

Threads-Process Interaction

The image features a hand typing on a keyboard, with a digital background of glowing lines and a grid pattern. The text "Threads-Process Interaction" is overlaid in the center in a yellow font.

CONTENTS

- Threads
- Process interaction

THREADS

The background features a dark blue gradient with intricate, glowing white and light blue lines that create a sense of depth and movement, resembling a complex network or data flow. On the right side, a hand is visible, with fingers slightly curled, appearing to interact with or hold a glowing blue object, possibly a stylus or a small device. The overall aesthetic is futuristic and digital.

→ **Process image** (program,
data, stack, PCB)

→ **Allocated resources:**
(open files, main
memory, I/O devices,...)



**Address space
of a process**

→ Process properties:
distinguished address
spaces (**Unix**)

→ The operations on
processes and context
switch are rather time
consuming (**overhead**)



A blurred terminal window in the background displays a list of processes. The visible text includes headers like 'PROCESS', 'PPID', 'PID', 'PPID', 'PPID', 'PPID' and various numerical values representing process IDs and other system parameters.

PROCESS	PPID	PID	PPID	PPID	PPID
71	39K	272K	192K	343K	
19	256K	62K	88K	74K	
55	24K	24K	72K	74K	
29	2832K	18K	252K	74K	
73	956K	236K	514K	24K	
19	24K	62K	88K	74K	
55	24K	24K	72K	74K	
30	64K	222K	194K	87K	
185	424K	118	170	419K	
744	648	318	910	503K	
585	458	250	758	514K	
724	1438	180	1748	627K	
250	856K	644K	220	448K	
25	28K	272K	194K	74K	
21	48K	104K	24K	74K	
58	16K	142K	128K	82K	
28	68K	176K	272K	74K	
919	1818	458	1858	1319K	
83	184K	2276K	948K	312K	
42	124K	2112K	68K	248K	
81	1244K	104K	128K	118K	
215	1272K	942K	234K	418K	

The concept of a process
embodies two requirements:

Resource ownership

Resources can be allocated
to processes, e.g., memory,
I/O devices...

Scheduling/execution

A process has an execution state, a priority, and processes are the entities scheduled by the O.S.

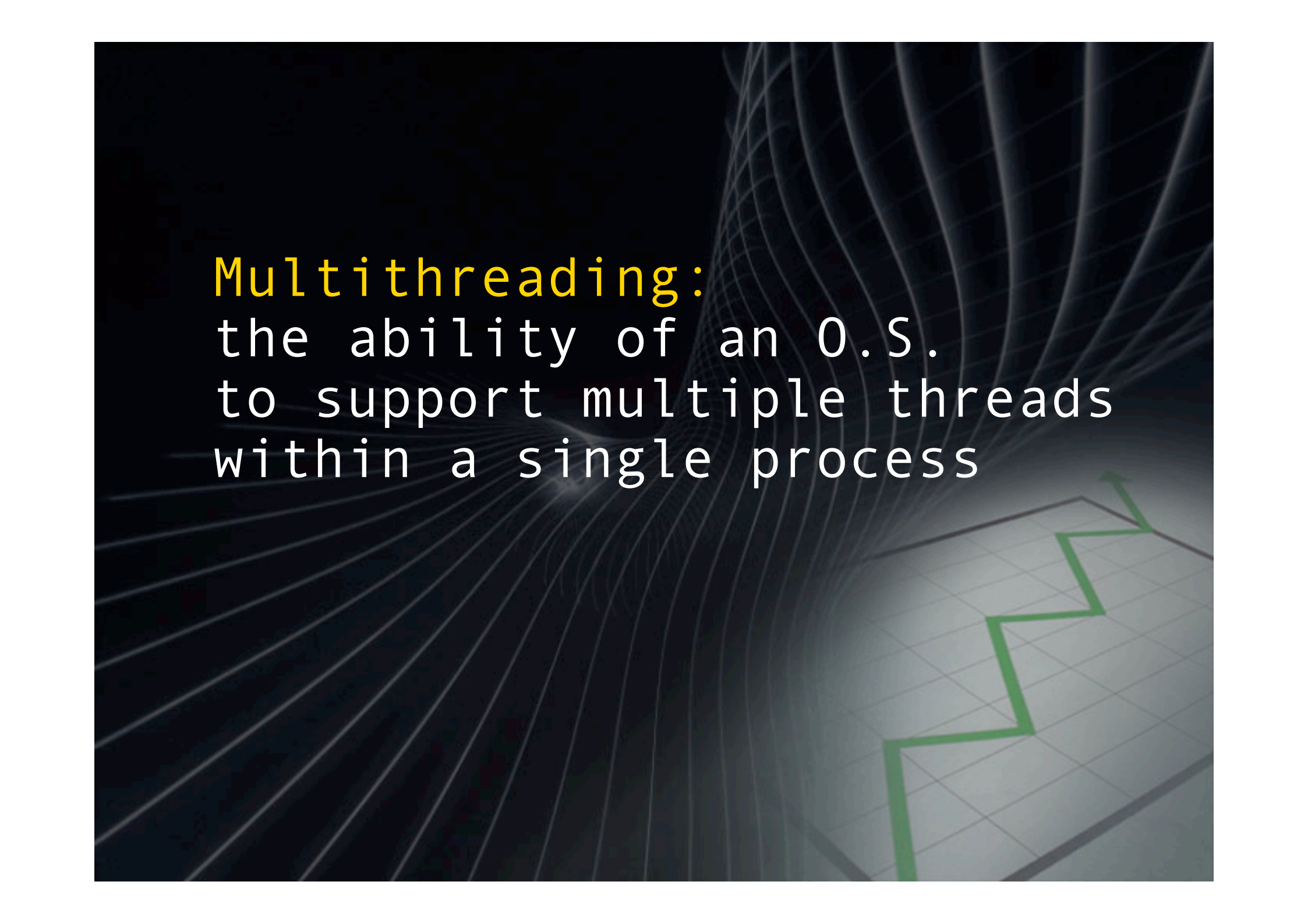


the unit of dispatching
is usually referred to as
a **thread or lightweight
process**


the unit of resource ownership is usually still referred to as a **process** or a **task**

The image shows a hand holding a pen over a notebook with a checklist. The notebook has a green header and a list of tasks with checkboxes. To the right, a computer monitor displays a similar task list with columns for 'Task', 'Due Date', and 'Status'.

Task	Due Date	Status
1. Review project plan	10/15/2023	Completed
2. Review project plan	10/15/2023	Completed
3. Review project plan	10/15/2023	Completed
4. Review project plan	10/15/2023	Completed
5. Review project plan	10/15/2023	Completed
6. Review project plan	10/15/2023	Completed
7. Review project plan	10/15/2023	Completed
8. Review project plan	10/15/2023	Completed
9. Review project plan	10/15/2023	Completed
10. Review project plan	10/15/2023	Completed
11. Review project plan	10/15/2023	Completed
12. Review project plan	10/15/2023	Completed
13. Review project plan	10/15/2023	Completed
14. Review project plan	10/15/2023	Completed
15. Review project plan	10/15/2023	Completed
16. Review project plan	10/15/2023	Completed
17. Review project plan	10/15/2023	Completed
18. Review project plan	10/15/2023	Completed
19. Review project plan	10/15/2023	Completed
20. Review project plan	10/15/2023	Completed

The background features a dark, almost black, space filled with numerous thin, light-colored, wavy lines that create a sense of depth and movement, similar to a topographical map or a data visualization. In the lower right quadrant, there is a green line graph plotted on a light gray grid. The graph shows a series of steps that generally trend upwards from left to right, ending with a small arrowhead pointing towards the top right.

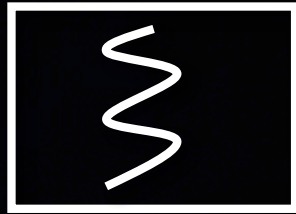
Multithreading:
the ability of an O.S.
to support multiple threads
within a single process

The background of the slide is a dark, futuristic digital tunnel. It features glowing blue and purple lines that curve and converge towards the center, creating a sense of depth and movement. The walls of the tunnel are composed of vertical columns of glowing blue and purple data streams, resembling binary code or digital information. The overall atmosphere is high-tech and abstract.

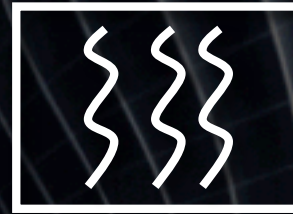
All the threads of a process share in **the same process address** space and have access to **the same data**

Any thread has several associated entities:

- A thread execution state (running, ready, ..)
- A saved thread context, when not in execution



One process
one thread



One process
multiple threads



Multiple processes
one thread
per process



Multiple processes
multiple threads
per process

THREADS BENEFITS:

It takes less time

- to create and terminate a thread than a process
- to switch between two threads within the same process

Threads enhance efficiency
in communication,
because threads within
the same process share
memory and files



THREAD IMPLEMENTATION

User-level threads

Thread management is in charge of the application and the **kernel is not aware** of thread existence

THREAD IMPLEMENTATION

User-level threads

A thread library
is typically used, which
is a **package of routines**
for thread management

The thread library
contains code for:

- creating and destroying threads
- passing messages and data between threads

The thread library
contains code for:

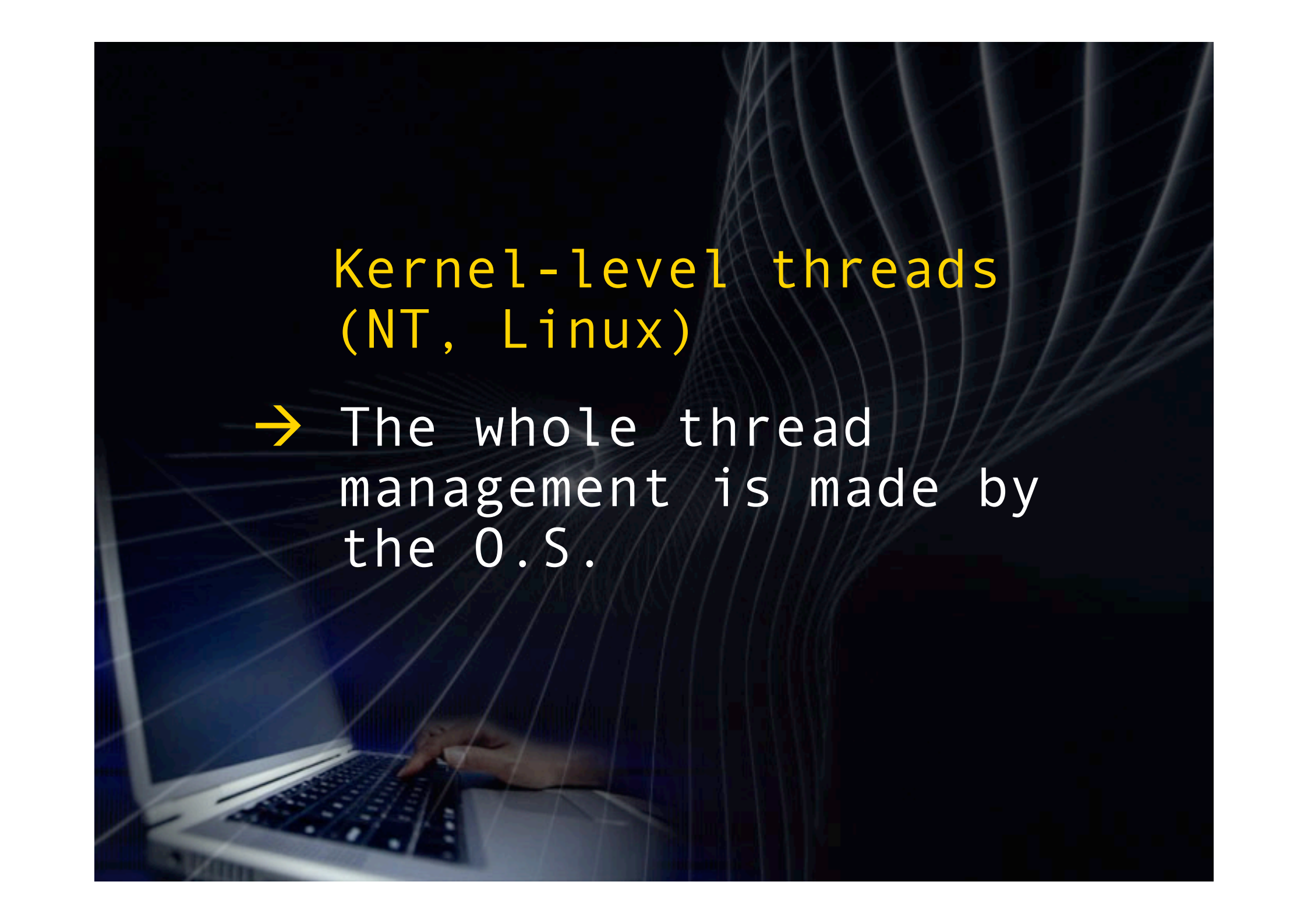
→ scheduling thread
execution and saving
and restoring thread
context

Disadvantages of user-level threads

When a thread executes
a system call, all
the threads within that
process are blocked

Disadvantages of user-level threads

A multithreaded
application cannot
take advantage
of multiprocessing



Kernel-level threads (NT, Linux)

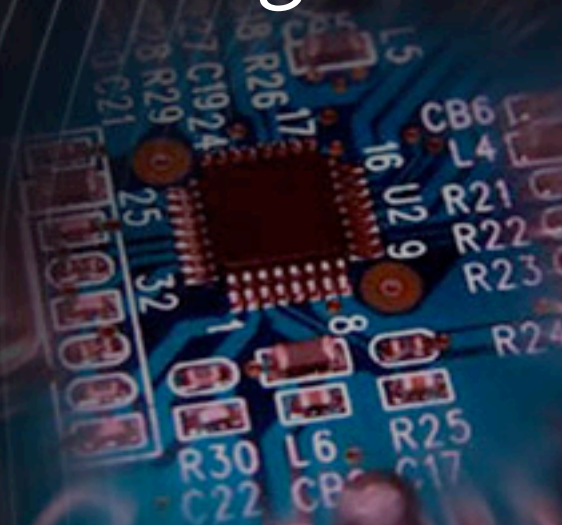
- The whole thread management is made by the O.S.

Kernel-level threads (NT, Linux)

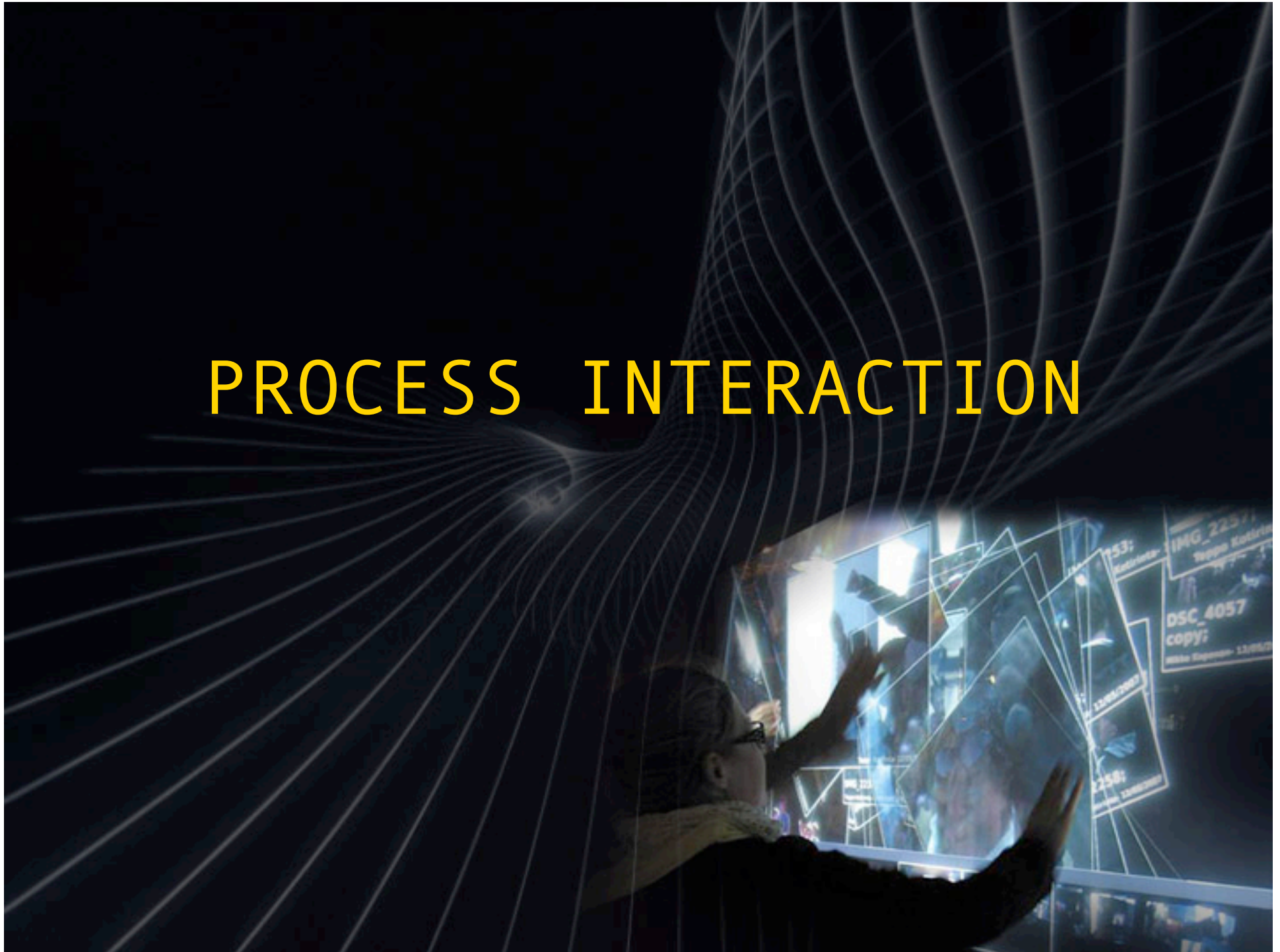
- There is no thread management code in the application area, but only a simple API to address to the kernel thread facility

The main disadvantage of this approach is that the switch between two threads within the same process requires a **mode switch to the kernel**

A multithreaded
application can take
advantage
of multiprocessing

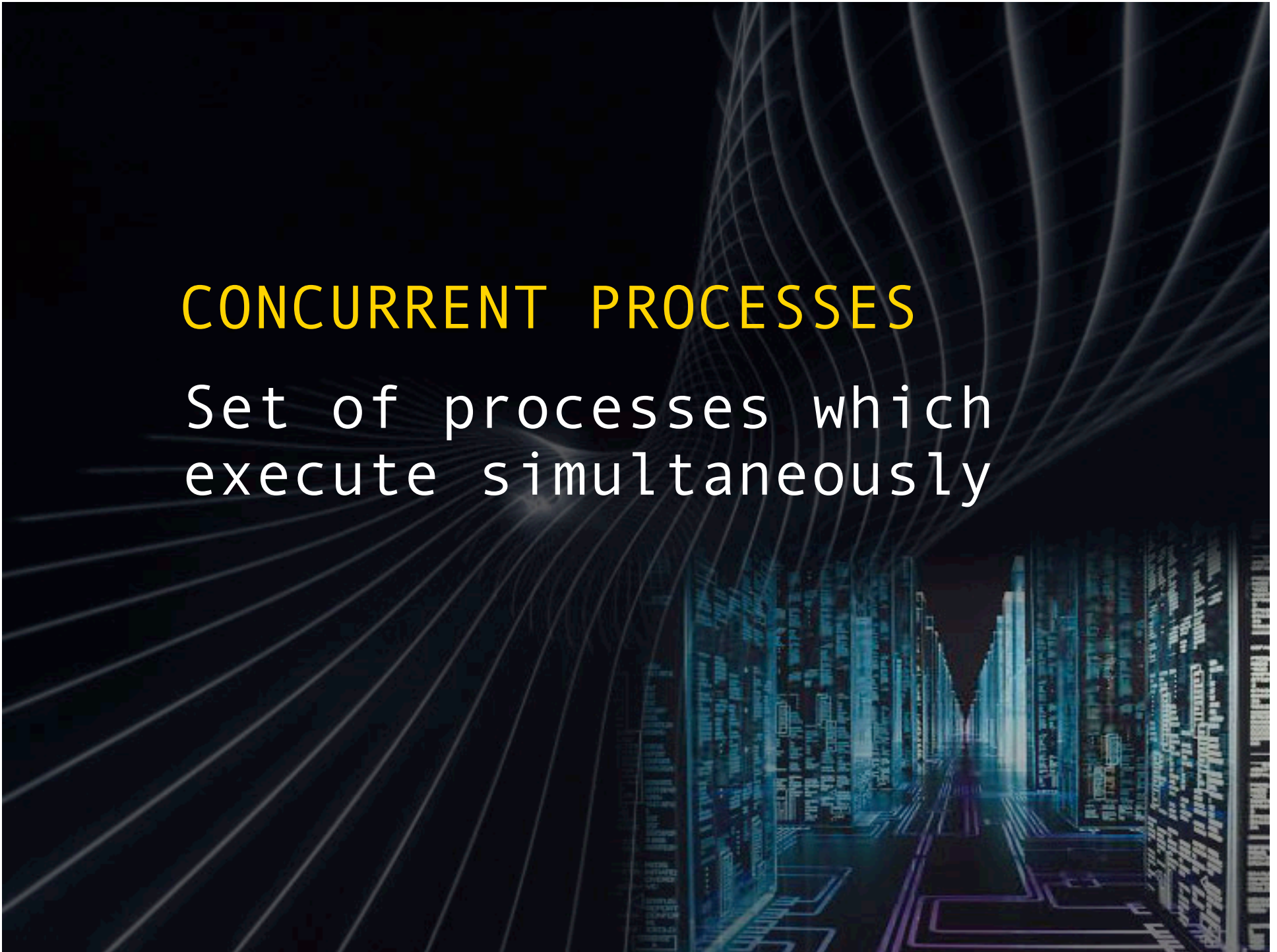


PROCESS INTERACTION



CONCURRENT PROCESSES

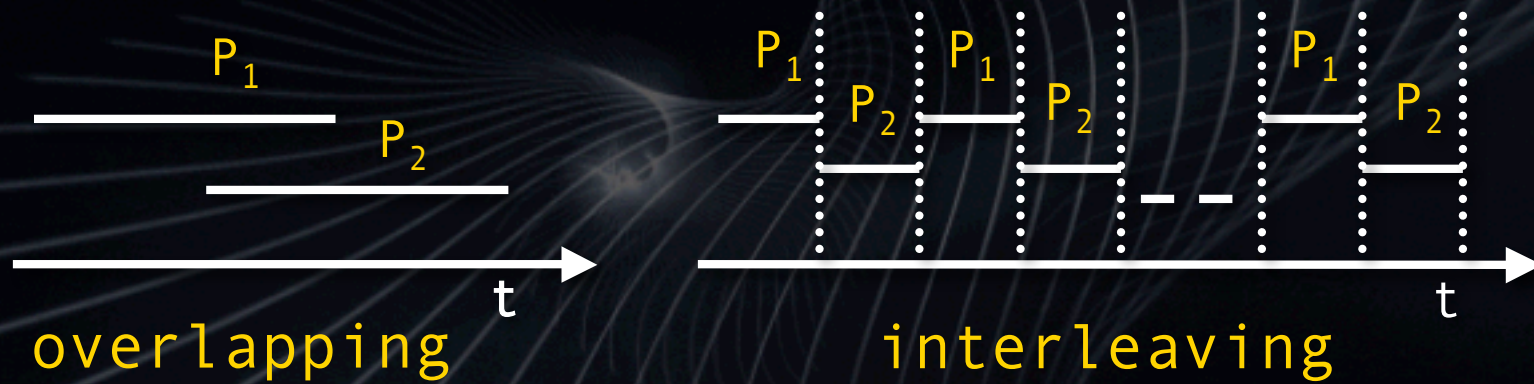
Set of processes which execute simultaneously



CONCURRENT PROCESSES

Two processes are **concurrent** if the first operation of one of them begins before the last operation of the other

CONCURRENT PROCESSES

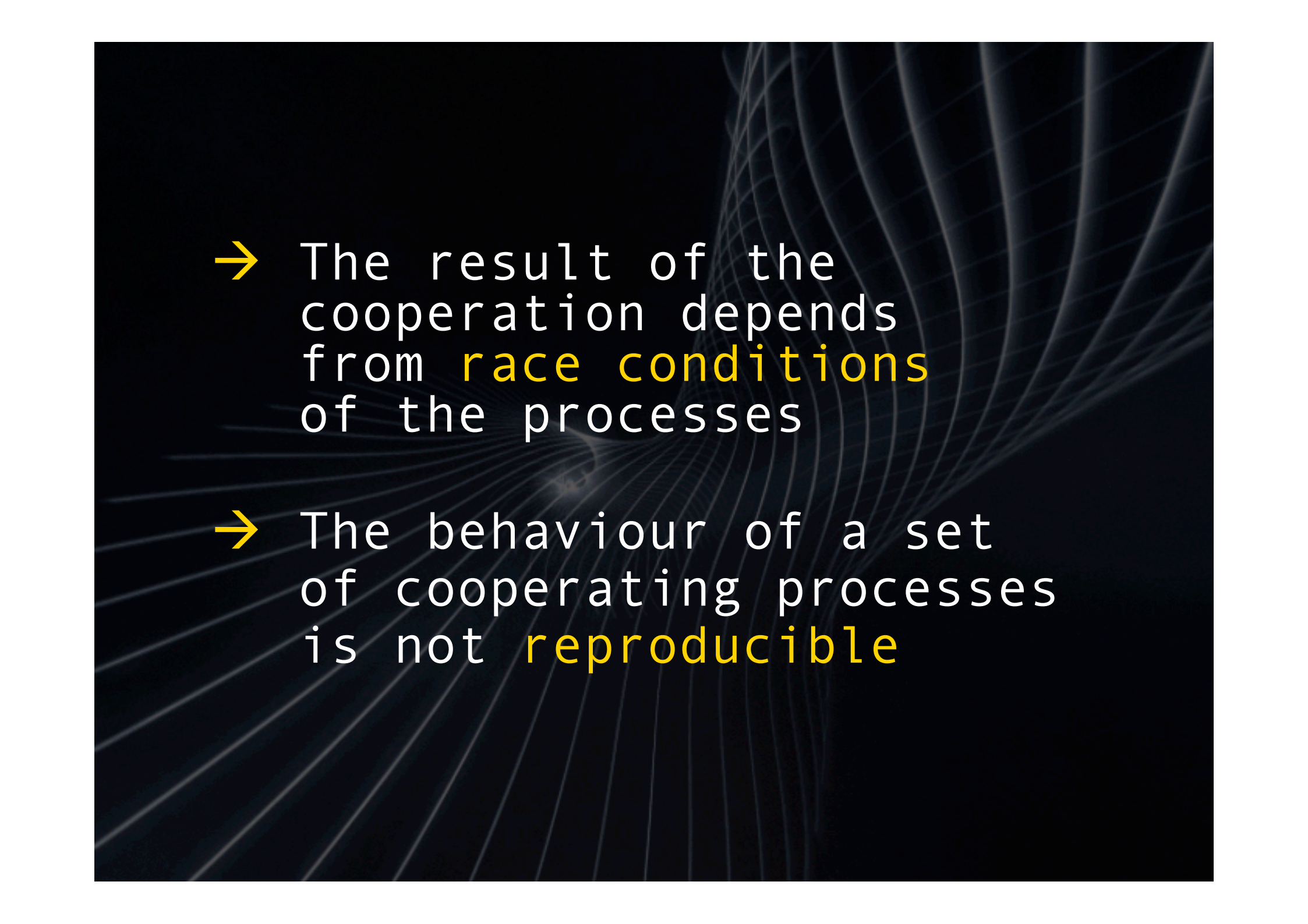


CONCURRENT PROCESSES

Independent processes:
a set of processes
is independent if each
process of the set cannot
affect or be affected
by the others processes

Cooperating processes:
a set of processes
is cooperating if each
of them can affect or be
affected by any other
process





→ The result of the cooperation depends from **race conditions** of the processes

→ The behaviour of a set of cooperating processes is not **reproducible**

PROCESS INTERACTION

Competition: behavior exhibited in the use of common resources that cannot be used simultaneously (**because of mutual exclusion**)

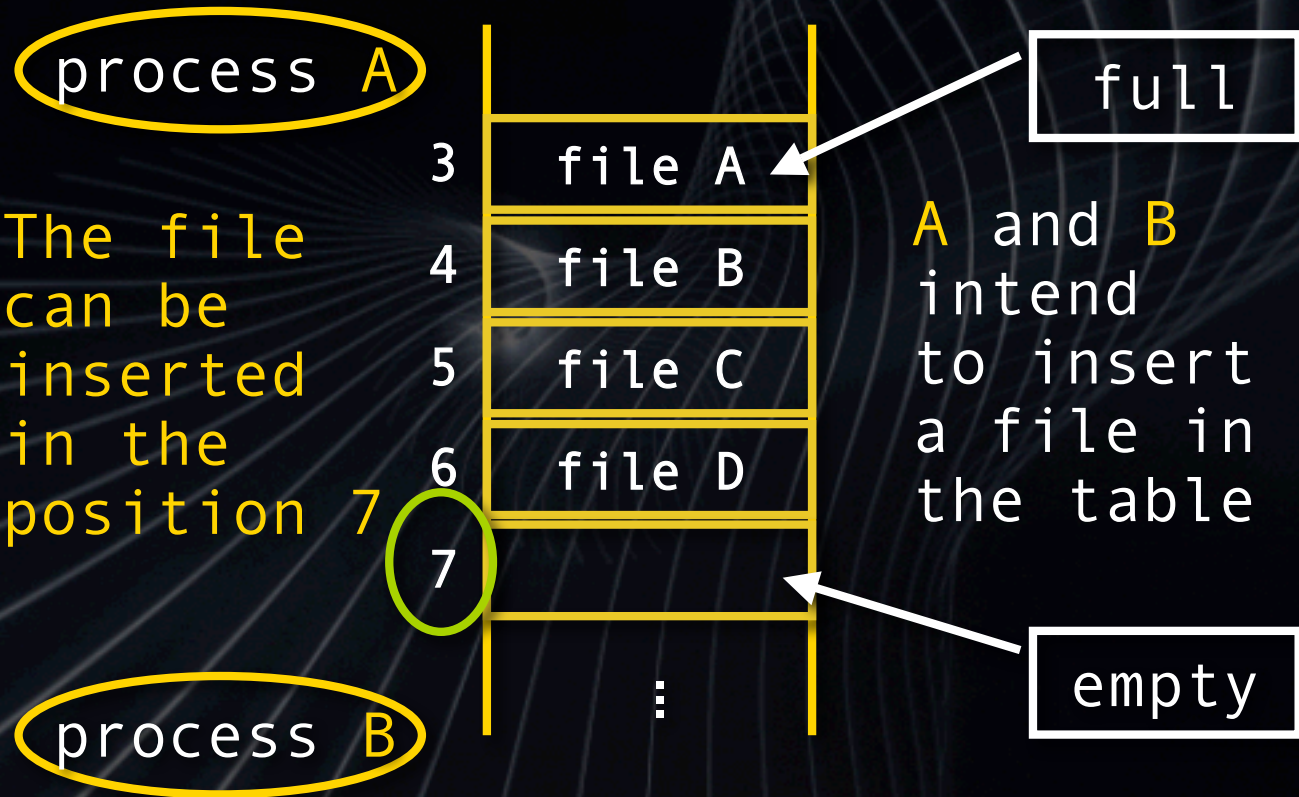
The background image shows a computer terminal window with a dark background and light-colored text. The text is mostly illegible due to blurring, but it appears to be system logs or a process monitoring tool. A table of process statistics is visible in the lower right quadrant of the terminal window. The table has several columns and rows of data, including what looks like process IDs, names, and various numerical values.

Process ID	Process Name	PPID	Mem	Res	Stk	...
1	init	0	0K	0K	0K	...
2	sm	1	0K	0K	0K	...
3	crond	1	0K	0K	0K	...
4	syslogd	1	0K	0K	0K	...
5	xinetd	1	0K	0K	0K	...
6	rsyncd	1	0K	0K	0K	...
7	sshd	1	0K	0K	0K	...
8	httpd	1	0K	0K	0K	...
9	rsyncd	1	0K	0K	0K	...
10	rsyncd	1	0K	0K	0K	...
11	rsyncd	1	0K	0K	0K	...
12	rsyncd	1	0K	0K	0K	...
13	rsyncd	1	0K	0K	0K	...
14	rsyncd	1	0K	0K	0K	...
15	rsyncd	1	0K	0K	0K	...
16	rsyncd	1	0K	0K	0K	...
17	rsyncd	1	0K	0K	0K	...
18	rsyncd	1	0K	0K	0K	...
19	rsyncd	1	0K	0K	0K	...
20	rsyncd	1	0K	0K	0K	...
21	rsyncd	1	0K	0K	0K	...
22	rsyncd	1	0K	0K	0K	...
23	rsyncd	1	0K	0K	0K	...
24	rsyncd	1	0K	0K	0K	...
25	rsyncd	1	0K	0K	0K	...
26	rsyncd	1	0K	0K	0K	...
27	rsyncd	1	0K	0K	0K	...
28	rsyncd	1	0K	0K	0K	...
29	rsyncd	1	0K	0K	0K	...
30	rsyncd	1	0K	0K	0K	...
31	rsyncd	1	0K	0K	0K	...
32	rsyncd	1	0K	0K	0K	...
33	rsyncd	1	0K	0K	0K	...
34	rsyncd	1	0K	0K	0K	...
35	rsyncd	1	0K	0K	0K	...
36	rsyncd	1	0K	0K	0K	...
37	rsyncd	1	0K	0K	0K	...
38	rsyncd	1	0K	0K	0K	...
39	rsyncd	1	0K	0K	0K	...
40	rsyncd	1	0K	0K	0K	...
41	rsyncd	1	0K	0K	0K	...
42	rsyncd	1	0K	0K	0K	...
43	rsyncd	1	0K	0K	0K	...
44	rsyncd	1	0K	0K	0K	...
45	rsyncd	1	0K	0K	0K	...
46	rsyncd	1	0K	0K	0K	...
47	rsyncd	1	0K	0K	0K	...
48	rsyncd	1	0K	0K	0K	...
49	rsyncd	1	0K	0K	0K	...
50	rsyncd	1	0K	0K	0K	...
51	rsyncd	1	0K	0K	0K	...
52	rsyncd	1	0K	0K	0K	...
53	rsyncd	1	0K	0K	0K	...
54	rsyncd	1	0K	0K	0K	...
55	rsyncd	1	0K	0K	0K	...
56	rsyncd	1	0K	0K	0K	...
57	rsyncd	1	0K	0K	0K	...
58	rsyncd	1	0K	0K	0K	...
59	rsyncd	1	0K	0K	0K	...
60	rsyncd	1	0K	0K	0K	...
61	rsyncd	1	0K	0K	0K	...
62	rsyncd	1	0K	0K	0K	...
63	rsyncd	1	0K	0K	0K	...
64	rsyncd	1	0K	0K	0K	...
65	rsyncd	1	0K	0K	0K	...
66	rsyncd	1	0K	0K	0K	...
67	rsyncd	1	0K	0K	0K	...
68	rsyncd	1	0K	0K	0K	...
69	rsyncd	1	0K	0K	0K	...
70	rsyncd	1	0K	0K	0K	...
71	rsyncd	1	0K	0K	0K	...
72	rsyncd	1	0K	0K	0K	...
73	rsyncd	1	0K	0K	0K	...
74	rsyncd	1	0K	0K	0K	...
75	rsyncd	1	0K	0K	0K	...
76	rsyncd	1	0K	0K	0K	...
77	rsyncd	1	0K	0K	0K	...
78	rsyncd	1	0K	0K	0K	...
79	rsyncd	1	0K	0K	0K	...
80	rsyncd	1	0K	0K	0K	...
81	rsyncd	1	0K	0K	0K	...
82	rsyncd	1	0K	0K	0K	...
83	rsyncd	1	0K	0K	0K	...
84	rsyncd	1	0K	0K	0K	...
85	rsyncd	1	0K	0K	0K	...
86	rsyncd	1	0K	0K	0K	...
87	rsyncd	1	0K	0K	0K	...
88	rsyncd	1	0K	0K	0K	...
89	rsyncd	1	0K	0K	0K	...
90	rsyncd	1	0K	0K	0K	...
91	rsyncd	1	0K	0K	0K	...
92	rsyncd	1	0K	0K	0K	...
93	rsyncd	1	0K	0K	0K	...
94	rsyncd	1	0K	0K	0K	...
95	rsyncd	1	0K	0K	0K	...
96	rsyncd	1	0K	0K	0K	...
97	rsyncd	1	0K	0K	0K	...
98	rsyncd	1	0K	0K	0K	...
99	rsyncd	1	0K	0K	0K	...
100	rsyncd	1	0K	0K	0K	...

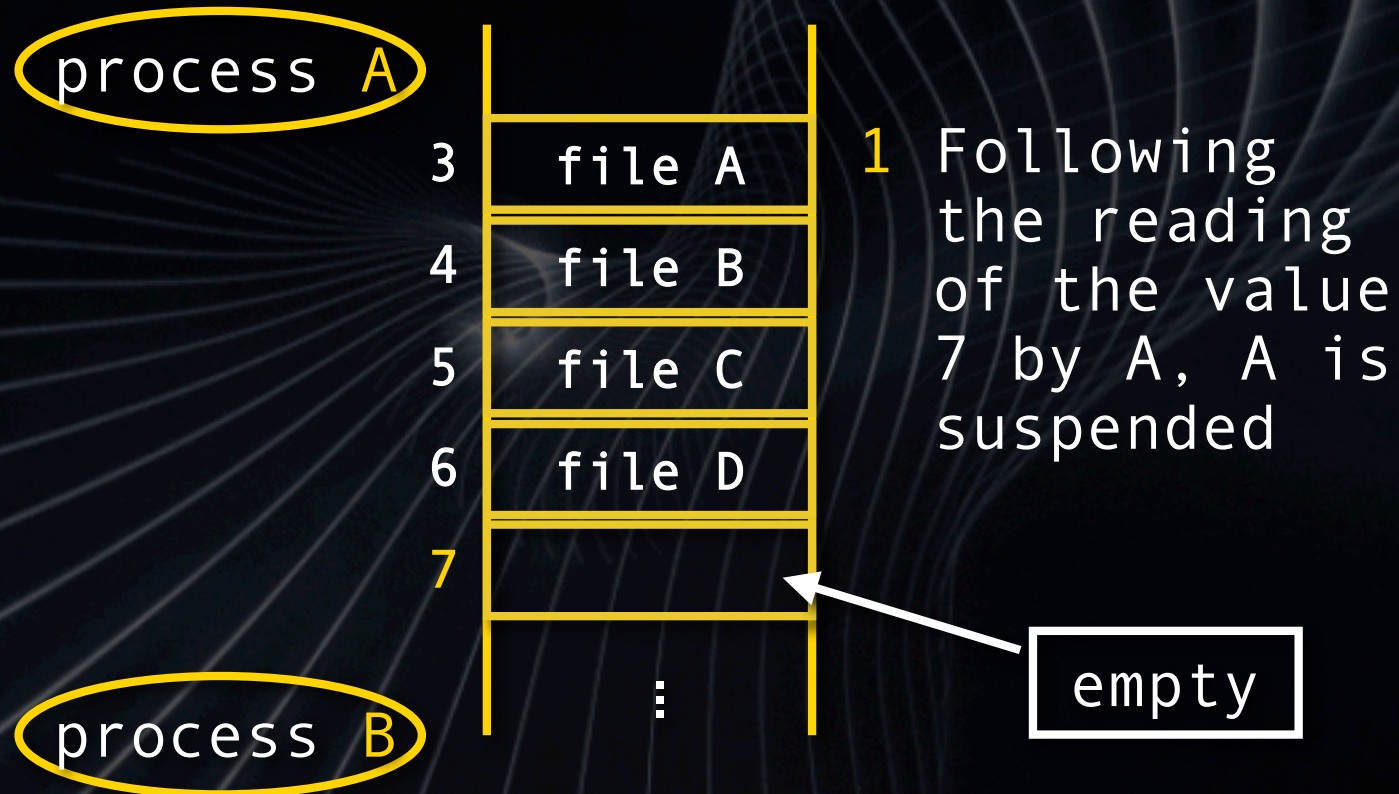
PROCESS INTERACTION

Cooperation: execution
of a common activity
obtained by the exchange
of information
(via communication)

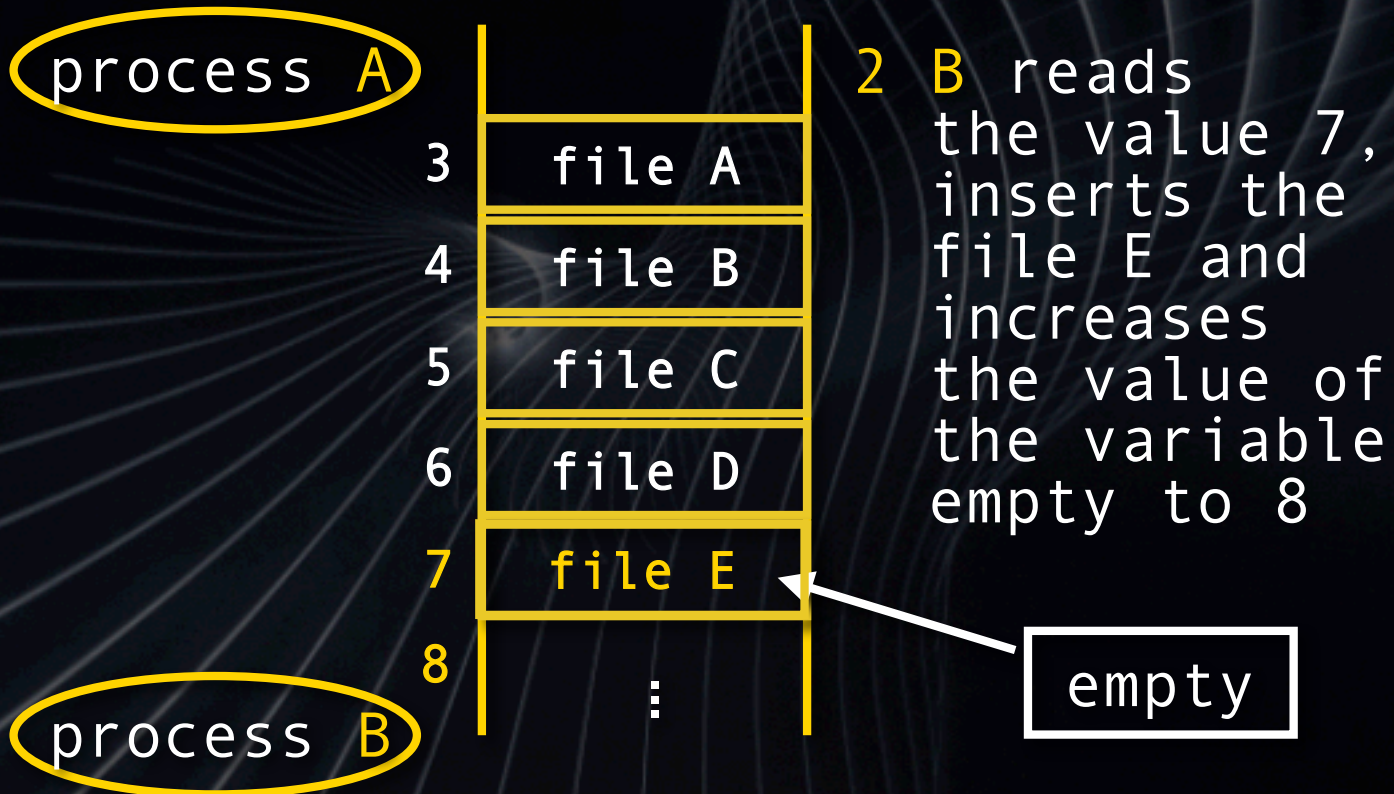
MUTUAL EXCLUSION



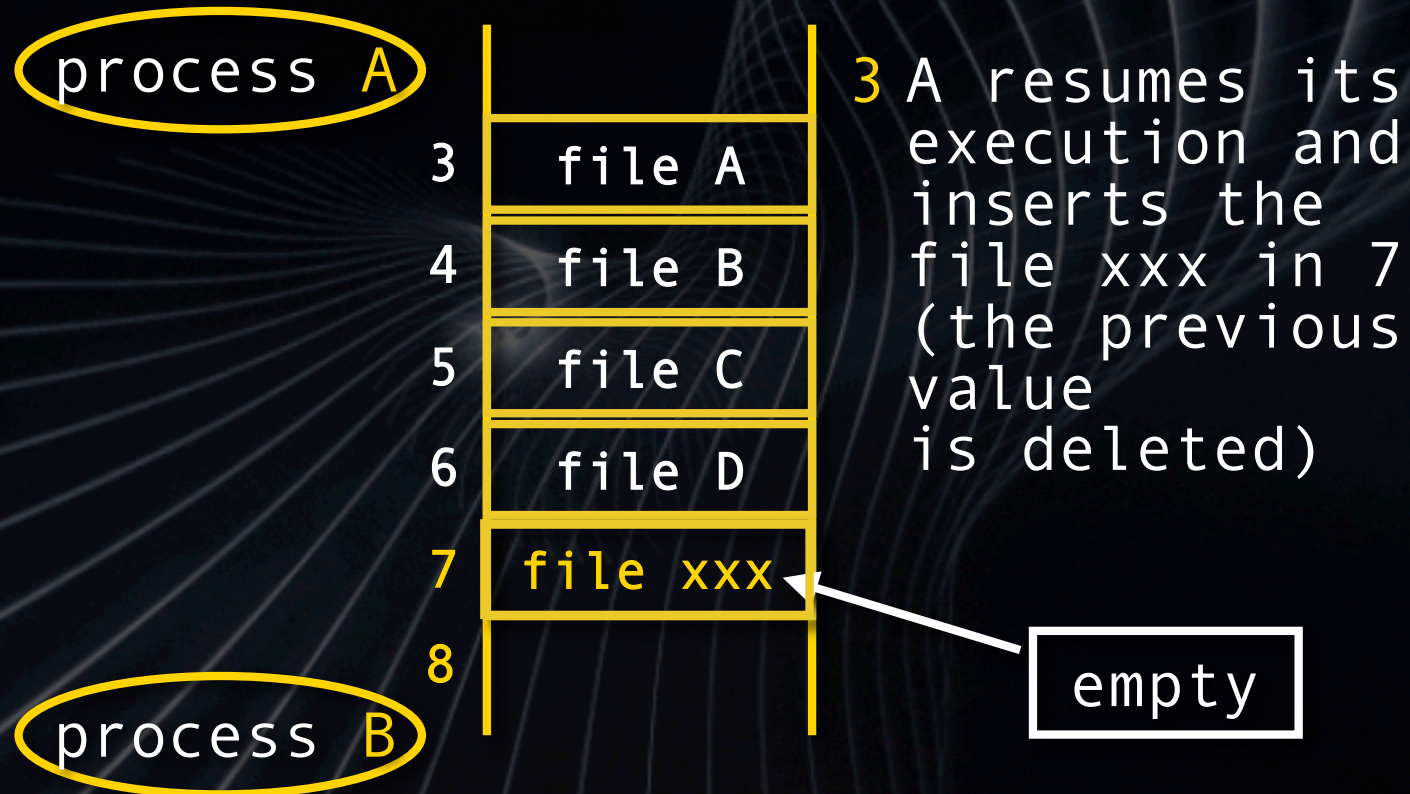
MUTUAL EXCLUSION



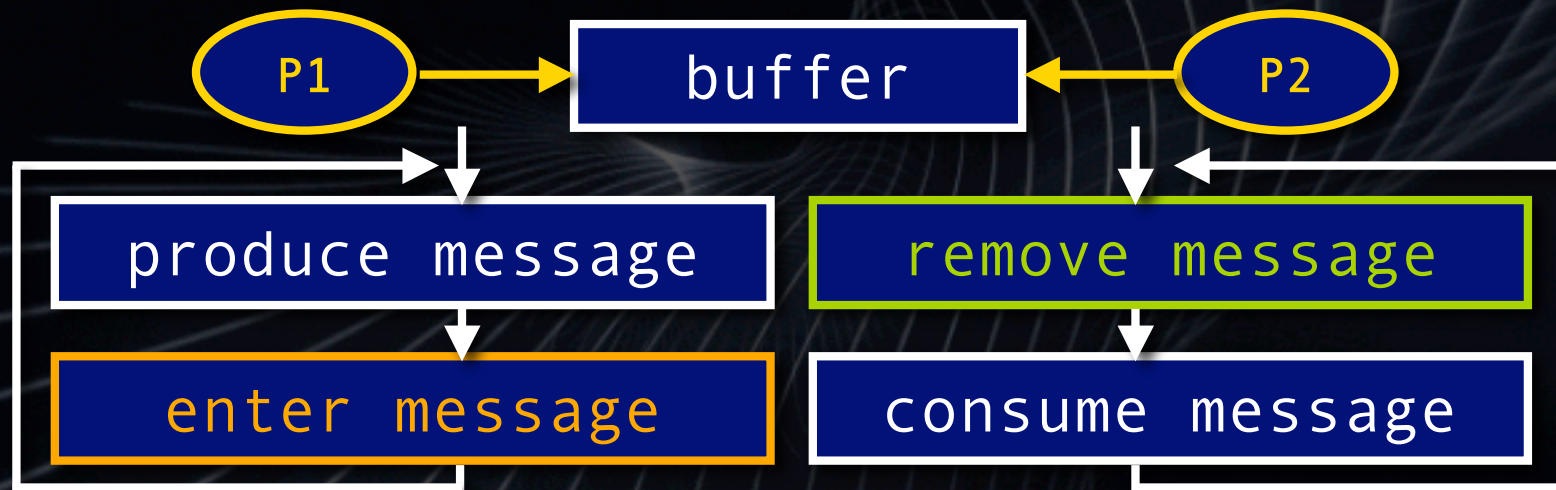
MUTUAL EXCLUSION



MUTUAL EXCLUSION



COMMUNICATION

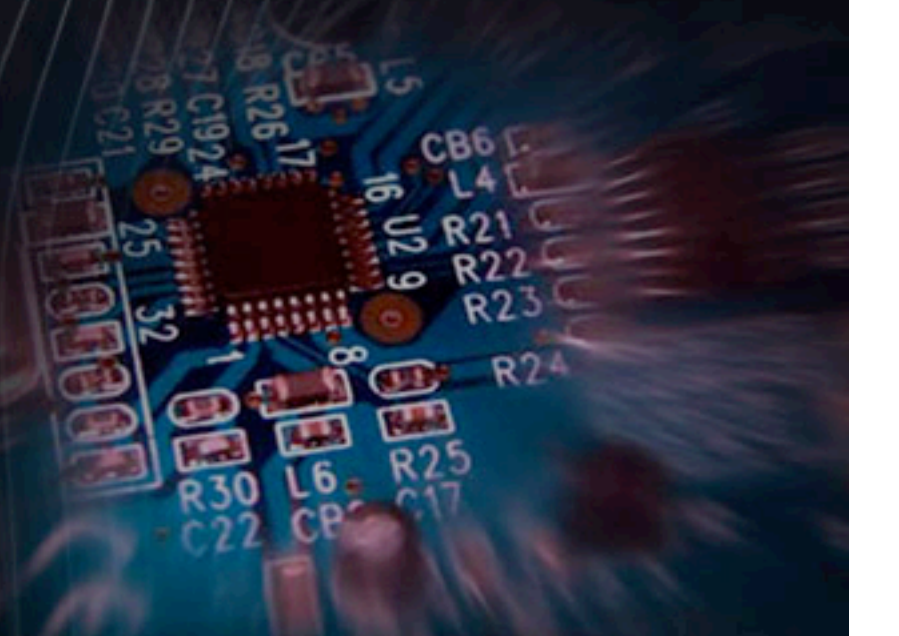


COMMUNICATION

- Correct sequence of operations:
enter - remove - enter -
remove - ...
- Incorrect sequence of operations: remove -
remove - enter - ...

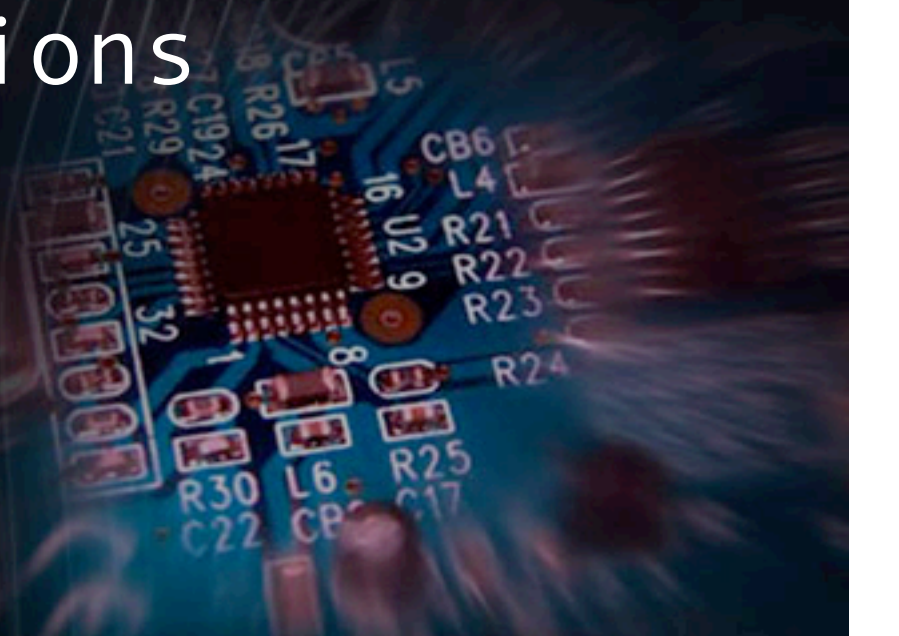
SYNCHRONIZATION

In the previous examples,
in order to obtain a
correct system behaviour,
...



SYNCHRONIZATION

... it is necessary to impose **timing constraints** to the execution of the process operations



SYNCHRONIZATION constraints

Competition: only one
process at a time
must access to a common
resource (**indirect or
implicit synchronization**)

SYNCHRONIZATION constraints

Cooperation: the order of operations observed by producers and consumers on the buffer must follow a fixed policy, ...



SYNCHRONIZATION constraints

... such as an alternation
schedule (direct or explicit
synchronization)

PROCESS INTERACTION MODELS

→ global environment
model

→ message passing model



GLOBAL ENVIRONMENT MODEL

The process system may
be considered as a set
of **processes** and **resources**
(objects)

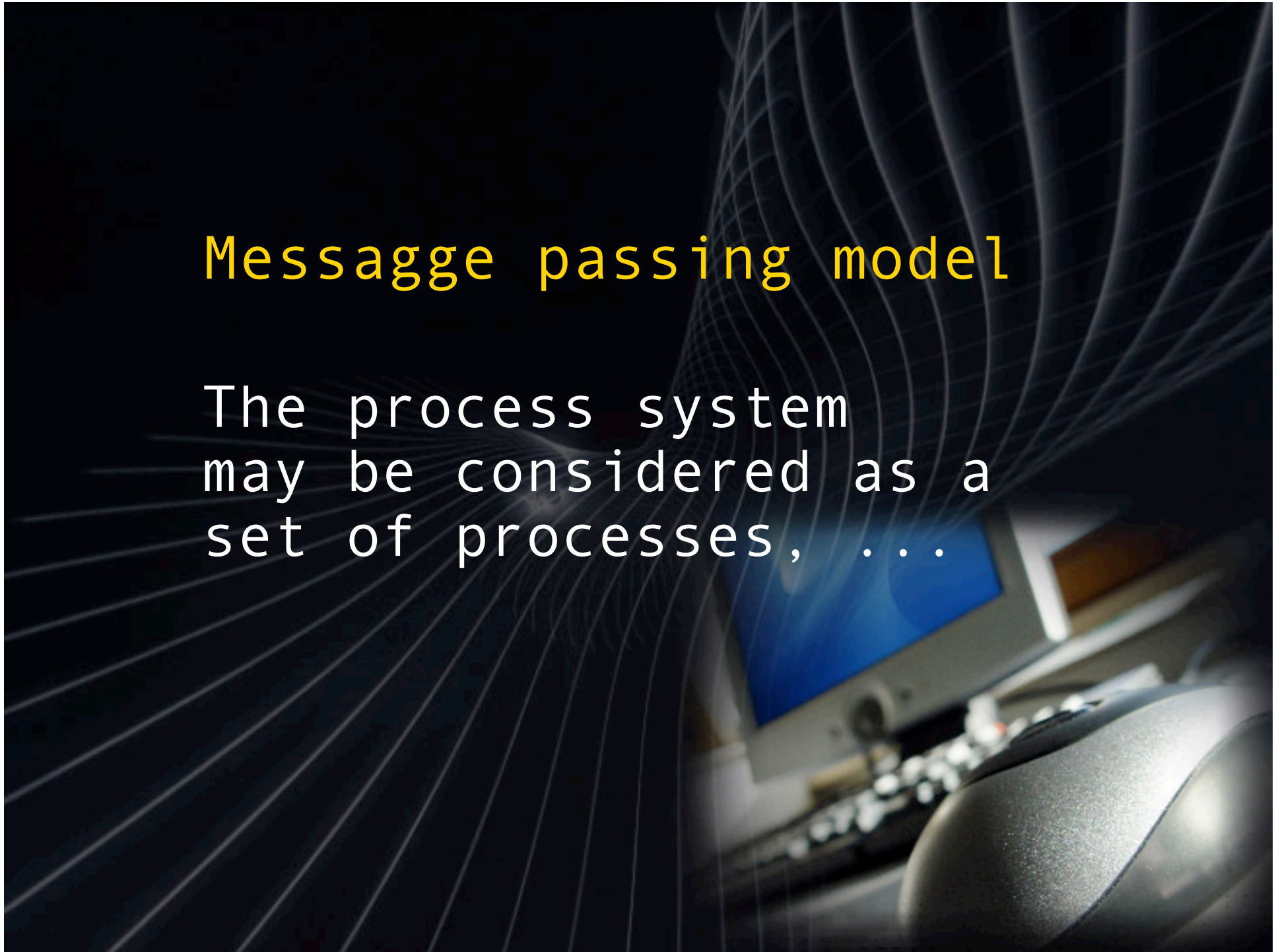
GLOBAL ENVIRONMENT MODEL



01, 04 private res. competition
02, 03 common res. cooperation

Message passing model

The process system
may be considered as a
set of processes, ...



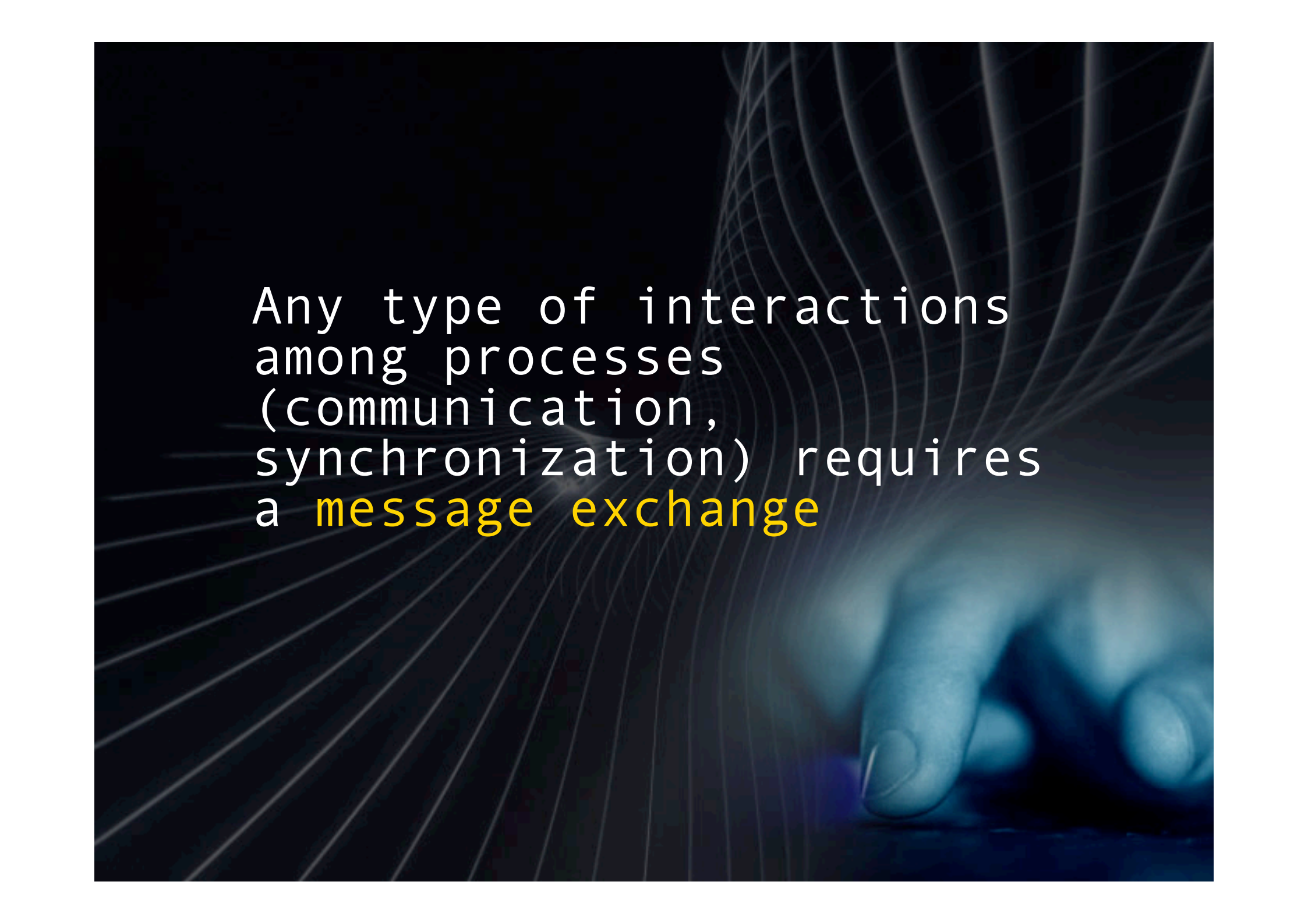
Message passing model

... each of them working
in a **local environment**,
i.e., not directly
accessible by other
processes



Message passing model



A hand is shown typing on a keyboard, with a blue glow emanating from the keys. The background is dark with abstract white lines that resemble a network or data flow. The text is overlaid on the left side of the image.

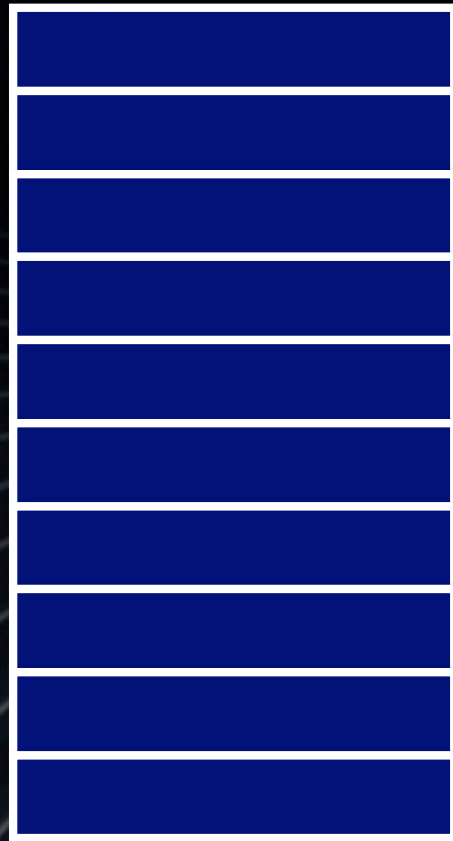
Any type of interactions
among processes
(communication,
synchronization) requires
a **message exchange**

Resources are typically
not directly accessible
to processes

Any resource is associated
to a specific **server process**



BUFFER



P

producer

C

consumer

→ The producer cannot enter a message in the buffer, if the buffer is full

→ The consumer cannot remove a message from the buffer, if the buffer is empty

d = num. of entered messages
e = num. of removed messages
N = size of the buffer

$$0 \leq d - e \leq N$$