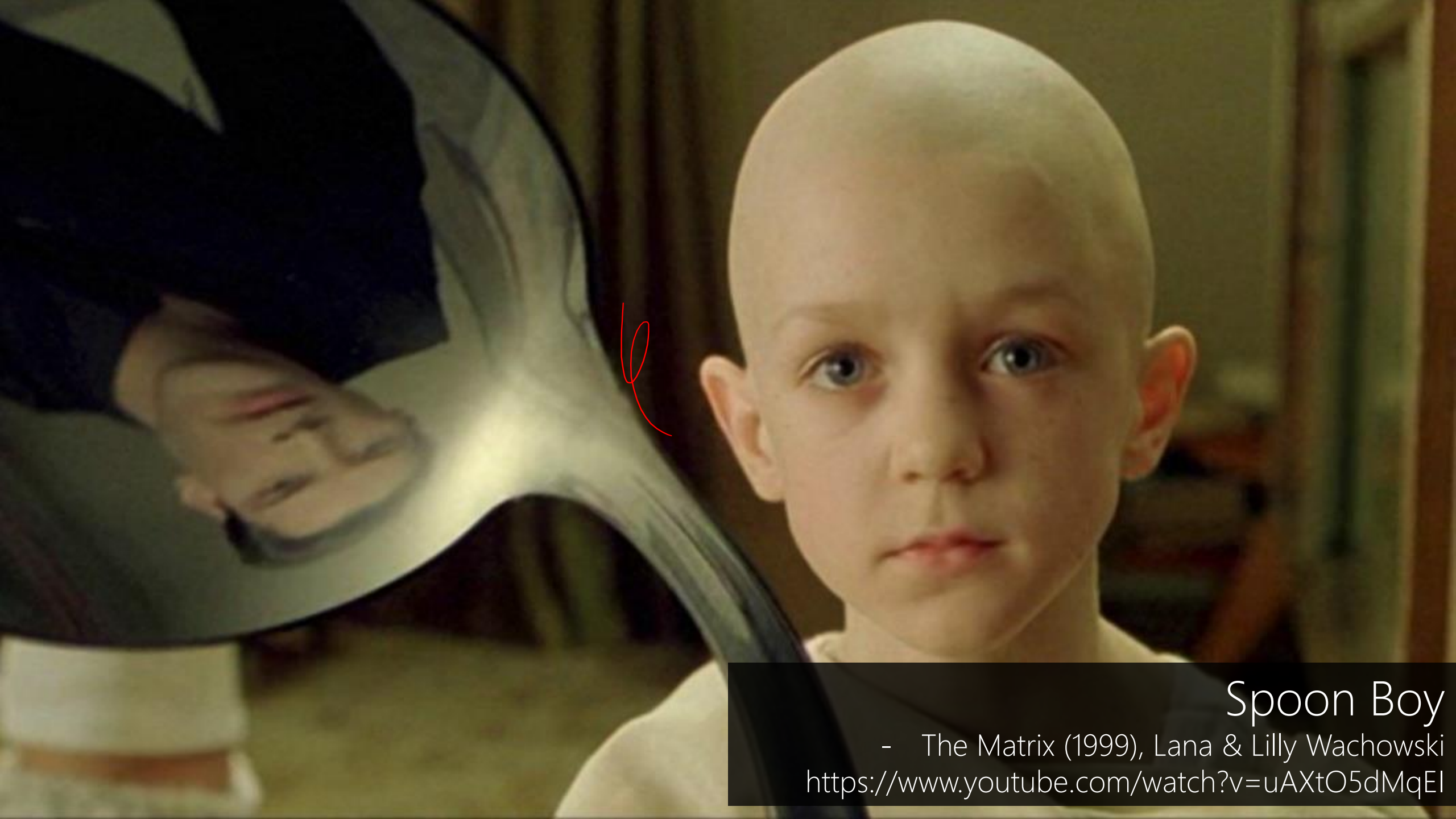
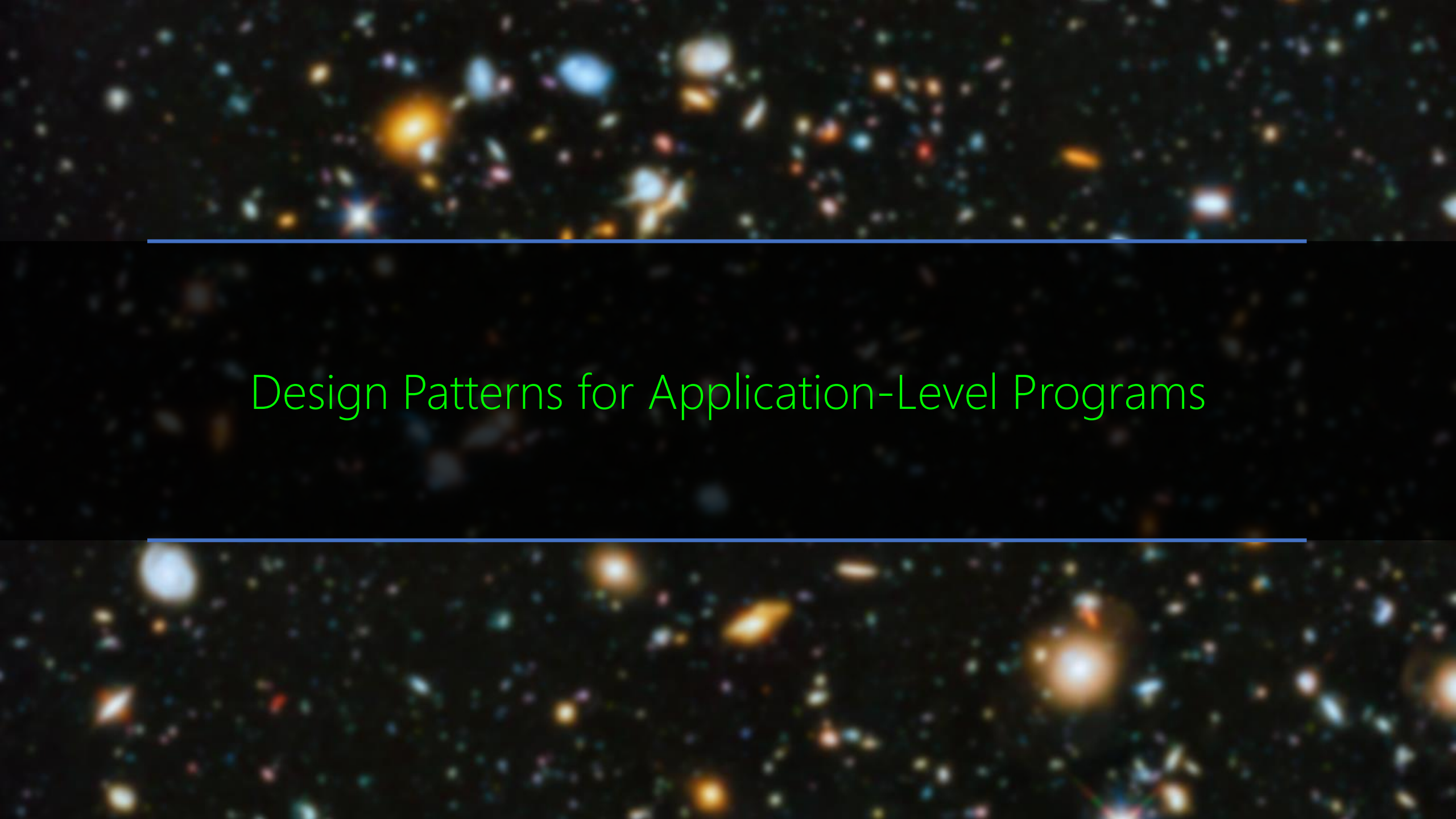


Spoon Boy

- The Matrix (1999), Lana & Lilly Wachowski

https://www.youtube.com/watch?v=uAXtO5dMqEI



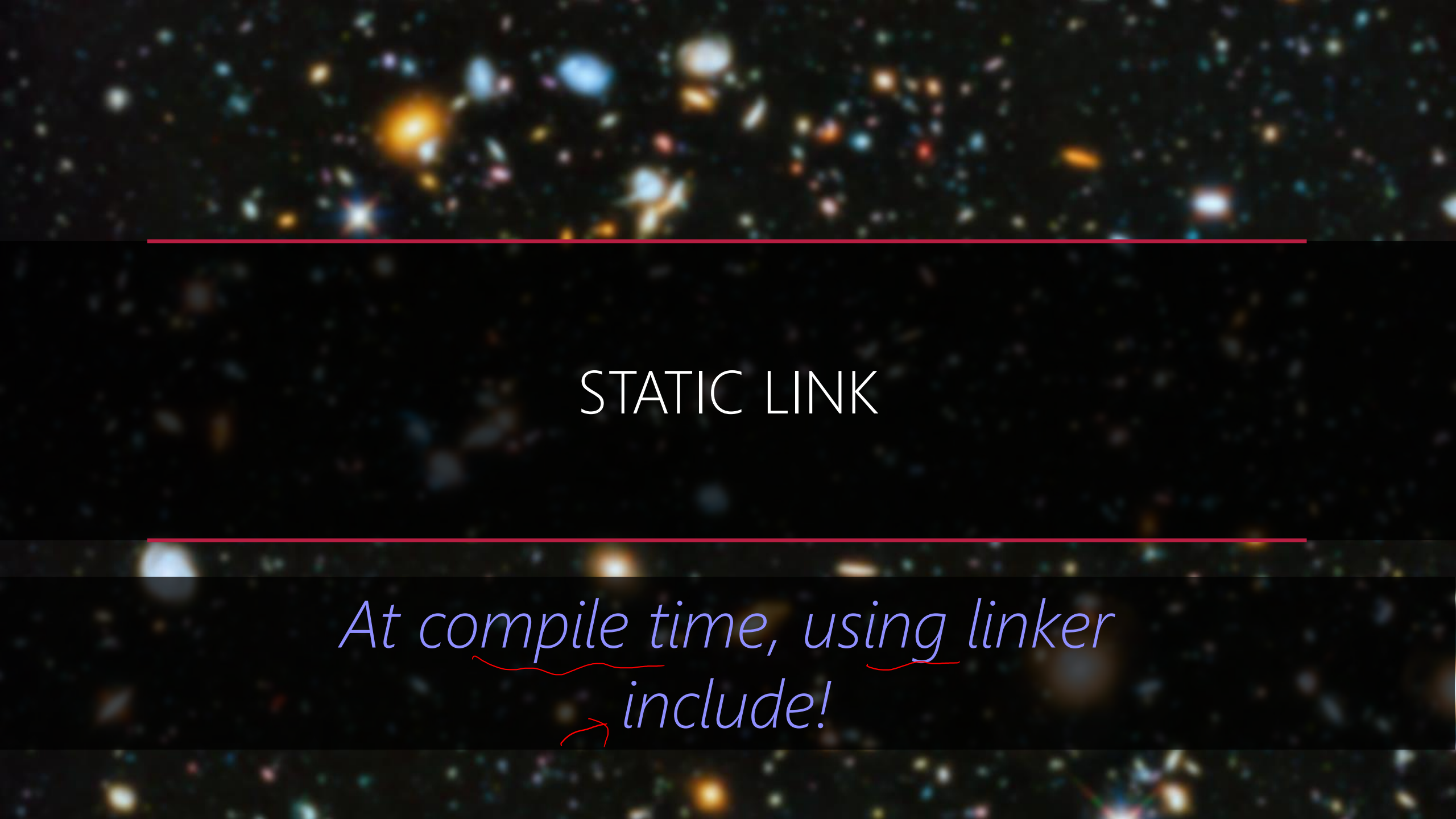
Design Patterns for Application-Level Programs

Shared Libraries

Is it possible to share standard library routines for all processes, inside or outside a computer?

add.c

https://cs-fundamentals.com/c-programming/static-and-dynamic-linking-in-c


STATIC LINK
At compile time, using linker include!

C

Compiler

```
#include <stdio.h>
void main(){
    printf("hello world!");
}
```

Assembly

Assembler

```
<printf@plt>:
jmpq        *0x3002(%rip)
pushq       $0x0
jmpq        401000 <.plt>
<main>:
push        %rbp
…
```

OP Code

stdio.h

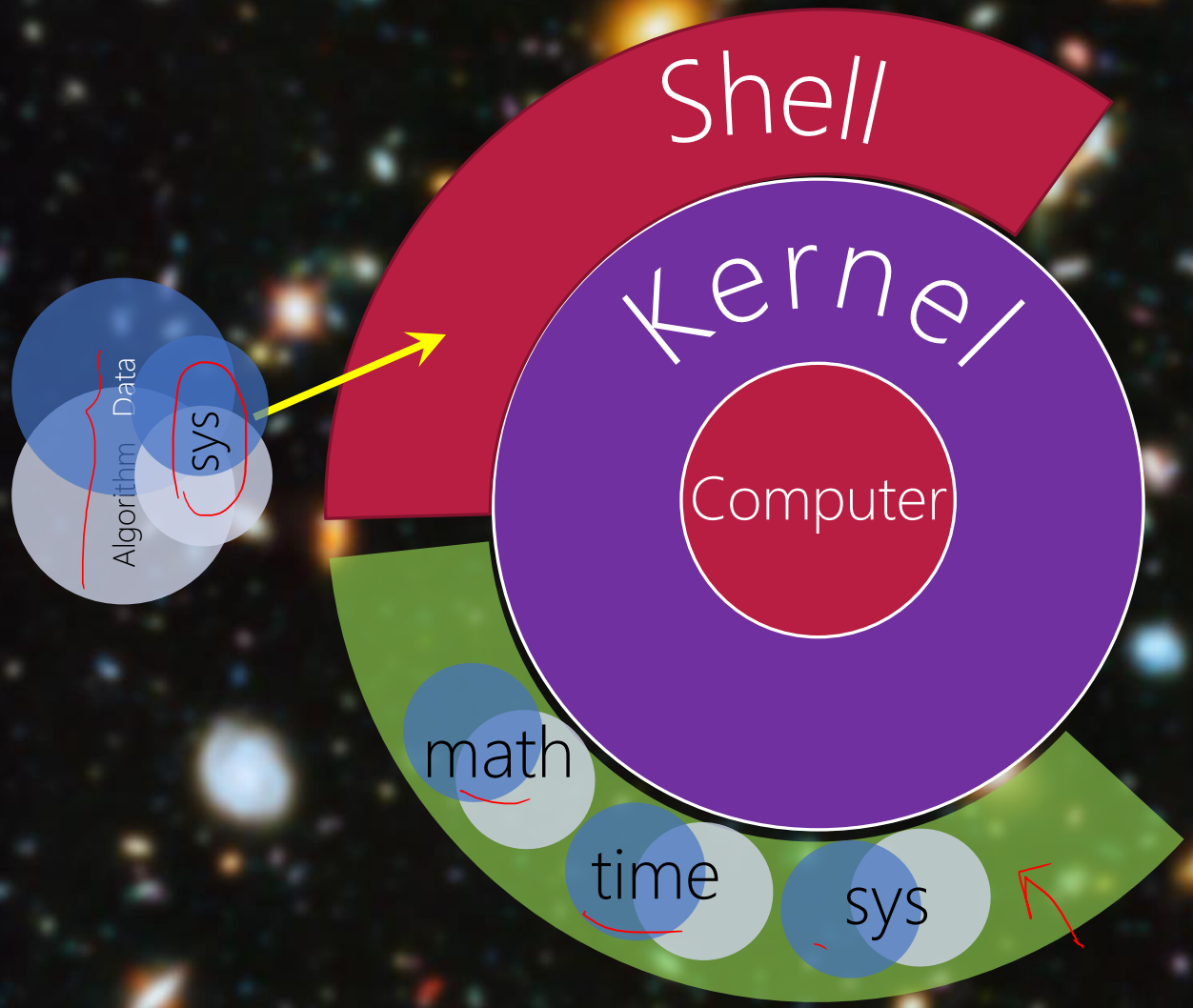
```
0001 0000 0000 0010 0000 0000 0000 1100
00f1 0000 0003 0000 0000 0000 0000 0000
0000 0000 0000 0000 3309 0110 0111 0000
0046 0000 0220 0000 0000 0000 0000 2200
0111 0000 0000 0000 e432 0000 0000 0000
```

OP Code

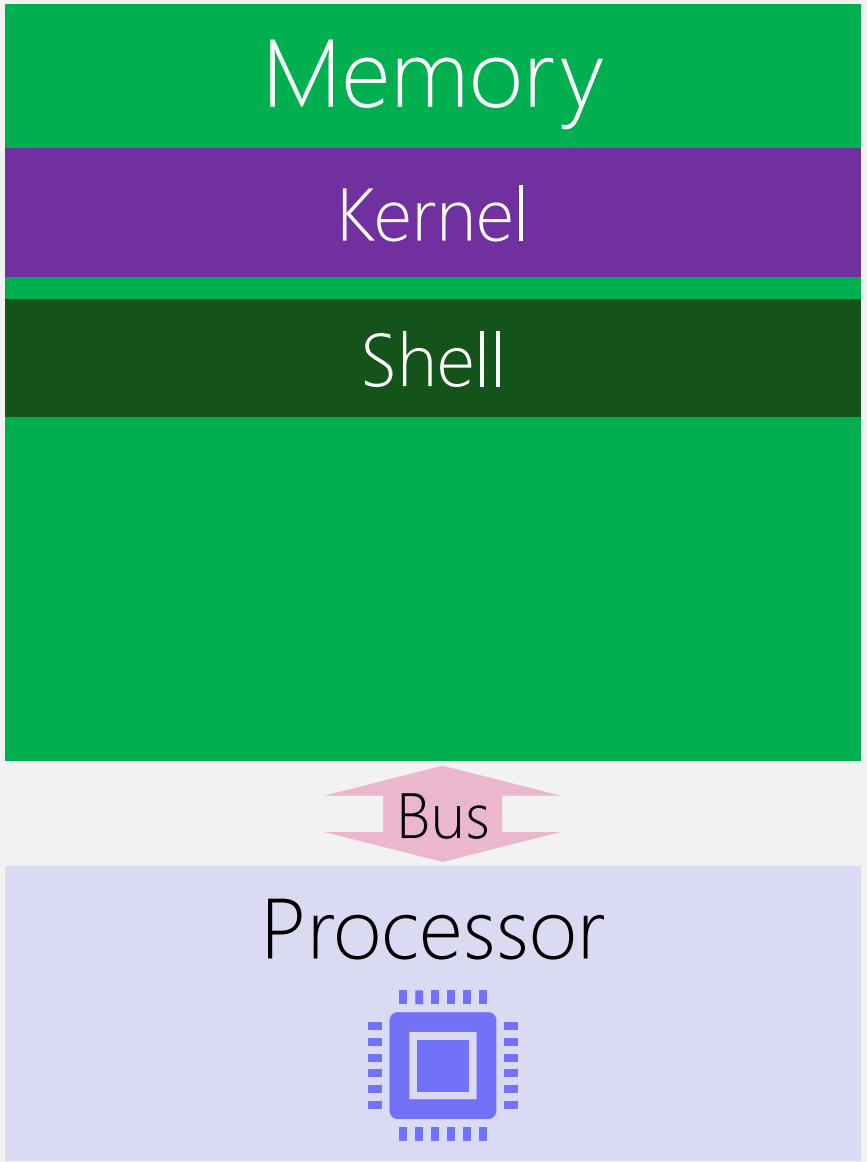
```
0001 0000 0000 0000 0000 0000 0000 0000
0011 0000 0003 0000 0000 0000 0000 0000
0000 0000 0000 0000 3309 0000 0000 0000
0096 0000 0000 0000 0000 0000 0000 0000
0001 0000 0000 0000 0000 0000 0000 0000
```

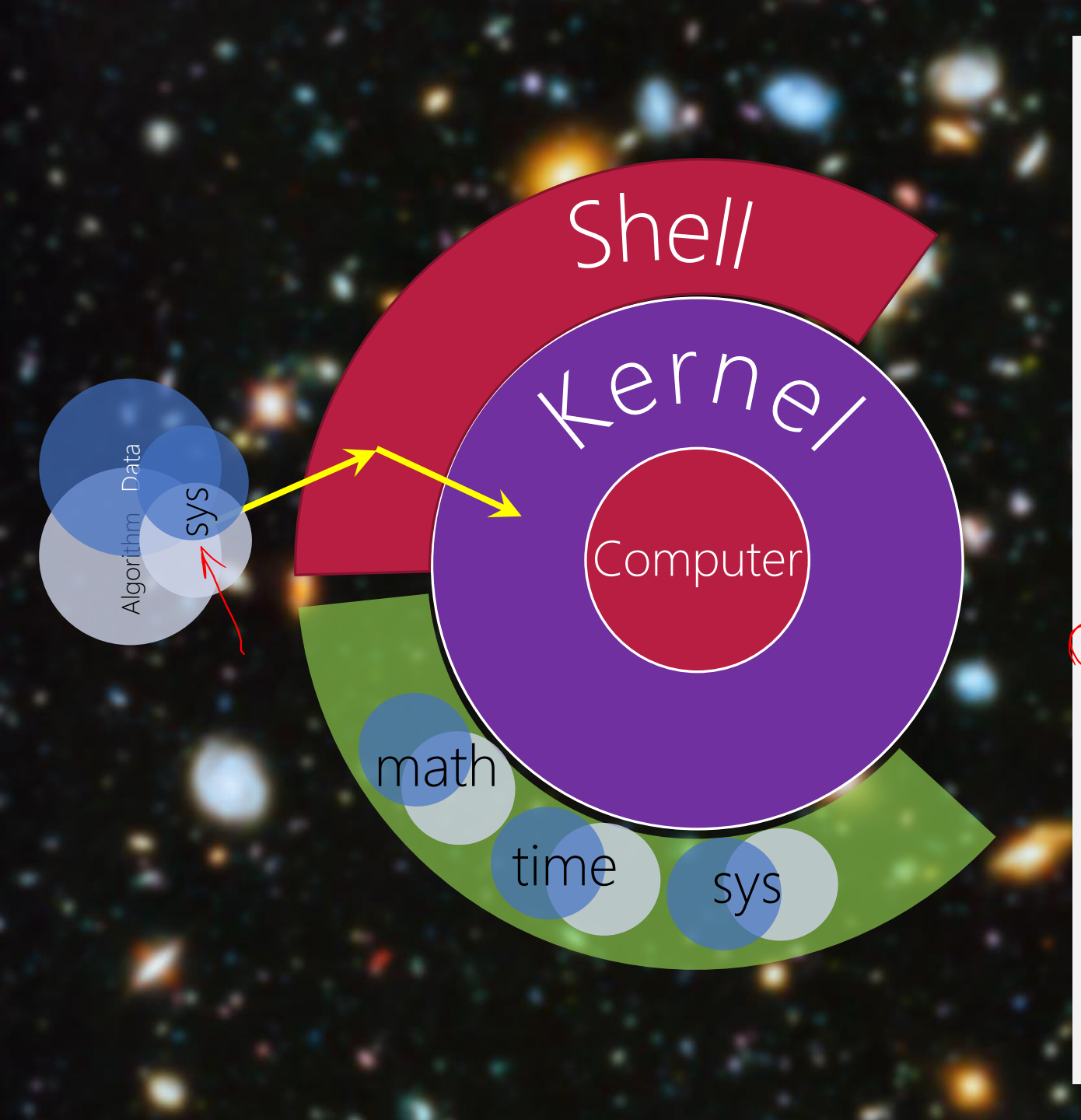
Linker

```
0001 0000 0000 0010 0000 0000 0000 1100
00f1 0000 0003 0000 0000 0000 0000 0000
0000 0000 0000 0000 3309 0110 0111 0000
0046 0000 0220 0000 0000 0000 0000 2200
0111 0000 0000 0000 e432 0000 0000 0000
0001 0000 0000 0000 0000 0000 0000 0000
0011 0000 0003 0000 0000 0000 0000 0000
0000 0000 0000 0000 3309 0000 0000 0000
0096 0000 0000 0000 0000 0000 0000 0000
0001 0000 0000 0000 0000 0000 0000 0000
```


Shell
Kernel
Computer
Algorithm
Data
sys
math
time
sys

Computer


Memory
Kernel
Shell
Bus
Processor



Computer

Memory

Kernel

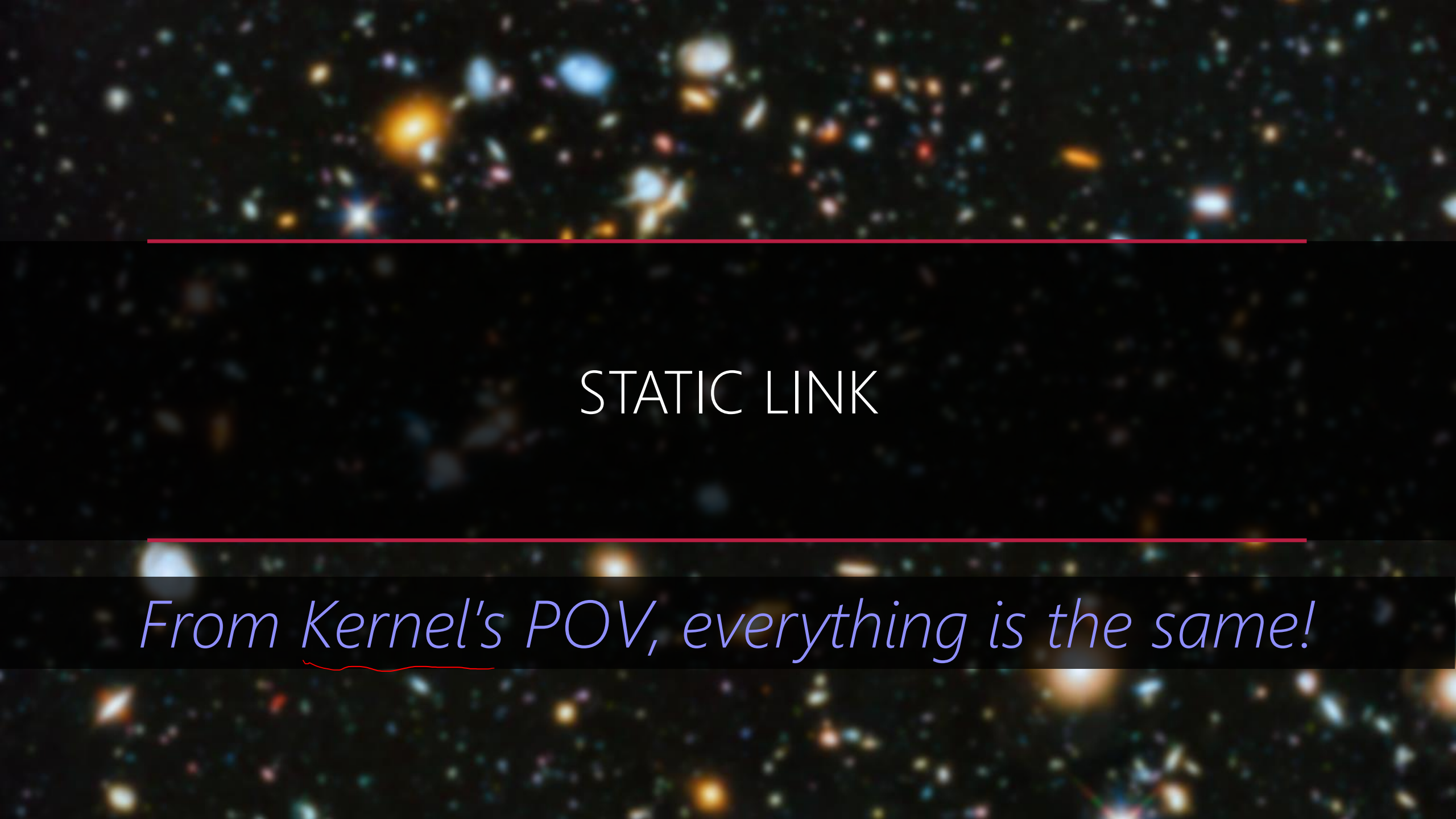
Shell

Process1: Program + Data

sys

Bus

Processor


STATIC LINK
From Kernel's POV, everything is the same!

1) Building the Static Library

```c
int add(int x, int y){
        return x + y;
}
```

```c
int sub(int x, int y){
        return x - y;
}
```

multiply

sqrt

Roots of a quadratic equation!

```
cc add.c -c
```



```
0001 0000 0000 0000 0000 0000 0000 0000
0011 0000 0003 0000 0000 0000 0000 0000
0000 0000 0000 0000 3309 0000 0000 0000
0096 0000 0000 0000 0000 0000 0000 0000
```

add.o

```
cc sub.c -c
```



```
0111 0110 0000 0000 0000 0000 0000 0000
0011 0000 0003 0000 0000 0000 0000 0000
0000 0000 0000 0000 3309 0000 0000 0000
0096 0000 0000 0000 0000 0000 0000 0000
```

sub.o

```
cc multi.c -c
```



```
0001 0000 0000 0000 0000 0000 0000 0000
0011 0000 0002 0000 0000 0000 0000 0000
0000 0000 1111 0000 3309 0000 0000 0000
0086 0000 0000 0000 0000 0000 0000 0000
```

multi.o

```
cc sqrt.c -c
```



```
0001 0000 0000 0000 0000 0000 0000 0000
0011 0000 0003 0000 0000 0000 0000 0000
0000 0000 0000 0000 2222 0000 0000 0000
0096 0000 0000 0000 0000 0000 0000 0000
```

sqrt.o

```
0001 0000 0000 0000 0000 0000 0000 0000
0011 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 5641 0000 0000 0000
0090 0000 0000 0000 0000 0000 0000 0000
```

```c
int add(int x, int y){
        return x + y;
}
```

cc add.c -c

```
0001 0000 0000 0000 0000 0000 0000 0000
0011 0000 0003 0000 0000 0000 0000 0000
0000 0000 0000 0000 3309 0000 0000 0000
0096 0000 0000 0000 0000 0000 0000 0000
```

add.o

```c
int sub(int x, int y){
        return x - y;
}
```

cc sub.c -c

```
0111 0110 0000 0000 0000 0000 0000 0000
0011 0000 0003 0000 0000 0000 0000 0000
0000 0000 0000 0000 3309 0000 0000 0000
0096 0000 0000 0000 0000 0000 0000 0000
```

sub.o

multiply

cc multi.c -c

```
0001 0000 0000 0000 0000 0000 0000 0000
0011 0000 0002 0000 0000 0000 0000 0000
0000 0000 1111 0000 3309 0000 0000 0000
0086 0000 0000 0000 0000 0000 0000 0000
```

multi.o

sqrt

cc sqrt.c -c

```
0001 0000 0000 0000 0000 0000 0000 0000
0011 0000 0003 0000 0000 0000 0000 0000
0000 0000 0000 0000 2222 0000 0000 0000
0096 0000 0000 0000 0000 0000 0000 0000
```

sqrt.o

Roots of a quadratic equation!

```
0001 0000 0000 0000 0000 0000 0000 0000
0011 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 5641 0000 0000 0000
0090 0000 0000 0000 0000 0000 0000 0000
```

Archive (https://linux.die.net/man/1/ar)

Object files to archive

```
-$ ar rs mathlib.a add.o sub.o multi.o sqrt.o
```

Name of the archive file
Convention: *.a

2) Using the Static Library

```c
#include <stdio.h>

int add(int, int);
int sub(int, int);
int multi(int, int);
int sqrt(int);
...

void main(void){
        printf("2 + 3 = %d\n", add(2,3));
        printf("2 - 3 = %d\n", sub(2,3));
        ...

}
```

Functions prototypes

Function calls

```c
#include <stdio.h>

int add(int, int);
int sub(int, int);
int multi(int, int);
int sqrt(int);
...

void main(void){
        printf("2 + 3 = %d\n", add(2,3));
        printf("2 - 3 = %d\n", sub(2,3));
        ...

}
```

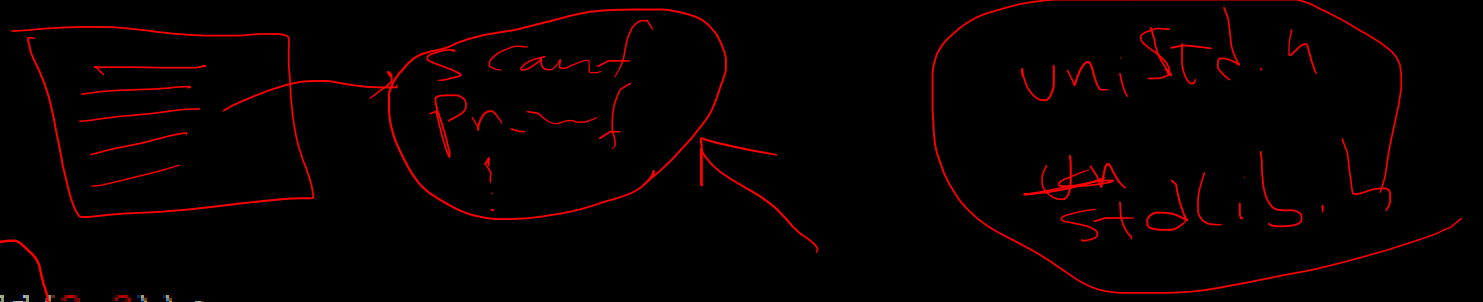
Functions prototypes into another file → Library Header File → mathlib.h

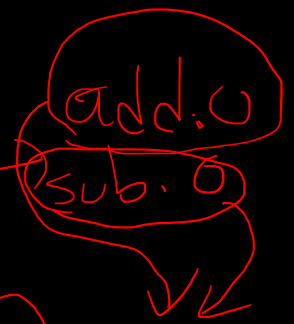
```
:~$ vi mathlib.h
int add(int, int);
int sub(int, int);
int multi(int, int);
int sqrt(int);
//...
~
~
```

```c
#include <stdio.h>

#include "mathlib.h"
void main(void){
        printf("2 + 3 = %d\n", add(2,3));
        printf("2 - 3 = %d\n", sub(2,3));
}
```

```
:~$ vi mathlib.h
int add(int, int);
int sub(int, int);
int multi(int, int);
int sqrt(int);
//...
~
~
```

```c
#include <stdio.h>

#include "mathlib.h"
void main(void){
        printf("2 + 3 = %d\n", add(2,3));
        printf("2 - 3 = %d\n", sub(2,3));
}
```



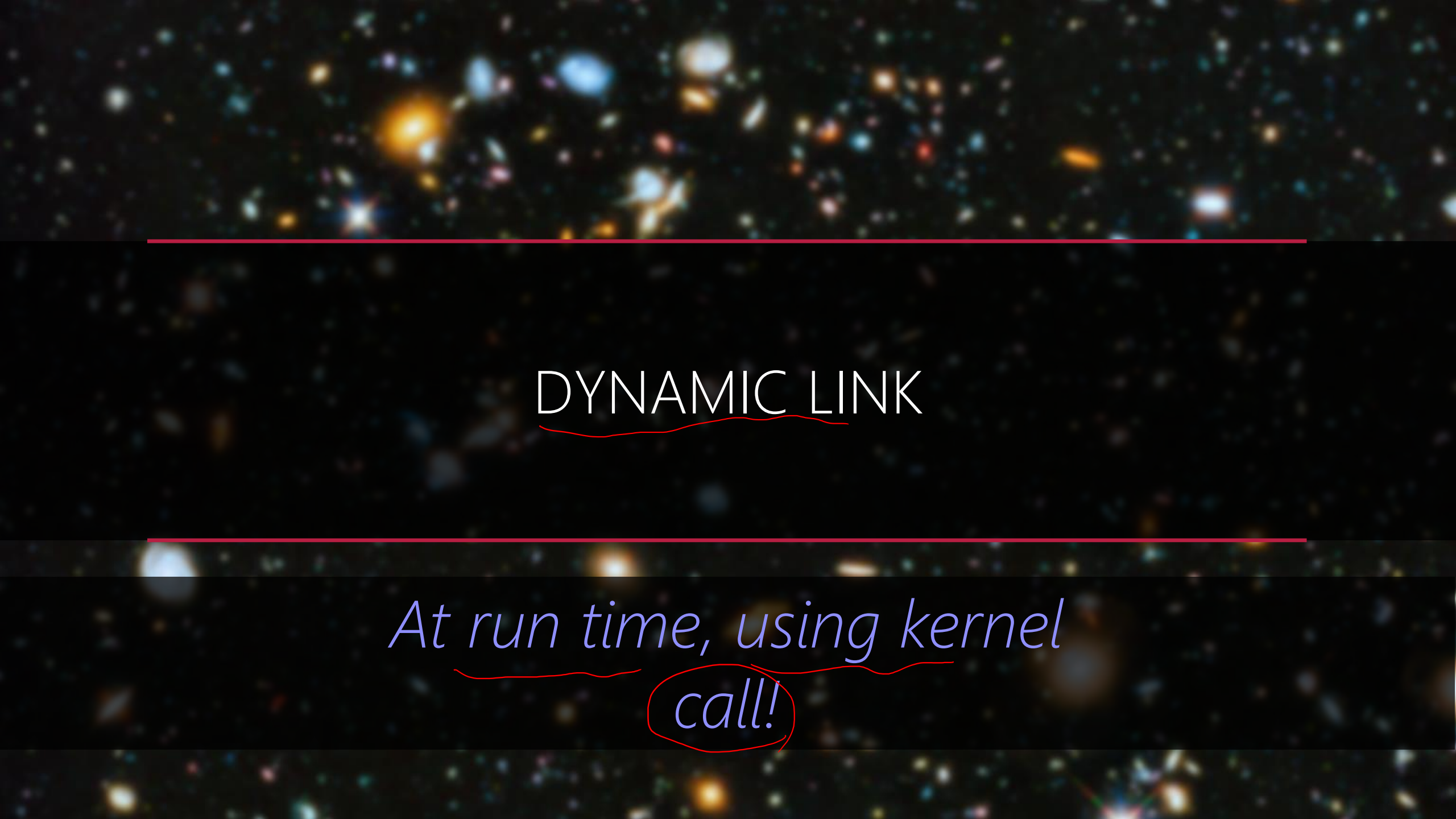



```
hfani@alpha:~$ cc main.c -c
hfani@alpha:~$ cc main.o mathlib.a -o main
hfani@alpha:~$ ./main
2 + 3 = 5
2 - 3 = -1
```

Includes main.o as well as object files of mathlib.a that are used

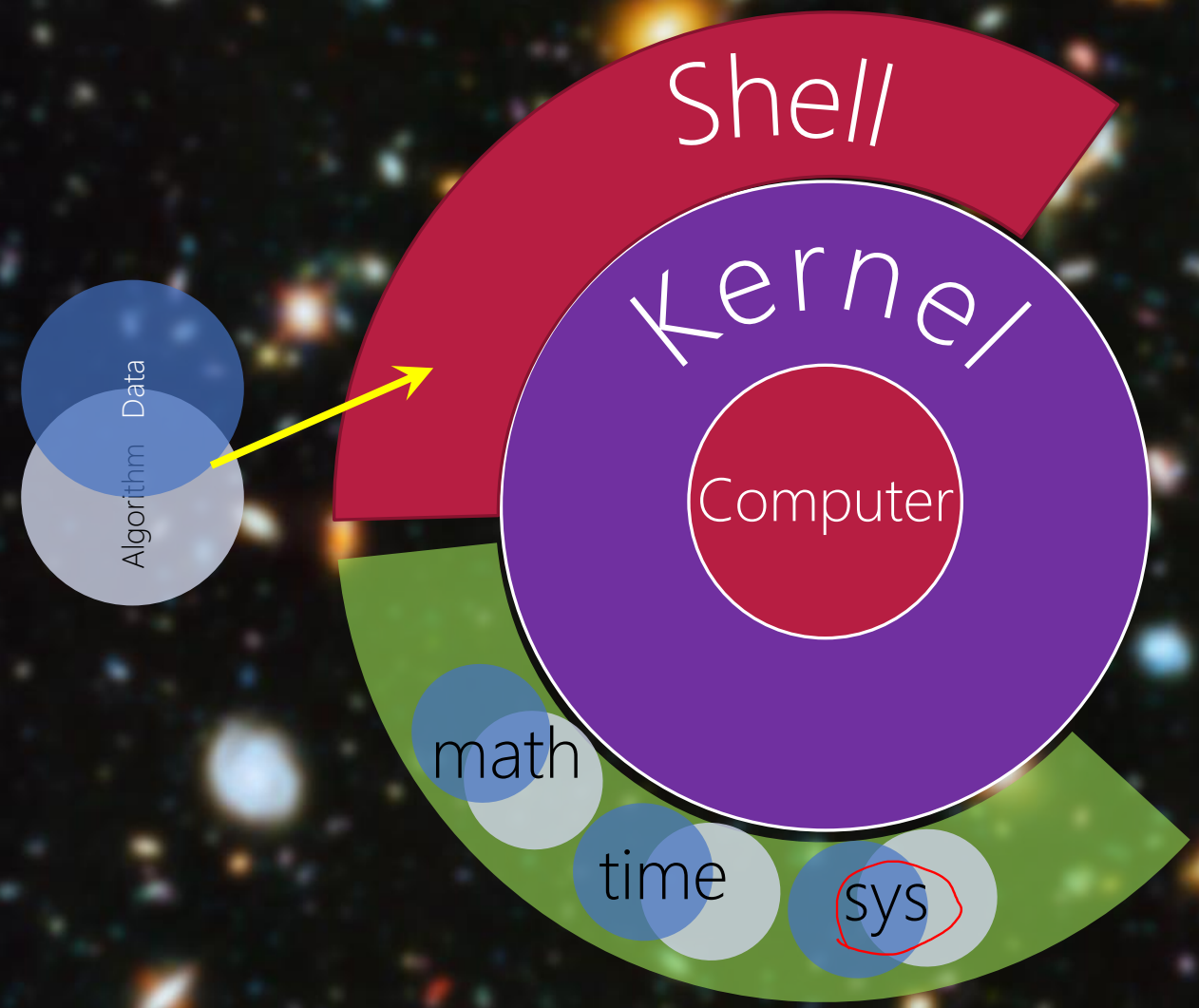
main.o
add.o
sub.o

```
hfani@alpha:~$ size ./main
   text    data     bss     dec     hex filename
   1647     584       8    2239     8bf ./main
```

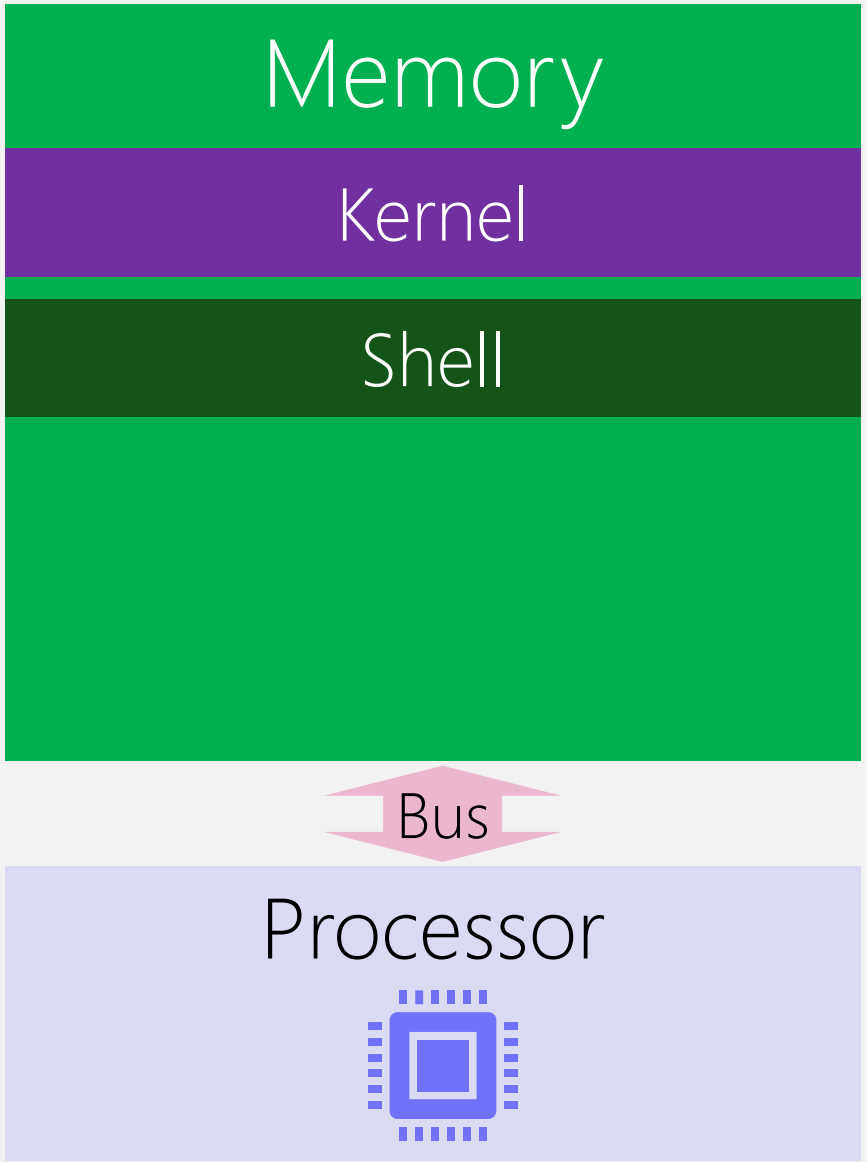


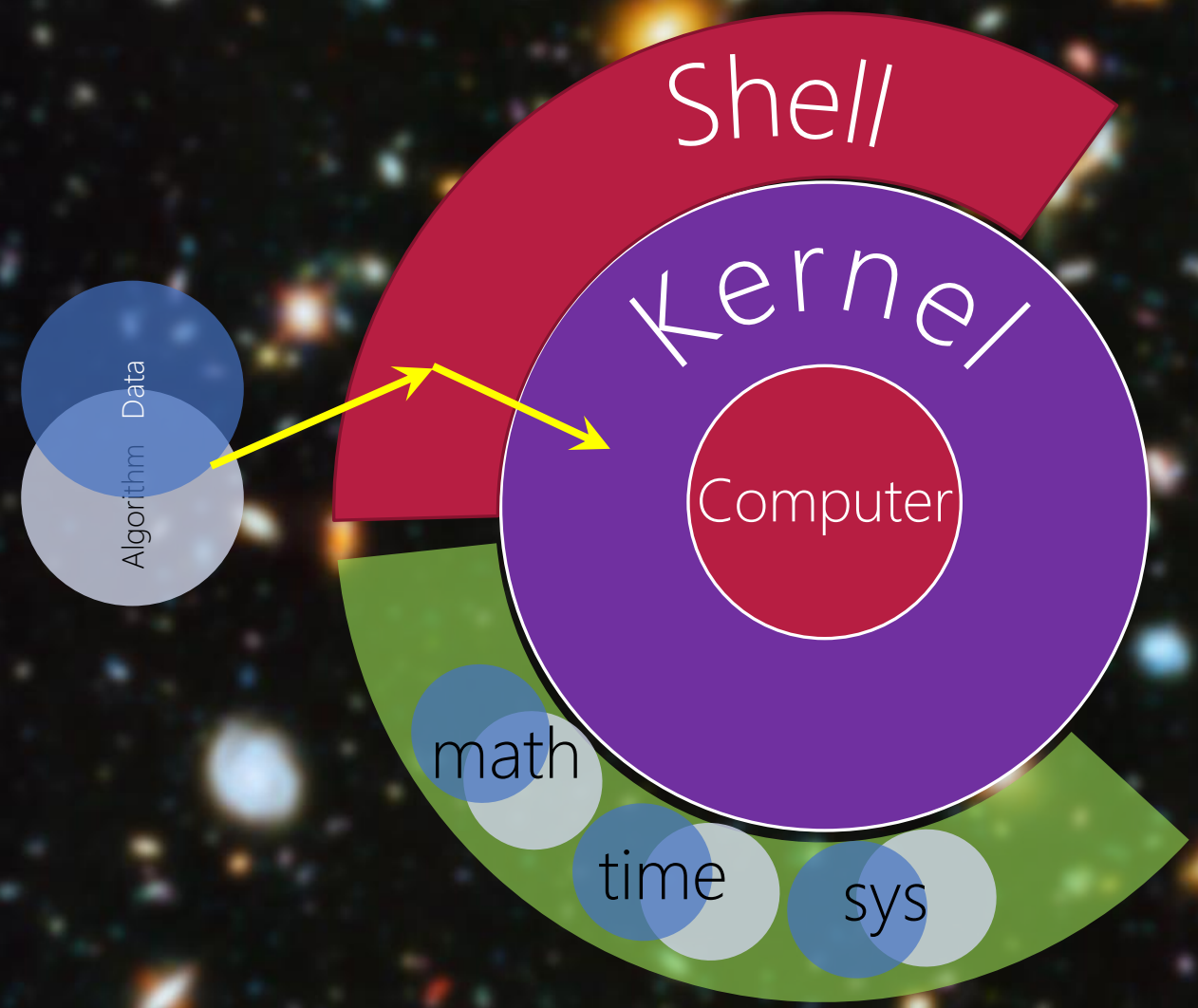
DYNAMIC LINK

At run time, using kernel call!

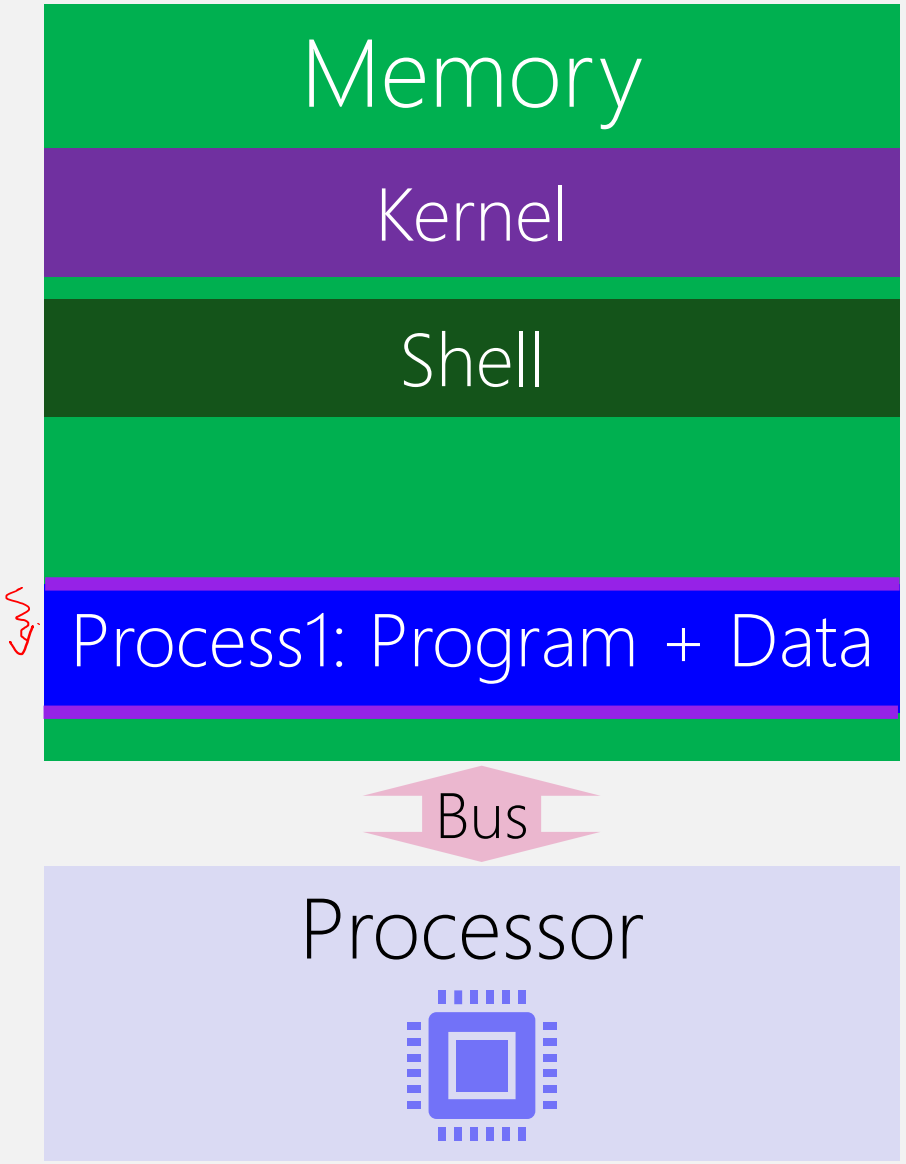

Shell
Kernel
Computer
Algorithm
Data
math
time
sys

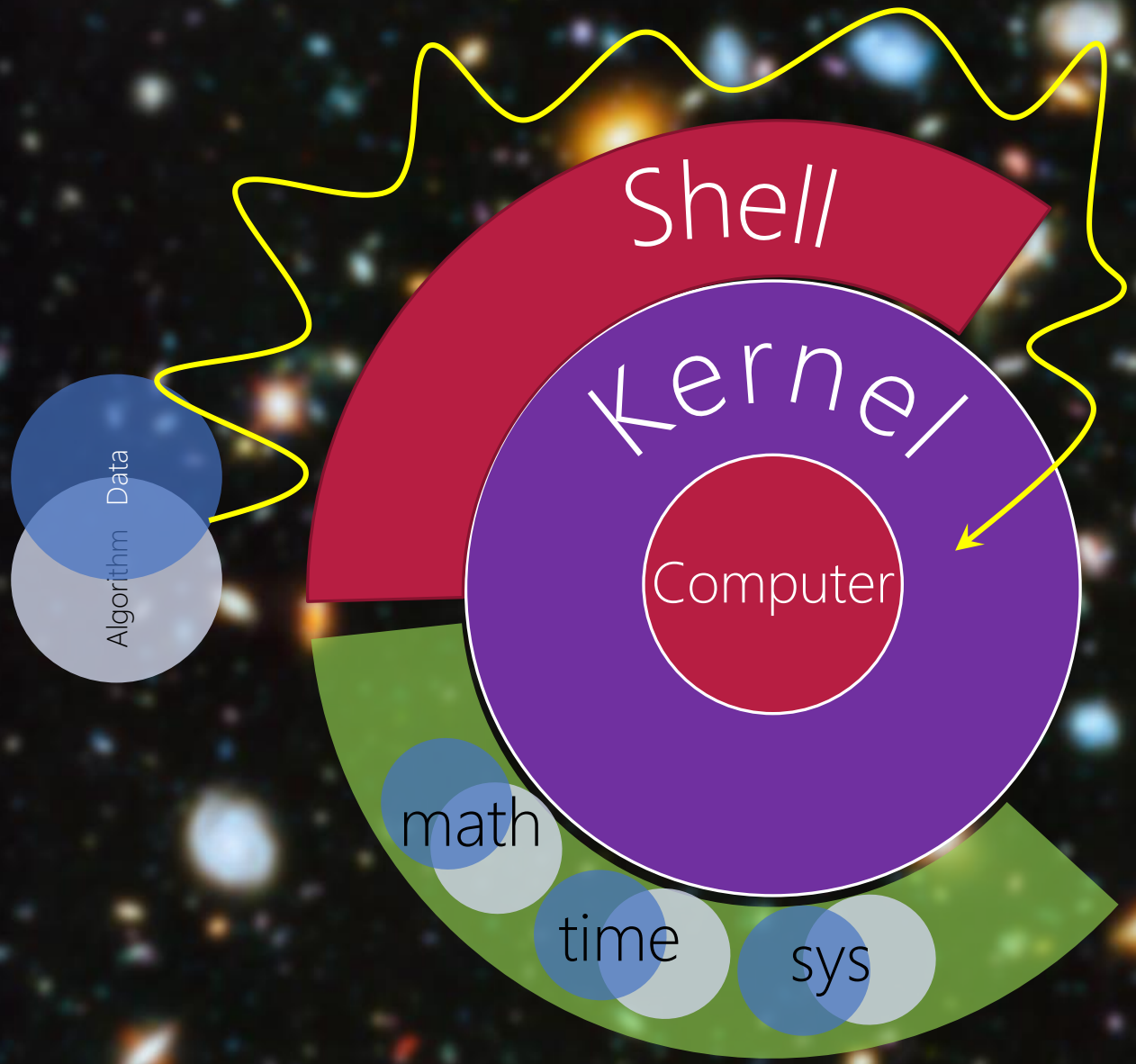
Computer


Memory
Kernel
Shell
Bus
Processor

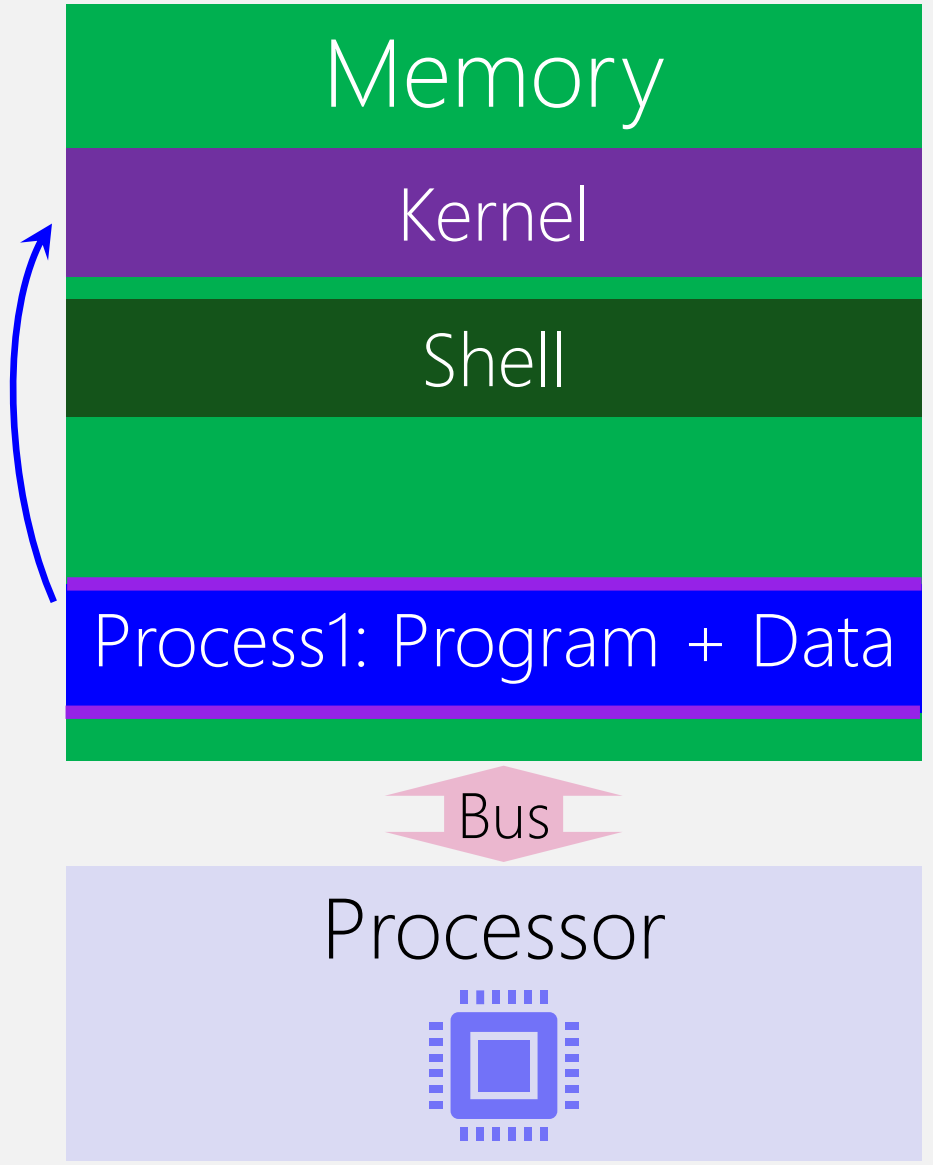

Shell
Kernel
Computer
Algorithm
Data
math
time
sys

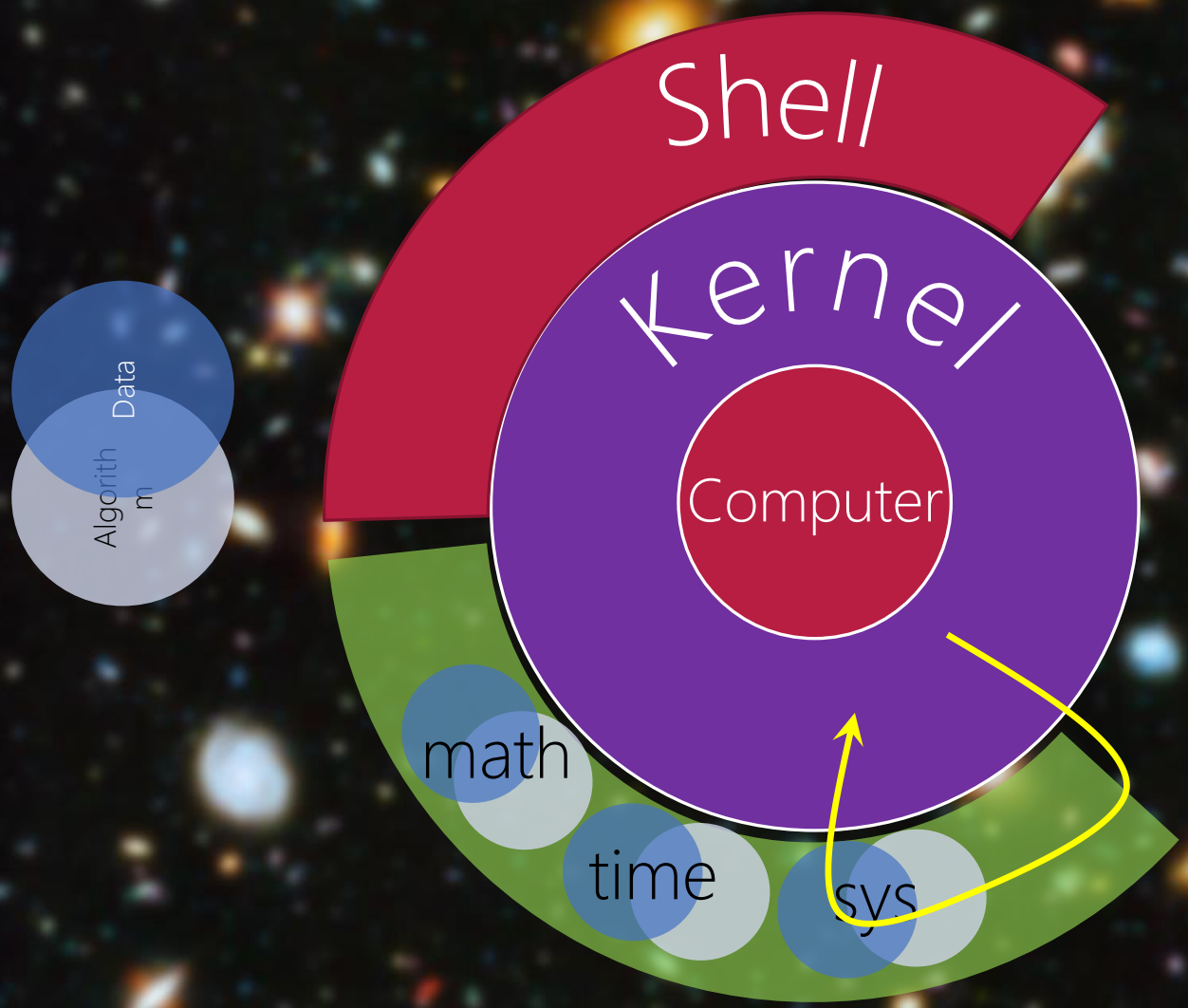
Computer


Memory
Kernel
Shell
Process1: Program + Data
Bus
Processor

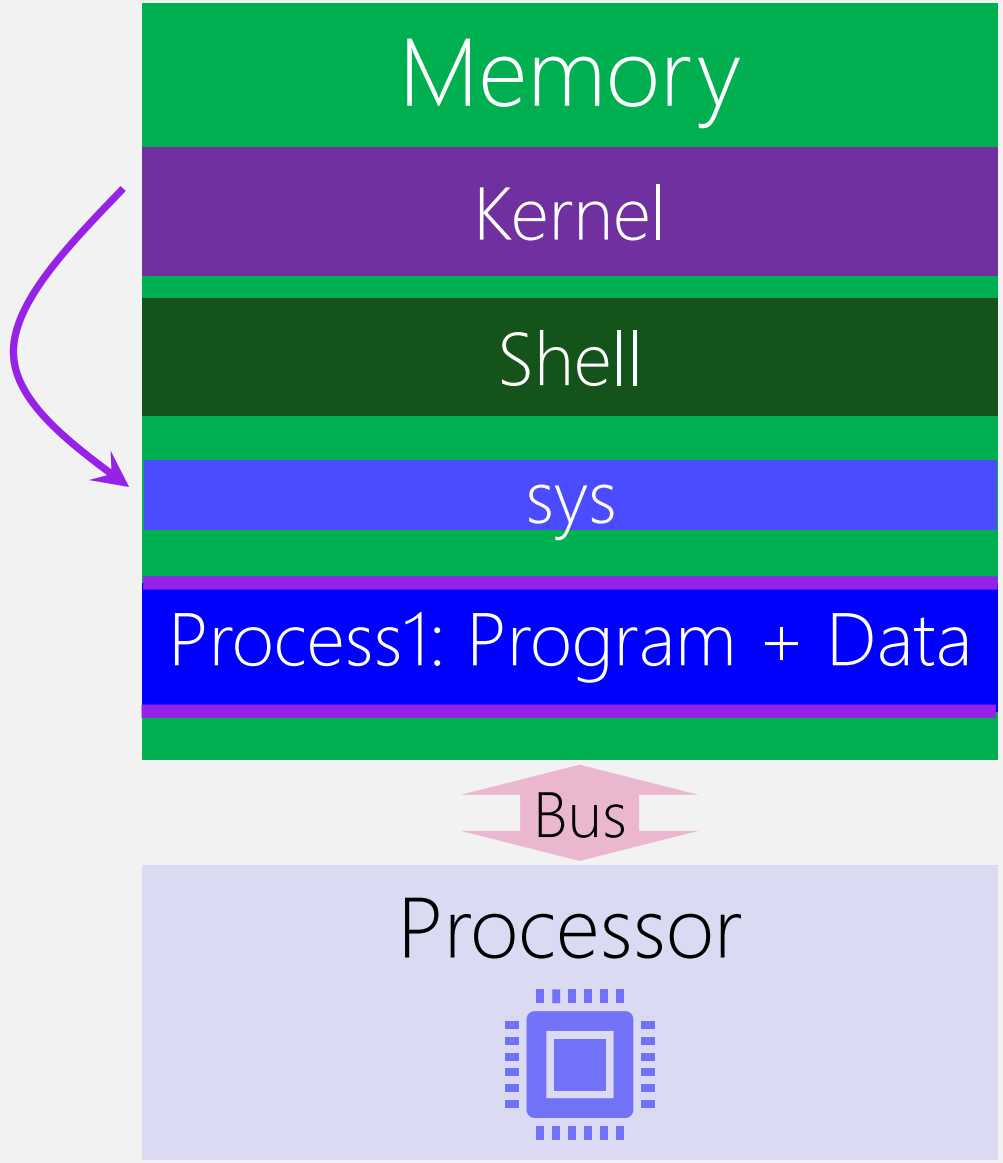

Shell
Kernel
Computer
Algorithm
Data
math
time
sys

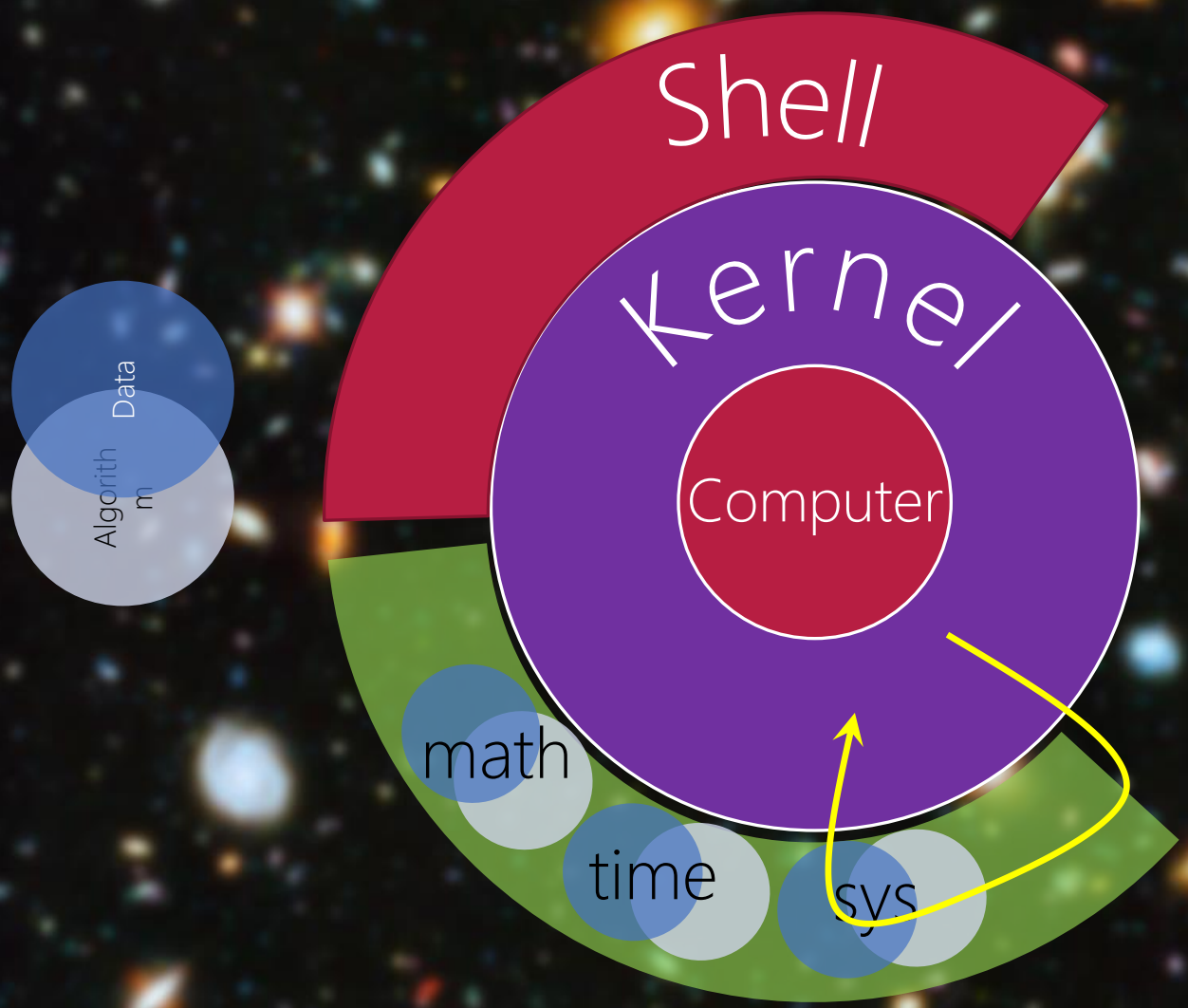
Computer


Memory
Kernel
Shell
Process1: Program + Data
Bus
Processor

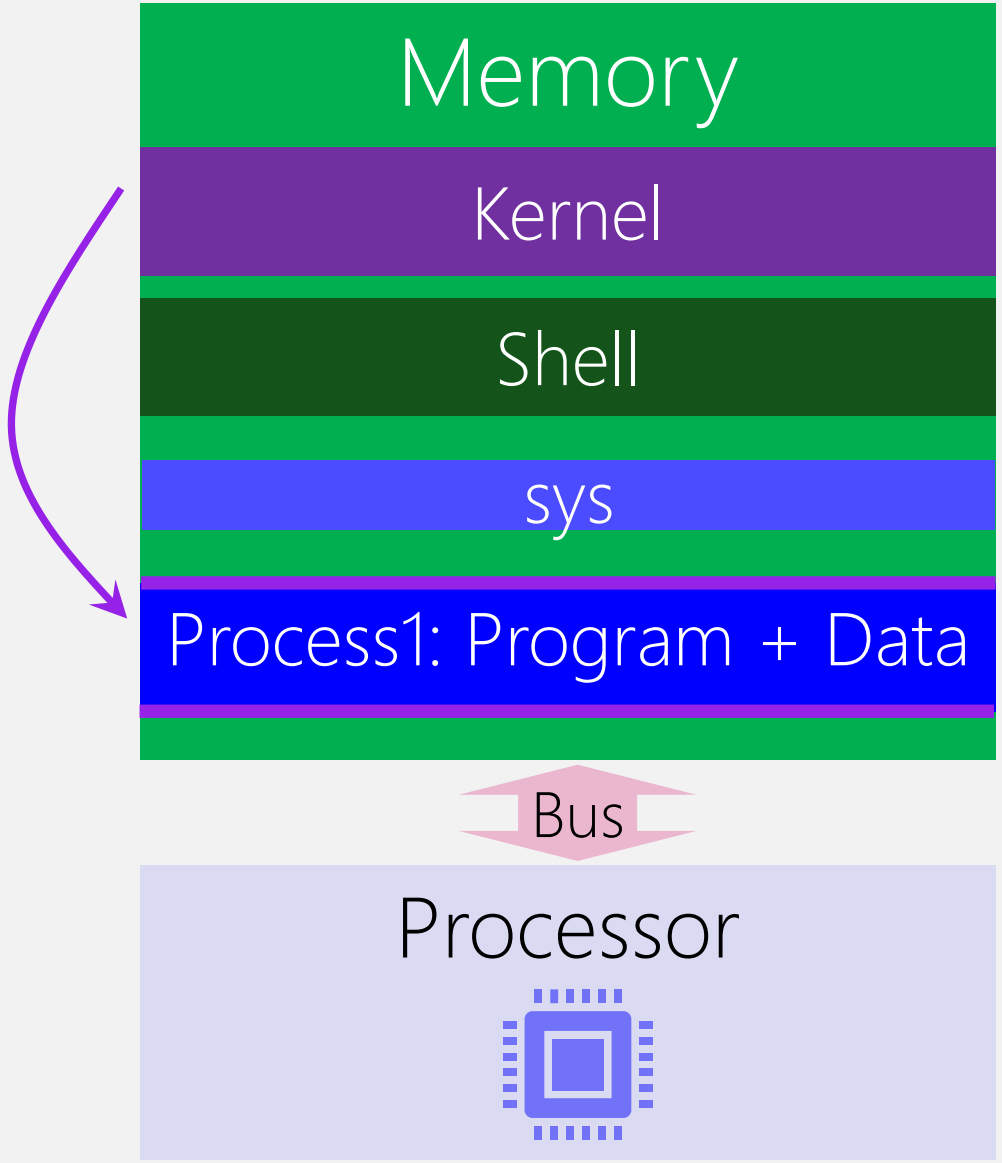

Shell
Kernel
Computer
Data
Algorithm
math
time
sys

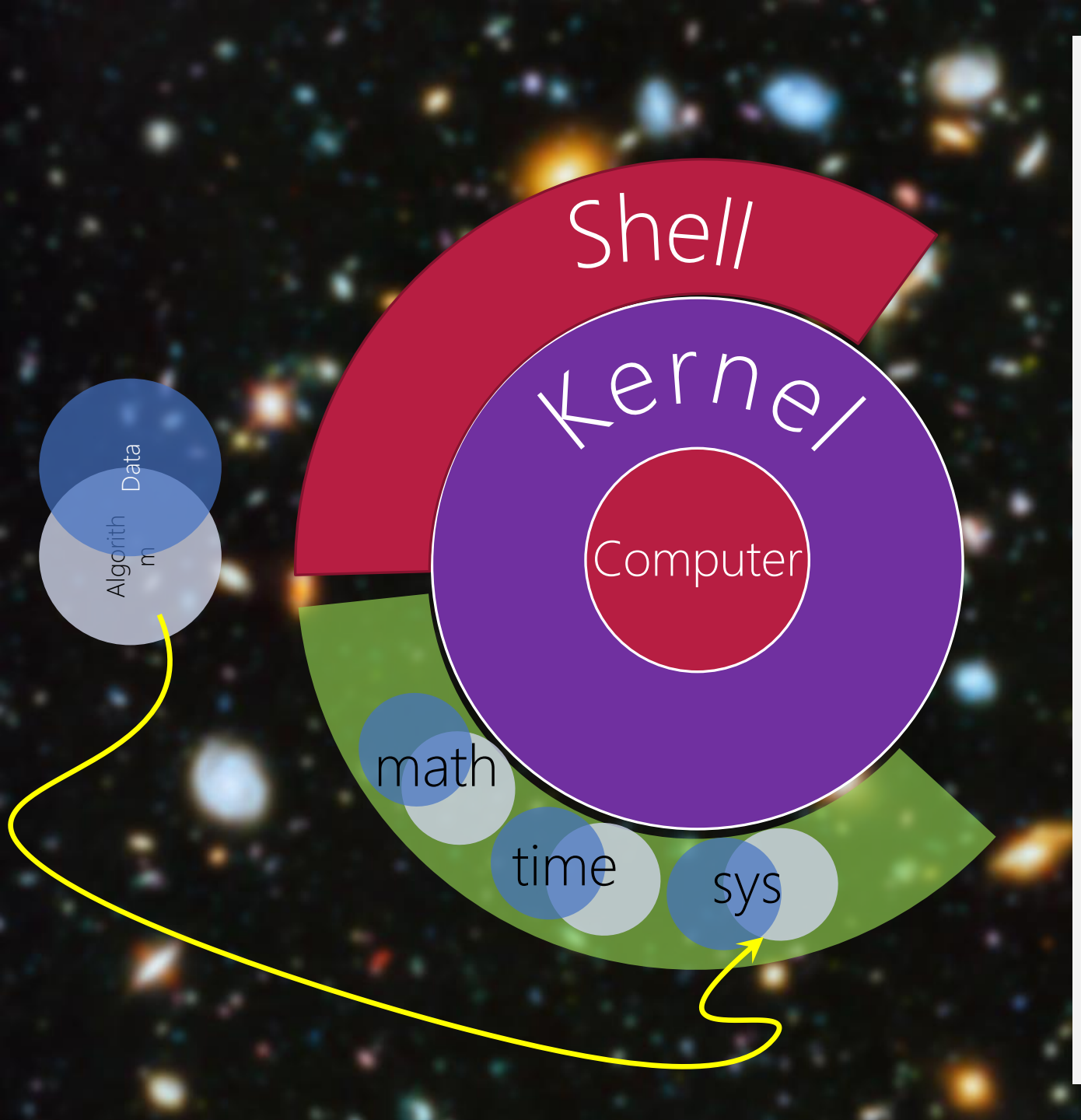
Computer


Memory
Kernel
Shell
sys
Process1: Program + Data
Bus
Processor

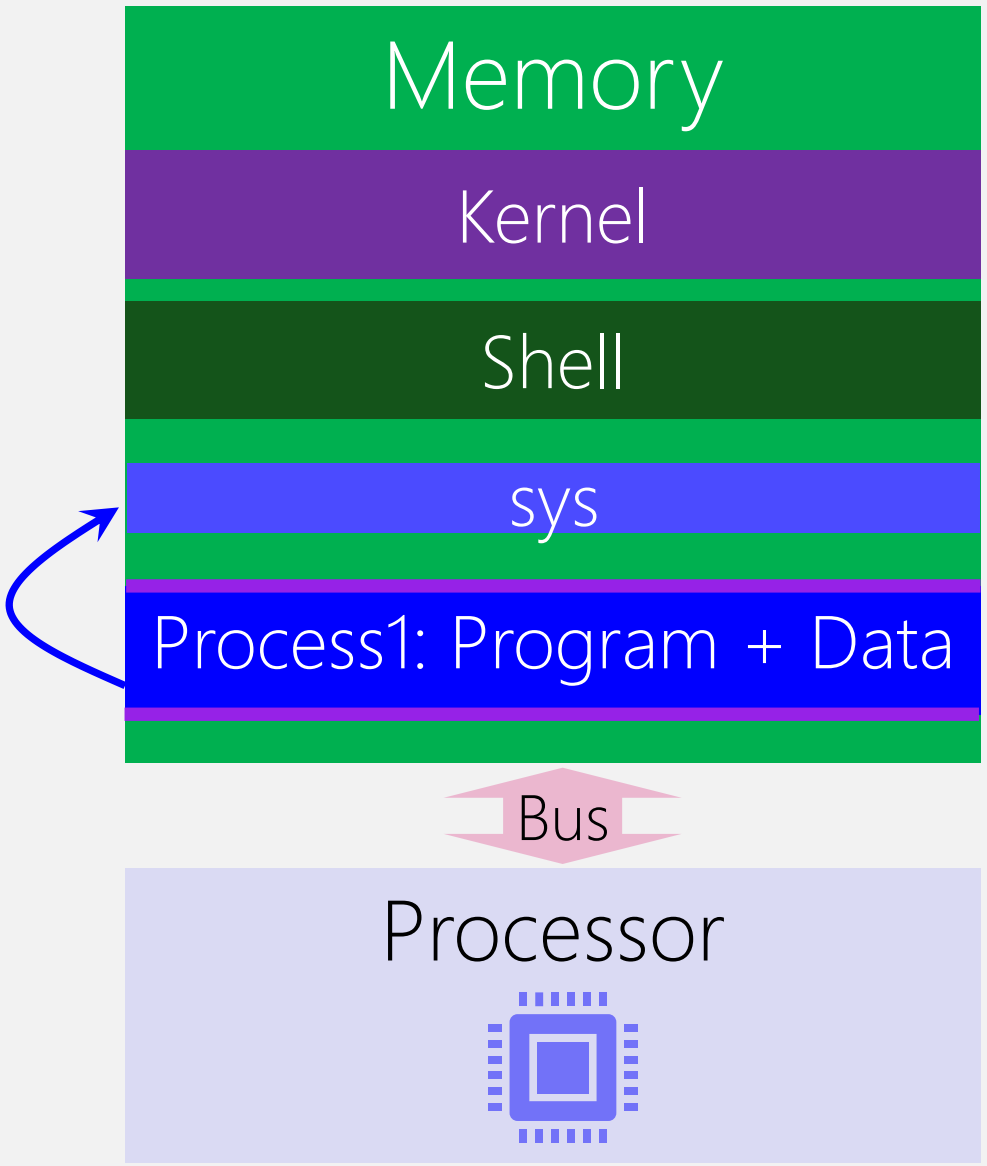

Shell
Kernel
Computer
math
time
sys
Data
Algorithm

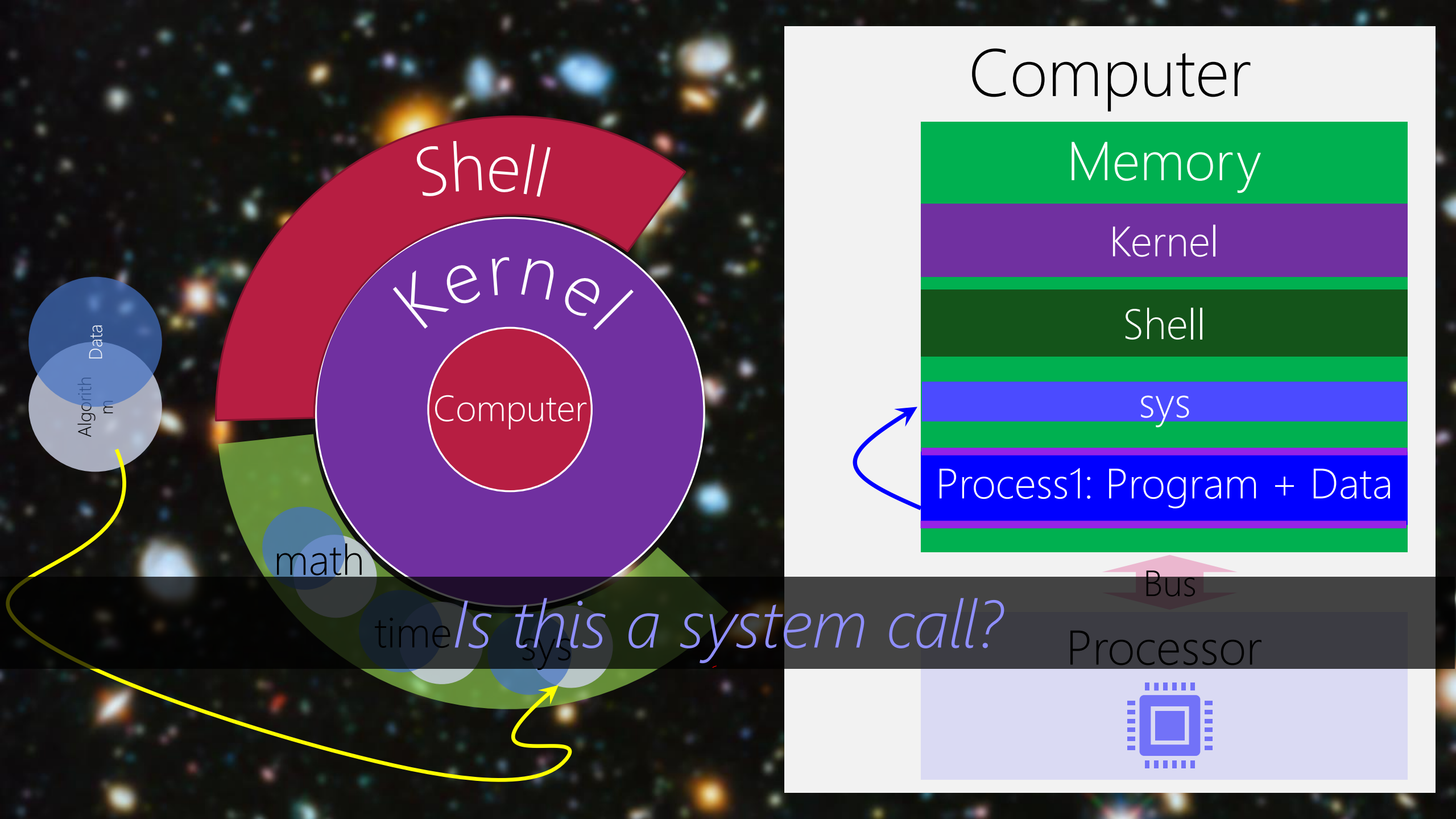
Computer

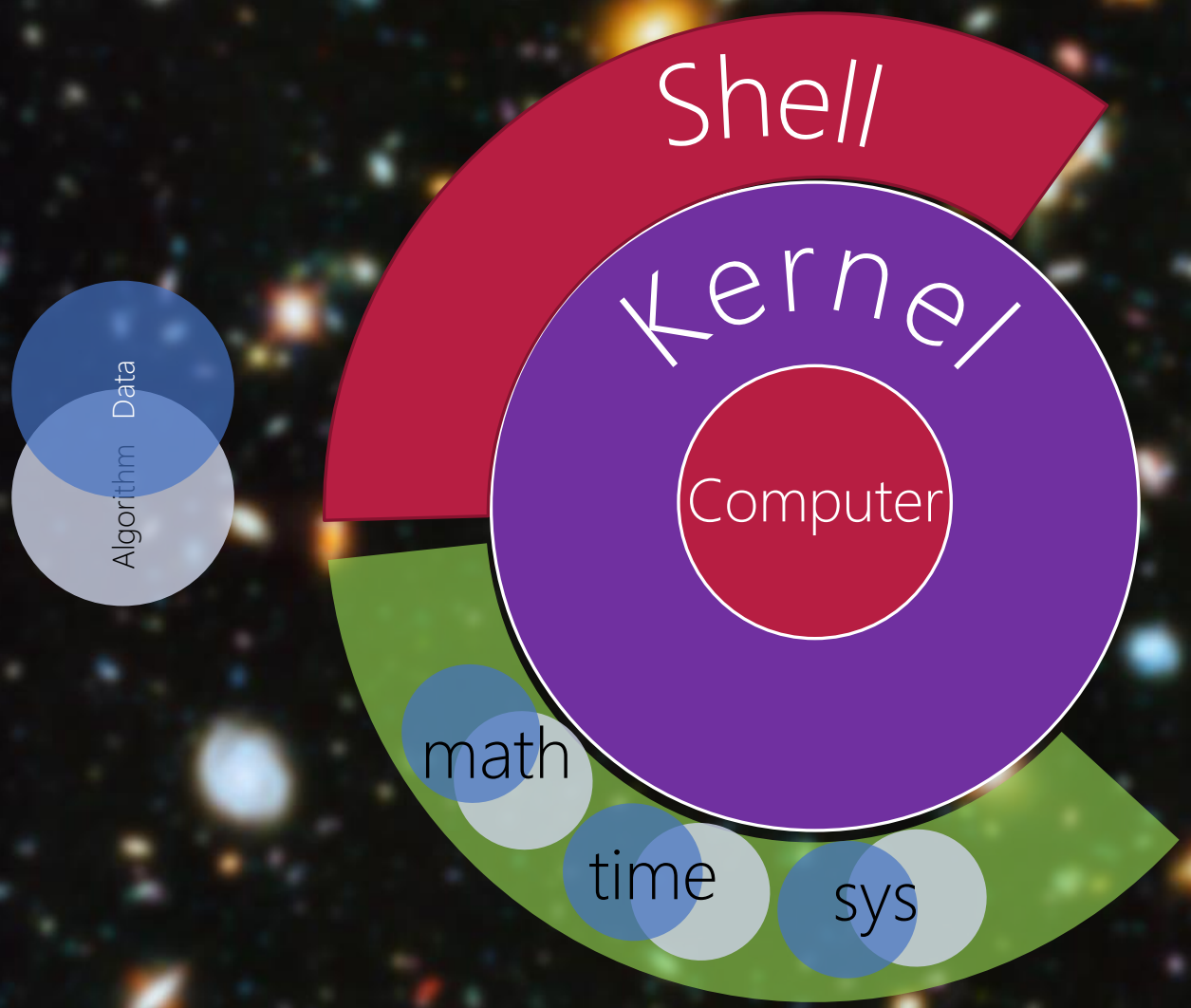

Memory
Kernel
Shell
sys
Process1: Program + Data
Bus
Processor

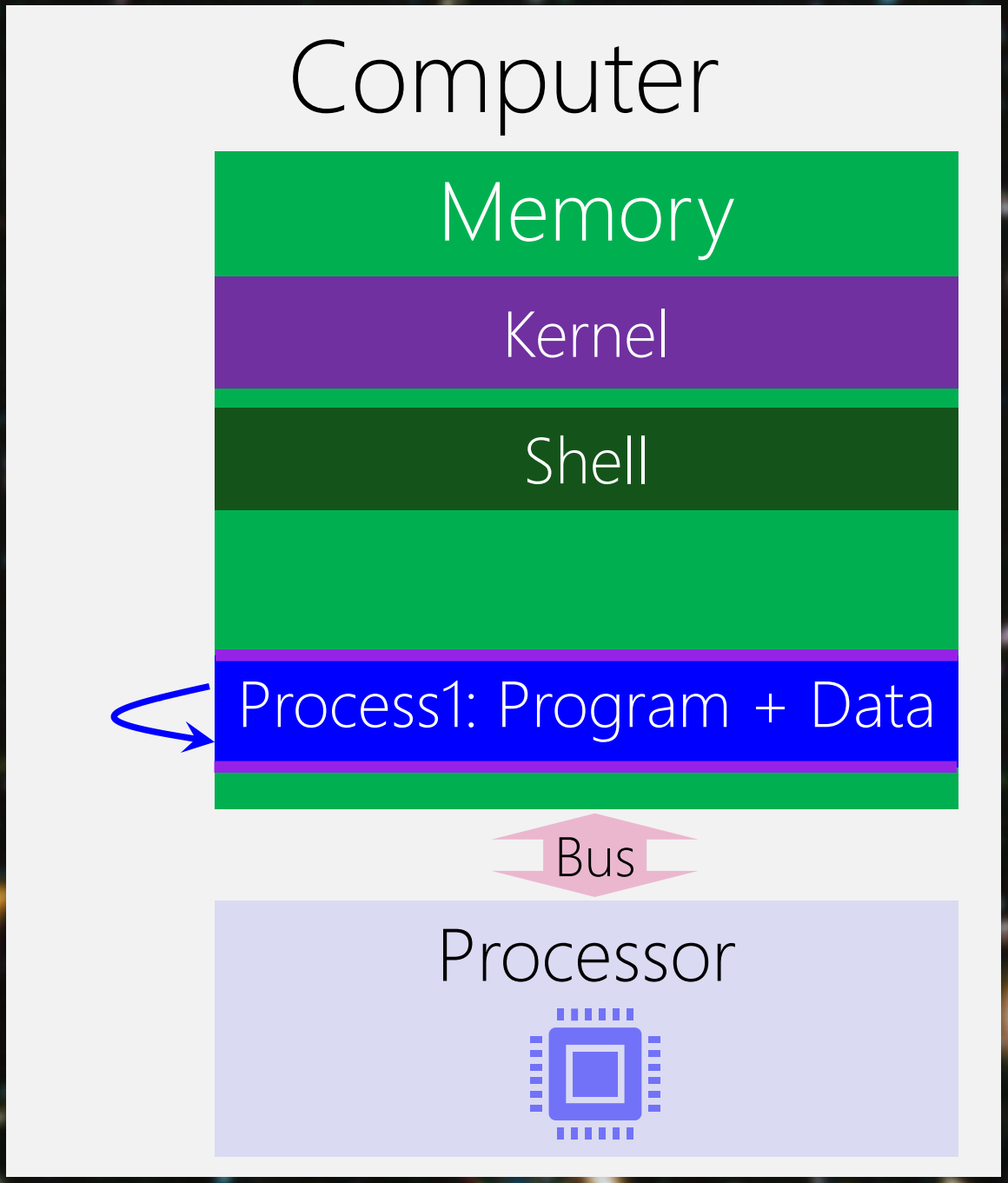

Shell
Kernel
Computer
Data
Algorithm
math
time
sys

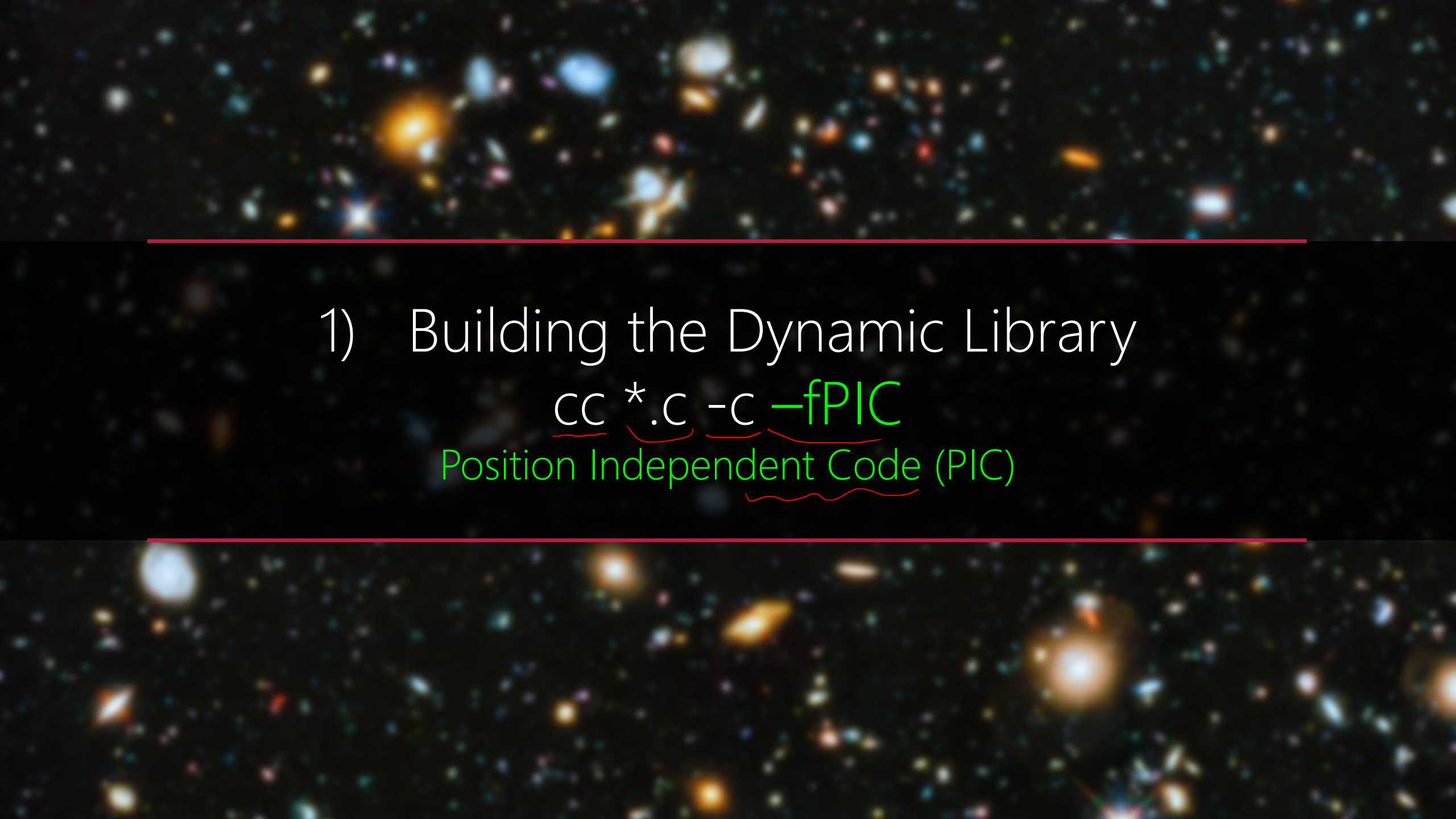
Computer


Memory
Kernel
Shell
sys
Process1: Program + Data
Bus
Processor


Shell
Kernel
Computer
Data
Algorithm
math
time
sys
Computer
Memory
Kernel
Shell
sys
Process1: Program + Data
Bus
Processor
Is this a system call?


Shell
Kernel
Computer
Algorithm
Data
math
time
sys


Computer
Memory
Kernel
Shell
Process1: Program + Data
Bus
Processor



1) Building the Dynamic Library

cc *.c -c –fPIC

Position Independent Code (PIC)

```
int add(int x, int y){
        return x + y;
}
```

```
int sub(int x, int y){
        return x - y;
}
```

multiply

sqrt

Roots of a quadratic equation!

```
cc add.c -c -fPIC
```

```
0001 0000 0000 0000 0000 0000 0000 0000
0011 0000 0003 0000 0000 0000 0000 0000
0000 0000 0000 0000 3309 0000 0000 0000
0096 0000 0000 0000 0000 0000 0000 0000
```

add.o

```
cc sub.c -c -fPIC
```

```
0111 0110 0000 0000 0000 0000 0000 0000
0011 0000 0003 0000 0000 0000 0000 0000
0000 0000 0000 0000 3309 0000 0000 0000
0096 0000 0000 0000 0000 0000 0000 0000
```

sub.o

```
cc multi.c -c -fPIC
```

```
0001 0000 0000 0000 0000 0000 0000 0000
0011 0000 0002 0000 0000 0000 0000 0000
0000 0000 1111 0000 3309 0000 0000 0000
0086 0000 0000 0000 0000 0000 0000 0000
```

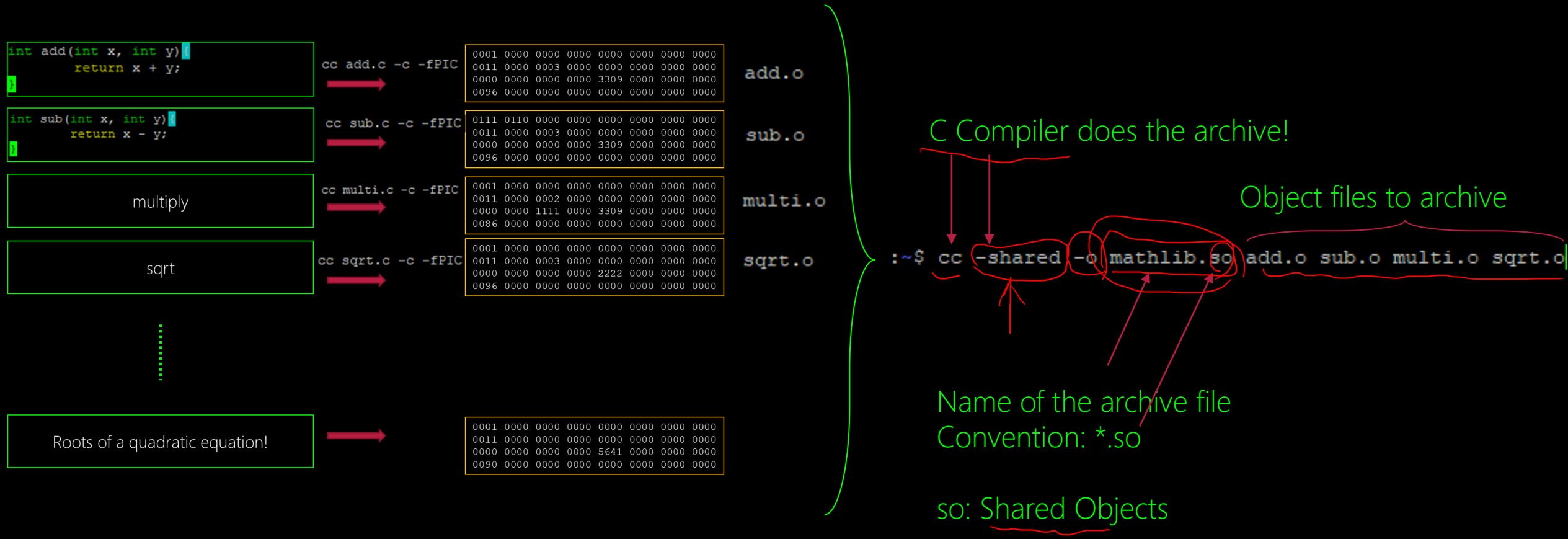
multi.o

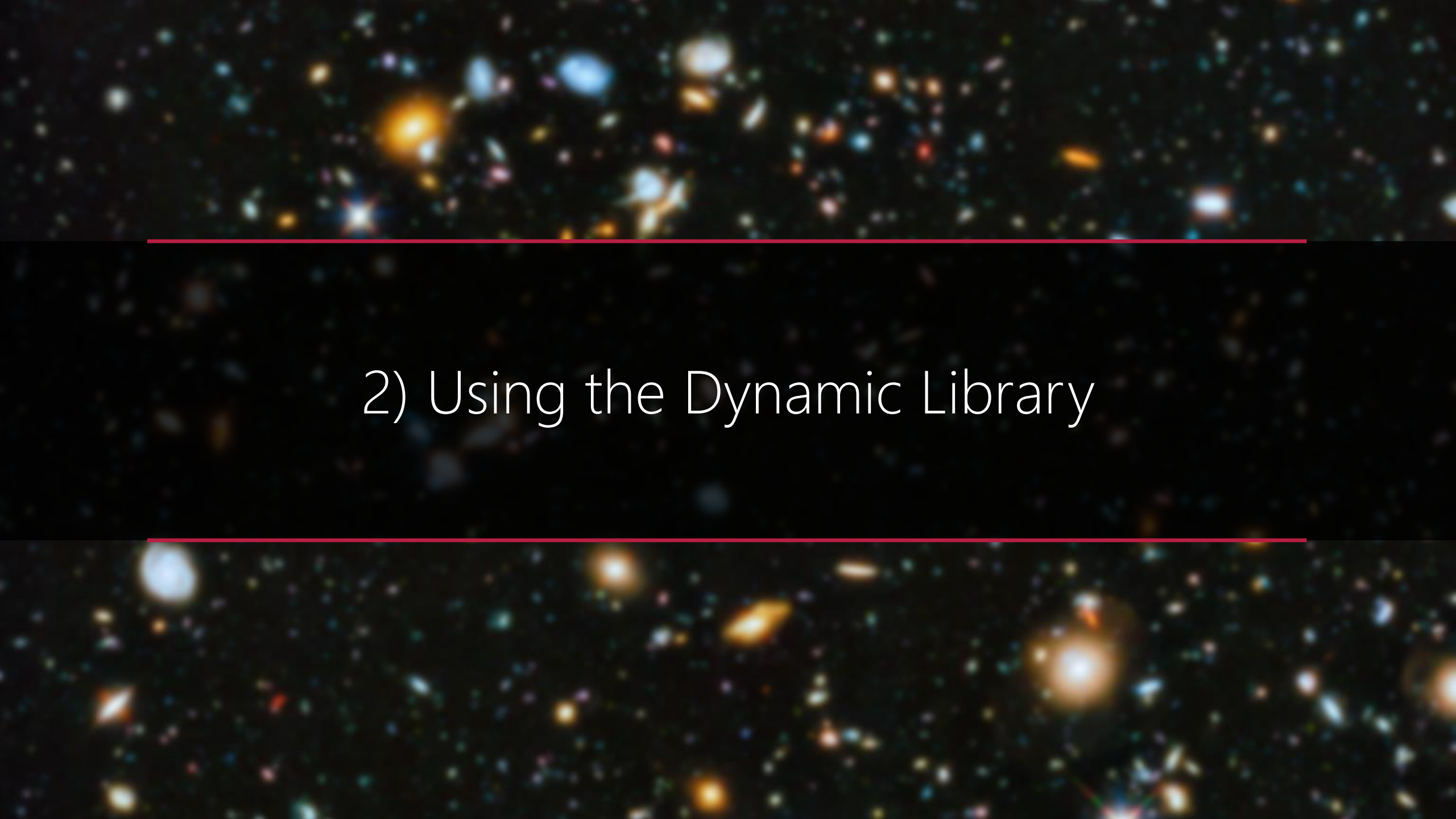
```
cc sqrt.c -c -fPIC
```

```
0001 0000 0000 0000 0000 0000 0000 0000
0011 0000 0003 0000 0000 0000 0000 0000
0000 0000 0000 0000 2222 0000 0000 0000
0096 0000 0000 0000 0000 0000 0000 0000
```

sqrt.o

```
0001 0000 0000 0000 0000 0000 0000 0000
0011 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 5641 0000 0000 0000
0090 0000 0000 0000 0000 0000 0000 0000
```


int add(int x, int y){
return x + y;
}
cc add.c -c -fPIC
0001 0000 0000 0000 0000 0000 0000 0000
0011 0000 0003 0000 0000 0000 0000 0000
0000 0000 0000 0000 3309 0000 0000 0000
0096 0000 0000 0000 0000 0000 0000 0000
add.o
int sub(int x, int y){
return x - y;
}
cc sub.c -c -fPIC
0111 0110 0000 0000 0000 0000 0000 0000
0011 0000 0003 0000 0000 0000 0000 0000
0000 0000 0000 0000 3309 0000 0000 0000
0096 0000 0000 0000 0000 0000 0000 0000
sub.o
multiply
cc multi.c -c -fPIC
0001 0000 0000 0000 0000 0000 0000 0000
0011 0000 0002 0000 0000 0000 0000 0000
0000 0000 1111 0000 3309 0000 0000 0000
0086 0000 0000 0000 0000 0000 0000 0000
multi.o
sqrt
cc sqrt.c -c -fPIC
0001 0000 0000 0000 0000 0000 0000 0000
0011 0000 0003 0000 0000 0000 0000 0000
0000 0000 0000 0000 2222 0000 0000 0000
0096 0000 0000 0000 0000 0000 0000 0000
sqrt.o
Roots of a quadratic equation!
0001 0000 0000 0000 0000 0000 0000 0000
0011 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 5641 0000 0000 0000
0090 0000 0000 0000 0000 0000 0000 0000
C Compiler does the archive!
Object files to archive
:~$ cc -shared -o mathlib.so add.o sub.o multi.o sqrt.o
Name of the archive file
Convention: *.so
so: Shared Objects



2) Using the Dynamic Library

```c
#include <stdio.h>

#include "mathlib.h"
void main(void){
        printf("2 + 3 = %d\n", add(2,3));
        printf("2 - 3 = %d\n", sub(2,3));
}
```

```
:~$ vi mathlib.h
int add(int, int);
int sub(int, int);
int multi(int, int);
int sqrt(int);
//...
~
~
```

The same as in using static library!

```c
#include <stdio.h>

#include "mathlib.h"
void main(void){
        printf("2 + 3 = %d\n", add(2,3));
        printf("2 - 3 = %d\n", sub(2,3));
}
```

```
hfani@alpha:~$ cc main.c -c
hfani@alpha:~$ cc main.o mathlib.so -o main
hfani@alpha:~$ ./main
./main: error while loading shared libraries: mathlib.so: cannot open shared o
```

Tell the shell the kernel where to find it

```
cc main.o ./mathlib.so -o main
```

OR copy your lib to the C linker (ld) standard location: LD_LIBRARY_PATH

OR add your library path to LD_LIBRARY_PATH

```c
#include <stdio.h>

#include "mathlib.h"
void main(void){
        printf("2 + 3 = %d\n", add(2,3));
        printf("2 - 3 = %d\n", sub(2,3));
}
```

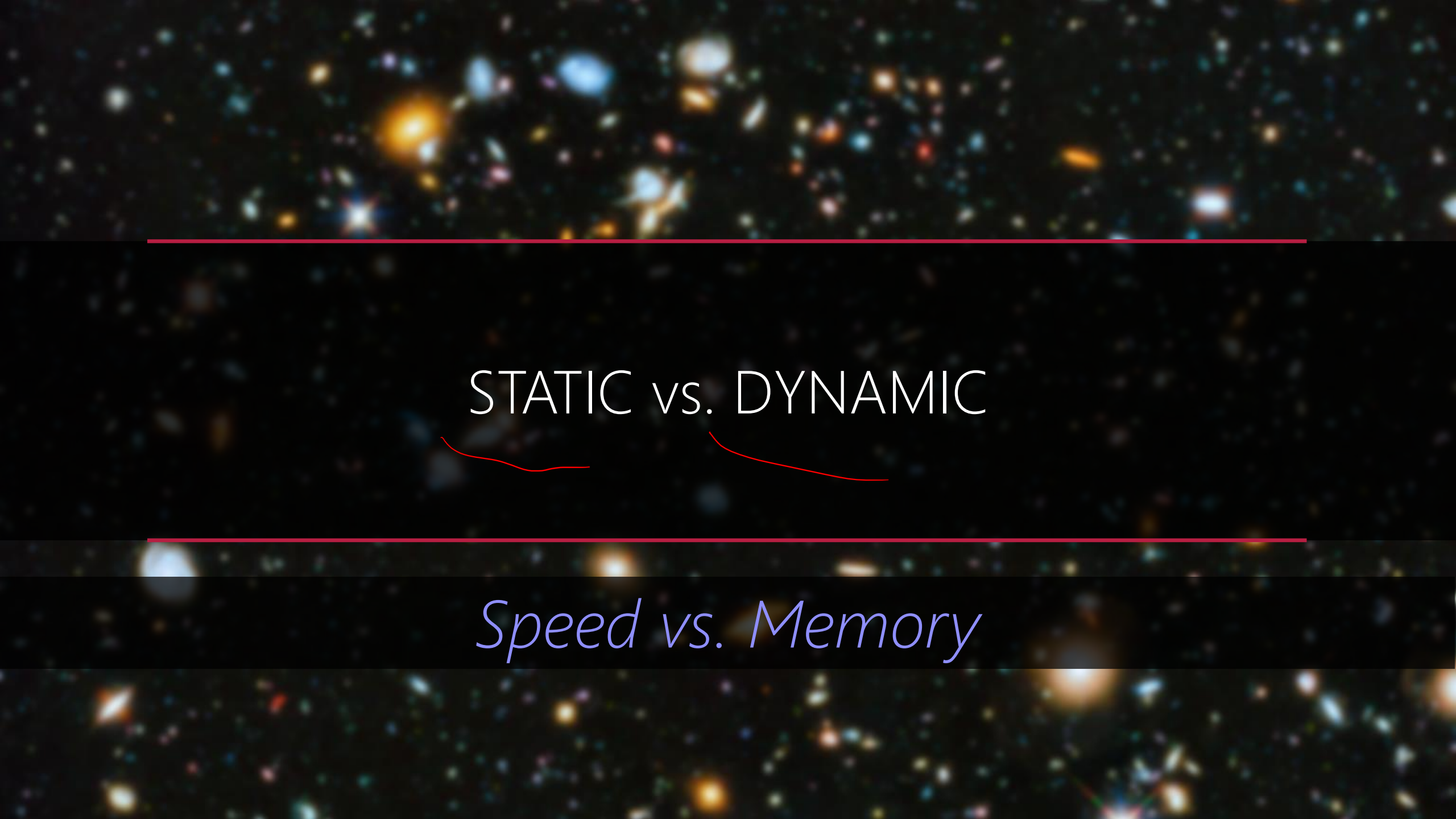
```
hfani@alpha:~$ cc main.c -c
hfani@alpha:~$ cc main.o mathlib.so -o main
hfani@alpha:~$ ./main
./main: error while loading shared libraries: mathlib.so: cannot open shared o

hfani@alpha:~$ LD_LIBRARY_PATH=$LD_LIBRARY_PATH:./
hfani@alpha:~$ export LD_LIBRARY_PATH
hfani@alpha:~$ ./main
2 + 3 = 5
2 - 3 = -1
```

System Variable for the path of shared libs
Other programs can use it!

```c
#include <stdio.h>

#include "mathlib.h"
void main(void){
        printf("2 + 3 = %d\n", add(2,3));
        printf("2 - 3 = %d\n", sub(2,3));
}
```

```
hfani@alpha:~$ cc main.c -c
hfani@alpha:~$ cc main.o mathlib.so -o main
hfani@alpha:~$ ./main
./main: error while loading shared libraries: mathlib.so: cannot open shared o
hfani@alpha:~$ LD_LIBRARY_PATH=$LD_LIBRARY_PATH:./
hfani@alpha:~$ export LD_LIBRARY_PATH
hfani@alpha:~$ ./main
2 + 3 = 5
2 - 3 = -1
hfani@alpha:~$ size ./main
   text    data     bss     dec     hex filename
      ?     616       8    2294     8f6 ./main
```

Does it include main.o only
Or also includes add.o, sub.o, ...?


STATIC vs. DYNAMIC
Speed vs. Memory

STATIC vs. DYNAMIC

*is *.h a static or dynamic library?*

When building our program, we never used any *.a or *.so in cc!

```
#include <stdio.h>
main(){
    printf(...);
}
```

```
stdio.h
stdlib.h
string.h
```

strcat
strlen

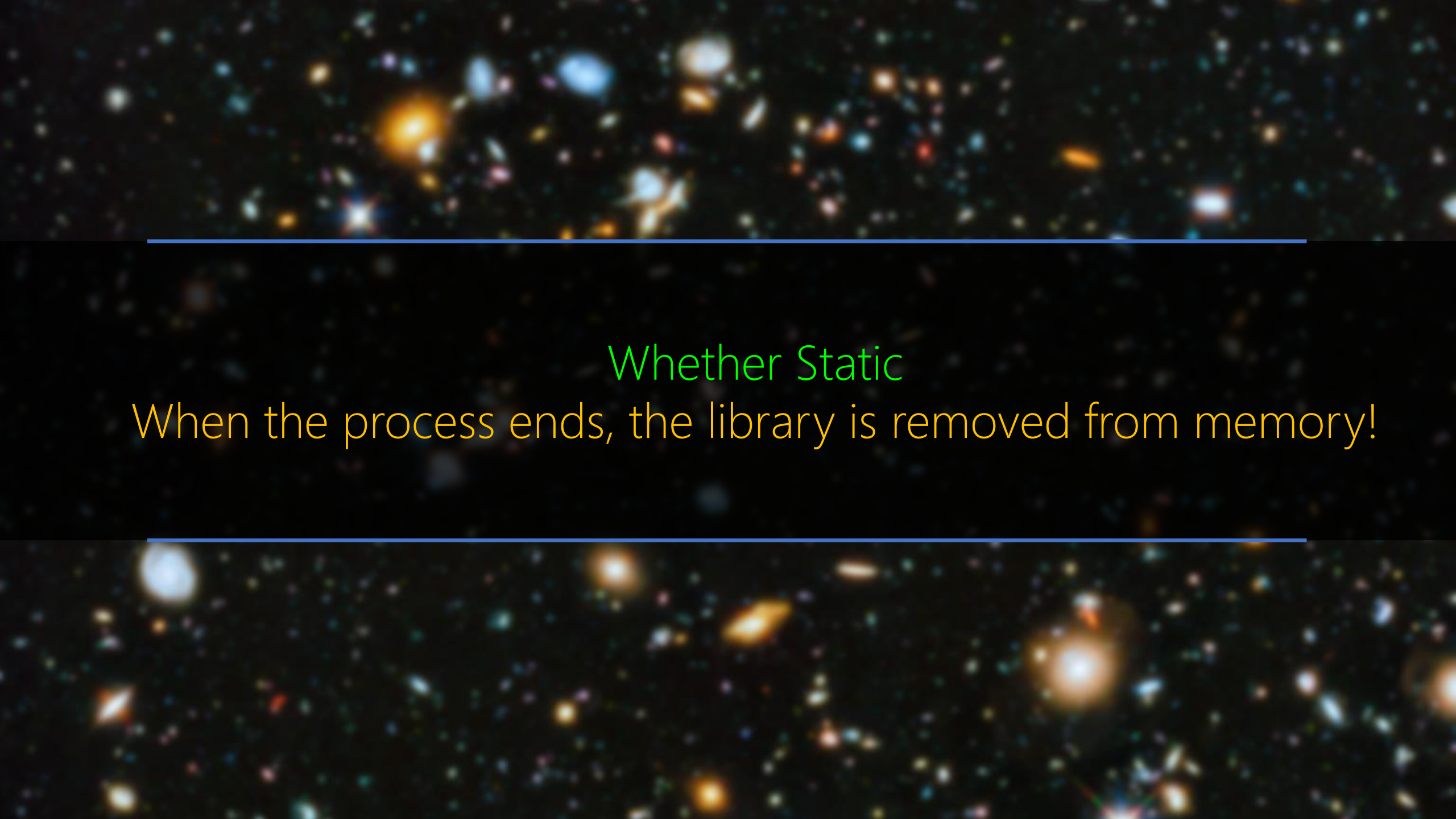
```
cc main.c -o main
./main
```

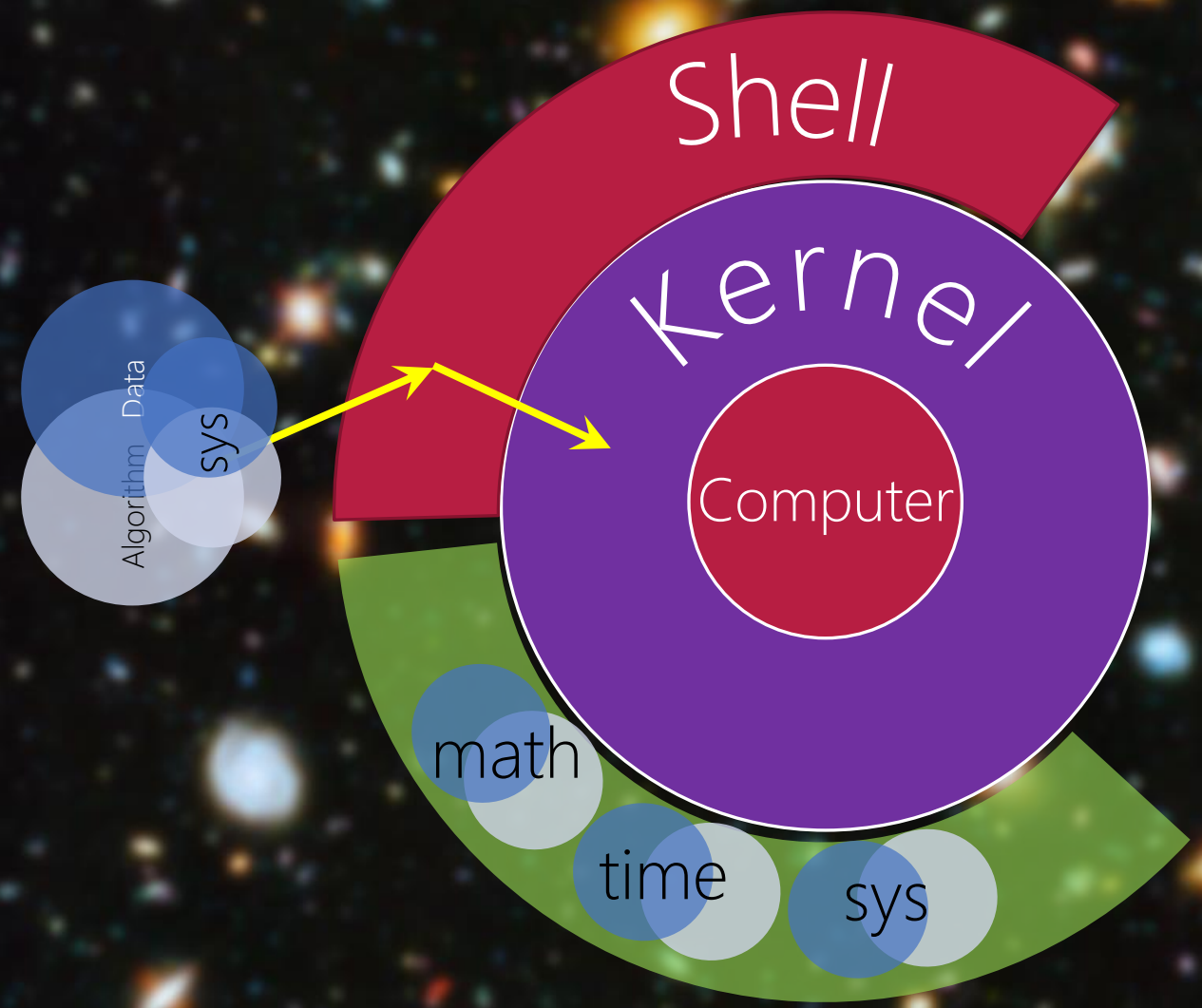
STATIC vs. DYNAMIC

List Dynamic Dependencies (ldd)

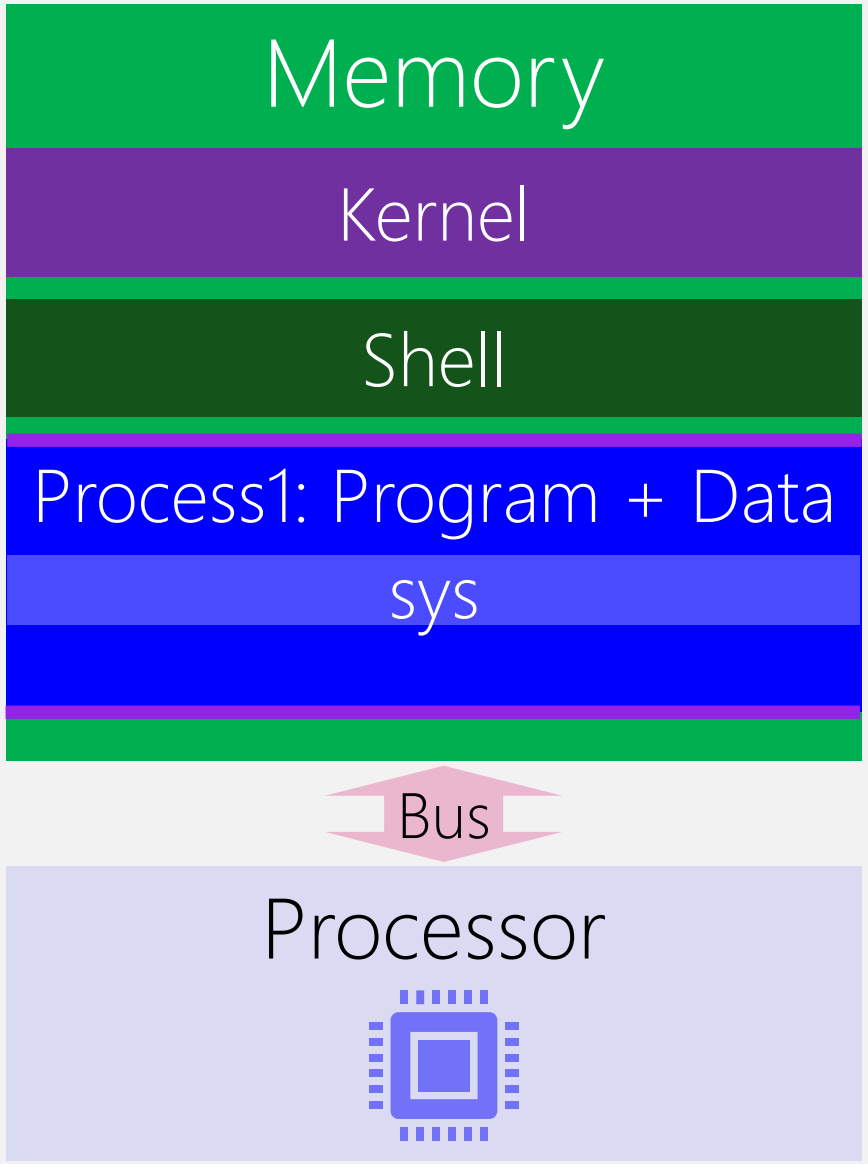
https://en.wikipedia.org/wiki/Ldd_(Unix)

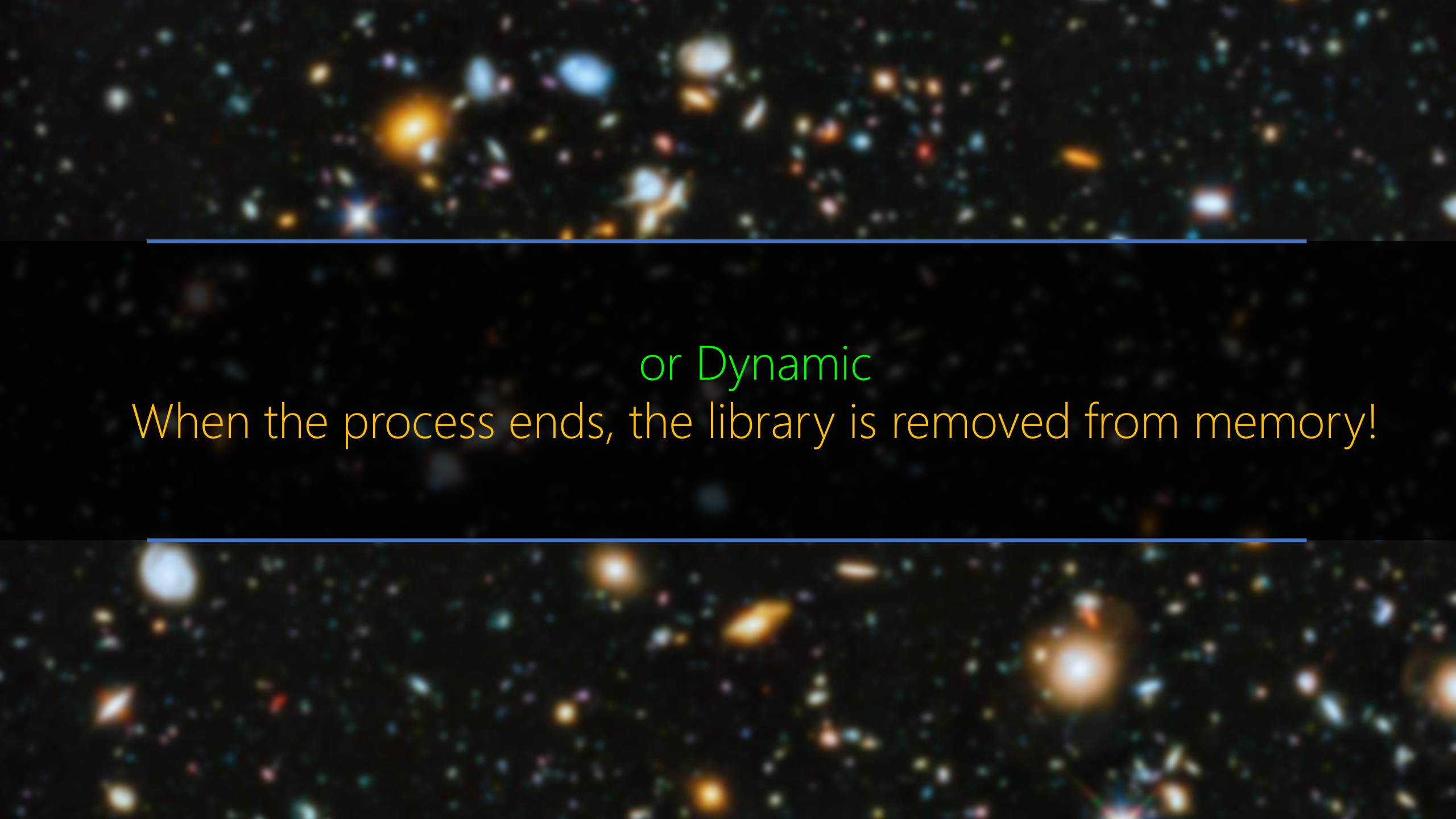
```
stdio.h
stdlib.h
string.h
```

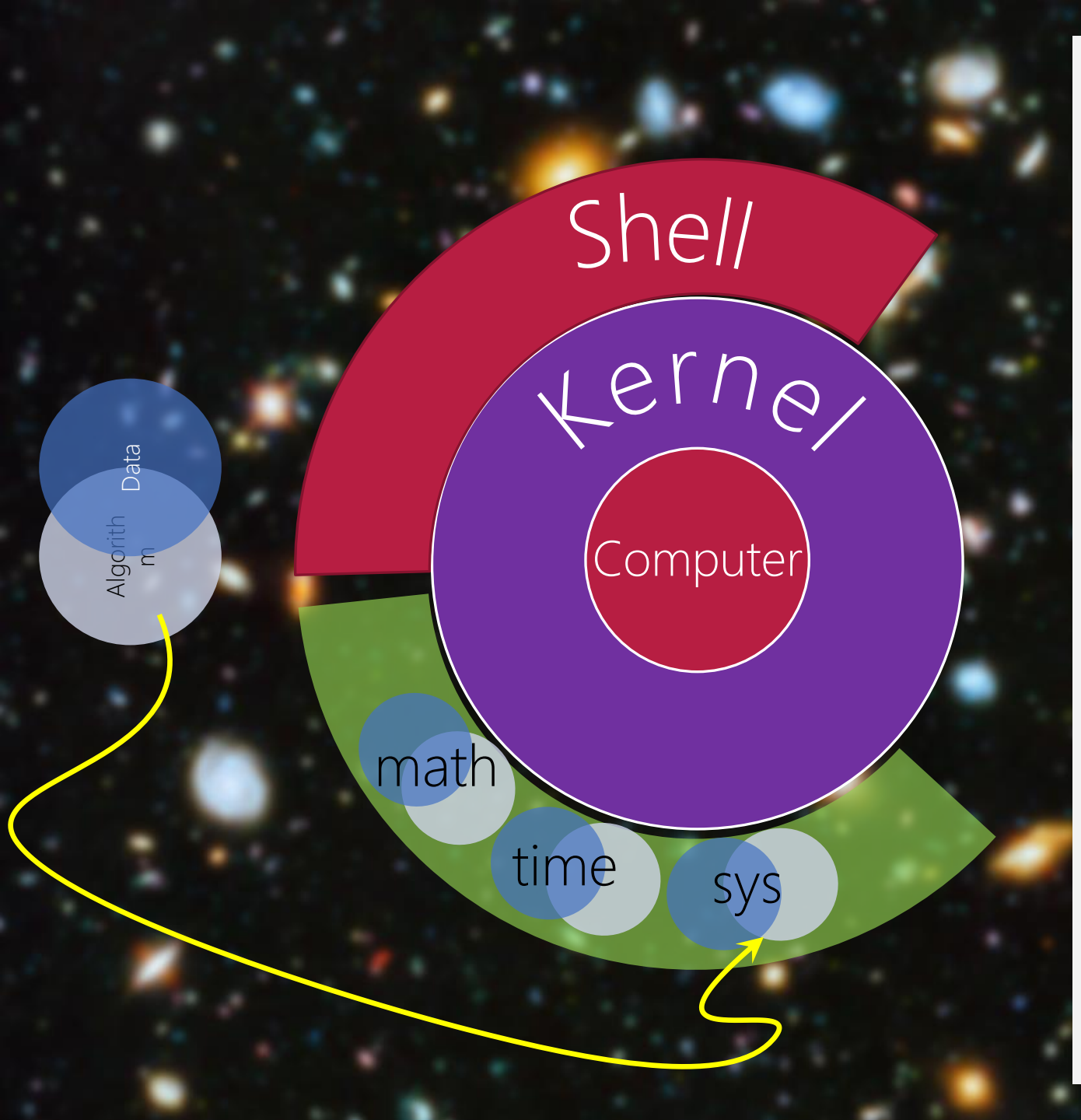

Whether Static
When the process ends, the library is removed from memory!


Shell
Kernel
Computer
Algorithm
Data
sys
math
time
sys

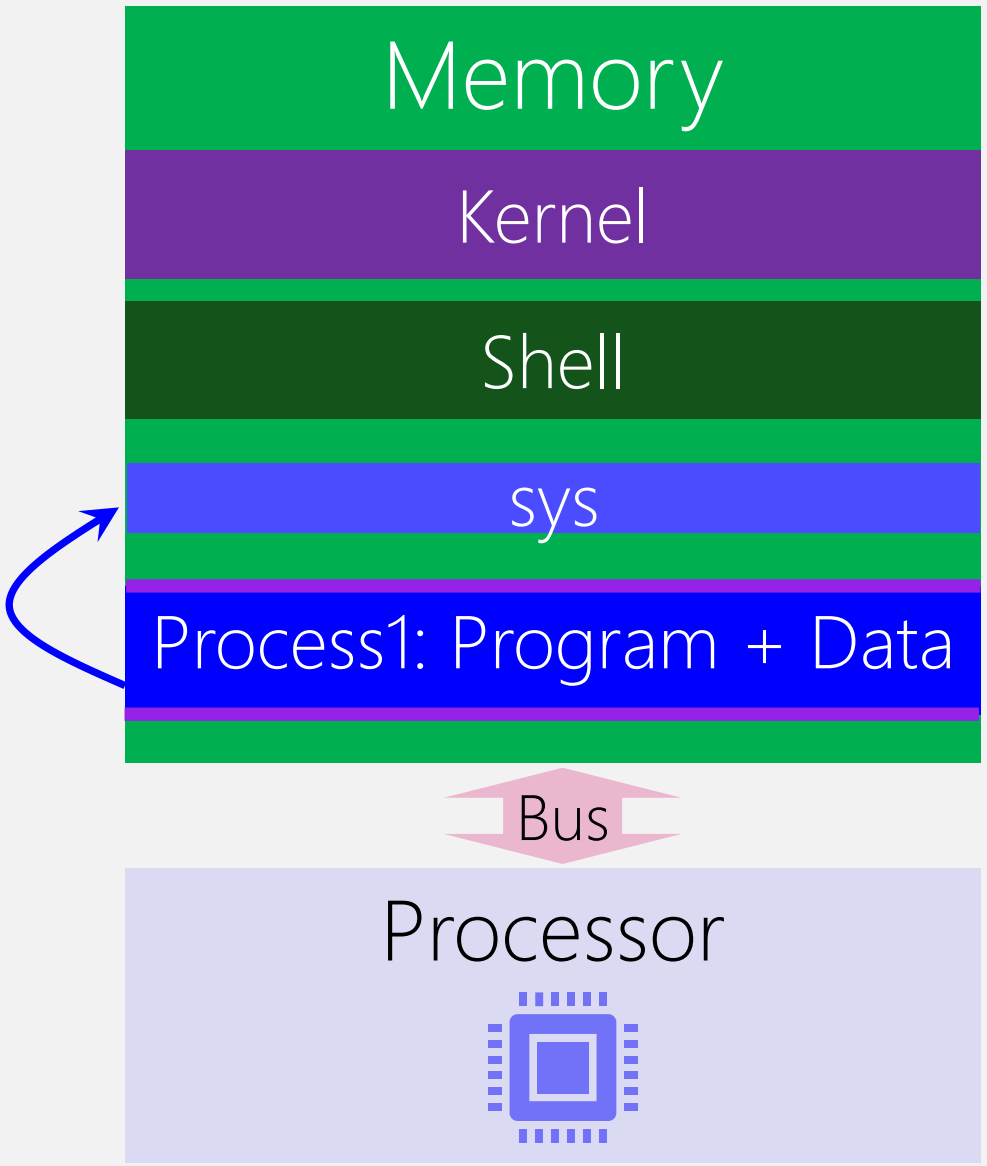
Computer

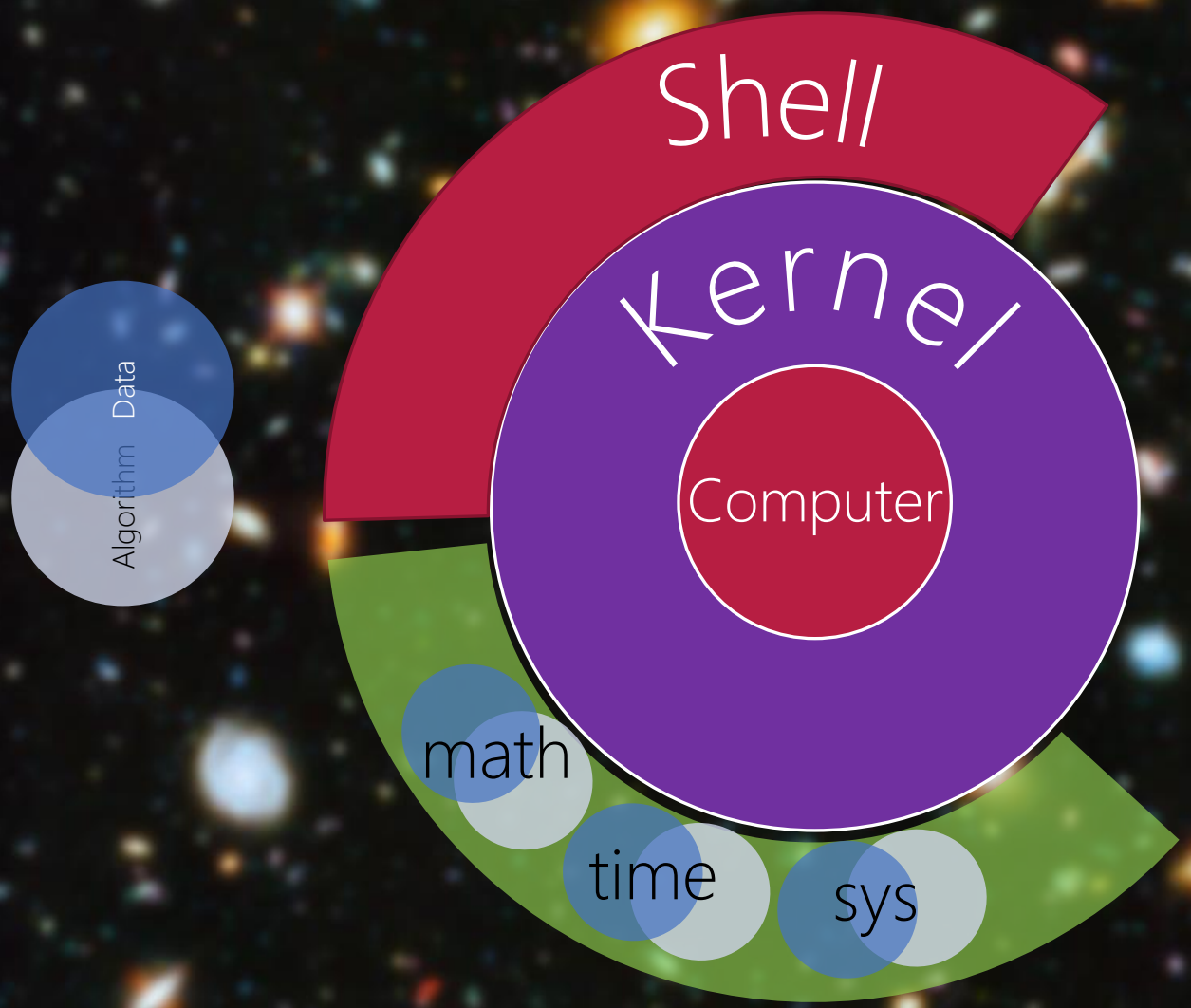

Memory
Kernel
Shell
Process1: Program + Data
sys
Bus
Processor

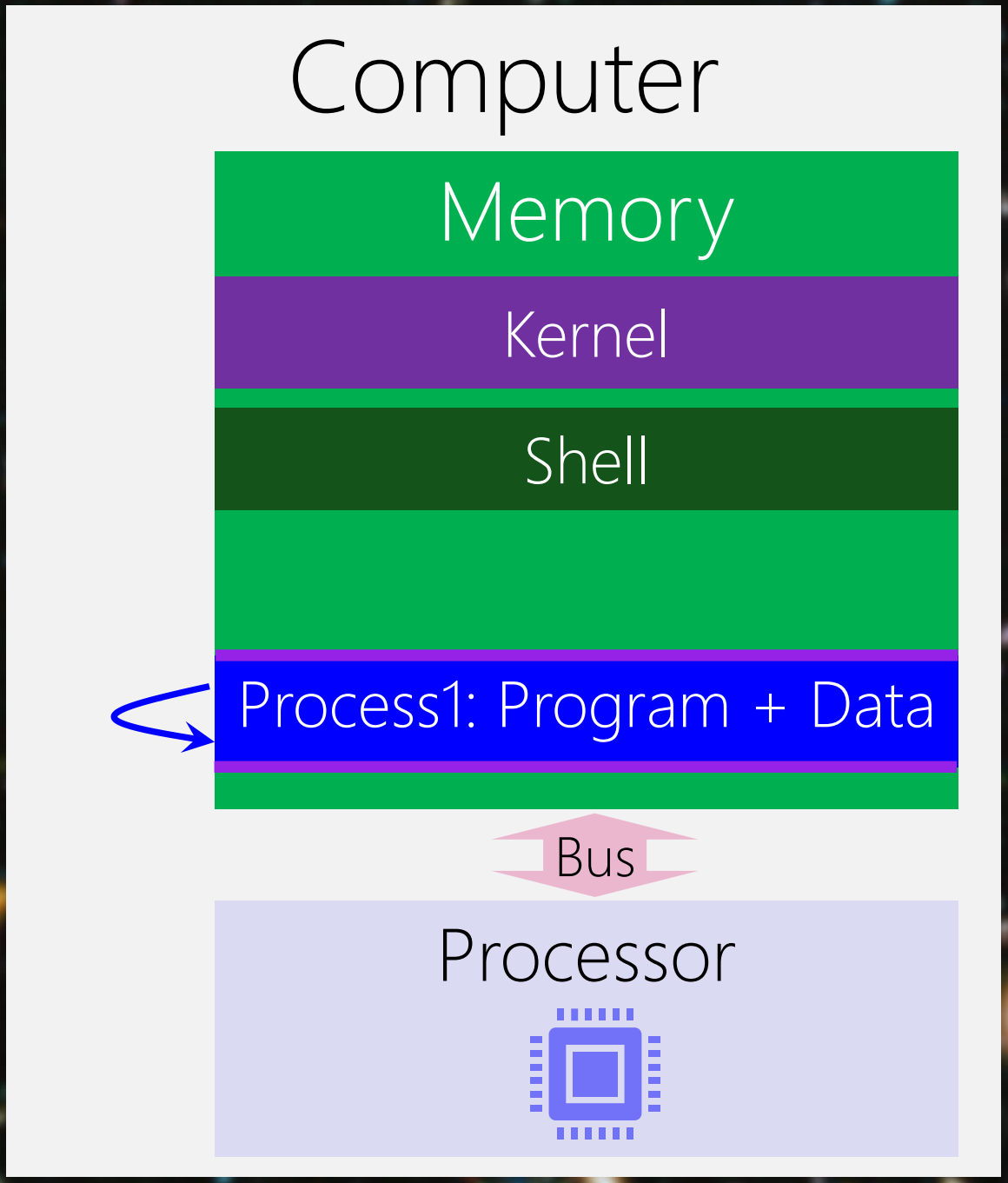

or Dynamic
When the process ends, the library is removed from memory!

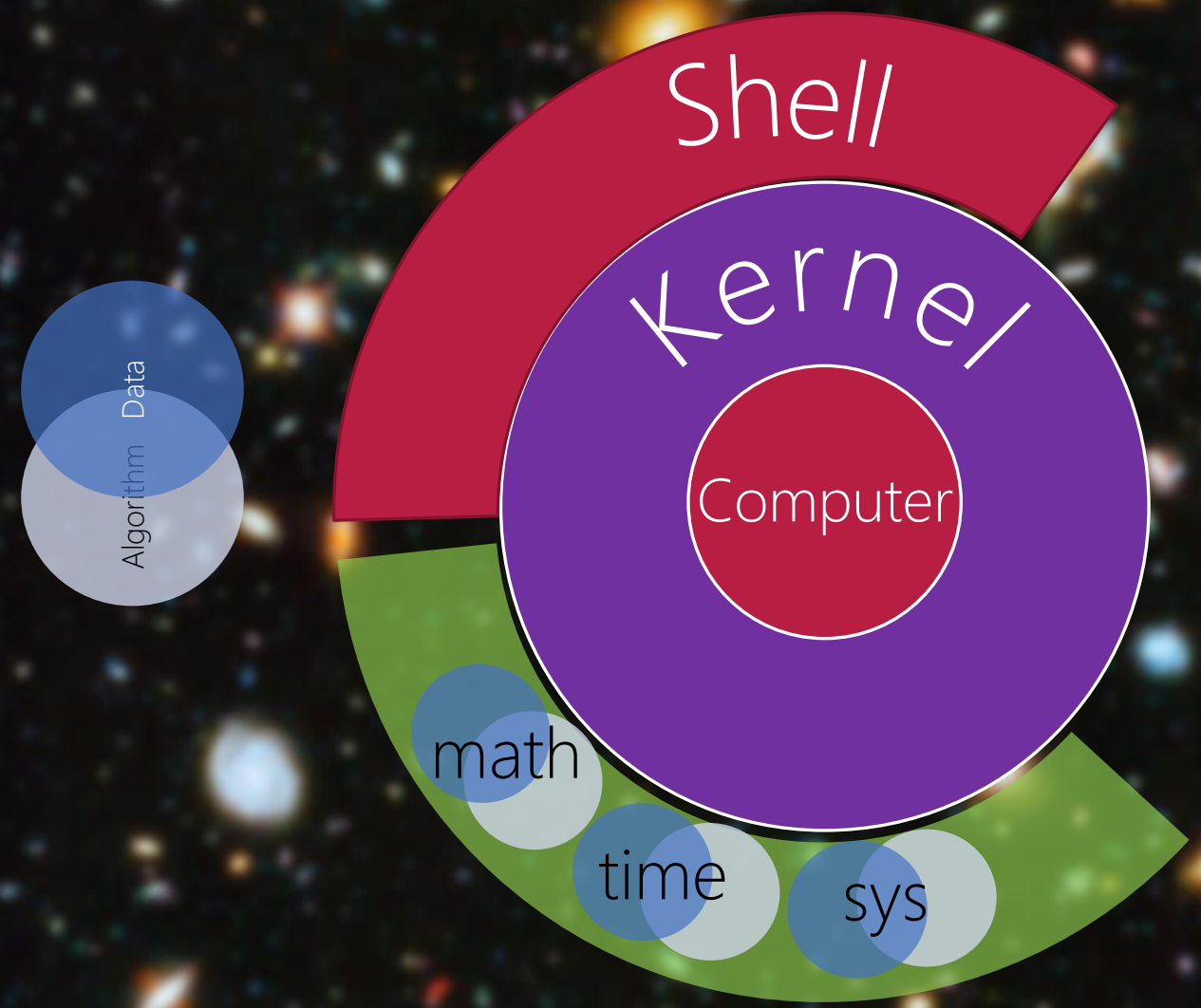

Shell
Kernel
Computer
Data
Algorithm
math
time
sys

Computer

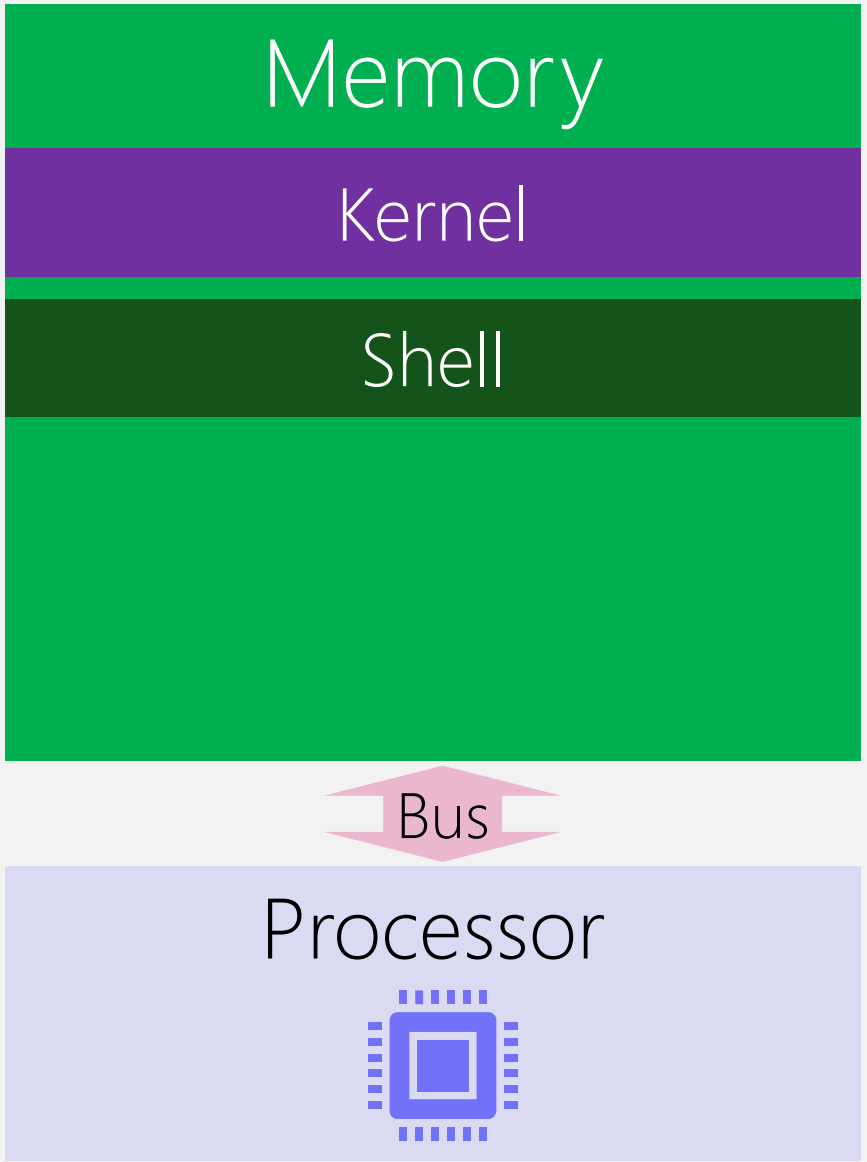

Memory
Kernel
Shell
sys
Process1: Program + Data
Bus
Processor

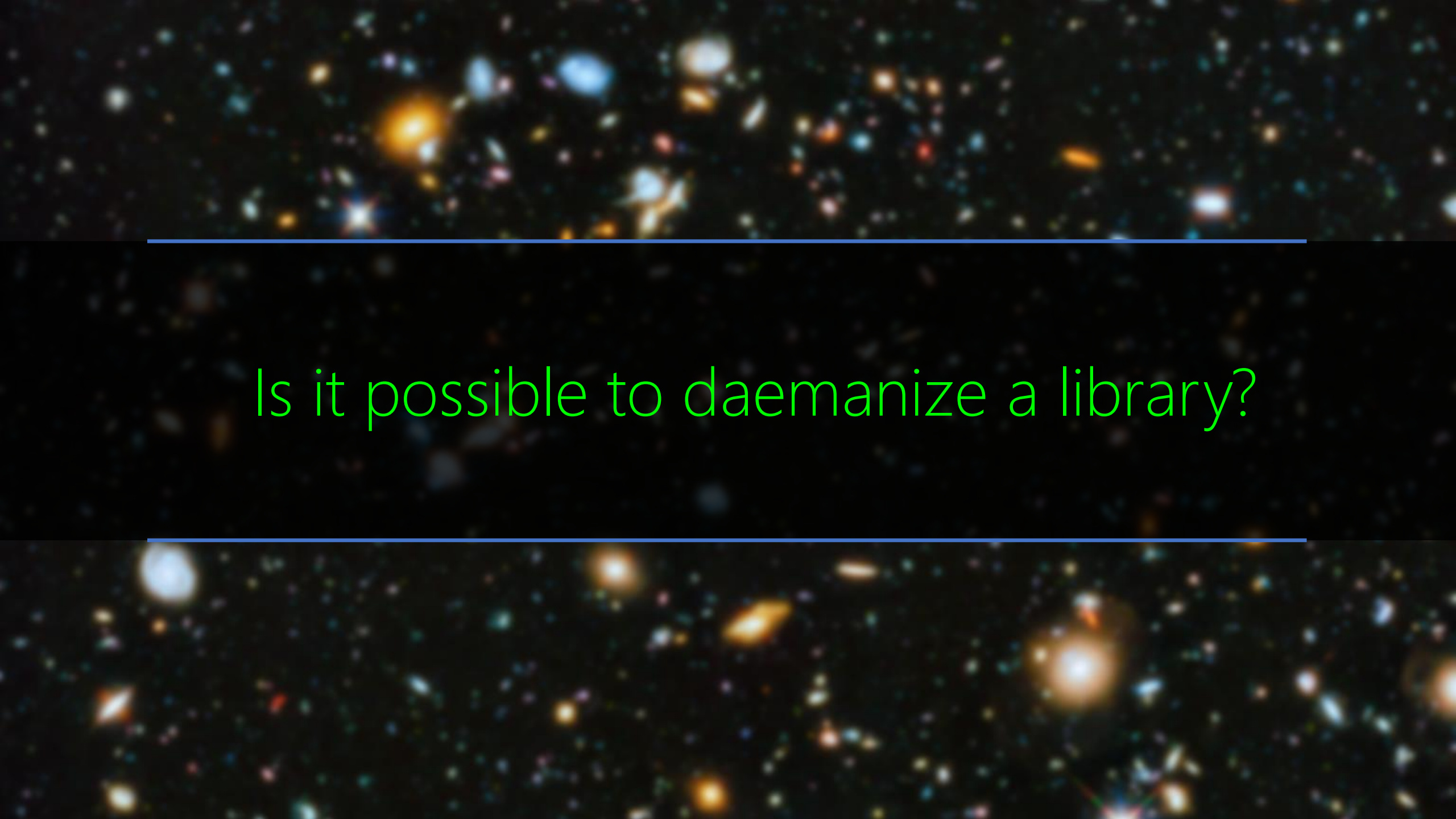

Shell
Kernel
Computer
Algorithm
Data
math
time
sys

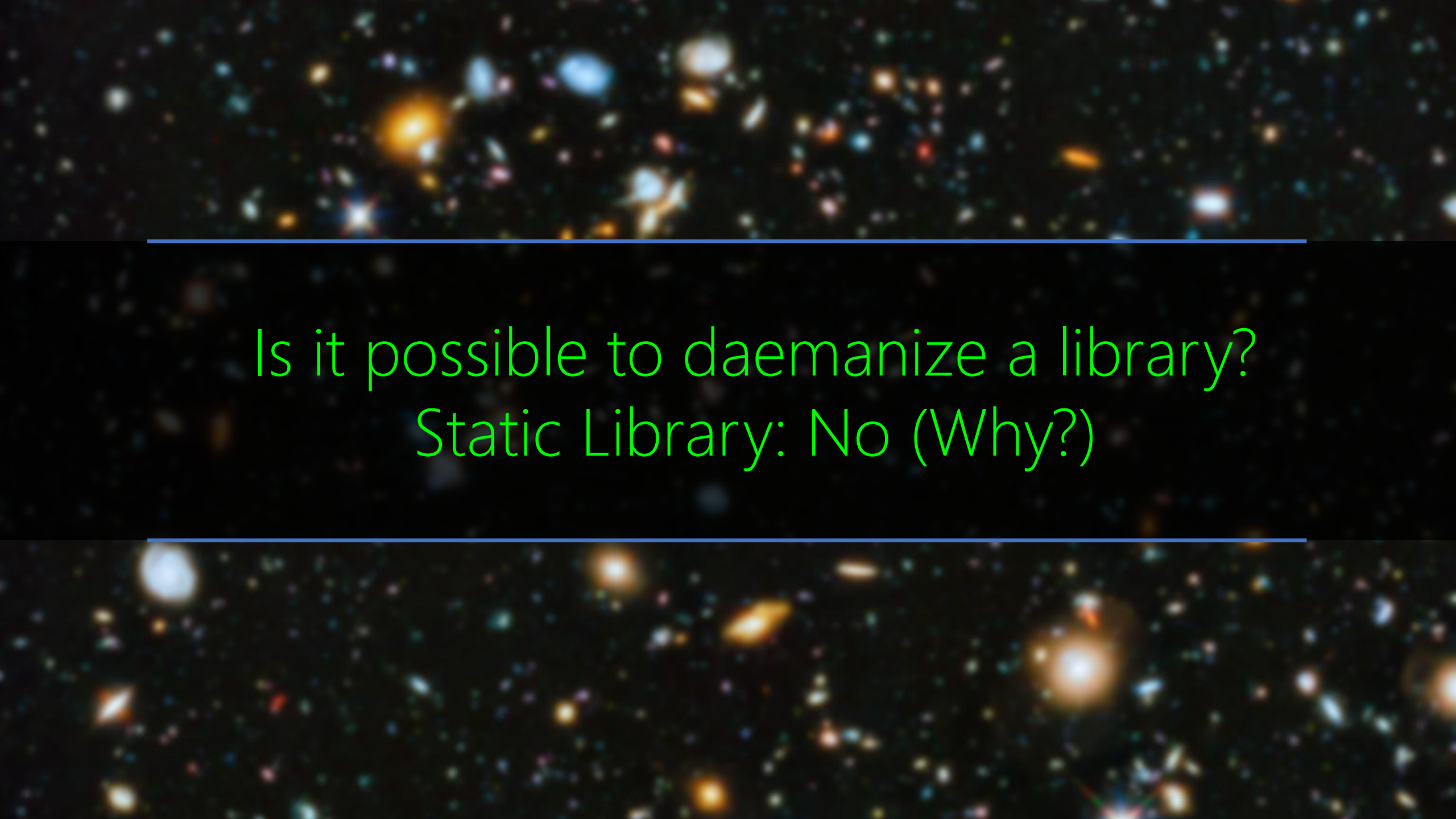

Computer
Memory
Kernel
Shell
Process1: Program + Data
Bus
Processor

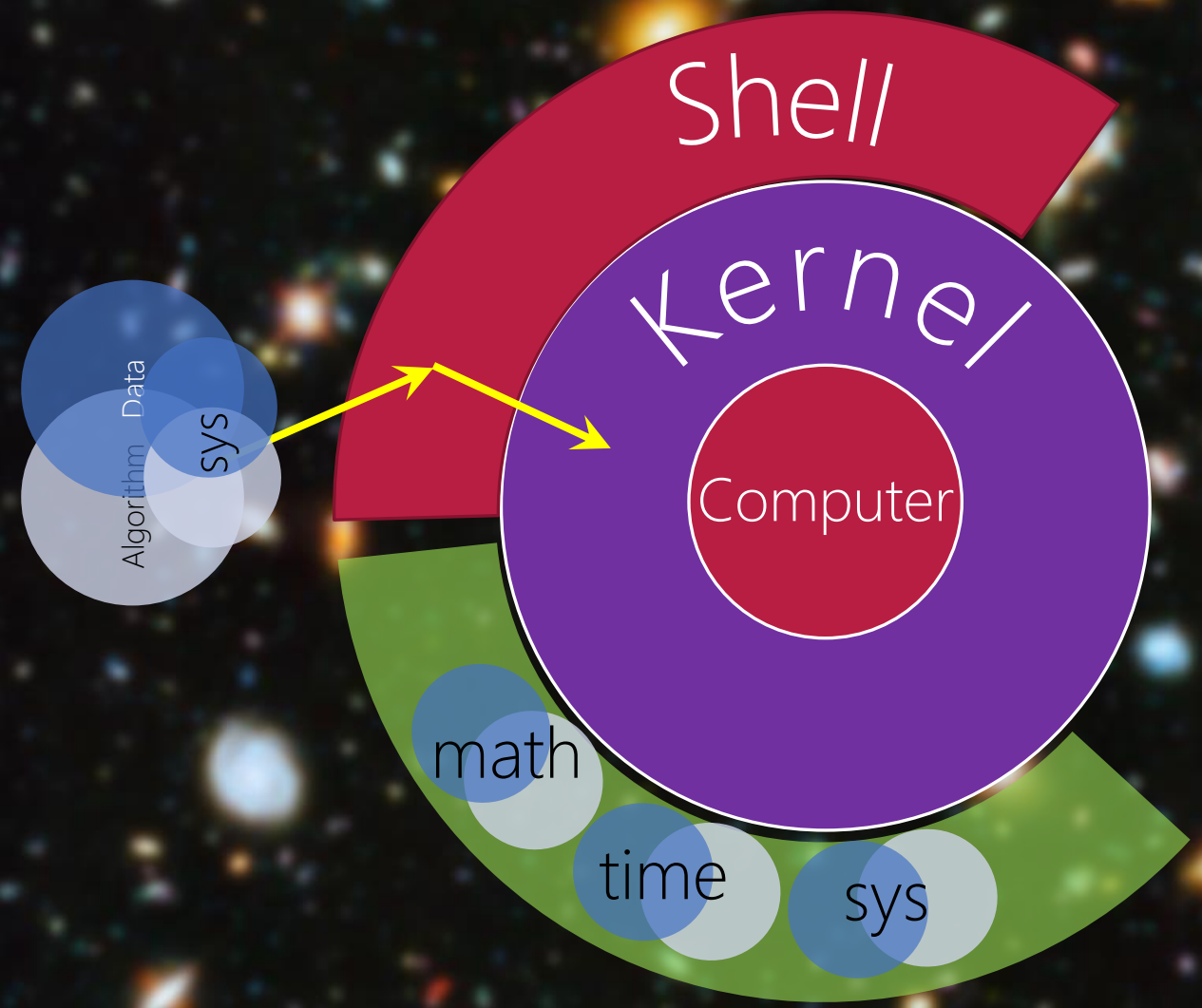

Shell
Kernel
Computer
Algorithm
Data
math
time
sys

Computer

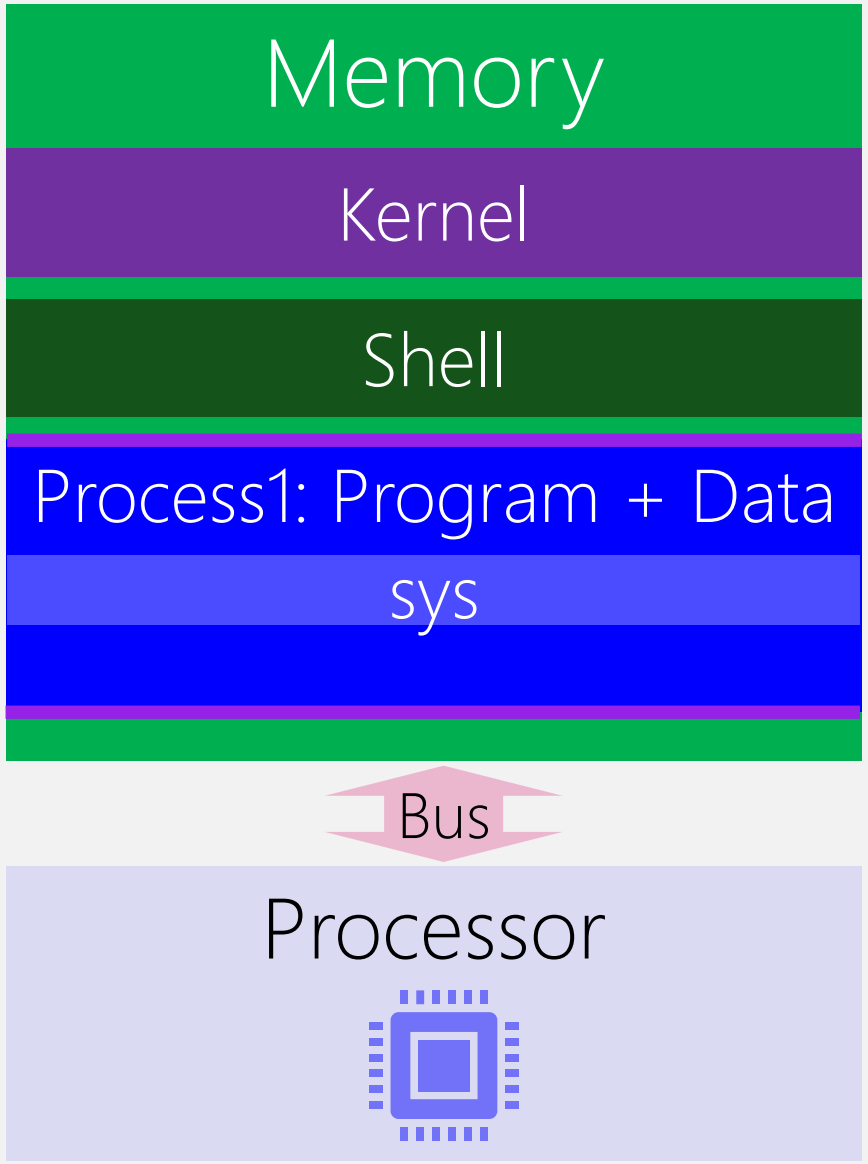

Memory
Kernel
Shell
Bus
Processor

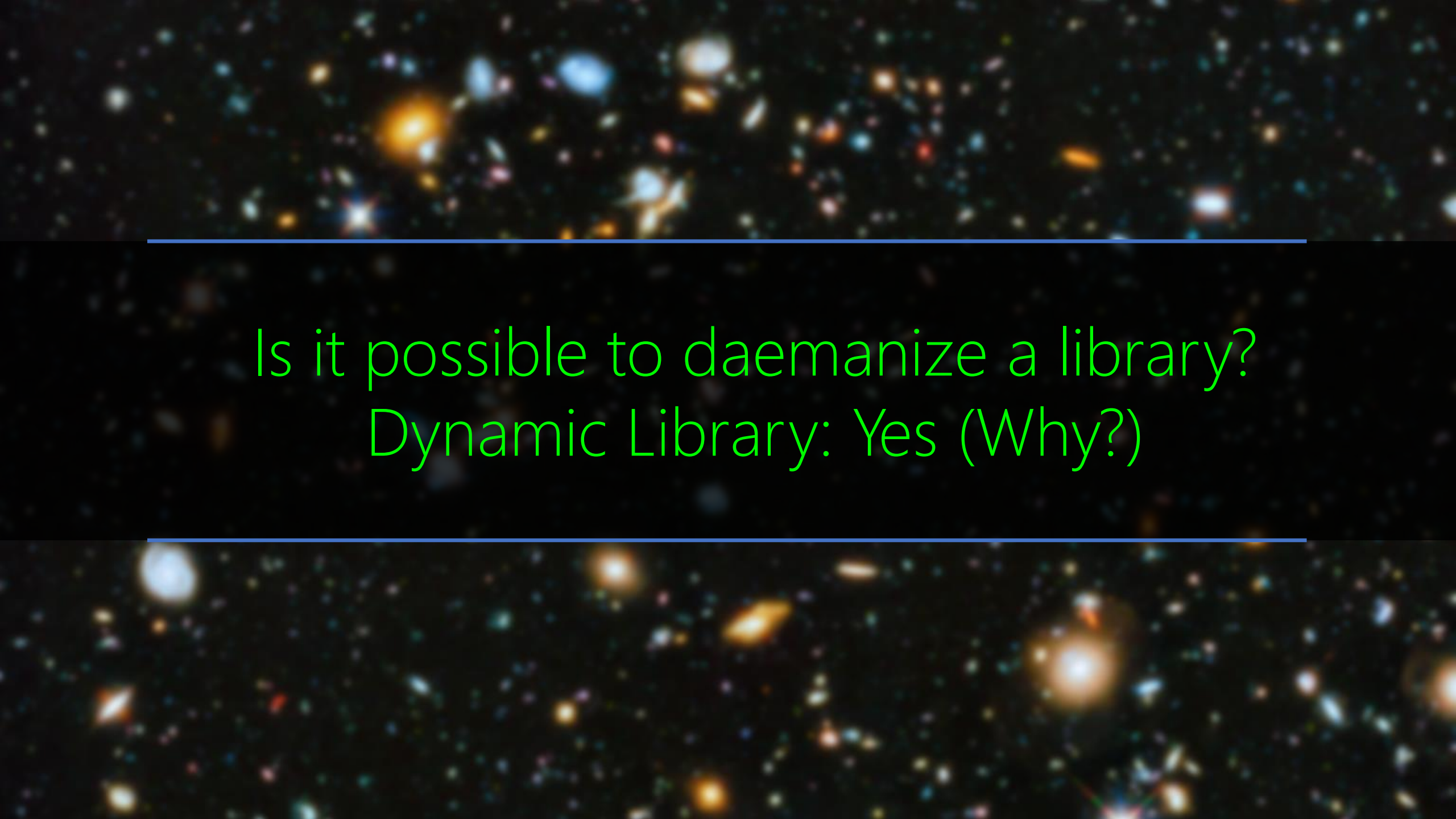

Is it possible to daemanize a library?

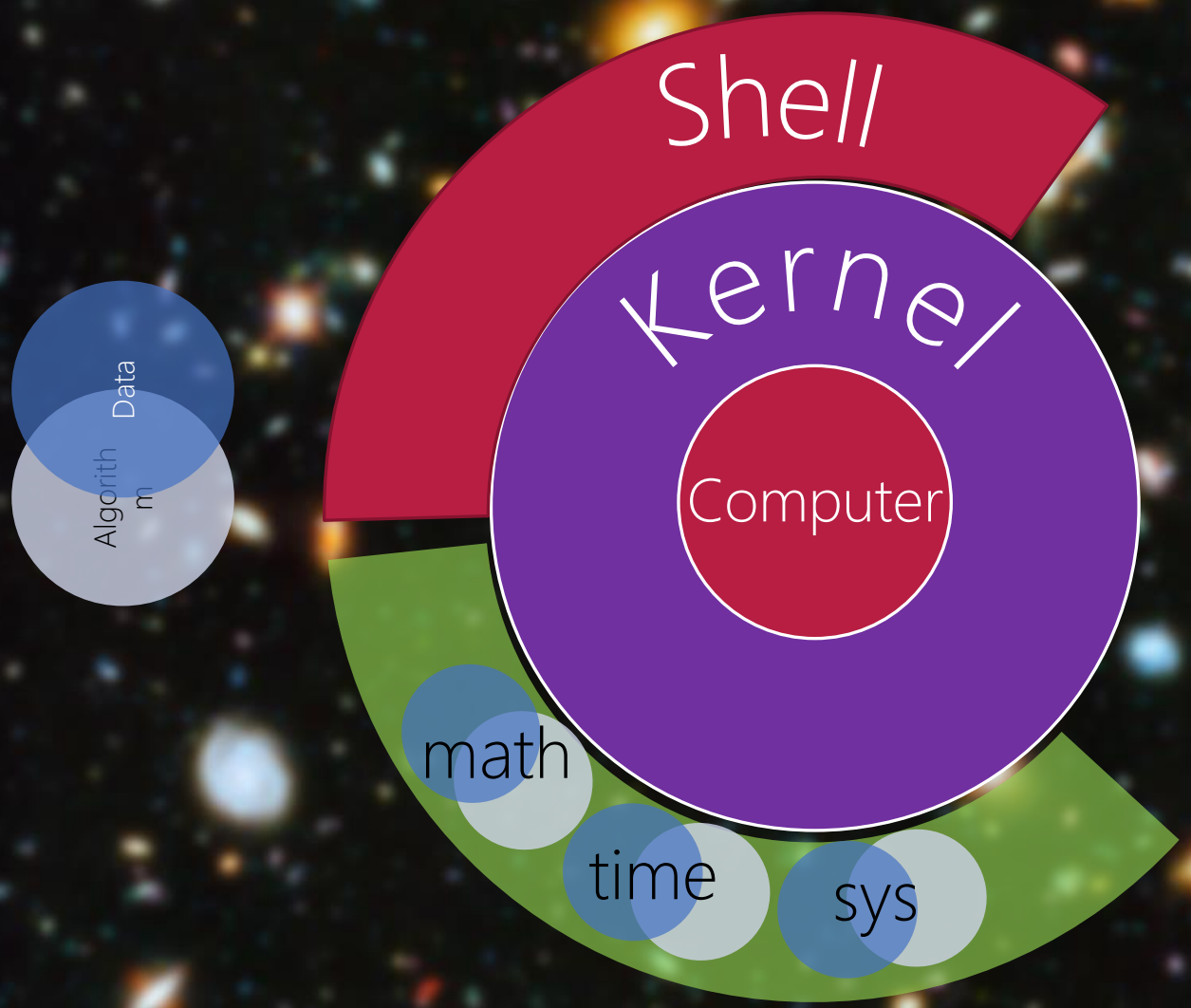

Is it possible to daemanize a library?
Static Library: No (Why?)


Shell
Kernel
Computer
Algorithm
Data
sys
math
time
sys

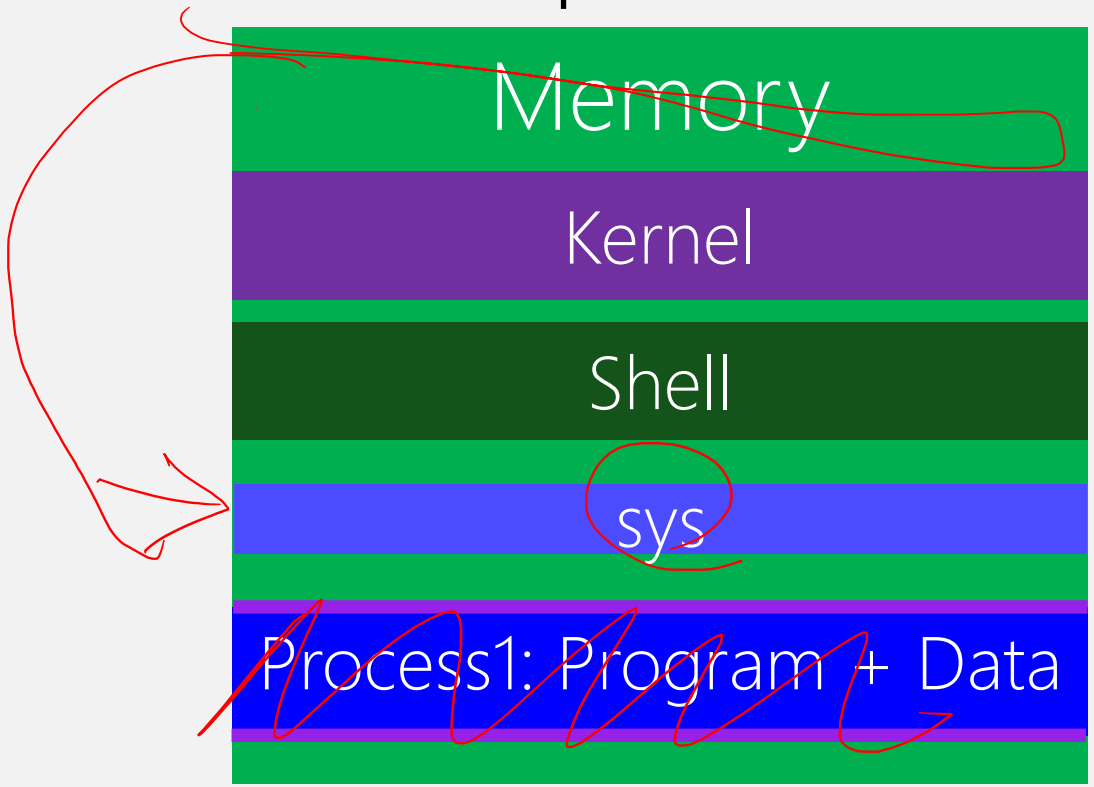
Computer


Memory
Kernel
Shell
Process1: Program + Data
sys
Bus
Processor


Is it possible to daemanize a library?
Dynamic Library: Yes (Why?)



Computer

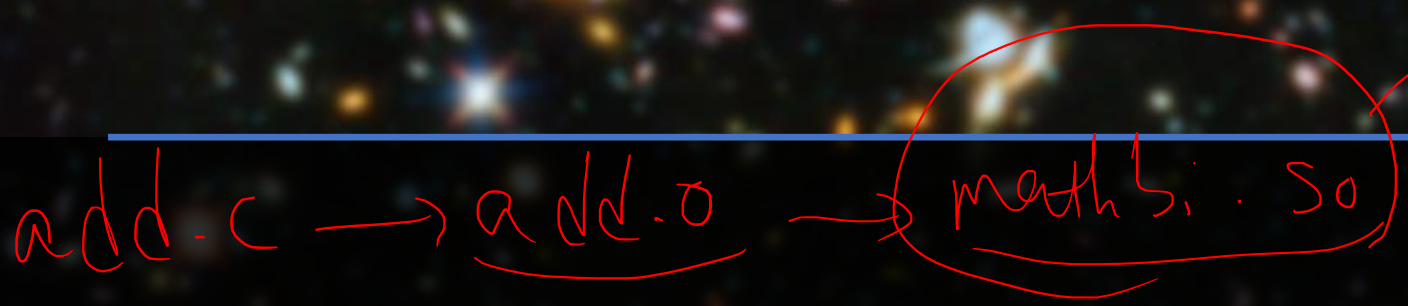


Bus

Processor


Shared Libraries
Daemon Dynamic Libraries
e.g., stdio.h → libc.so


Shared Libraries
Daemon Dynamic Libraries
e.g., stdio.h → libc.so
The sharing scope is within the computer!



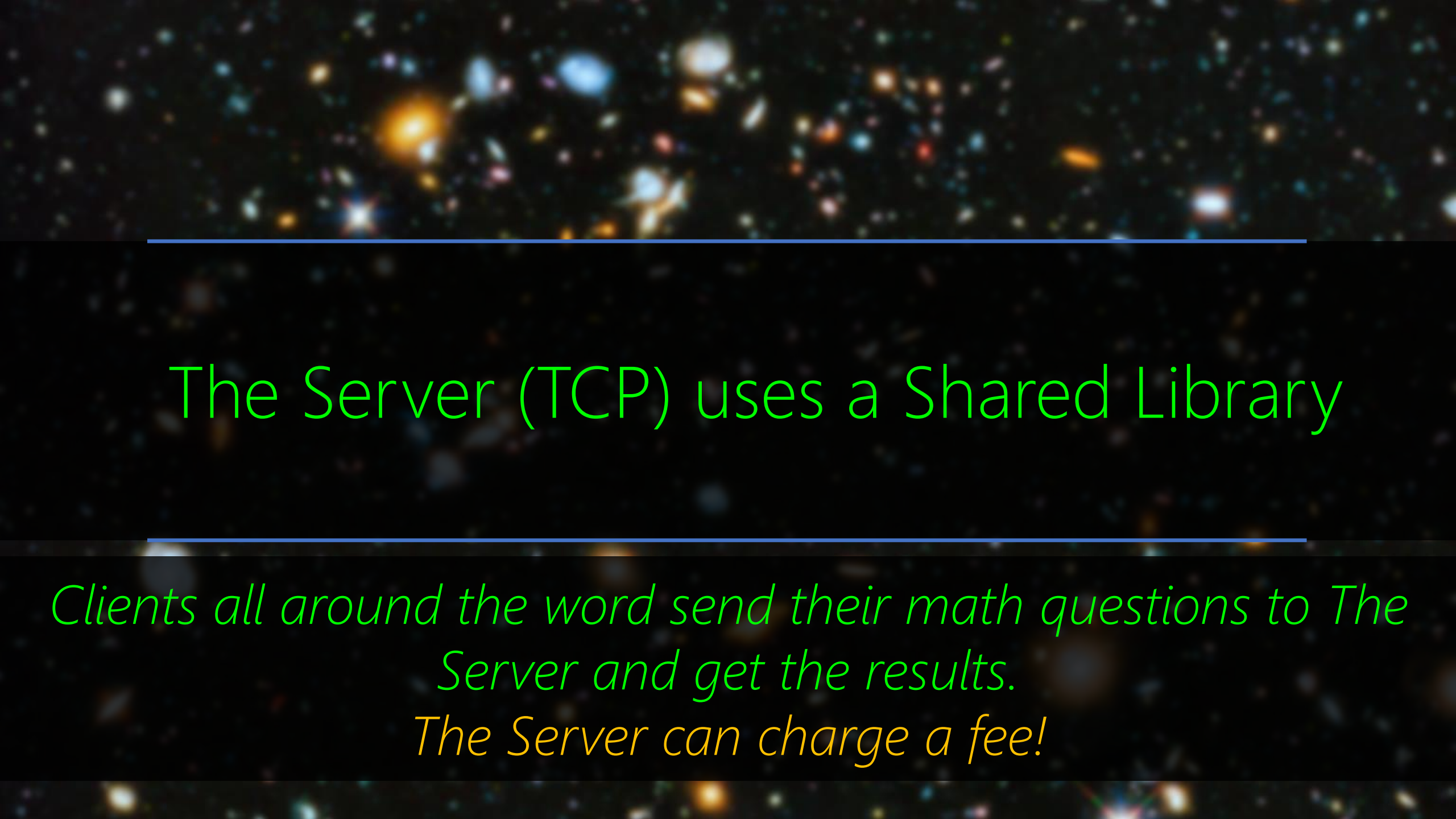
Globally Shared Libraries

All around the world, processes can use a single shared library

printf
Scanf

Is it possible to share the library with other computers?

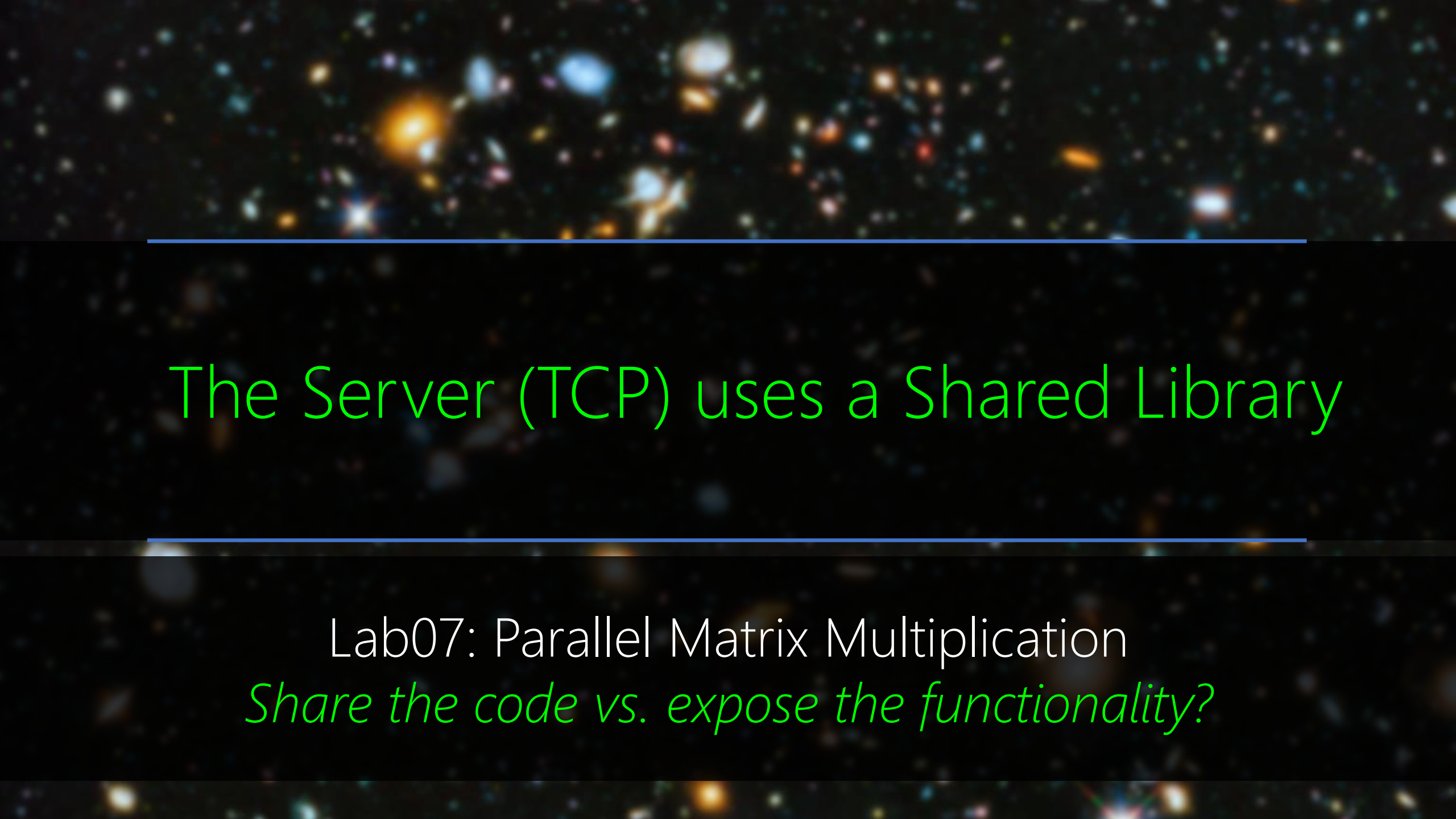
sample final exam question



The Server (TCP) uses a Shared Library

Clients all around the word send their math questions to The Server and get the results.

The Server can charge a fee!



The Server (TCP) uses a Shared Library

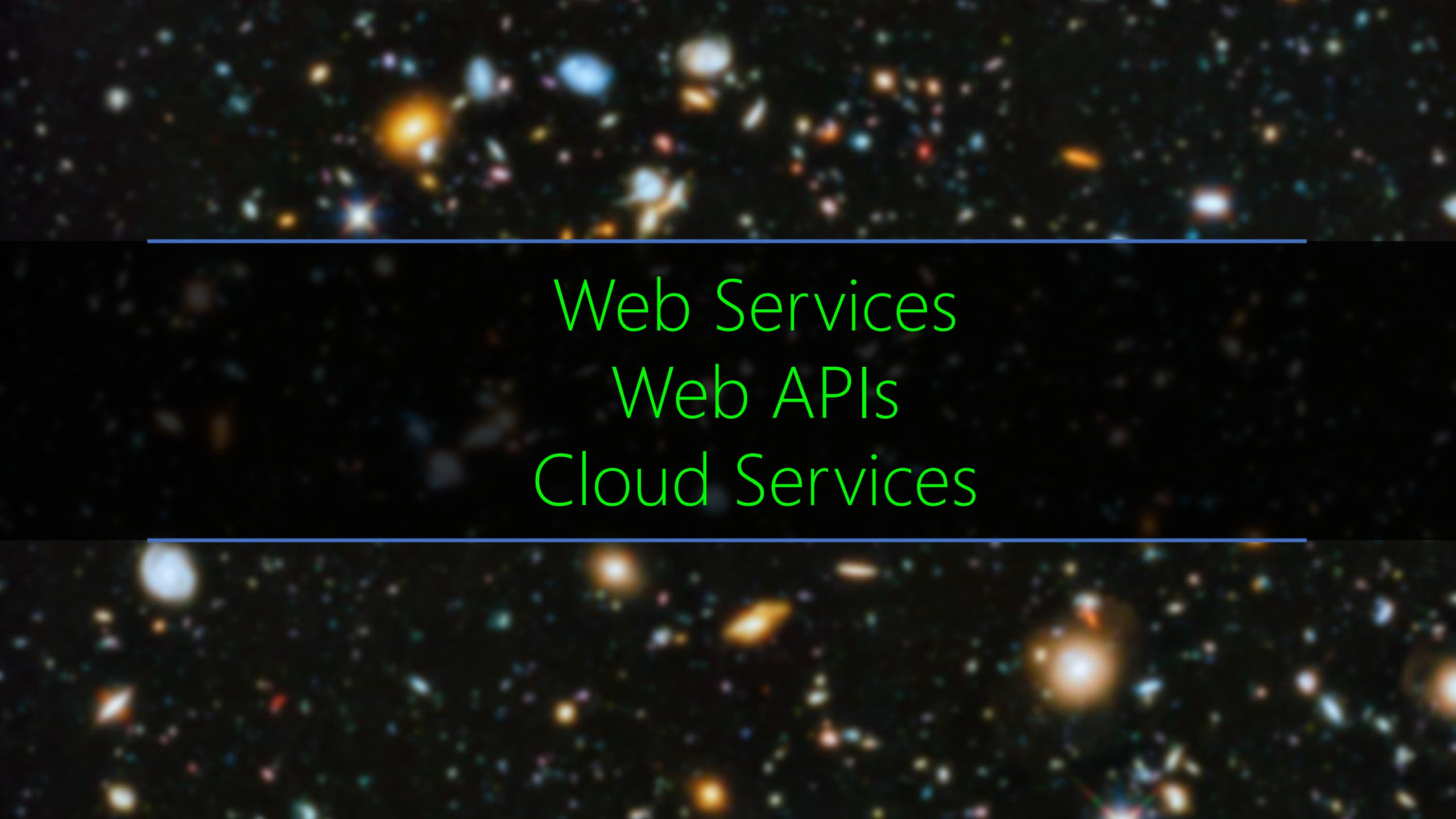
Lab07: Parallel Matrix Multiplication

Share the code vs. expose the functionality?

The Server (TCP) uses a Shared Library

1) Wrap the parallel matrix multiplication in a function X
2) Make it a dynamic library X.so
3) Write The Server include X.h and build it with X.so
4) The Server accept input matrices from clients
5) The Server call X and return the result to clients


Online C Compiler
https://www.onlinegdb.com/online_c_compiler
What is the design?


Web Services
Web APIs
Cloud Services