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

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

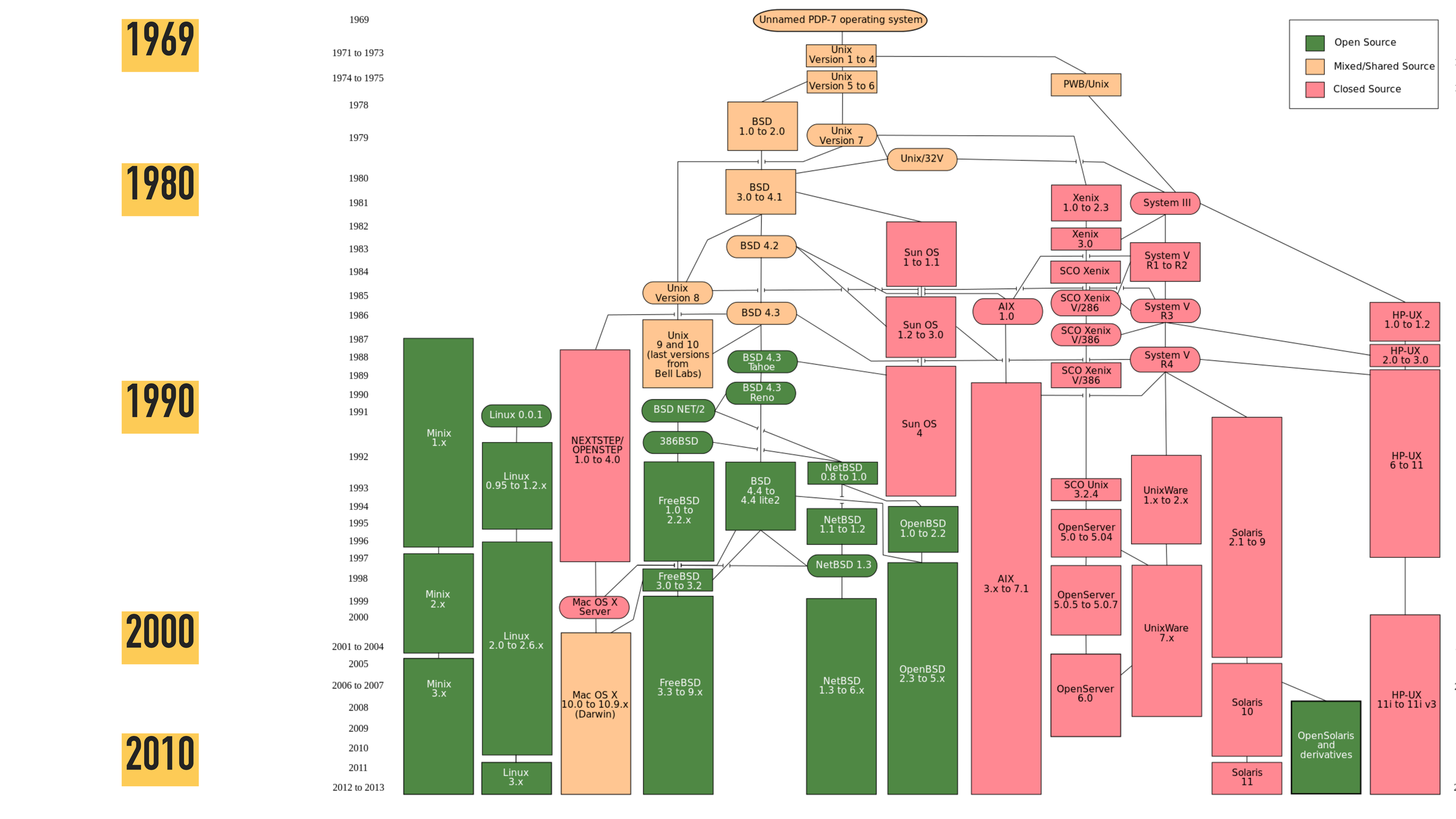
- Just imagine if I told you — “To play the next version of World of Warcraft you have create a file system, device drivers, scheduler, the kernel and the OS to run your game on, but after that — enjoy yourself”.
- Sounds insane, right?
- But that’s exactly what Ken Thompson and Dennis Richie decided 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
Open Source
Mixed/Shared Source
Closed Source
Unnamed PDP-7 operating system
Unix Version 1 to 4
Unix Version 5 to 6
PWB/Unix
BSD 1.0 to 2.0
Unix Version 7
Unix/32V
BSD 3.0 to 4.1
Xenix 1.0 to 2.3
System III
Xenix 3.0
BSD 4.2
Sun OS 1 to 1.1
System V R1 to R2
SCO Xenix
Unix Version 8
SCO Xenix V/286
AIX 1.0
System V R3
BSD 4.3
Sun OS 1.2 to 3.0
HP-UX 1.0 to 1.2
SCO Xenix V/386
HP-UX 2.0 to 3.0
Unix 9 and 10 (last versions from Bell Labs)
System V R4
BSD 4.3 Tahoe
BSD 4.3 Reno
SCO Xenix V/386
Minix 1.x
Linux 0.0.1
NEXTSTEP/ OPENSTEP 1.0 to 4.0
BSD NET/2
Sun OS 4
386BSD
HP-UX 6 to 11
NetBSD 0.8 to 1.0
SCO Unix 3.2.4
UnixWare 1.x to 2.x
Linux 0.95 to 1.2.x
FreeBSD 1.0 to 2.2.x
BSD 4.4 to 4.4 lite2
NetBSD 1.1 to 1.2
OpenBSD 1.0 to 2.2
OpenServer 5.0 to 5.04
Solaris 2.1 to 9
Minix 2.x
FreeBSD 3.0 to 3.2
NetBSD 1.3
AIX 3.x to 7.1
OpenServer 5.0.5 to 5.0.7
UnixWare 7.x
Linux 2.0 to 2.6.x
Mac OS X Server
OpenBSD 2.3 to 5.x
Minix 3.x
FreeBSD 3.3 to 9.x
NetBSD 1.3 to 6.x
OpenServer 6.0
Solaris 10
HP-UX 11i to 11i v3
Mac OS X 10.0 to 10.9.x (Darwin)
OpenSolaris and derivatives
Linux 3.x
Solaris 11

CORE CONCEPTS

THREE DISTINCT LEVELS

- **User Level:**

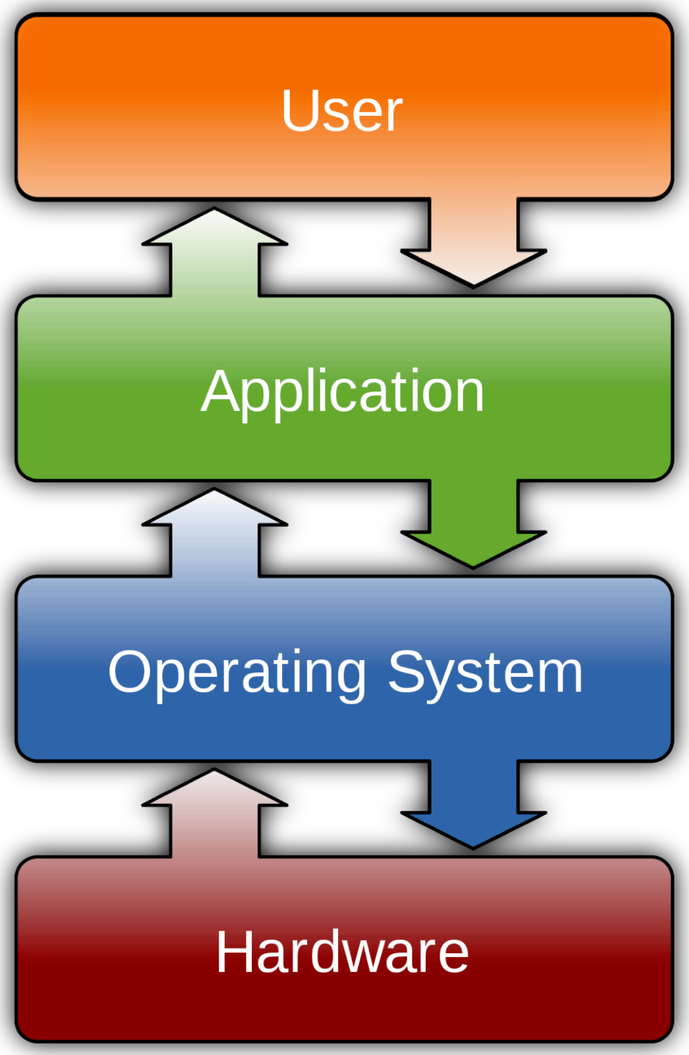
  Apps, shell, UI, libraries, higher level languages (ruby)

- **Kernel**

  Processes, scheduling, device drivers

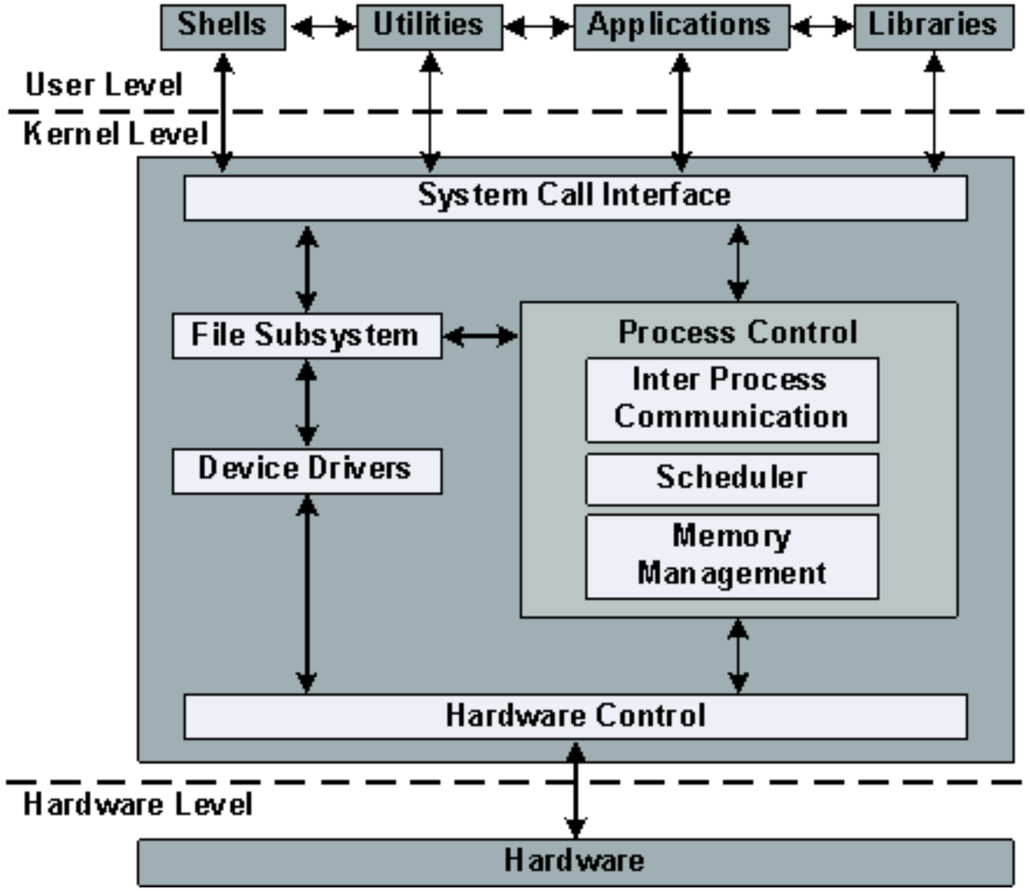
- **Hardware**

  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()
2. write()
3. fork()
4. ~~swap()~~
5. exit()
6. kill()
7. wait()

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

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...

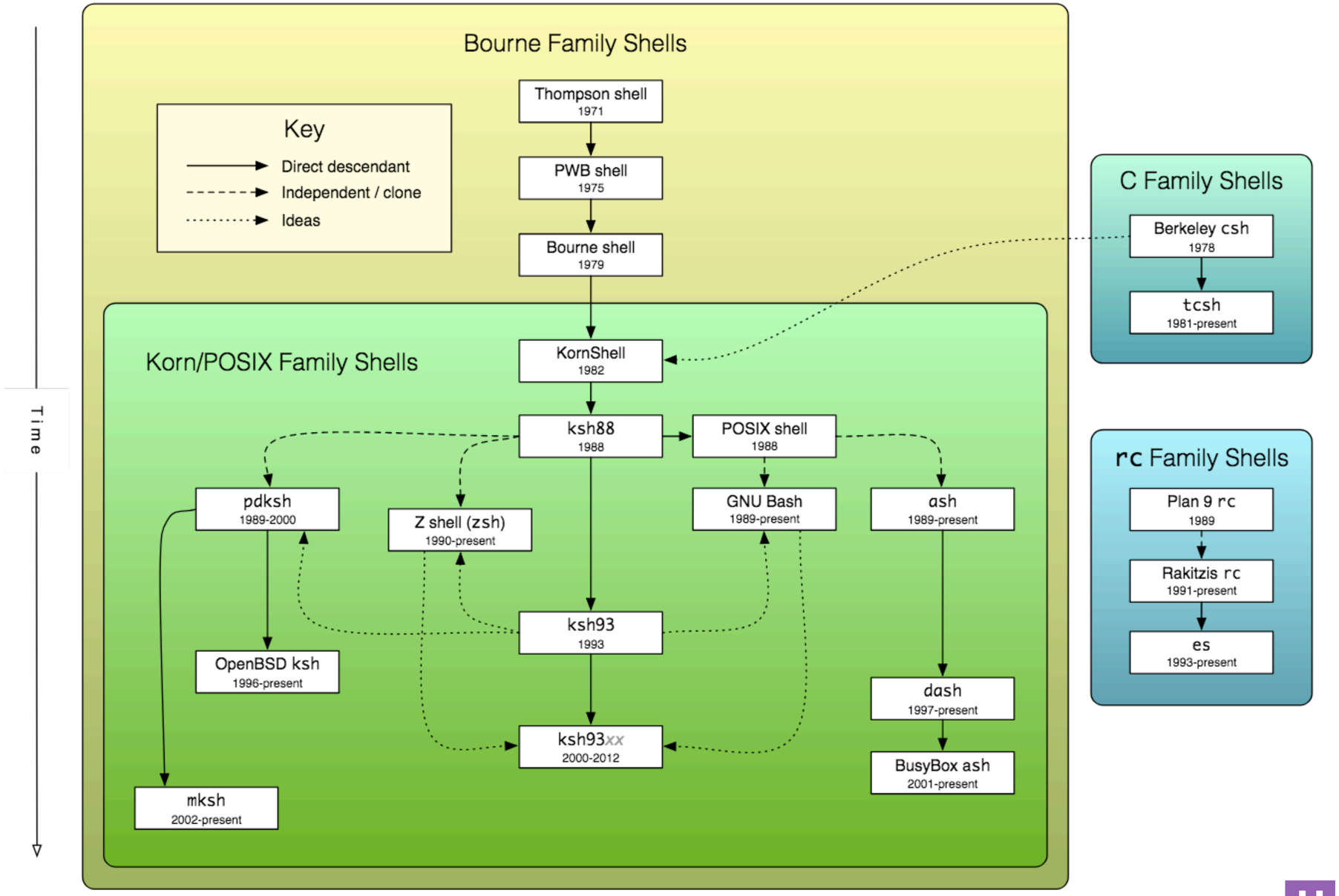
SHELL EXECUTES YOU

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 Effff Slash"



IN SOVIET RUSSIA, SHELL COMMITS SUICIDE

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`

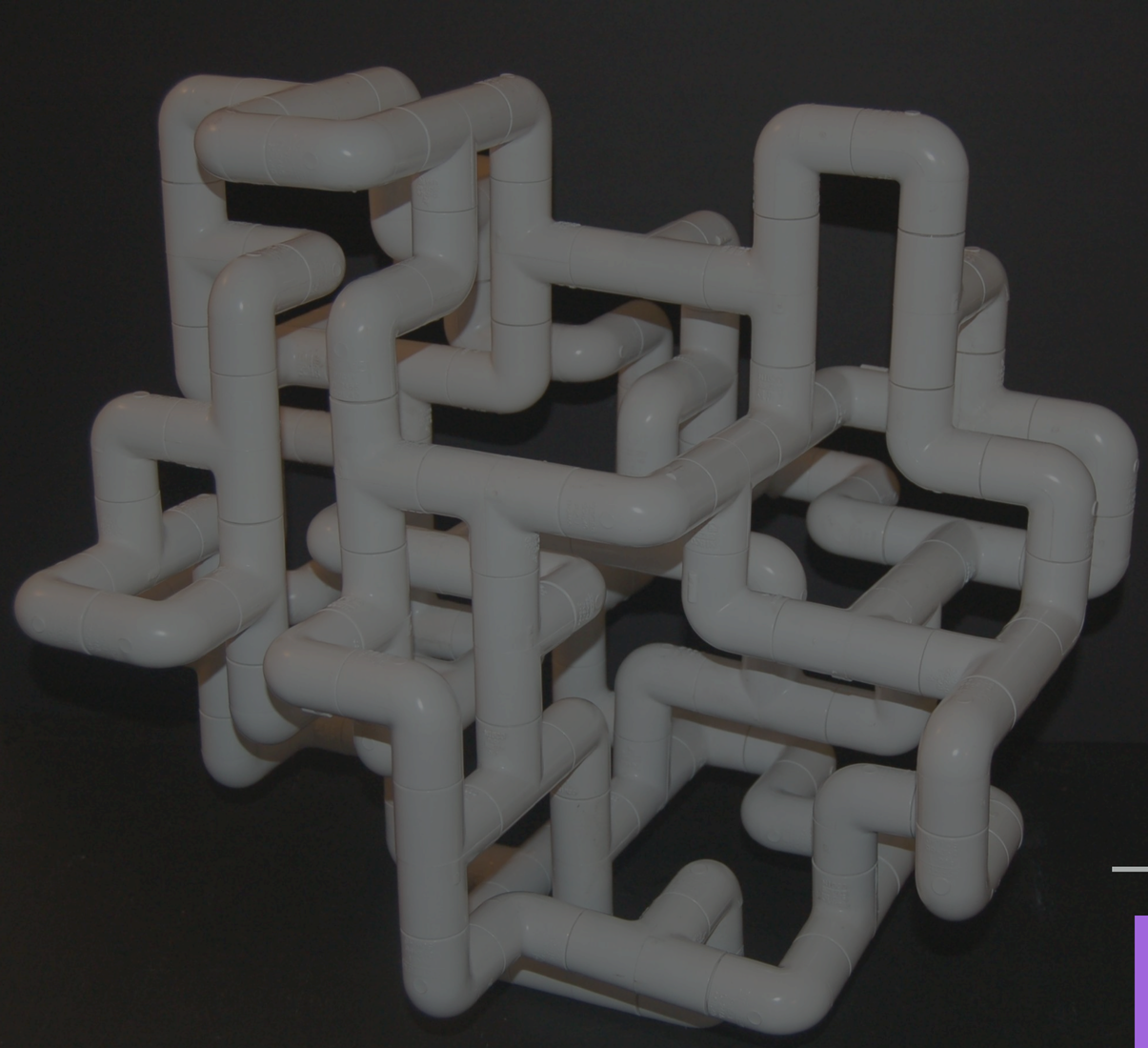
- Unfortunately, Mac OS X ships with a very old version of BASH, version 3
- This is why during app setup, we install the newer BASH (version 4) using HomeBrew

```
› 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
```



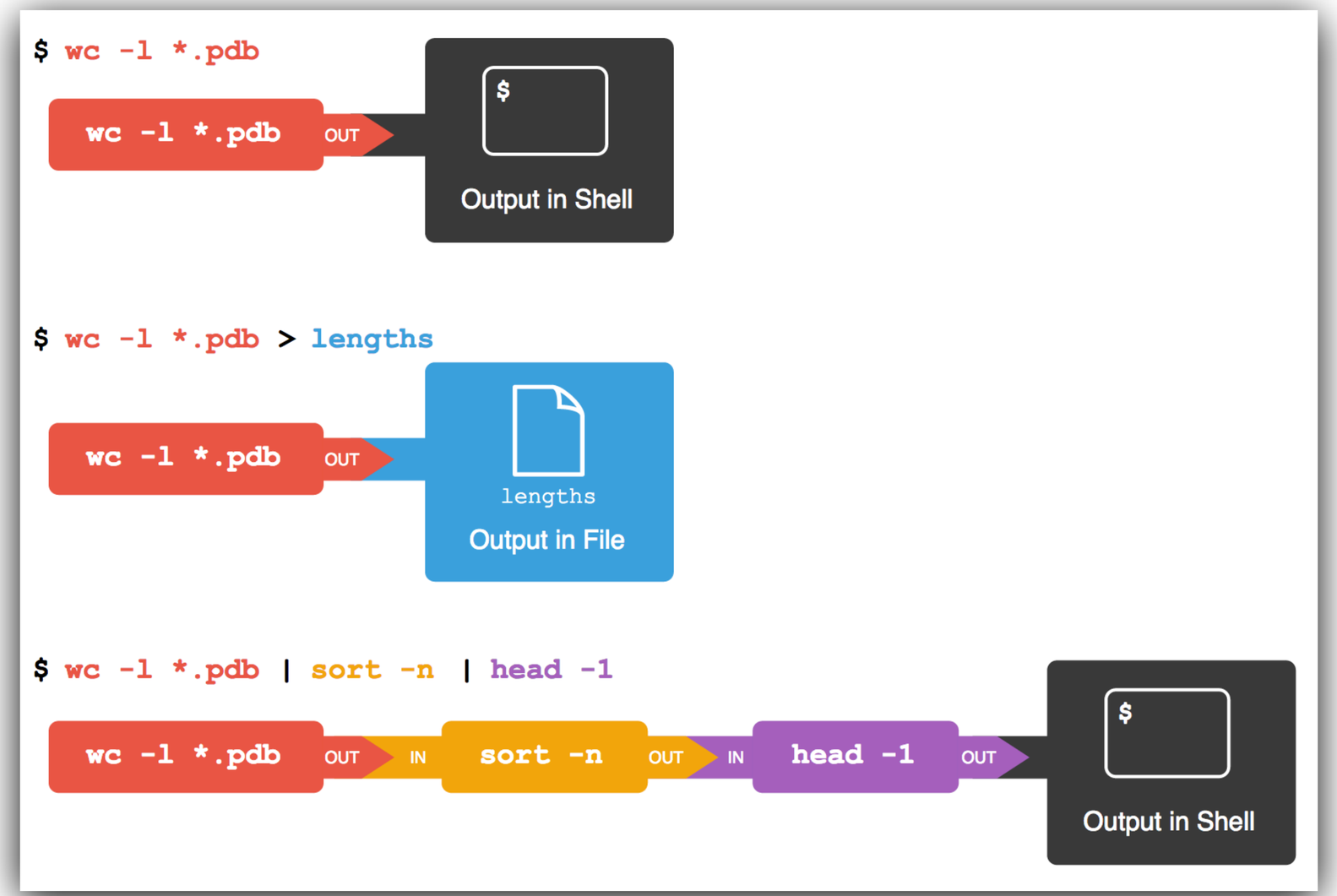
DON’T ROLL IT BY HAND, USE A PIPE!

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



WARNINGS: JEDI TRICKS MAY FOLLOW

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

Command	External	csh(1)	sh(1)
fc	No**	No	Yes
fg	No**	Yes	Yes
filetest	No	Yes	No
fi	No	No	Yes
for	No	No	Yes
foreach	No	Yes	No
getopts	No**	No	Yes
glob	No	Yes	No
goto	No	Yes	No
hash	No	No	Yes
hashstat	No	Yes	No
history	No	Yes	No
hup	No	Yes	No
if	No	Yes	Yes
jobid	No	No	Yes
jobs	No**	Yes	Yes
kill	Yes	Yes	No
limit	No	Yes	No
local	No	No	Yes
log	No	Yes	No
login	Yes	Yes	No
logout	No	Yes	No
ls-F	No	Yes	No
nice	Yes	Yes	No
nohup	Yes	Yes	No
notify	No	Yes	No
onintr	No	Yes	No
popd	No	Yes	No
printenv	Yes	Yes	No
pushd	No	Yes	No
pwd	Yes	No	Yes
read	No**	No	Yes
readonly	No	No	Yes
rehash	No	Yes	No
repeat	No	Yes	No
return	No	No	Yes

Command	External	csh(1)	sh(1)
set	No	Yes	Yes
setenv	No	Yes	No
settc	No	Yes	No
setty	No	Yes	No
setvar	No	No	Yes
shift	No	Yes	Yes
source	No	Yes	No
stop	No	Yes	No
suspend	No	Yes	No
switch	No	Yes	No
telltc	No	Yes	No
test	Yes	No	Yes
then	No	No	Yes
time	Yes	Yes	No
times	No	No	Yes
trap	No	No	Yes
true	Yes	No	Yes
type	No	No	Yes
ulimit	No	No	Yes
umask	No**	Yes	Yes
unalias	No**	Yes	Yes
uncomplete	No	Yes	No
unhash	No	Yes	No
unlimit	No	Yes	No
unset	No	Yes	Yes
unsetenv	No	Yes	No
until	No	No	Yes
wait	No**	Yes	Yes
where	No	Yes	No
which	Yes	Yes	No
while	No	Yes	Yes

COMMON UNIX SHELL COMMANDS

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
```

COMMON UNIX SHELL COMMANDS

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
```

COMMON UNIX SHELL COMMANDS

NETWORK

```
netstat
trace
ping
ssh
telnet
finger
ftp
```

```
sftp
curl
wget
scp
```

EXPLORING FILE SYSTEM

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

CHANGING PERMISSIONS

```
❯ 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.

`[dl-] rwx rwx rwx [@]`

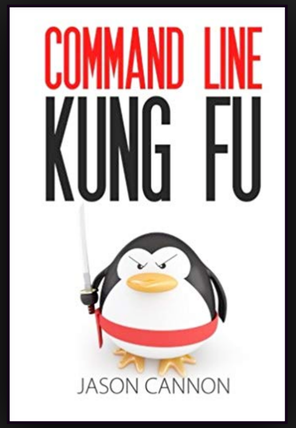
CHANGING PERMISSIONS

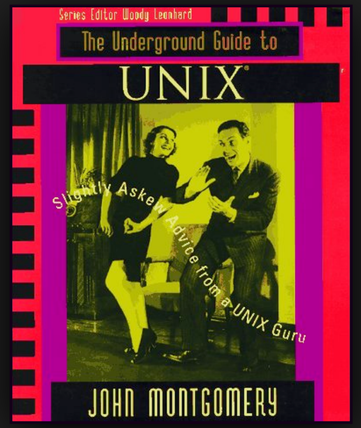
```
❯ ls -al
total 0
drwxr-xr-x   3 kig  staff   96 Aug 16 10:48 .
drwxr-xr-x@ 23 kig  staff  736 Aug 16 10:48 ..
-rw-r-----   1 kig  staff    0 Aug 16 10:48 README.md

❯ chmod o+rw,u+x README.md
❯ ls -al R*
-rwxr--rw-   1 kig  staff    0 Aug 16 10:48 README.md
```

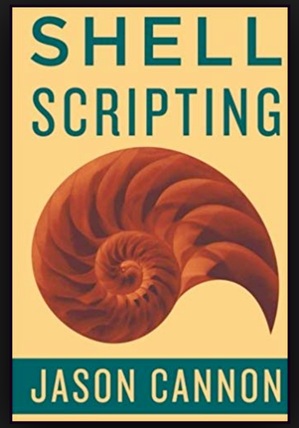
TERMINAL SESSION / DEMO

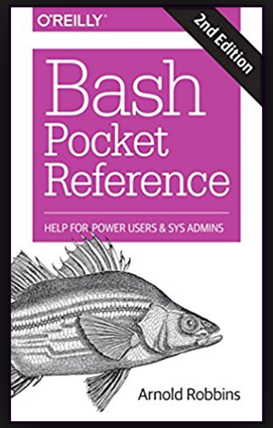
FURTHER READING

PRACTICAL AND SHORT

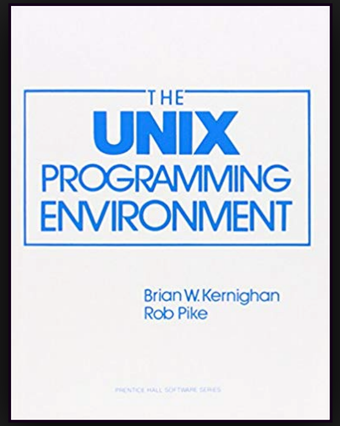


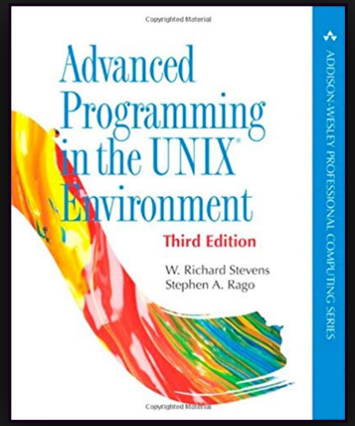


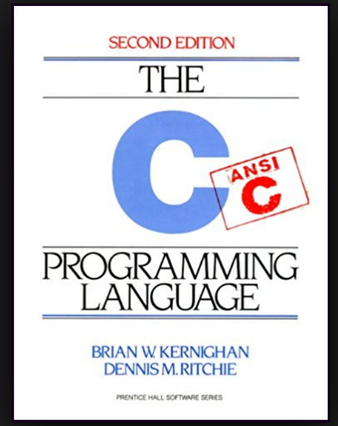




CLASSICS AND IN DEPTH







```
❯ 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