shell-fu

3 short talks, NYC*BUG, 2016.02.03

Isaac (.ike) Levy <ike@blackskyresearch.net>

disclaimer

The views expressed in this presentation are my own and do not represent the views of my current or former employers, nor do they represent the *BSD projects, or NYC*BUG as a whole.

No animals were harmed in the making of this presentation.

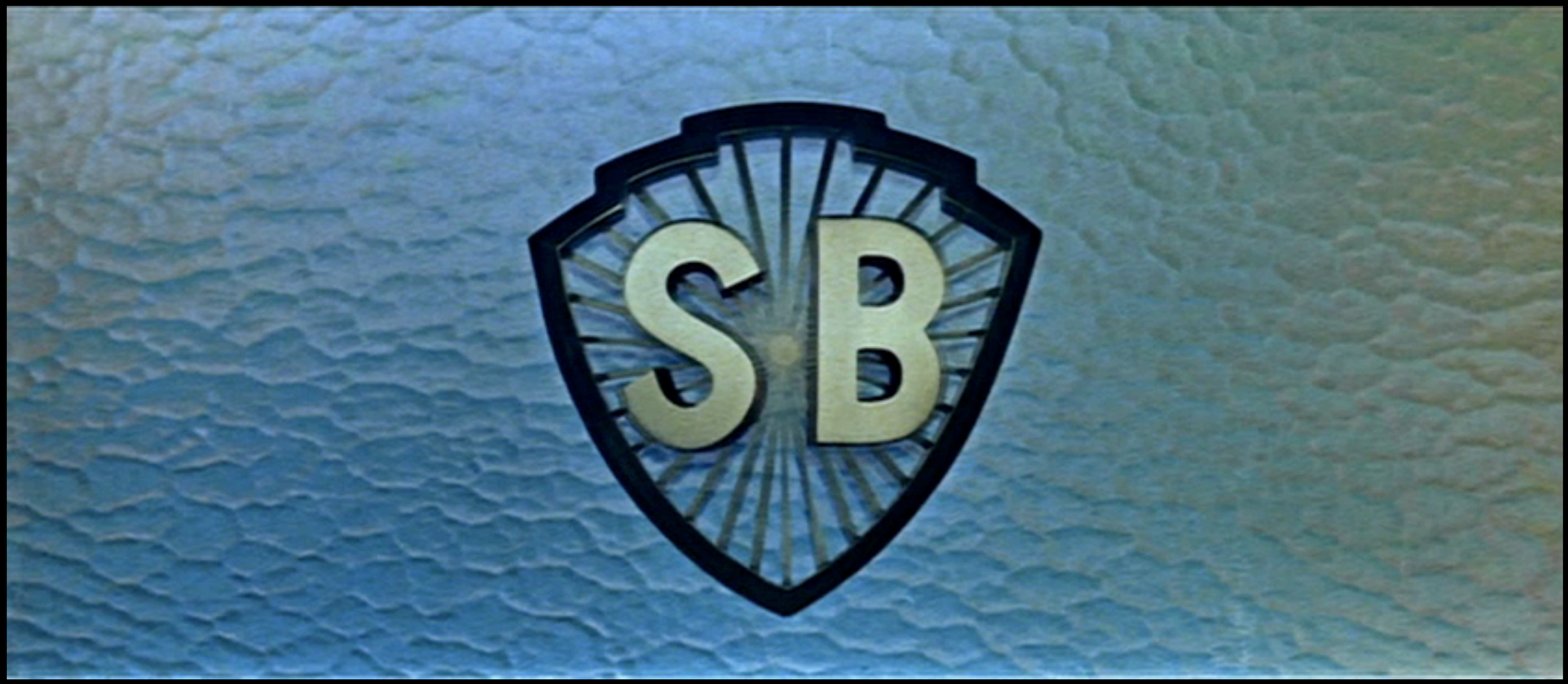
community sidenote

standing strong against
ongoing abuse and harassment

3 short talks

the 3 finger claw technique
using atomic filesystem operations
general shell-fu, (input and variable handling)

shell-fu



3 short talks, NYC*BUG, 2016.02.03

Isaac (.ike) Levy <ike@blackskyresearch.net>


UNIX
NULL
Foglio

ber

ike

bub

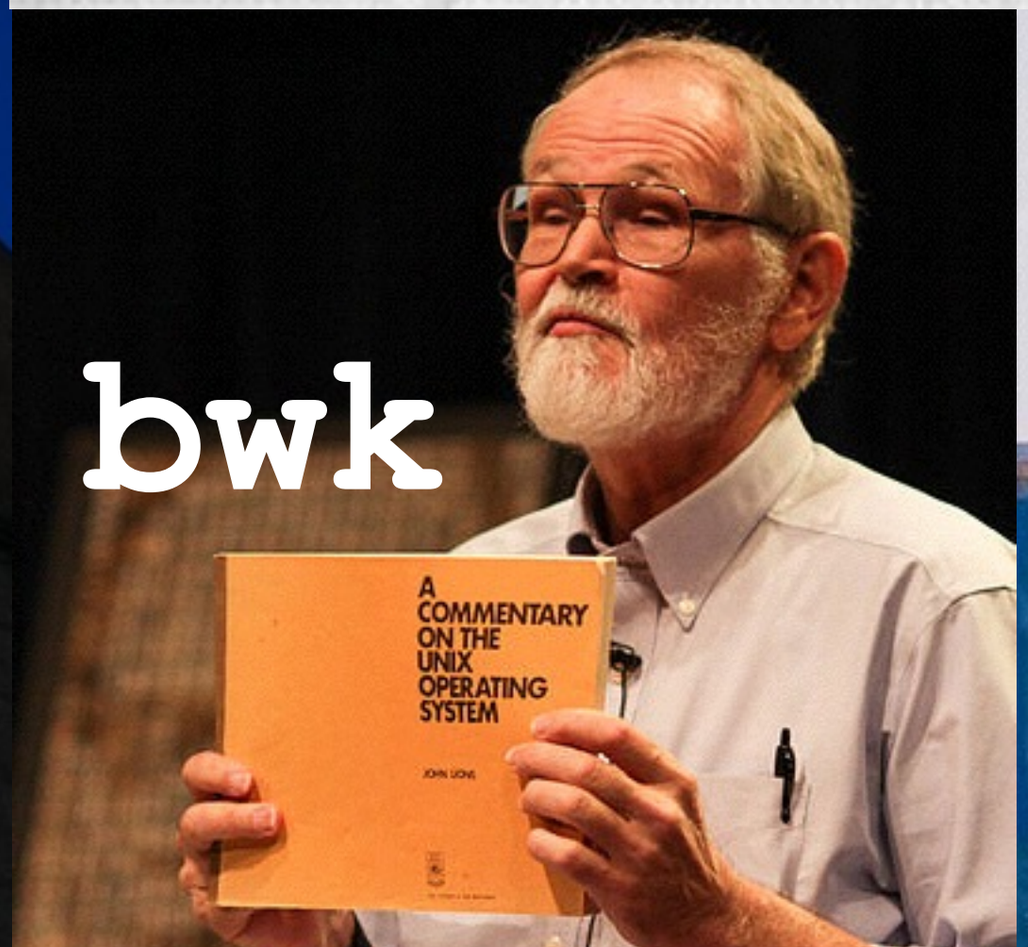
web

matt


djb


ken

dmr



bwk


doug

First with UNIX
(the first unix shell)

https://www.youtube.com/watch?v=tc4ROCJYbm0

AT&T Archives: The UNIX Operating System

ken

Original sh

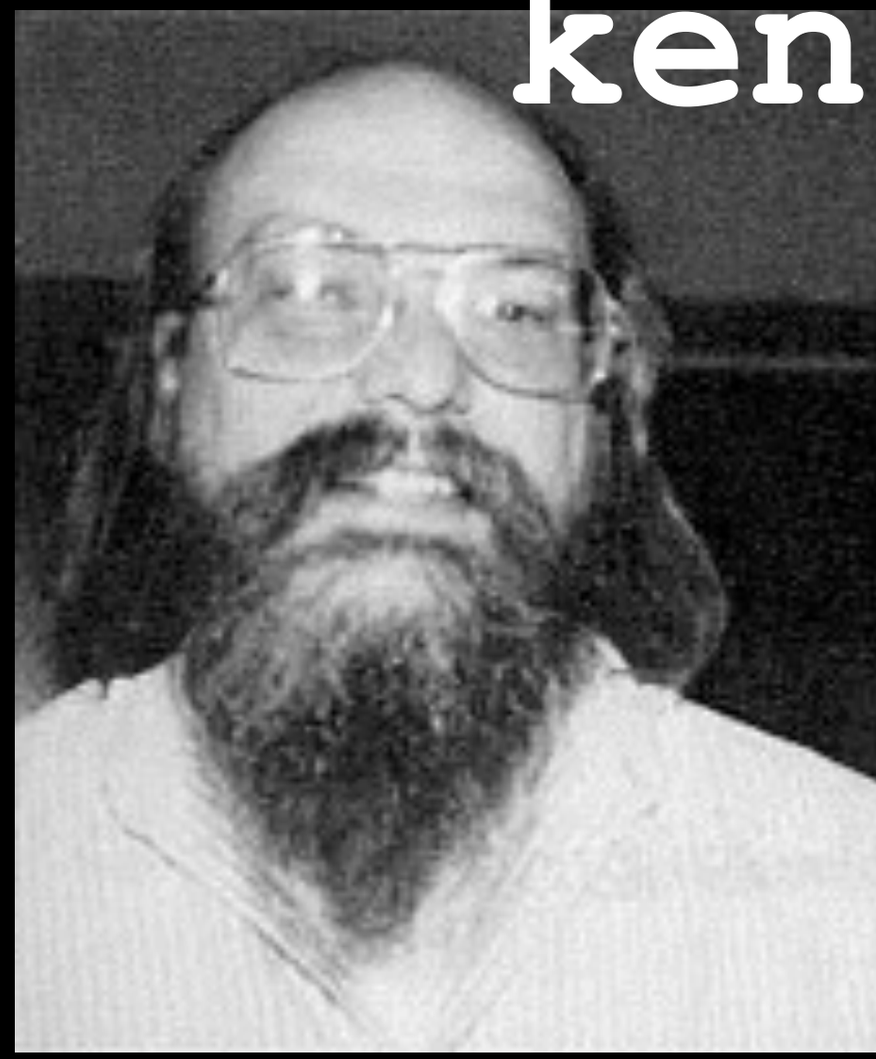
- Thompson Shell
  - Script recording with simple GOTO
  - /etc/glob for wild card characters
  - Shell scripts are not filters
  - Script was standard input
  - No control flow or variables ($1 - $9, $a - $z)
- The UNIX Time-sharing System - A Retrospective by Dennis Ritchie

SRB, NYC*BUG 2015.11.19
http://www.nycbug.org/event/10636/NYCBug.20151119.srb.pdf

https://www.bell-labs.com/usr/dmr/www/retro.pdf

1975, Ken Thompson was in Berkeley for a year

"In 1975, Thompson took a sabbatical from Bell Labs and went to his alma mater, UC Berkeley. **There, he helped to install Version 6 Unix on a PDP-11/70.** Unix at Berkeley would later become maintained as its own system, known as the Berkeley Software Distribution (BSD)."



ken

mckusick

joy

beastie

by 1982, TCP/IP, UFS, etc...


FreeBSD
FOUNDATION
2012

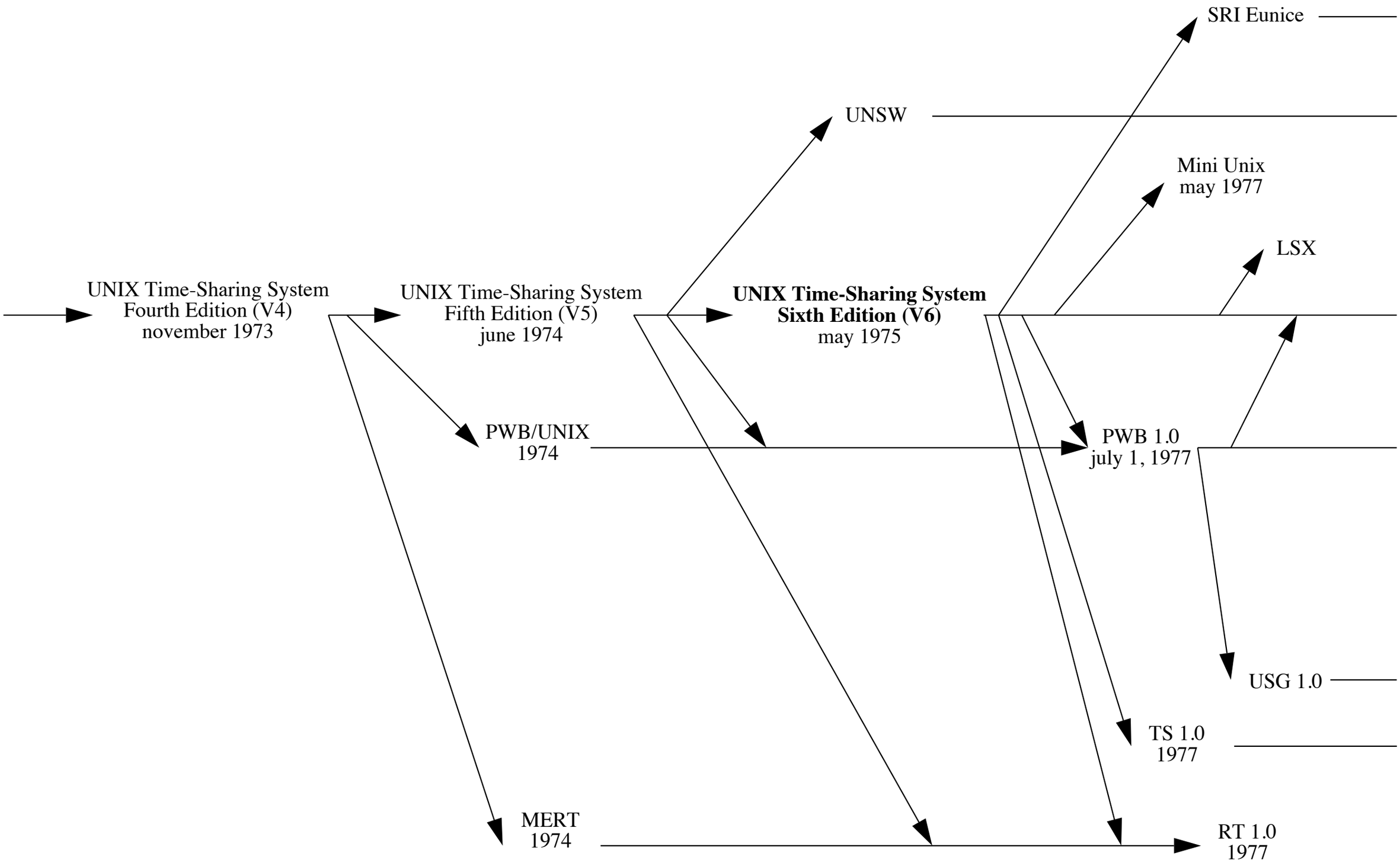
1976, Ken Thompson returned to Murray Hill

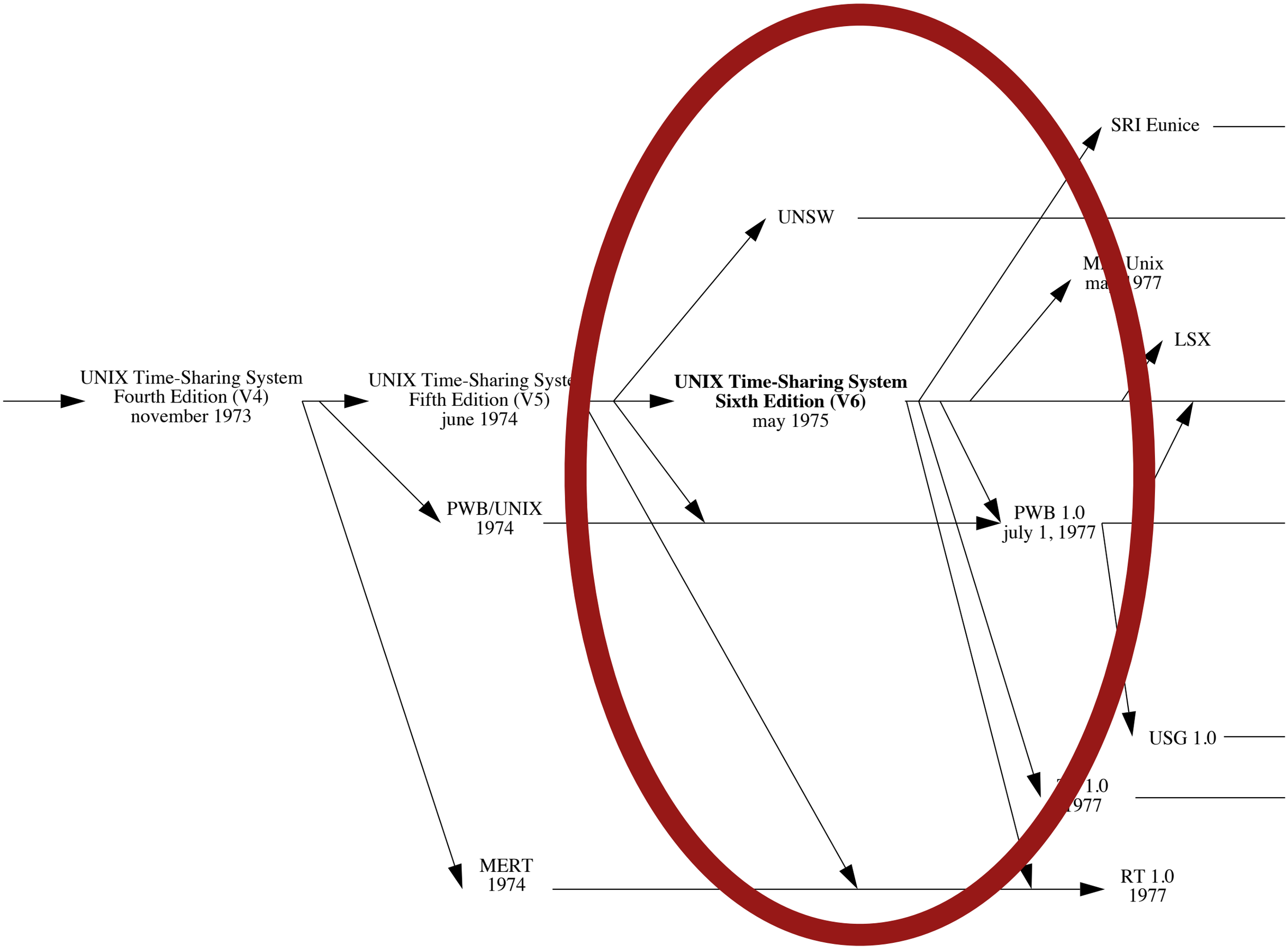
~ 1976, the bourne shell

not the right bourne

srb

http://www.levenez.com/unix





~ 1976, the bourne shell

sh
Thompson Shell
Ken Thompson
1969

sh
PWB Mashey Shell
John Mashey
1975

tcsh
T C shell
Bill Joy
1978

reaction

reaction

sh
bourne shell
Steven Bourne
1976

ash
Almquist
Kenneth Almqu
~1989

bash
Gnu Bourne A
Brian Fo
1989

ksh
Korn shell
David Korn
1983

ksh88
Korn shell
POSIX
1988

sh
WB Mashey Shell
John Mashey
1975

tcsh
T C shell
Bill Joy
1978

reaction

sh
bourne shell
Steven Bourne
1976

ash
Almquist
Kenneth Almquist
~1989

dash
Debian Almq
Debian Lin
2011

ksh
Korn shell
David Korn
1983

bash
Gnu Bourne Again sh
Brian Fox
1989

sh
FreeBSD sh
Debian Linux
~1994

ksh88
Korn shell
POSIX
1988

sh
POSIX Shell
POSIX
1992

pk
Public Do

sh
bourne shell
Steven Bourne
1976

ash
Almquist
Kenneth Almquist
~1989

dash
Debian Almquist
Debian Linux
2011

ksh
Korn shell
David Korn
1983

bash
Gnu Bourne Again sh
Brian Fox
1989

sh
FreeBSD sh
Debian Linux
~1994

ksh88
Korn shell
POSIX
1988

sh
POSIX Shell
POSIX
1992

pkdsh
Public Domain Korn
OpenBSD
1999

ksh88
Korn shell
POSIX
1988

ksh93
Korn shell
Linux Distros
1994?

rc
Plan9, UNIX v10
Tom Duff
1989

zsh
Z Shell
Paul Falstad
1990

wacky observations...

sh
bourne shell
Steven Bourne
1976

ash
Almquist
Kenneth Almquist
~1989

Debian Almq
Debian Linu
2011

ksh
Korn shell
David Korn
1983

bash
Gnu Bourne Again sh
Brian Fox
1989

sh
FreeBSD sh
Debian Linux
~1994

srb!

ksh88
Korn shell
POSIX
1988

sh
POSIX Shell
POSIX
1992

pkc
Public Dor
Oper
19

ksh88
Korn shell
POSIX
1988

ksh93
Korn sh
Linux Dis
1994

Algol68
Language
Stephen Bourne
1968

rc
Plan9, UNIX v10
Tom Duff
1989

ABC Language
Z Shell
Paul Falstad
1990

zsh
Z Shell
Paul Falstad
1990

Python
Language
Guido van Rossum
1989

sh
Thompson Shell
Ken Thompson
1969

sh
PWB Mashey Shell
John Mashey
1975

tcsh
T C shell
Bill Joy
1978

reaction

reaction

sh
bourne shell
Steven Bourne
1976

ash
Almquist
Kenneth Almquist
~1989

dash
Debian Almquist
Debian Linux
2011

ksh
Korn shell
David Korn
1983

bash
Gnu Bourne Again sh
Brian Fox
1989

sh
FreeBSD sh
Debian Linux
~1994

srb!

ksh88
Korn shell
POSIX
1988

sh
POSIX Shell
POSIX
1992

pkdsh
Public Domain Korn
OpenBSD
1999

ksh88
Korn shell
POSIX
1988

ksh93
Korn shell
Linux Distros
1994?

Algol68
Language
Stephen Bourne
1968

rc
Plan9, UNIX v10
Tom Duff
1989

ABC Language
Z Shell
Paul Falstad
1990

zsh
Z Shell
Paul Falstad
1990

Python
Language
Guido van Rossum
1989

sh
Thompson Shell
Ken Thompson
1969

sh
PWB Mashey Shell
John Mashey
1975

tcsh
T C shell
Bill Joy
1978

reaction

reaction

sh
bourne shell
Steven Bourne
1976

ash
Almquist
Kenneth Almquist
~1989

dash
Debian Almquist
Debian Linux
2011

ksh
Korn shell
David Korn
1983

bash
Gnu Bourne Again sh
Brian Fox
1989

sh
FreeBSD sh
Debian Linux
~1994

srb!

ksh88
Korn shell
POSIX
1988

sh
POSIX Shell
POSIX
1992

pkdsh
Public Domain Korn
OpenBSD
1999

ksh88
Korn shell
POSIX
1988

ksh93
Korn shell
Linux Distros
1994?

Algol68
Language
Stephen Bourne
1968

rc
Plan9, UNIX v10
Tom Duff
1989

ABC Language
Z Shell
Paul Falstad
1990

zsh
Z Shell
Paul Falstad
1990

Python
Language
Guido van Rossum
1989

people start agreeing on data type primitives

sh
Thompson Shell
Ken Thompson
1969

sh
PWB Mashey Shell
John Mashey
1975

tcsh
T C shell
Bill Joy
1978

reaction

reaction

bourneish

sh
bourne shell
Steven Bourne
1976

ash
Almquist
Kenneth Almquist
~1989

dash
Debian Almquist
Debian Linux
2011

ksh
Korn shell
David Korn
1983

bash
Gnu Bourne Again sh
Brian Fox
1989

sh
FreeBSD sh
Debian Linux
~1994

srb!

ksh88
Korn shell
POSIX
1988

sh
POSIX Shell
POSIX
1992

pkdsh
Public Domain Korn
OpenBSD
1999

ksh88
Korn shell
POSIX
1988

ksh93
Korn shell
Linux Distros
1994?

Algol68
Language
Stephen Bourne
1968

rc
Plan9, UNIX v10
Tom Duff
1989

ABC Language
Z Shell
Paul Falstad
1990

zsh
Z Shell
Paul Falstad
1990

Python
Language
Guido van Rossum
1989

people start agreeing on data type primitives

sh
bourne shell
Steven Bourne
1976

ash
Almquist
Kenneth Almquist
~1989

dash
Debian Almquist
Debian Linux
2011

ksh
Korn shell
David Korn
1983

bash
Gnu Bourne Again sh
Brian Fox
1989

sh
FreeBSD sh
Debian Linux
~1994

ksh88
Korn shell
POSIX
1988

sh
POSIX Shell
POSIX
1992

pkdsh
Public Domain Korn
OpenBSD
1999

ksh88
Korn shell
POSIX
1988

ksh93
Korn shell
Linux Distros
1994?

rc
Plan9, UNIX v10
Tom Duff
1989

zsh
Z Shell
Paul Falstad
1990

portability

3 short talks

the 3 finger claw technique
using atomic filesystem operations
general shell-fu, (input and variable handling)

the 3 finger claw technique

existential functions

Pai Mei

White Lotus Clan, **Shaolin** Legendary 5 Elders, Kill Bill, Wu Tang Clan, etc...

```
yell() { echo "$0: $*" >&2; }
die() { yell "$*"; exit 111; }
try() { "$@" || die "cannot $*"; }
```

```
shout() { echo "$0: $*" >&2; }
barf() { shout "$*"; exit 100; }
safe() { "$@" || barf "cannot $*"; }
```

original

```
yell() { echo "$0: $*" >&2; }
die() { yell "$*"; exit 111; }
try() { "$@" || die "cannot $*"; }
```

```
#!/bin/sh

yell() { echo "$0: $*" >&2; }
die() { yell "$*"; exit 111; }
try() { "$@" || die "cannot $*"; }

# using it
try cd /some/place
try tar xzvfp /elsewhere/stuff.tbz

true
```

the 3 finger claw technique, 3 short functions

```
#!/bin/sh
cd /some/place
tar xzvfp /elsewhere/stuff.tbz
```

```
#!/bin/sh
cd /some/place
tar xzvfp /elsewhere/stuff.tbz
```

(a terrible program, stick that in cron?!)

```
#!/bin/sh

yell() { echo "$0: $*" >&2; }
die() { yell "$*"; exit 111; }
try() { "$@" || die "cannot $*"; }

# using it
try cd /some/place
try tar xzvfp /elsewhere/stuff.tbz

true
```

```
#!/bin/sh

yell() { echo "$0: $*" >&2; }
die() { yell "$*"; exit 111; }
try() { "$@" || die "cannot $*"; }

# using it
try cd /some/place
try tar xzvfp /elsewhere/stuff.tbz

true
```

```
#!/bin/sh

yell() { echo "$0: $*" >&2; }
die() { yell "$*"; exit 111; }
try() { "$@" || die "cannot $*"; }

# using it
try cd /some/place
try tar xzvfp /elsewhere/stuff.tbz

true
```

```
#!/bin/sh

yell() { echo "$0: $*" >&2; }
die() { yell "$*"; exit 111; }
try() { "$@" || die "cannot $*"; }

# using it
try cd /some/place
try tar xzvfp /elsewhere/stuff.tbz

true
```

```
#!/bin/sh

yell() { echo "$0: $*" >&2; }
die() { yell "$*"; exit 111; }
try() { "$@" || die "cannot $*"; }

# using it
try cd /some/place
try tar xzvfp /elsewhere/stuff.tbz

true
```

```
#!/bin/sh

yell() { echo "$0: $*" >&2; }
die() { yell "$*"; exit 111; }
try() { "$@" || die "cannot $*"; }

# using it
try cd /some/place
try tar xzvfp /elsewhere/stuff.tbz

true
```

```
#!/bin/sh

yell() { echo "$0: $*" >&2; }
die() { yell "$*"; exit 111; }
try() { "$@" || die "cannot $*"; }

# using it
try cd /some/place
try tar xzvfp /elsewhere/stuff.tbz

true
```

```
#!/bin/sh

yell() { echo "$0: $*" >&2; }
die() { yell "$*"; exit 111; }
try() { "$@" || die "cannot $*"; }

# using it
try cd /some/place
try tar xzvfp /elsewhere/stuff.tbz

true
```

```
#!/bin/sh

yell() { echo "$0: $*" >&2; }
die() { yell "$*"; exit 111; }
try() { "$@" || die "cannot $*"; }

# using it
try cd /some/place
try tar xzvfp /elsewhere/stuff.tbz

true
```

```
#!/bin/sh

yell() { echo "$0: $*" >&2; }
die() { yell "$*"; exit 111; }
try() { "$@" || die "cannot $*"; }

# using it
try cd /some/place
try tar xzvfp /elsewhere/stuff.tbz

true
```

```
#!/bin/sh

yell() { echo "$0: $*" >&2; }
die() { yell "$*"; exit 111; }
try() { "$@" || die "cannot $*"; }

# using it
try cd /some/place
try tar xzvfp /elsewhere/stuff.tbz

true
```

```
#!/bin/sh

yell() { echo "$0: $*" >&2; }
die() { yell "$*"; exit 111; }
try() { "$@" || die "cannot $*"; }

# using it
try cd /some/place
try tar xzvfp /elsewhere/stuff.tbz

true
```

```
#!/bin/sh

yell() { echo "$0: $*" >&2; }
die() { yell "$*"; exit 111; }
try() { "$@" || die "cannot $*"; }

# using it
try cd /some/place
try tar xzvfp /elsewhere/stuff.tbz

true
```

```
#!/bin/sh

yell() { echo "$0: $*" >&2; }
die() { yell "$*"; exit 111; }
try() { "$@" || die "cannot $*"; }

# using it
try cd /some/place
tar xzvfp /elsewhere/stuff.tbz

true
```

```
#!/bin/sh

yell() { echo "$0: $*" >&2; }
die() { yell "$*"; exit 111; }
try() { "$@" || die "cannot $*"; }

# using it
try cd /some/place
tar xzvfp /elsewhere/stuff.tbz

exit 0
```

why not?

`set -e`

`unset -e`

(errexit, POSIX, Exit the shell when any command exits with nonzero status)

```
yell() { echo "$0: $*" >&2; }
die() { yell "$*"; exit 111; }
try() { "$@" || die "cannot $*"; }
```

sh
bourne shell
Steven Bourne
1976

ash
Almquist
Kenneth Almquist
~1989

dash
Debian Almquist
Debian Linux
2011

ksh
Korn shell
David Korn
1983

bash
Gnu Bourne Again sh
Brian Fox
1989

sh
FreeBSD sh
Debian Linux
~1994

ksh88
Korn shell
POSIX
1988

sh
POSIX Shell
POSIX
1992

pkdsh
Public Domain Korn
OpenBSD
1999

ksh88
Korn shell
POSIX
1988

ksh93
Korn shell
Linux Distros
1994?

rc
Plan9, UNIX v10
Tom Duff
1989

zsh
Z Shell
Paul Falstad
1990

portability

http://blackskyresearch.net/

try.sh.txt

Better, Beat, Break it?

<talk@lists.nycbug.org>

using atomic filesystem operations

Do or do not, there is no try.


Do or do not, there is no try.

mv(1) or cp(1)

Which is atomic?

every file has 3 parts

(and everything UNIX is a file)

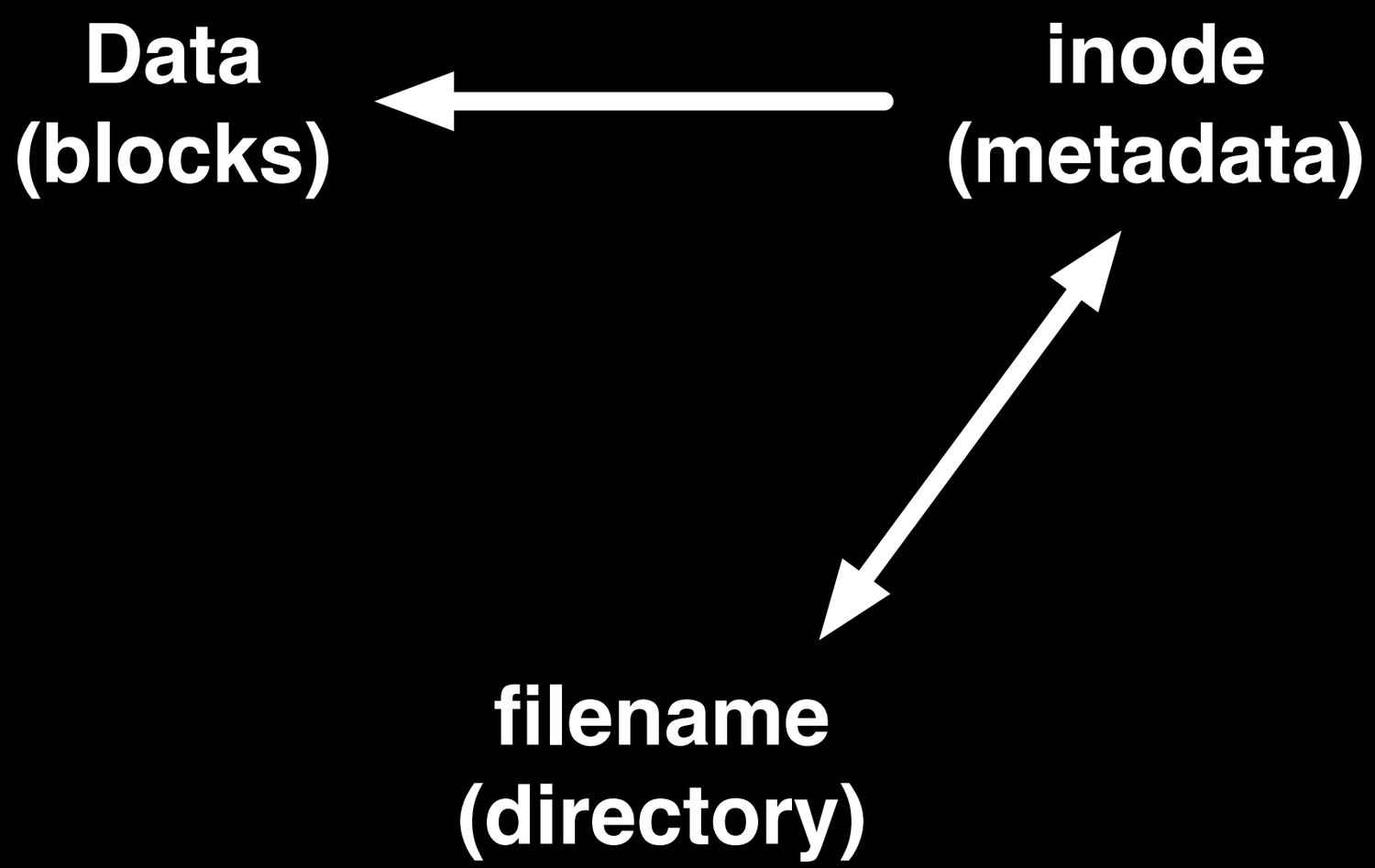
Data
(blocks)

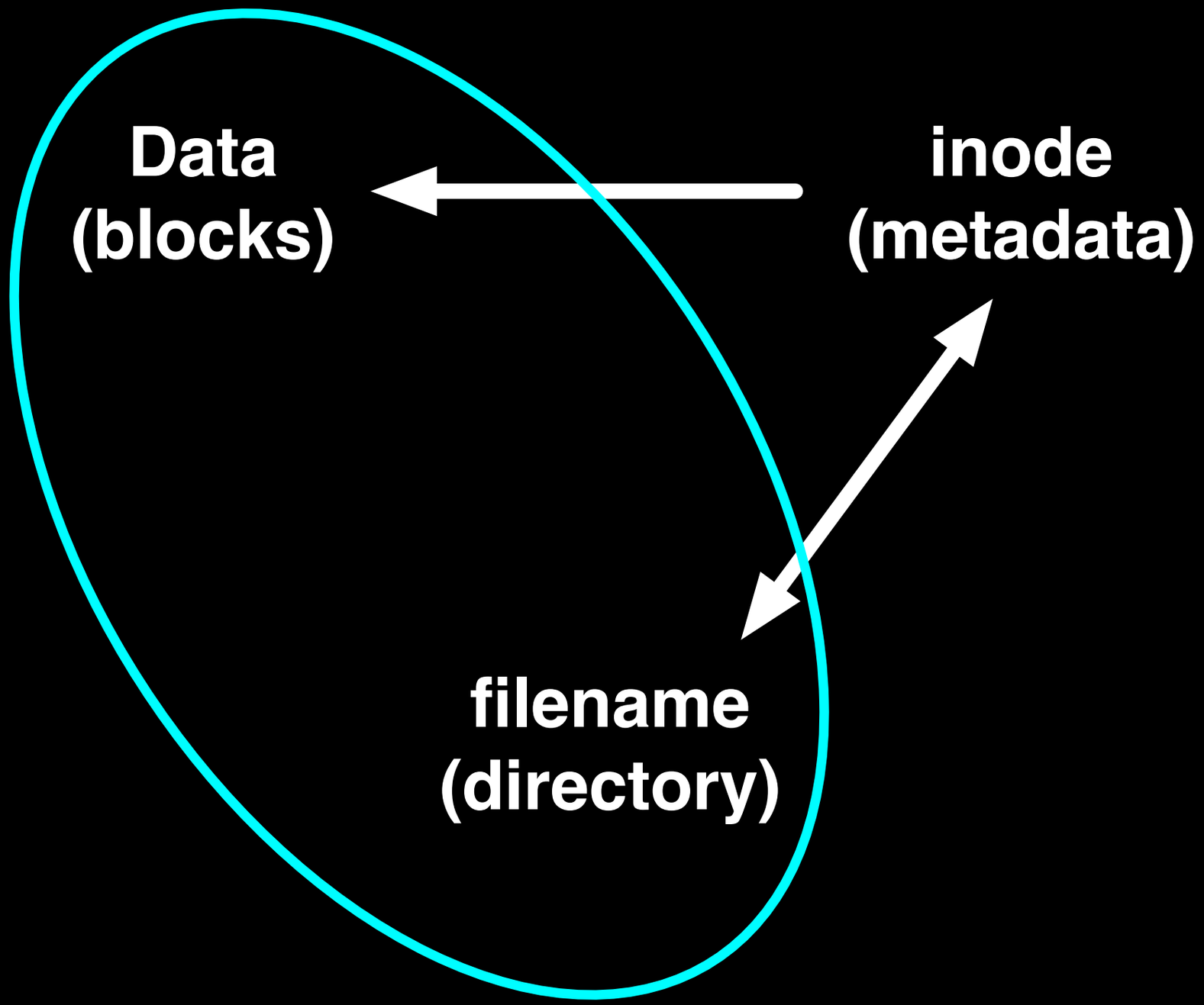
Data
(blocks)

inode
(metadata)


Data
(blocks)
inode
(metadata)

filename?


Data
(blocks)
inode
(metadata)
filename
(directory)


Data
(blocks)
inode
(metadata)
filename
(directory)

/anydir/

file.txt

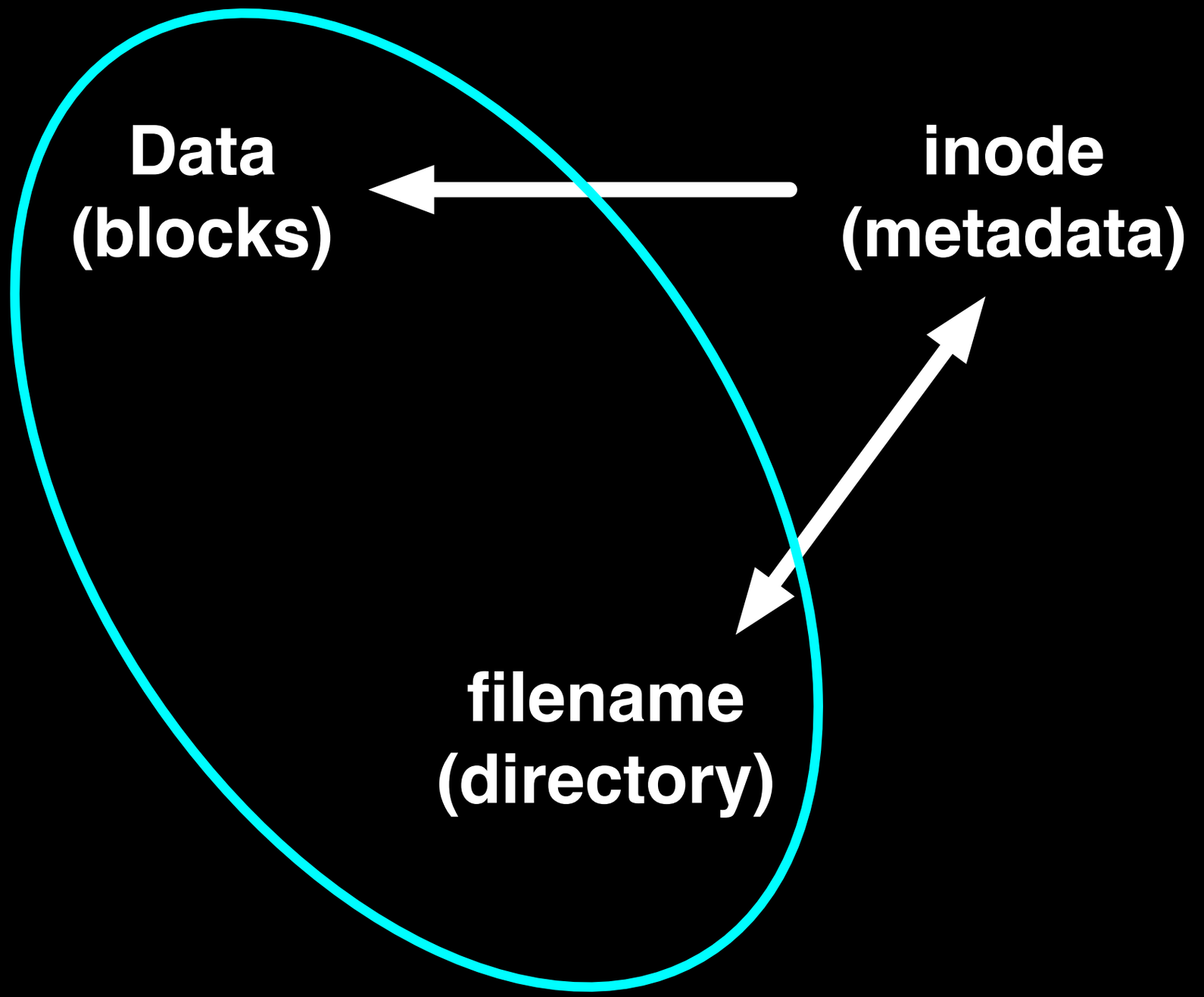
execu.sh

link

symlink

/dev/device

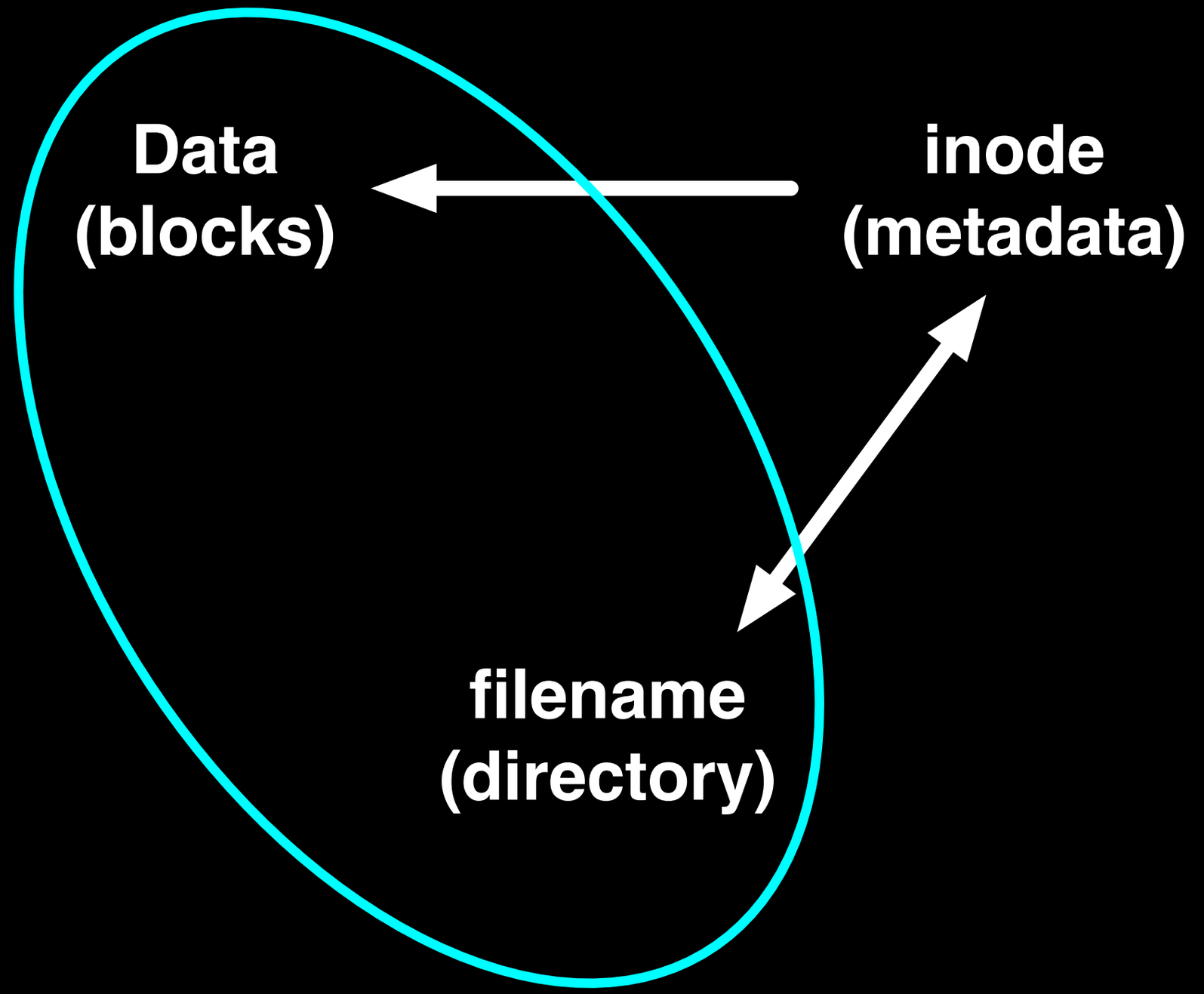
kernel



/anydir/

/anydir/..

/anydir/.



mv(1)

cp(1)

mv(1)

cp(1)

1) create new filename entry in new parent dir

mv(1)

cp(1)

1) create new filename entry in new parent dir

2) update link to filename in inode

mv(1) cp(1)

1) create new filename entry in new parent dir

2) update link to filename in inode

3) remove old filename entry from old parent dir

mv(1)

1) create new filename entry in new parent dir

2) update link to filename in inode

3) remove old filename entry from old parent dir

cp(1)

mv(1)

1) create new filename entry in new parent dir

2) update link to filename in inode

3) remove old filename entry from old parent dir

cp(1)

1) create new filename entry in new parent dir

mv(1)

1) create new filename entry in new parent dir

2) update link to filename in inode

3) remove old filename entry from old parent dir

cp(1)

1) create new filename entry in new parent dir

2) grab new inode for new file

mv(1)

1) create new filename entry in new parent dir

2) update link to filename in inode

3) remove old filename entry from old parent dir

cp(1)

1) create new filename entry in new parent dir

2) grab new inode for new file

3) copy data blocks until completed

mv(1)

1) create new filename entry in new parent dir

2) update link to filename in inode

3) remove old filename entry from old parent dir

cp(1)

1) create new filename entry in new parent dir

2) grab new inode for new file

3) copy data blocks until completed

mv(1)

1) create new filename entry in new parent dir

2) update link to filename in inode

3) remove old filename entry from old parent dir

cp(1)

1) create new filename entry in new parent dir

2) grab new inode for new file

3) copy data blocks until completed

Thanks POSIX!

mv(1)

1) create new filename entry in new parent dir

2) update link to filename in inode

3) remove old filename entry from old parent dir

cp(1)

1) create new filename entry in new parent dir

2) grab new inode for new file

3) copy data blocks until completed

when is mv actually cp?

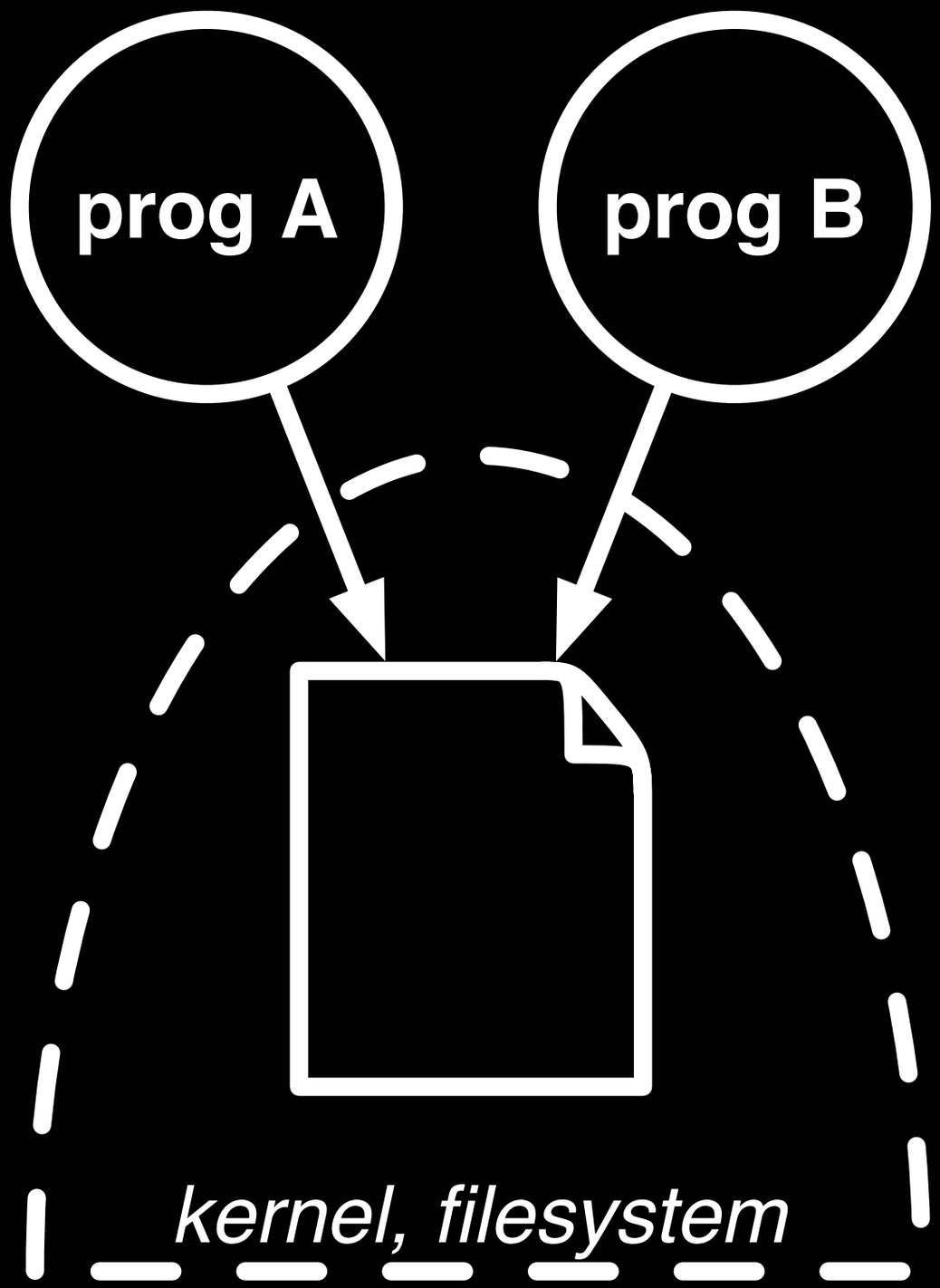
when is mv actually cp?

across logical filesystem
partitions or disks
across networks, etc...

bonus question

What is size of any file copy that must be guaranteed atomic, according to POSIX?

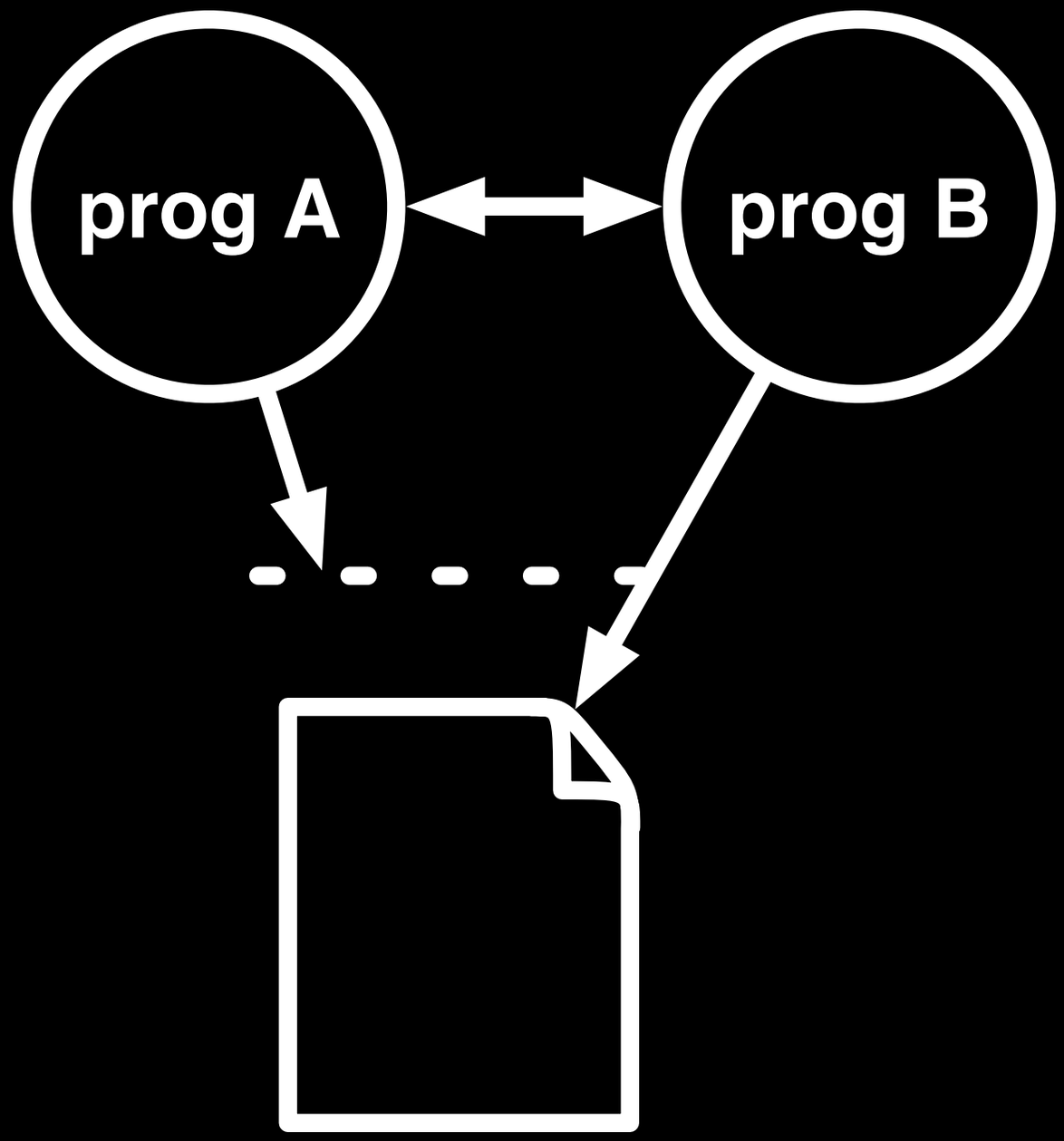
inversion of control



trust the OS

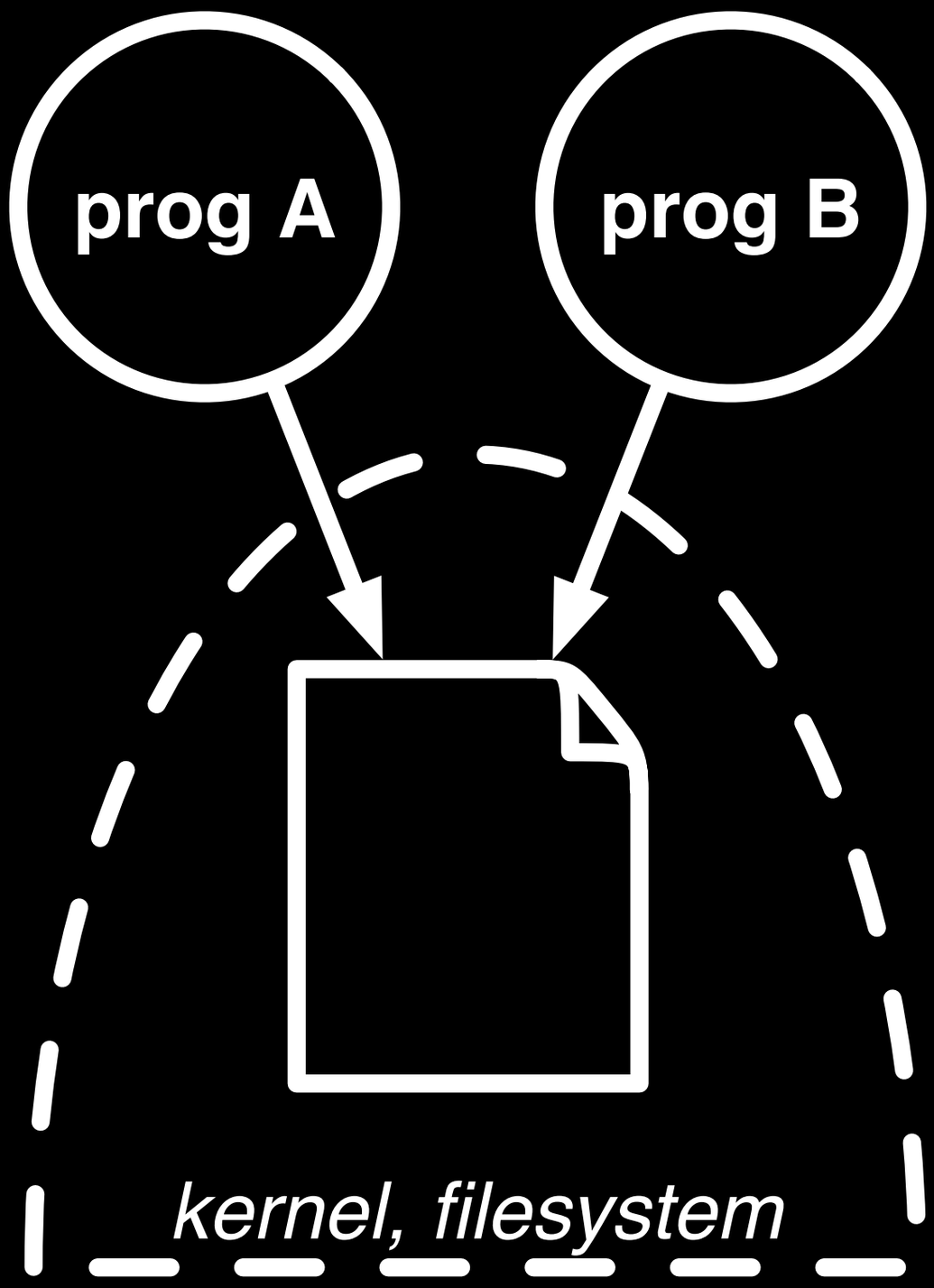

KERNEL
SHELL
UTILITIES

Coordinated control

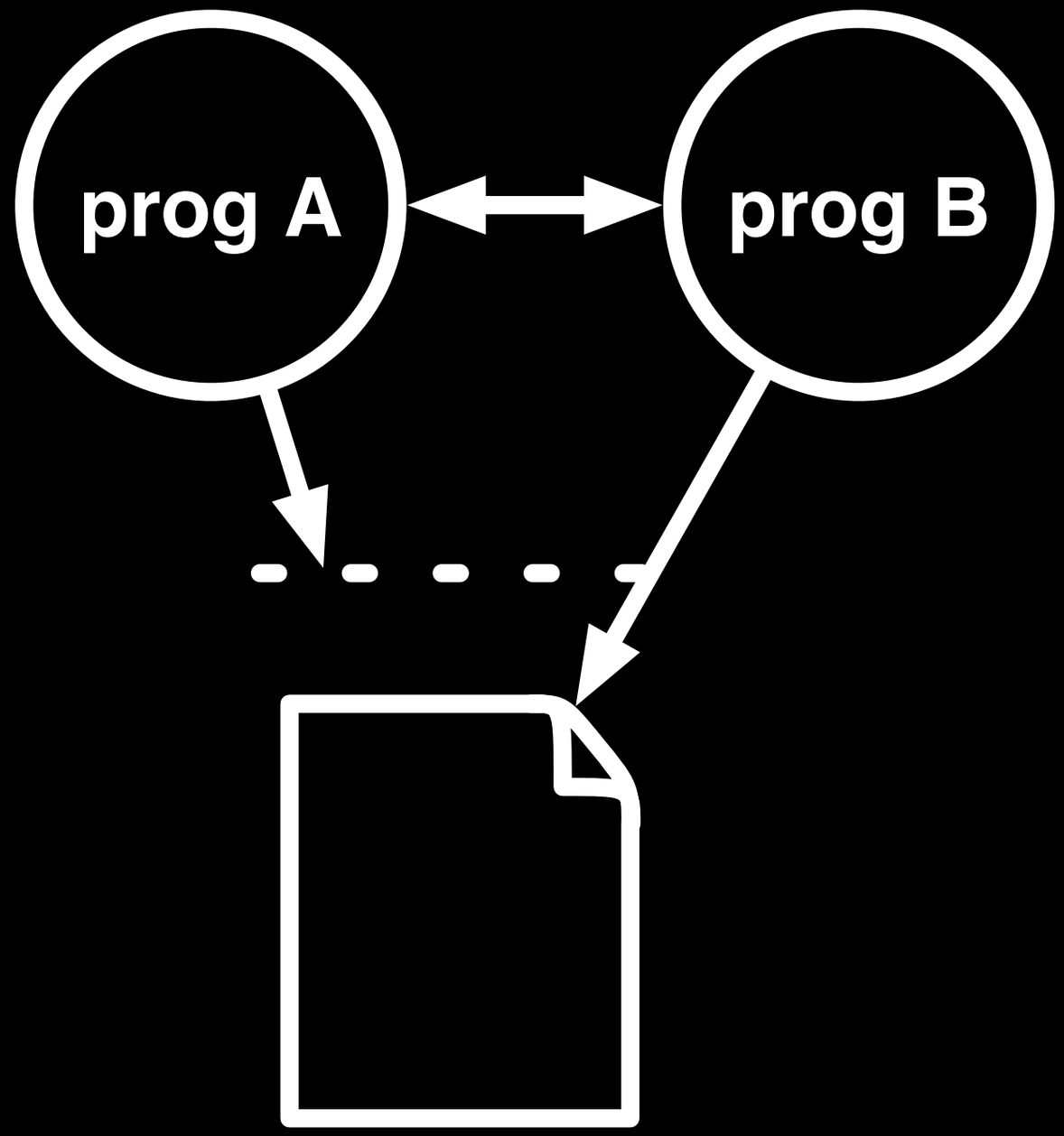


tight collaboration

inversion of control



Coordinated control



examples

general shell-fu

extra bits

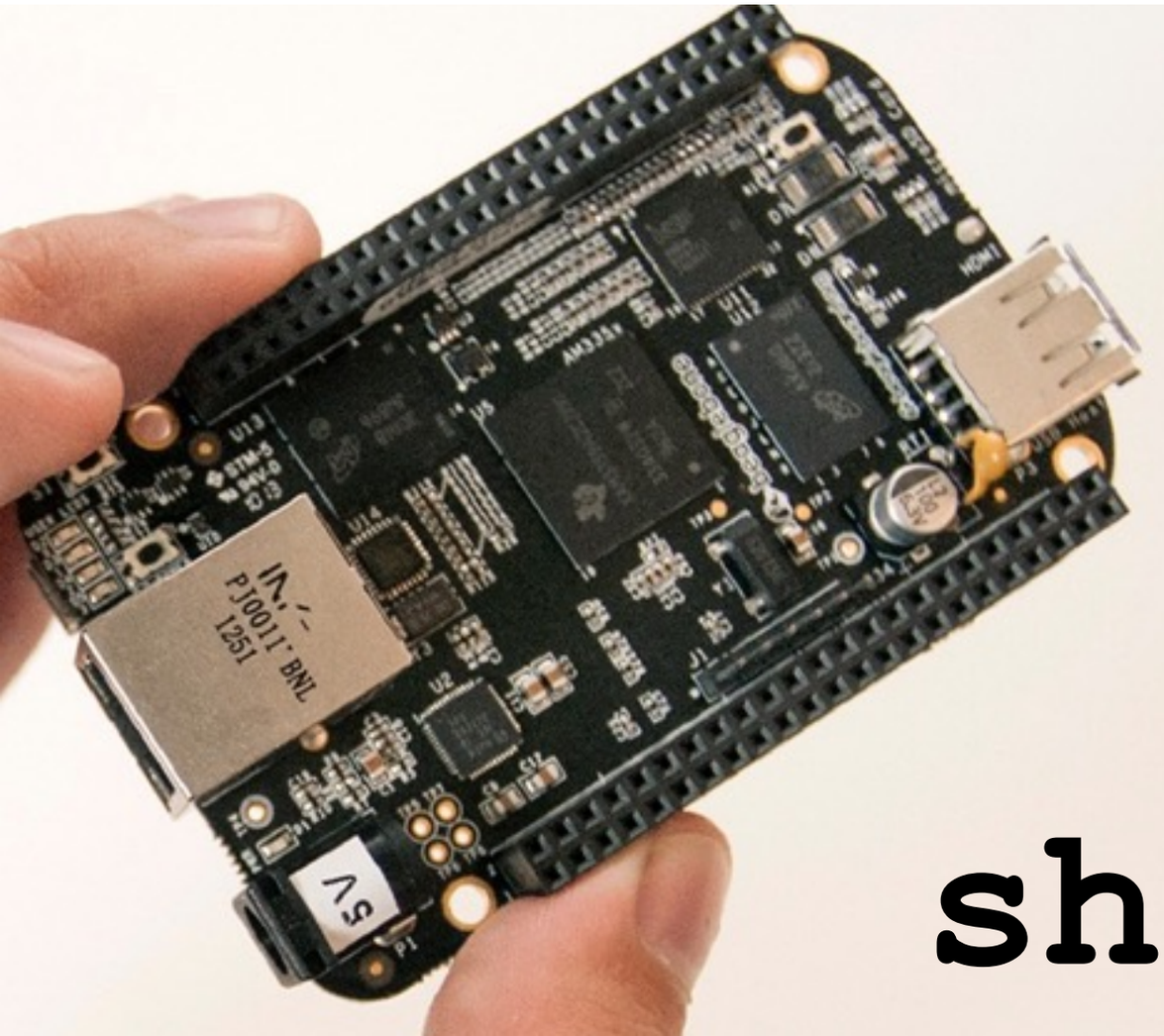
```
# a real gem:
var=${var:-word}
```

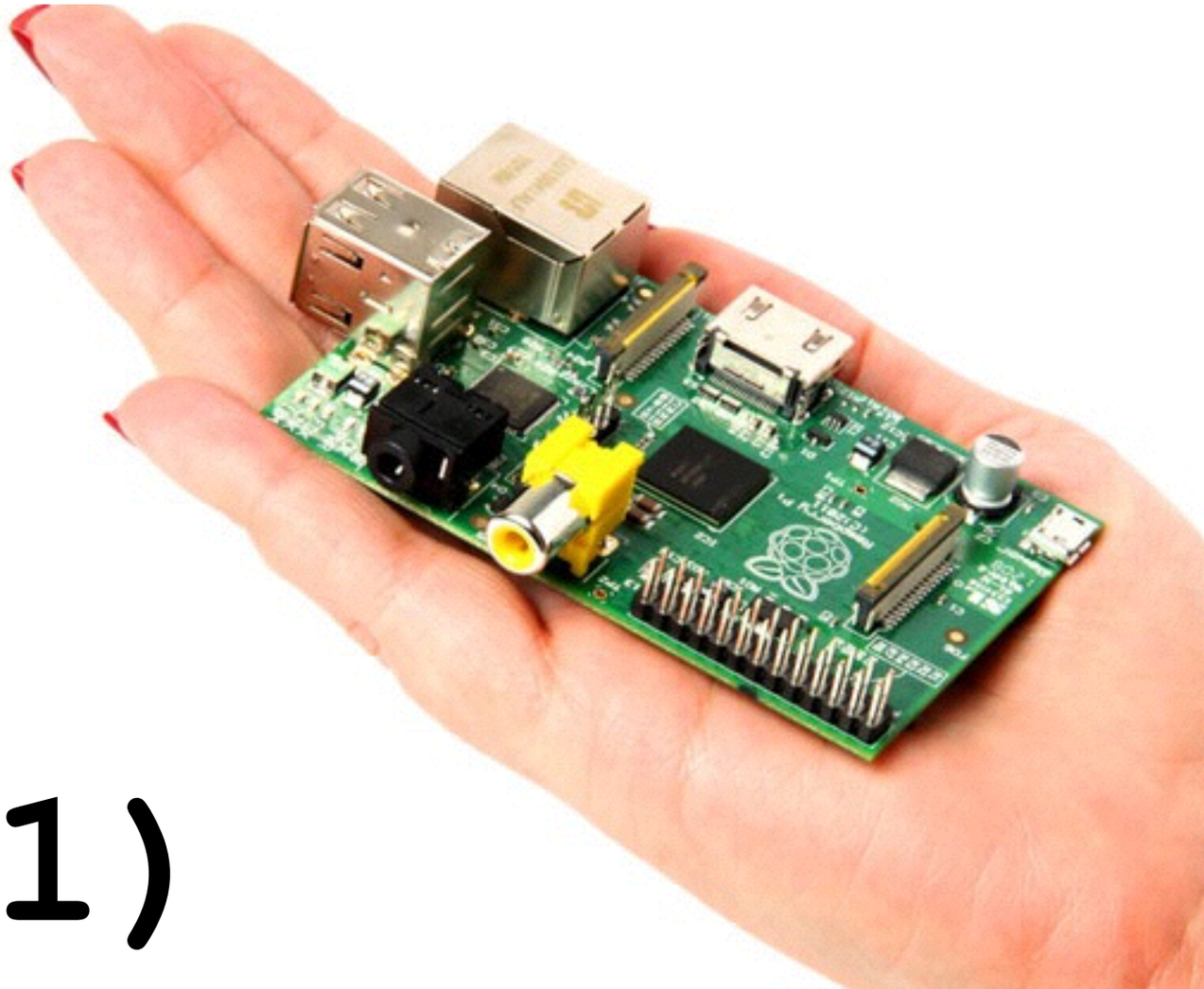

var=${var:-word}

why portability is exciting

Last login: Wed Feb 3 16:21:59 on ttys050
$


sh(1)



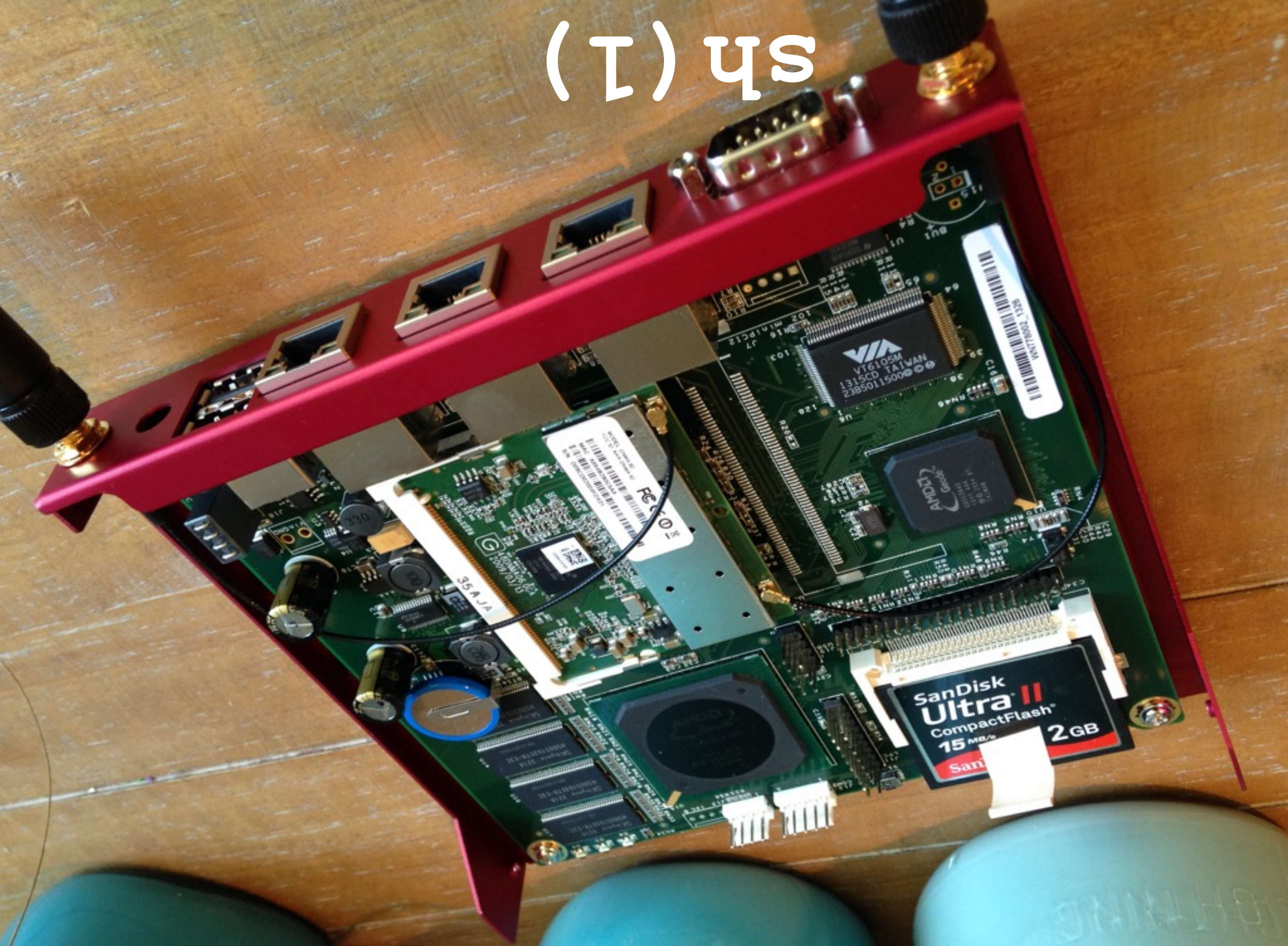


sh(1)


sh(1)

sh(1)

(I)4S


VIA
VT6105M
1315CD TAIWAN
SanDisk
Ultra II
CompactFlash
2GB
15 MB/s

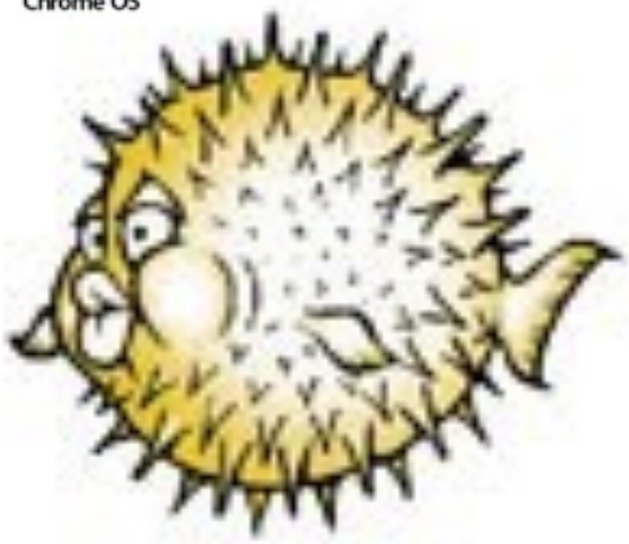
sh(1)


digital pdp-11
sh(1)


sh(1)

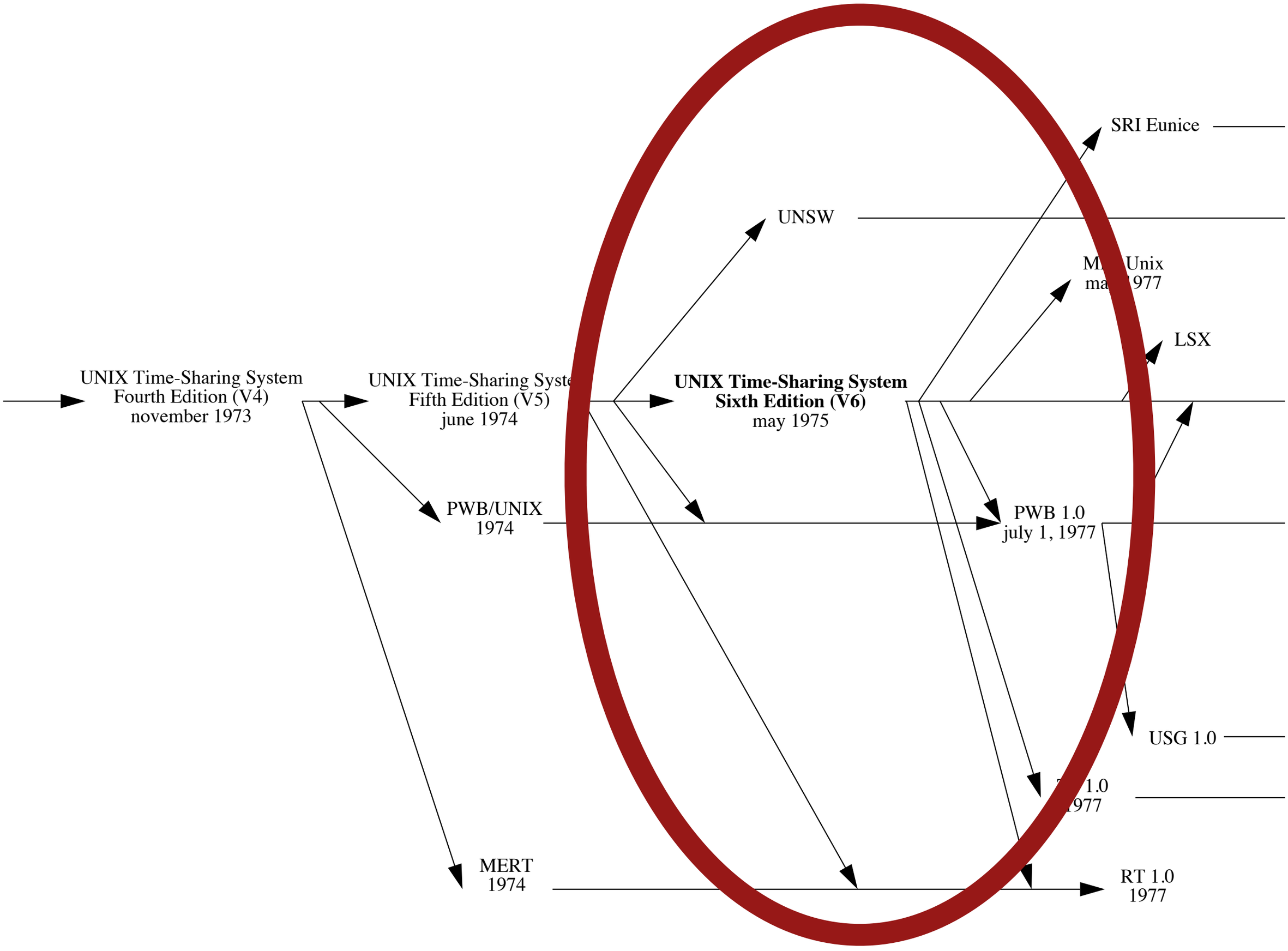
redhat.

Chrome OS



debian

ubuntu



2014

2015

etBSD 6.1.3
uary 25, 2014

NetBSD 6.1.4
april 12, 2014

NetBSD 6.1.5
september 22, 2014

NetBSD 7.0
october 8, 2015

PC-BSD 10.0
january 29, 2014

PC-BSD 10.0.2
june 24, 2014

PC-BSD 10.1
november 16, 2014

PC-BSD 10.1.1
february 2, 2015

PC-BSD 10.1.2
may 18, 2015

PC-BSD 10.2
august 21, 2015

DragonFly BSD 3.6.1
february 25, 2014

DragonFly BSD 3.6.2
april 11, 2014

DragonFly BSD
3.8.0
june 4, 2014

DragonFly BSD
3.8.1
june 16, 2014

DragonFly
BSD 3.8.2
august 8, 2014

DragonFly
BSD 4.0
nov. 25, 2014

DragonFly BSD 4.0.5
march 22, 2015

DragonFly BSD 4.2.0
june 29, 2015

DragonFly BSD 4.2.4
august 10, 2015

BSD 10.0
y 20, 2014

FreeBSD 10.1
november 14, 2014

FreeBSD 10.2
august 14, 2015

7.0.5
ry 29,
014

iOS 7.0.6
february 21,
2014

iOS 7.1
march 10,
2014

iOS 7.1.1
april 22,
2014

iOS 7.1.2
june 30,
2014

iOS 8.0
sept. 17,
2014

iOS 8.1
october 20,
2014

iOS 8.1.2
december 9,
2014

iOS 8.1.3
january 27,
2015

iOS 8.2
march 9,
2015

iOS 8.3
april 8,
2015

iOS 8.4
june 30, 2015

iOS 9.0
sept. 16, 2015

iOS 9.0.2
sept. 30, 2015

iOS 9.1
october 21, 2015

OpenBSD 5.5
may 1, 2014

OpenBSD 5.6
november 1, 2014

OpenBSD 5.7
may 1, 2015

OpenBSD 5.8
october 18, 2015

QNX 6.6
february 28, 2014

OS X Mavericks
10.9.2
february 25, 2014

OS X
Mavericks
10.9.3
may 15, 2014

OS X
Mavericks
10.9.4
june 30, 2014

OS X
Mavericks
10.9.5
sept. 17, 2014

OS X
Yosemite
10.10
oct. 16, 2014

OS X
Yosemite
10.10.1
nov. 17, 2014

OS X Yosemite
10.10.2
january 27, 2015

OS X Yosemite
10.10.3
april 8, 2015

OS X Yosemite
10.10.4
june 30, 2015

OS X Yosemite
10.10.5
august 13, 2015

OS X El Capitan
10.11
september 30, 2015

OS X El Capitan
10.11.1
october 21, 2015

ckBerry 10.2.1
uary 28, 2014

BlackBerry 10.2.2
august 8, 2014

BlackBerry 10.3.0
october 10, 2014

BlackBerry 10.3.1
march 27, 2015

BlackBerry 10.3.2
september 8, 2015

Oracle Solaris 11.2
april 29, 2014

Oracle Solaris 11.3
october 26, 2015

GNU/Hurd 0.6
april 15, 2015

Debian GNU/Hurd 2015
april 30, 2015

GNU/Hurd 0.7
october 31, 2015

Android 4.4.3
june 2, 2014

Android 4.4.4
june 19, 2014

*Android 5.0
Lollipop
(announced)*
june 19, 2014

Android 5.0
november 3, 2014

Android 5.1
march 9, 2015

Android 5.1.1
april 21, 2015

Android 6.0
Marshmallow
october 5, 2015

ux 3.13
y 19, 2014

Linux 3.14
march 30, 2014

Linux 3.15
june 8, 2014

Linux 3.16
august 3, 2014

Linux 3.17
october 5, 2014

Linux 3.18
dec. 8, 2014

Linux 3.19
february 10, 2015

Linux 4.0
april 17, 2015

Linux 4.1
june 29, 2015

Linux 4.2
august39, 2015

Linux 4.3
november 1, 2015

z/OS Unix V2R2
june 28, 2015

HP-UX 11.31/11iv3
Update 13
april 2014

HP-UX 11.31/11iv3
Update 14
march 2015

Minix 3.3.0
september 16, 2014

AIX 7.2
october 5, 2015

best practices?

quote all vars

```
var="foo bar baz"

for i in "$var" ; do
  echo "$i"
done


for i in $var ; do
  echo "$i"
done
```

{bracket} vars too

```
var="foo bar baz"

for i in "${var}" ; do
  echo "$i"
done


for i in ${var} ; do
  echo "$i"
done
```

{bracket} vars too

"/path/${var}/thing"

(it's faster to boot)

ENV convention

$VAR is for ENV vars

$var is for your program

ENV convention

$VAR is for ENV vars

$var is for your program

when to use another language

technically, hammers can drive nails,
but...


fork(2), exec, wait(2)

```
while read line; do
  echo "$line" | cut -c2
done
```

```
while read line; do
  echo "$line" | cut -c2
done
```

fork(2),
exec,
wait(2)

```
while read line; do
   echo "$line" | cut -c2 |\
   grep kern |\
   grep shared |\
   grep '^Feb'
done
```

```
cat /some/file.txt |\
  sed 's/foo/bar/g' |\
  sed 's/kernel/devo/g' |\
  sed 'G'
```

vs.

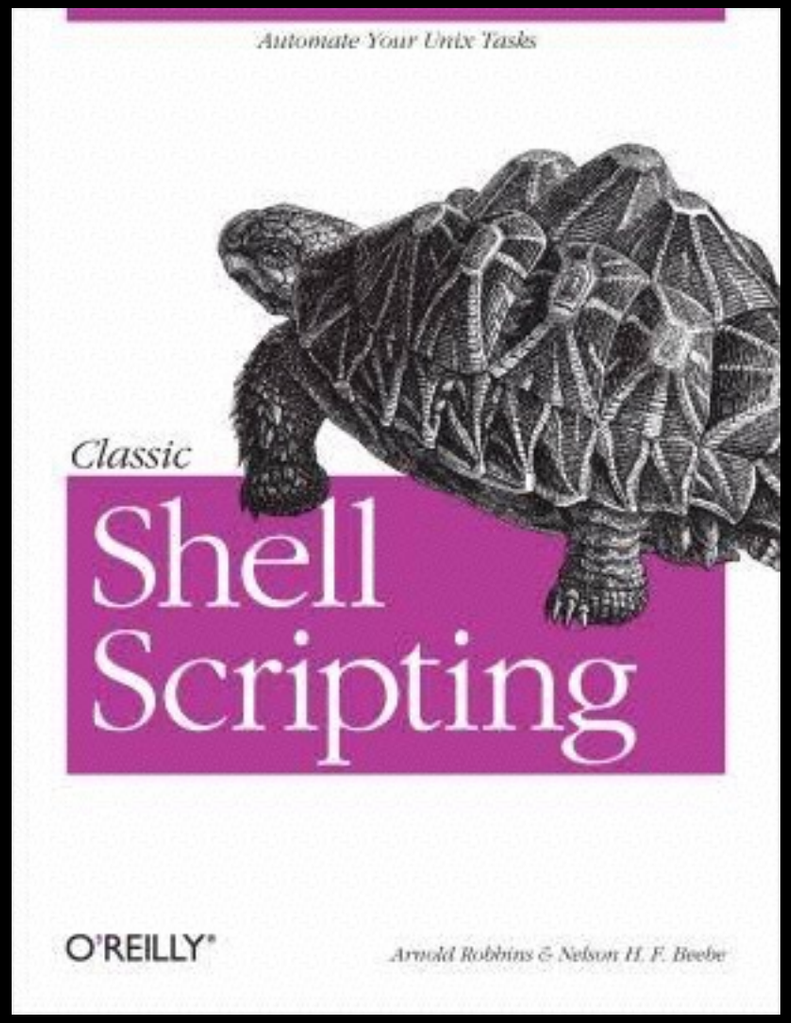
```
sed 's/foo/bar/g;s/kernel/devo/g;G' /some/file.txt
```

sh(1) manual page, hrm.

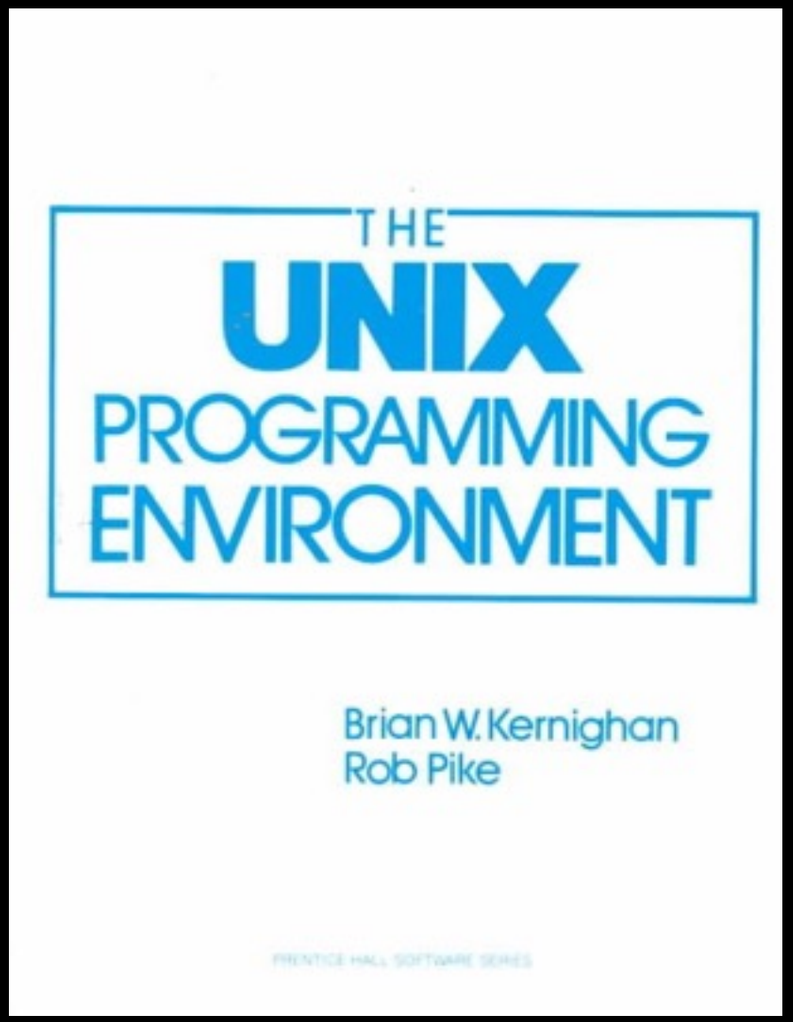
```
$ man sh | wc -l
    1618
```

http://blackskyresearch.net/

shelltables.txt



Classic Shell Scripting
Arnold Robbins



The Unix Programming Environment
Brian Kernighan and Rob Pike

my team is hiring...

enigma

http://blackskyresearch.net/**try.sh.txt**
http://blackskyresearch.net/**shelltables.txt**

ike@blackskyresearch.net