

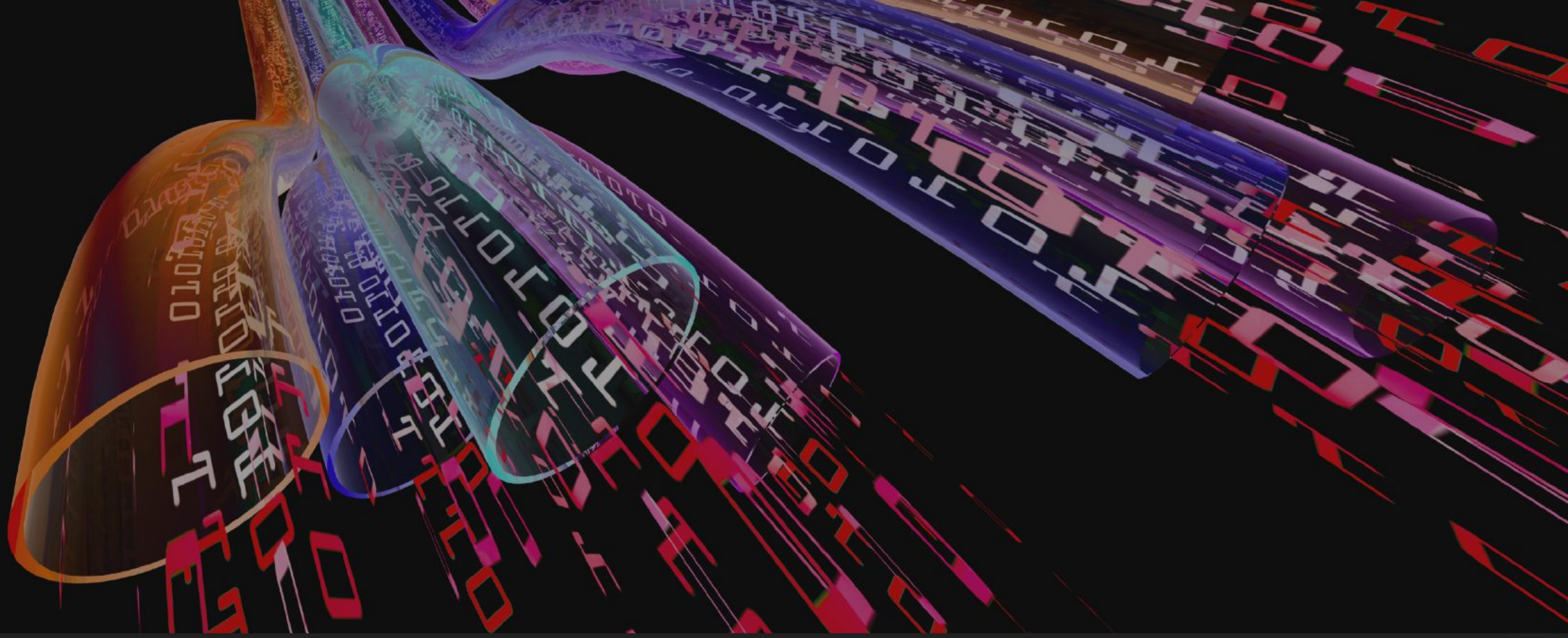


BEHOLD, THE UNIX COMMAND LINE

INTRODUCTION TO UNIX, SHELLS AND YOUR COMMAND LINE

WHY LEARN UNIX?

- ▶ UNIX is the dominant server operating system “flavor”
- ▶ Even Microsoft Cloud — Azure runs 90% on Linux
- ▶ Your MacBook Pro runs a variant of UNIX called Darwin, based on BSD
- ▶ You have some powerful tools at your fingertips



BRIEF HISTORY OF UNIX

BRIEF HISTORY OF UNIX

IN THE BEGINNING... THERE WAS DARKNESS

- ▶ It was called MULTICS (short for Multiplexed Information and Computing System) — OS created by Bell Labs
- ▶ It was written 100% in Assembly, and was too slow to run on affordable computers at the time.
- ▶ So Bell Labs cancelled the MULTICS project
- ▶ But MULTICs had one thing going for it — a game called “Space Travel”

MAMA, I WANT TO PLAY MY GAME!!!!!!

- ▶ Ken Thompson tried therapy, meditation, yoga and drugs — but none provided the same level of enlightenment as Space Travel.
- ▶ So he decided to port it to a “little used PDP-7 sitting in the corner”



MAMA, I WANT TO PLAY MY GAME!!!!!!

- ▶ PDP was made by DEC (Digital Equipment Corporation) — a computer giant of that era
- ▶ It was far less powerful, but it was already there and it was already purchased by AT&T



BEFORE YOU CAN RESCUE THE PRINCESS, PLEASE WRITE THE OS KERNEL

- ▶ Just imagine if I told you — “To play the next version of World of Warcraft you have create a file system, device drivers, scheduler, the kernel and the OS to run your game on, but after that — enjoy yourself”.
- ▶ Sounds insane, right?
- ▶ But that’s exactly what Ken Thompson and Dennis Richie decided to do on PDP-7.
- ▶ The game quickly turned into a full-blown Operating System
- ▶ The year was 1969.

AND OUT OF ADAM (KEN) AND DENNIS (EVE) UNIX WAS BORN

- ▶ AT&T wasn't in the business of selling Operating Systems
- ▶ It wasn't until 1977 that Sun Microsystems resold UNIX
- ▶ At the same time University of Berkeley began to redistribute UNIX with it's own shell, editor, and some additional drivers.
- ▶ And that's when the split occurred. Berkeley took the sources from Bell Labs, modified, gave it to students, added things, and created a totally different OS that happened to support the same commands.
- ▶ For years Berkeleyites had the edge.

BSD UNIX AND THE SPLIT

- ▶ Berkeley had a liberal licensing policy (pun intended)
- ▶ Many universities licensed it, and gave it to their own students to play with
- ▶ Many graduates were appalled that, upon joining software industry, they discovered the original UNIX lacked a good editor, networking, and many other cool things invented by BSD.
- ▶ AT&T responded by “borrowing” these features into the main UNIX

1969

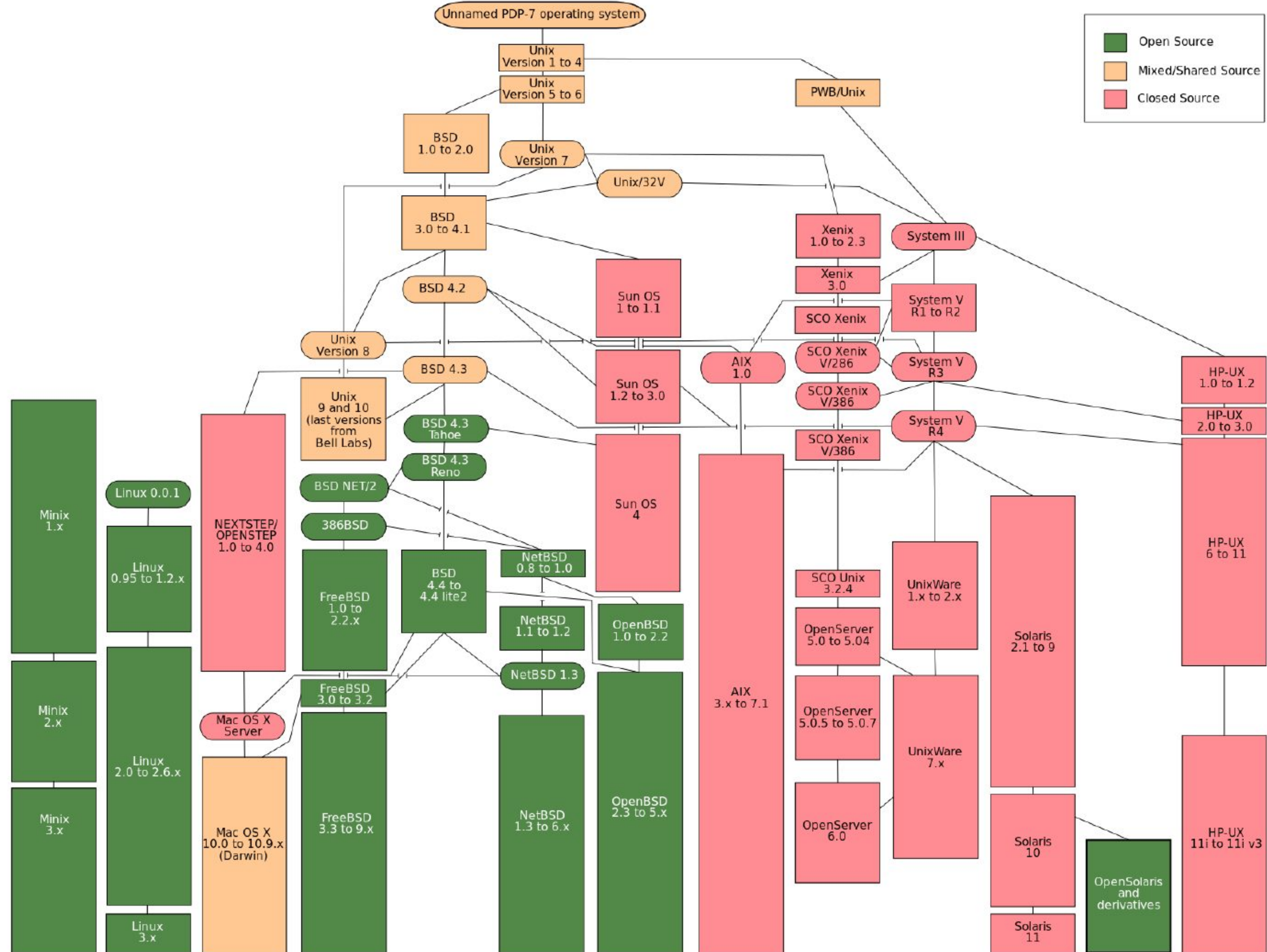
1980

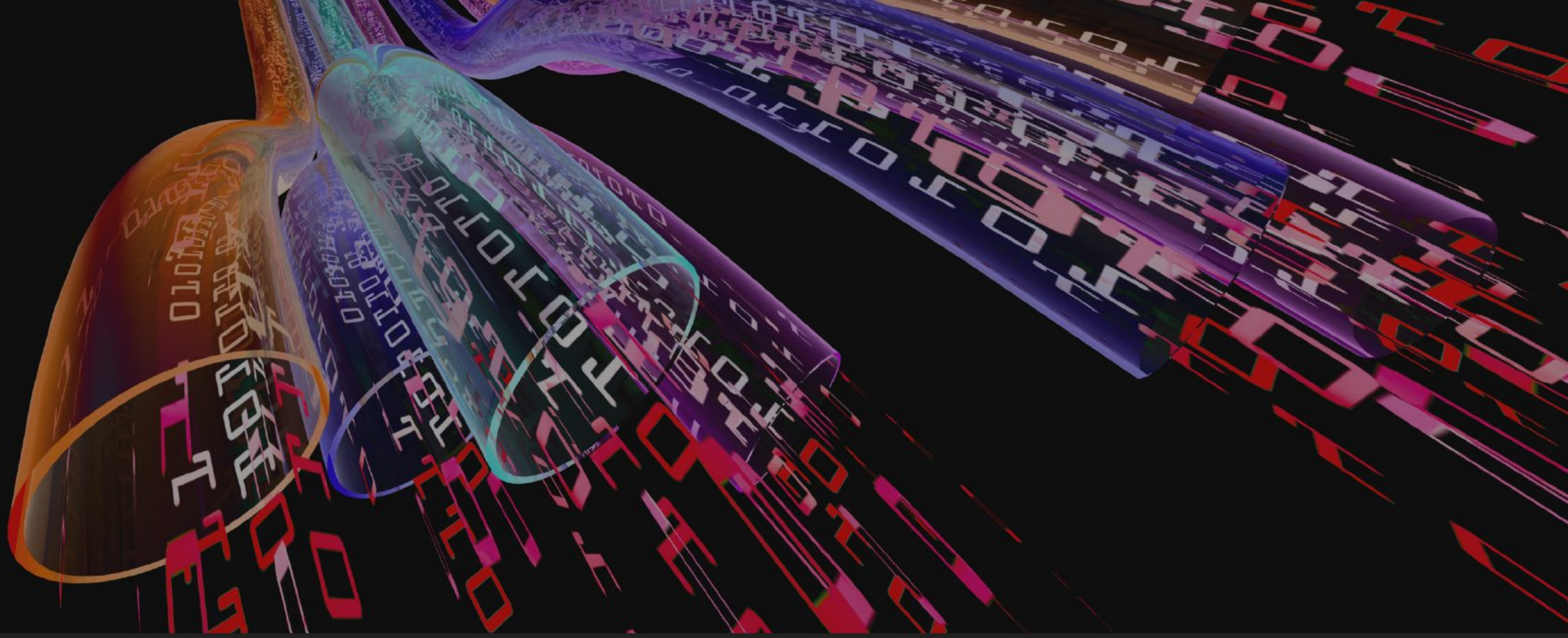
1990

2000

2010

1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2004
2005
2006 to 2007
2008
2009
2010
2011
2012 to 2013





CORE CONCEPTS

THREE DISTINCT LEVELS

★ User Level:

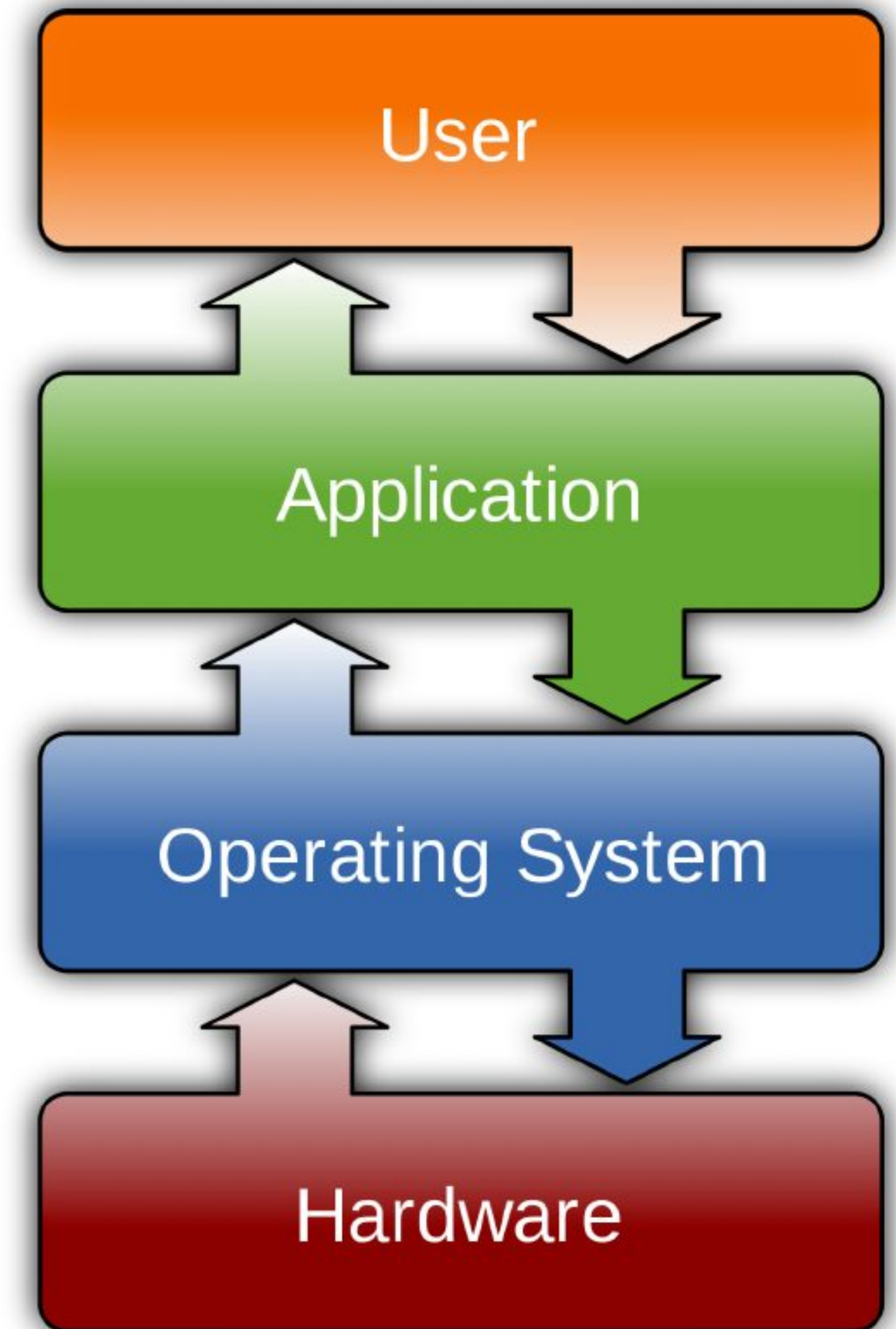
Apps, shell, UI, libraries, higher level languages (ruby)

★ Kernel

Processes, scheduling, device drivers

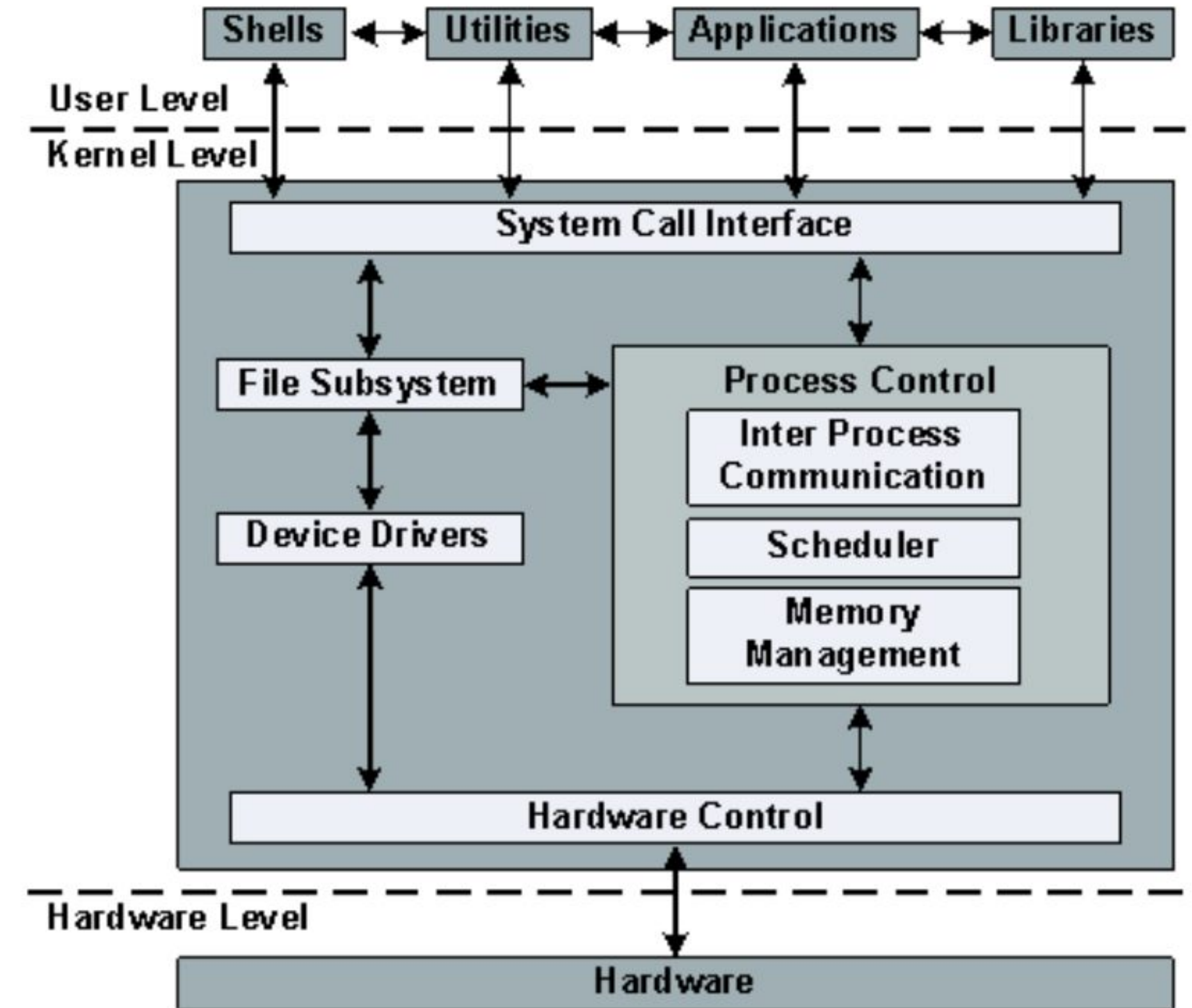
★ Hardware

Storage, RAM, terminal, peripherals



THREE LAYERS — ZOOMING IN

- ▶ User level apps, utilities, libraries, shells
- ▶ System Call Interface
- ▶ File System
- ▶ Device Drivers
- ▶ Process Control (IPCs, Scheduler, Memory)
- ▶ Hardware



A UNIX SYSTEM CALL

- ▶ **A system call** is the programmatic way in which a program requests a service from the kernel of the operating system it is executed on.
- ▶ This may include hardware-related services, creation and execution of new processes, and communication with integral kernel services such as process scheduling.
- ▶ System calls provide an essential interface between a process and the operating system.
- ▶ In most systems, system calls can only be made from user space processes (i.e. not from kernel drivers)

TEST YOUR KNOWLEDGE — WHICH OF THE FOLLOWING IS NOT A SYSTEM CALL?

1. `open()`

5. `exit()`

2. `write()`

6. `kill()`

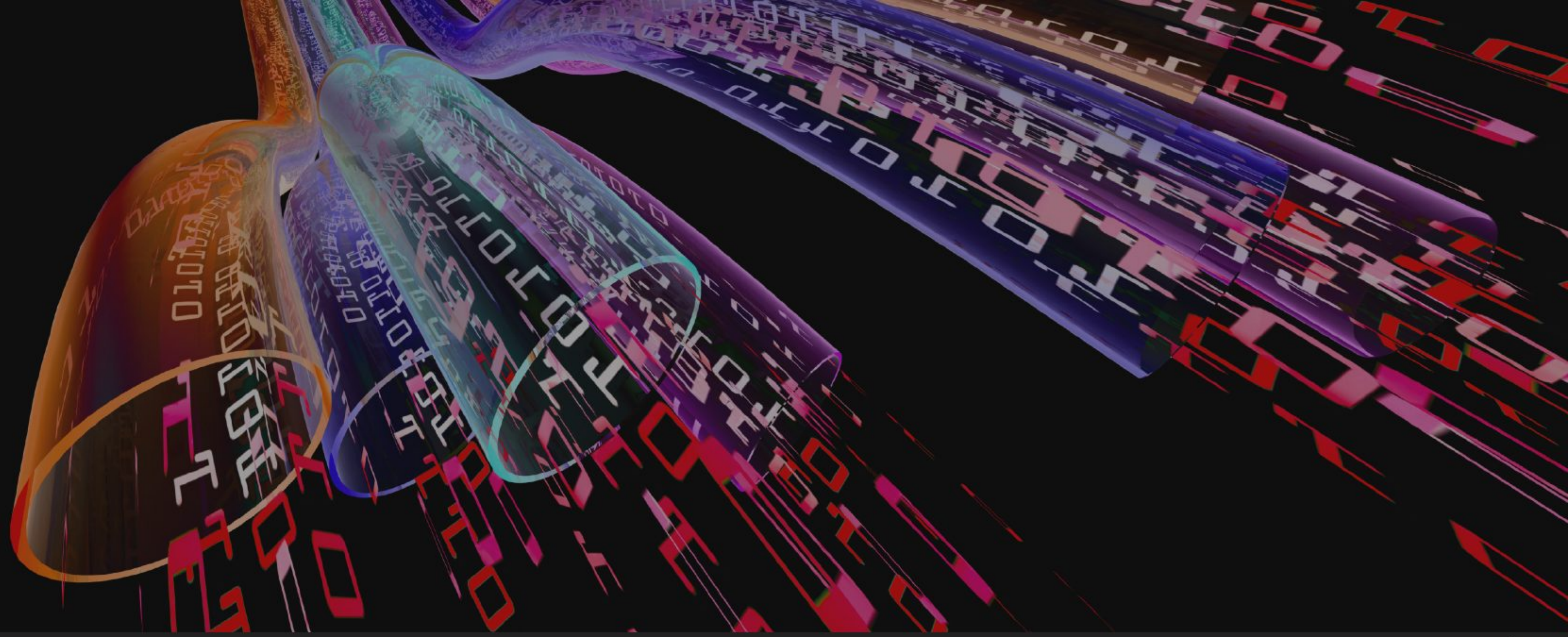
3. `fork()`

7. `wait()`

4. ~~`swap()`~~

RUBY OFFERS MOST OF THE SYSTEM CALLS “AS IS”

- ▶ When you call `File.open()` in ruby, you are executing C function “`open()`” which, in turn, is a system call.
- ▶ But there are many C functions that are offered by libraries, which are NOT system calls. For example, OpenSSL library’s encryption functions.
- ▶ System calls are limited to what’s offered by the kernel.
- ▶ You CAN NOT ADD new system calls to a kernel without recompiling it.



A NEW WAY OF THINKING

UNIX PHILOSOPHY

CLAY LUTER

A BIT OF HISTORY

- ▶ The Unix philosophy, originated by Ken Thompson, is a set of cultural norms and philosophical approaches to minimalist, modular software development.
- ▶ It is based on the experience of leading developers of the Unix operating system.
- ▶ Over time, the leading developers of Unix (and programs that ran on it) established a set of cultural norms for developing software, norms which became as important and influential as the technology of Unix itself; this has been termed the "**Unix philosophy**."

The Unix philosophy emphasizes building simple, short, clear, modular, and extensible code that can be easily maintained and repurposed by developers other than its creators. The Unix philosophy favors composability as opposed to monolithic design.

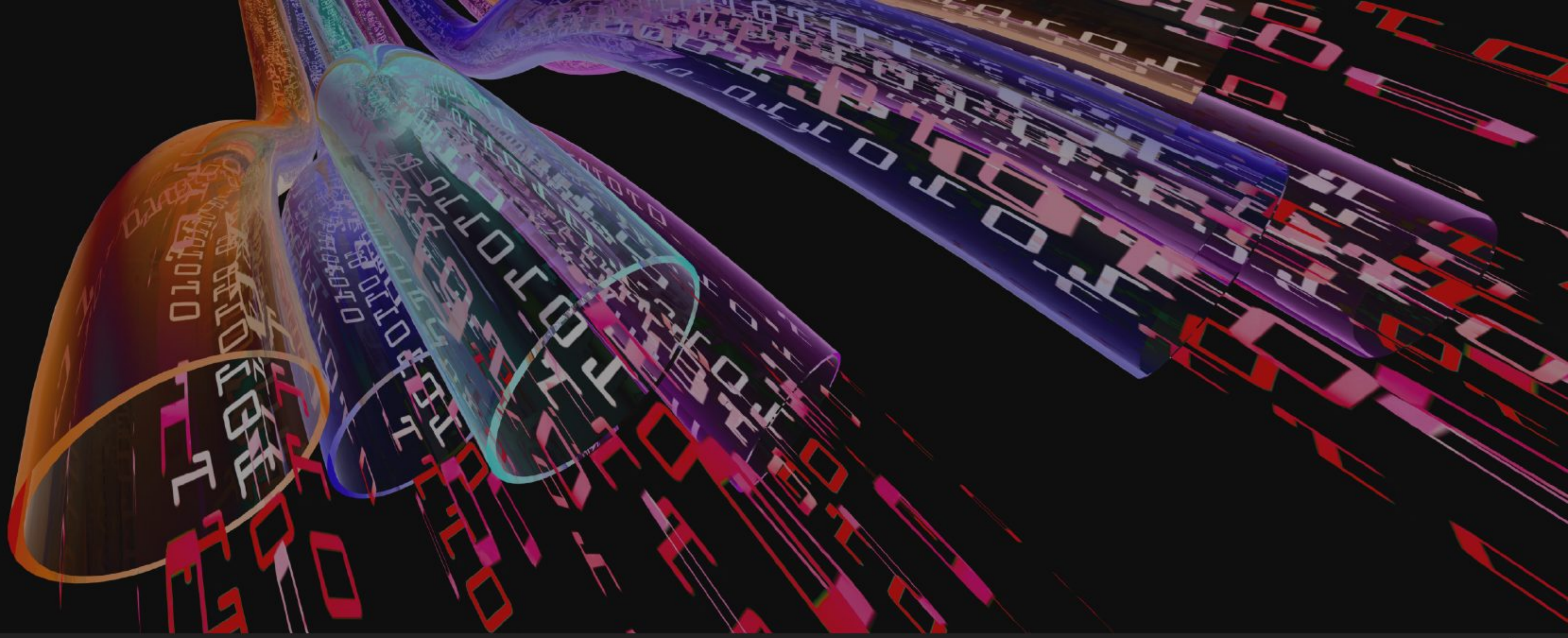
Doug McIlroy

UNIX PHILOSOPHY, BELL SYSTEM TECHNICAL JOURNAL, 1978

1. **Make each program do one thing well.** To do a new job, build afresh rather than complicate old programs by adding new "features".
2. **Expect the output of every program to become the input to another,** as yet unknown, program. Don't clutter output with extraneous information. Avoid stringently columnar or binary input formats. Don't insist on interactive input.
3. Design and build software, even operating systems, **to be tried early, ideally within weeks.** Don't hesitate to throw away the clumsy parts and rebuild them.

RITCHIE AND THOMPSON, UNIX PAPER OF 1974

1. Write programs that **do one thing and do it well**
2. Write programs to **work together**
3. **Write programs to handle text streams**, because that is a universal interface
4. Stay out of the assembler



IN SOVIET RUSSIA...

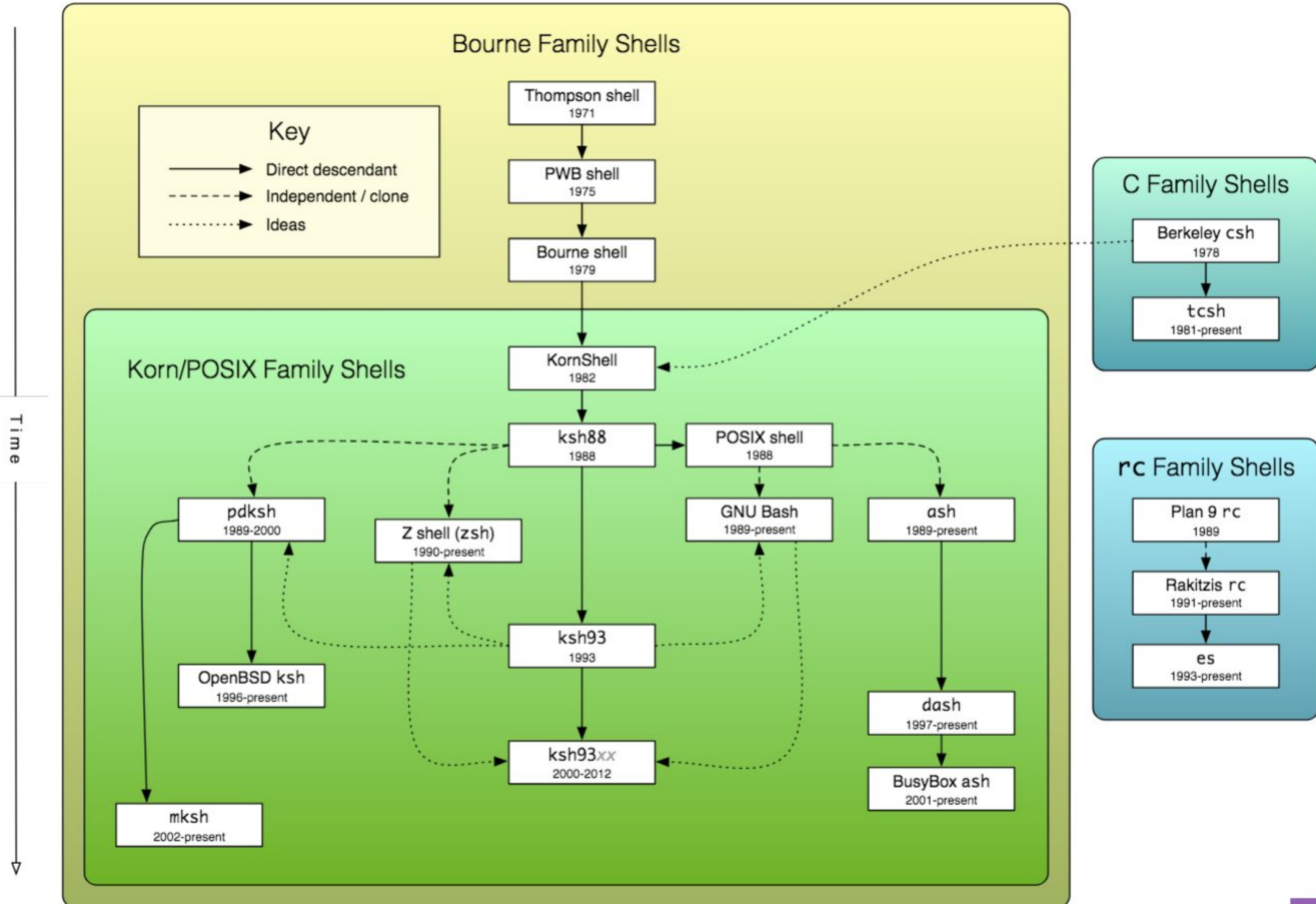
SHELL EXECUTES YOU

UNIX SHELL IS.....

- ▶ A place where Crustaceans live and hide from danger
- ▶ A program that lives between YOU and the Kernel
 - ▶ It interprets command that YOU (mis-) type, often correctly
 - ▶ Most commonly Shell is used to execute commands
 - ▶ Anytime you open your Terminal application you are interacting with a shell

BRIEF HISTORY OF SHELLS

- ▶ Who cares!
- ▶ I use IDE and Alexa.
- ▶ Hey, Alexa, “Arrr Emm Dash Arrr Efff Slash”



```
| 1. Bash Shell |-----| 2. Tcsh Shell |-----| 3. Ksh Shell |  
+-----+      +-----+      +-----+
```

```
+-----+      +-----+  
| 4. Zsh Shell |-----| 5. Fish Shell |  
+-----+      +-----+
```

5 Most Frequently Used Shells for Linux

5 Most Frequently Used Shells for Linux

MAC OS-X COMES BUILT IN WITH SEVERAL SHELLS

- ▶ To see what shells you have on your Mac, run:

```
➤ cat /etc/shells
```

- ▶ Note: ➤ sign here represents a “shell prompt”
- ▶ The shell prompt (or command line) is where one types commands.

THE DEFAULT SHELL IS... `/bin/bash`

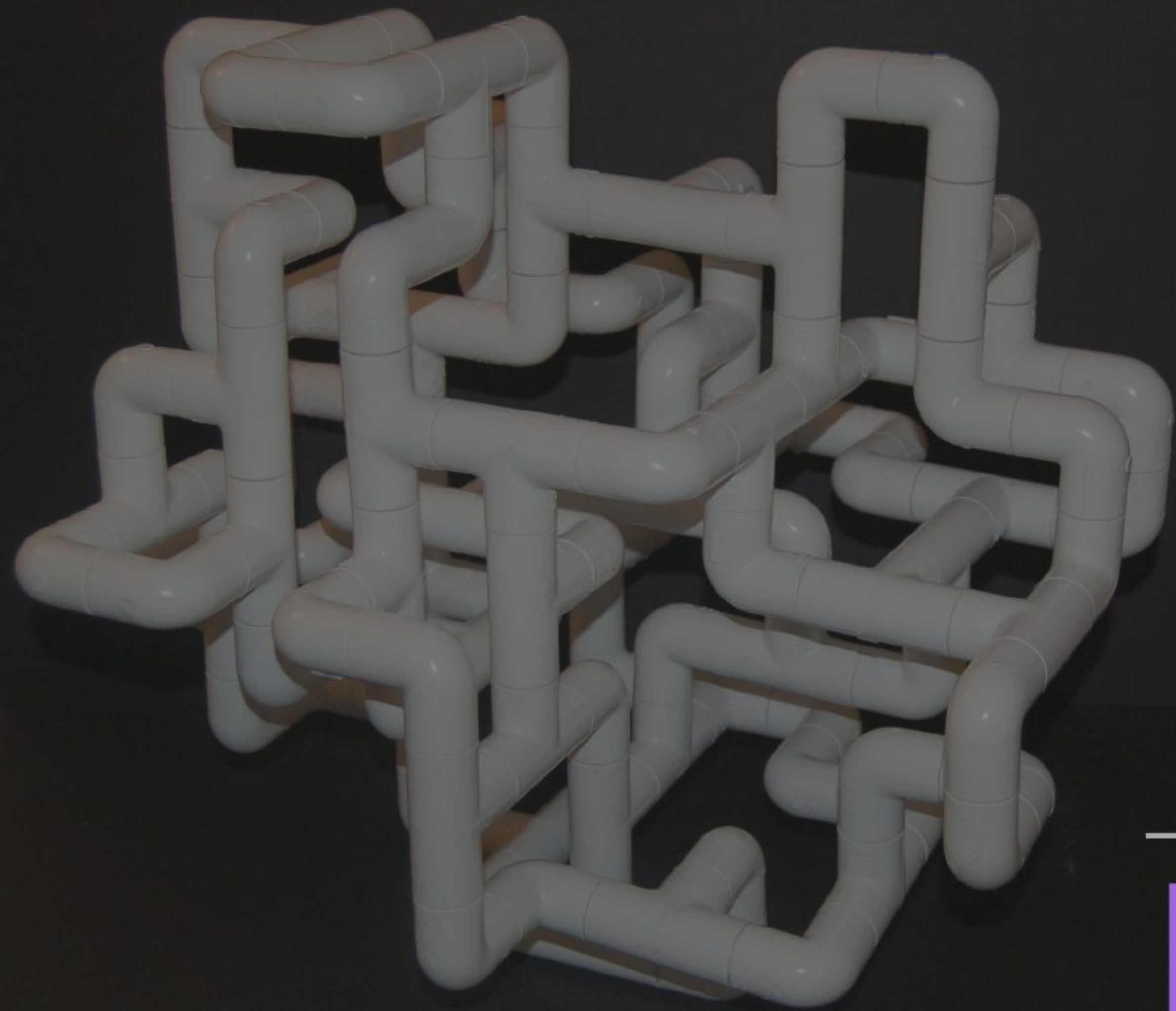
- ▶ Unfortunately, Mac OS X ships with a very old version of BASH, version 3
- ▶ This is why during app setup, we install the newer BASH (version 4) using HomeBrew

```
> echo $BASH_VERSION  
4.4.23(1)-release
```

IMPORTANT CONCEPT: STREAMS

- ▶ Standard streams are pre-connected input and output communication channels between a computer program and its environment when it begins execution.
- ▶ The three input/output (I/O) connections are called standard input (stdin), standard output (stdout) and standard error (stderr)

```
> echo hello > /dev/stdout  
hello
```



**DON'T ROLL IT BY HAND, USE A
PIPE!**

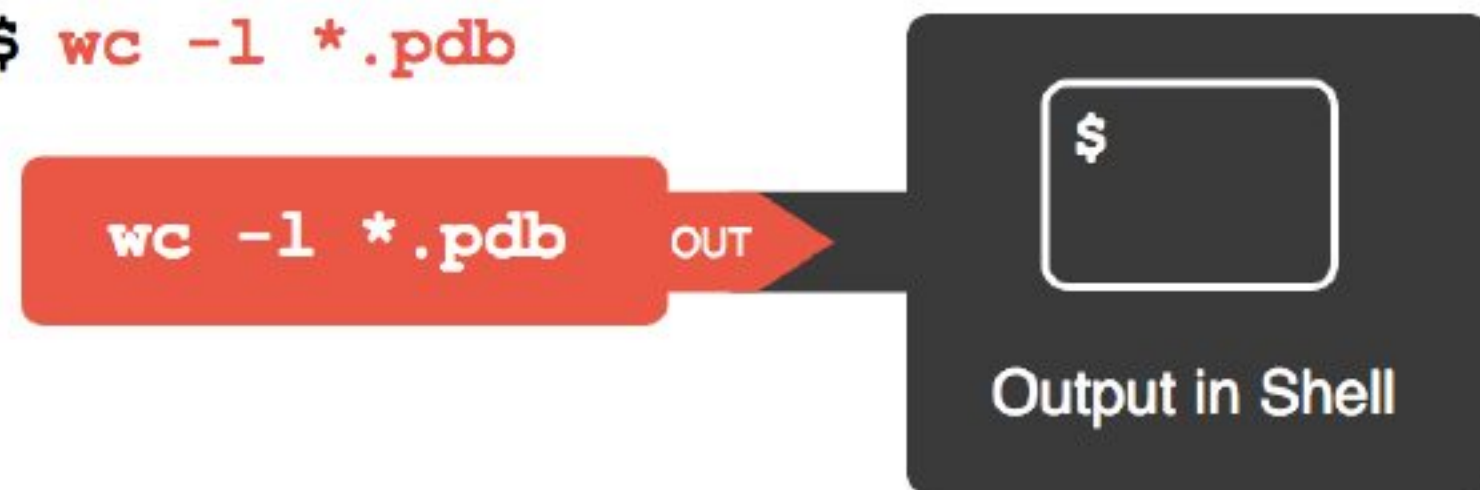
PIPES'R US

IN UNIX-LIKE COMPUTER OPERATING SYSTEMS, A PIPELINE IS A SEQUENCE OF PROCESSES CHAINED TOGETHER BY THEIR STANDARD STREAMS, SO THAT THE OUTPUT OF EACH PROCESS (STDOUT) FEEDS DIRECTLY AS INPUT (STDIN) TO THE NEXT ONE.

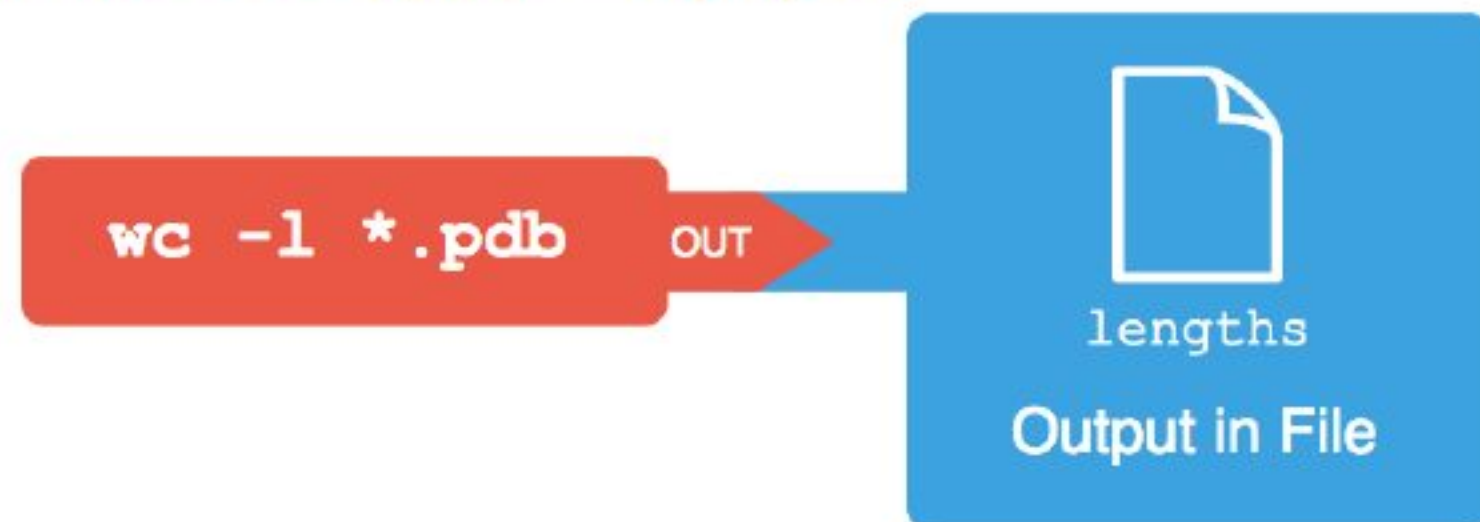
Wikipedia

OUTPUT REDIRECTION AND PIPES

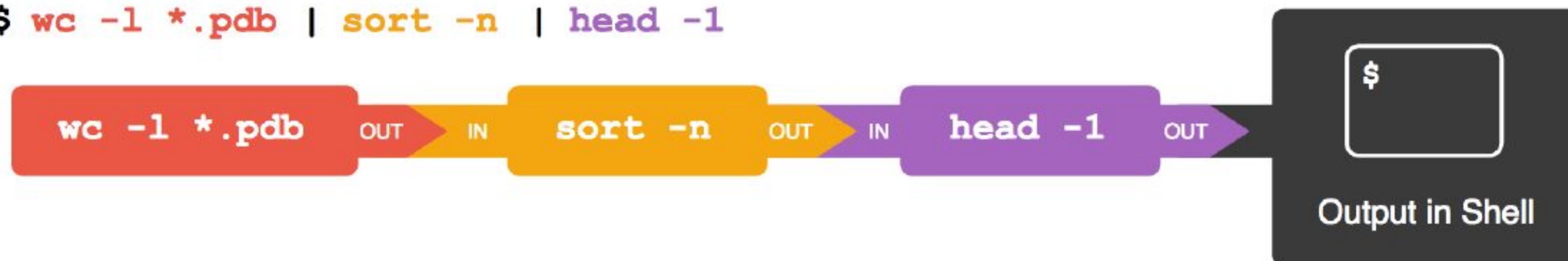
```
$ wc -l *.pdb
```

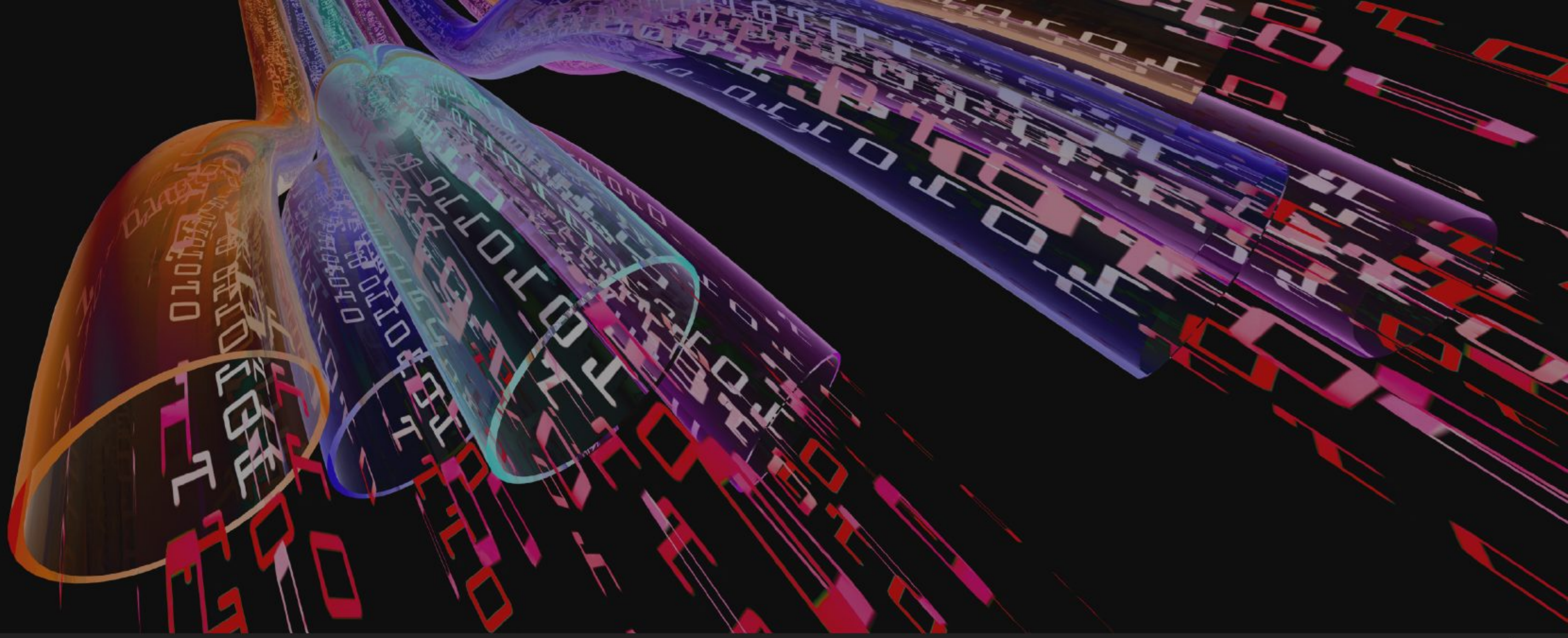


```
$ wc -l *.pdb > lengths
```



```
$ wc -l *.pdb | sort -n | head -1
```





WARNINGS: JEDI TRICKS MAY FOLLOW

MASTER THE COMMAND LINE

MASTER THE COMMAND LINE

SHELL BUILT-IN OR A SEPARATE COMMAND?

- ▶ A command you type that executes can be one of:
 - ▶ A standalone program, eg. `/bin/ls`
 - ▶ A shell built-in construct, eg. `for`
 - ▶ To see what is a command and what's not, run:

➤ **man builtin**

	Command	External	csh(1)	sh(1)
	!	No	No	Yes
	%	No	Yes	No
	.	No	No	Yes
	:	No	Yes	Yes
	@	No	Yes	Yes
	{	No	No	Yes
	}	No	No	Yes
	alias	No**	Yes	Yes
	alloc	No	Yes	No
	bg	No**	Yes	Yes
	bind	No	No	Yes
	bindkey	No	Yes	No
	break	No	Yes	Yes
	breaksw	No	Yes	No
	builtin	No	No	Yes
	builtins	No	Yes	No
	case	No	Yes	Yes
	cd	No**	Yes	Yes
	chdir	No	Yes	Yes
	command	No**	No	Yes
	complete	No	Yes	No
	continue	No	Yes	Yes
	default	No	Yes	No
	dirs	No	Yes	No
	do	No	No	Yes
	done	No	No	Yes
	echo	Yes	Yes	Yes
	echotc	No	Yes	No
	elif	No	No	Yes
	else	No	Yes	Yes
	end	No	Yes	No
	endif	No	Yes	No
	endsw	No	Yes	No
	esac	No	No	Yes
	eval	No	Yes	Yes
	exec	No	Yes	Yes
	exit	No	Yes	Yes
	export	No	No	Yes
	false	Yes	No	Yes

[illegible]

```

[1139260.978988] audit(1203037004.027:5): dev=eth1 prom=0 old_prom=256 auid=4294967295
[1139260.979014] device eth0 left promiscuous mode
[1139260.979019] audit(1203037004.027:6): dev=eth0 prom=0 old_prom=256 auid=4294967295
[1139266.044339] device eth1 entered promiscuous mode
[1139266.044350] audit(1203037009.091:7): dev=eth1 prom=256 old_prom=0 auid=4294967295
[1139266.044350]
[1139331]
[1139331] Command External csh(1) sh(1)
[1139331] set No Yes Yes
[1139331] setenv No Yes No
[1139335] settc No Yes No
[1139335] setty No Yes No
[1139343] setvar No No Yes
[1139343] shift No Yes Yes
[1139372] source No Yes No
[1139372] stop No Yes No
[1139372] suspend No Yes No
[1139372] switch No Yes No
[1139372] telltc No Yes No
[1139372] test Yes No Yes
[1139372] then No No Yes
[1139372] time Yes Yes No
[1139372] times No No Yes
[1139372] trap No No Yes
[1139372] true Yes No Yes
[1139372] type No No Yes
[1139372] ulimit No No Yes
[1139372] umask No** Yes Yes
[1139372] unalias No** Yes Yes
[1139372] uncomplete No Yes No
[1139372] unhash No Yes No
[1139372] unlimit No Yes No
[1139372] unset No Yes Yes
[1139372] unsetenv No Yes No
[1139372] until No No Yes
[1139372] wait No** Yes Yes
[1139372] where No Yes No
[1139372] which Yes Yes No
[1139372] while No Yes Yes

vte.Terminal.frame_notify
vte.Terminal.__gdoc__
vte.Terminal.get_accessible
vte.Terminal.get_action
vte.Terminal.get_activate_signal
vte.Terminal.get_adjustment
vte.Terminal.get_allocation
vte.Terminal.get_style
vte.Terminal.realize
vte.Terminal.__reduce__
vte.Terminal.__reduce_ex__
vte.Terminal.ref_accessible
vte.Terminal.region_intersect
vte.Terminal.remove_accelerator

```

FILE SYSTEM

ls
cd
mv
cp
mkdir
rmdir
rm
touch

TEXT PROCESSING

grep
uniq
sort
sed
awk
cut
comp
wc

HELP & SCREEN

man
locate
which
clear
reset
more
less

SCREEN IO, SEARCH & IDENTIFICATION

echo
printf
cat
find
file
read
type

PERMISSIONS

chmod
chown
chgrp
chsh
sudo
su

DISK & FILE SYSTEM

du
df
mount
umount
dd
fsck
fdisk

PROCESSES

ps
jobs
kill
pgrep
pkill
trace
dtrace

SYSTEM

mem
free
vmstat
iostat
top
htop
systemstats

NETWORK

netstat
trace
ping
ssh
telnet
finger
ftp

sftp
curl
wget
scp

```
> ls -al
> alias dir='ls -al'
> dir
> cd ~/
> cd ~/Desktop
> ls -alF      # show folders with /
> ls -alrtF    # sort by reverse time
```

```
> cd ~/Desktop
> mkdir UNIX
> cd UNIX
> touch README.md
> ls -al
```

```
total 0
drwxr-xr-x   3 kig  staff   96 Aug 16 10:48 .
drwxr-xr-x@ 23 kig  staff  736 Aug 16 10:48 ..
-rw-r-----  1 kig  staff    0 Aug 16 10:48 README.md
```

WAT?

-rw-r-----@

- ▶ First hyphen:
 - ▶ - is for a regular file
 - ▶ "d" is for a directory
 - ▶ "l" is for a link
 - ▶ @ means extended attributes on Mac OS-X
 - ▶ You can run "xattr -l <file>" to find out what they are

WAT?

-rw-r-----@

- ▶ The rest is three groups of three characters: r, w, x or -.
- ▶ r = read, w = write, x = execute (or, if it's a folder, cd into)
- ▶ three groups are: owner, group and other.

[d1-] rwx rwx rwx [@]

CHANGING PERMISSIONS

```
> ls -al
```

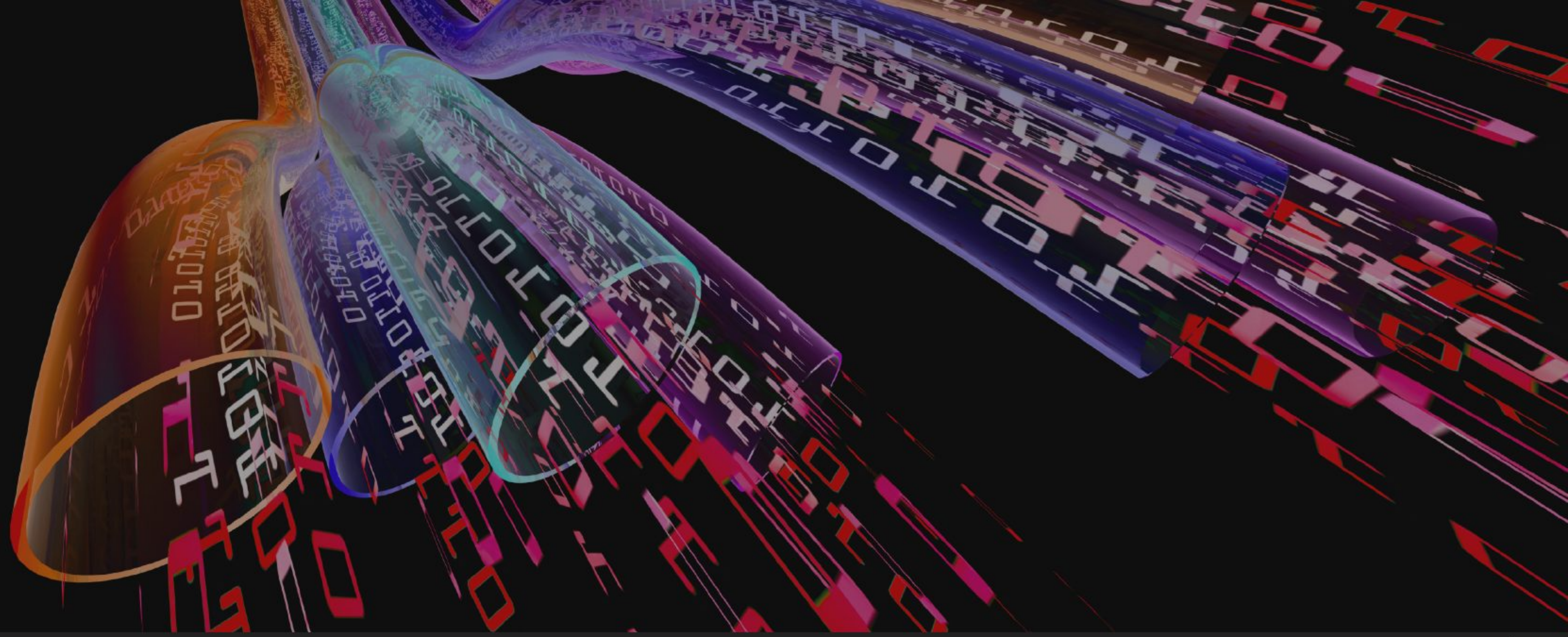
```
total 0
```

```
drwxr-xr-x   3 kig  staff   96 Aug 16 10:48 .  
drwxr-xr-x@ 23 kig  staff  736 Aug 16 10:48 ..  
-rw-r-----  1 kig  staff    0 Aug 16 10:48 README.md
```

```
> chmod o+rw,u+x README.md
```

```
> ls -al R*
```

```
-rwxr--rw-   1 kig  staff    0 Aug 16 10:48 README.md
```

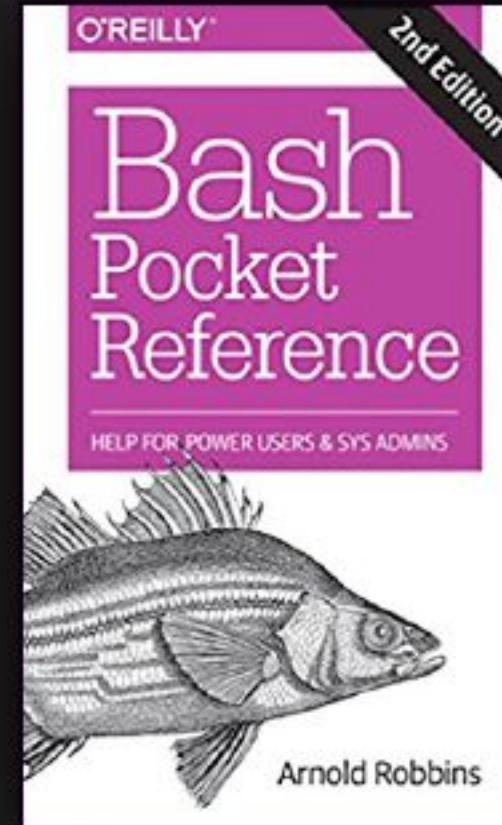
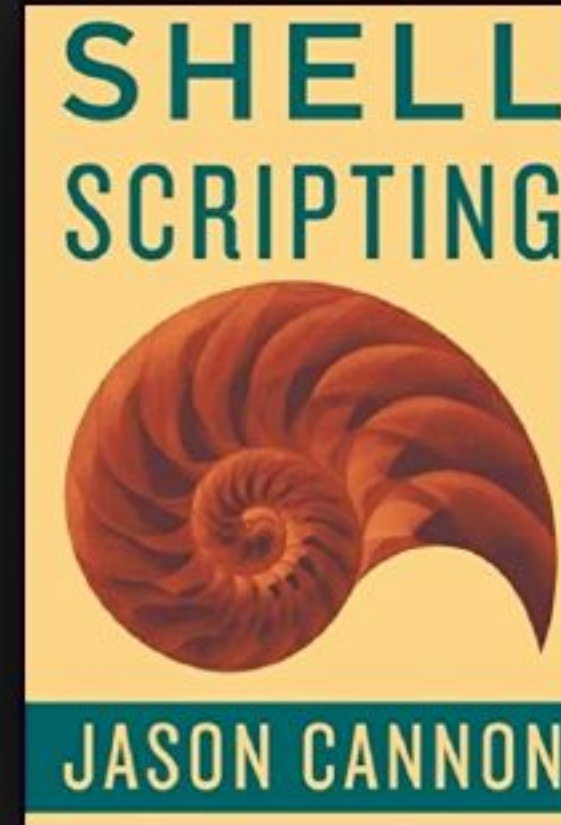
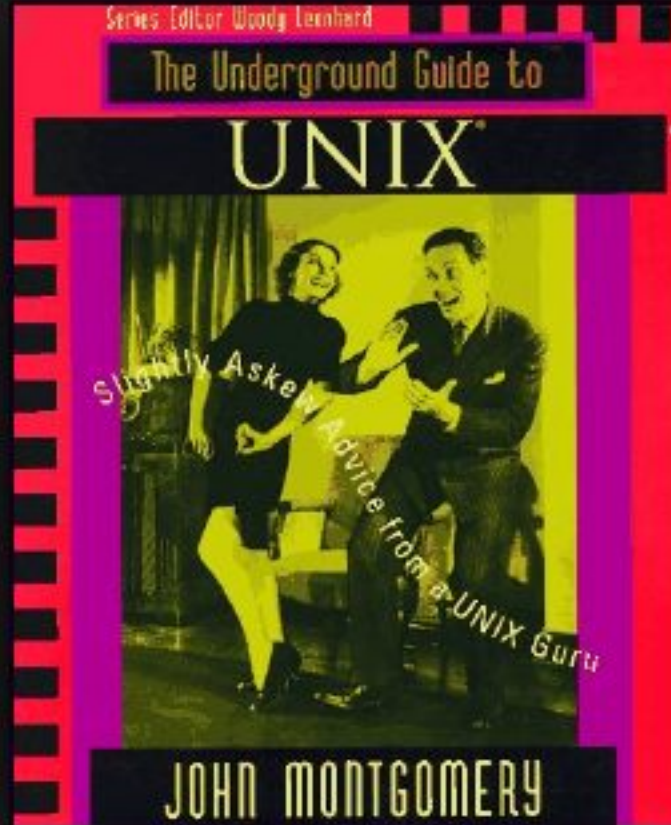
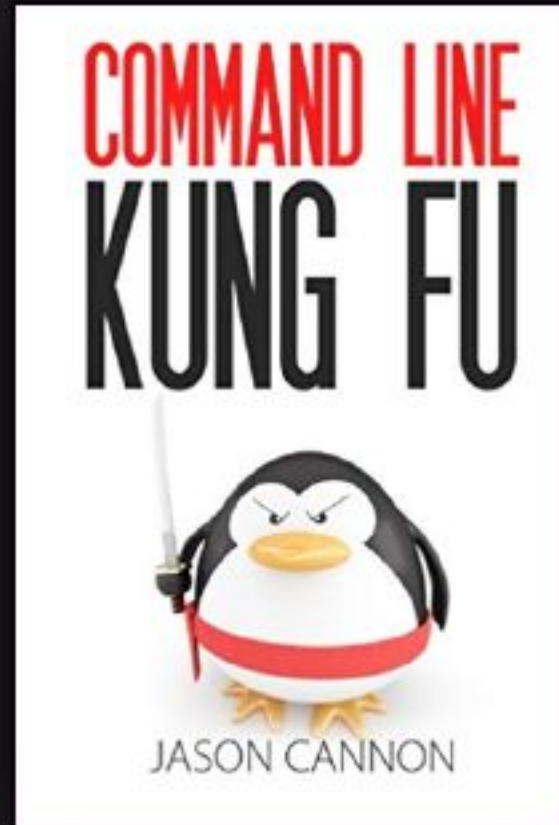


TERMINAL SESSION / DEMO

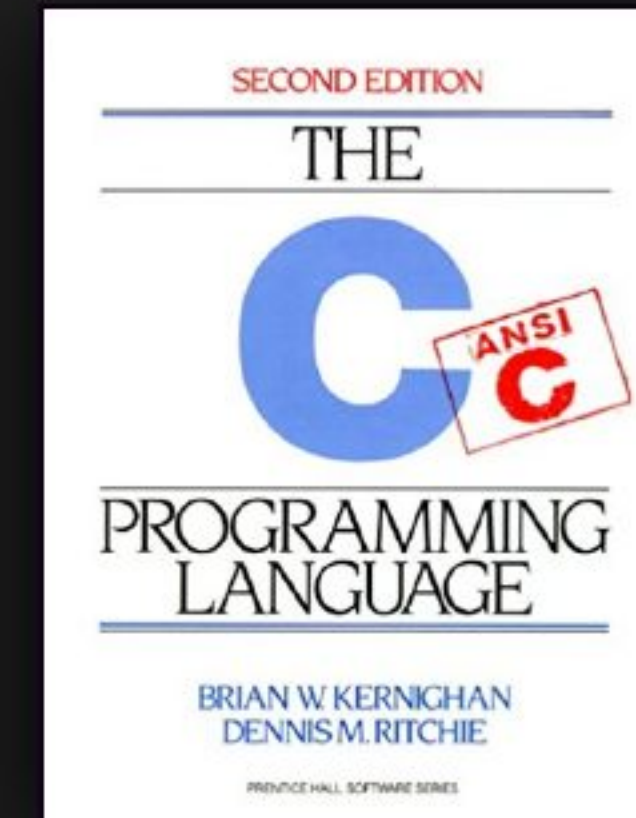
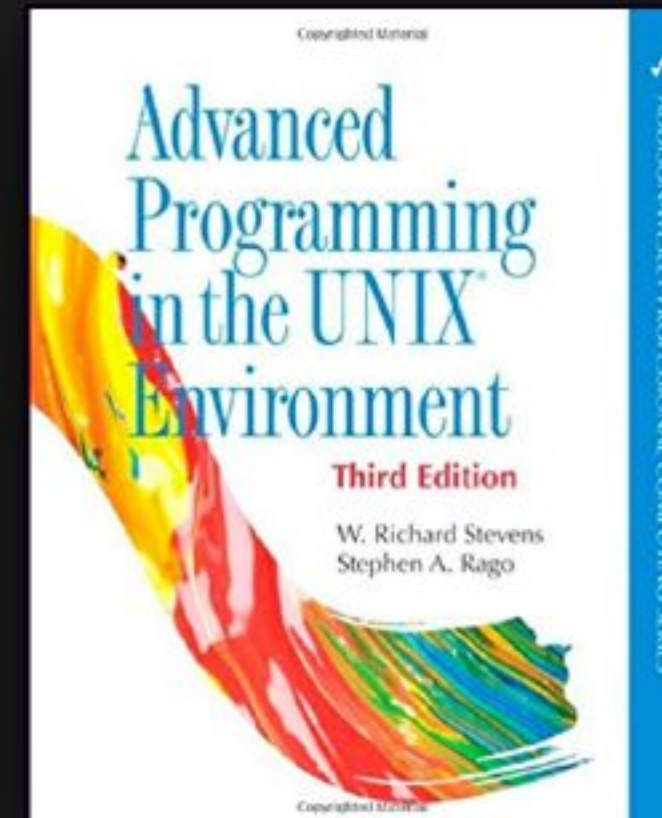
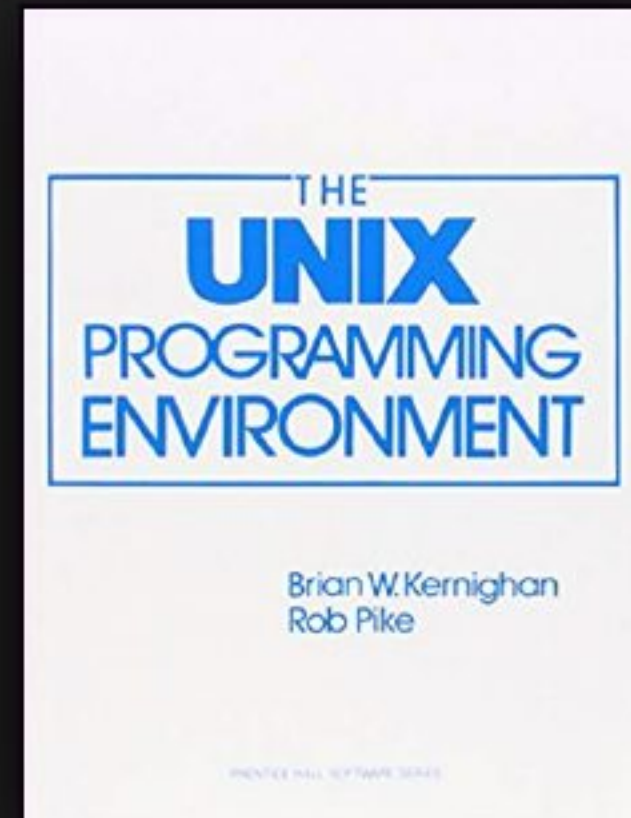
TERMINAL SESSION / DEMO

FURTHER READING

PRACTICAL AND SHORT



CLASSICS AND IN DEPTH



```
> h1 'Thanks for watching!' 'Hope you enjoyed it, and please, ' \  
> '— dont hesitate to ask questions!'
```

Thanks for watching!

Hope you enjoyed it, and please,
— dont hesitate to ask questions!

```
↗ develop {1} kg → ® 2.4.3 → ~/homebase/code/homebase3 ← kig@kiggie-poo 12:29  
> |
```



THANKS!

KONSTANTIN GREDESKOUL
github.com/kigster
twitter.com/kig