Contrary to popular belief, Unix *is* user friendly. It just happens to be very selective about who it decides to make friends with.

Unknown
Unix is a family of multiuser, multitasking operating systems stemming from the original Unix developed in the 1970’s at Bell Labs by Ken Thompson, Dennis Ritchie\(^1\), and others.

Some consider Unix to be the second most important invention to come out of AT&T Bell Labs after the transistor.

\[\text{Dennis Ritchie}\]

\(^{1}\) Also famous for creating the C programming language.
Unix is a family of multiuser, multitasking operating systems stemming from the original Unix developed in the 1970’s at Bell Labs by Ken Thompson, Dennis Ritchie\(^1\), and others.

Some consider Unix to be the second most important invention to come out of AT&T Bell Labs after the transistor.

Dennis Ritchie

Initially used at Bell Labs, but soon licensed to academy (notably, U. of California, Berkeley) and commercial vendors (IBM, Sun, etc).

There are two major products that came out of Berkeley: LSD and Unix. We don’t believe this to be a coincidence.

Jeremy S. Anderson, Unix systems administrator

\(^1\) Also famous for creating the C programming language.
GNU/Linux is a Unix-like computer operating system assembled under the model of free and open-source software development and distribution.

The use of the Unix philosophy just for Unix was a great waste. Fortunately, Linux came along.

Unknown
GNU/Linux

GNU/Linux is a Unix-like computer operating system assembled under the model of free and open-source software development and distribution.

The use of the Unix philosophy just for Unix was a great waste. Fortunately, Linux came along.

Unknown

The core component—the Linux kernel—was first released in 1991 by Linus Torvalds.

Today, every Android device runs Linux kernel. (And iOS devices run another Unix-derived kernel.)
UNIX is very simple, it just needs a genius to understand its simplicity.

Dennis Ritchie

A kernel connects the application software (programs & utilities) to the hardware of a computer.
Unix shell

In fact, we started off with two or three different shells and the shell had life of its own.

Ken Thompson

Shell is an interpreter that provides a *command-line* interface (CLI), contrary to a graphical user interface (GUI).

- For kernel, shell is just another program/utility
- For user, first among equals
- There are several shells: `bash`, `tcsh`, `zsh`,...
- We will use `bash`
The one thing I stole was the hierarchical file system because it was a really good idea...

Ken Thompson
File system II

- top-level entry = “/” (root of the file system);
- “/” (forward slash) is a path separator;

For example, full path to my home directory is

/afs/ictp.it/home/e/estambul
File system II

- top-level entry = “/” (root of the file system);
- “/” (forward slash) is a path separator;

For example, full path to my home directory is

/afs/ictp.it/home/e/estambul

Naming rules:
- a file or directory name may contain any character/symbol except for “/”;
- no special meaning for “file extension”.
top-level entry = “/” (root of the file system);
“/” (forward slash) is a path separator;

For example, full path to my home directory is

/afs/ictp.it/home/e/estambul

Naming rules:

- a file or directory name may contain any character/symbol except for “/”;
- no special meaning for “file extension”.

Having said that, though...

Good practice for file naming:

- Don’t use spaces! Use _ (underscore) instead.
- Only alphanumeric characters (A–Z, a–z, 0–9).
- Use file extensions consistently.
Basic file-system navigation I

- cd – change directory
- ls – list file(s)
- pwd – print pathname of the working directory

OK, let's get our hands dirty... Open a terminal emulator:
Basic file-system navigation I

- `cd` – change directory
- `ls` – list file(s)
- `pwd` – print pathname of the working directory

OK, let’s get our hands dirty... Open a terminal emulator:
Special directory names:

- `/` – file-system root
- `.`  – current directory
- `..` – parent directory (up one level)
- `~`  – home, sweet home directory
- `-`  – previously visited directory (only for `cd`)
Special directory names:
- / – file-system root
- . – current directory
- .. – parent directory (up one level)
- ~ – home, sweet home directory
- – previously visited directory (only for cd)

Wildcards:
- * – any number of any characters
- ? – a single character (any)
Shell movie making. Part I – write your script

- `script` – make typescript of terminal session

Try:

```bash
% script lesson1.out
Script started, file is lesson1.out
% (now play with the commands learned so far - ls, cd, etc)
% ...
% (at the end, type exit):
% exit
Script done, file is lesson1.out
%
```

Note: here and below, “%” denotes the shell prompt - don’t type it!
more – a simple text pager

% more lesson1.out

Hint: Ctrl+L to clear the screen.
Shell movie making. Part II – review the script

- more – a simple text pager

  % more lesson1.out

  Hint: **Ctrl**+L to clear the screen.

- less – when less is better than more

  % less lesson1.out
Shell movie making. Part II – review the script

- **more** – a simple text pager

  ![Command](more lesson1.out)

  **Hint:** `Ctrl+L` to clear the screen.

- **less** – when less is better than more

  ![Command](less lesson1.out)

While in **less**:

- `Ctrl+F` or `Space` – move forward
- `Ctrl+B` – move backward
- `/` (followed by a pattern) to search forward; `?` – backward
- `n` to repeat a previous search;
- `q` to quit.
Shell movie making. Part III – the real thing

% script --timing=lesson1.tm lesson1.out
Script started, file is lesson1.out
% (make some fun again with ls, cd, etc)
% script --timing=lesson1.tm lesson1.out
Script started, file is lesson1.out
% *(make some fun again with ls, cd, etc)*

**Note:** most shell commands accept various options (here, --timing=filename). For basic usage, try the --help flag with a command you know, e.g.:

% script --help

or

% more --help
OK, time to finish shooting:

```
% exit
Script done, file is lesson1.out
%
```

**Hint:** you could hit \( \text{Ctrl} + \text{D} \) instead of exit.
OK, time to finish shooting:

```
% exit
Script done, file is lesson1.out
%
```

**Hint:** you could hit \[Ctrl\]+\[D\] instead of exit.

- `scriptreplay` – play back typescripts, using timing information

```
% scriptreplay --timing=lesson1.tm lesson1.out
```

Enjoy!
Creating and modifying files and directories

- `mkdir` – make directory/ies
- `rmdir` – remove directory/ies
- `cp` – copy file(s) or directory/ies
- `mv` – move file(s) or directory/ies
- `rm` – remove file(s) or directory/ies

Note 1: `cp`, `mv`, and `rm` are dangerous commands – by default, overwrite/delete files without warning.

Note 2: There is no “undelete” command...

Hint: Use `cp -i`, `mv -i`, `rm -i` for safety.
Creating and modifying files and directories

- `mkdir` – make directory/ies
- `rmdir` – remove directory/ies
- `cp` – copy file(s) or directory/ies
- `mv` – move file(s) or directory/ies
- `rm` – remove file(s) or directory/ies

**Note 1:** `cp`, `mv`, and `rm` are dangerous commands – by default, overwrite/delete files without warning.

**Note 2:** There is no “undelete” command...

**Hint:** Use `cp -i`, `mv -i`, `rm -i` for safety.
Instead of typing “rm -i” etc everytime, one can use aliases:

- alias – define or display alias(es).
- unalias – remove alias(es).

```bash
% alias rm='rm -i'
% alias mv='mv -i'
% alias cp='cp -i'
```
Aliases

Instead of typing “rm -i” etc everytime, one can use aliases:

- alias – define or display alias(es).
- unalias – remove alias(es).

```bash
% alias rm='rm -i'
% alias mv='mv -i'
% alias cp='cp -i'
```

In fact, you’ve used some “aliased” commands unknowingly:

```bash
% alias ls
alias ls='ls --color=auto'
```

**Hint:** to invoke a “pristine” command, prepend it with a backslash:

```bash
% \ls
```
Editing files

Good old times: Church of Emacs vs. Cult of vi...

There are more things that vi can do, Horatio, than are dreams in your philosophy.

Vimlet, prince of Benchmark

Even though I no longer use Emacs regularly, I’m still on the side of good in the editor wars.

Unknown

Unfortunately, there are many more editors today!..
There are more things that vi can do, Horatio, than are dreams in your philosophy.

Vimlet, prince of Benchmark

Even though I no longer use Emacs regularly, I’m still on the side of good in the editor wars.

Unknown

Unfortunately, there are many more editors today!..

In fact, you can invoke GUI programs from the shell prompt. Try

% gedit
While gedit is running, try using the shell. What happens?
While gedit is running, try using the shell. What happens?

- **Ctrl**+Z – suspend the current job
- bg – send a suspended job to background
- fg – send a job to foreground
- ... & – execute a command in the background
- jobs – list jobs
- **Ctrl**+C – stop the current job
- kill – stop (send a signal to) a job (or process)
Cat superpowers & redirections

- cat – concatenate² files and print on the standard output

Redirections: of standard output (>) and (>>) and standard input (<).

²No animals were harmed in the making of this slide.
References and further reading (and watching)


Stephenson, N. (1999). *In the Beginning...was the Command Line*. William Morrow Paperbacks, New York.