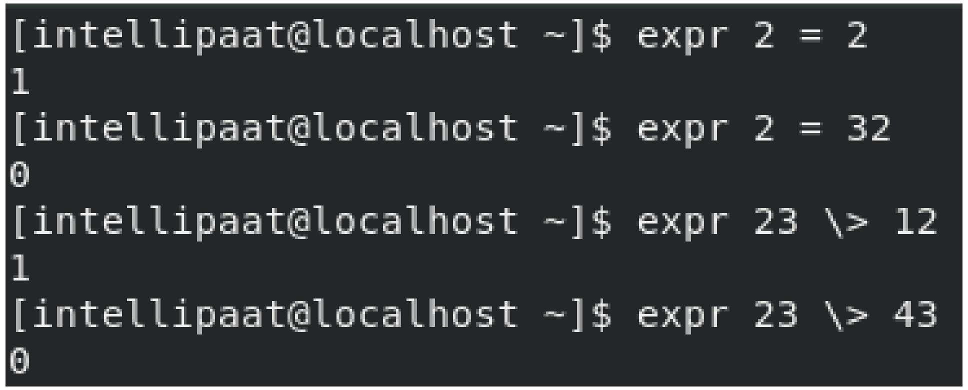
```
ARG1 | ARG2        ARG1 if it is neither null nor 0, otherwise ARG2

ARG1 & ARG2        ARG1 if neither argument is null or 0, otherwise 0

ARG1 < ARG2        ARG1 is less than ARG2
ARG1 <= ARG2       ARG1 is less than or equal to ARG2
ARG1 = ARG2        ARG1 is equal to ARG2
ARG1 != ARG2       ARG1 is unequal to ARG2
ARG1 >= ARG2       ARG1 is greater than or equal to ARG2
ARG1 > ARG2        ARG1 is greater than ARG2

ARG1 + ARG2        arithmetic sum of ARG1 and ARG2
ARG1 - ARG2        arithmetic difference of ARG1 and ARG2

ARG1 * ARG2        arithmetic product of ARG1 and ARG2
ARG1 / ARG2        arithmetic quotient of ARG1 divided by ARG2
ARG1 % ARG2        arithmetic remainder of ARG1 divided by ARG2
```

Using Expr

Few examples of expr



```
[intellipaat@localhost ~]$ expr 21 + 2
23
[intellipaat@localhost ~]$ expr 21 - 2
19
[intellipaat@localhost ~]$ expr 21 \* 2
42
[intellipaat@localhost ~]$ expr 21 \/ 2
10
[intellipaat@localhost ~]$ a=hello
[intellipaat@localhost ~]$ expr length $a
5
```



```
[intellipaat@localhost ~]$ expr 2 = 2
1
[intellipaat@localhost ~]$ expr 2 = 32
0
[intellipaat@localhost ~]$ expr 23 \> 12
1
[intellipaat@localhost ~]$ expr 23 \> 43
0
```

Header file of shell script - using Shebang (#!)

Header file of shell script - using Shebang (#!)

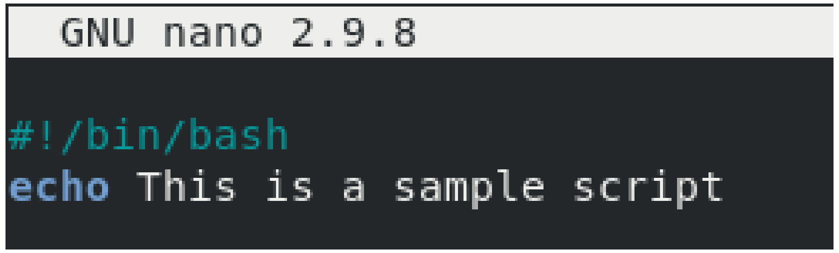
#! – This represents which interpreter a script should be interpreted with

#!/bin/bash – This is a header command which represents it is a bash/shell script

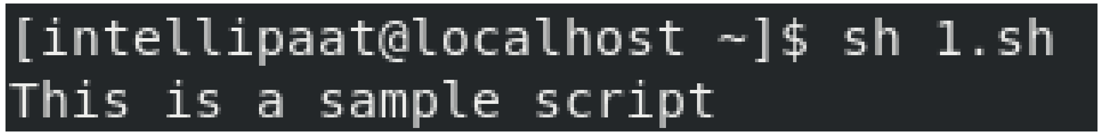
#!/bin/bash is this is not provided it often considers #!/bin/sh which would be same in most cases. When you put #!/bin/bash in your script, even if you run the script in a different shell, the kernel will know which shell to interpret it with

Header file of shell script - using Shebang (#!)

A sample script



```
  GNU nano 2.9.8

#!/bin/bash
echo This is a sample script
```



```
[intellipaat@localhost ~]$ sh 1.sh
This is a sample script
```

Quiz

Quiz

1. Is Linux the world's largest open source project?

A. True

B. False

Quiz

1. Is Linux the world's largest open source project?

[✓ A. True]

B. False

Quiz

2. Which command is used to check which users are logged in to the system?

A. whoami

B. who

C. w

D. why

Quiz

2. Which command is used to check which users are logged in to the system?

A. whoami

[✓ B. who]

C. w

D. why

Quiz

3. Which is the Linux command when provided with the command name as the argument, gives a document explaining the use of that command?

A. what

B. info

C. man

D. dog

Quiz

3. Which is the Linux command when provided with the command name as the argument, gives a document explaining the use of that command?

A. what

B. info

[✓ C. man]

D. dog

Quiz

4. What is the shebang symbol?

A. ##

B. !#

C. #!

D. #$

Quiz

4. What is the shebang symbol?

A. ##

B. !#

[✓ C. #!]

D. #$

Quiz

5. What is the default shell of CentOS 8?

A. zsh

B. csh

C. ksh

D. bash

Quiz

5. What is the default shell of CentOS 8?

A. zsh

B. csh

C. ksh

[✓ D. bash]

India: +91-7847955955

US: 1-800-216-8930 (TOLL FREE)

support@intellipaat.com

24/7 Chat with Our Course Advisor