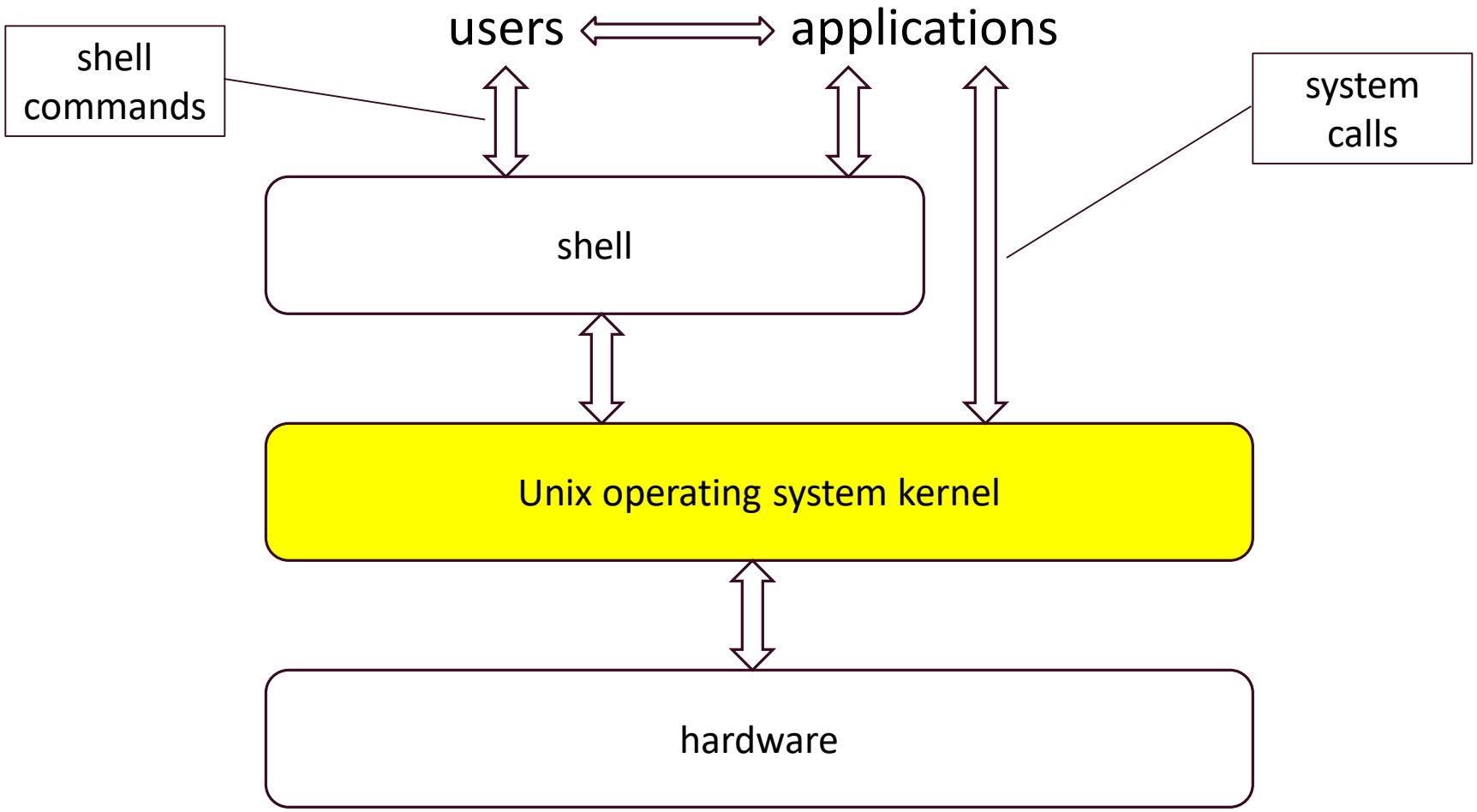
01_01_Introduction to Unix

What is Unix?

- **Unix is an operating system**
  - **sits between the hardware and the user/applications**
  - **provides high-level abstractions (e.g., files) and services (e.g., multiprogramming)**

- **Linux:**
  - **a "Unix-like" operating system: user-level interface very similar to Unix**
  - **code base is different from original Unix code**

Layers of a Unix system



The file system

- **Unix provides files and directories.**
- **A directory is like a folder: it contains pieces of paper, or files.**
- **A large folder can even hold other folders-*directories can be inside directories*.**
- **In unix, the collection of directories and files is called the file system. Initially, the file system consists of one directory, called the "root" directory**
- **Inside "root" directory, there are more directories, and inside those directories are files and yet more directories.**

The file system

- Each file and each directory has a name.
- A short name for a file could be joe,
- while it's "full name" would be /home/larry/joe. The full name is usually called the path.
- The path can be divide into a sequence of directories.
- For example, here is how /home/larry/joe is read:

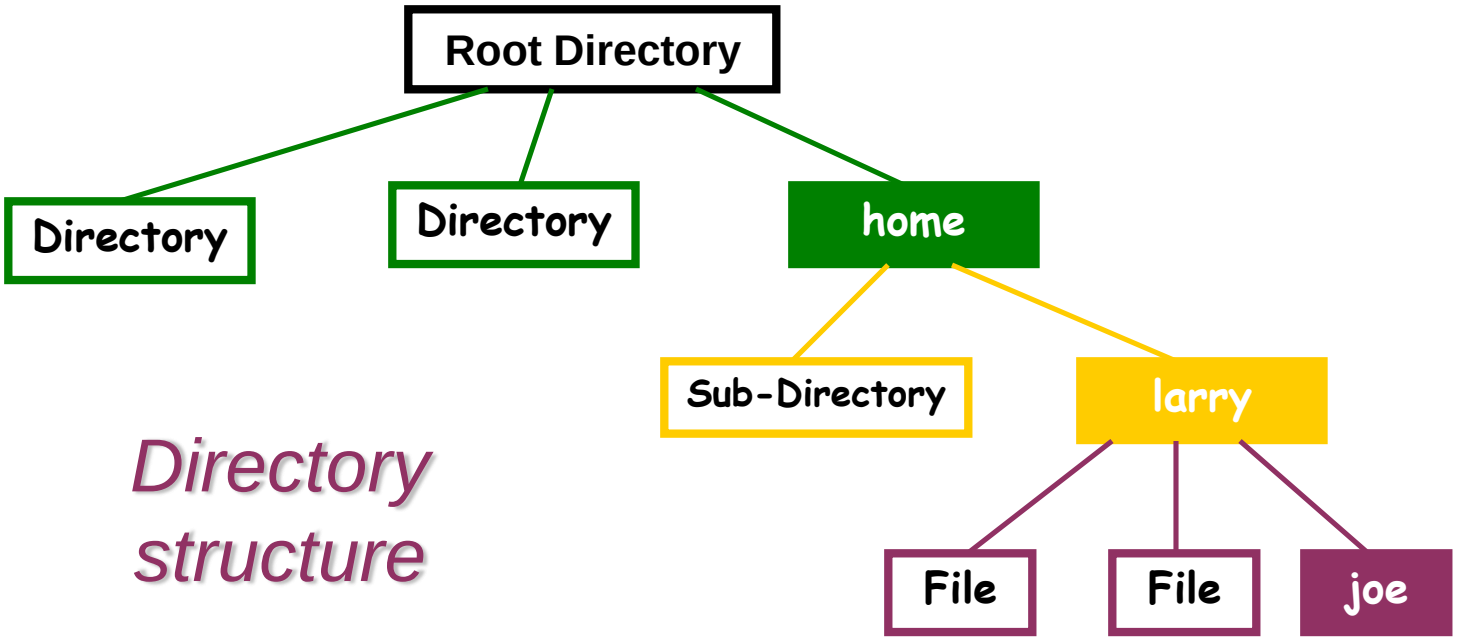
/home/larry/joe

The initial slash indicates the root directory. This signifies the directory called home. It is inside the root

The second slash corresponds to the directory larry, which is inside home.

joe is inside larry.

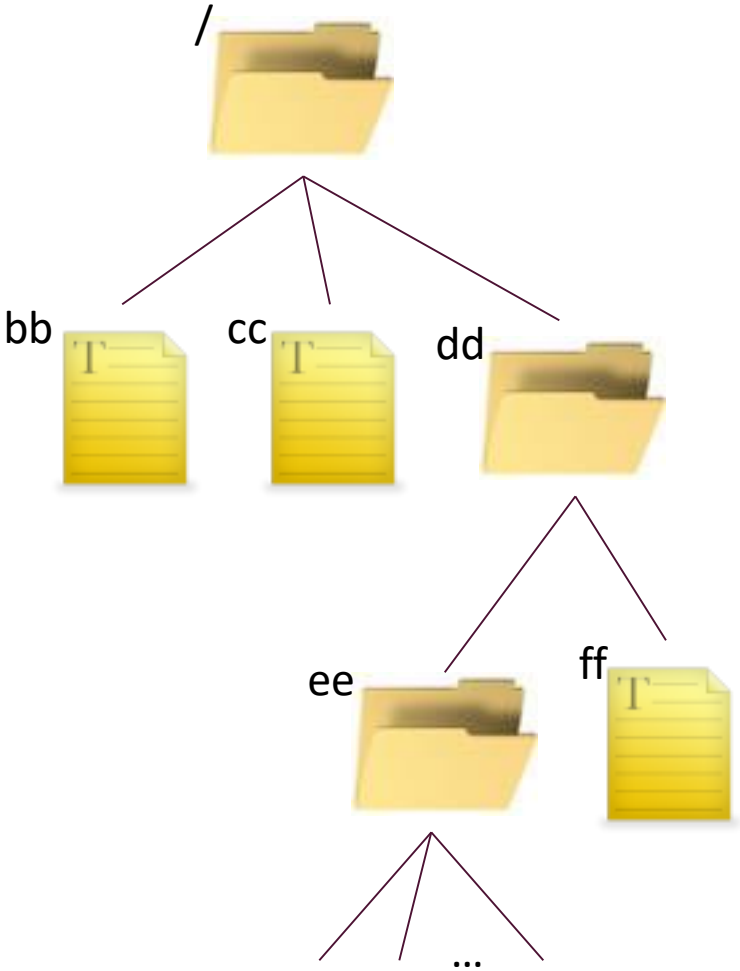
- A path could refer to either a directory or a filename, so joe could be either.
- All the items before the short name must be directories.



Directory structure

The file system

- A *file* is basically a sequence of bytes
- Collections of files are grouped into *directories* (≈ folders)
- A directory is itself a file
  - ➡ **file system has a hierarchical structure (i.e., like a tree)**
    - the root is referred to as "/"

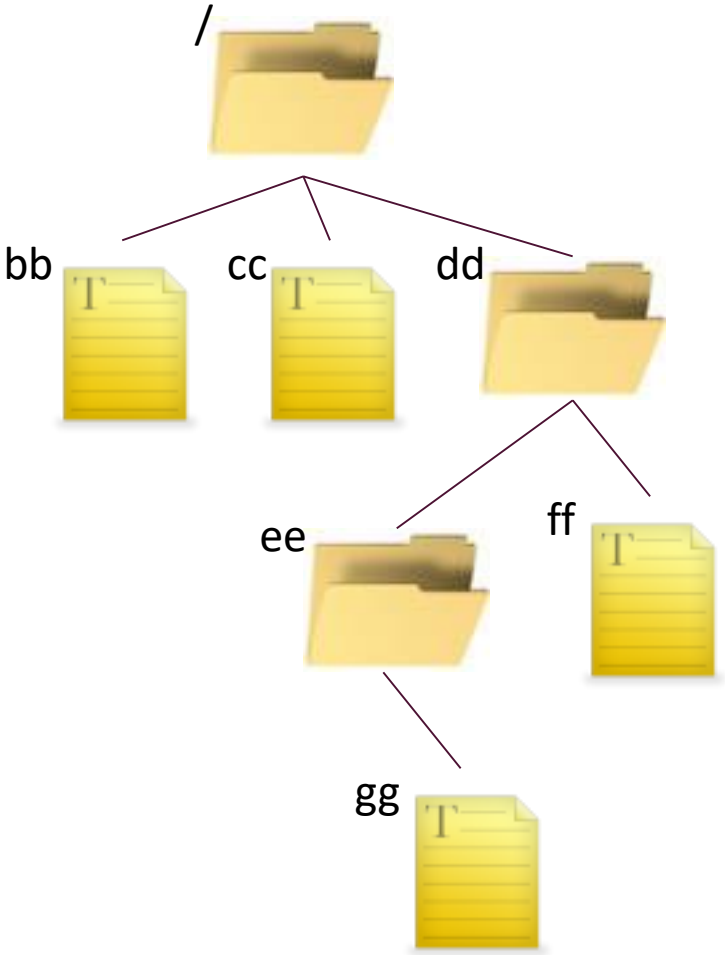


"Everything is a file"

- **In Unix, everything looks like a file:**
  - **documents stored on disk**
  - **directories**
  - **inter-process communication**
  - **network connections**
  - **devices (printers, graphics cards, interactive terminals, …)**
- **They are accessed in a uniform way:**
  - **consistent API (e.g., read, write, open, close, …)**
  - **consistent naming scheme (e.g., /home/debray, /dev/cdrom)**

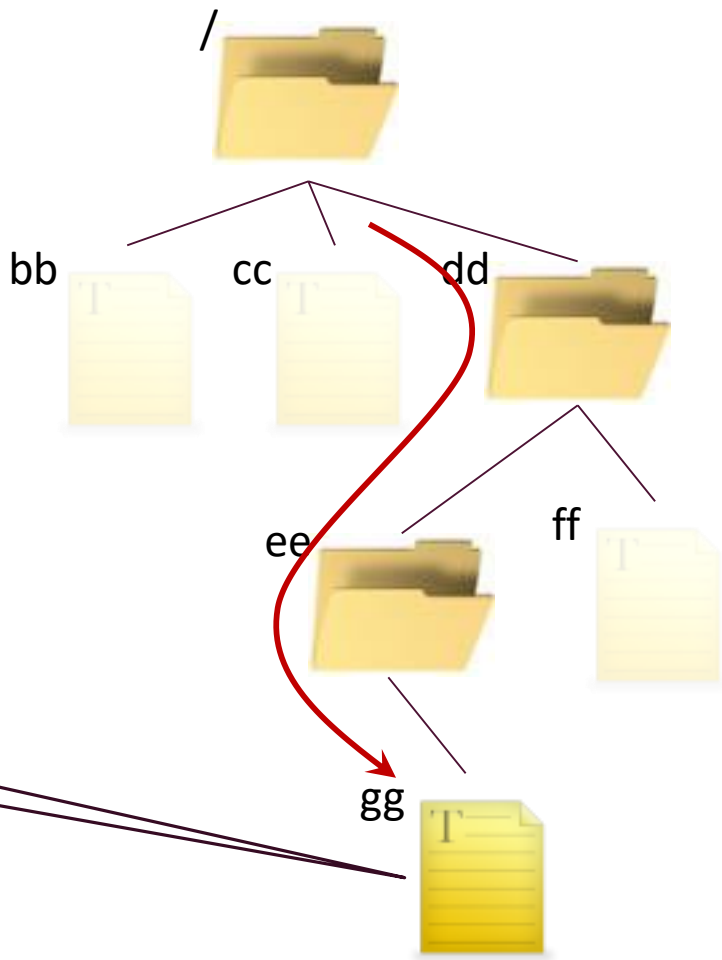
Referring to files: Absolute Paths

- An ***absolute path*** specifies how to get to a file starting at the file system root
  - list the directories on the path from the root ("/"), separated by "/"



Referring to files: Absolute Paths

- **An *absolute path* specifies how to get to a file starting at the file system root**
  - **list the directories on the path from the root ("/"), separated by "/"**

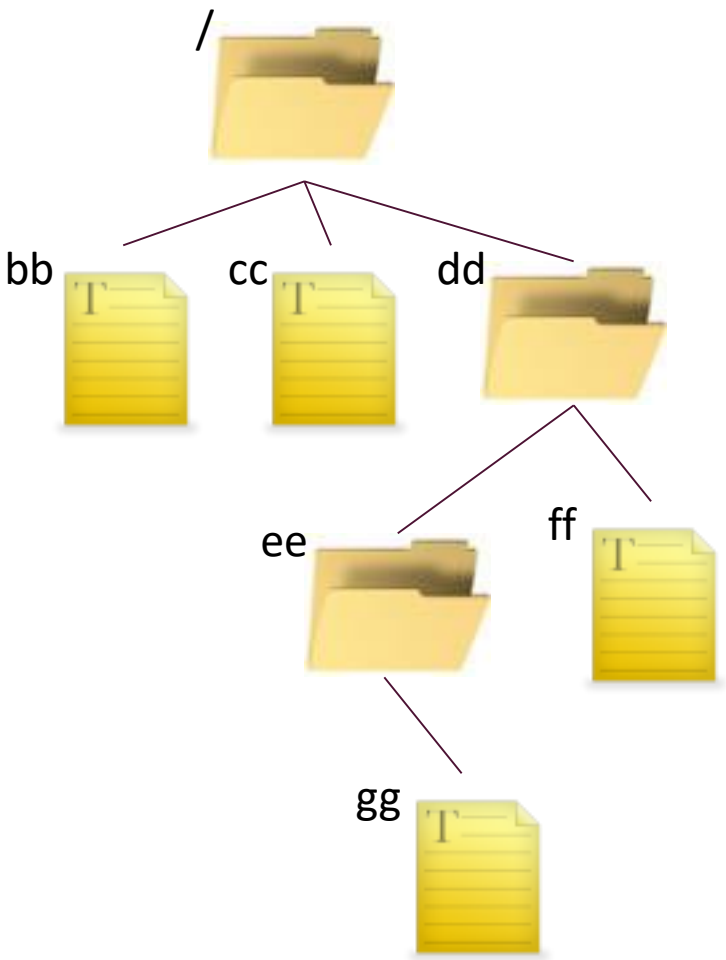


absolute path: **/dd/ee/gg**

Referring to Files: Relative Paths

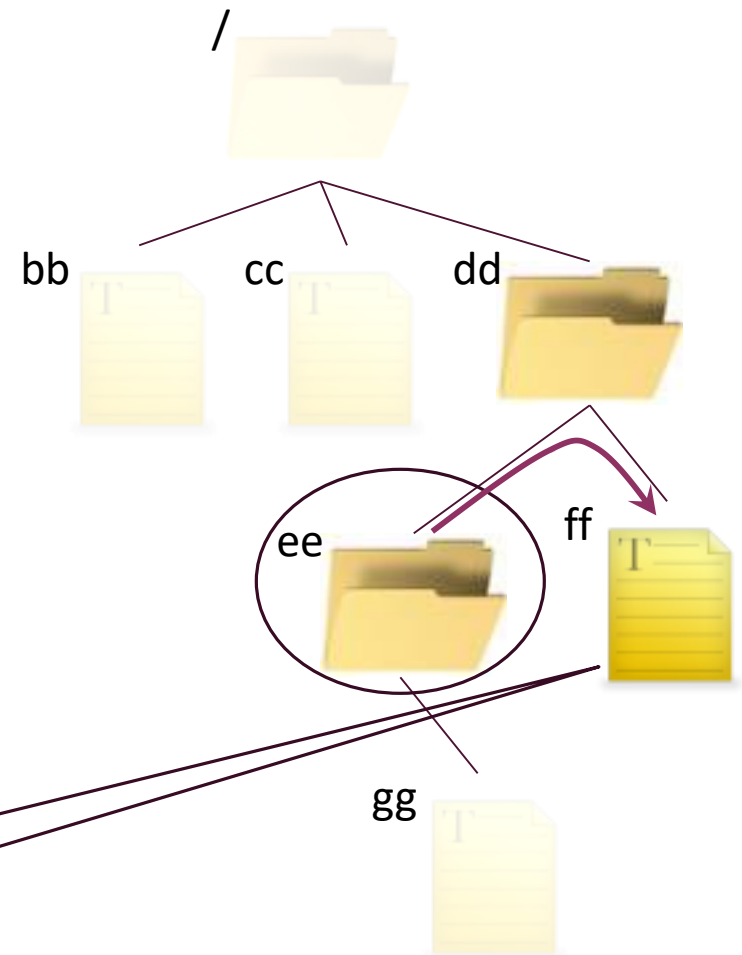
- **Typically we have a notion of a "*current directory*"**

- **A *relative path* specifies how to get to a file starting from the current directory**
  - **'..' means "move up one level"**
  - **'.' means current directory**
  - **list the directories on the path separated by "/"**



Referring to files: Relative Paths

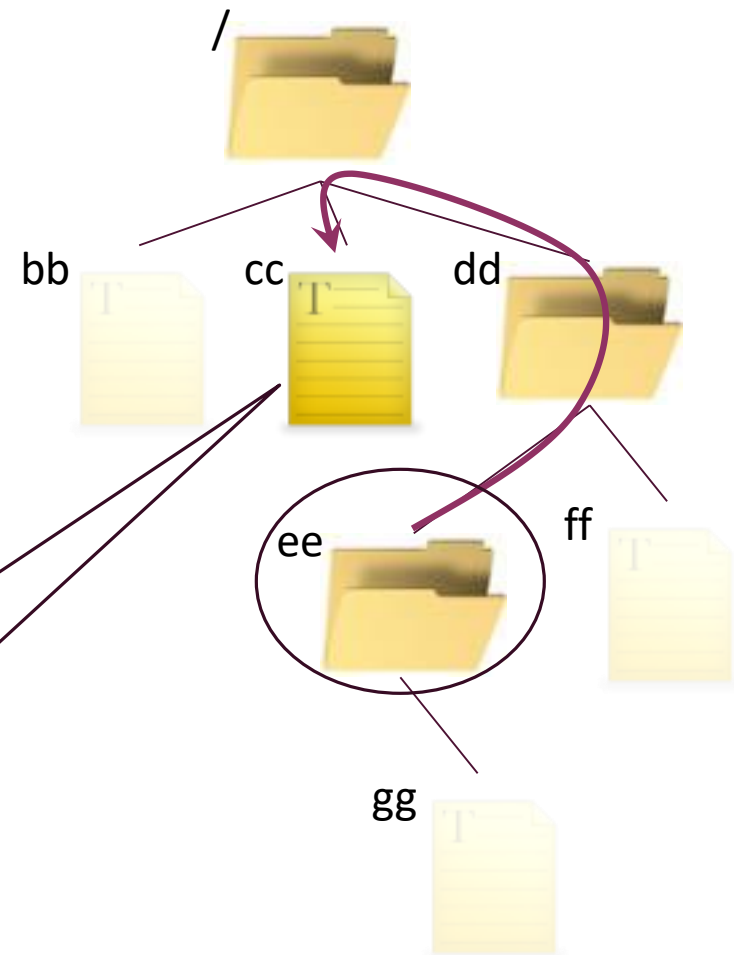
- **Typically we have a notion of a "*current directory*"**
- **A *relative path* specifies how to get to a file starting from the current directory**
  - **'..' means "move up one level"**
  - **'.' means current directory**
  - **list the directories on the path separated by "/"**



Example:
ff relative to **ee** is: **../ff**

Referring to files: Relative Paths

- **Typically we have a notion of a "*current directory*"**
- **A *relative path* specifies how to get to a file starting from the current directory**
  - **'..' means "move up one level"**
  - **'.' means current directory**
  - **list the directories on the path separated by "/"**



Example:
cc relative to **ee** is: **../../cc**

Home directories

- **Each user has a "home directory"**
  - **specified when the account is created**
  - **given in the file /etc/passwd**
- **When you log in, your current directory is your home directory**
  - **can then start a *shell* and issue commands**

A shell is just an interpreter for Unix commands

- **Notational shorthand:**
  - **one's own home directory: ~**
  - **some other user joe's home directory: ~joe**

UNIX Commands

- **Unix is also case-sensitive. This means that cat and Cat are different commands.**

- **The prompt is displayed by a special program called the shell.**

- **Shells accept commands, and run those commands.**

- **They can also be programmed in their own language. These programs are called "shell scripts".**

- There are two major types of shells in unix:
  - Bourne shells
  - C shells
- *Steven Bourne* wrote the original Unix shell sh and most shells since then end in the letters sh to indicate they are extensions on the original idea.
- Linux comes with a Bourne shell called bash written by the Free Software Foundation.
- bash stands for Bourne Again Shell and is the default shell to use running Linux.

Unix Commands

- **When you first login, the prompt is displayed by bash, and you are running your first unix program, the bash shell.**
- **As long as you are logged in, the *bash shell* will constantly be running.**

Unix Commands

Prompt

Command

```
bagriy@sariyer:~> cat
Help! I'm stuck in a Linux program!
Help! I'm stuck in a Linux program!
```

İf you type this row and then press enter

The text indicates what we typed to cat

- **To end many unix command, type end-of-file command (EOF) [hold down the key labeled "Ctrl" and press "d" (Ctrl+d)]**

Contents

Shell Intro

- **A system program that allows a user to execute:**
  - **shell functions (internal commands)**
  - **other programs (external commands)**
  - **shell scripts**
- **Linux/UNIX has a bunch of them, the most common are**
  - **`tcsh`, an expanded version of `csh` (Bill Joy, Berkley, Sun)**
  - **`bash`, one of the most popular and rich in functionality shells, an expansion of `sh` (AT&T Bell Labs)**
  - **`ksh`, Korn Shell**
  - **`zhs`**
  - **`. . .`**

Command Format

- **Format: command name and 0 or more arguments:**
  `% commandname [arg1] ... [argN]`

- **By % sign I mean prompt here and hereafter.**

- **Arguments can be**
  - **options (switches to the command to indicate a mode of operation) ; usually prefixed with a hyphen (-) or two (--) in GNU style**
  - **non-options, or operands, basically the data to work with (actual data, or a file name)**

Shell I/O

- **Shell is a “power-user” interface, so the user interacts with the shell by typing in the commands.**

- **The shell interprets the commands, that may produce some results, they go back to the user and the control is given back to the user when a command completes (in general).**

Shell I/O

- **In the case of external commands, shell executes actual programs that may call functions of the OS kernel.**

- **These system commands are often wrapped around a so-called system calls, to ask the kernel to perform an operation (usually privileged) on your behalf.**

Command I/O

- **Input to shell:**
  - **Command name and arguments typed by the user**
- **Input to a command:**
  - **Keyboard, file, or other commands**
- **Standard input: keyboard.**
- **Standard output: screen.**
- **These STDIN and STDOUT are often together referred to as a terminal.**
- **Both standard input and standard output can be redirected from/to a file or other command.**
- **File redirection:**
  - **< input**
  - **> output**
  - **>> output append**

Commands

man

- **Manual Pages**
- **The first command to remember**
- **Contains info about almost everything :-)**
  - **other commands**
  - **system calls**
  - **c/library functions**
  - **other utils, applications, configuration files**
- **To read about man itself type:**

```
% man man
```

which

- **Displays a path name of a command.**

- **Searches a path environmental variable for the command and displays the absolute path.**

- **To find which `tcsh` and `bash` are actually in use, type:**
  ```
  % which tcsh
  % which bash
  ```

- **`% man which` for more details**

chsh

- **Change Login Shell**
- **Login shell is the shell that interprets commands after you logged in by default.**
- **You can change it with `chsh` (provided that your system admin allowed you to do so).**
- **To list all possible shells, depending on implementation:**
  `% cat /etc/shells`

- `%` **`chsh` with no arguments will prompt you for the shell.**

- `% echo $SHELL` **shows which shell you are using.**

whereis

- **Display all locations of a command (or some other binary, man page, or a source file).**
- **Searchers all directories to find commands that match `whereis`' argument**
- `% whereis tcsh`

General Commands

passwd

- **Change your login password.**
- **A very good idea after you got a new one.**
- **It's usually a paranoid program asking your password to have at least 6 chars in the password, at least two alphabetical and one numerical characters. Some other restrictions (e.g. dictionary words or previous password similarity) may apply.**
- **Depending on a privilege, one can change user's and group passwords as well as real name, login shell, etc.**
- `% passwd`

date

- Guess what :-)

- Displays dates in various formats
- `% date`
- `% date -u`
  - in GMT
- `% man date`

cal

- Calendar
  - for month
  - entire year
- Years range: 1 - 9999
- No year 0
- Calendar was corrected in 1752 - removed 11 days!!

- `% cal` current month
- `% cal 2 2000` Feb 2000, leap year
- `% cal 2 2100` not a leap year
- `% cal 9 1752` 11 days skipped
- `% cal 0` error
- `% cal 2015` whole year

clear

- **Clears the screen**
- **There's an alias for it: Ctrl+L**
- **Example sequence:**
  - `% cal`
  - `% clear`
  - `% cal`
  - `Ctrl+L`

sleep

- **“Sleeping” is doing nothing for some time.**
- **Usually used for delays in shell scripts.**

- `% sleep 30` **30 seconds pause**
- `% sleep 18000` **5 hours pause**

Command Grouping

- **Semicolon: “;”**
- **Often grouping acts as if it were a single command, so an output of different commands can be redirected to a file:**

- `% (date; cal) > out.txt`

alias

- Aliases are used to customize the shell session interface.

- Using alias, frequently-used commands can be invoked using a different, preferred term; and complex or commonly-used options can be used as the defaults for a given command.

alias

- Aliases persist for the current session.
- They can be loaded at login time by modifying the shell's .rc file.
- The invocation and usage of alias differs depending on the shell.

alias

- Try `% ~/.bashrc`
- this will open the `.bashrc` file
- Unix shells when starting read the `.bashrc` file and execute commands written in.
- You can put any command in that file that you could type at the command prompt.
- You put commands here to set up the shell for use in your particular environment, or to customize things to your preferences.
- A common thing to put in `.bashrc` are aliases that you want to always be available.
- (If it doesn't open, then you don't have permission to access) ☹

alias

- Defined a new name for a command
- `% alias`
  - with no arguments lists currently active aliases
- `% alias newcommand oldcommand`
  - defines a `newcommand`
- `% alias cal2015 cal 2015`

unalias

- **Removes alias**
- **Requires an argument.**
- `% unalias cal2015`

history

- Display a history of recently used commands
- `% history`
  - all commands in the history
- `% history 10`
  - last 10
- `% history -r 10`
  - reverse order
- `% !!`
  - repeat last command
- `% !n`
  - repeat command n in the history
- `% !-1`
  - repeat last command = ! !
- `% !-2`
  - repeat second last command
- `% !ca`
  - repeat last command that begins with 'ca'

exit / logout

- **Exit from your login session.**
- `% exit`
- `% logout`

shutdown

- **Causes system to shutdown or reboot cleanly.**
- **May require super-user privileges**
- `% shutdown -h now` **- stop**
- `% shutdown -r now` **- reboot.**

- `% shutdown -h +5 "Server is going down for upgrade. Please save your work."`

- `% shutdown -r +5 "Server is going down for upgrade. Please save your work."`

- The h option is for **halt** which means to stop. The second parameter is the time parameter. "now" means that shutdown the system right away.

shutdown

```
[root@dhcppc1 ~]#
Broadcast message from root@dhcppc1
        (/dev/tty1) at 21:35 ...

The system is going down for reboot in 5 minutes!
Server will restart in 5 minutes. Please save your work.
```

Files

ls

- List directory contents
- Has whole bunch of options, see man `ls` for details.
- `% ls`
  - all files except those starting with a "."
- `% ls -a`
  - all
- `% ls -A`
  - all without "." and ".."
- `% ls -F`
  - append "/" to dirs and "*" to executables
- `% ls -l`
  - long format
- `% ls -alrt`
- `% ls -lt`
  - sort by modification time (latest - earliest)
- `% ls -ltr`
  - reverse

pwd (path work directory?)

- **to find out the absolute pathname of your home-directory, type cd to get back to your home-directory and then type**
- **`% pwd`**

more / less

- **Pagers to display contents of large files page by page or scroll line by line up and down.**
- **Have a lot of viewing options and search capability.**
- **Interactive. To exit: press '`q`' for quit**

less

- `less` ("less is more") a bit more smart than the `more` command

- to display contents of a file:
  - `% less filename`
- To display line numbers:
  - `% less -N filename`
- To display a prompt:
  - `% less -P "Press 'q' to quit" filename`

touch

- **By *touching* a file you either create it if it did not exists (with 0 length).**

- **Or you update it's last modification and access times.**
- `% touch hello.txt`

cp

- **Copies files / directories.**
- `% cp [options] <source> <destination>`
- `% cp file1 file2`

- **Useful option: `-i` to prevent overwriting existing files and prompt the user to confirm.**

mv

- Moves or renames files/directories.
- `% mv <source> <destination>`
  - The <source> gets removed
- Moves files to directories.
- `% mv file1 dir/`
- (Tricky) Moves files to files.
- `% mv file1 file2`
  - rename
- Moves 2 files to directories.
- `% mv file1 file2 dir/`
- Moves directories to directories.
- `% mv dir1 dir2`

rm

- Removes file(s) and/or directories.
- `% rm file1 [file2]`

- (will not work if directory has file in it)
- `% rm dir1`

- (Danger) Removes file(s) recursively .
- `% rm -r dir1`

- Removes file(s) and directories.
- `% rm -r file1 dir1 dir2 file4`

script

- **Writes a log (a typescript) of whatever happened in the terminal to a file.**
- `% script [file]`
- `% % script file`
  - **all log is saved into a file named file**
- **To exit logging, type:**
  - `% exit`

find

- **Looks up a file in a directory tree.**
- `% find . -name` ***`whathappened`***

mkdir

- **Creates a directory.**
- `% mkdir tomandjerry`
- **Often people make an alias of `md` for it.**

cd

- Changes your current directory to a new one.
- `% cd /some/other/dir`
  - Absolute path
- `% cd subdir`
  - Assuming `subdir` is in the current directory.
- `% cd`
  - Returns you to your home directory.
- `% cd .`
  - Returns you to current directory.
- `% cd ..`
  - Returns you to parent directory.

rmdir

- **Removes a directory.**
- `% rmdir dirname`

- **(Danger) Equivalent:**
  - `% rm -r dirname`

extra

- **Who?**
- `% who`

- **Whoami?**
- `% whoami`

- **logname**
- `% logname`

- **List all Unix users**
- `% users`

extra

- % rev
- Then type what is going on here ?

- % factor
- Then type 10

- Check OS version
- % uname -a

extra

- **Check Unix distribution**
- **% cat /etc/lsb-release**

- **Check OS version**
- **% cat /etc/issue.net**

- **List all Unix users**
- **% cat /etc/passwd**

- **Find info about CPUs**
- **% cat /proc/cpuinfo**

File permissions

- **Each file in Unix/Linux has an associated permission level**
- **This allows the user to prevent others from reading/writing/executing their files or directories**
- **Use "ls -l *filename*" to find the permission level of that file**

File permissions

- **Following are the symbolic representation of three different roles:**
  - **u is for user,**
  - **g is for group,**
  - **and o is for others.**
- **Following are the symbolic representation of three different permissions:**
  - **r is for read permission,**
  - **w is for write permission,**
  - **x is for execute permission.**
  - **In case of directory, “x” grants permission to list directory contents**

File permissions

- **Each digit in the mode parameter represents the permissions for a user or a class of users:**

- **the first digit corresponds to the owner**
- **the second digit corresponds to group**
- **the third digit corresponds to others**

File permissions

- **There are eight digits that can be used in the mode parameter.**

  0 - Deny all

  1 - Execute Only

  2 - Write Only

  3 - Write + Execute

  4 - Read Only

  5 - Read + Execute

  6 - Read + Write

  7 - Read + Write + Execute

Permission levels

- **"r" means "read" permission**
- **"w" means "write" permission**
- **"x" means "execute" permission**
  - **In case of directory, "x" grants permission to list directory contents**

File Permissions

wiehe@zhome:~/linux_tutorial

```
zhome:~/linux_tutorial$ ls -l
total 28
-rw-rw-r--  1 wiehe wiehe  169 Aug 30 12:20 aa_sequence.pl
-rw-rw-r--  1 wiehe wiehe   92 Aug 30 11:54 ACTG.pl
-rw-rw-r--  1 wiehe wiehe   21 Aug 30 12:23 data.dat
-rw-rw-r--  1 wiehe wiehe   42 Aug 30 12:22 hello_world.pl
-rw-rw-r--  1 wiehe wiehe   24 Aug 30 12:23 input.txt
-rw-rw-r--  1 wiehe wiehe   50 Aug 30 13:13 lines.txt
drwxrwxr-x  2 wiehe wiehe 4096 Aug 30 13:19 new_directory
zhome:~/linux_tutorial$
```

User (you)

File Permissions

wiehe@zhome:~/linux_tutorial

```
zhome:~/linux_tutorial$ ls -l
total 28
-rw-rw-r--  1 wiehe wiehe  169 Aug 30 12:20 aa_sequence.pl
-rw-rw-r--  1 wiehe wiehe   92 Aug 30 11:54 ACTG.pl
-rw-rw-r--  1 wiehe wiehe   21 Aug 30 12:23 data.dat
-rw-rw-r--  1 wiehe wiehe   42 Aug 30 12:22 hello_world.pl
-rw-rw-r--  1 wiehe wiehe   24 Aug 30 12:23 input.txt
-rw-rw-r--  1 wiehe wiehe   50 Aug 30 13:13 lines.txt
drwxrwxr-x  2 wiehe wiehe 4096 Aug 30 13:19 new_directory
zhome:~/linux_tutorial$
```

Group

File Permissions

wiehe@zhome:~/linux_tutorial

```
zhome:~/linux_tutorial$ ls -l
total 28
-rw-rw-r--  1 wiehe wiehe  169 Aug 30 12:20 aa_sequence.pl
-rw-rw-r--  1 wiehe wiehe   92 Aug 30 11:54 ACTG.pl
-rw-rw-r--  1 wiehe wiehe   21 Aug 30 12:23 data.dat
-rw-rw-r--  1 wiehe wiehe   42 Aug 30 12:22 hello_world.pl
-rw-rw-r--  1 wiehe wiehe   24 Aug 30 12:23 input.txt
-rw-rw-r--  1 wiehe wiehe   50 Aug 30 13:13 lines.txt
drwxrwxr-x  2 wiehe wiehe 4096 Aug 30 13:19 new_directory
zhome:~/linux_tutorial$
```

“The World”

chmod

	USER	GROUP	WORLD
READ	4	4	4
WRITE	2	2	0
EXECUTE	1	0	1
Add together the columns and your permission would be	7	6	5

- Example: `chmod 123 example.html`
- 644 meaning owner can read and write and group and world can read only

0	1	2	3	4	5	6	7
Deny all	X	W	W & X	R	R& X	R & W	RWX

File permissions

- **Examples: You set your directory permissions to 755. This means that the directory owner can read, write, and execute, while group and world can read and execute (use) the directory.**

- **You set your file permissions to 644. This means that the file owner can read and write (edit) the file, while everyone else can only read it.**

0	1	2	3	4	5	6	7
Deny all	X	W	W & X	R	R& X	R & W	RWX

chmod

0	1	2	3	4	5	6	7
Deny all	X	W	W & X	R	R& X	R & W	RWX

- We will start with the easier way by using numbers to set permissions

```
chmod  777  example.html
```

- will set the file named example in the current directory to read write and execute for everyone

```
chmod  755 *.cgi
```

- will set all the files with the extension cgi in the current directory to read write and execute for the user read and execute for the group and world.

```
chmod  777  directory
```

- will set the permissions for all files and sub directories of the directory named directory to read, write and execute for everyone.

Editors

- **There are a lot of available editors under linux operating system.**
- **Amongst these vi is the most common one. One can claim that every unix system has vi.**
- **The other famous editor is emacs which has some artificial intelligence properties.**
- **The mailing facility pine uses the pico editor.**

End of 01_01

- **This is a nice tutorial for beginners, give it a try**

- **Unix Tutorial for Beginners**
- **http://www.ee.surrey.ac.uk/Teaching/Unix/**