GATA Awards

The Only Way In Or Out Is Through Phone Lines
- The Matrix (1999), Lana & Lilly Wachowski

LAB04

Labs > Lab04: Bash Script for Test Cases

LEC04

Lectures > Lec04: Shell


MATT DAMON
THE
BOURNE ULTIMATUM

Have you watched the recommended movies?

Exams may have question about the movies!

Have you watched the recommended movies?

Exams may have question about the movies!

kidding :p

Variables

(Key, Value) Pairs

Questions (not commands) whose answers are already provided!
Like Frequently Asked Questions (FAQs)

System Variables

aka. Environment Variables, Global Variables, Unix Variables

Hossein: Kernel Variables

By convention, keys are UPPERCASE

To see the value, `echo ${KEY}`

```
hfani@alpha:~$ echo $OSTYPE
linux-gnu
hfani@alpha:~$ echo $USER
hfani
hfani@alpha:~$ echo $LOGNAME
hfani
hfani@alpha:~$ echo $HOME
/home/hfani
hfani@alpha:~$ echo $HOST

hfani@alpha:~$ echo $DISPLAY

hfani@alpha:~$ echo $EDITOR

hfani@alpha:~$ echo $SHELL
/bin/bash
hfani@alpha:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/opt/maple2021/bin:/opt/netlogo:/
ome/hfani/.dotnet/tools
hfani@alpha:~$
```

Not Set! Unset.

An important one! Very important actually.

PATH

Colon(:)-delimited list of directories

Tells the shell where to look/search when you request a particular program

```
hfani@alpha:~$ cd /usr/bin
hfani@alpha:/usr/bin$ ./cc hello.c -o hello
cc: error: hello.c: No such file or directory
cc: fatal error: no input files
compilation terminated.
hfani@alpha:/usr/bin$ cd ~
hfani@alpha:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/opt/maple2021/bin:/opt/netlogo:/opt/eclip
ome/hfani/.dotnet/tools
hfani@alpha:~$ cc hello.c -o hello
hfani@alpha:~$
```

Back to home directory

Because shell also searched these locations

Other Variables

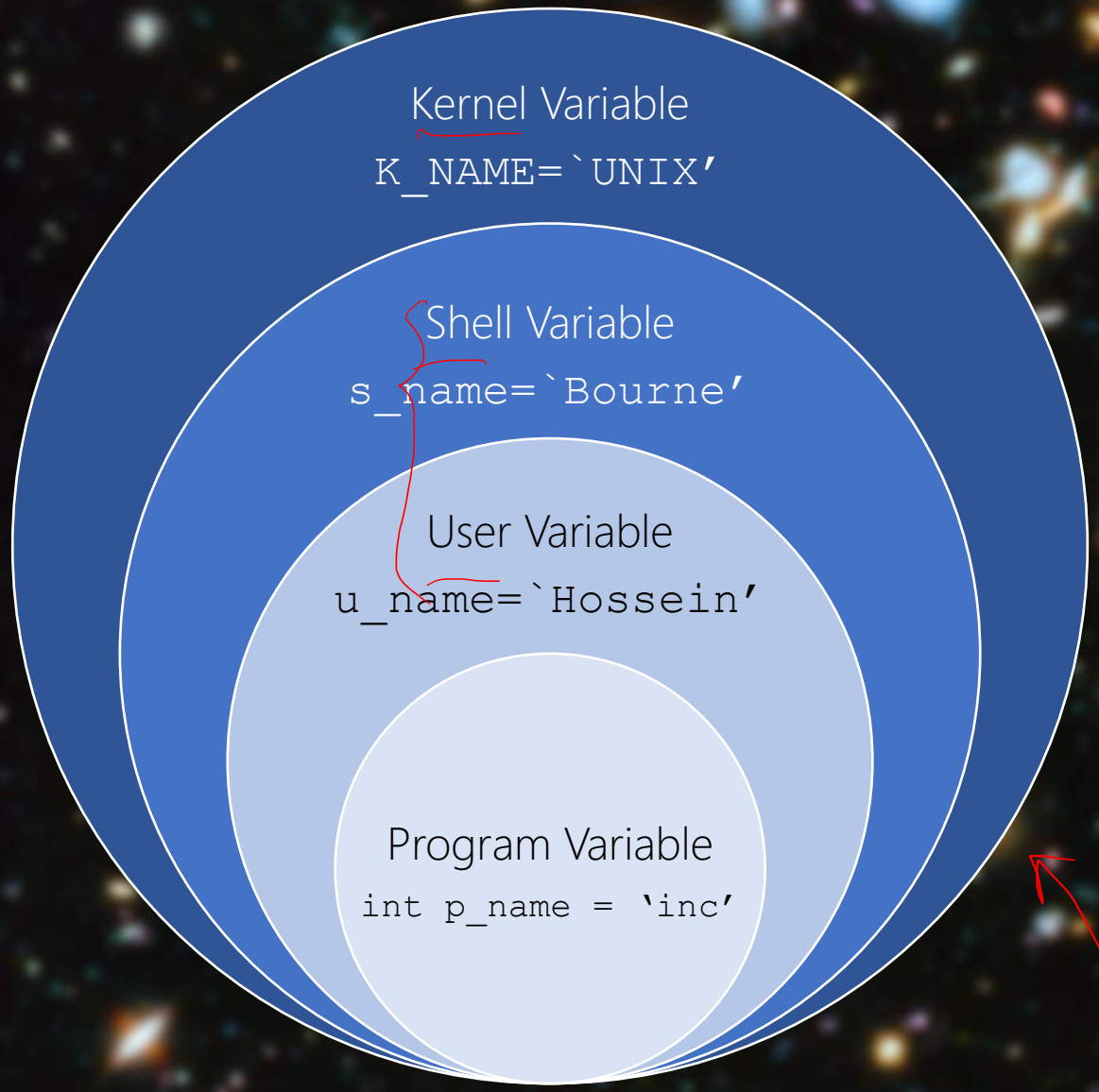
aka. User Variables, Local Variable, Shell Variables

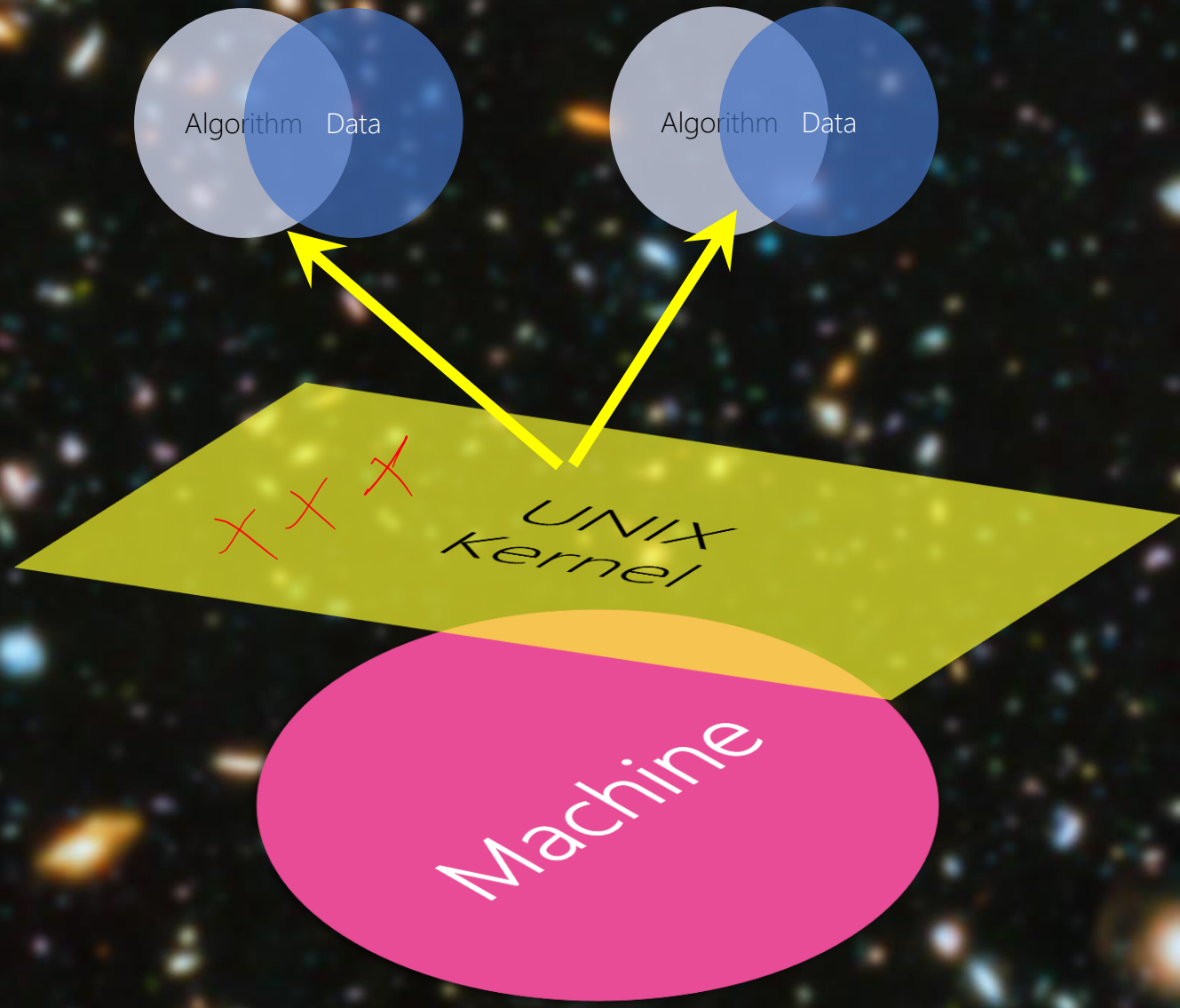
By convention, keys are lowercase
To see the value, `echo ${key}`

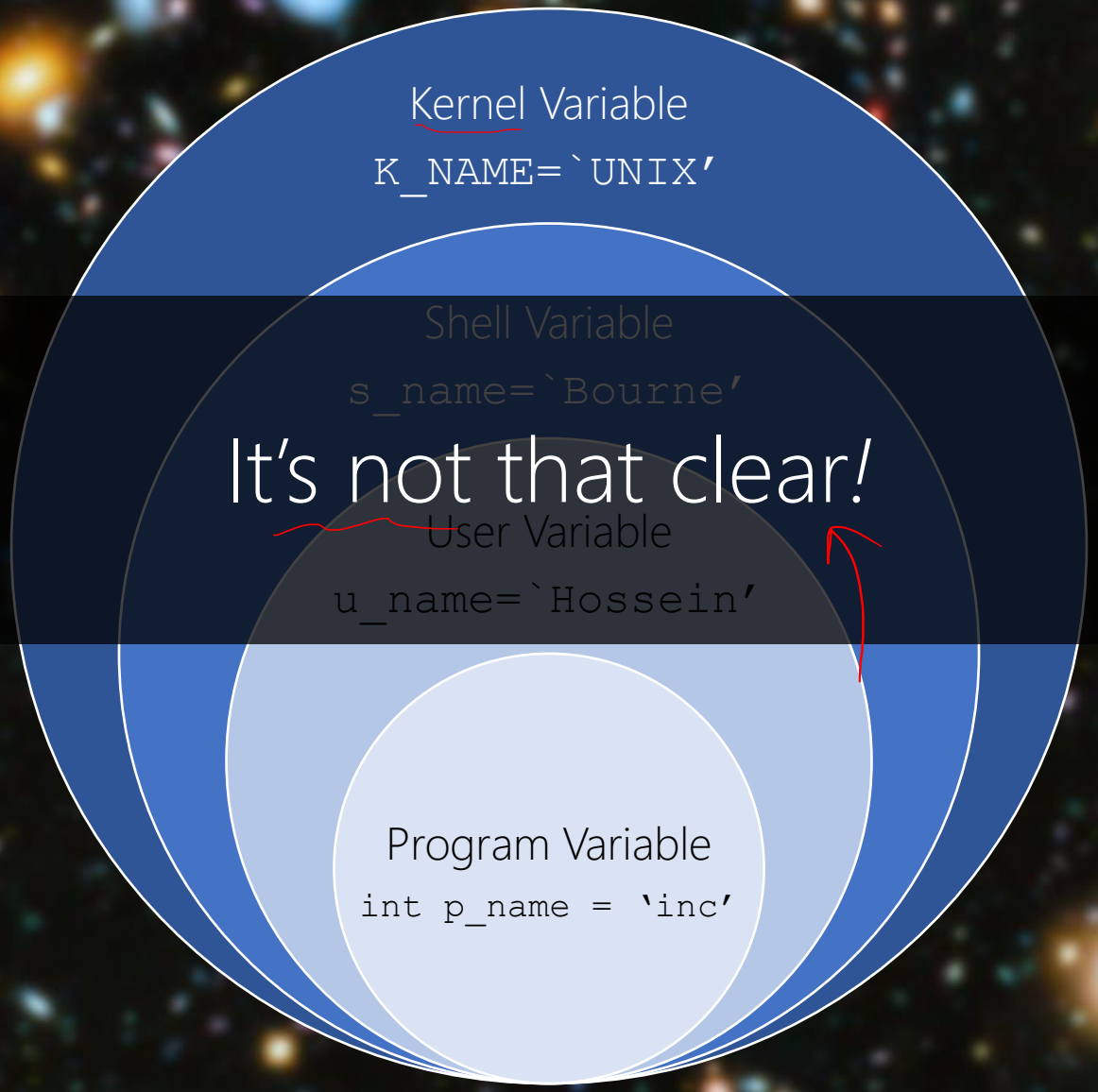
Variables

Kernel and non-Kernel

Is it able to modify the key's value? Yes.
Is it able to unset the key's value? Yes.
Is it able to add a new key=value pair? Yes.
Is it able to persist the change? Yes.
How? It depends on the shell ☹


Kernel Variable
K_NAME=`UNIX'
Shell Variable
s_name=`Bourne'
User Variable
u_name=`Hossein'
Program Variable
int p_name = 'inc'


Algorithm
Data
Algorithm
Data
UNIX
Kernel
Machine

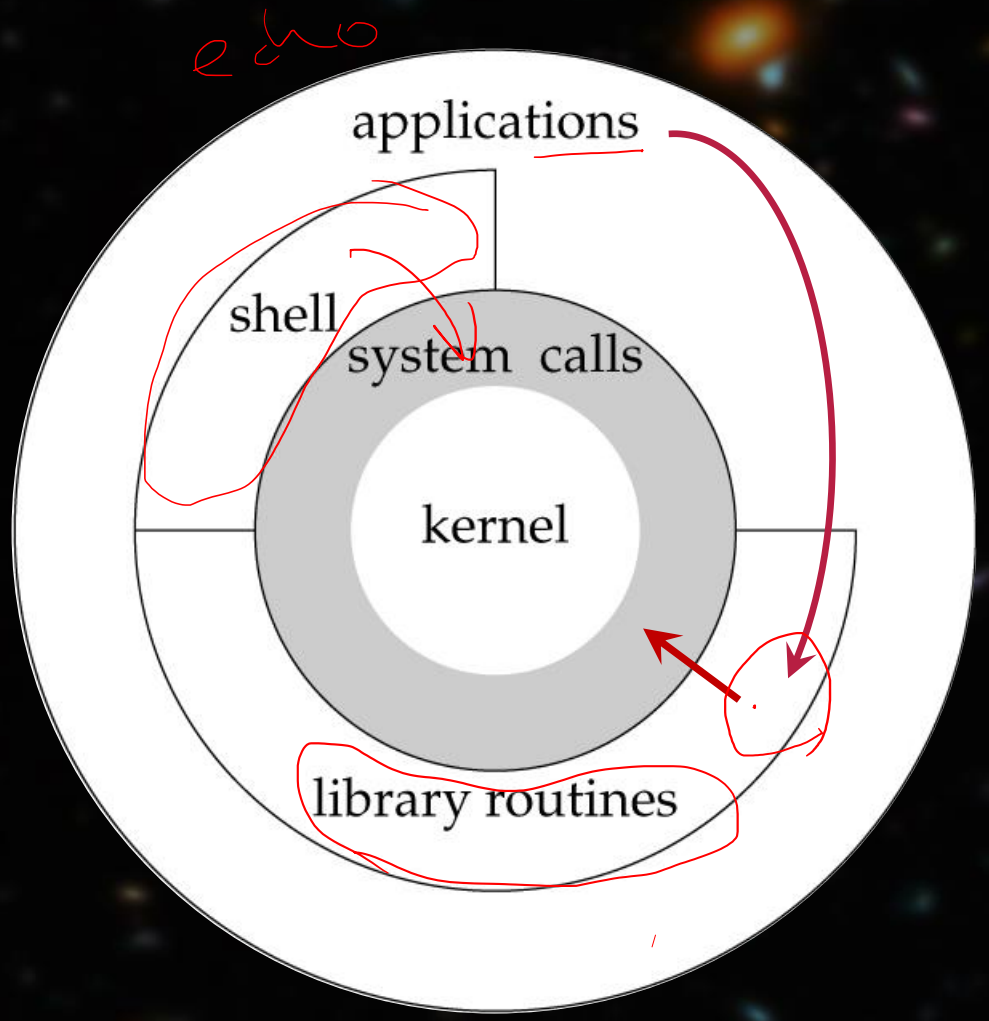


It's not that clear!

Access Kernel Variables by Call to Library Routine

Lab02

```
#include <stdlib.h>;
char *getenv(const char *KEY)
```



Header	FreeBSD 8.0	Linux 3.2.0	Mac OS X 10.6.8	Solaris 10	Description
`<assert.h>`	•	•	•	•	verify program assertion
`<complex.h>`	•	•	•	•	complex arithmetic support
`<ctype.h>`	•	•	•	•	character classification and mapping support
`<errno.h>`	•	•	•	•	error codes (Section 1.7)
`<fenv.h>`	•	•	•	•	floating-point environment
`<float.h>`	•	•	•	•	floating-point constants and characteristics
`<inttypes.h>`	•	•	•	•	integer type format conversion
`<iso646.h>`	•	•	•	•	macros for assignment, relational, and unary operators
`<limits.h>`	•	•	•	•	implementation constants (Section 2.5)
`<locale.h>`	•	•	•	•	locale categories and related definitions
`<math.h>`	•	•	•	•	mathematical function and type declarations and constants
`<setjmp.h>`	•	•	•	•	nonlocal goto (Section 7.10)
`<signal.h>`	•	•	•	•	signals (Chapter 10)
`<stdarg.h>`	•	•	•	•	variable argument lists
`<stdbool.h>`	•	•	•	•	Boolean type and values
`<stddef.h>`	•	•	•	•	standard definitions
`<stdint.h>`	•	•	•	•	integer types
`<stdio.h>`	•	•	•	•	standard I/O library (Chapter 5)
`<stdlib.h>`	•	•	•	•	utility functions
`<string.h>`	•	•	•	•	string operations
`<tgmath.h>`	•	•	•	•	type-generic math macros
`<time.h>`	•	•	•	•	time and date (Section 6.10)
`<wchar.h>`	•	•	•	•	extended multibyte and wide character support
`<wctype.h>`	•	•	•	•	wide character classification and mapping support

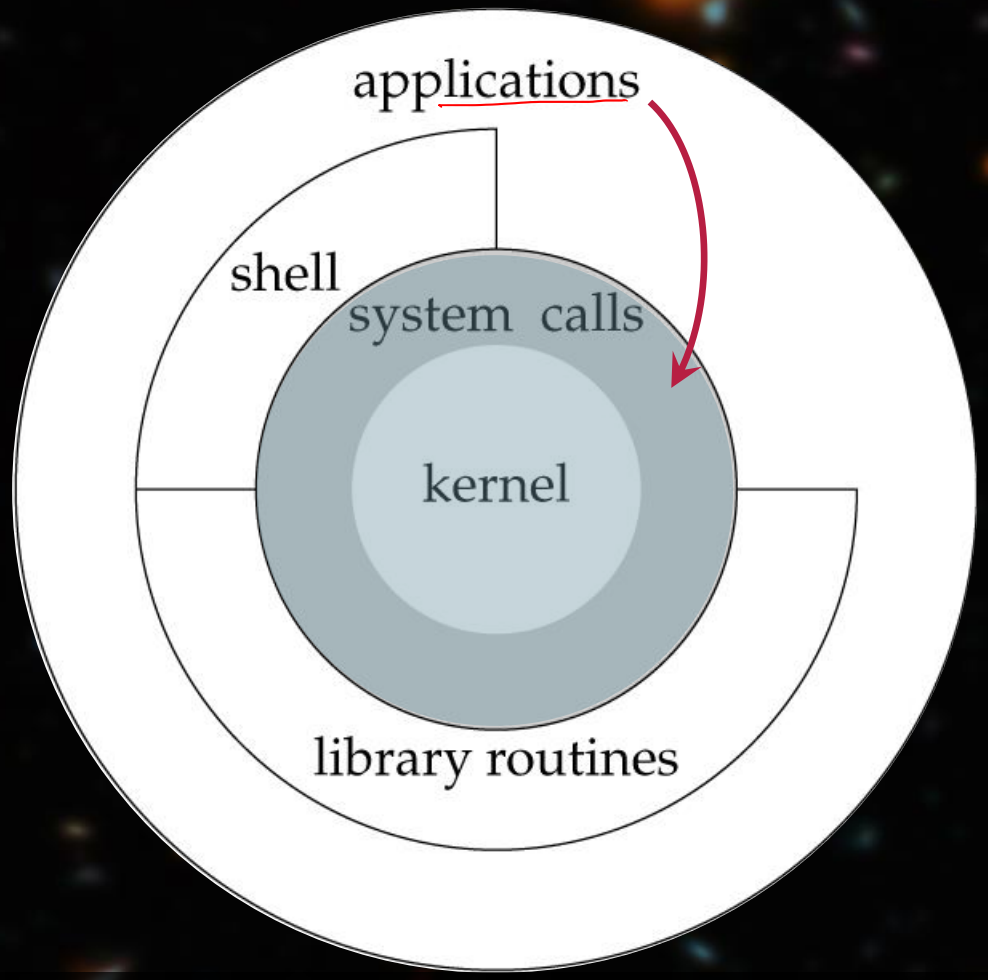
```c
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
int main()
{
  printf("Hello World! This is hfani@uwindsor.ca, StudentID: 123456789\n");
  time_t t = time(NULL);
  struct tm tm = *localtime(&t);
  printf("now: %d-%02d-%02d %02d:%02d:%02d\n", tm.tm_year + 1900, tm.tm_mon + 1, tm.tm_m
  printf("%s@shell:%s$\n", getenv("USER"),getenv("PWD"));

}
```

The library routine that does system call to Kernel
Your program statically linked this library!

Getting the values of USER and PWD (path of working directory)

Access Kernel Variables by System Call

```
#include <unistd.h>;
extern char **environ
```



Header	FreeBSD 8.0	Linux 3.2.0	Mac OS X 10.6.8	Solaris 10	Description
<aio.h>	•	•	•	•	asynchronous I/O
<cpio.h>	•	•	•	•	cpio archive values
<dirent.h>	•	•	•	•	directory entries (Section 4.22)
<dlfcn.h>	•	•	•	•	dynamic linking
<fcntl.h>	•	•	•	•	file control (Section 3.14)
<fnmatch.h>	•	•	•	•	filename-matching types
<glob.h>	•	•	•	•	pathname pattern-matching and generation
<grp.h>	•	•	•	•	group file (Section 6.4)
<iconv.h>	•	•	•	•	codeset conversion utility
<langinfo.h>	•	•	•	•	language information constants
<monetary.h>	•	•	•	•	monetary types and functions
<netdb.h>	•	•	•	•	network database operations
<nl_types.h>	•	•	•	•	message catalogs
<poll.h>	•	•	•	•	poll function (Section 14.4.2)
<pthread.h>	•	•	•	•	threads (Chapters 11 and 12)
<pwd.h>	•	•	•	•	password file (Section 6.2)
<regex.h>	•	•	•	•	regular expressions
<sched.h>	•	•	•	•	execution scheduling
<semaphore.h>	•	•	•	•	semaphores
<strings.h>	•	•	•	•	string operations
<tar.h>	•	•	•	•	tar archive values
<termios.h>	•	•	•	•	terminal I/O (Chapter 18)
<unistd.h>	•	•	•	•	symbolic constants
<wordexp.h>	•	•	•	•	word-expansion definitions
<arpa/inet.h>	•	•	•	•	Internet definitions (Chapter 16)
<net/if.h>	•	•	•	•	socket local interfaces (Chapter 16)
<netinet/in.h>	•	•	•	•	Internet address family (Section 16.3)
<netinet/tcp.h>	•	•	•	•	Transmission Control Protocol definitions
<sys/mman.h>	•	•	•	•	memory management declarations
<sys/select.h>	•	•	•	•	select function (Section 14.4.1)
<sys/socket.h>	•	•	•	•	sockets interface (Chapter 16)
<sys/stat.h>	•	•	•	•	file status (Chapter 4)
<sys/statvfs.h>	•	•	•	•	file system information
<sys/times.h>	•	•	•	•	process times (Section 8.17)
<sys/types.h>	•	•	•	•	primitive system data types (Section 2.8)
<sys/un.h>	•	•	•	•	UNIX domain socket definitions (Section 17.2)
<sys/utsname.h>	•	•	•	•	system name (Section 6.9)
<sys/wait.h>	•	•	•	•	process control (Section 8.6)

```c
#include <stdio.h>
#include <unistd.h>

extern char **environ;
int main(int argc, char *argv[])
{
    int index = 0;
    printf("Environment variables:\n");
    index = 0;
    while (environ[index])
    {
        printf("envp[%d]: %s\n", index, environ[index]);
        ++index;
    }
    return 0;
}

~
~
~
~
~
```

*System call to Kernel! (**unix** **standard** library)*
Either statically or dynamically linked.

```
Environment variables:
envp[0]: SHELL=/bin/bash
envp[1]: LANGUAGE=en_CA:en
envp[2]: NO_AT_BRIDGE=1
envp[3]: TWO_TASK=cs01
envp[4]: PWD=/home/hfani
envp[5]: LOGNAME=hfani
envp[6]: XDG_SESSION_TYPE=tty
envp[7]: PRINTER=cs_commons
envp[8]: MOTD_SHOWN=pam
envp[9]: VIRTUALENVWRAPPER_SCRIPT=/usr/share/virtualenvwrapper/virtualenvwrapper.sh
envp[10]: HOME=/home/hfani
envp[11]: LANG=en_CA.UTF-8
envp[12]: LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:
:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;
=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.l
*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01
r=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:
;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35
1;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:
;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.webp=01;35:*.ogm=01;35:*.mp4=01;35:
;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.f
:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00
=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36
0;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
envp[13]: _VIRTUALENVWRAPPER_API= mkvirtualenv rmvirtualenv lsvirtualenv showvirtualenv workon add2virtualenv cdsitepackages cd
nv lssitepackages togglegloballsitepackages cpvirtualenv setvirtualenvproject mkproject cdproject mktmpenv wipeenv allvirtualenv
alenv rmvirtualenv lsvirtualenv showvirtualenv workon add2virtualenv cdsitepackages cdvirtualenv lssitepackages togglegloballsit
s cpvirtualenv setvirtualenvproject mkproject cdproject mktmpenv wipeenv allvirtualenv
envp[14]: ORACLE_HOME=/usr/lib/oracle/12.1/client64
envp[15]: SSH_CONNECTION=137.207.140.134 63217 137.207.82.51 22
envp[16]: WINEDLLOVERRIDES=winemenubuilder.exe=d
envp[17]: LESSCLOSE=/usr/bin/lesspipe %s %s
envp[18]: XDG_SESSION_CLASS=user
envp[19]: TERM=xterm
envp[20]: LESSOPEN=| /usr/bin/lesspipe %s
envp[21]: USER=hfani
```

Shell Script for Lab03

Sequence of Built-ins (commands) to be executed line by line the shell

```
hfani@alpha:~$ vi build_lab03.sh

#!/bin/bash
echo "start building lab03 program:"
echo "compiling to assembly lines ..."
cc main.c -S
cc increment.c -S
echo "translating to opcodes ..."
cc main.s -c
cc increment.s -c
echo "statically linking all required opcodes ..."
cc main.o increment.o -o main
echo "build successfully done!"
~
~
~
~
~
~
~
~
~
~
~
~
~
```

if input=='echo'

./main

```
hfani@alpha:~$ chmod +x build_lab03.sh
hfani@alpha:~$ ./build_lab03.sh
start building lab03 program:
compiling to assembly lines ...
main.c: In function 'main':
main.c:5:6: warning: implicit declaration of function 'increment' [-Wimplicit-function-de
    5 |  a = increment(a);
      |      ^~~~~~~~~
translating to opcodes ...
statically linking all required opcodes ...
build successfully done!
hfani@alpha:~$
```

1 *Important: make it executable*

2 bash

Shell Script

Any compilations to assembly? No! (Yes)
Any translation to opcodes? No! (Yes)
Who runs the scripts? Shell (Processor)
Are shell scripts programs? Yes.

main( ) {

" main() ; "

}

cc

run

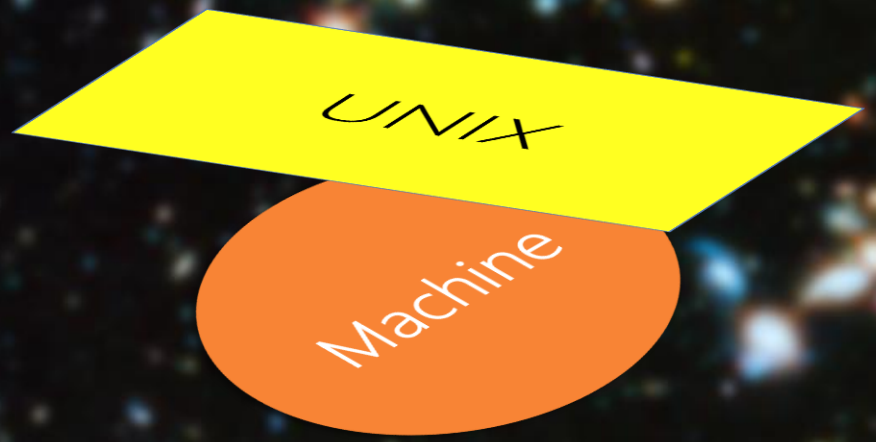
Shell as a Programming Language

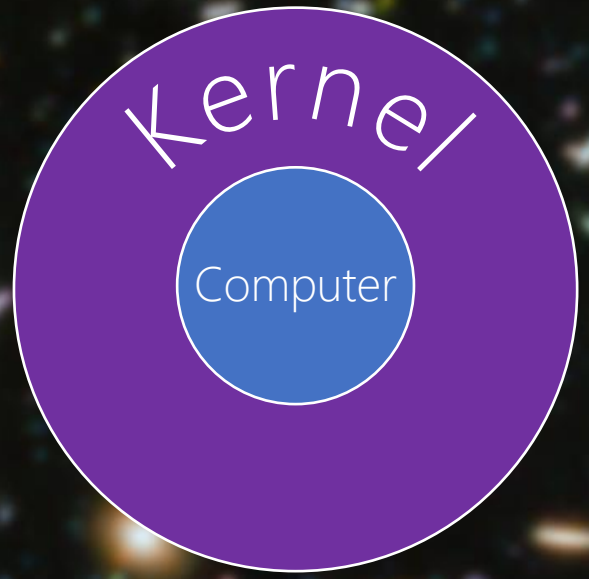
This Week's Lab: Lab04

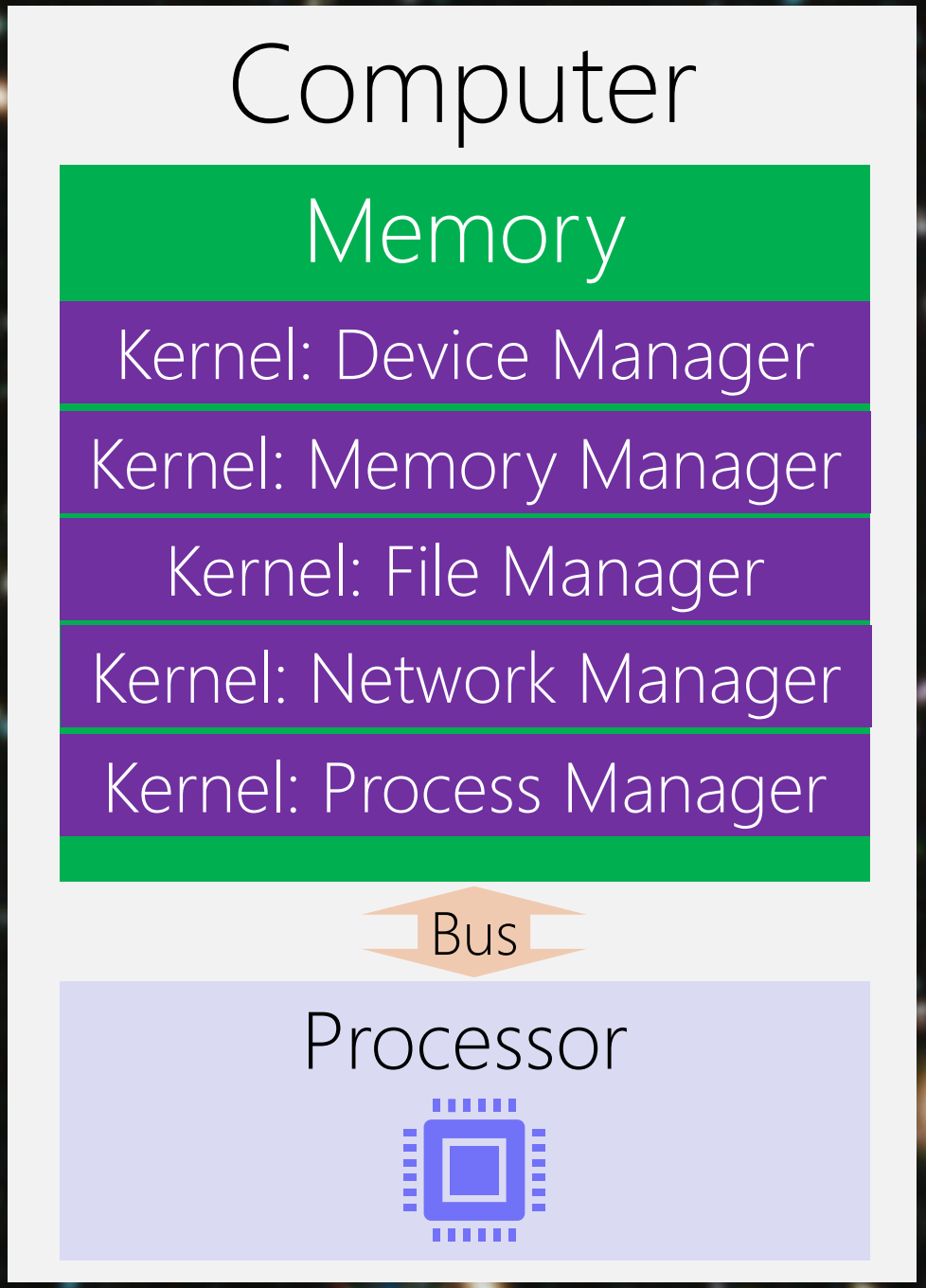
Cheat Sheet → https://devhints.io/bash

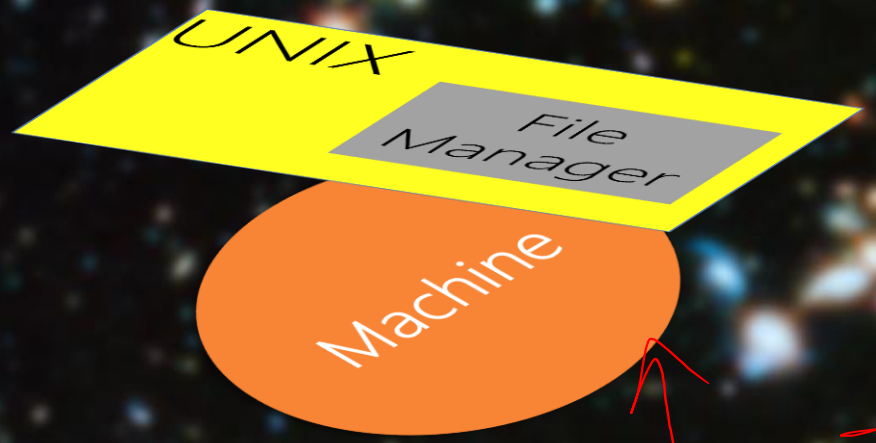
The Only Way In Or Out Is Through Phone Lines

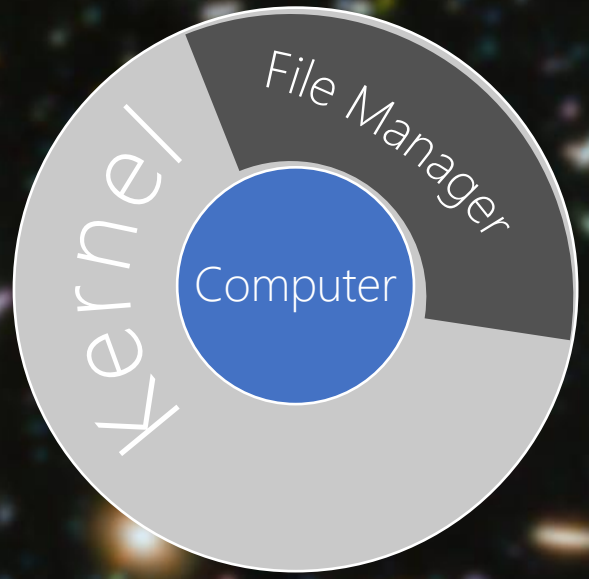
- The Matrix (1999), Lana & Lilly Wachowski
- https://www.youtube.com/watch?v=aLbdfgY_IhM

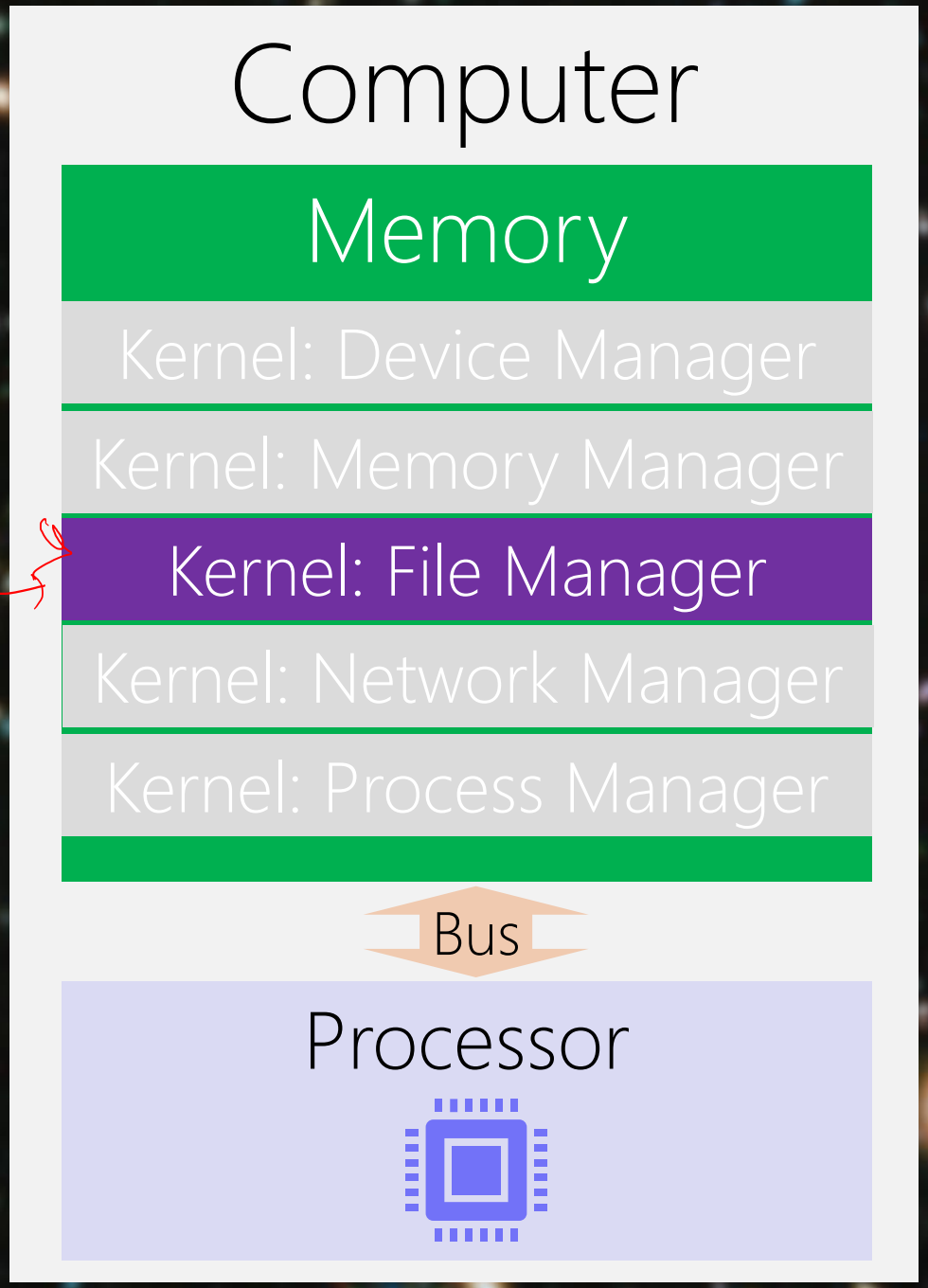

UNIX
Machine

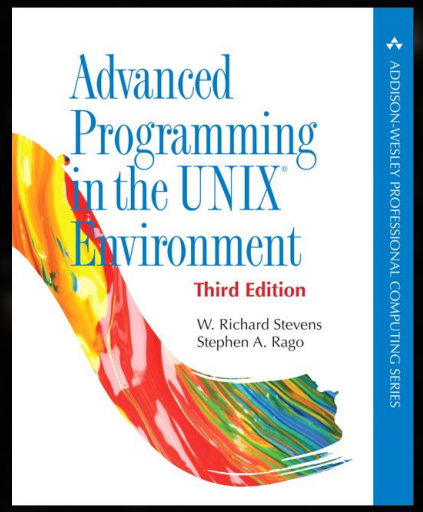

Kernel
Computer


Computer
Memory
Kernel: Device Manager
Kernel: Memory Manager
Kernel: File Manager
Kernel: Network Manager
Kernel: Process Manager
Bus
Processor

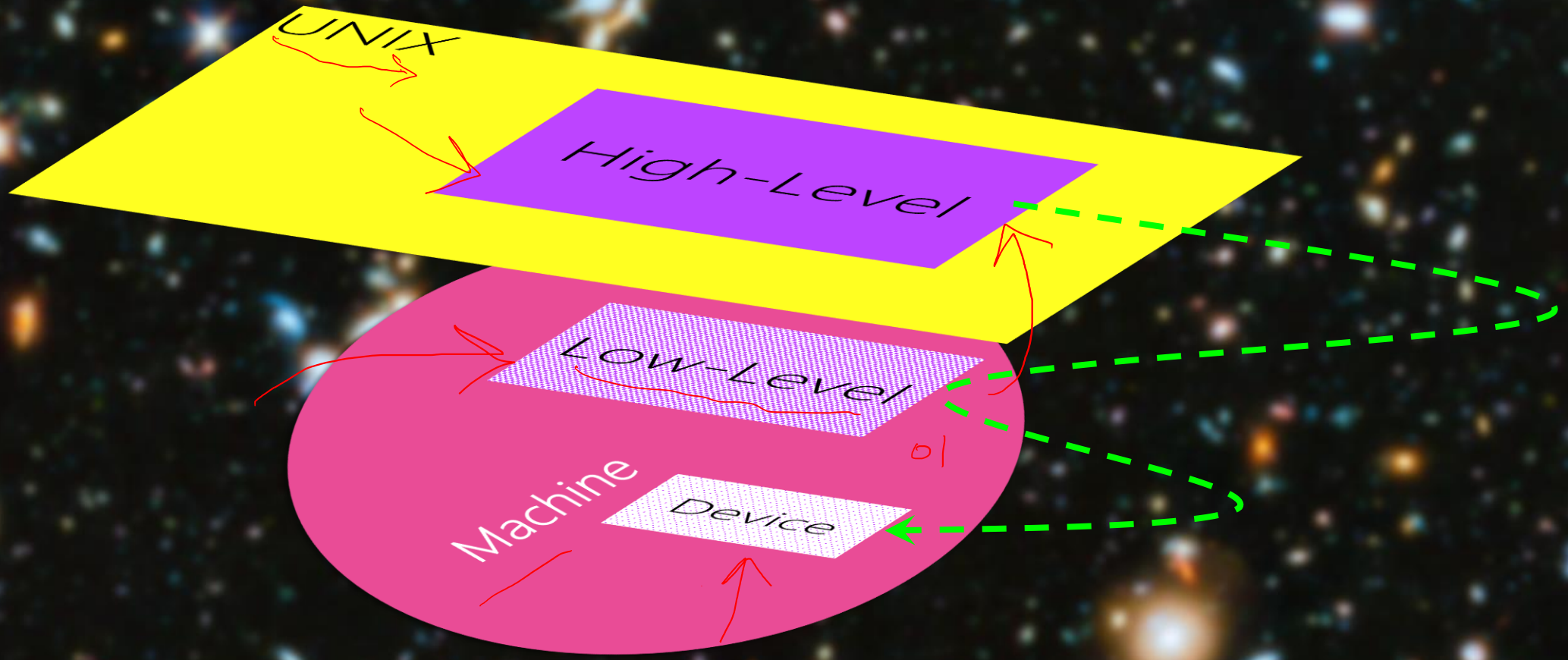

UNIX
File Manager
Machine


Kernel
File Manager
Computer


Computer
Memory
Kernel: Device Manager
Kernel: Memory Manager
Kernel: File Manager
Kernel: Network Manager
Kernel: Process Manager
Bus
Processor



Chapter 03: File I/O


UNIX
High-Level
Low-Level
Machine
Device

File Manager

widely known as File System

NTFS

exFAT

FAT

High-Level

0 1 0 0 0 0 0 0 0 0

Device is a Single 1-D Array (String) of Bytes

Even Memory and Processor!

File System: High-Level

Device is a Single 1-D Array (String) of Bytes

Please give up memory & processor.
Leave them for Process Manager!

File System: High-Level

Device is a Single 1-D Array (String) of Bytes

Keyboard: Read Only (RD)

File System: High-Level

Device is a Single 1-D Array (String) of Bytes

Printer: Write Only (WR)

File System: High-Level

Device is a Single 1-D Array (String) of Bytes

Monitor: Write Only (WR)

File System: High-Level

Device is a Single 1-D Array (String) of Bytes

Touchscreen: Read Write (RDWR)

File System: High-Level

Device is a Single 1-D Array (String) of Bytes

Storage: Read Write (RDWR)

HDD, USB, SSD, NVMe, CD-RW, DVD-RW

File System: High-Level

Storage Device == String of Bytes

File

File System: High-Level

Storage Device 1 == File 1
Storage Device 2 == File 2

File System: High-Level

Large Storage Device == *Set of* Files

set of sub-devices

File System: High-Level

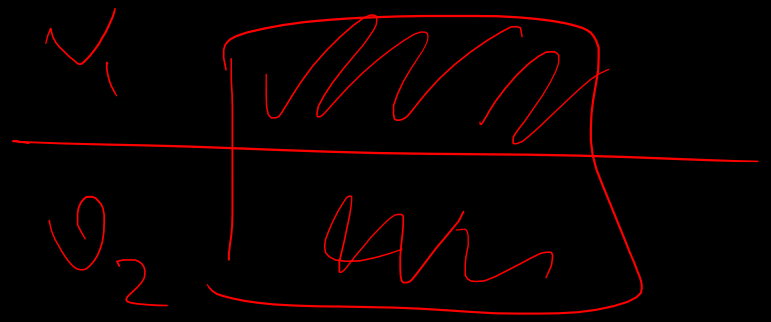
Large Monitor == *Set of sub-*Monitors == *Set of* Files

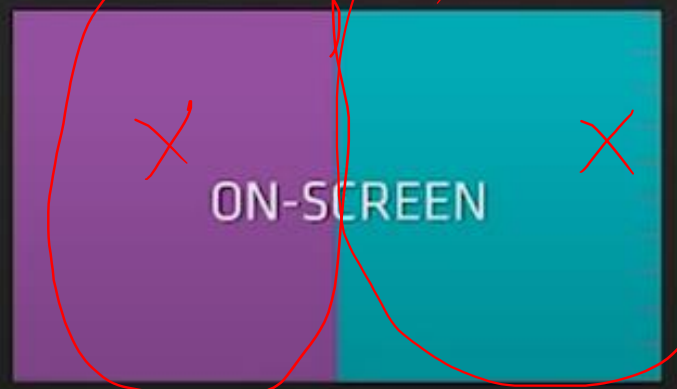
set of sub-devices

File System: High-Level

SPLIT-FRAME RENDERING

SUPREMELY RESPONSIVE MULTI-GPU GAMING





DIRECTX® 12

- New mode available to devs: split-frame rend
- Each frame of a game is split into a tile
- Each GPU in the system renders one tile
- Frames no longer need to be queued; time be frame completion and user viewing reduced b
- Using the GPUs in parallel to work on one fra multiple GPUs to behave like one much more GPU

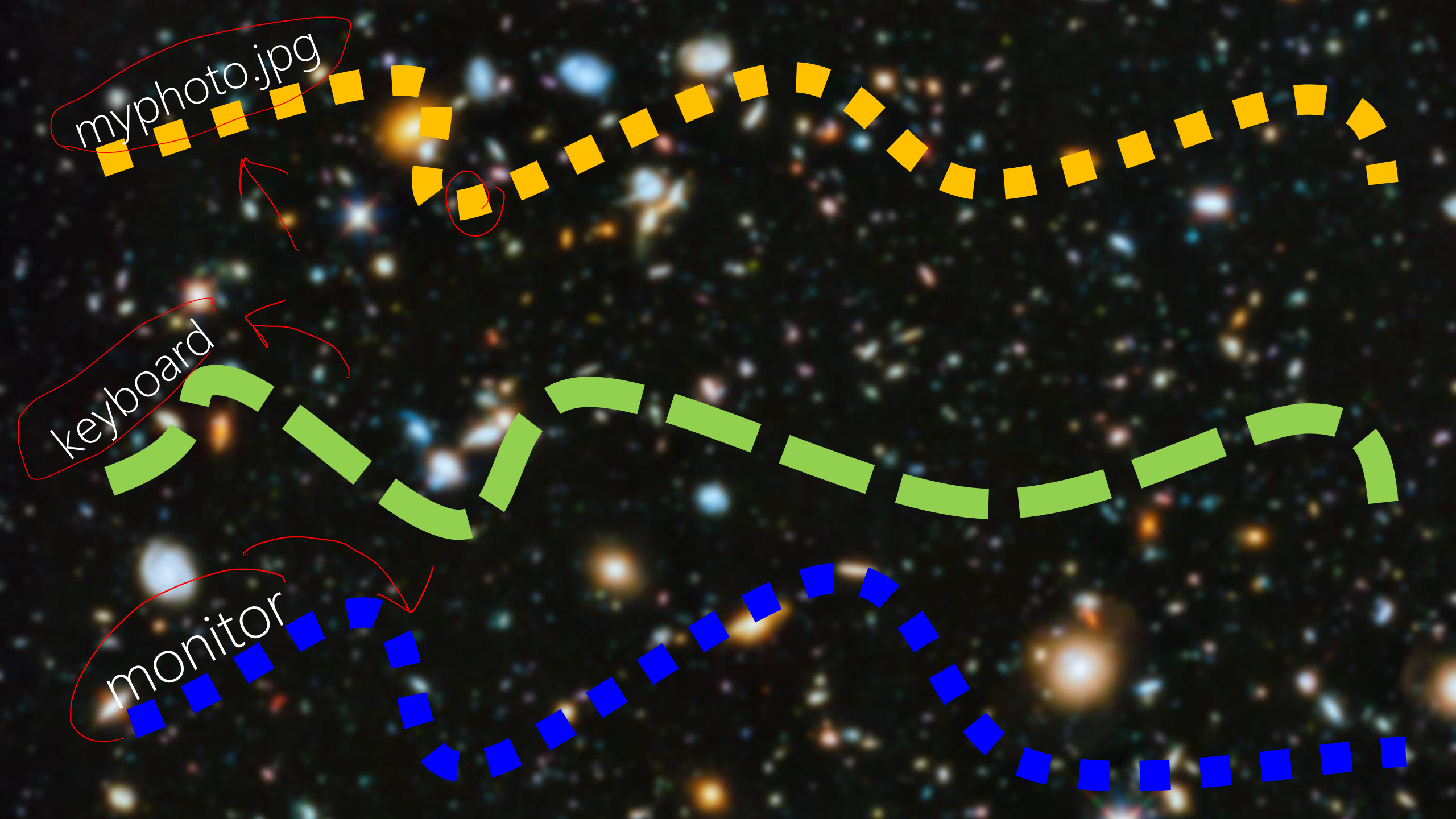
Play (k)

Device == File

Keyboard: Read Only File
Monitor: Write Only File
Printer: Write Only File
Touchscreen: Read Write File
NIC: Read Write File
HDD: Read Write Files
USB: Read Write Files

What else?

File System: High-Level


myphoto.jpg
keyboard
monitor

Operator - The Matrix (1999), Lana & Lilly Wachowski

Operation

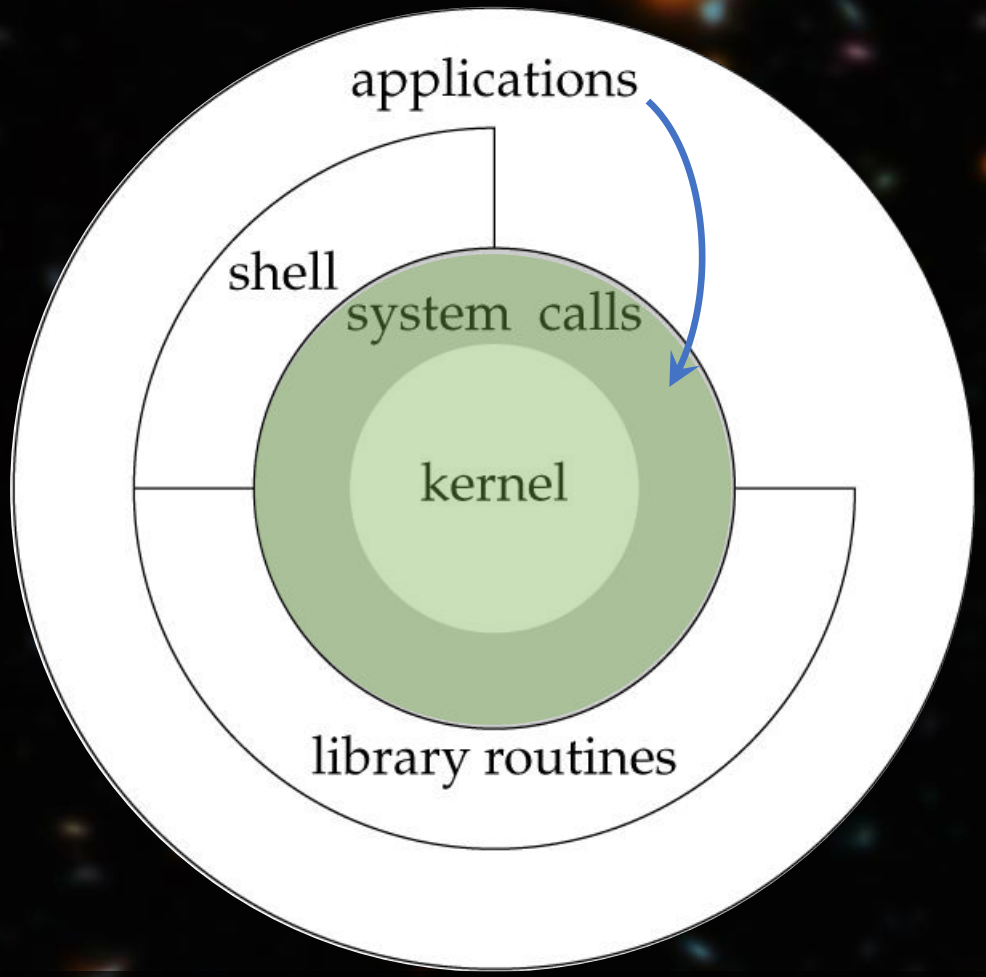
What do you expect from a kernel about string of bytes | device | file?

File System: High-Level

Create a New One
Open an Existing One
Write to an Opened One
Read from an Opened One
Move Forward/Backward in an Opened One
Delete an Existing One
Check the Existence of One
Hide an Existing One
Prevent Others to Open an Existing One
Prevent Others to Write to an Existing One

What else?

File System: High-Level



Header	FreeBSD 8.0	Linux 3.2.0	Mac OS X 10.6.8	Solaris 10	Description
<aio.h>	•	•	•	•	asynchronous I/O
<cpio.h>	•	•	•	•	cpio archive values
<dirent.h>	•	•	•	•	directory entries (Section 4.22)
<dlfcn.h>	•	•	•	•	dynamic linking
<fcntl.h>	•	•	•	•	file control (Section 3.14)
<fnmatch.h>	•	•	•	•	filename-matching types
<glob.h>	•	•	•	•	pathname pattern-matching and generation
<grp.h>	•	•	•	•	group file (Section 6.4)
<iconv.h>	•	•	•	•	codeset conversion utility
<langinfo.h>	•	•	•	•	language information constants
<monetary.h>	•	•	•	•	monetary types and functions
<netdb.h>	•	•	•	•	network database operations
<nl_types.h>	•	•	•	•	message catalogs
<poll.h>	•	•	•	•	poll function (Section 14.4.2)
<pthread.h>	•	•	•	•	threads (Chapters 11 and 12)
<pwd.h>	•	•	•	•	password file (Section 6.2)
<regex.h>	•	•	•	•	regular expressions
<sched.h>	•	•	•	•	execution scheduling
<semaphore.h>	•	•	•	•	semaphores
<strings.h>	•	•	•	•	string operations
<tar.h>	•	•	•	•	tar archive values
<termios.h>	•	•	•	•	terminal I/O (Chapter 18)
<unistd.h>	•	•	•	•	symbolic constants
<wordexp.h>	•	•	•	•	word-expansion definitions
<arpa/inet.h>	•	•	•	•	Internet definitions (Chapter 16)
<net/if.h>	•	•	•	•	socket local interfaces (Chapter 16)
<netinet/in.h>	•	•	•	•	Internet address family (Section 16.3)
<netinet/tcp.h>	•	•	•	•	Transmission Control Protocol definitions
<sys/mman.h>	•	•	•	•	memory management declarations
<sys/select.h>	•	•	•	•	select function (Section 14.4.1)
<sys/socket.h>	•	•	•	•	sockets interface (Chapter 16)
<sys/stat.h>	•	•	•	•	file status (Chapter 4)
<sys/statvfs.h>	•	•	•	•	file system information
<sys/times.h>	•	•	•	•	process times (Section 8.17)
<sys/types.h>	•	•	•	•	primitive system data types (Section 2.8)
<sys/un.h>	•	•	•	•	UNIX domain socket definitions (Section 17.2)
<sys/utsname.h>	•	•	•	•	system name (Section 6.9)
<sys/wait.h>	•	•	•	•	process control (Section 8.6)

creat

POSIX

```
#include <fcntl.h>
int creat(const char *path, mode_t mode);
```

non-negative number (file descriptor) for write-only if OK

–1 on error

creat

System Call

No static linking! Why?

Sort of a dynamic linking. Instead of call to another process, call to Kernel.

```
#include <fcntl.h>
int creat(const char *path, mode_t mode);
```

non-negative number (file descriptor) for write-only if OK

–1 on error

'test.txt'

creat

Name of the File (Device) to Create

Create a new keyboard | monitor | ... ?!

```
#include <fcntl.h>
int creat(const char *path, mode_t mode);
```

non-negative number (file descriptor) for write-only if OK

–1 on error

creat

Permission to Access the Created File (Device)

R W RW

```
#include <fcntl.h>
int creat(const char *path, mode_t mode);
```

non-negative number (file descriptor) for write-only if OK

–1 on error

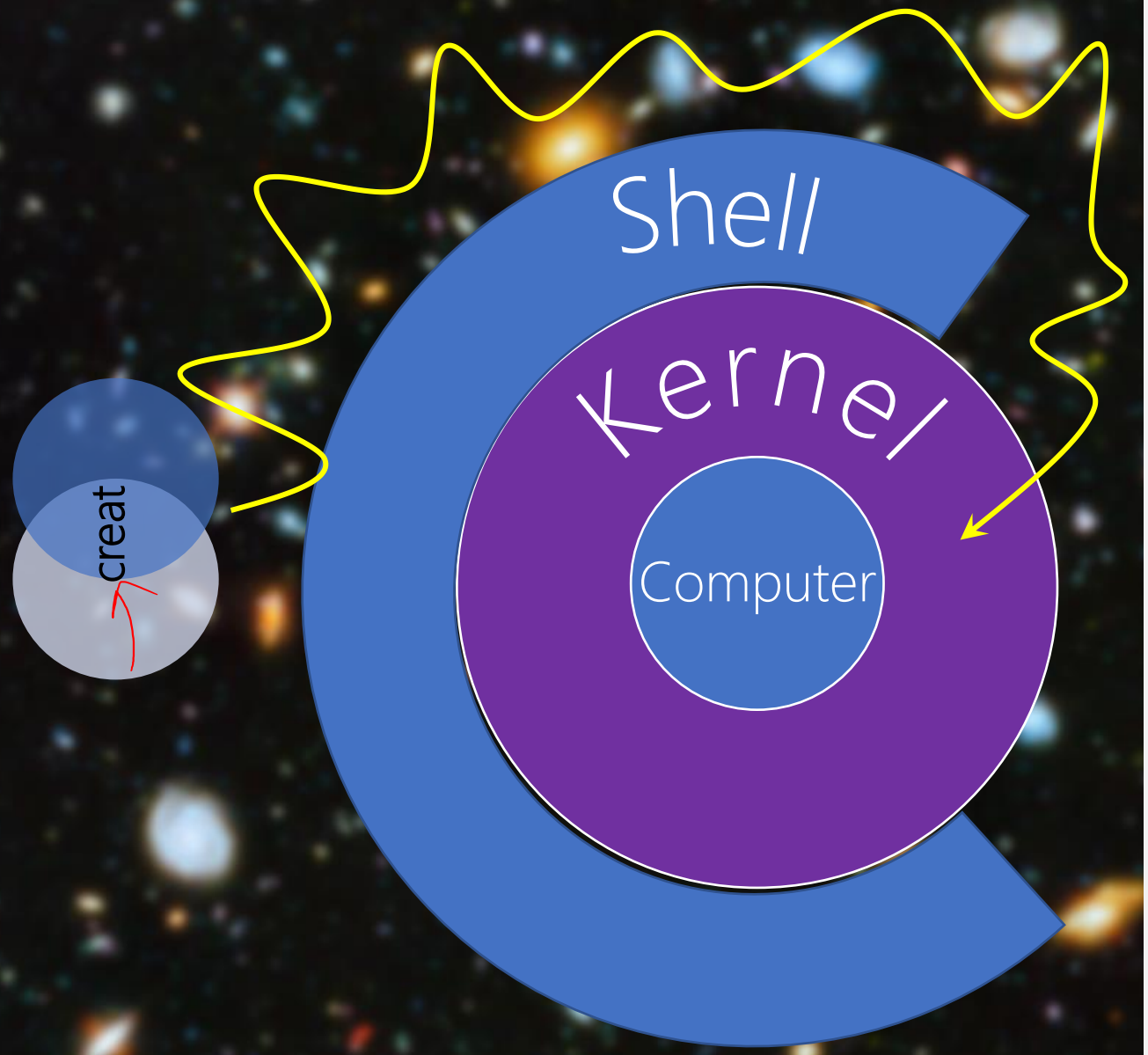
creat

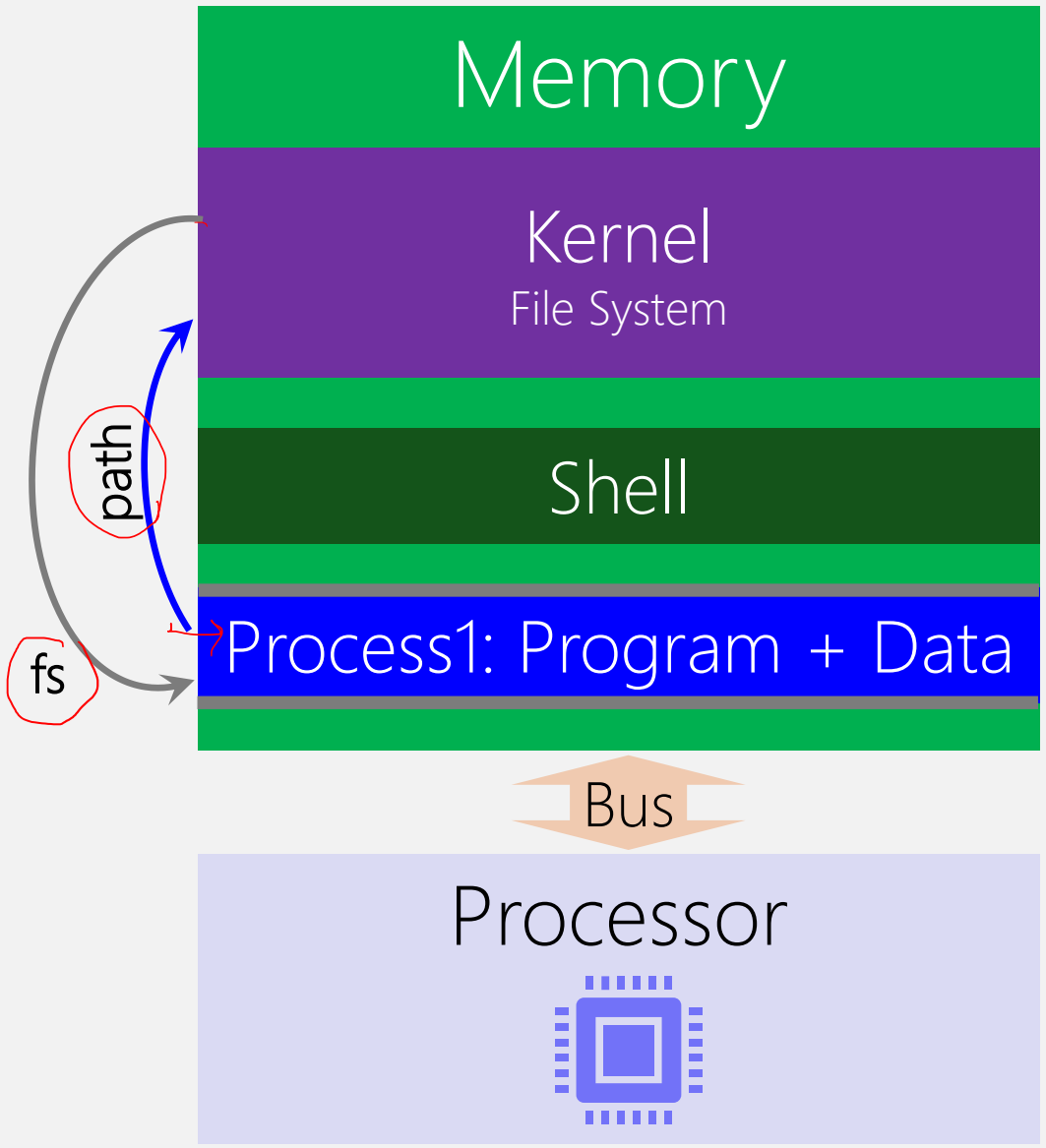
File Descriptor (fs)

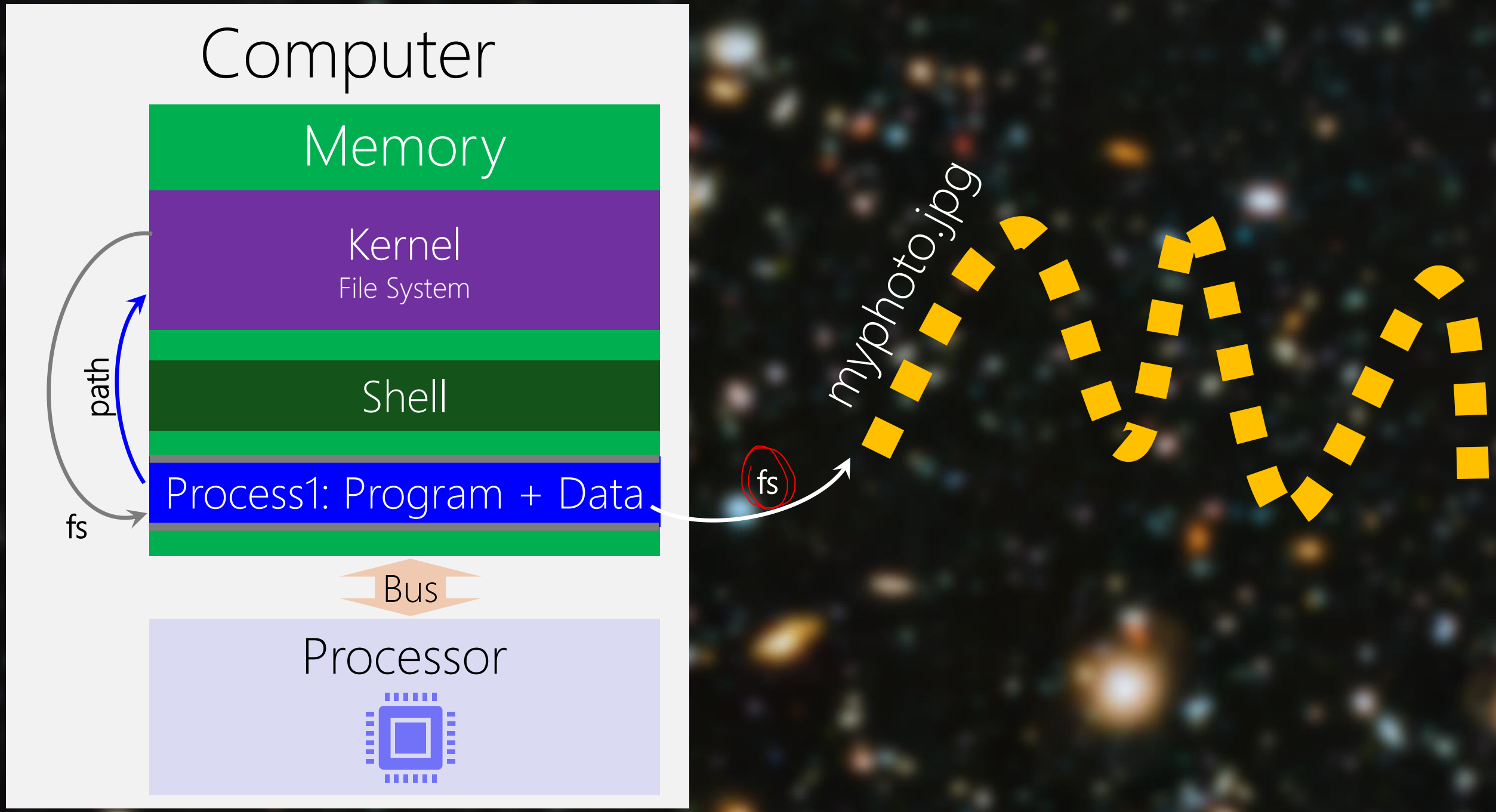
```
#include <fcntl.h>
int creat(const char *path, mode_t mode);
```

non-negative number (file descriptor) for write-only if OK

−1 on error


Shell
Kernel
Computer
creat


Computer
Memory
Kernel
File System
Shell
Process1: Program + Data
path
fs
Bus
Processor


Computer
Memory
Kernel
File System
Shell
Process1: Program + Data
path
fs
Bus
Processor
fs
myphoto.jpg

File Descriptor (fs)

Number does not Matter, Connection Matters

Imagine a dynamic phone#, dynamic postal code, dynamic ip (DHCP) for wifi access

The Only Way In Or Out Is Through Phone Connection

Trinity escapes from Agents
- The Matrix (1999), Lana & Lilly Wachowski

Identifier

Uniquely Identifies an Entity, from Birth to ever (even after death)

SIN# is Identifier or Descriptor?
Phone# is Identifier or Descriptor?

File Descriptor (fs) != File Identifier

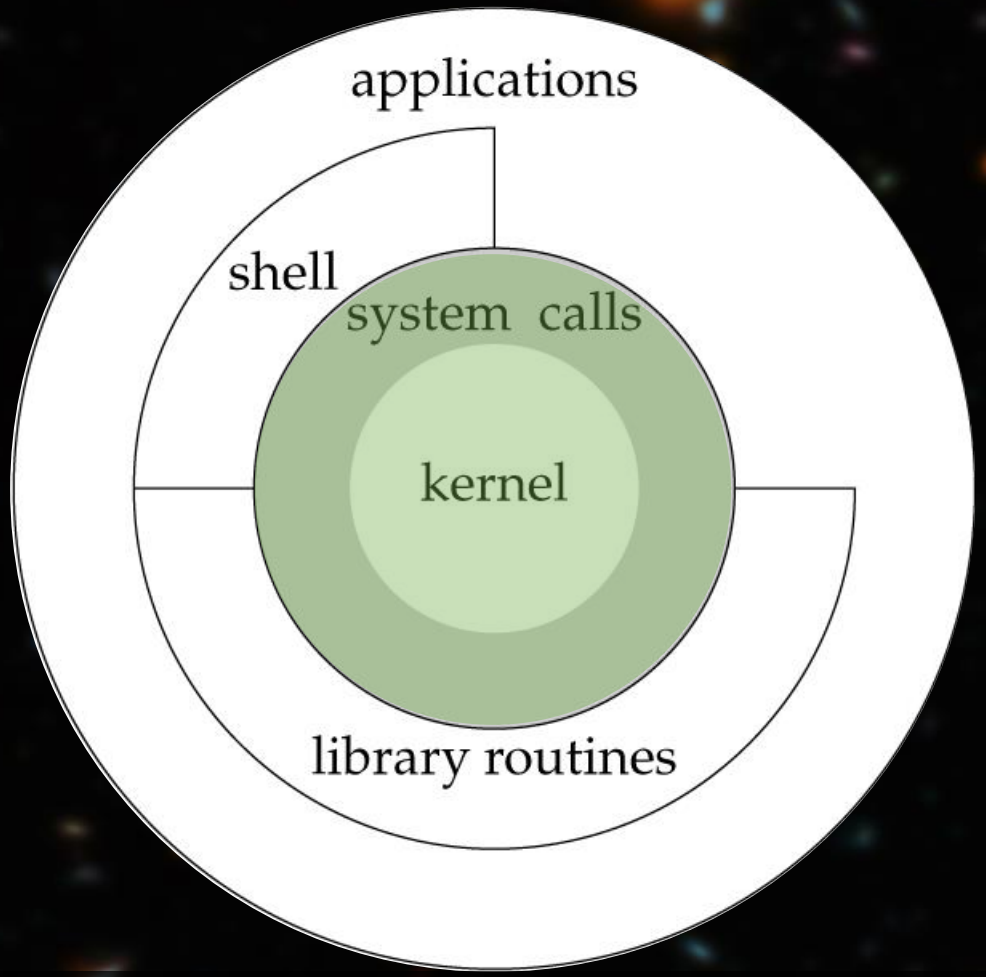
Because Kernel reuse them for other files and devices, when available!

File Descriptor (fs)

fd ∈ [0 : OPEN_MAX – 1]

```
unistd.h
- #define OPEN_MAX 20
- #define OPEN_MAX 63
```

- No limit, maximum integer number supported by the system



Header	FreeBSD 8.0	Linux 3.2.0	Mac OS X 10.6.8	Solaris 10	Description
<aio.h>	•	•	•	•	asynchronous I/O
<cpio.h>	•	•	•	•	cpio archive values
<dirent.h>	•	•	•	•	directory entries (Section 4.22)
<dlfcn.h>	•	•	•	•	dynamic linking
<fcntl.h>	•	•	•	•	file control (Section 3.14)
<fnmatch.h>	•	•	•	•	filename-matching types
<glob.h>	•	•	•	•	pathname pattern-matching and generation
<grp.h>	•	•	•	•	group file (Section 6.4)
<iconv.h>	•	•	•	•	codeset conversion utility
<langinfo.h>	•	•	•	•	language information constants
<monetary.h>	•	•	•	•	monetary types and functions
<netdb.h>	•	•	•	•	network database operations
<nl_types.h>	•	•	•	•	message catalogs
<poll.h>	•	•	•	•	poll function (Section 14.4.2)
<pthread.h>	•	•	•	•	threads (Chapters 11 and 12)
<pwd.h>	•	•	•	•	password file (Section 6.2)
<regex.h>	•	•	•	•	regular expressions
<sched.h>	•	•	•	•	execution scheduling
<semaphore.h>	•	•	•	•	semaphores
<strings.h>	•	•	•	•	string operations
<tar.h>	•	•	•	•	tar archive values
<termios.h>	•	•	•	•	terminal I/O (Chapter 18)
<unistd.h>	•	•	•	•	symbolic constants
<wordexp.h>	•	•	•	•	word-expansion definitions
<arpa/inet.h>	•	•	•	•	Internet definitions (Chapter 16)
<net/if.h>	•	•	•	•	socket local interfaces (Chapter 16)
<netinet/in.h>	•	•	•	•	Internet address family (Section 16.3)
<netinet/tcp.h>	•	•	•	•	Transmission Control Protocol definitions
<sys/mman.h>	•	•	•	•	memory management declarations
<sys/select.h>	•	•	•	•	select function (Section 14.4.1)
<sys/socket.h>	•	•	•	•	sockets interface (Chapter 16)
<sys/stat.h>	•	•	•	•	file status (Chapter 4)
<sys/statvfs.h>	•	•	•	•	file system information
<sys/times.h>	•	•	•	•	process times (Section 8.17)
<sys/types.h>	•	•	•	•	primitive system data types (Section 2.8)
<sys/un.h>	•	•	•	•	UNIX domain socket definitions (Section 17.2)
<sys/utsname.h>	•	•	•	•	system name (Section 6.9)
<sys/wait.h>	•	•	•	•	process control (Section 8.6)

File Descriptor (fs)

STDIN_FILENO, STDOUT_FILENO, STDERR_FILENO

e.g.: 10 , 12 , 504

File Descriptor (fs)

STDIN_FILENO, STDOUT_FILENO, STDERR_FILENO

```
unistd.h
#define STDIN_FILENO 0
#define STDOUT_FILENO 1
#define STDERR_FILENO 2
```

File Descriptor (fs)

STDIN_FILENO, STDOUT_FILENO, STDERR_FILENO

```
unistd.h
#define STDIN_FILENO 0
#define STDOUT_FILENO 1
#define STDERR_FILENO 2
```

Keyboard, Mouse, File, ...

Monitor, Printer, File, ...

Monitor, Printer, File, ...

creat

~~STDIN_FILENO~~, STDOUT_FILENO, STDERR_FILENO

```
#include <fcntl.h>
int creat(const char *path, mode_t mode);
```

non-negative number for write-only if OK

–1 on error

write

POSIX

```
#include <unistd.h>
ssize_t write(int fd, const void *buf, size_t nbytes);
```

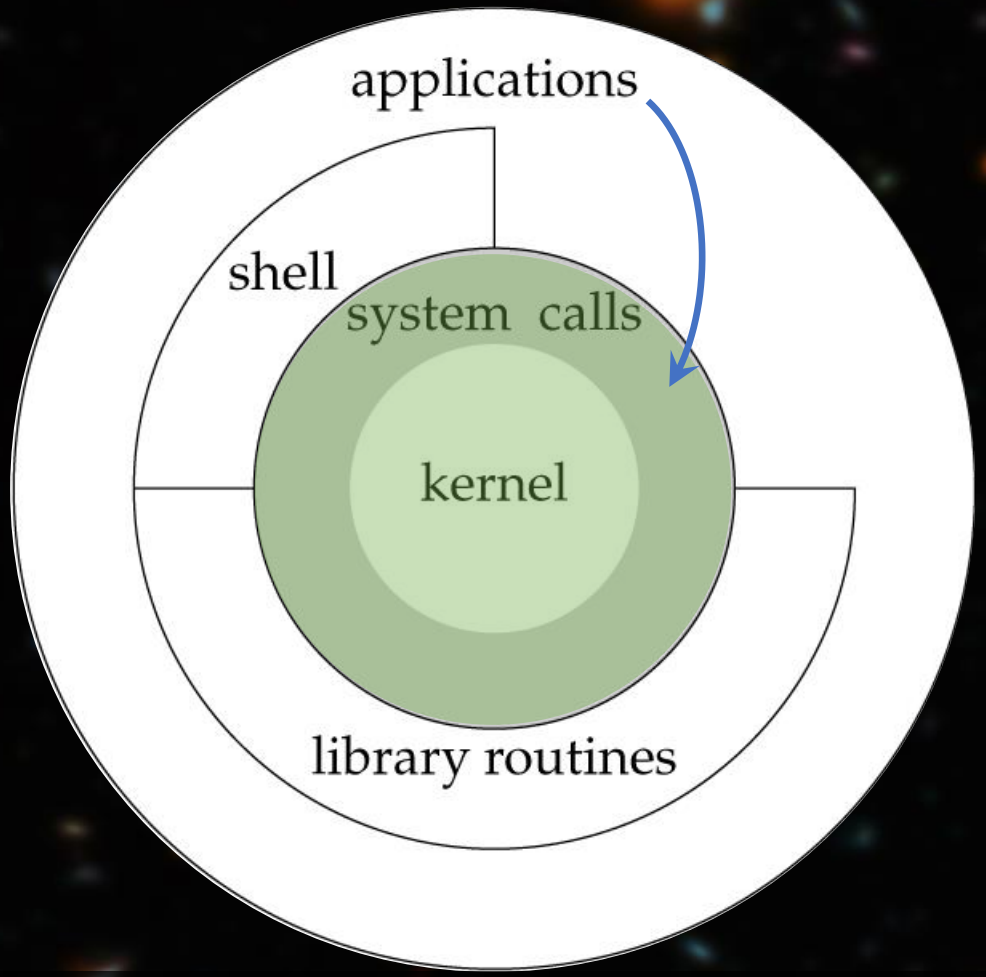
number of bytes written if OK, –1 on error

write

System Call

```
#include <unistd.h>
ssize_t write(int fd, const void *buf, size_t nbytes);
```

number of bytes written if OK, –1 on error



Header	FreeBSD 8.0	Linux 3.2.0	Mac OS X 10.6.8	Solaris 10	Description
<aio.h>	•	•	•	•	asynchronous I/O
<cpio.h>	•	•	•	•	cpio archive values
<dirent.h>	•	•	•	•	directory entries (Section 4.22)
<dlfcn.h>	•	•	•	•	dynamic linking
<fcntl.h>	•	•	•	•	file control (Section 3.14)
<fnmatch.h>	•	•	•	•	filename-matching types
<glob.h>	•	•	•	•	pathname pattern-matching and generation
<grp.h>	•	•	•	•	group file (Section 6.4)
<iconv.h>	•	•	•	•	codeset conversion utility
<langinfo.h>	•	•	•	•	language information constants
<monetary.h>	•	•	•	•	monetary types and functions
<netdb.h>	•	•	•	•	network database operations
<nl_types.h>	•	•	•	•	message catalogs
<poll.h>	•	•	•	•	poll function (Section 14.4.2)
<pthread.h>	•	•	•	•	threads (Chapters 11 and 12)
<pwd.h>	•	•	•	•	password file (Section 6.2)
<regex.h>	•	•	•	•	regular expressions
<sched.h>	•	•	•	•	execution scheduling
<semaphore.h>	•	•	•	•	semaphores
<strings.h>	•	•	•	•	string operations
<tar.h>	•	•	•	•	tar archive values
<termios.h>	•	•	•	•	terminal I/O (Chapter 18)
<unistd.h>	•	•	•	•	symbolic constants
<wordexp.h>	•	•	•	•	word-expansion definitions
<arpa/inet.h>	•	•	•	•	Internet definitions (Chapter 16)
<net/if.h>	•	•	•	•	socket local interfaces (Chapter 16)
<netinet/in.h>	•	•	•	•	Internet address family (Section 16.3)
<netinet/tcp.h>	•	•	•	•	Transmission Control Protocol definitions
<sys/mman.h>	•	•	•	•	memory management declarations
<sys/select.h>	•	•	•	•	select function (Section 14.4.1)
<sys/socket.h>	•	•	•	•	sockets interface (Chapter 16)
<sys/stat.h>	•	•	•	•	file status (Chapter 4)
<sys/statvfs.h>	•	•	•	•	file system information
<sys/times.h>	•	•	•	•	process times (Section 8.17)
<sys/types.h>	•	•	•	•	primitive system data types (Section 2.8)
<sys/un.h>	•	•	•	•	UNIX domain socket definitions (Section 17.2)
<sys/utsname.h>	•	•	•	•	system name (Section 6.9)
<sys/wait.h>	•	•	•	•	process control (Section 8.6)

write

fd: Write to What File (Device)

```
#include <unistd.h>
ssize_t write(int fd, const void *buf, size_t nbytes);
```

number of bytes written if OK, –1 on error

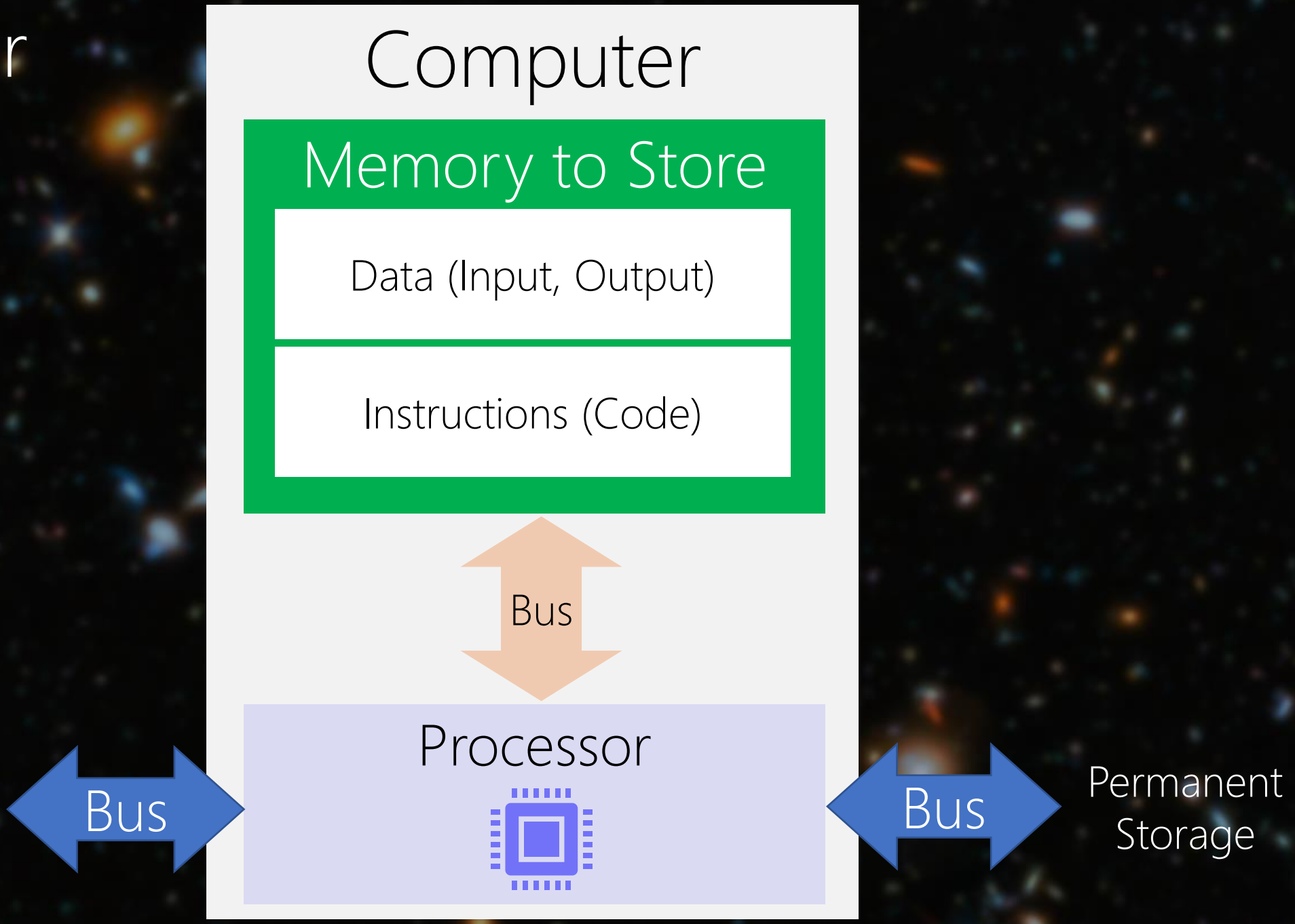
write

buf: Write from this Location in Memory to the File (Device)

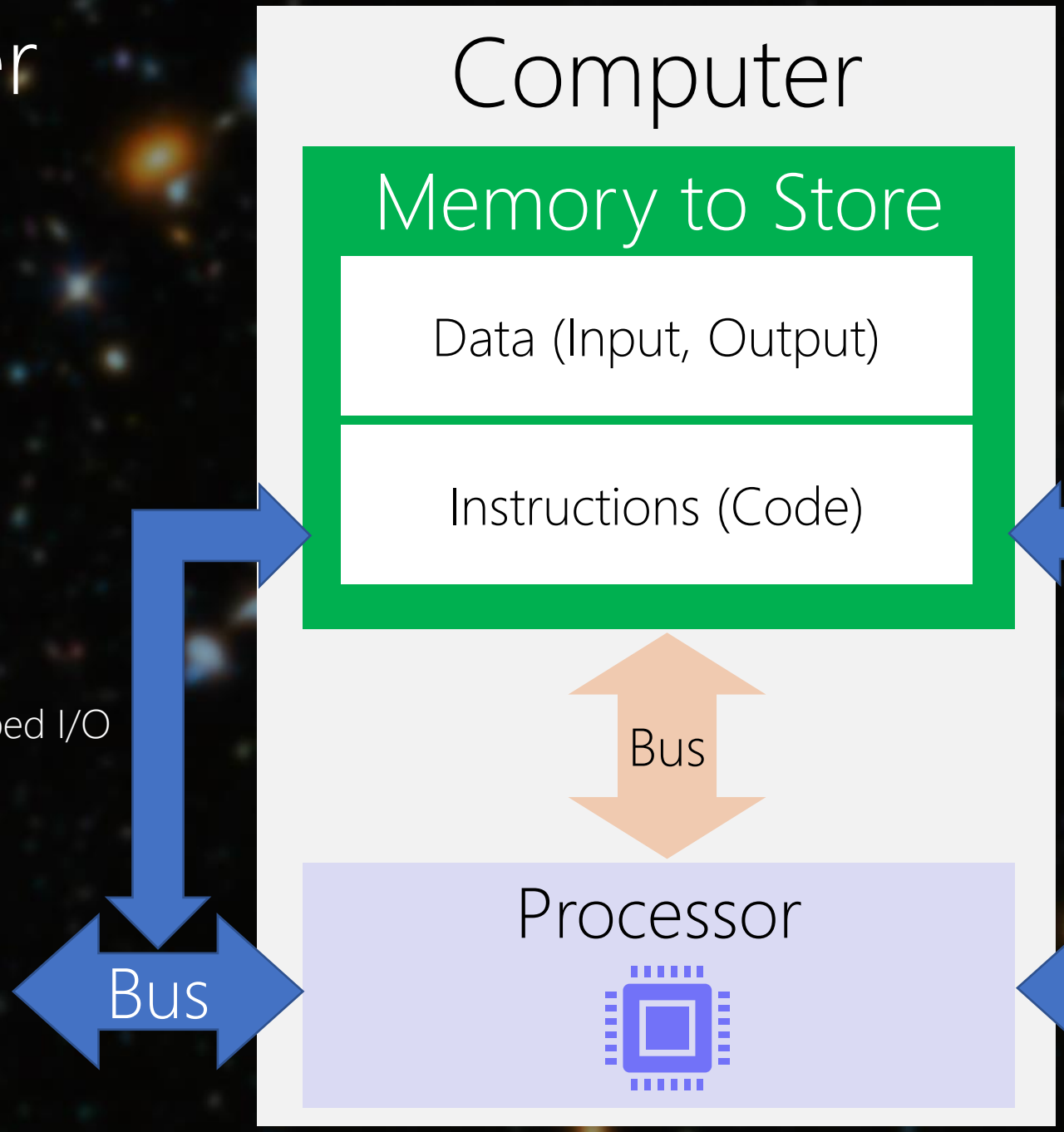
```
#include <unistd.h>
ssize_t write(int fd, const void *buf, size_t nbytes);
```

number of bytes written if OK, –1 on error

Computer System



Computer System



write

void *: Type of Data does not Matter (char, int, float, ...)

```
#include <unistd.h>
ssize_t write(int fd, const void *buf, size_t nbytes);
```

number of bytes written if OK, –1 on error

write

nbytes: Write this Amount of Byte to the File (Device)
Your Responsibility to Provide a Correct Conversion to Number of Bytes!
How about 2 bits (not a whole Byte)?

```
#include <unistd.h>
ssize_t write(int fd, const void *buf, size_t nbytes);
```

number of bytes written if OK, –1 on error

write

nbytes: Write this Amount of Byte to the File (Device)
Your Responsibility to Provide a Correct Conversion to Number of Bytes!

```
#include <unistd.h>
ssize_t write(int fd, const void *buf, size_t nbytes);
```

number of bytes written if OK, –1 on error

typedef

ssize_t, size_t, ..., and many other data types

```
#include <sys/types.h>
```

https://pubs.opengroup.org/onlinepubs/009604599/basedefs/sys/types.h.html

typedef

ssize_t, size_t, ..., and many other data types

```
#include <sys/types.h>
typedef size_t unsigned long
typedef ssize_t signed long //for -1 if fails
```

https://www.ibm.com/docs/en/zos/2.2.0?topic=files-systypesh

close

POSIX

```
#include <unistd.h>
int close(int fd);
```

0 if OK, –1 on error

close

fd: Releases the File Descriptor (Available for Reuse by Kernel)
No Further Access to the File (Device)

```
#include <unistd.h>
int close(int fd);
```

0 if OK, –1 on error

close

Sometimes Optional, but only Sometimes!

When a process terminates, all of its open files are closed automatically by the kernel.
That is all the File Descriptors (fs) are released.
You can take advantage of this fact and don't explicitly close open files in your programs (not recommended!)

```
hfani@alpha:~$ vi create_file_system_call.c
```

```c
#include <fcntl.h>
#include <unistd.h>

#include <sys/types.h>
#include <string.h>

#include <stdio.h>
void main(void){
        int fd;//file descriptor
        mode_t mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH;//for permisison settings
        char *filename = "./my_new_file.txt";

        fd = creat(filename, mode);
        printf("The file descriptor is: %d \n", fd);

        if(fd == -1){
                printf("Error in creating file!\n");
                return;
        }

        char buf[20];
        size_t nbytes;
        ssize_t bytes_written;

        strcpy(buf, "Hello File!\n");
        nbytes = strlen(buf);

        bytes_written = write(fd, buf, nbytes);
        if(bytes_written != nbytes){
                printf("Error in writing to the file!");
        }


        int result = close(fd);
        if(result == -1){
                printf("Error in closing the file!");
        }

}
```

```c
#include <fcntl.h>
#include <unistd.h>

#include <sys/types.h>
#include <string.h>

#include <stdio.h>
void main(void){
        int fd;//file descriptor
        mode_t mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH;//for permisison settings
        char *filename = "./my_new_file.txt";

        fd = creat(filename, mode);
        printf("The file descriptor is: %d \n", fd);

        if(fd == -1){
                printf("Error in creating file!\n");
                return;
        }

        char buf[20];
        size_t nbytes;
        ssize_t bytes_written;

        strcpy(buf, "Hello File!\n");
        nbytes = strlen(buf);

        bytes_written = write(fd, buf, nbytes);
        if(bytes_written != nbytes){
                printf("Error in writing to the file!");
        }


        int result = close(fd);
```

```c
#include <fcntl.h>
#include <unistd.h>

#include <sys/types.h>
#include <string.h>

#include <stdio.h>
void main(void){
        int fd;//file descriptor
        mode_t mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH;//for permission settings
        char *filename = "./my_new_file.txt";

        fd = creat(filename, mode);
        printf("The file descriptor is: %d \n", fd);

        if(fd == -1){
                printf("Error in creating file!\n");
                return;
        }

        char buf[20];
        size_t nbytes;
        ssize_t bytes_written;

        strcpy(buf, "Hello File!\n");
        nbytes = strlen(buf);

        bytes_written = write(fd, buf, nbytes);
        if(bytes_written != nbytes){
                printf("Error in writing to the file!");
        }


        int result = close(fd);
```

```c
#include <fcntl.h>
#include <unistd.h>

#include <sys/types.h>
#include <string.h>

#include <stdio.h>
void main(void){
        int fd;//file descriptor
        mode_t mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH;//for permission settings
        char *filename = "./my_new_file.txt";

        fd = creat(filename, mode);
        printf("The file descriptor is: %d \n", fd);

        if(fd == -1){
                printf("Error in creating file!\n");
                return;
        }

        char buf[20];
        size_t nbytes;
        ssize_t bytes_written;

        strcpy(buf, "Hello File!\n");
        nbytes = strlen(buf);

        bytes_written = write(fd, buf, nbytes);
        if(bytes_written != nbytes){
                printf("Error in writing to the file!");
        }


        int result = close(fd);
        if(result == -1){
                printf("Error in closing the file!\n");
```

```c
#include <fcntl.h>
#include <unistd.h>

#include <sys/types.h>
#include <string.h>

#include <stdio.h>
void main(void){
        int fd;//File descriptor
        mode_t mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH;//for permision settings
        char *filename = "./my_new_file.txt";

        fd = creat(filename, mode);
        printf("The file descriptor is: %d \n", fd);

        if(fd == -1){
                printf("Error in creating file!\n");
                return;
        }

        char buf[20];
        size_t nbytes;
        ssize_t bytes_written;

        strcpy(buf, "Hello File!\n");
        nbytes = strlen(buf);

        bytes_written = write(fd, buf, nbytes);
        if(bytes_written != nbytes){
                printf("Error in writing to the file!");
        }


        int result = close(fd);
        if(result == -1){
                printf("Error in closing the file!");
        }
```

```
hfani@alpha:~$ cc create_file_system_call.c -o create_file_system_call
hfani@alpha:~$ ./create_file_system_call
The file descriptor is: 3
hfani@alpha:~$
```

```
hfani@alpha:~$ vi my_new_file.txt

Hello File!
~
~
```