

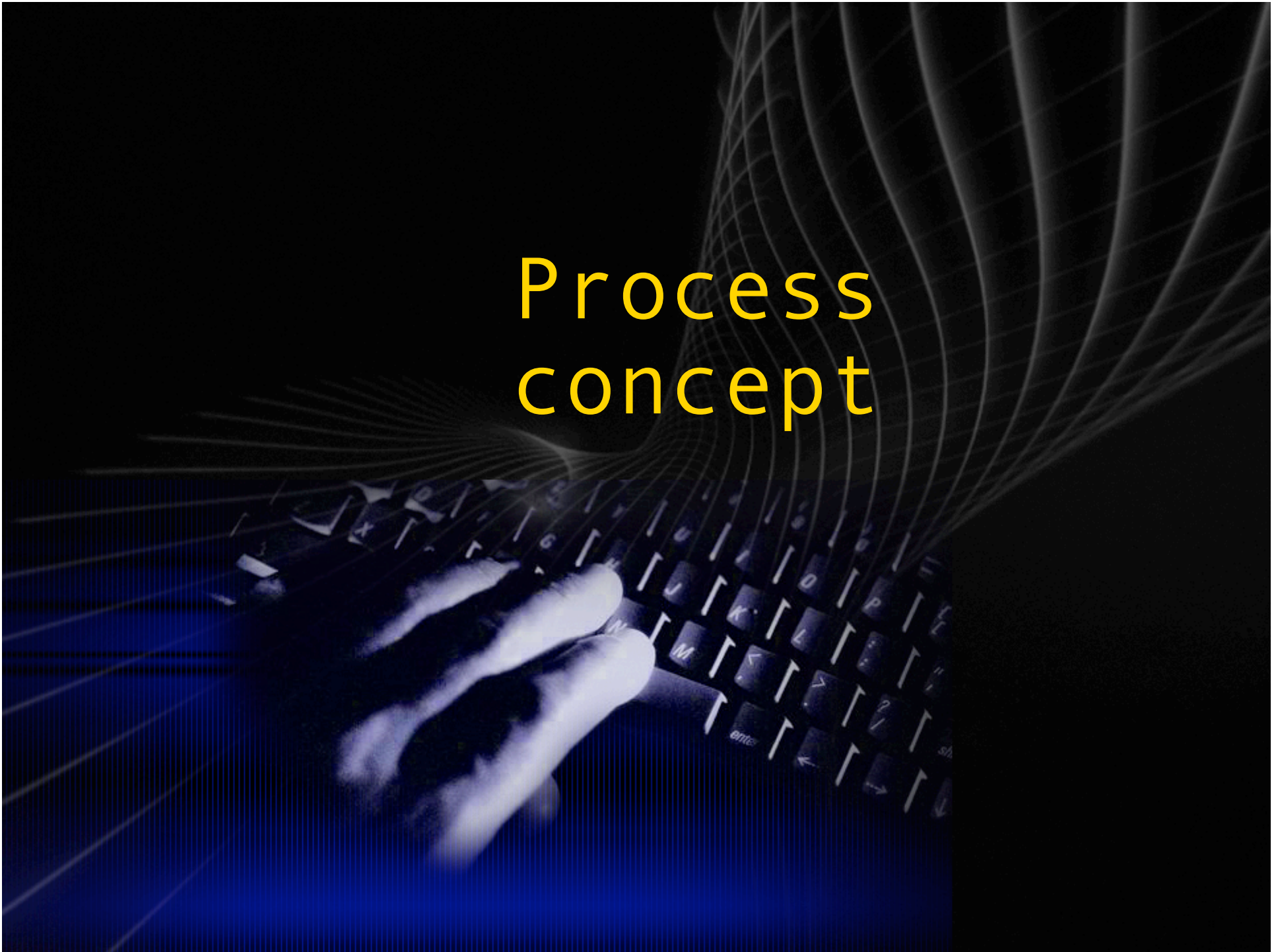
# PROCESS MANAGEMENT

The image features a hand typing on a keyboard, with a dark, abstract background of glowing digital lines and curves. The text 'PROCESS MANAGEMENT' is centered in a bold, yellow, sans-serif font.

# Contents

- Process concept
- Process states
- Process description

# Process concept



## Process definition

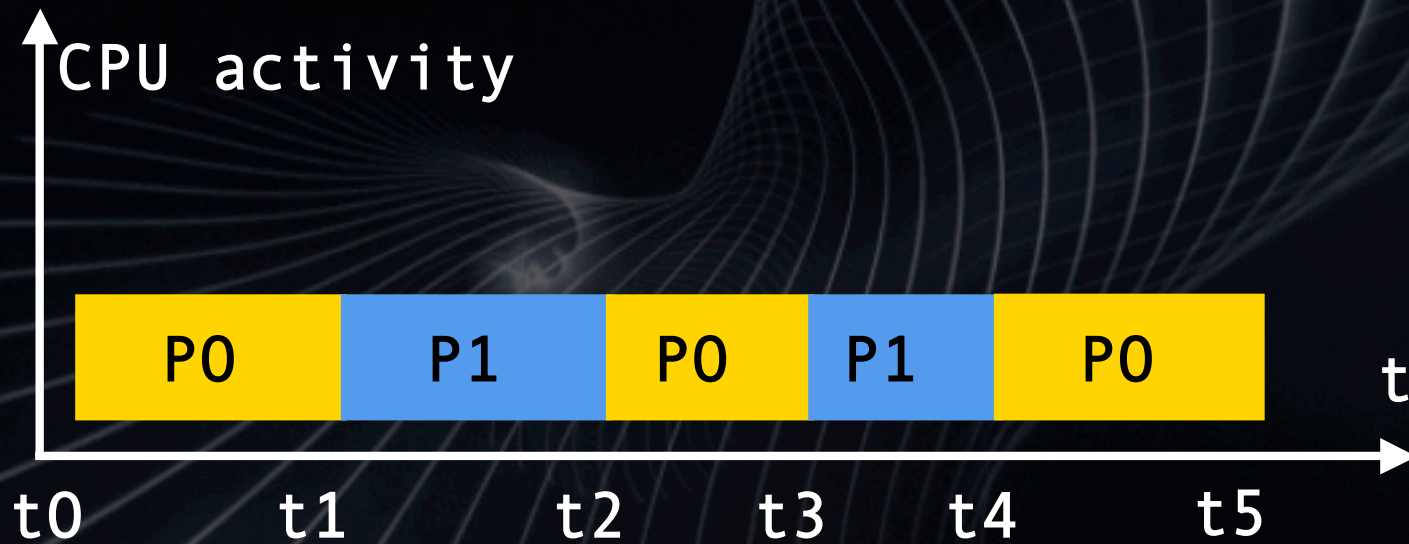
Informally, a process is  
a **program in execution**

All the programs in  
execution are organized  
as a set of **sequential  
processes (process model)**

A process is the **unit of work**  
in a multiprogrammed system

→ a multiprogrammed O.S.  
allows concurrent execution  
of more processes

# Multiprogramming



P0 process :  $t_1 - t_0$ ,  $t_3 - t_2$ ,  $t_5 - t_4$

→ **Algorithm**: the logical sequence of operations that must be executed to solve a problem

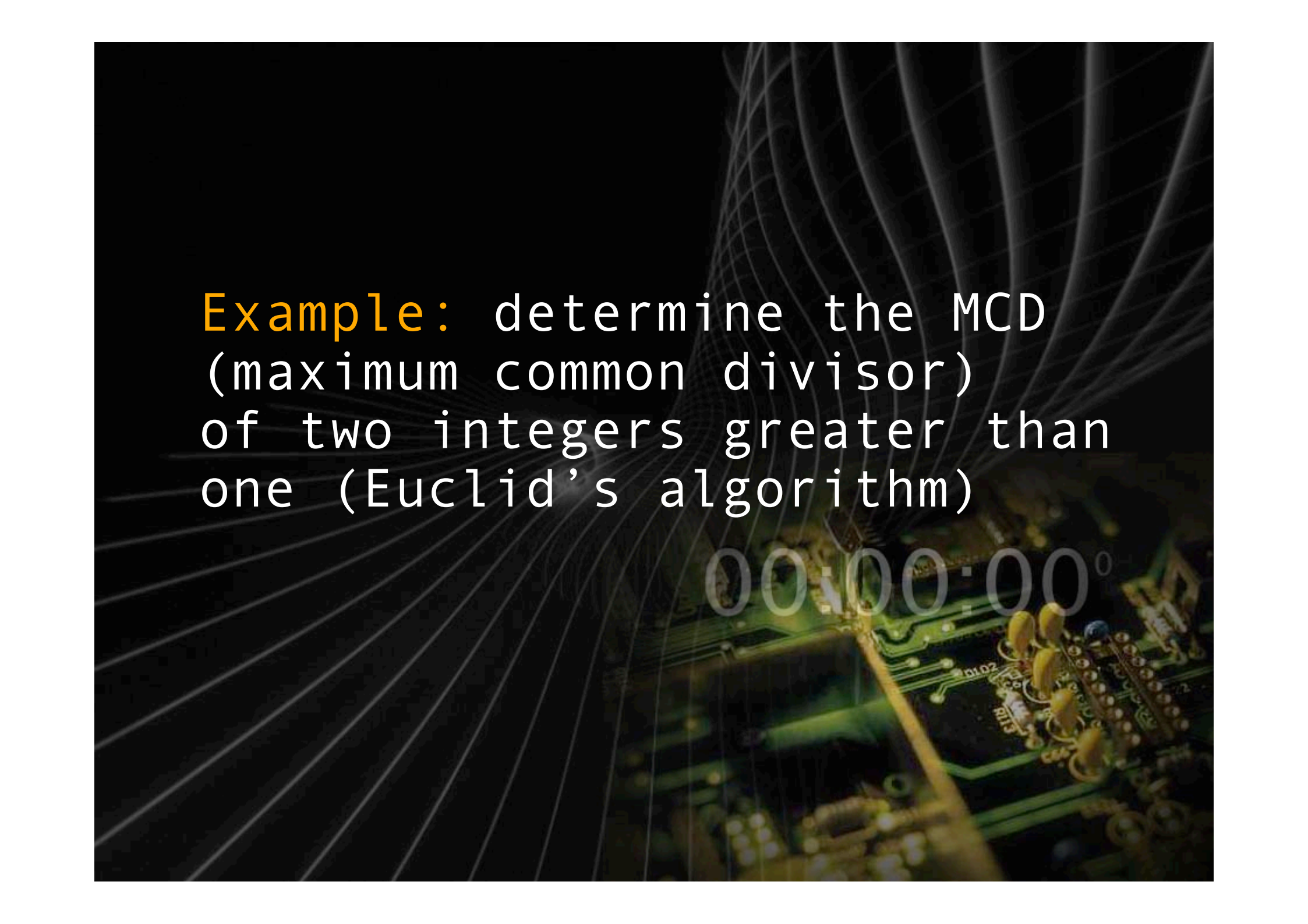
→ **Program**: algorithm description by a programming language



Process: the sequence of computer operations during the execution of the program

00:00:00<sup>0</sup>





**Example:** determine the MCD  
(maximum common divisor)  
of two integers greater than  
one (Euclid's algorithm)

00:00:00°

```
{  
  a = x; b = y;  
  while (a != b)  
    if (a > b) a = a - b;  
    else b = b - a;  
}
```

Subtract the smaller number  
from the larger one; repeat  
until you get a zero or a one

initial  
state

final  
state

x	18	18	18	18	18	18
y	24	24	24	24	24	24
a	-	18	18	18	12	6
b	-	-	24	6	6	6

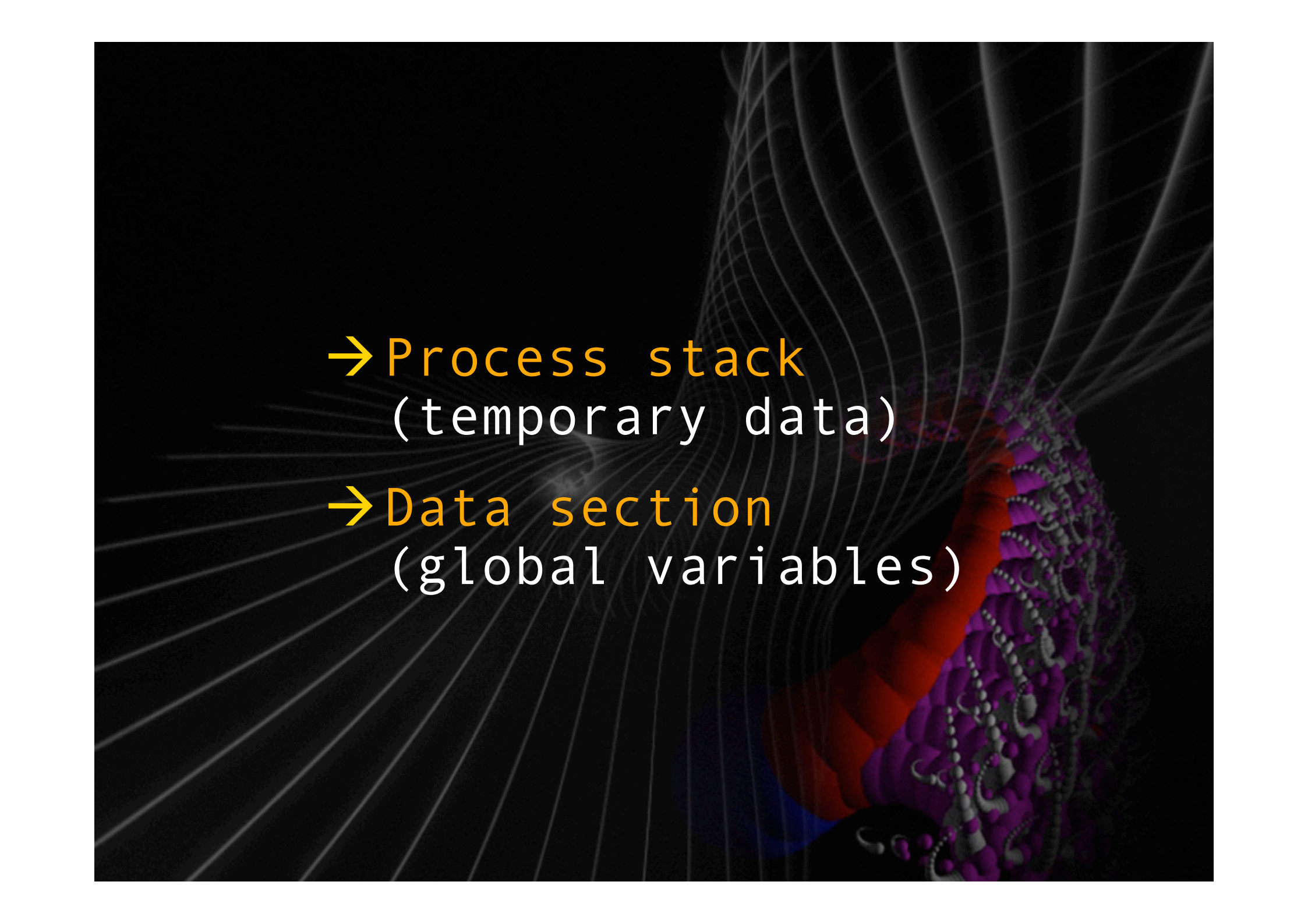
## Process instances

More processes (**instances**) can be associated to the same program; each of them represents the execution of the same code with different input data

A process is identified by:

→ Program code  
(text section)

→ Its current activity,  
represented by the value  
of the PC and of the  
processor's registers

- 
- The background features a dark space filled with numerous thin, white, curved lines that create a sense of depth and movement. On the right side, there is a complex, colorful structure resembling a molecular model or a data visualization, with shades of red, purple, and blue. The text is overlaid on the left side of the image.
- Process stack  
(temporary data)
  - Data section  
(global variables)

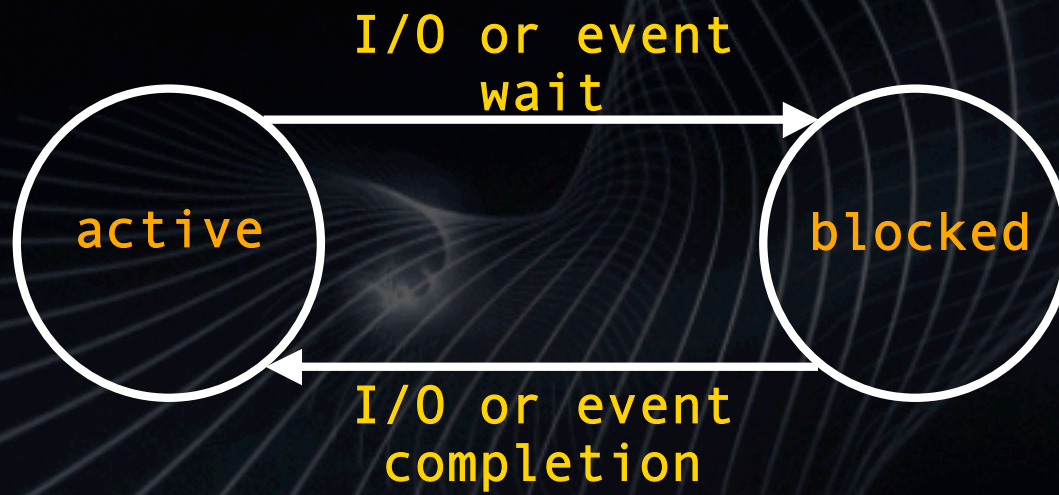
Several resources can be associated to one process:

- Memory
- Open Files
- I/O devices

# Process states

The image features a hand typing on a keyboard, with a dark, abstract background of glowing digital lines and curves. The text 'Process states' is centered in a yellow, sans-serif font.





The number of CPU is generally less than the number of available processes

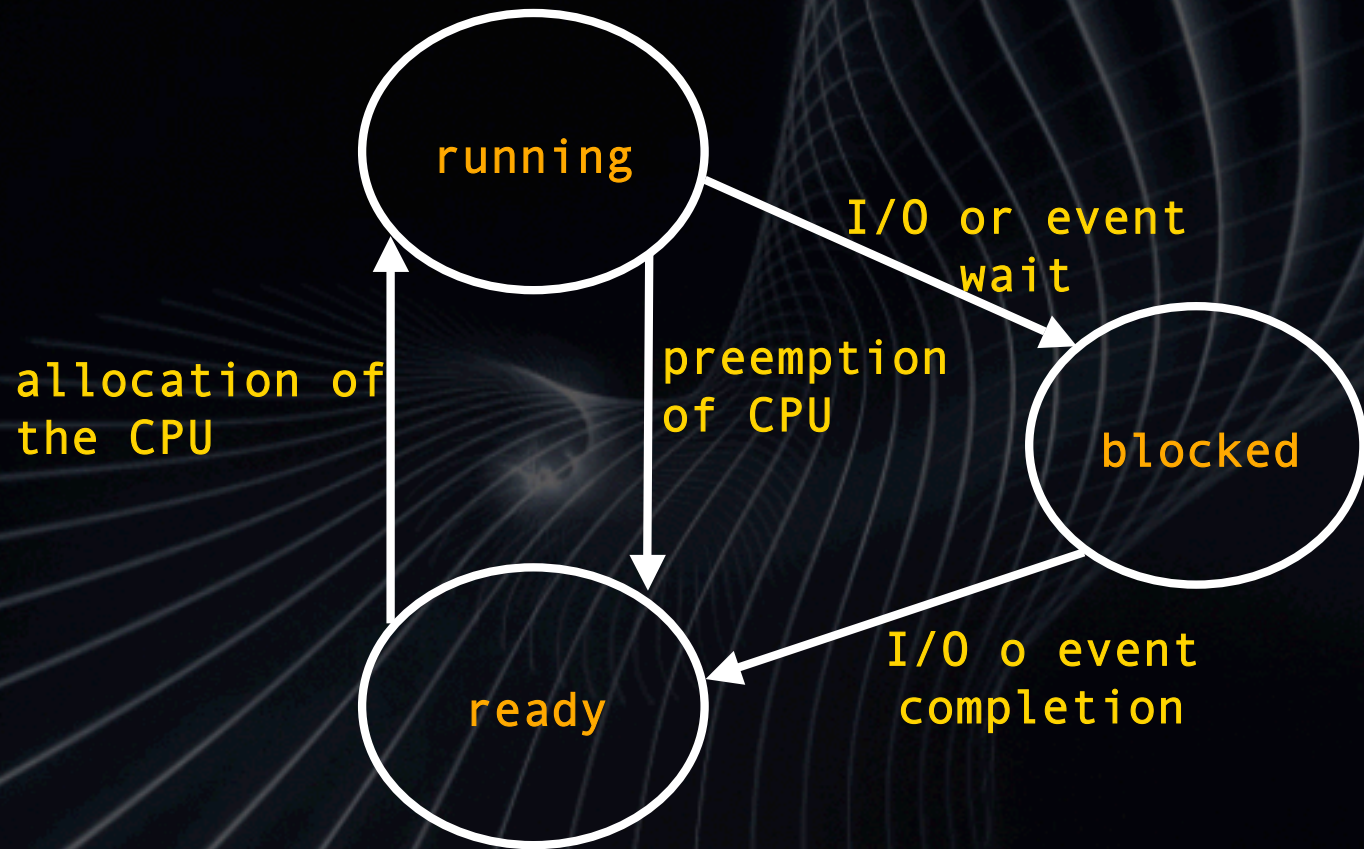
The active state is subdivided in two states:

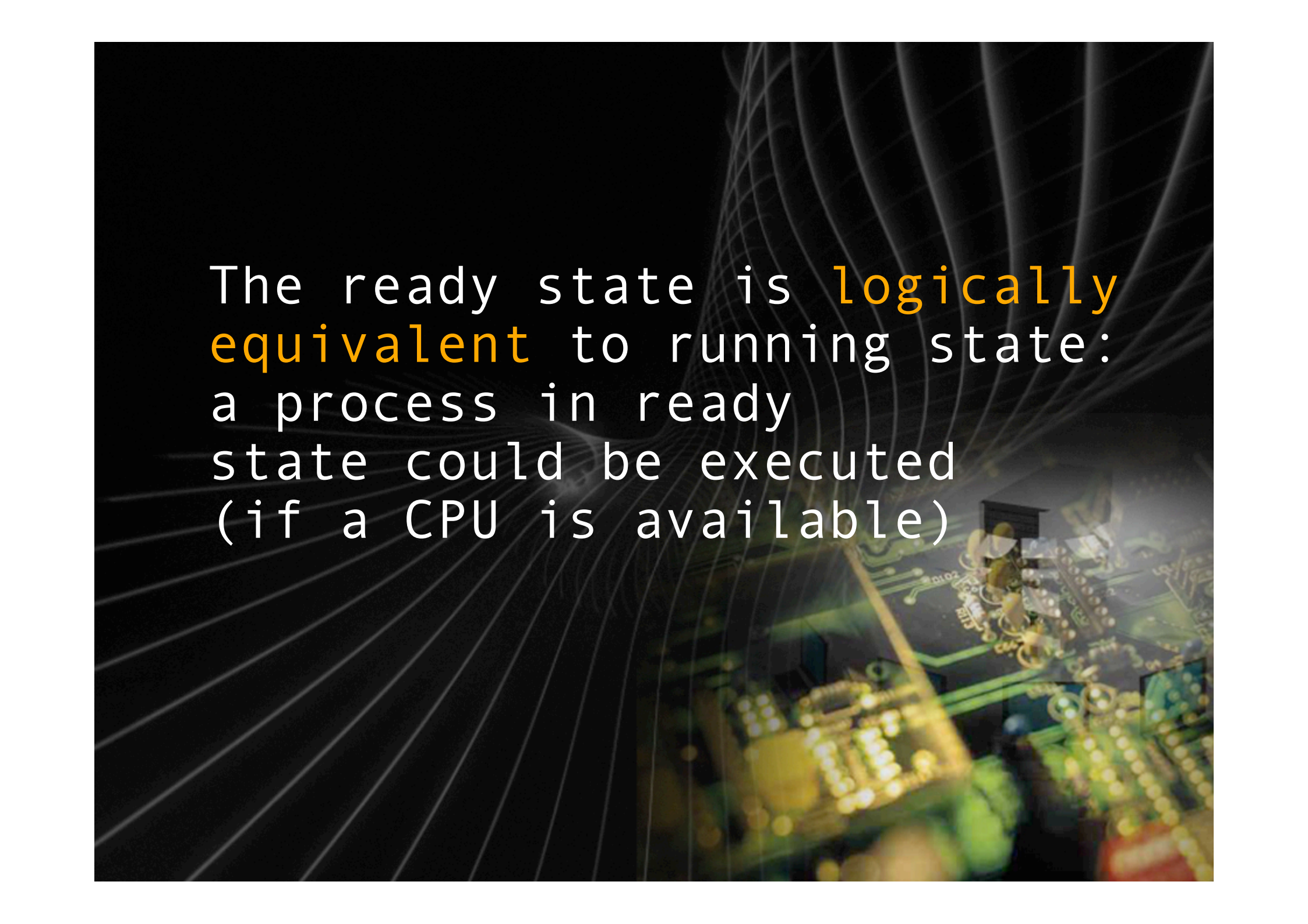
→ ready

(the process is waiting  
to be assigned to a CPU)

→ running

(a CPU is executing  
instructions of the  
process)





The ready state is **logically equivalent** to running state:  
a process in ready state could be executed  
(if a CPU is available)

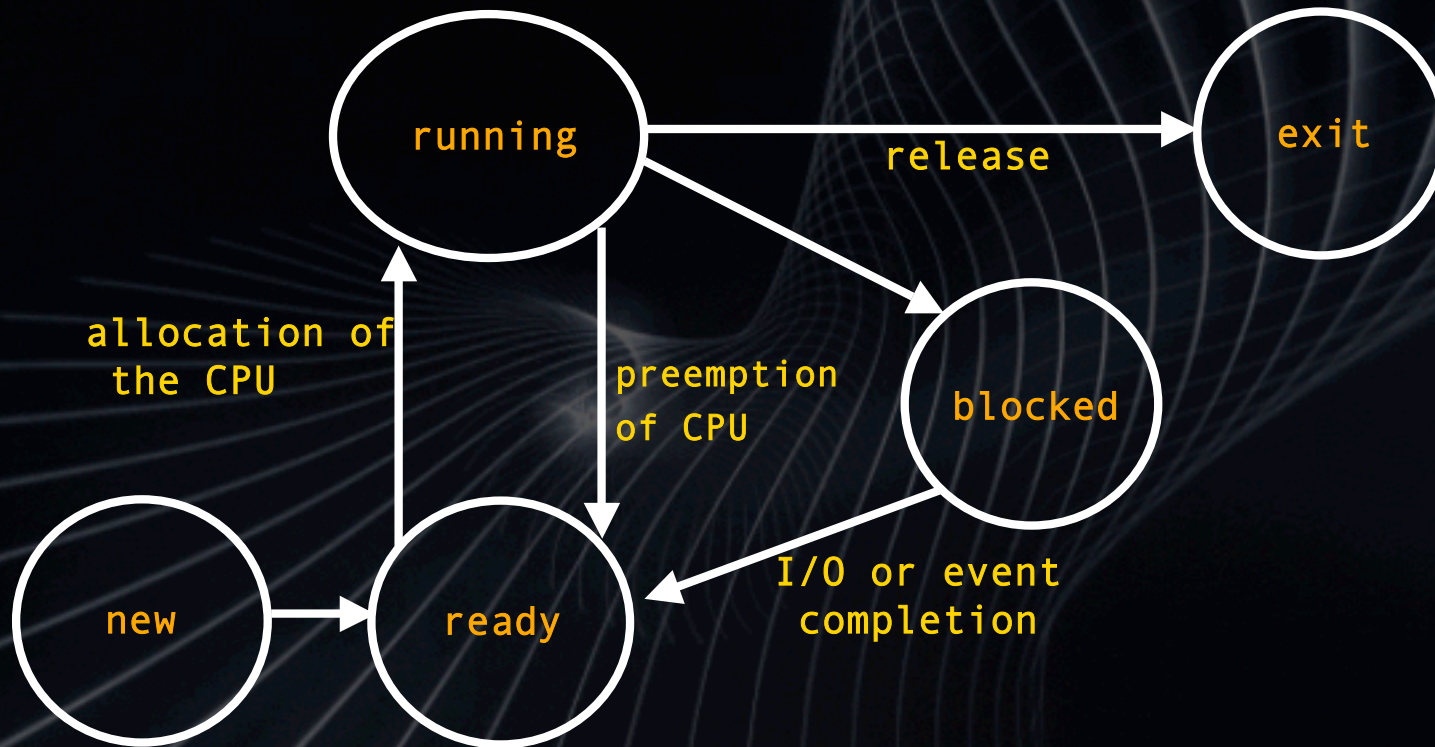


A **blocked** process  
cannot execute (even if  
the CPU is available)

Additional states:

→ **new** (typically, a new process created to execute a program, but not yet loaded in the ready queue)

→ **exit** (a process that has completed or aborted)





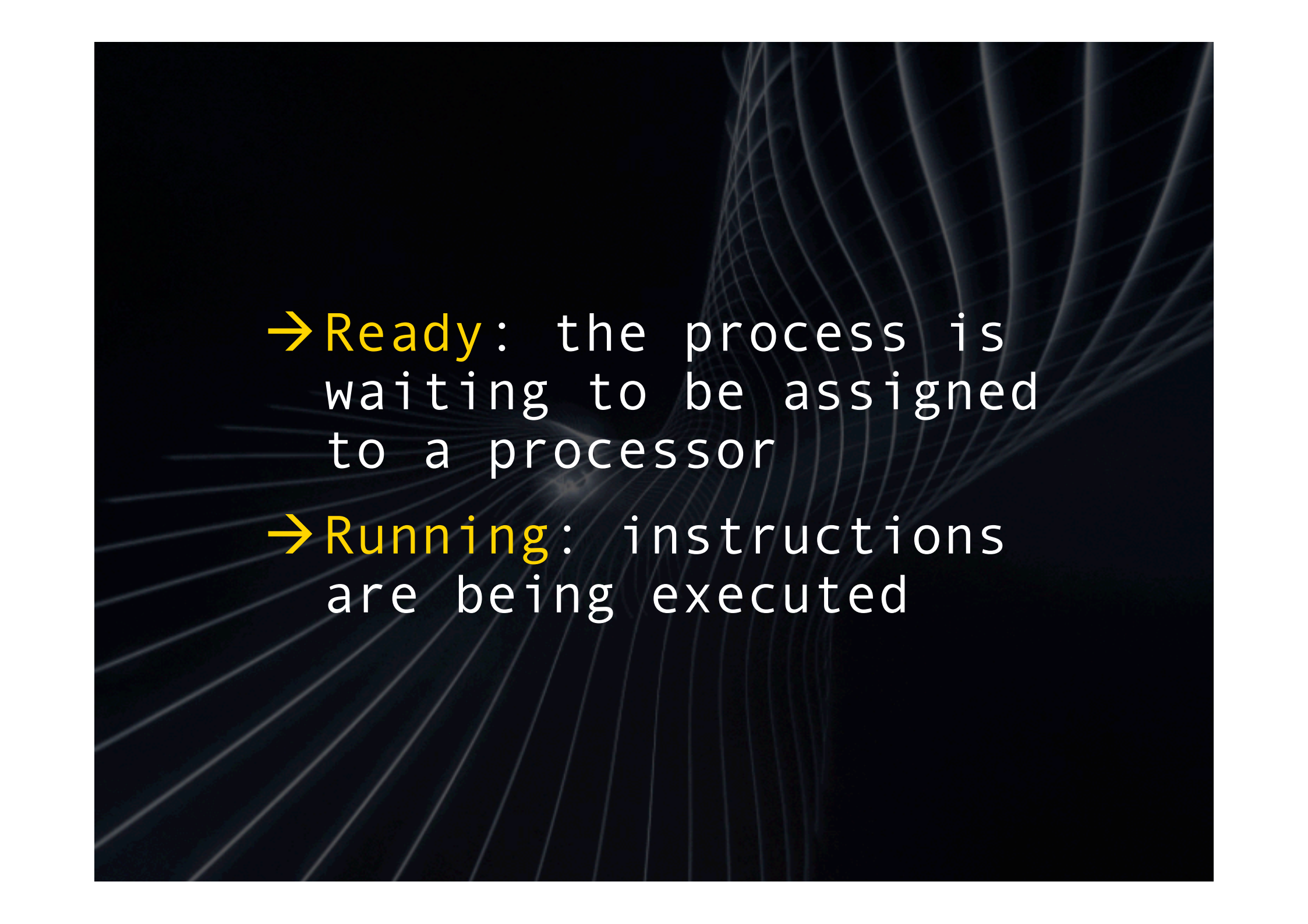
Additional state:

→ **swapped** (a process can be swapped temporarily out of memory to a backing store and then brought back into memory for continued execution)

# States of a UNIX process



→ **Init**: process  
is loaded in memory.  
The O.S. must build some  
tables to manage it

- 
- **Ready**: the process is waiting to be assigned to a processor
  - **Running**: instructions are being executed

→ **Sleeping**: the process is waiting for some event to occur

→ **Terminated**: the process has finished its execution

→ **Swapped**: the process  
is temporarily transferred  
to a backing store

→ **Zombie**: the process is terminated, but it is waiting that the parent process collects its status information

Multiprogrammed system  
(only one CPU):

→ One process  
is being executed  
(running state)



More others processes are  
waiting

→ to be executed by the CPU  
(ready state)

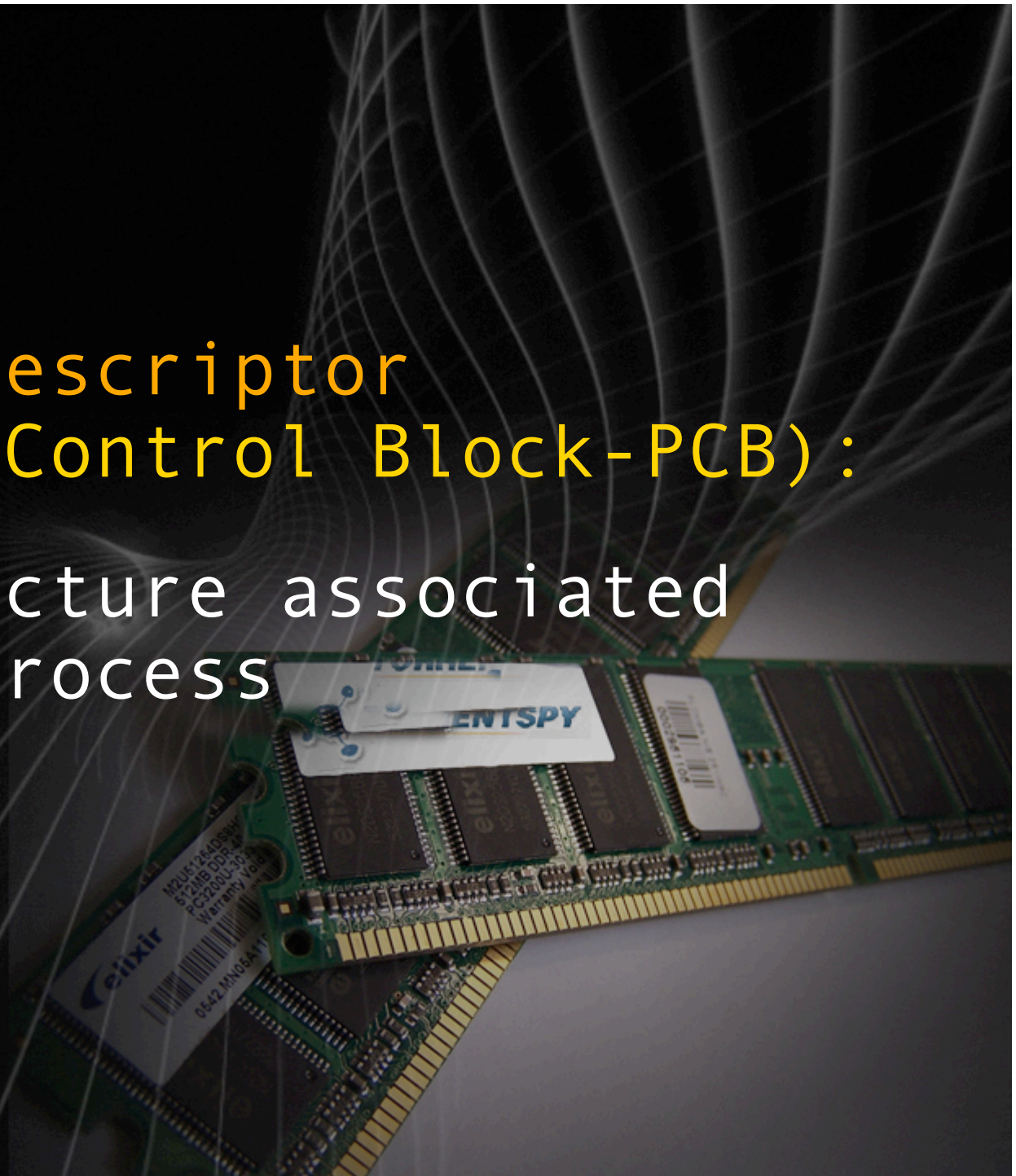
→ for some event to occur  
(I/O completion or  
reception of a signal)  
(waiting state)

# Process description

The image features a hand typing on a keyboard, with a dark, abstract background of glowing digital lines and curves. The text 'Process description' is centered in a yellow, sans-serif font.

# Process Descriptor (Process Control Block-PCB):

Data structure associated  
to each process



## Process descriptor

- Process state
- Process context
- CPU scheduling information
- Mem. management information
- Resource utilization
- Accounting information



Process state  
the state may be new,  
ready, running, waiting,  
halted, and so on

## Context

→ PC (the address of the next instruction to be executed)

→ PS (cond.codes- flags int. en./dis. bit, sup./user mode bit)

## Cpu registers

The registers vary in number and type depending on the computer architecture:  
accumulators, index registers,  
stack pointers, general  
purpose registers

## CPU scheduling information

This information includes  
a process priority, pointers  
to scheduling queues, others  
scheduling parameters



## Memory manag. information

the base and limit registers,  
the page tables or  
the segment tables depending  
on the memory system adopted

## Accounting information

amount of CPU and real time used, time limits, job or process numbers, and so on

## Resource utilization

I/O devices allocated to this process, a list of open files and so on.

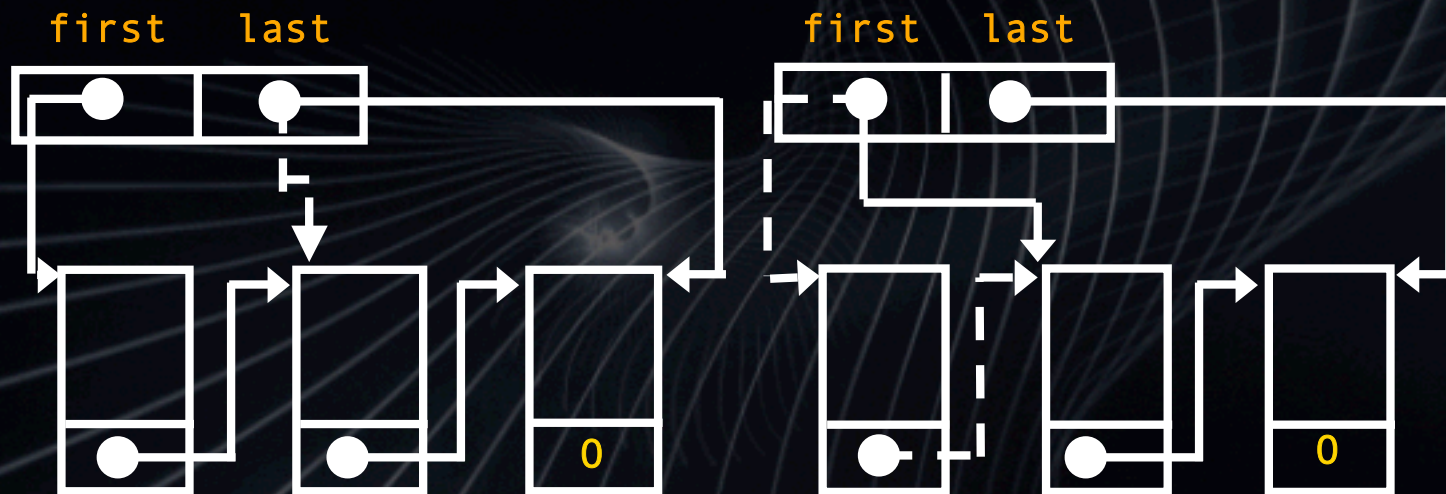
## Scheduling queues

**Ready queue:** The processes that are residing in main memory and are waiting to execute are kept on a list called the **ready queue**



This queue is generally  
implemented as a **linked  
list**

# Process queues



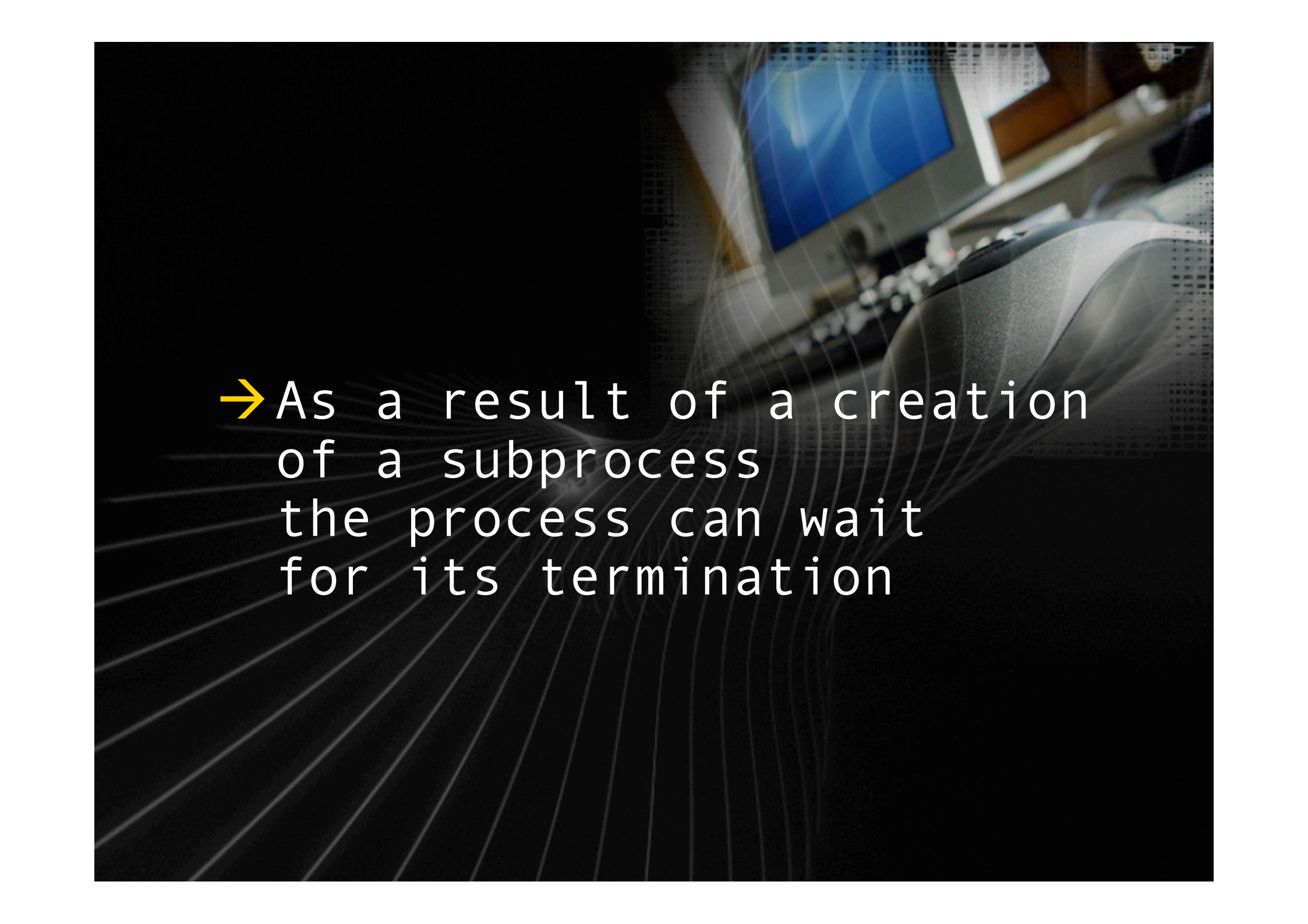
**Blocked queue:** The processes that are waiting for some event to occur (I/O completion or reception of a signal) are kept on a list called the **blocked queue**

- A new process is initially put in the **ready queue**. It waits there until it is selected for execution
- One of the following events can occur to promote it:

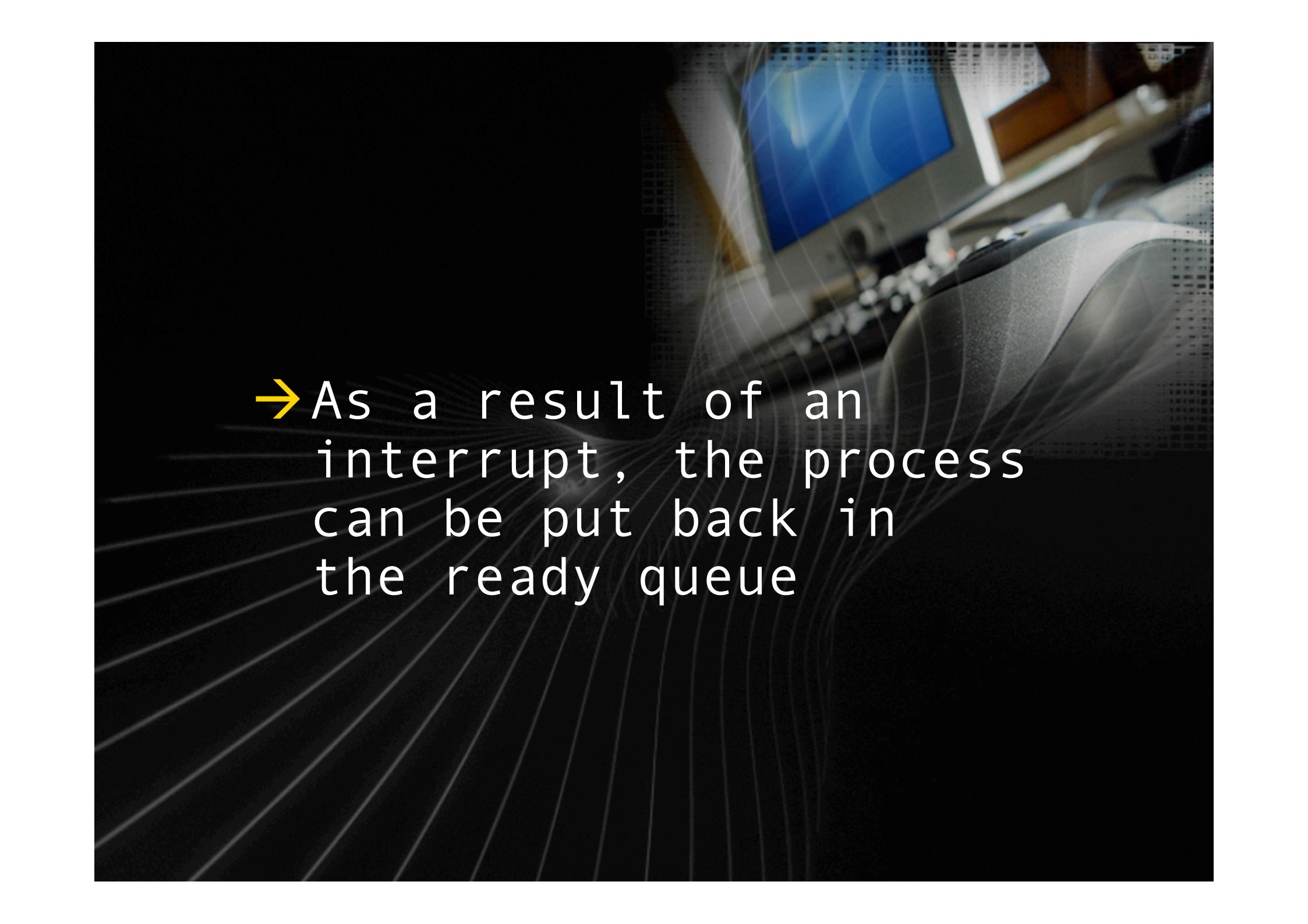




→ As a result of an I/O request the process is placed in an I/O queue



→ As a result of a creation  
of a subprocess  
the process can wait  
for its termination



→ As a result of an interrupt, the process can be put back in the ready queue

## Context switch

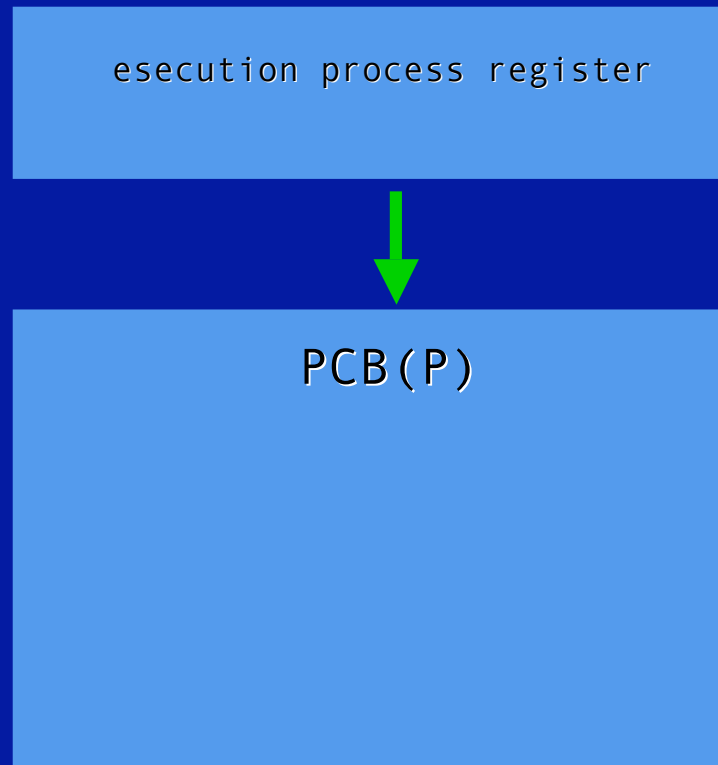
- ▶ The kernel **saves the context** of the old process in its PCB.
- ▶ The kernel inserts the PCB in the **blocked or ready queues** (depending on the process state.)

- ▶ The kernel selects a new process from the ready queue and its identifier is loaded in the execution process register. (short term scheduling)
- ▶ The context of the new process is loaded in the CPU registers.

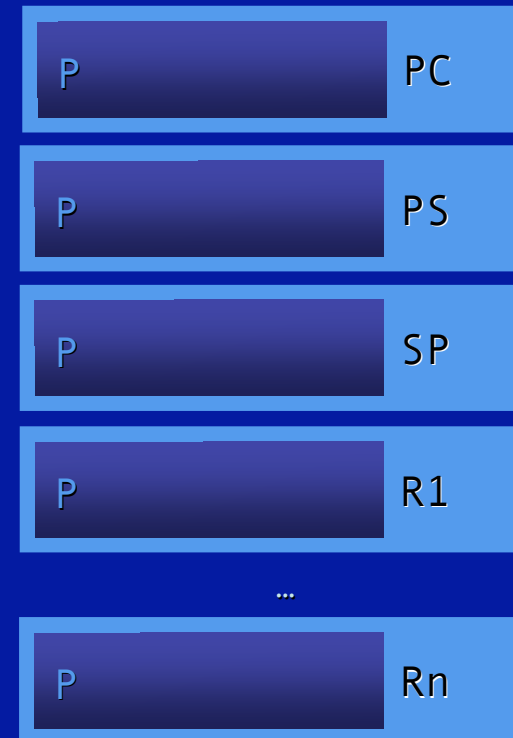
## Context switch

- ▶ Context-switch time is **pure overhead**, because the system does no useful work while switching.
- ▶ Its **speed varies**, depending on the mem. speed, the number of regist. and **special instructions**.

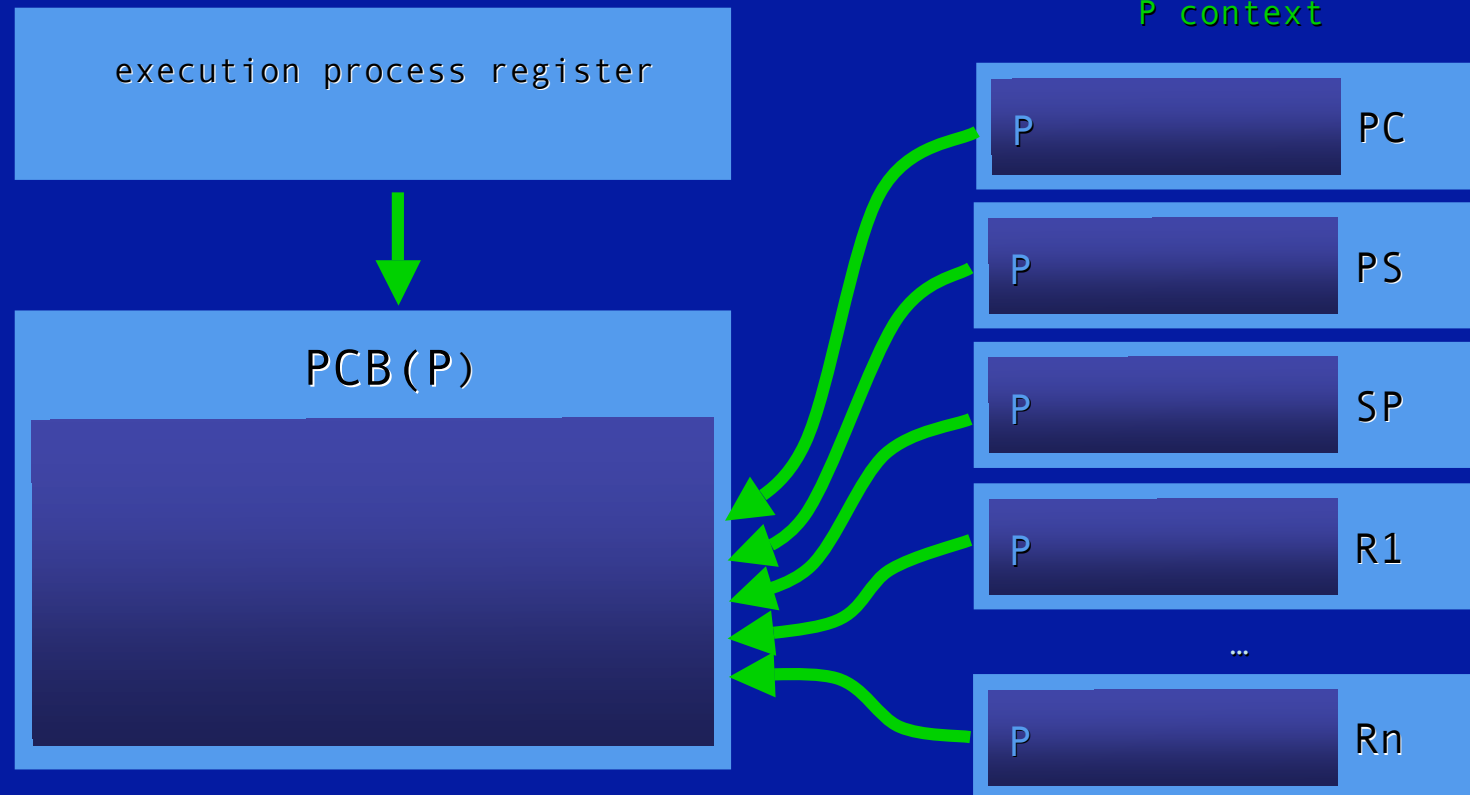
## Proces P execution



P context

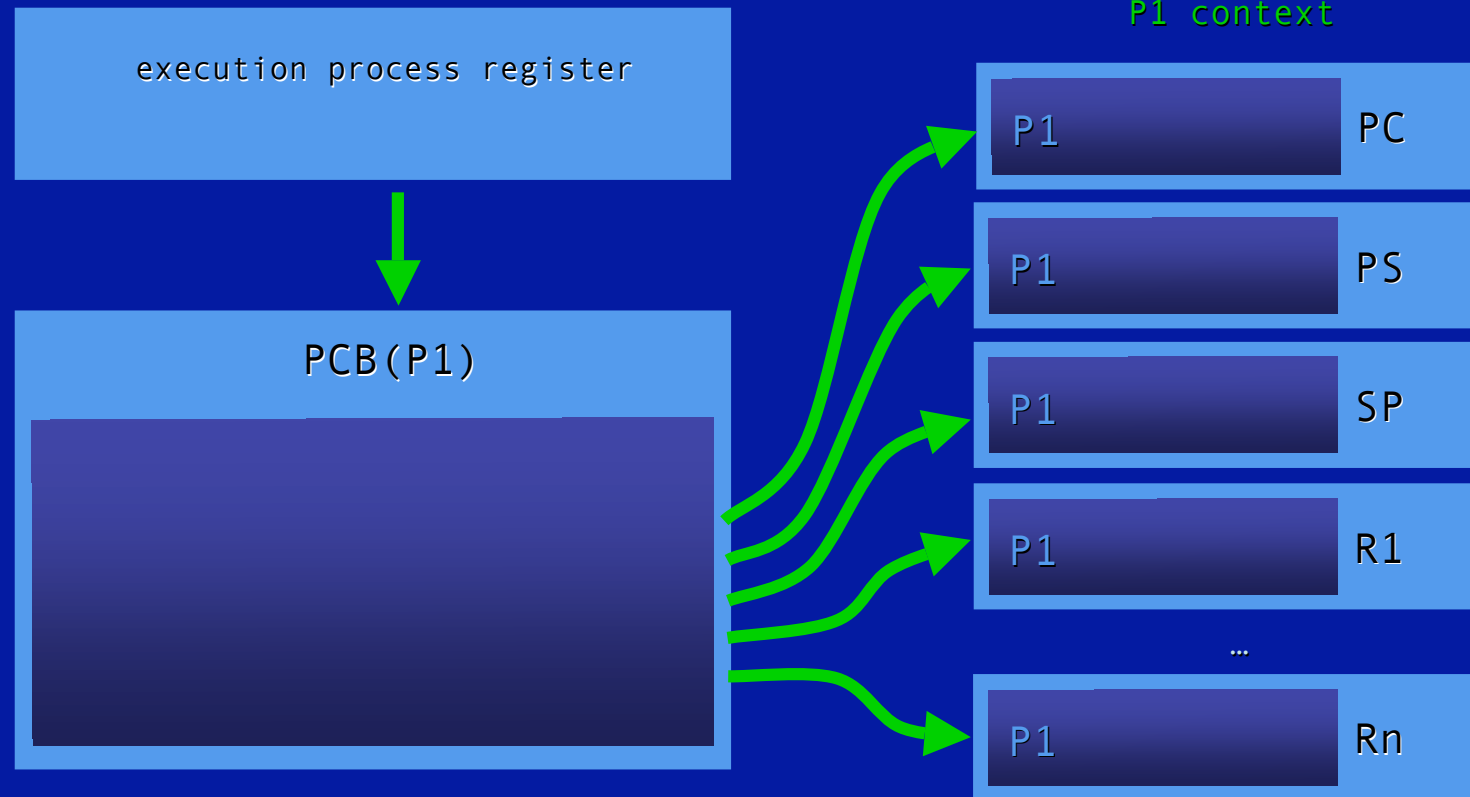


## P context saving





## P1 context loading



Operating Systems

Lezione 6 - Proprietà dei processi

# Scheduling

**NETTUNO**  
NETWORK PER L'UNIVERSITÀ OVUNQUE

Prof. Maurelio Boari

Alma Mater Studiorum - Università di Bologna

## Process scheduling

- ▶ A process is executed until it must wait, typically for the completion of some I/O requests or because of termination.
- ▶ When the CPU becomes idle, the O.S must select one of the processes in the ready queue to be executed.

## Process scheduling

- ▶ The selection process is carried out by the **short term scheduler**, following particular policies.

## Short term scheduler

CPU scheduling decision may take place:

- When a process transits from the running state to the waiting state (for example, I/O request).
- When a process switches from the running state to the ready state (for example, when an interrupt occurs).

## Short term scheduler

3. When a process switches from the waiting state to the ready state ( for example, I/O completion) and its priority is higher than the one of the executing process
4. When a process complites

## Short term scheduler

- ▶ **Non preemptive scheduling:** once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either:
  - by terminating (4) or by switching to the waiting state (1).

## Short term scheduler

- ▶ **Preemptive scheduling:** the currently running process may be interrupted and moved to the ready state.
- ▶ The decision to preempt may be performed when conditions 2) or 3) occur.



## Alternative scheduling policies

### First-Come-First Served (FIFO)

When the currently running process ceases to execute, the process that has been in the ready queue the longest is selected for running.

(nonpreemptive)

## RR (round robin)

- ▶ Time sharing systems.
- ▶ RR is **preemptive**: a small unit of time is defined (**time quantum**) The ready queue is treated as a circular queue.
- ▶ RR goes around the ready queue, allocating the CPU to each process for a time quantum.

### Priority scheduling

- ▶ A priority is associated with each process and the CPU is allocated to the ready process with the highest priority.
- ▶ Equal priority processes are scheduled in FCFS order
- ▶ Priorities are generally some fixed range of numbers (generally, low numbers represent high priority)

### Multilevel queue scheduling

- ▶ This scheduling algorithm partitions the ready queue into several separate queues.
- ▶ one process is permanently assigned to a queue, generally based on some properties of the process such as memory size, process priority, process type,...

- ▶ Foreground (interactive) processes
- ▶ Background (batch) processes
- ▶ Foreground processes have priority (externally defined) over the background processes.
- ▶ Background processes run when the CPU is not executing any foreground process

▶ Starvation problem

A condition in which a process is indefinitely delayed because other processes are always given the preference.

For instance, a priority scheduling algorithm can leave some low-priority process waiting indefinitely

## Medium term scheduling

- ▶ Medium term scheduling is part of the **swapping function**
- ▶ The swapping decisions are based on the needs to handle the correct degree of multiprogramming.

## Long term scheduling

- ▶ That determines which programs are allowed to enter the system for processing needs.
- ▶ Once admitted the job becomes a process and it is added to the ready queue managed by the short term scheduling



## Long term scheduling

- ▶ In a batch system, newly submitted jobs are routed to disk and held in a batch queue
- ▶ The decision as to which jobs admit may include priority or to keep a mix of CPU bound and I/O bound processes

### Queueing diagram for scheduling

