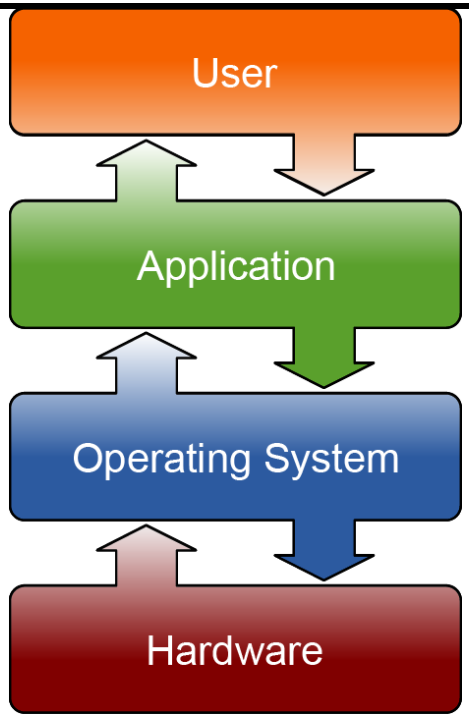
The UNIX Operating System

HORT 59000

Lecture 2

Instructor: Kranthi Varala

Operating Systems



- Operating systems allow the separation of hardware management from applications/programs.
- This allows the applications to work across different hardware platforms, although the applications are still specific to the OS.

Operating Systems

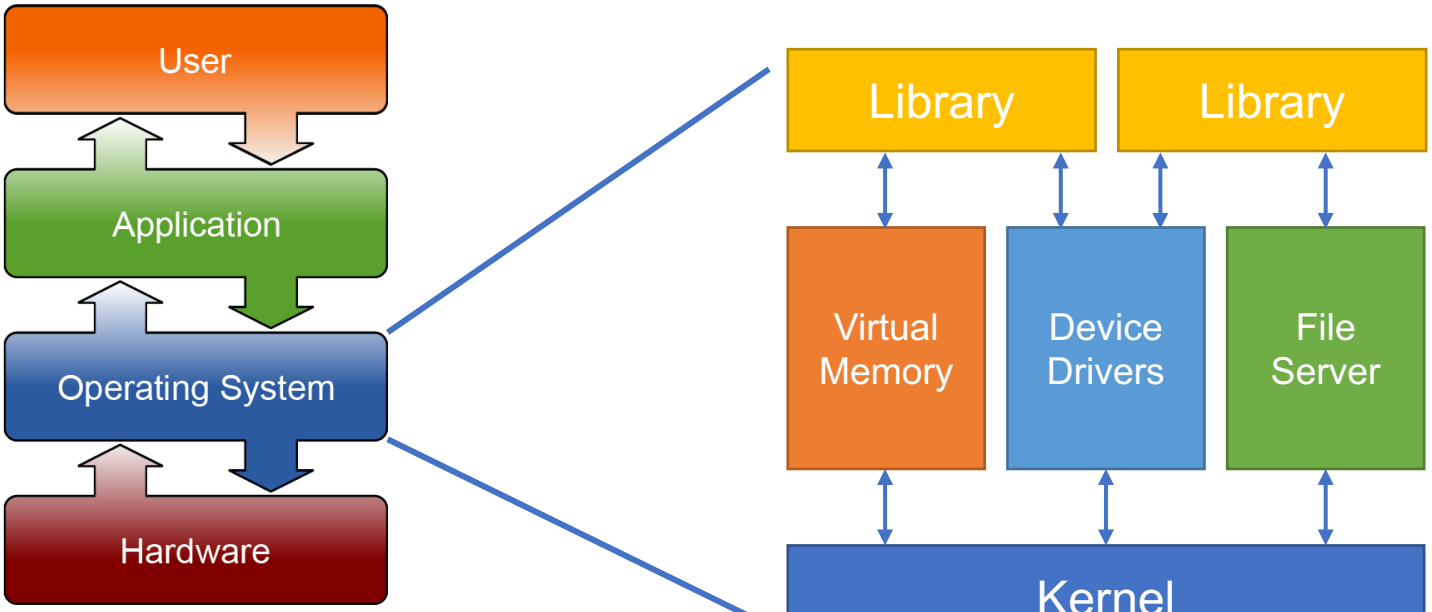
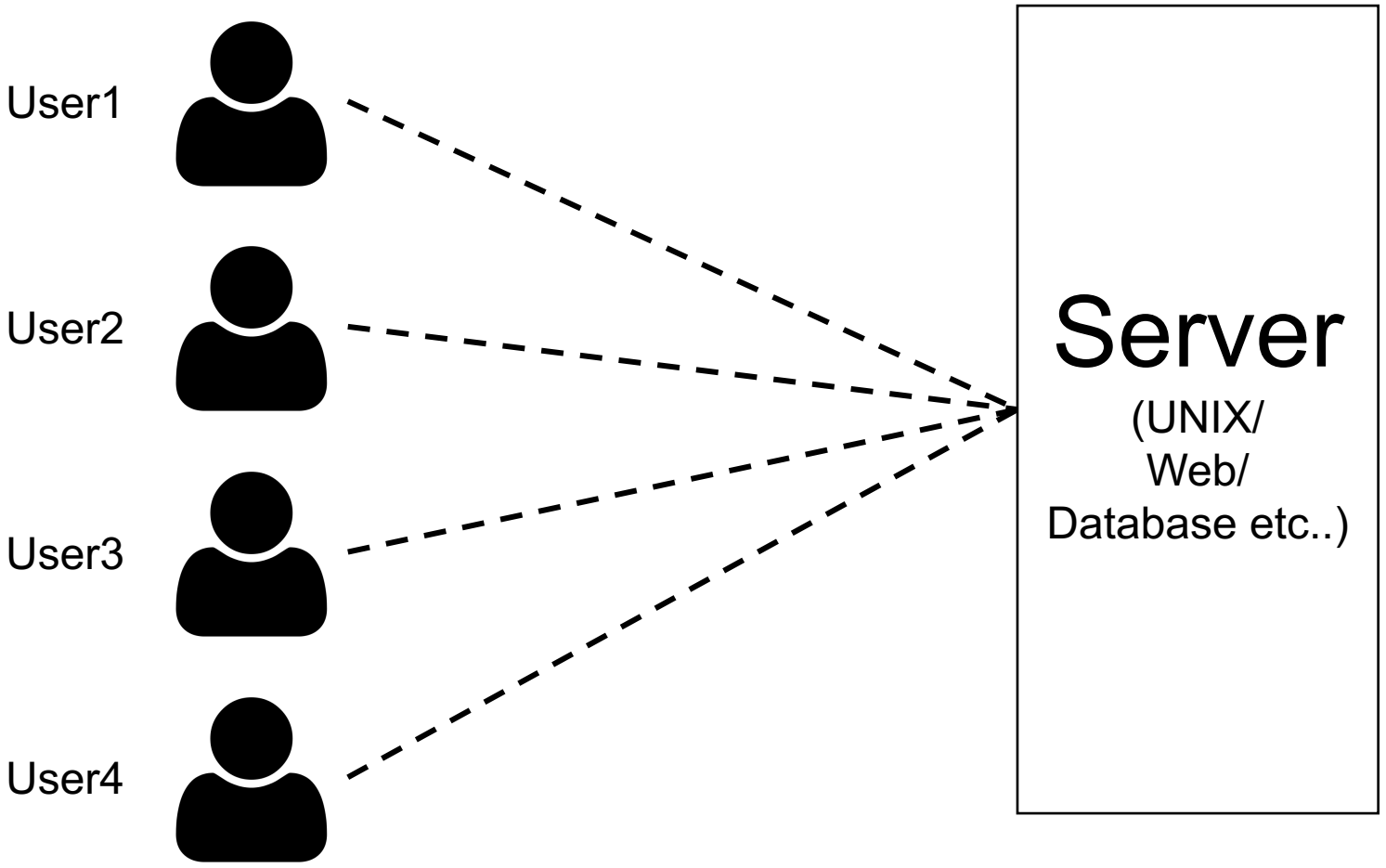


Image By Golftheman - https://commons.wikimedia.org/w/index.php?curid=4558519

- The Kernel is the core function of the OS and handles basic-level communication between the various processes and the hardware.
- Libraries provide applications with standardized access to kernel functions.

Client/Server architecture



Terminology

- Terminal: Device or Program used to establish a connection to the UNIX server
- Shell: Program that runs on the server and interprets the commands from the terminal.
- Command line: The text-interface you use to interact with the shell.

UNIX operating system

- First developed in 1970s, it is a multitasking OS that supports simultaneous use by multiple users.
- Strengths
  - Command-line based.
  - Supports thousands of small programs running simultaneously.
  - Easy to create pipelines from individual programs.
  - Multi-user support and partitioning is baked in.
- Challenges
  - Command-line based.
  - Finding help and documentation can be onerous.
  - Many different variants.

UNIX and its derivatives

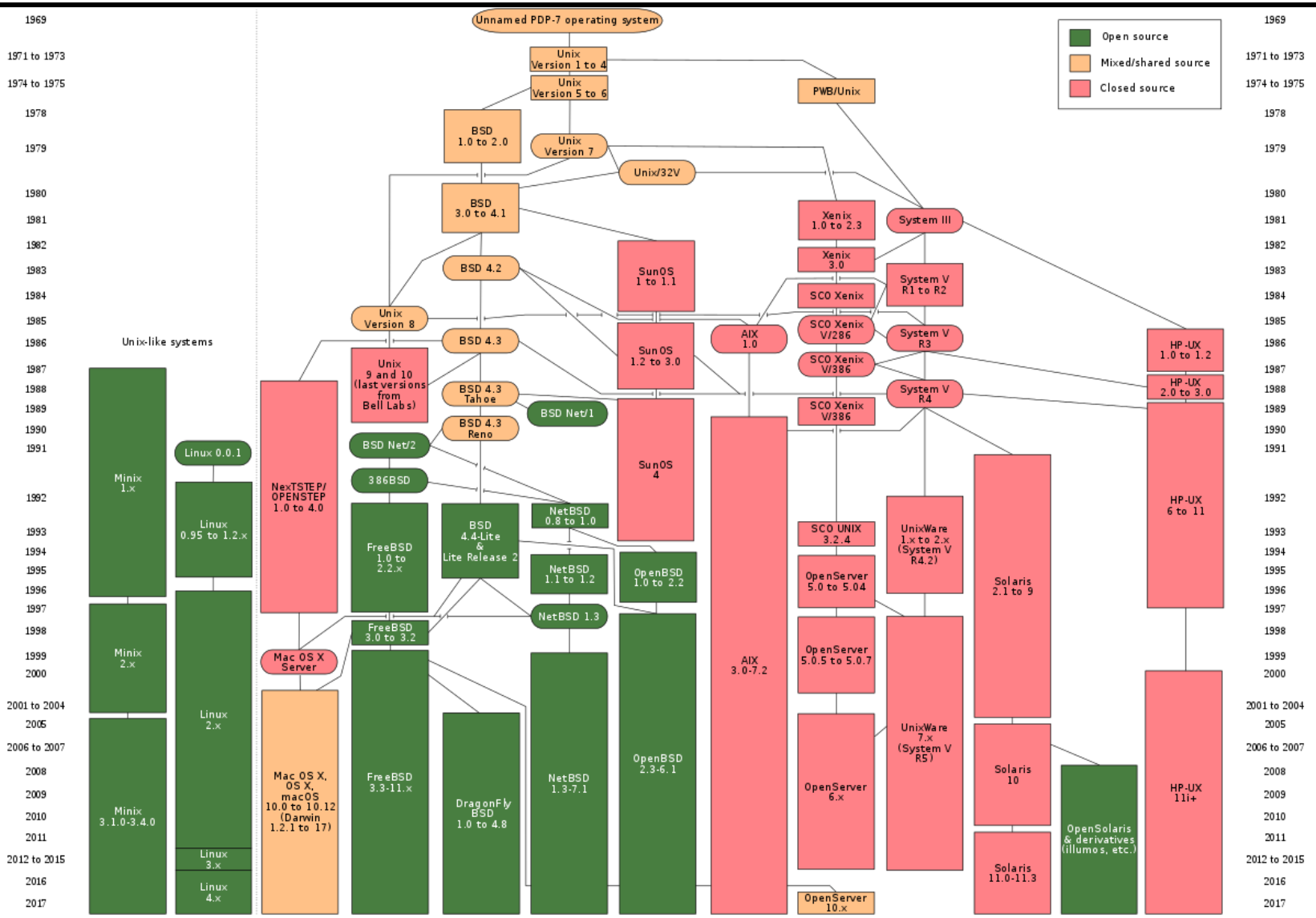


Image by Eraserhead1, Infinity0, Sav_vas - Levenez Unix History Diagram, Information on the history of IBM's AIX on ibm.com, https://commons.wikimedia.org/w/index.php?curid=1801948

Application Programming Interface (API)

- Biggest strength is the ability to connect different programs together.
- Programs/Applications need to be able to communicate.
- A pre-defined set of methods to communicate with an application is called it's API.
- Each program comes with its own API.

Programs vs. libraries

- Programs/Applications: Perform a defined task that accepts one or more inputs and produces an output.
  - Example: ls – lists contents of current location.

- Libraries: Collection of related functions that may be used by different programs.
  - Example: GNU Scientific library – provides a set of complex math functions

- Users typically interact with programs, while programmers use libraries within their code.

- Both Programs and Libraries have APIs.

POSIX standards

- Standards defined by IEEE computer society to maintain portability between different UNIX OS's.
- Originally defined standard API for core processes eg., kernel level access.
- Later expanded to include programs and utilities used directly by the user.
- Net result: Common UNIX commands you learn will be usable across UNIX/ Linux/MacOS etc.
- UNIX and MacOS (version 10.5 and above) are POSIX-certified

UNIX and its derivatives

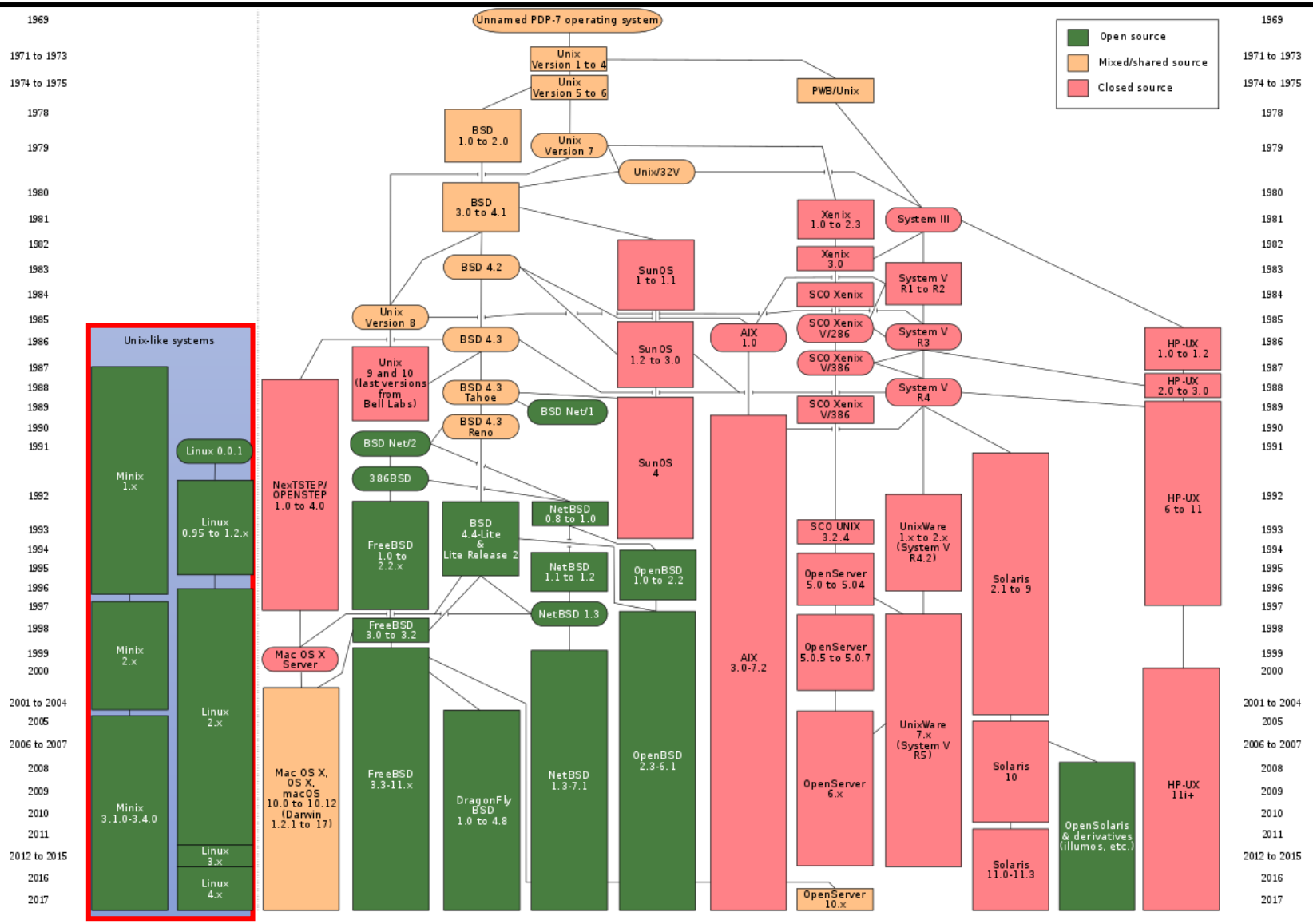


Image by Eraserhead1, Infinity0, Sav_vas - Levenez Unix History Diagram, Information on the history of IBM's AIX on ibm.com, https://commons.wikimedia.org/w/index.php?curid=1801948

GNU’s not Unix

- GNU started as a movement to create an open-source AND free OS and set of utilities.
- Open-source: The source code of the software is visible to everyone.
- Free: No limitation on how to use/distribute the software.
- By default, all software is copy-righted with the rights belonging to the programmer/organization.
- Software can be open-source and/or free.
- When you write your own pipelines, be aware of the difference in these two concepts.

GNU/Linux OS

- Most Unix-like operating systems are a variant of this scheme.
- Linux is typically the kernel of this OS.
- The rest of the utilities/applications were derived from the GNU project.
- Vast majority of commands you will type fall into the GNU portion of the OS.
- This user-interfacing part of the OS is often called User space.
- GNU/Linux is POSIX-compliant i.e., it mostly follows POSIX with a few exceptions.

User vs. Kernel space

- User space: Set of applications/utilities that interact with the user. Also includes, the portions of the file system where these files reside AND the portion of memory (RAM) where the programs are loaded and run.

- Kernel space: Set of applications that form middle layer between hardware and user applications. These program operate in a separate, protected portion of the RAM.

UNIX file system

- All UNIX files, including system and user files reside in a hierarchical directory structure.
- The file system maintains the record of where each file resides on the hardware.
- The lowest level or base of this structure is called the ‘root’ directory represented as /
- Every user has a defined home directory
  - My home is: /home/kvarala

Files and Directories

- Files are the basic unit of storage. Eg., This presentation file.
- Directories are containers that hold sets of related files. Eg., Set of presentations for this course.
- Each file name within a directory has to be unique.
- UNIX is case-sensitive i.e., the file example.txt is different from the file Example.txt
- Directory names are also case-sensitive.

Files contd..

- File extensions eg., .txt or .jpg or .doc etc. have no relevance in UNIX.
- It is good practice for users to use a file extension that describes the file type.
- Use long descriptive names for your files. File name length is allowed up to 255 characters.
- File size limits are defined by the file system used by the OS.
- Current file systems support file sizes larger than the capacity of current hardware ($2^{\wedge 63}$ bytes).

Typical structure of UNIX

/bin ==> Programs/Utilities. Typically OS files.

/etc ==> Administrative files. Usually, OS related.

/home ==> Home directories of users.

/lib ==> Libraries. OS or installed software.

/mnt ==> Mounted devices. Eg., CD/DVD, USB drive.

/root ==> Home for root/administrative user.

/tmp ==> Temporary files. OS, software and user.

/usr ==> User-space programs/Utilities

/var ==> System generated temporary files.

File paths

- In a shell you are always in a particular location. Default location after login is your home
  - Eg: /home/kvarala
- Every file has a location on the server.
- Path defines the location of the file/directory.
- Path can be defined two ways:
  - Absolute: Path starts from Root. Eg., /scratch/scholar/k/kvarala/Week1/Lecture_1.pdf
  - Relative: Path starts from current location. Eg., ../Week2/Lecture_2.pdf
- Special characters:
  - . Means current directory
  - .. Means parent directory
  - ~ Means home directory

Commands

- Every command is a program.
- UNIX philosophy is write simple programs that do one job very well.
- Complex functionality can be built by combining simple programs.
- User can add commands by writing their own program
- Commands are of course Case-sensitive and the OS needs to know the path to the program.

Command line

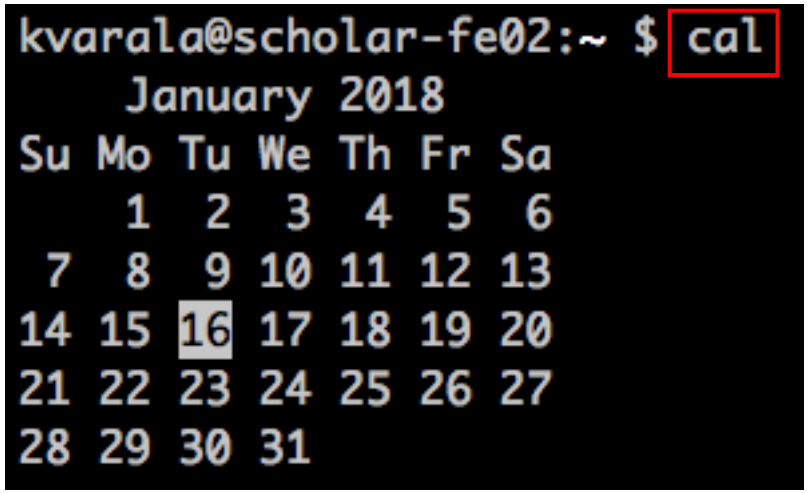
- Every word you type in the command line is interpreted by the shell as a command.
- If the shell cannot interpret the command it returns the error: "*command not found*".
- The shell looks for the program matching the typed command in the locations defined by PATH.
- User can add commands by adding programs to their PATH.

I/O streams

- Each command has 3 Input/Output streams:
  - STDIN : Standard Input is the default stream that inputs data into a command. Example: keyboard, file etc.
  - STDOUT : Standard Output is the default output stream of the command. Example: Terminal
  - STDERR: Standard Error is where the errors from the program are displayed: Example: Terminal

Anatomy of a command

- Let's explore commands with a easy command called cal:
- This command by default displays the calendar for this month with today highlighted.



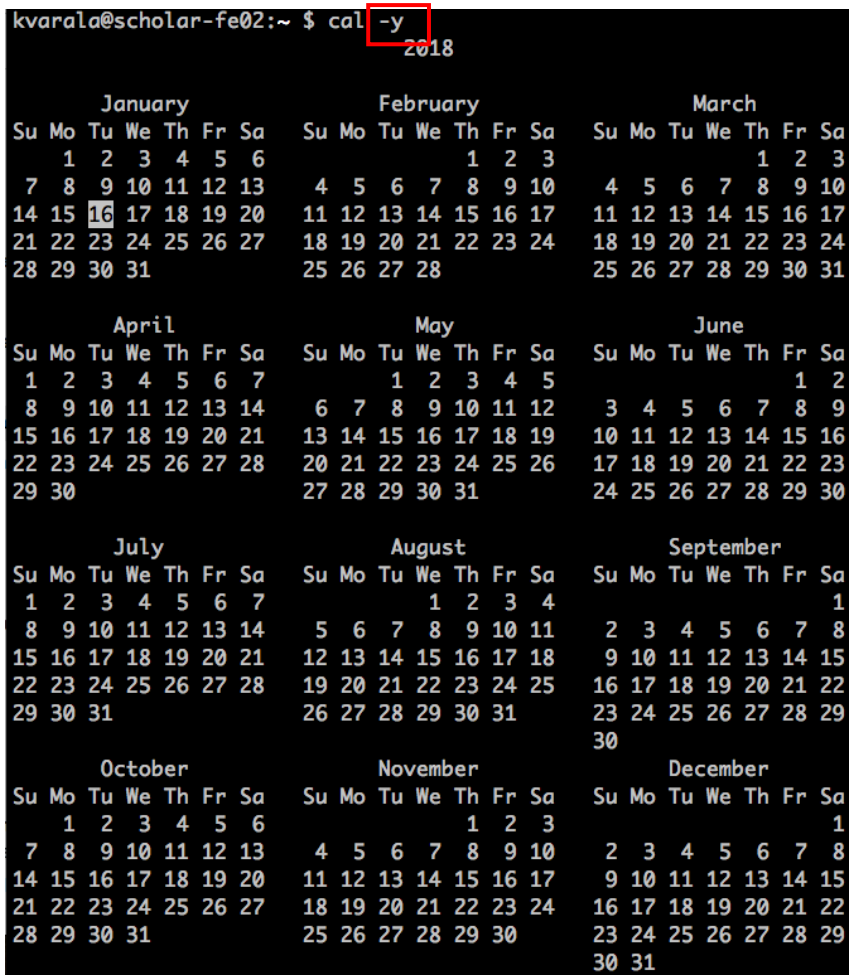
Anatomy of a command

- Commands can often take arguments, which change the default settings.
- The argument -3 changes display to 3 months with current month in the center.

```
kvarala@scholar-fe02:~ $ cal -3
    December 2017         January 2018          February 2018
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                1  2      1  2  3  4  5  6                1  2  3
 3  4  5  6  7  8  9   7  8  9 10 11 12 13   4  5  6  7  8  9 10
10 11 12 13 14 15 16  14 15 16 17 18 19 20  11 12 13 14 15 16 17
17 18 19 20 21 22 23  21 22 23 24 25 26 27  18 19 20 21 22 23 24
24 25 26 27 28 29 30  28 29 30 31           25 26 27 28
31
```

Anatomy of a command

- Argument -y displays the whole year.



Anatomy of a command

- Commands may take multiple arguments that change multiple defaults.
- First argument changes the month and second argument changes the year.

```
kvarala@scholar-fe02:~ $ cal 4 2018
     April 2018
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30
```

```
kvarala@scholar-fe02:~ $ cal 4 2019
     April 2019
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30
```

Anatomy of a command

- Arguments that start with - are some times called switches.
- Arguments that do not have a preceding - are called parameters.

```
kvarala@scholar-fe02:~ $ cal -3 4 2018
     March 2018            April 2018            May 2018
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
             1  2  3   1  2  3  4  5  6  7         1  2  3  4  5
 4  5  6  7  8  9 10   8  9 10 11 12 13 14   6  7  8  9 10 11 12
11 12 13 14 15 16 17  15 16 17 18 19 20 21  13 14 15 16 17 18 19
18 19 20 21 22 23 24  22 23 24 25 26 27 28  20 21 22 23 24 25 26
25 26 27 28 29 30 31  29 30                 27 28 29 30 31
```

- NOTE: This naming convention is not always followed.

Example UNIX commands

- ls stands for list. This command lists the contents of the current location.
- pwd stands for print working directory. This command tells you your current location in the shell.
- hostname gives you the name of the host that the shell resides on.
- who lists all the users currently logged into the server.

Let’s explore ls

- ls is a basic but powerful command.
- It has over 50 arguments to alter its behavior.
- The most commonly used one is the ‘long list format’ specified by the switch -l .
- Try typing in ls in your scratch folder:
- Now, try typing in ls -l

ls on my scratch

```
kvarala@scholar-fe02:/scratch/scholar/k/kvarala $ ls
rcac_cluster_reference.pdf  Week1  Week2
```

Contents of my scratch folder

```
kvarala@scholar-fe02:/scratch/scholar/k/kvarala $ ls -l
total 232
-rw-r--r-- 1 kvarala student 227905 May  5  2017 rcac_cluster_reference.pdf
drwxr-xr-x 2 kvarala student   4096 Jan 16 12:02 Week1
drwxr-xr-x 2 kvarala student   4096 Jan 16 12:02 Week2
```

Metadata about the contents

Contents of my scratch folder

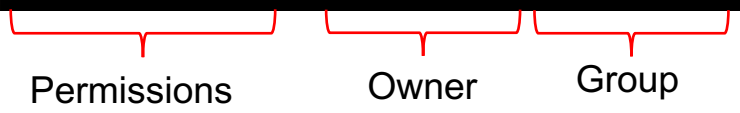
Metadata on files and directories

- Metadata is information about the file that are not part of the contents of the file.
- Three main parts to it:
  - Ownership and access permissions
  - Size
  - Timestamp

Ownership and Access

- Every file/directory has a defined owner, which is one user.

```
kvarala@scholar-fe02:/scratch/scholar/k/kvarala $ ls -l
total 232
-rw-r--r-- 1 kvarala student 227905 May  5  2017 rcac_cluster_reference.pdf
drwxr-xr-x 2 kvarala student   4096 Jan 16 12:02 Week1
drwxr-xr-x 2 kvarala student   4096 Jan 16 12:02 Week2
```



- Owner controls who can access the file/directory by setting the permissions.
- Each user is a part of one or more groups. Each file belongs to one of the groups that the user belongs to.

UNIX permissions

- Execute == 1
- Write == 2
- Read == 4

Common Permission settings	Indicator	Numeric code
Read-only	r--	4
Read & execute	r-x	5
Read & write	rw-	6
Read, write, execute	rwx	7

UNIX permissions

```
kvarala@scholar-fe02:/scratch/scholar/k/kvarala $ ls -l
total 232
-rw-r--r-- 1 kvarala student 227905 May  5  2017 rcac_cluster_reference.pdf
drwxr-xr-x 2 kvarala student   4096 Jan 16 12:02 Week1
drwxr-xr-x 2 kvarala student   4096 Jan 16 12:02 Week2
```

Permissions

- First character is - for a file and d for a directory.
- Characters 2-4 refer to permissions the owner sets for himself.
- Characters 5-7 are permissions for the group listed.
- Characters 8-10 are permissions for the world (i.e., every other user)

Common Permission settings	Indicator	Numeric code
Read-only	r--	4
Read & execute	r-x	5
Read & write	rw-	6
Read, write, execute	rwx	7

Working with directories

- pwd -> lists the present working directory
- mkdir -> makes a new directory
- cd -> change directory
- rmdir -> remove directory
- Try using cd with path:
  - cd /scratch/scholar/k/kvarala
  - cd ./Week1
  - cd ../Week2

File commands

- mv is the move command that moves a file. This command is also used for renaming files.
- rm is the remove command and will remove the file or empty directory listed as argument.
- cat is the catenate command that joins the contents of all files given as arguments.

Creating pipelines from commands

- The STDIN and STDOUT of each command can be redirected to combine programs together.
- For example, the STDOUT of one program can be sent to the STDIN of another program.
- We will go over examples in tomorrow's lab section.

Summary

- UNIX is a text-based, multiuser OS, that supports simultaneous execution of thousands of commands.
- UNIX is case-sensitive for file names and command names.
- Each command is a program stored as a file in specified location.
- Commands can be combined by redirecting I/O streams.
- Each file has a path that uniquely identifies its location.
- Access to files and directories is controlled via permissions set by the owner of the file.