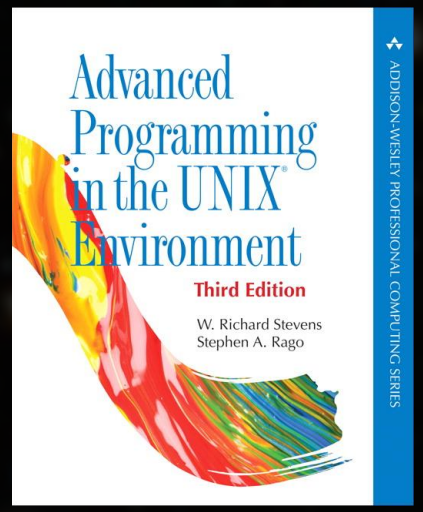
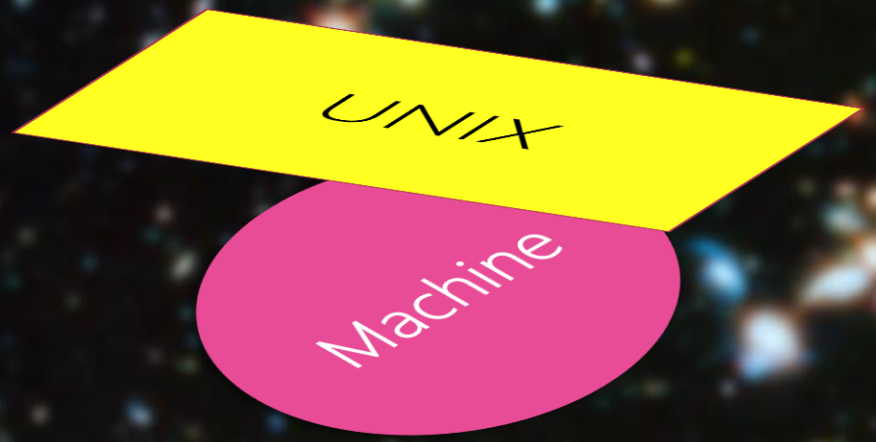
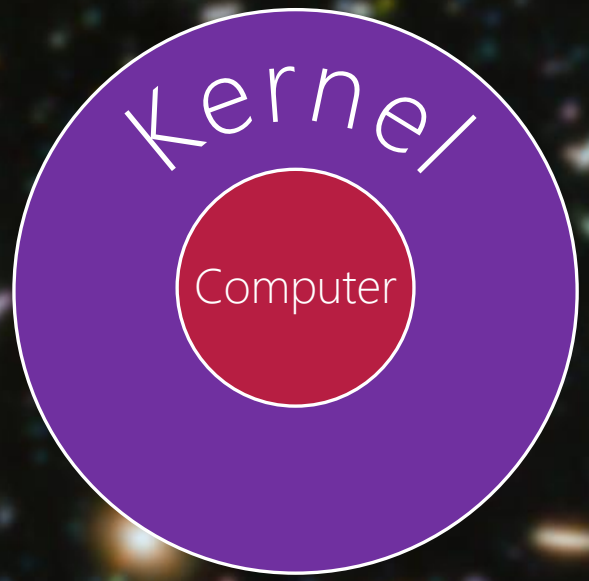
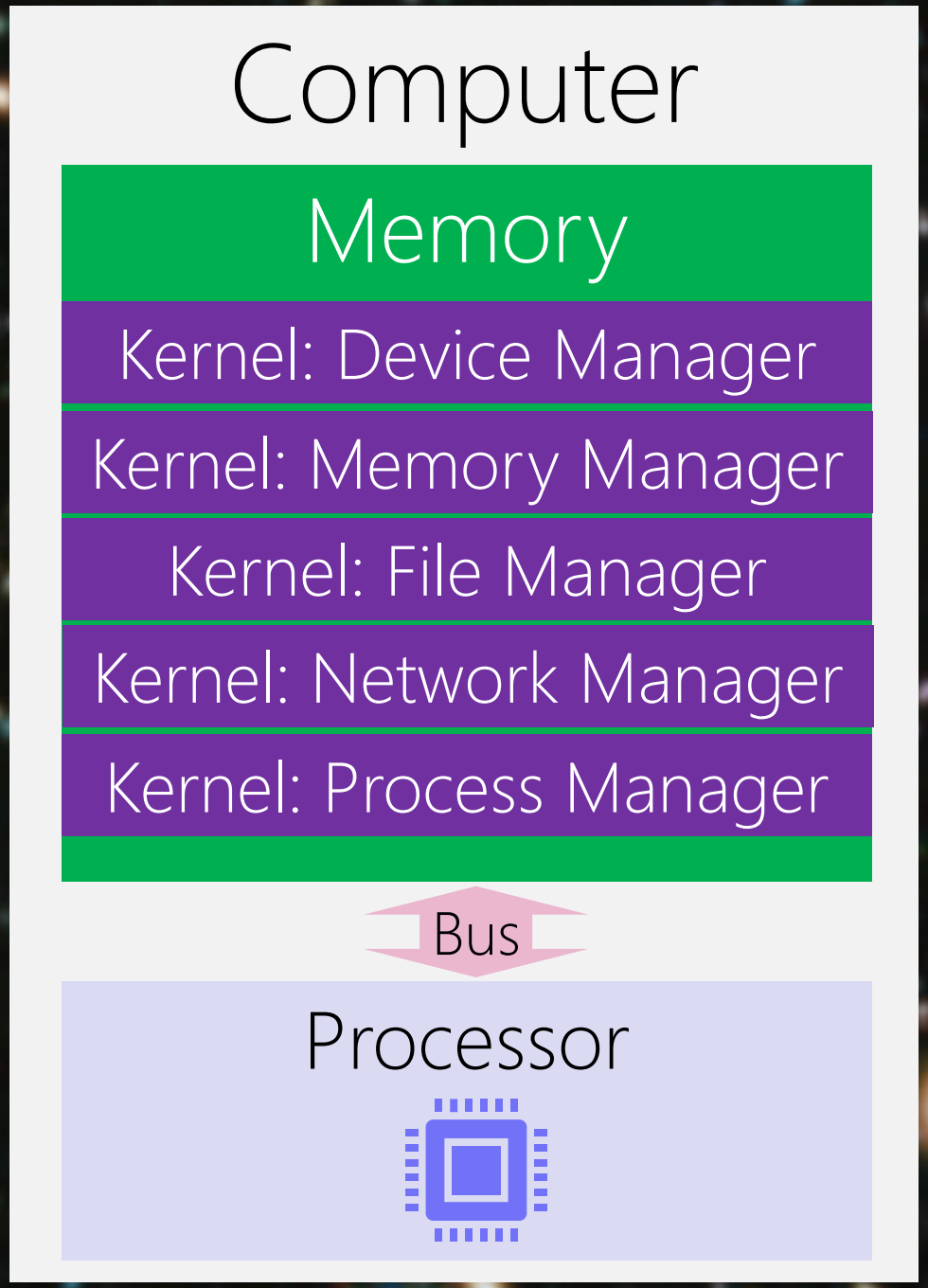
The Only Way In Or Out Is Through Phone Lines
- The Matrix (1999), Lana & Lilly Wachowski

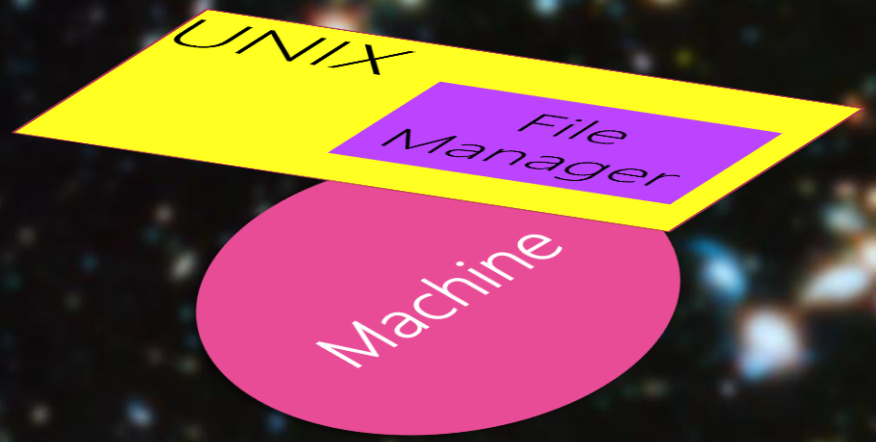


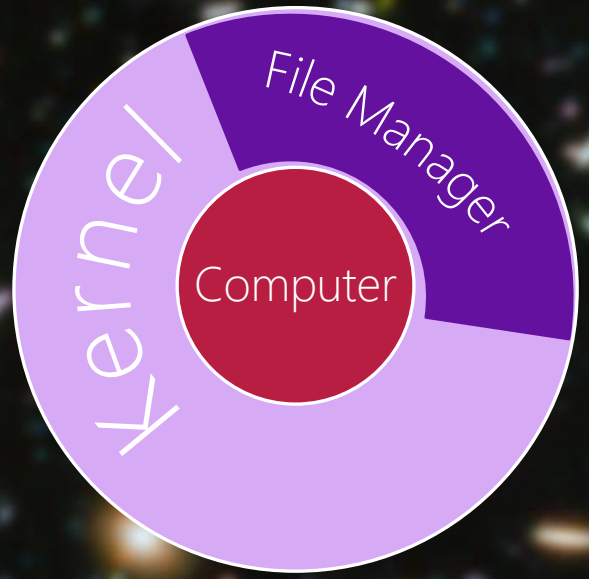
Chapter 03: File I/O

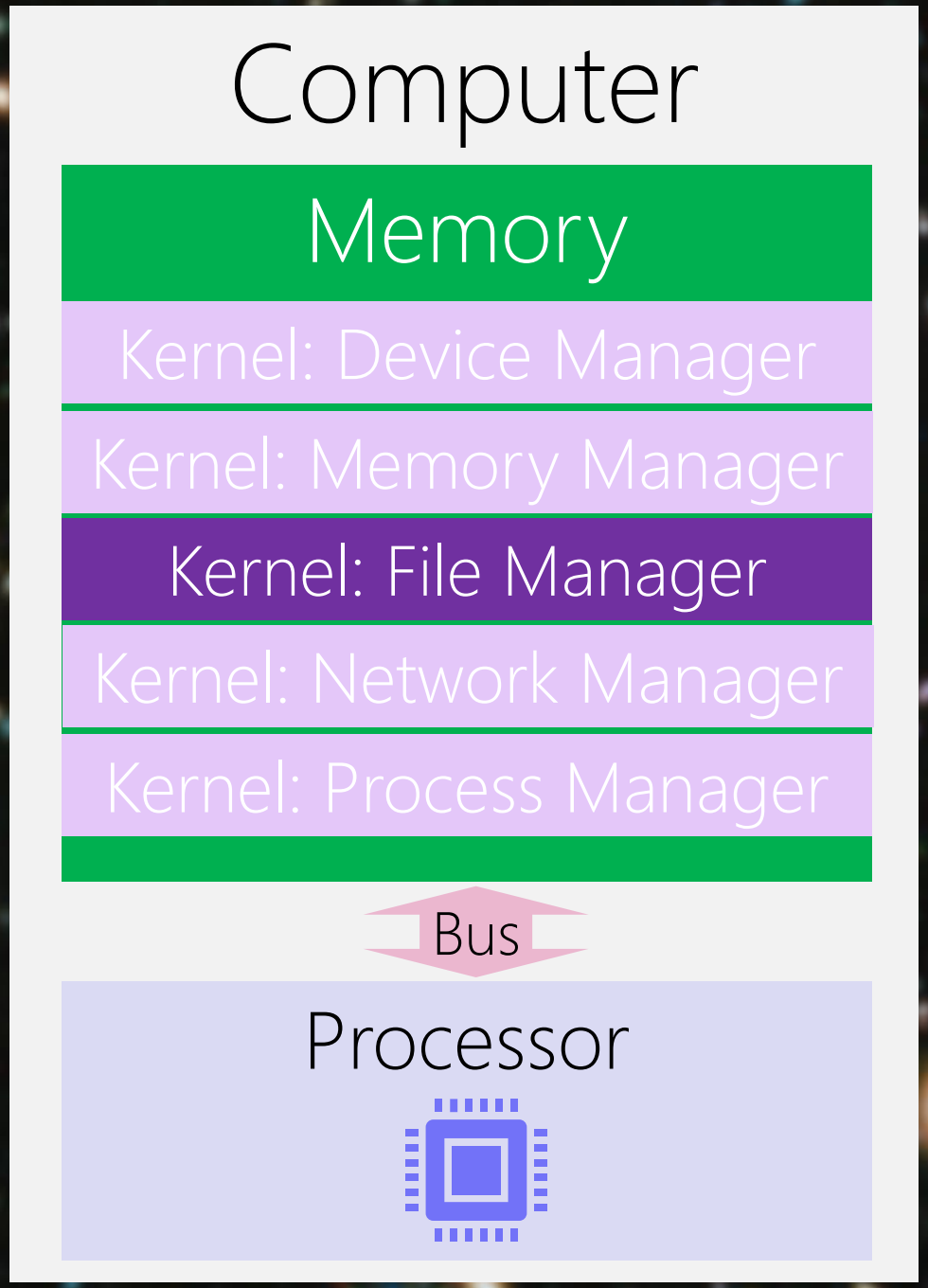

UNIX
Machine

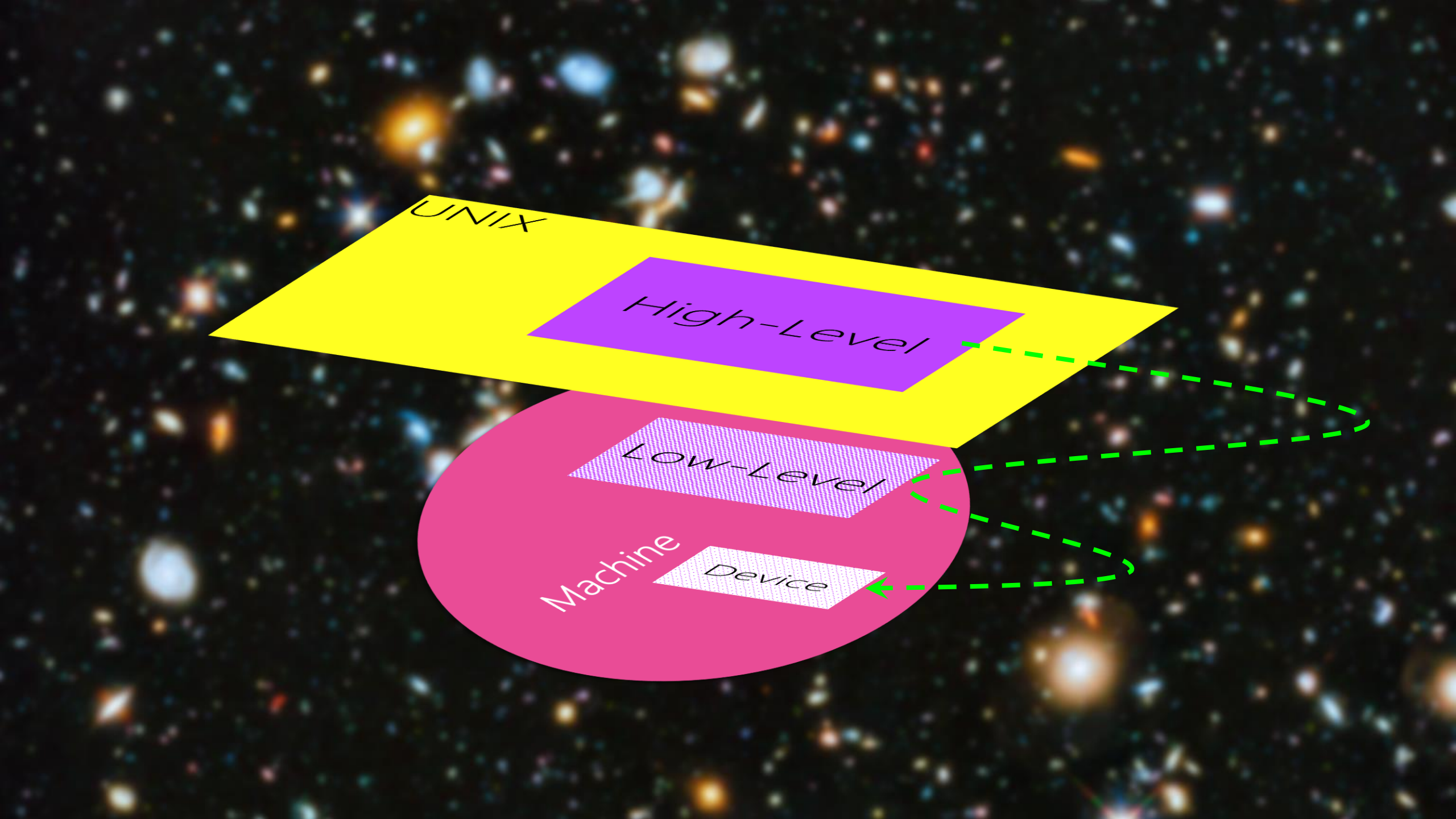

Kernel
Computer


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


UNIX
High-Level
Low-Level
Machine
Device

File Manager

widely known as File System

High-Level

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

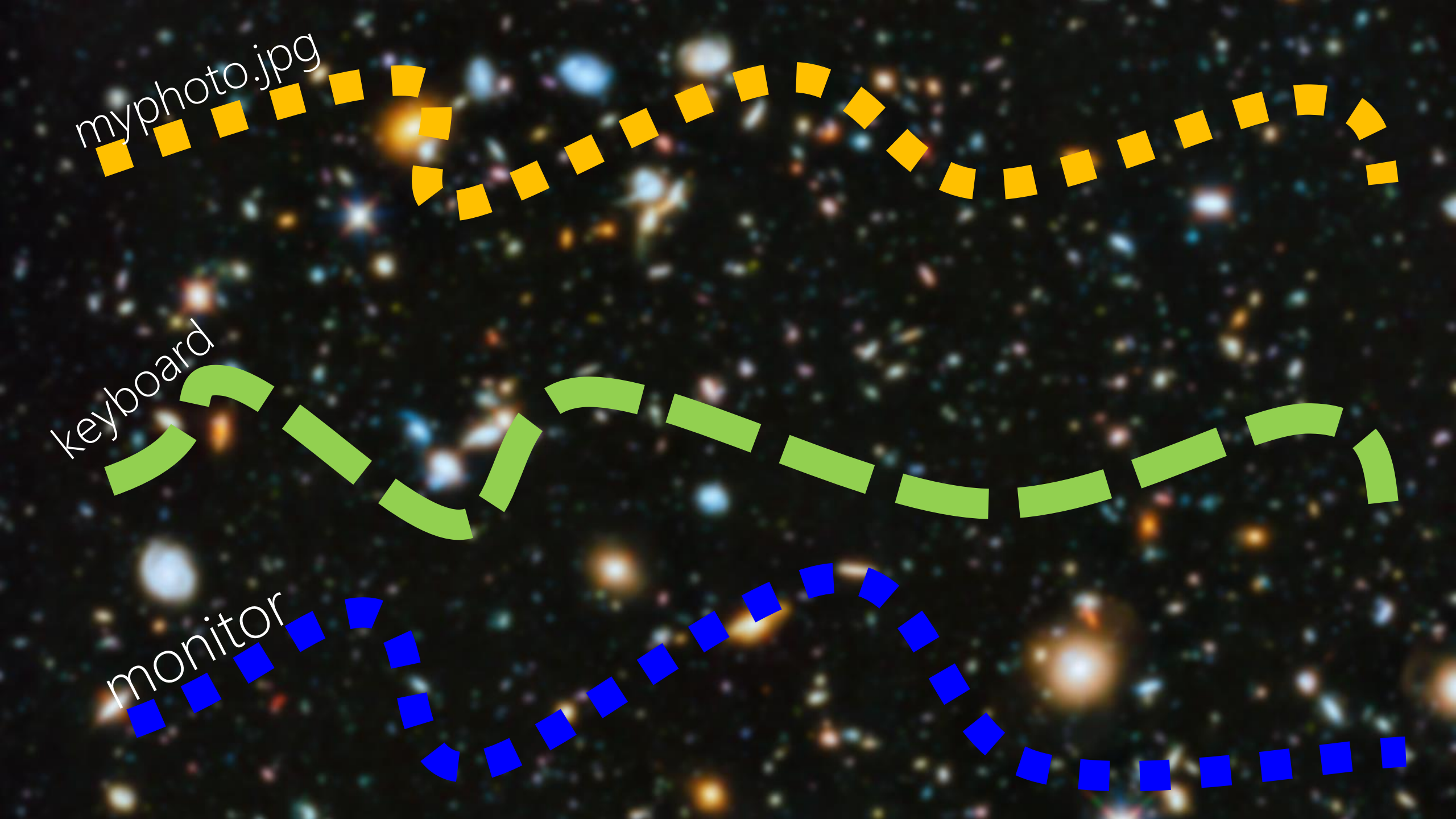
set of sub-devices

File System: High-Level

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

creat

POSIX

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

non-negative number for write-only if OK

−1 on error

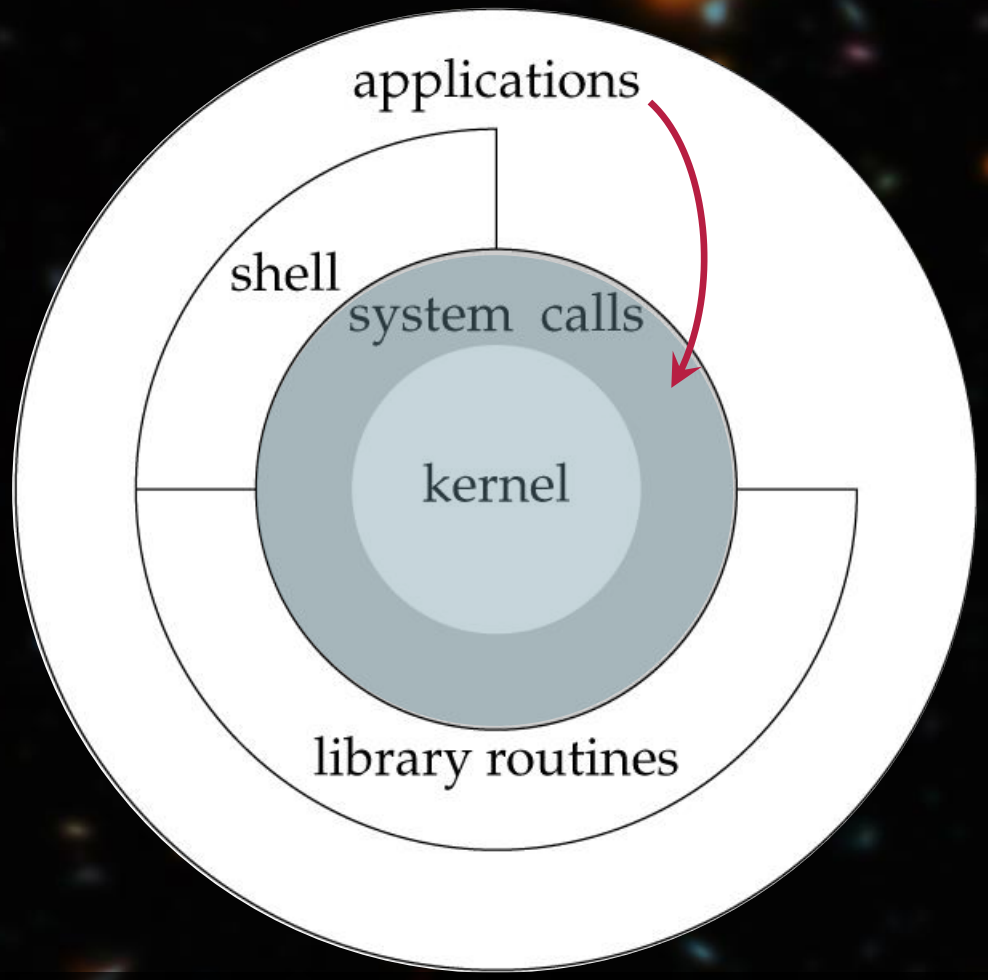
creat

System Call

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

non-negative number for write-only 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)

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 for write-only if OK
–1 on error

creat

Permission to Access the Created File (Device)

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

non-negative number for write-only if OK

–1 on error

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

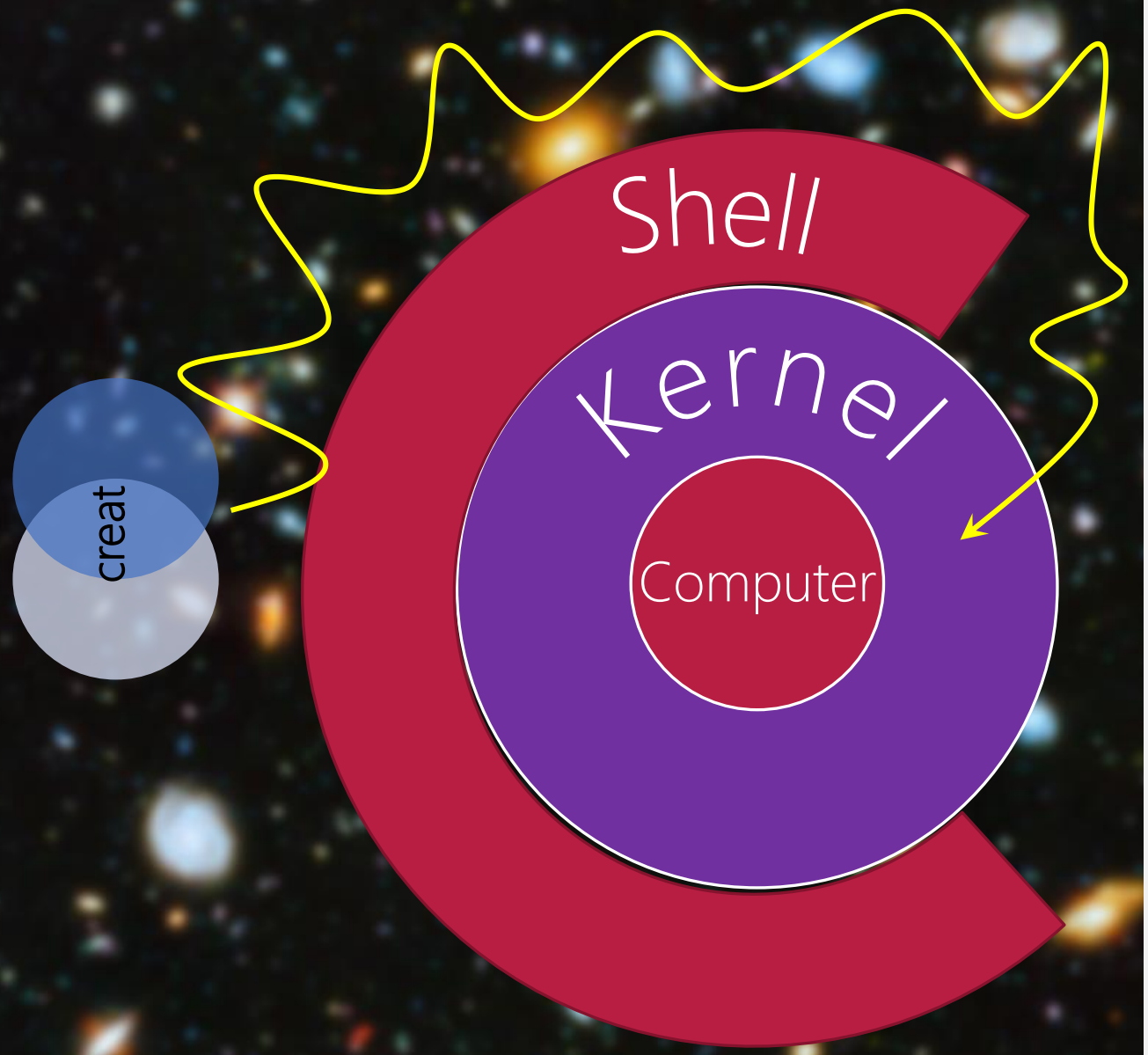
creat

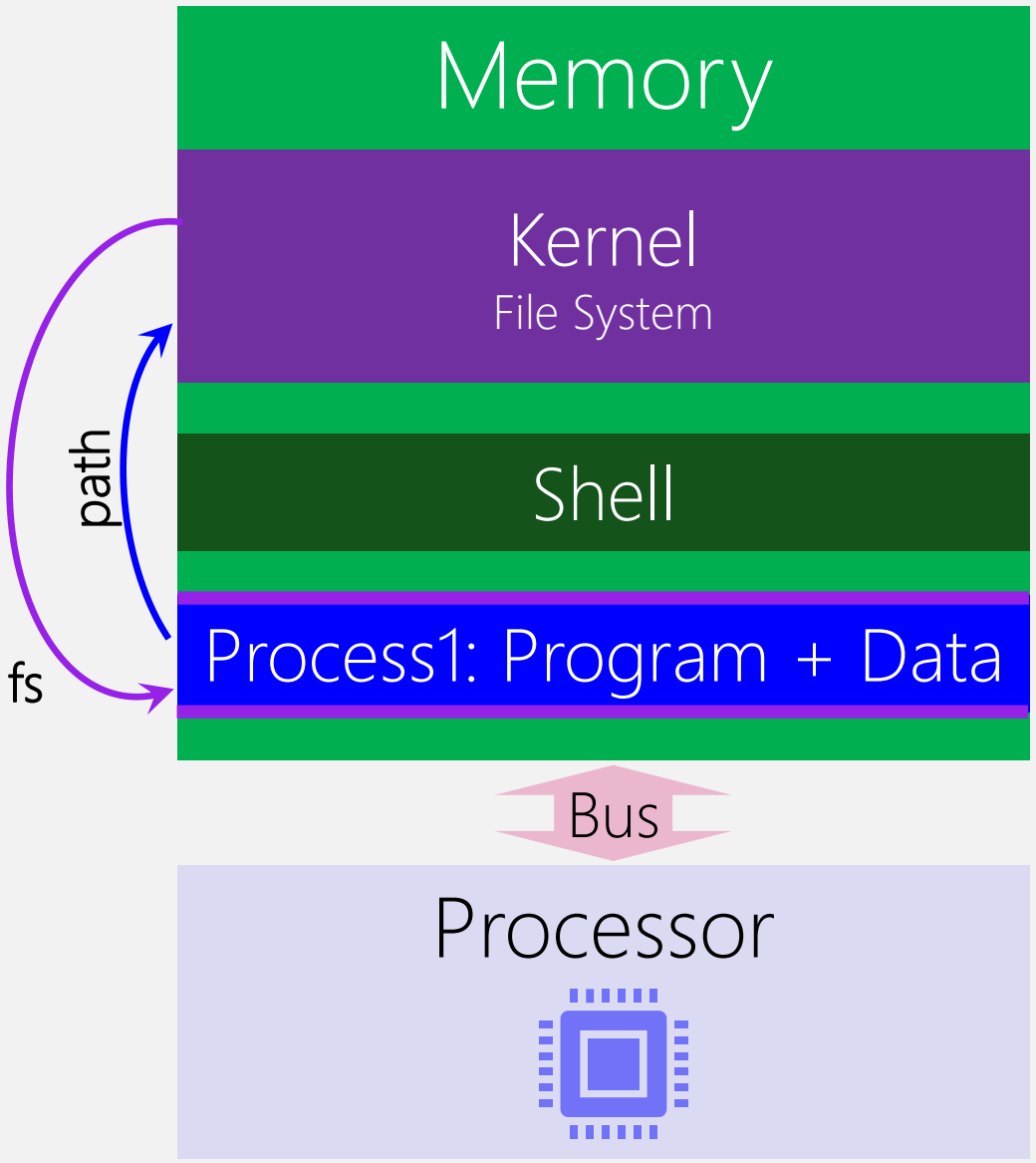
File Descriptor (fs)

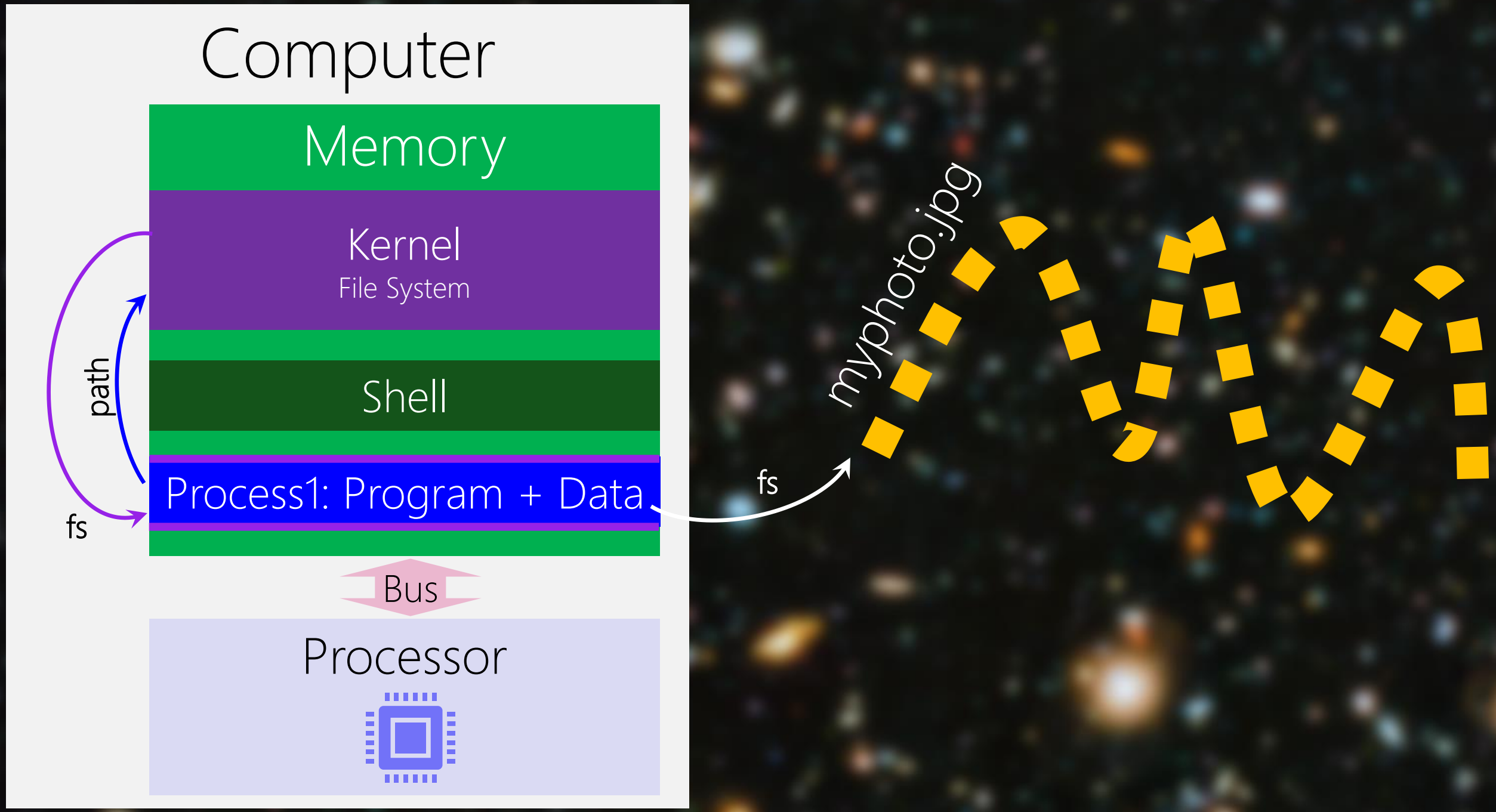
```
#include <fcntl.h>
int creat(const char *path, mode_t mode);
```

non-negative number 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
fs
myphoto.jpg
Bus
Processor

File Descriptor (fs)

Number does not Matter, Connection Matters

The Only Way In Or Out Is Through Phone Lines
The number does not matter, the connection is important!
Imagine a dynamic phone#, dynamic postal code, dynamic ip (DHCP)

File Descriptor (fs) != File Identifier

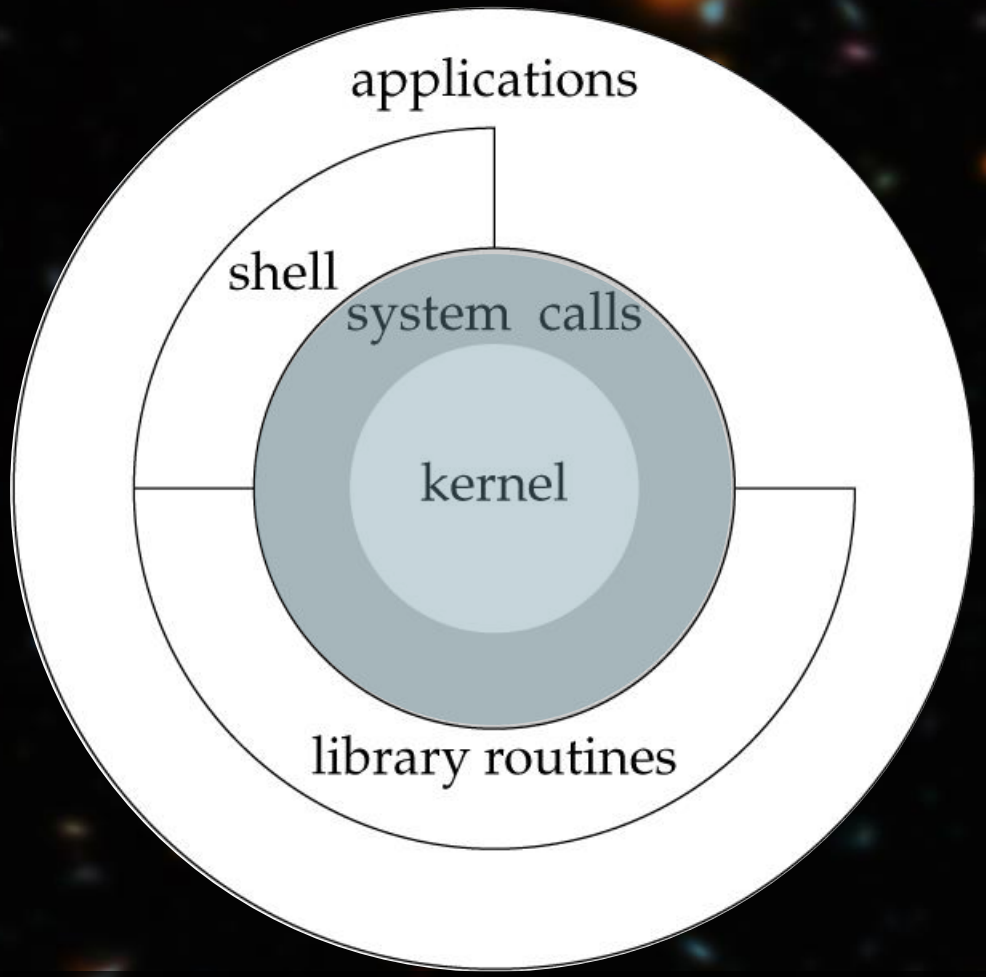
Because kernel reuse them for other files and devices, when available!

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

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);
```

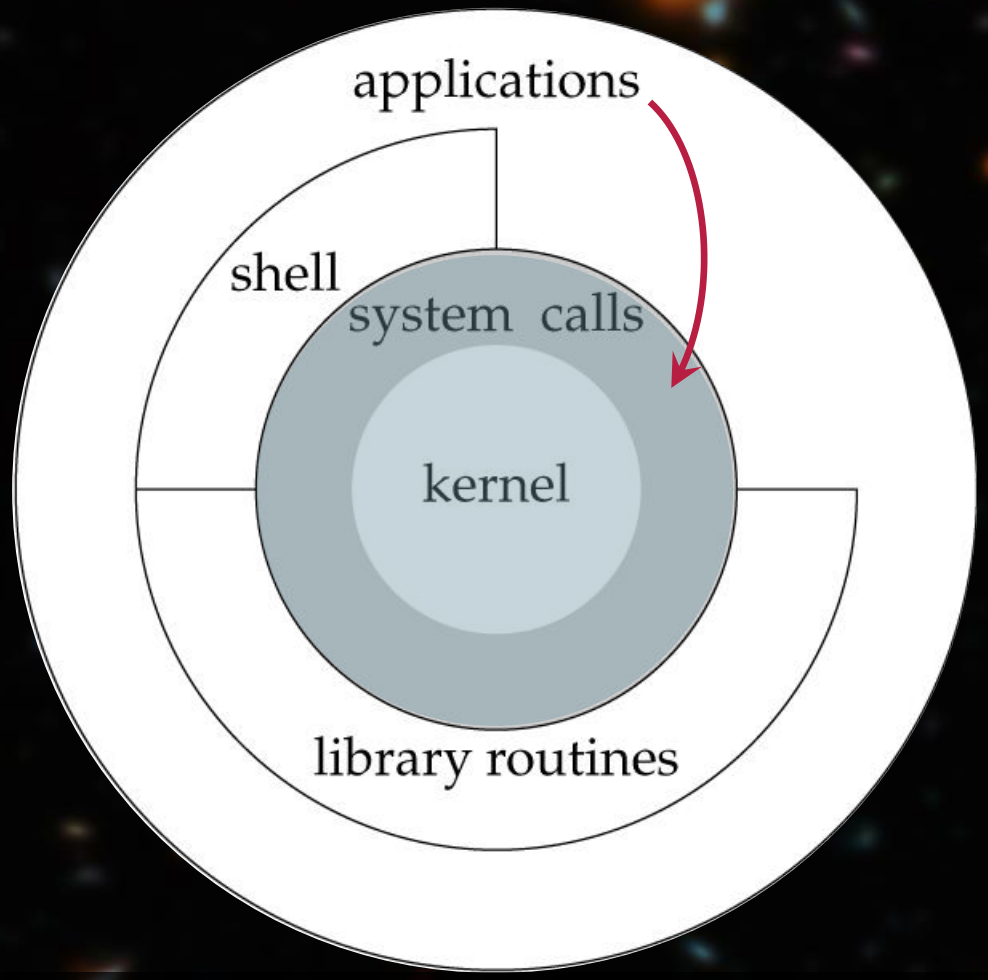
number of bytes written if OK, –1 on error

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

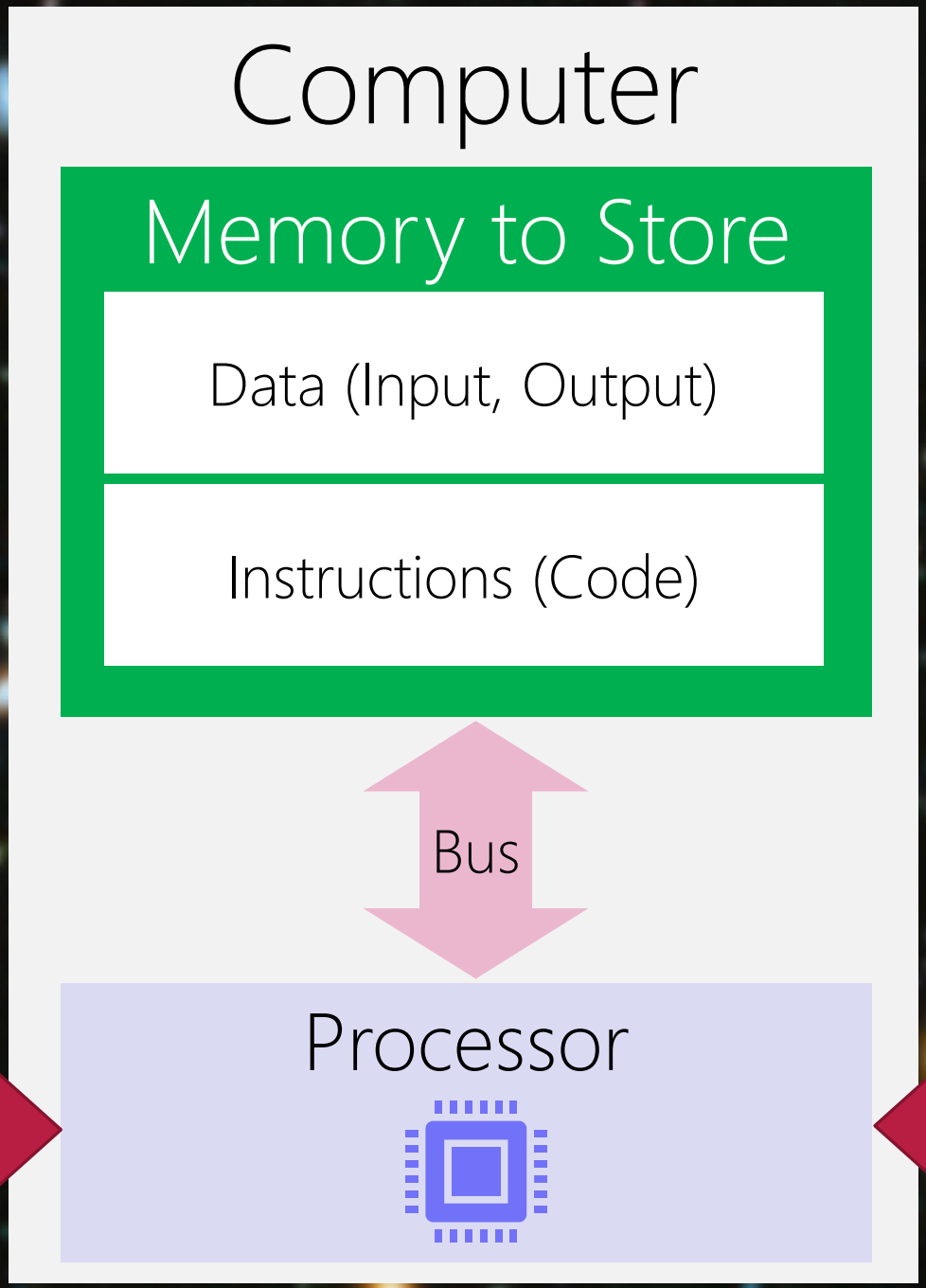
write

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

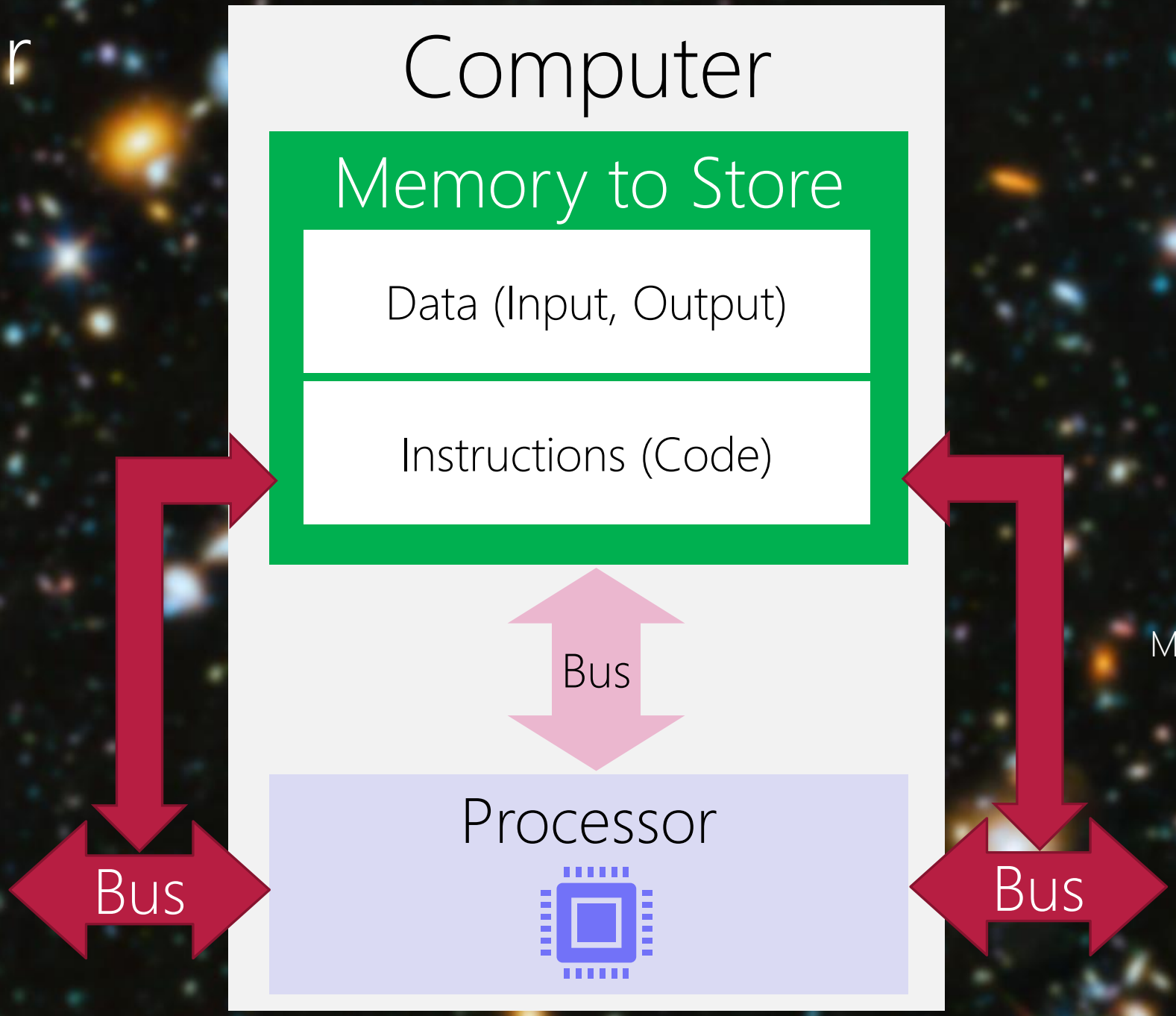
Computer System



Input/Output Devices

Permanent Storage

Computer System



Memory Mapped I/O

Memory Mapped I/O

Input/Output Devices

Permanent Storage

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!

```
#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
```

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!
~
~
```