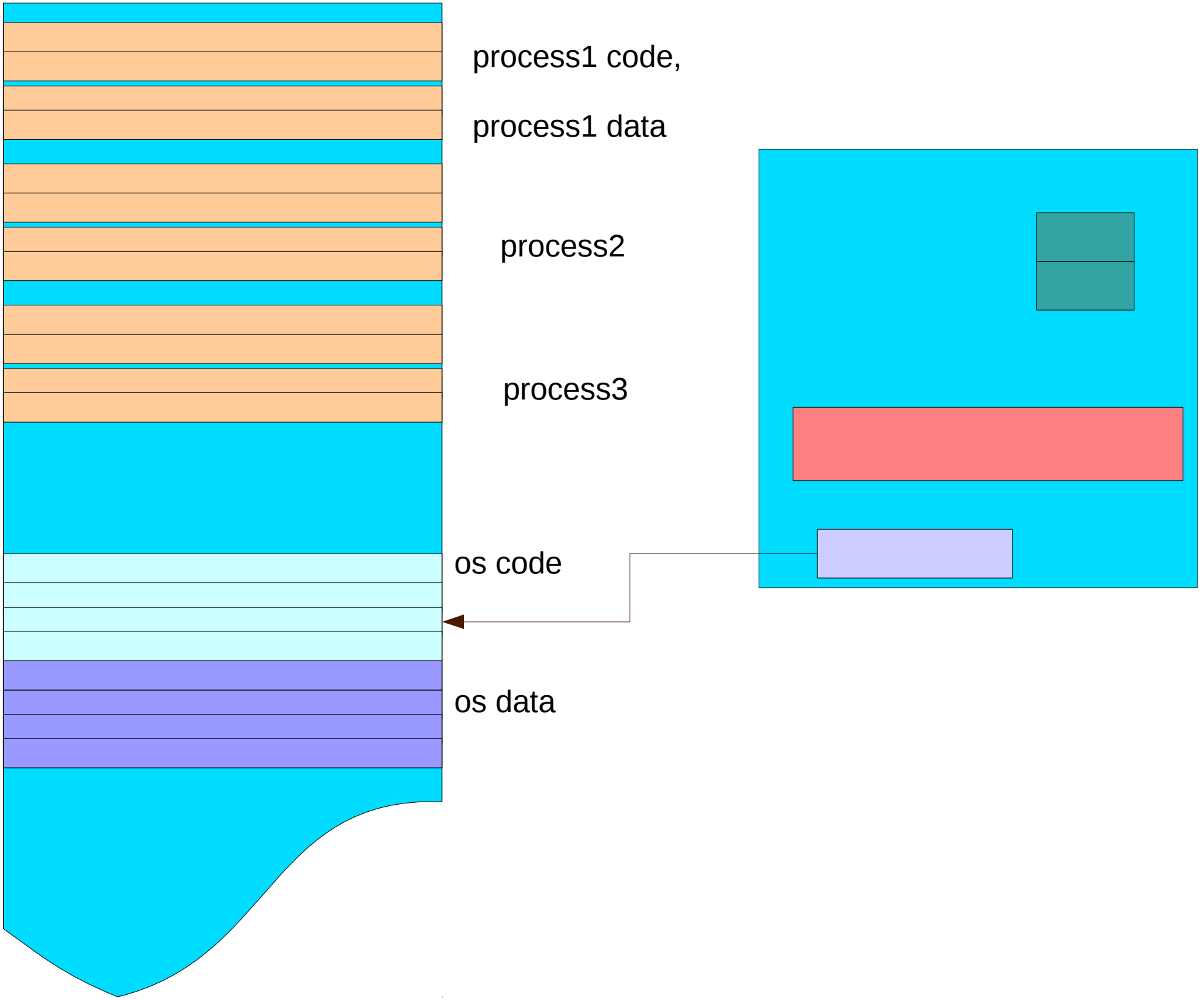
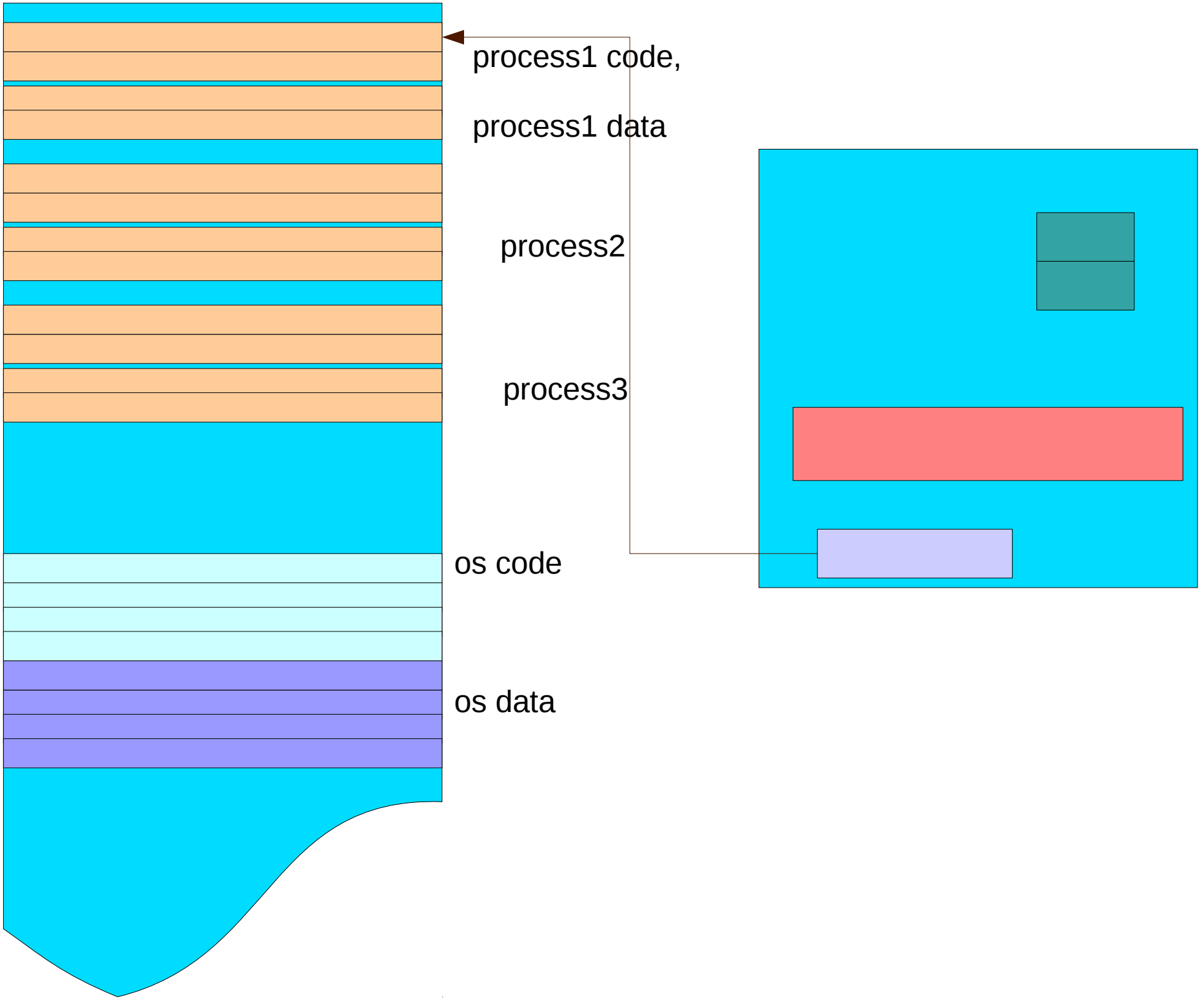
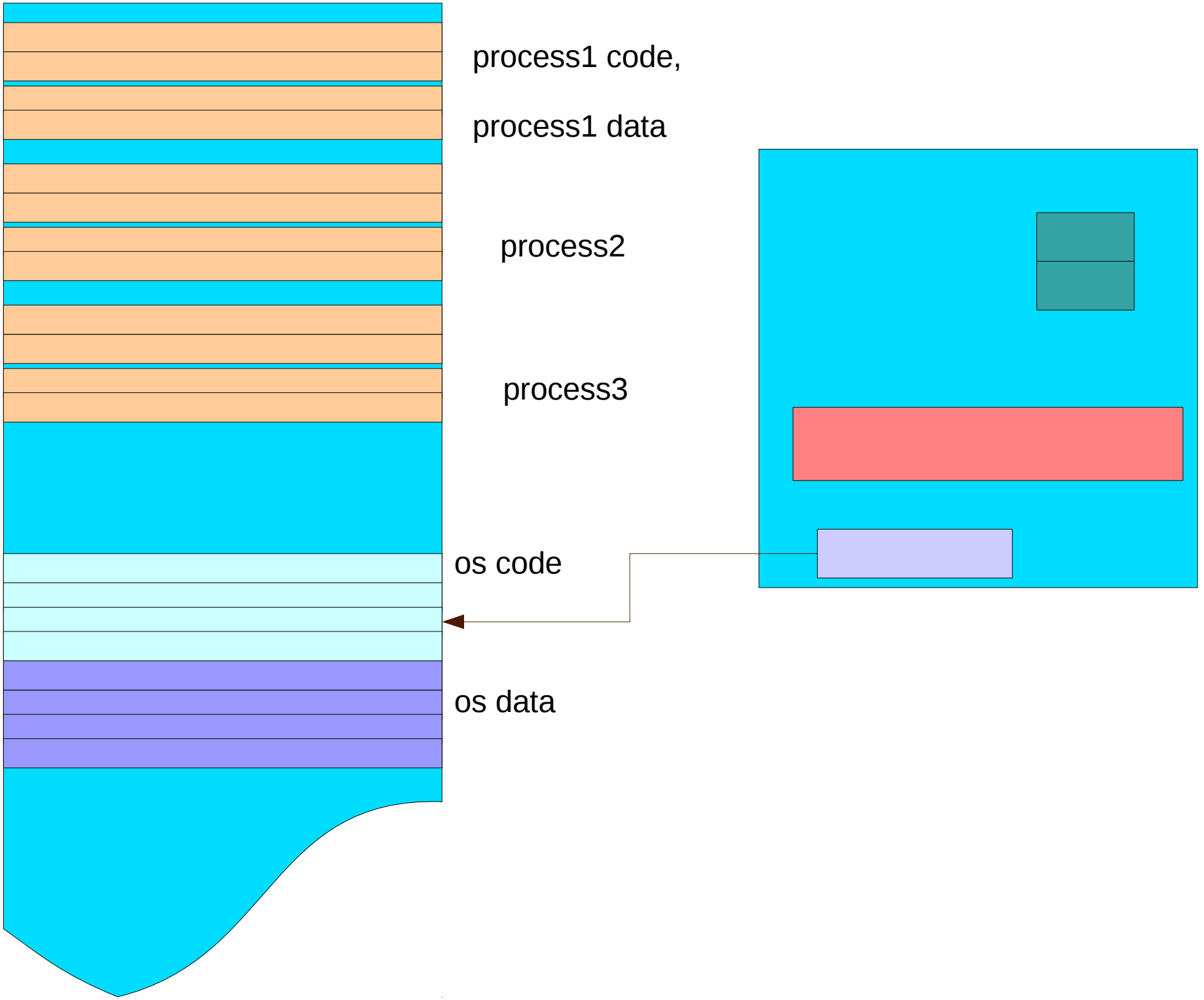
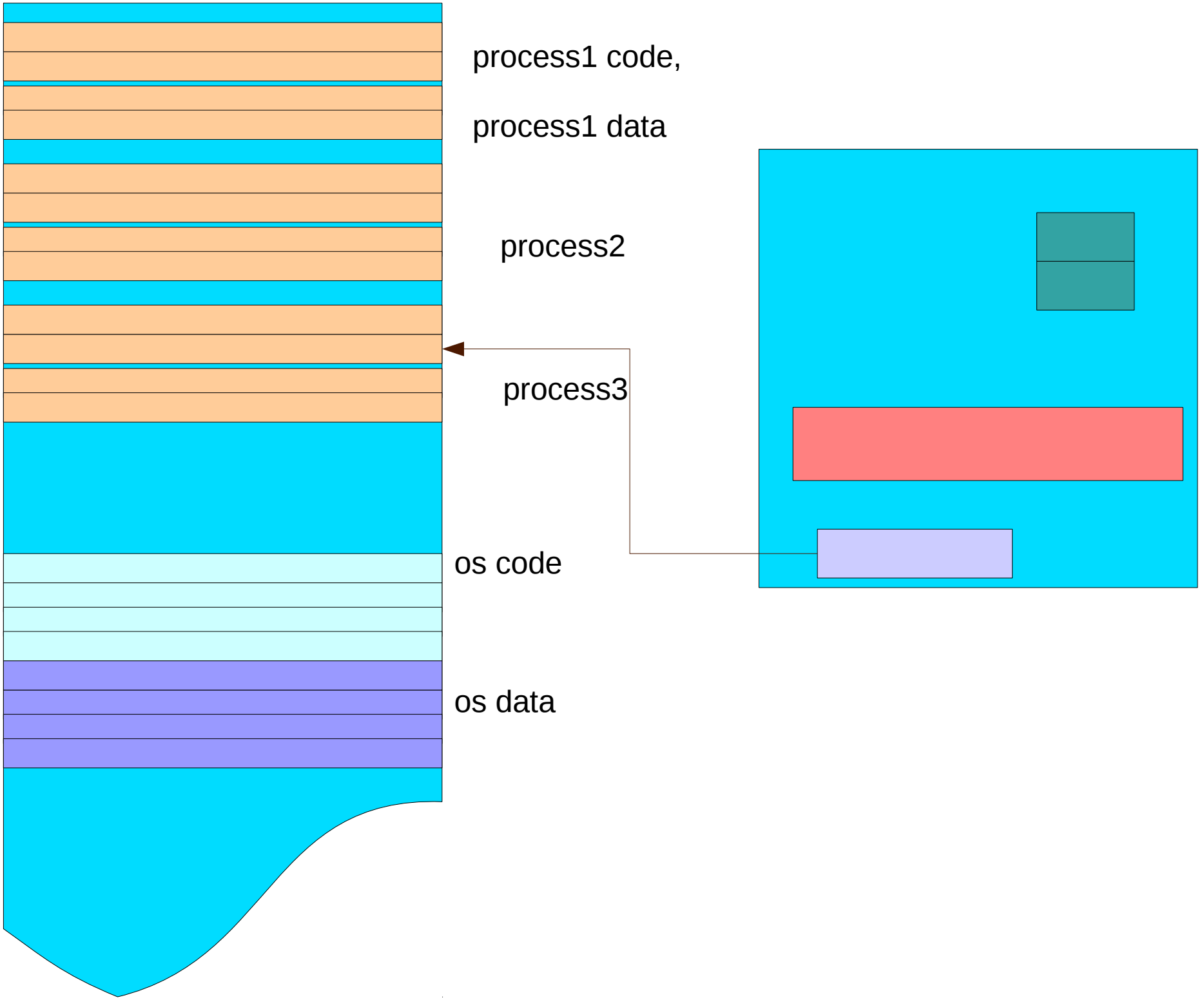
An entry into an OS..
It's a world of PROCESSES

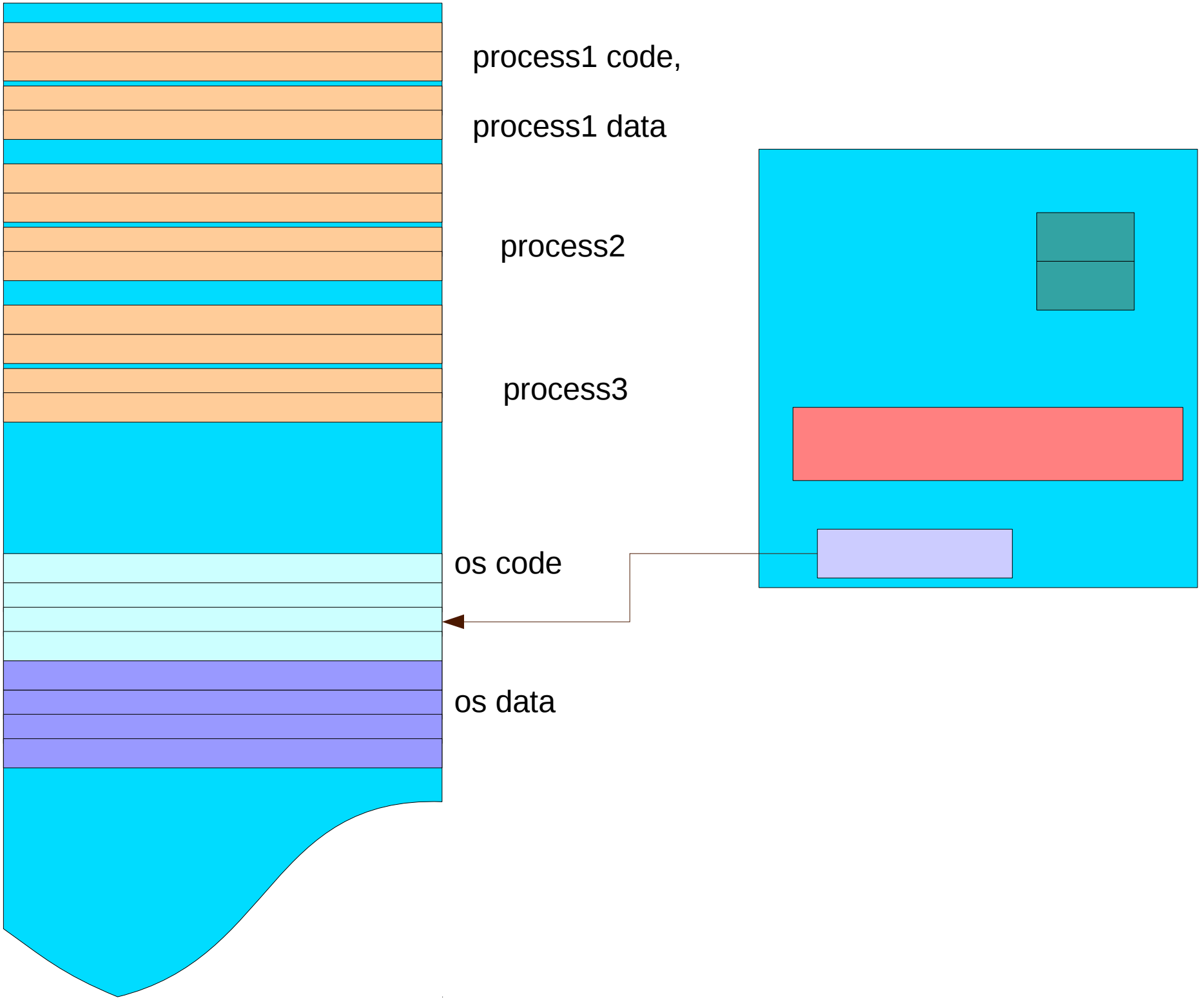
Rushikesh K Joshi
IIT Bombay

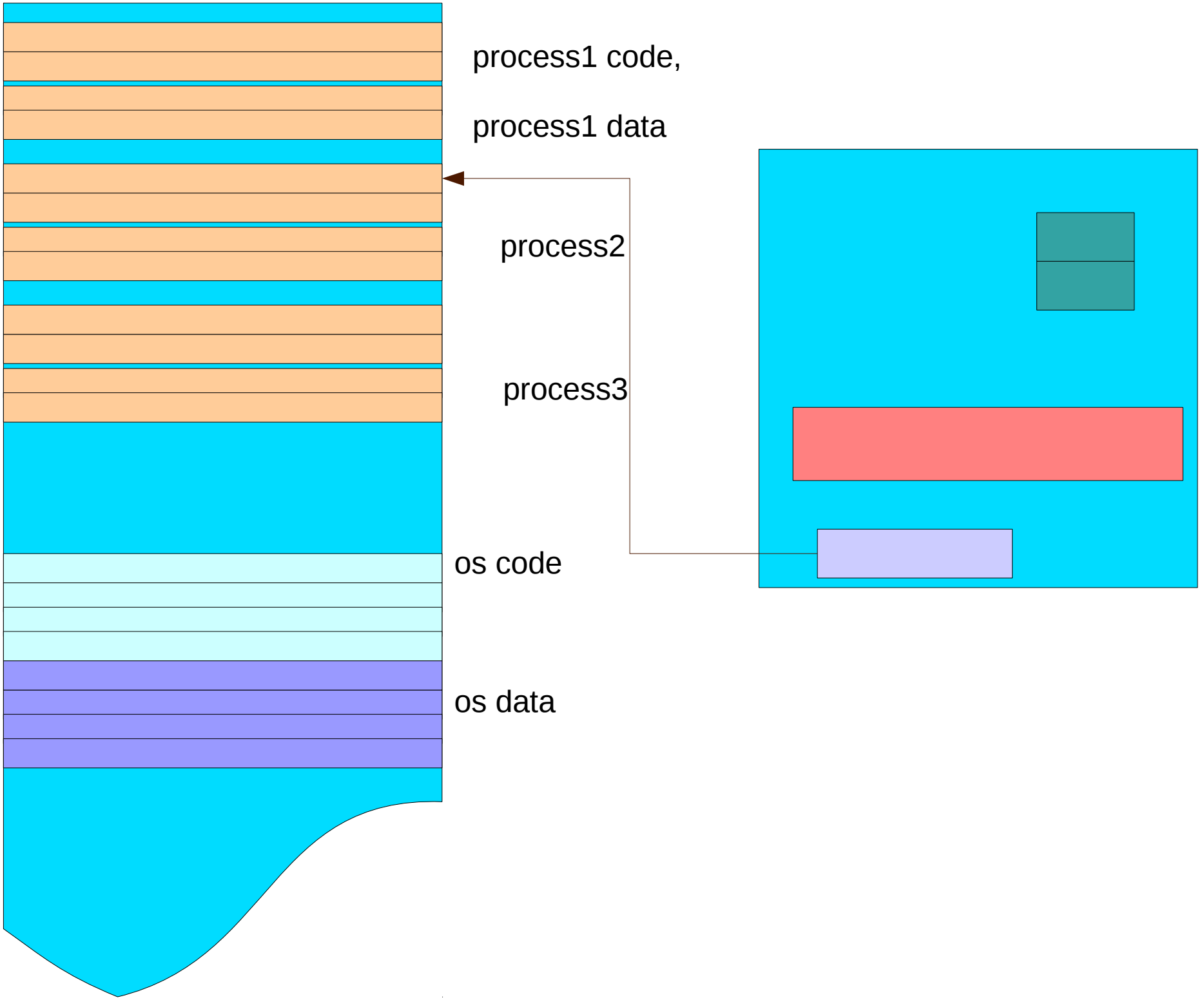

process1 code,
process1 data
process2
process3
os code
os data

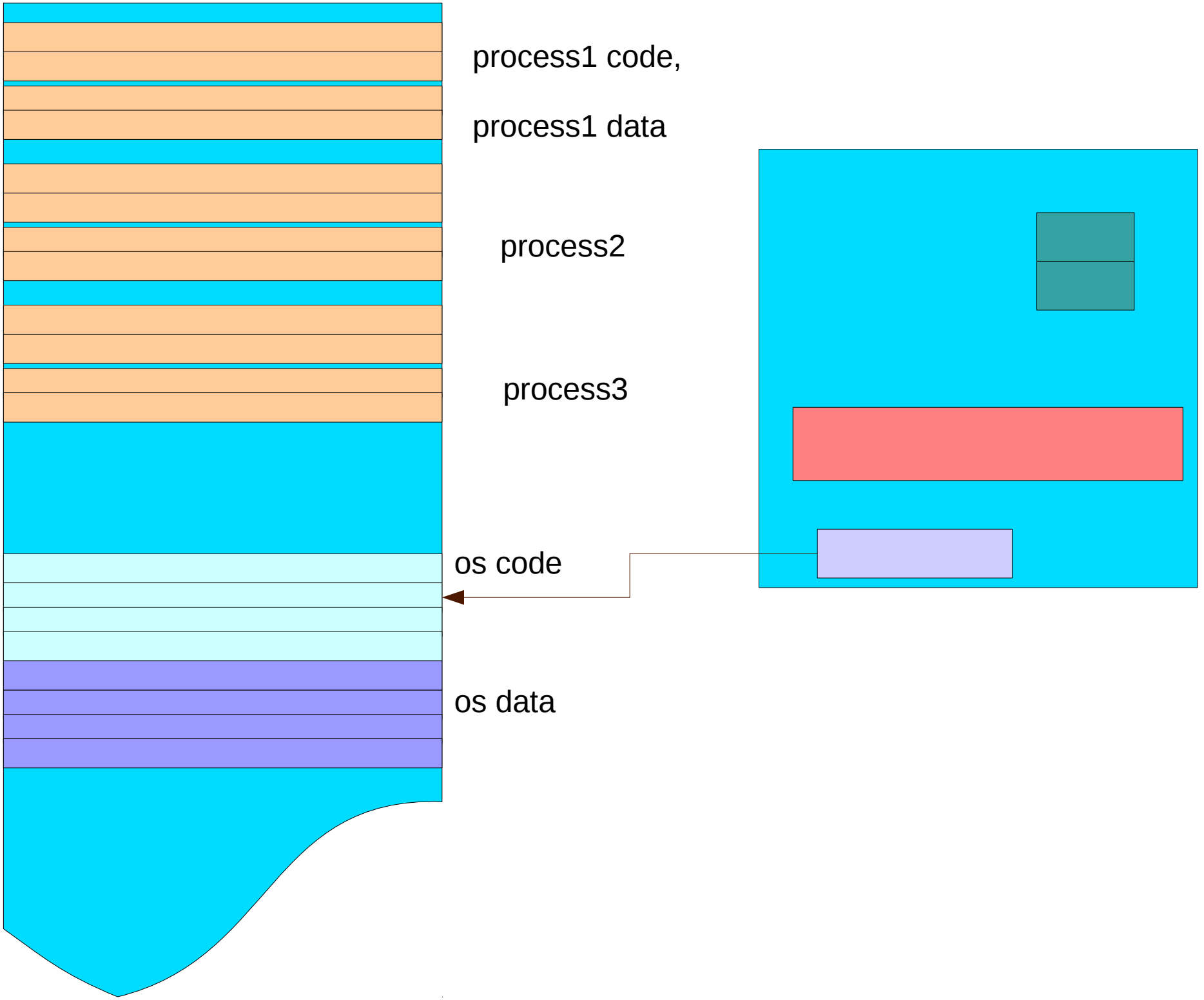

process1 code,
process1 data
process2
process3
os code
os data

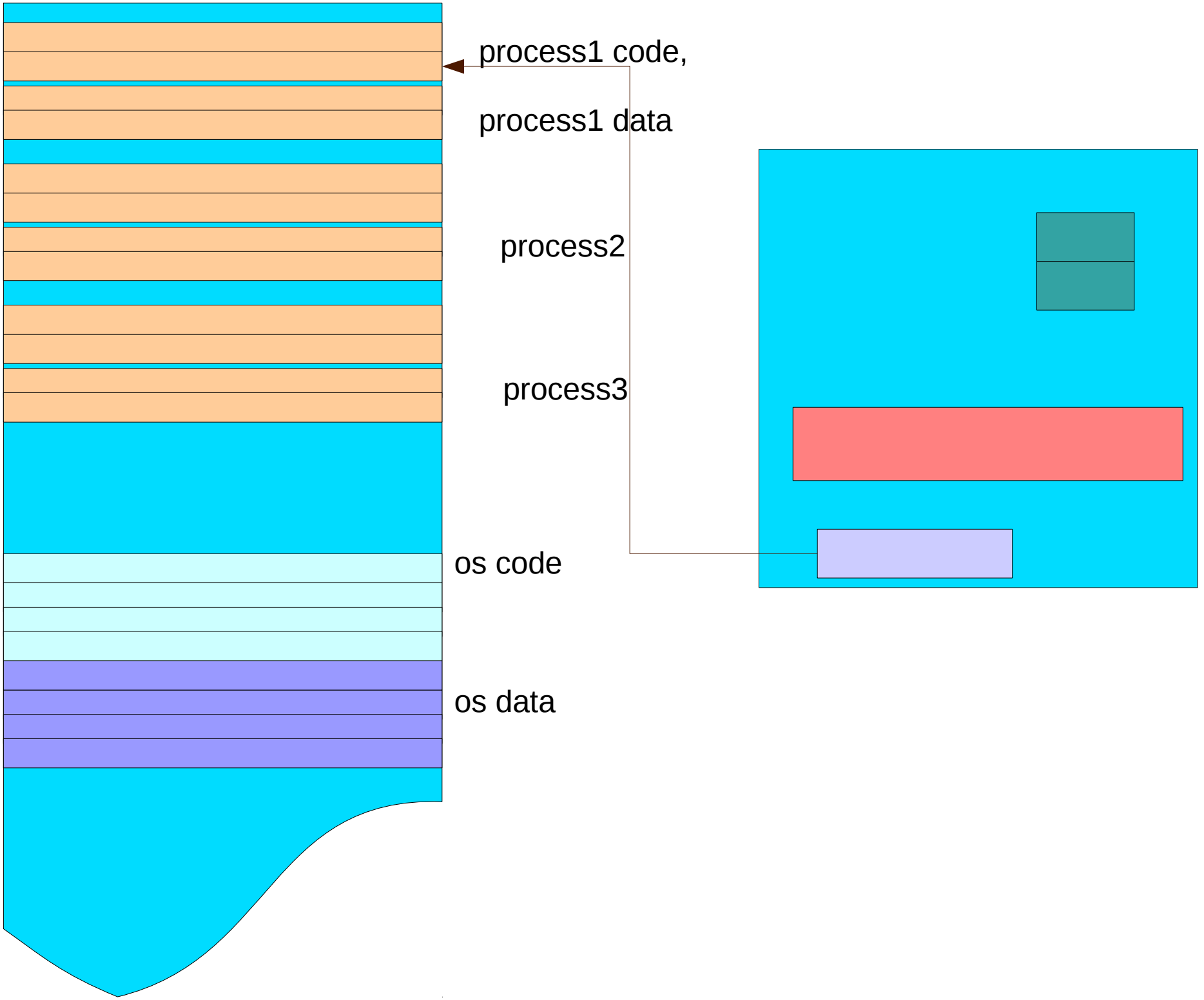

process1 code,
process1 data
process2
process3
os code
os data

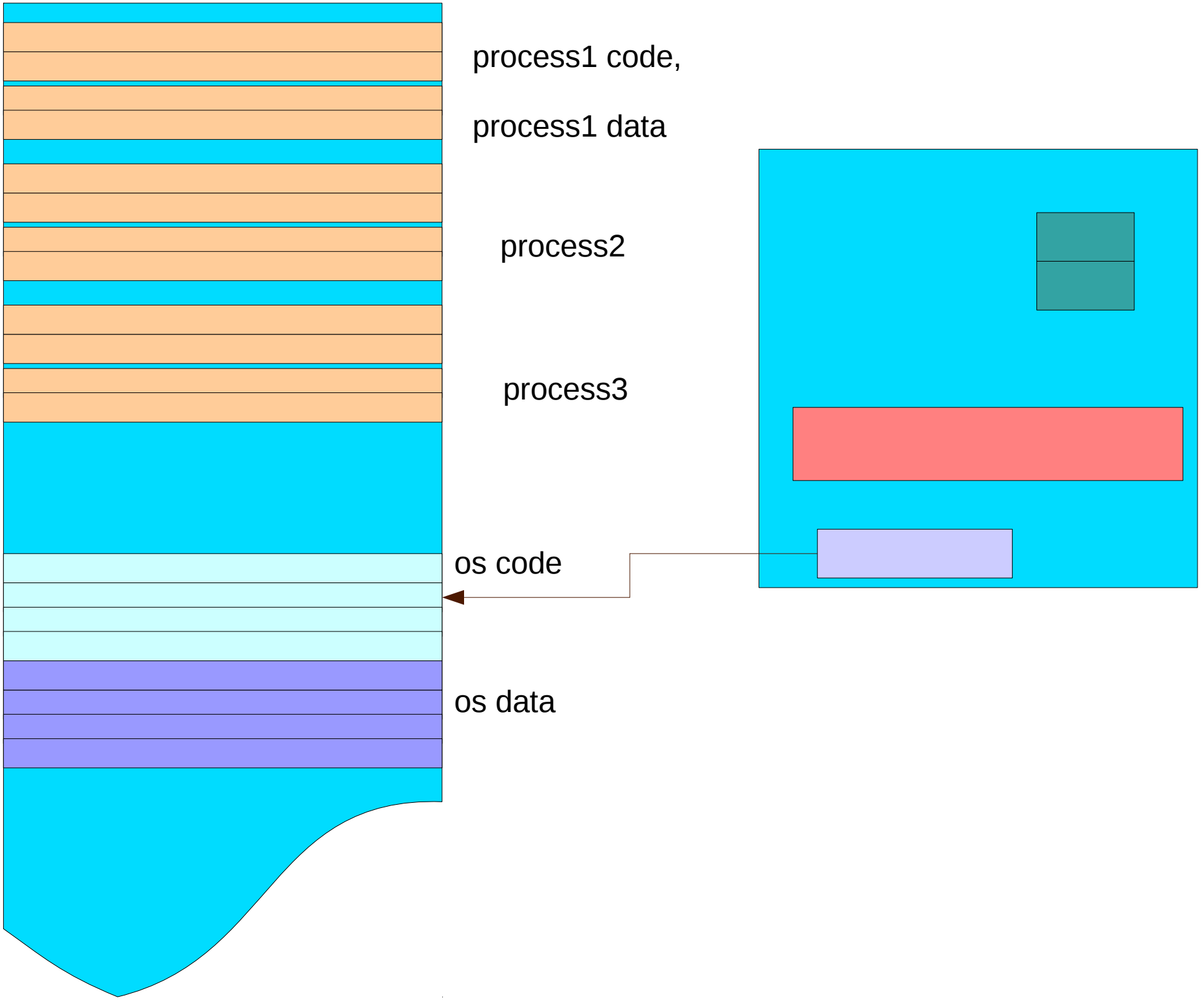

process1 code,
process1 data
process2
process3
os code
os data

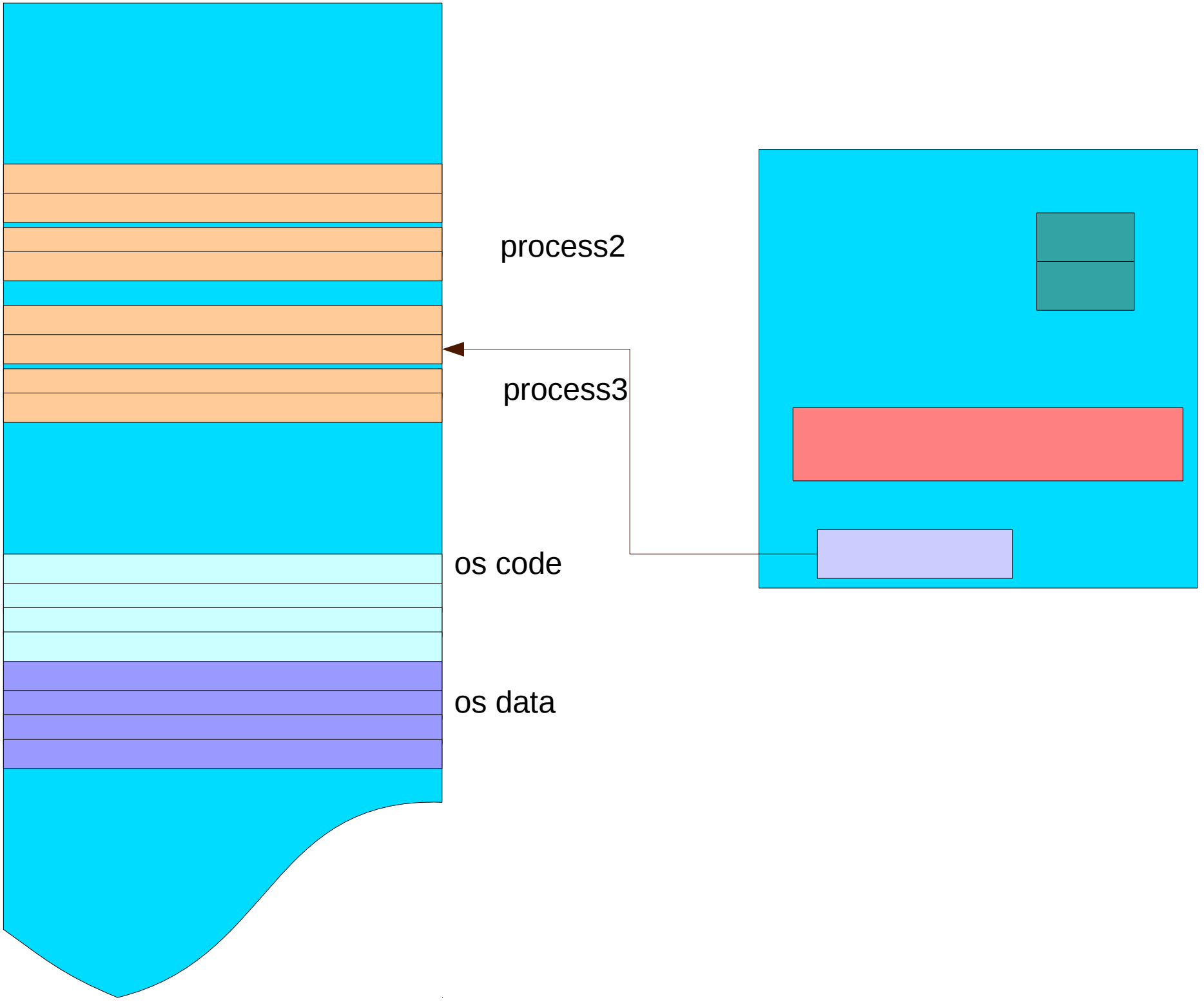

process1 code,
process1 data
process2
process3
os code
os data

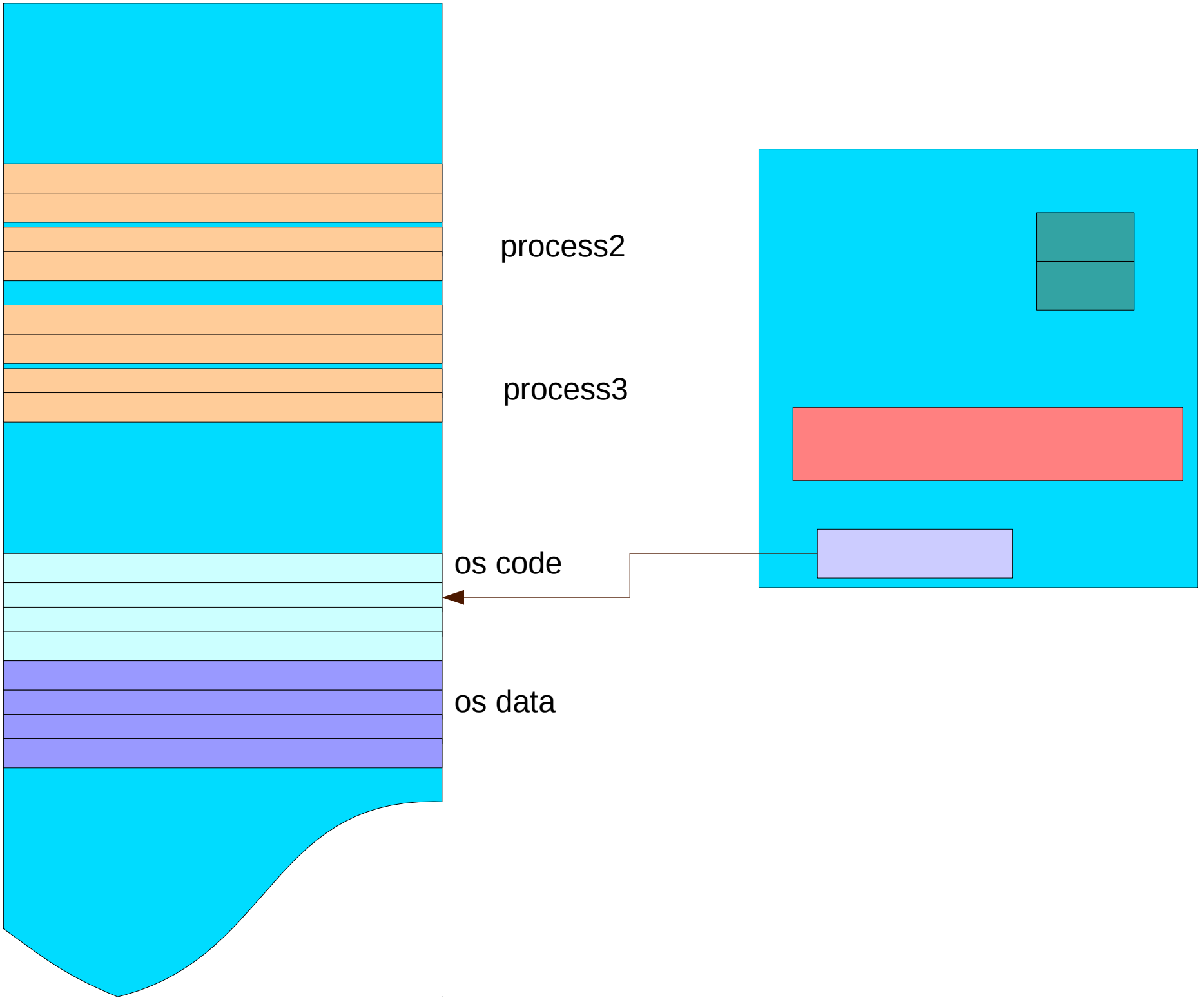

process1 code,
process1 data
process2
process3
os code
os data

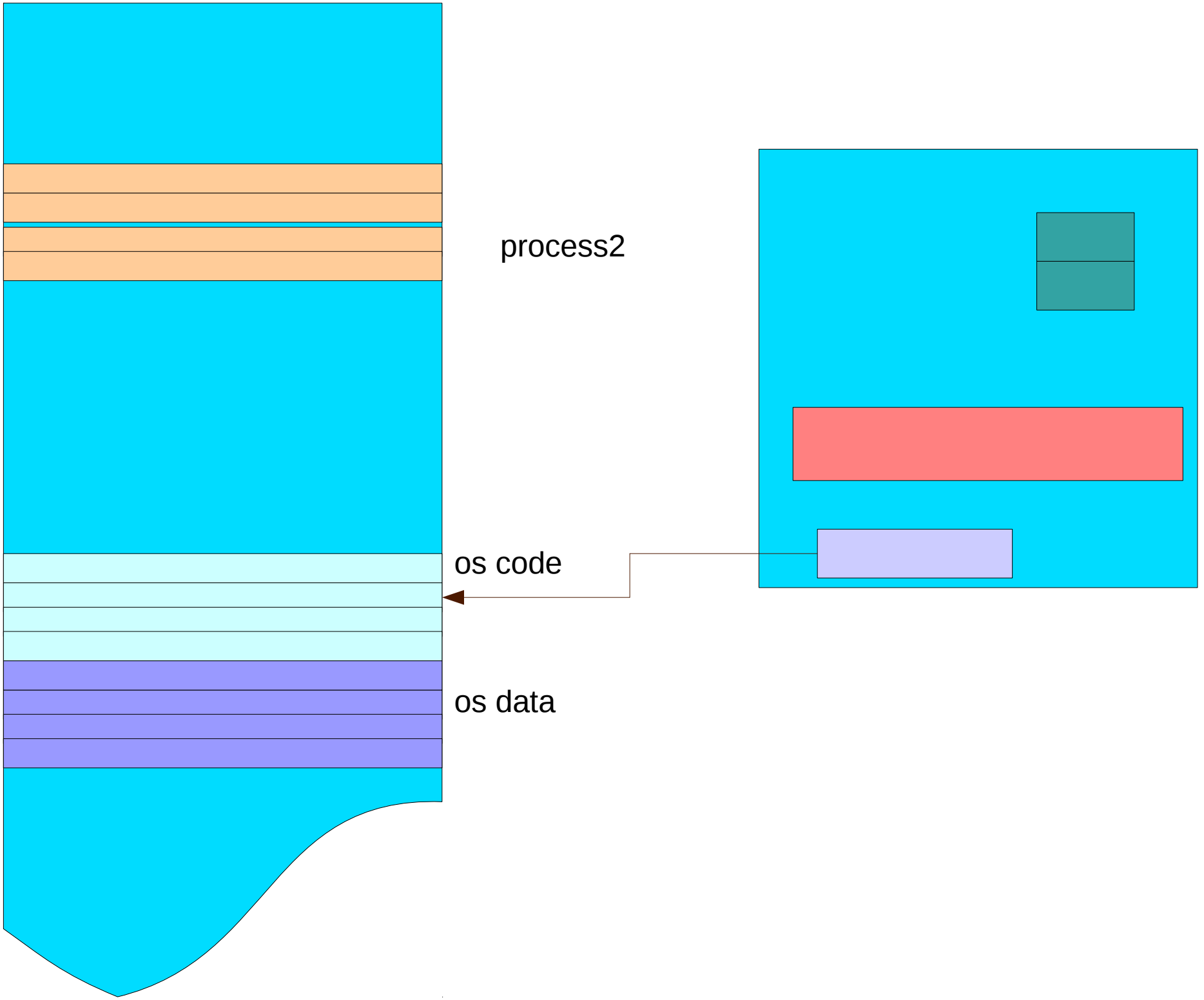

process1 code,
process1 data
process2
process3
os code
os data

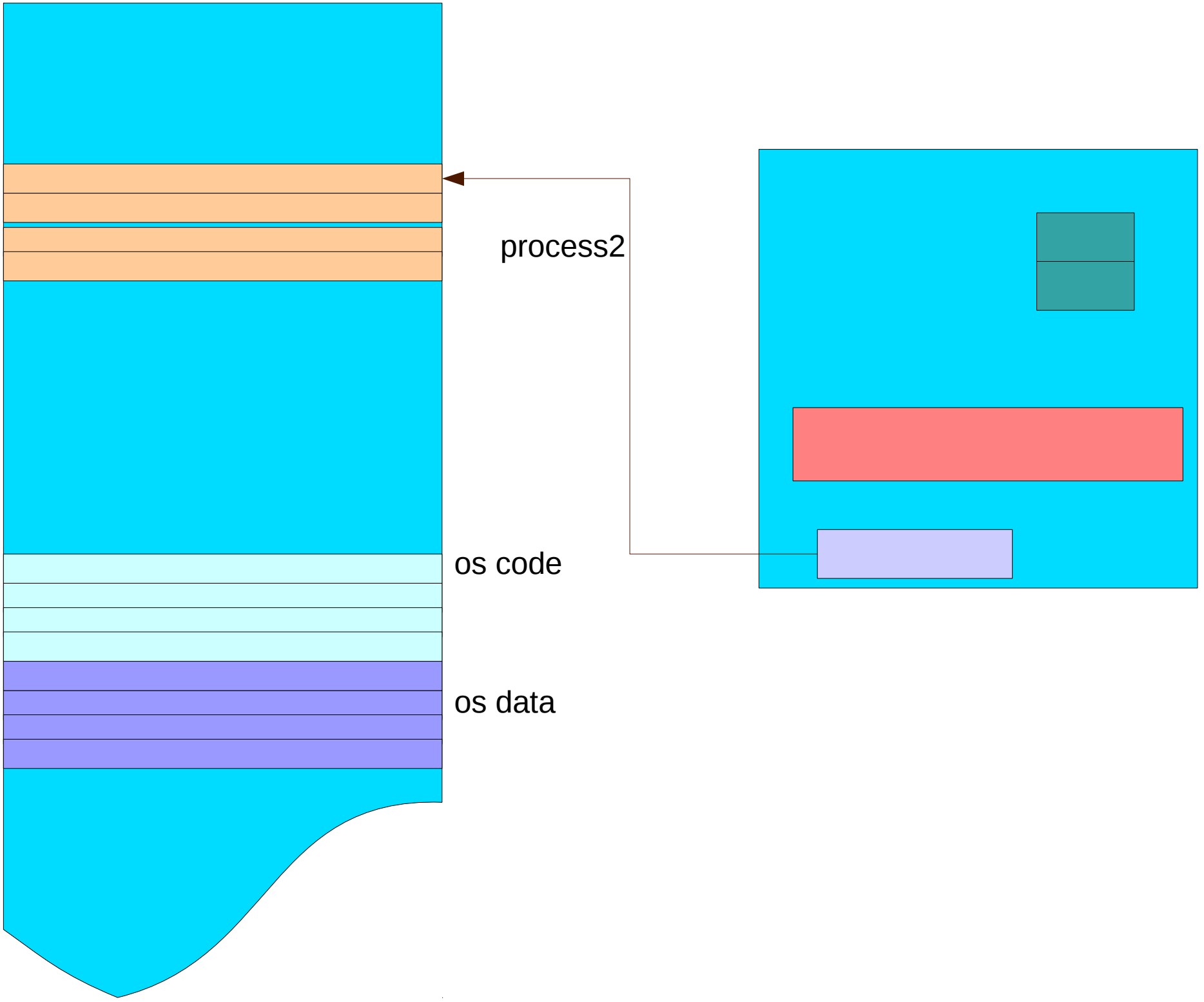

process1 code,
process1 data
process2
process3
os code
os data

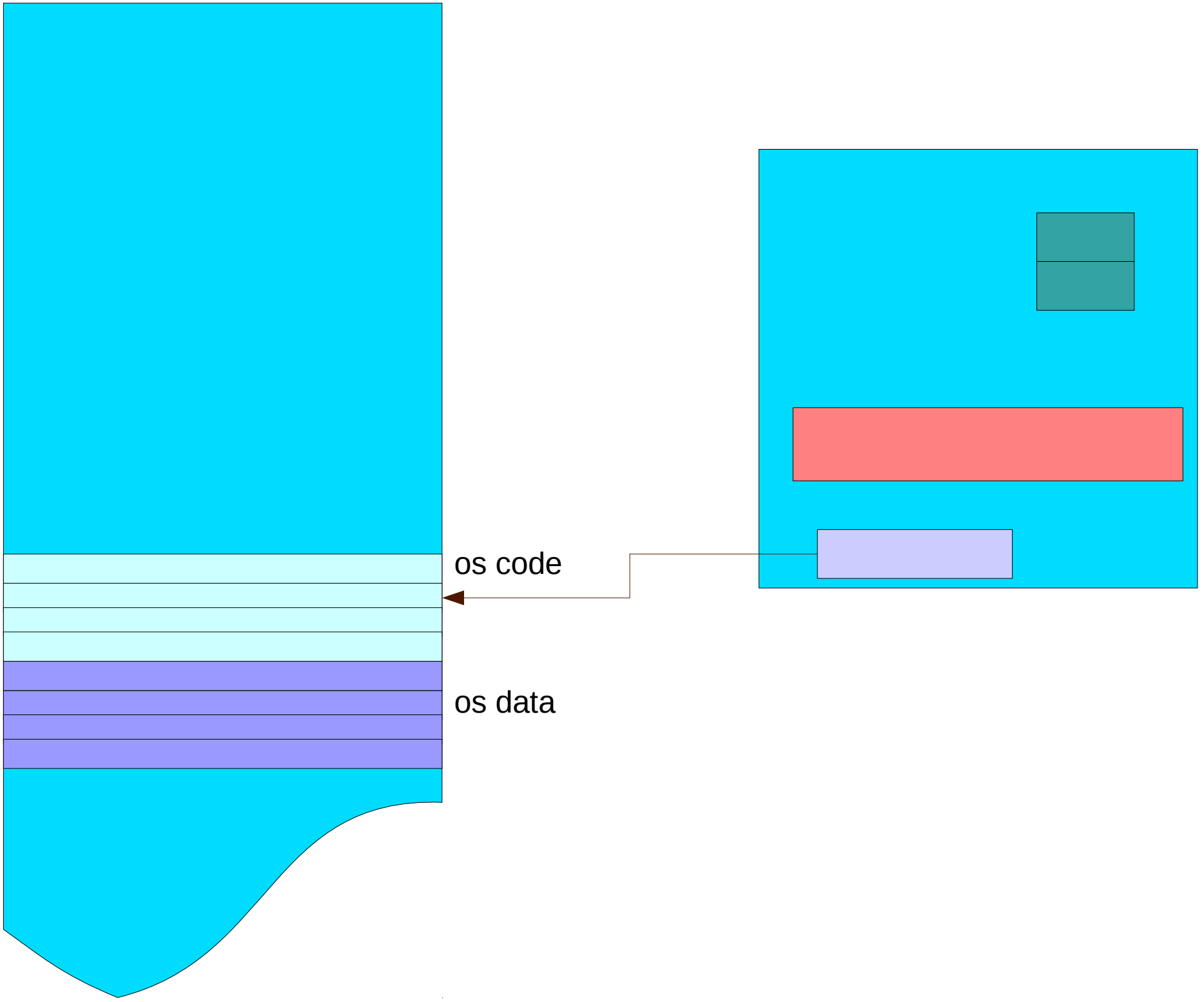

process1 code,
process1 data
process2
process3
os code
os data

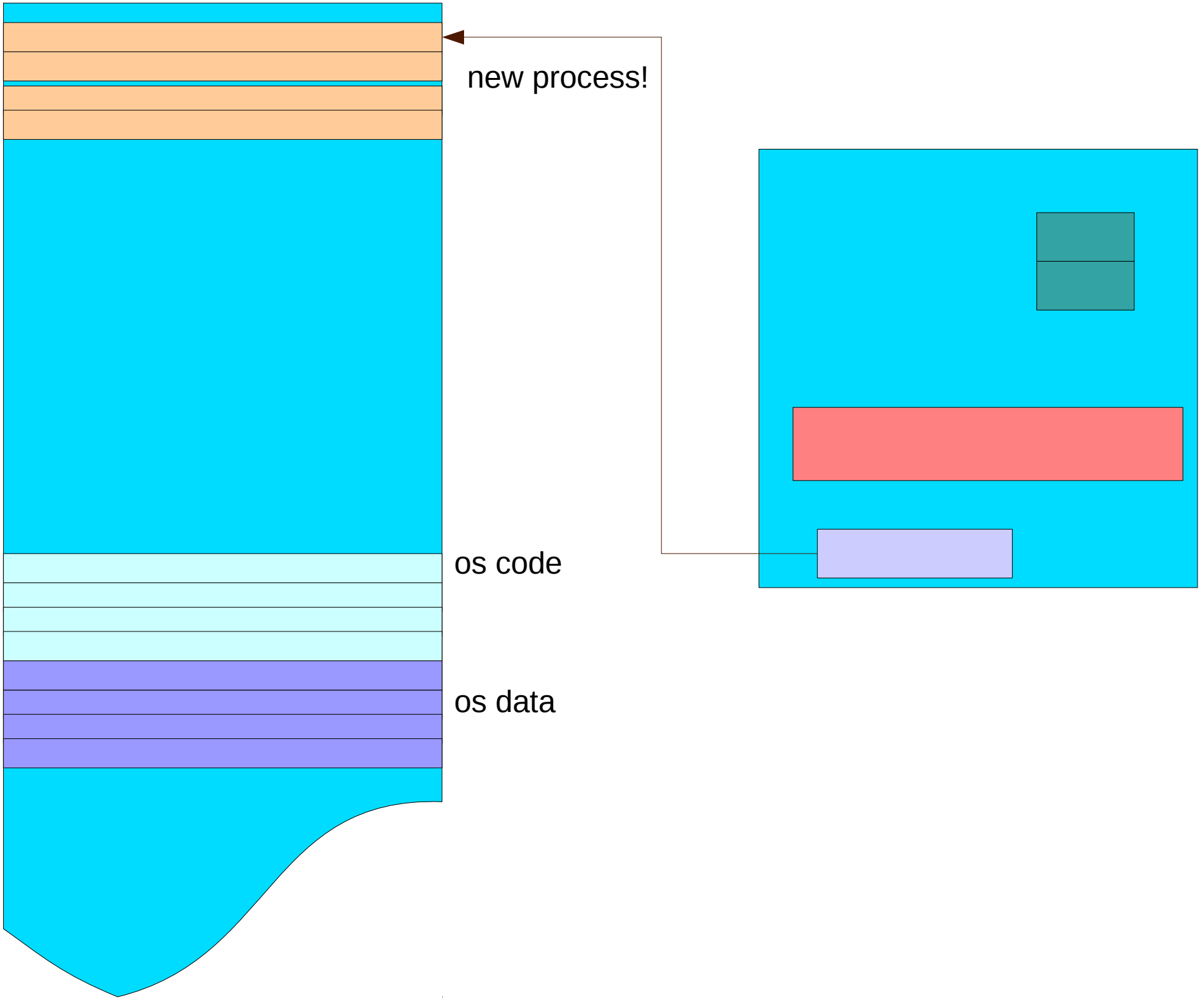

process2
process3
os code
os data


process2
process3
os code
os data

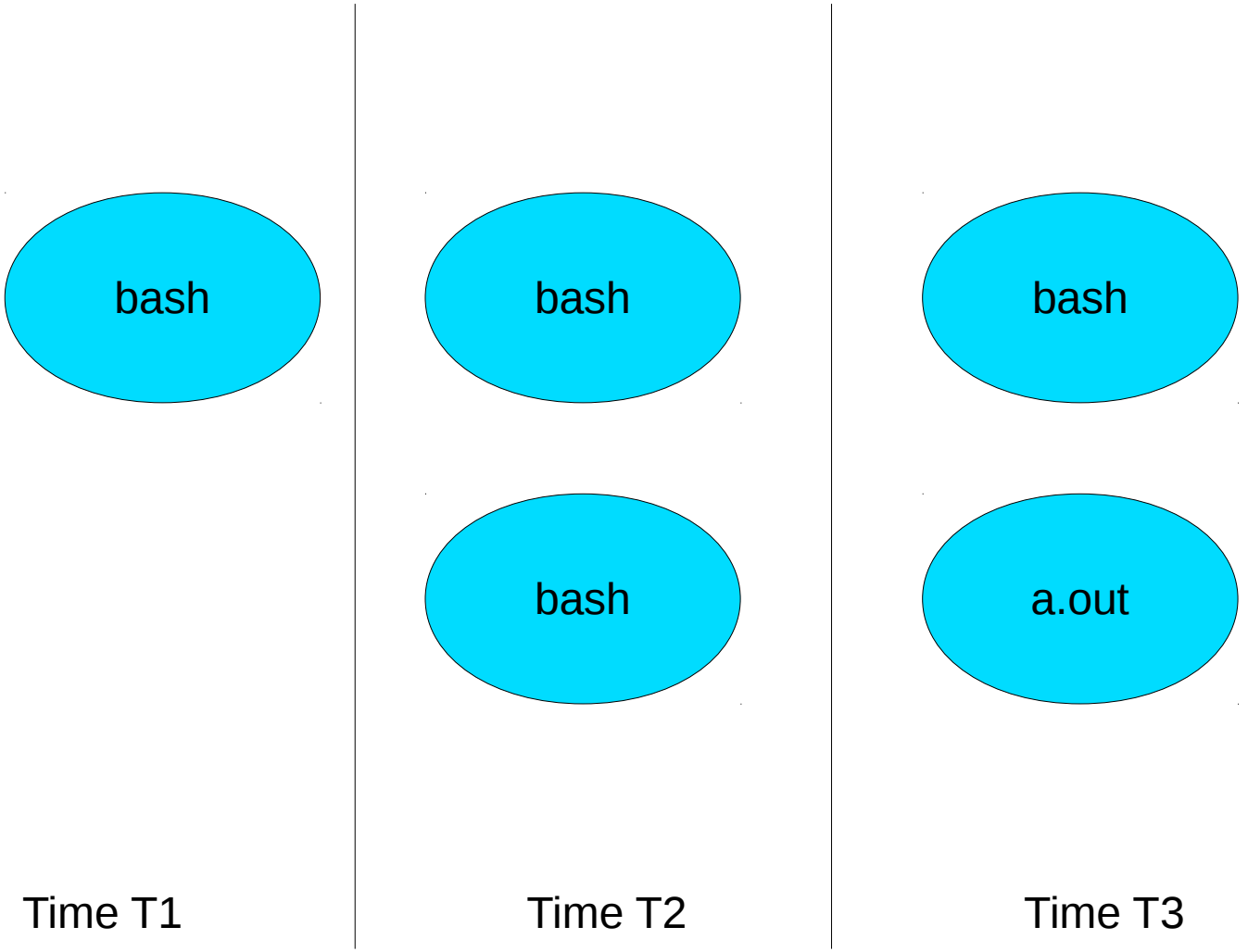

process2
os code
os data


process2
os code
os data


os code
os data


new process!
os code
os data

How is a new process created

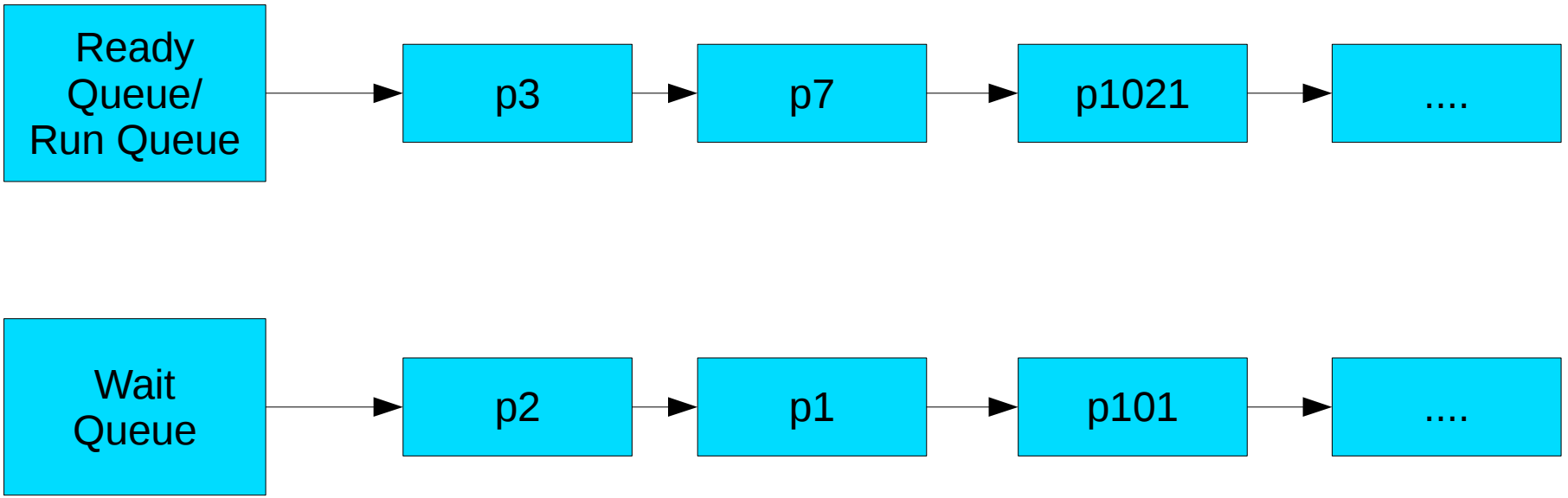


shell receives a command on command prompt: $./a.out

shell, a program, reads it and clones (forks) itself

the clone changes itself with the executable image residing inside './a.out'

Processes are organized into Queues..

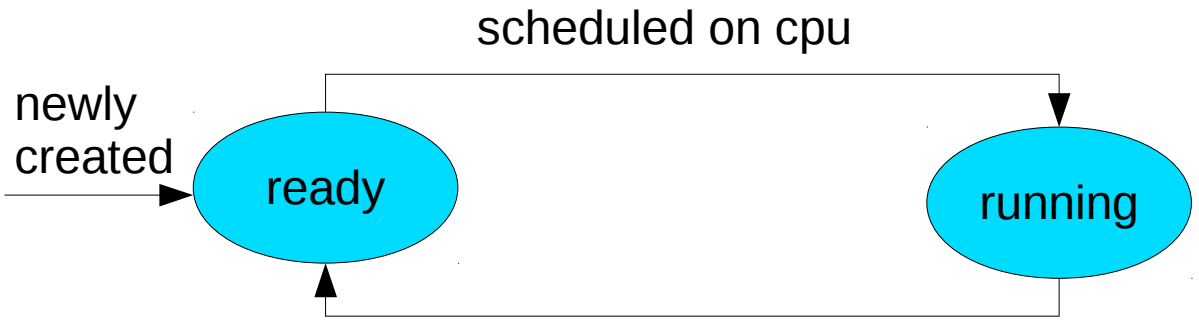


The cpu scheduler picks up context switches to a process from ready queue

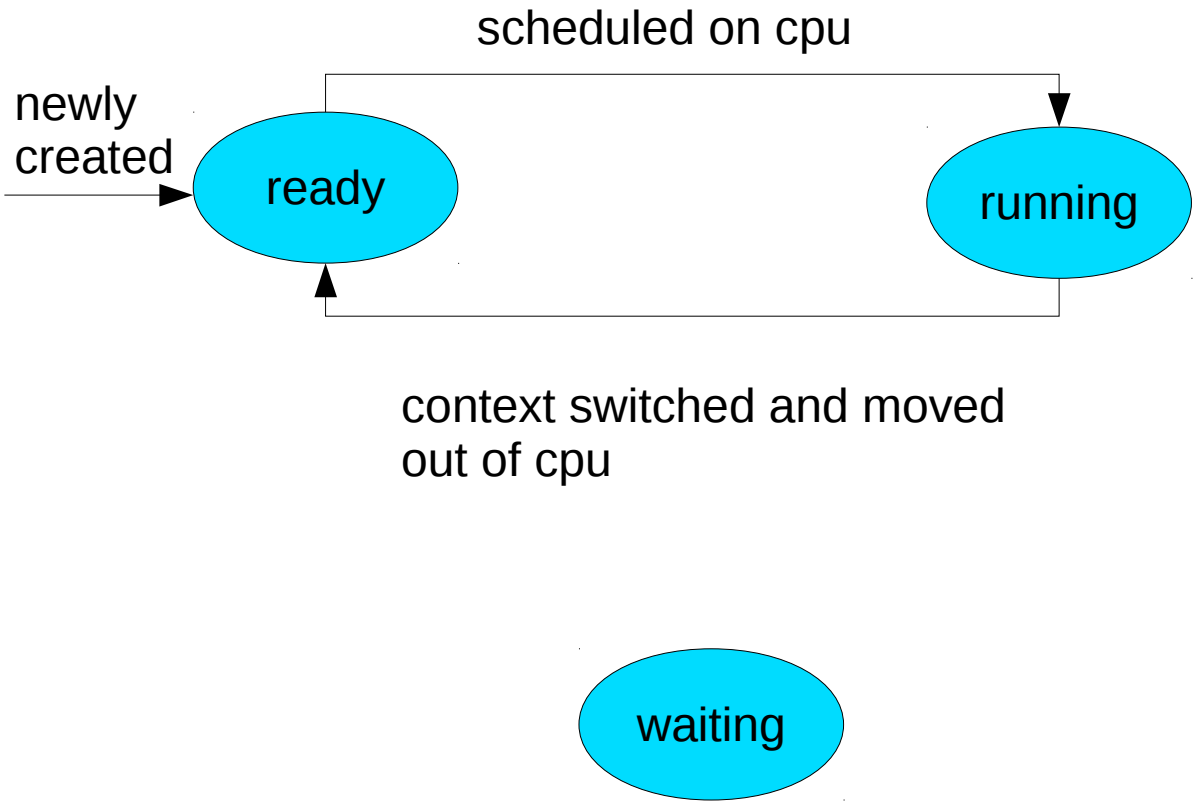
When a process waits for a device, it's moved into wait queue

When the corresponding device becomes ready, the process moves back to ready queue

A process undergoes state changes
So we have a State machine to represent the states and state transitions



A process undergoes state changes
So we have a State machine to represent the states and state transitions



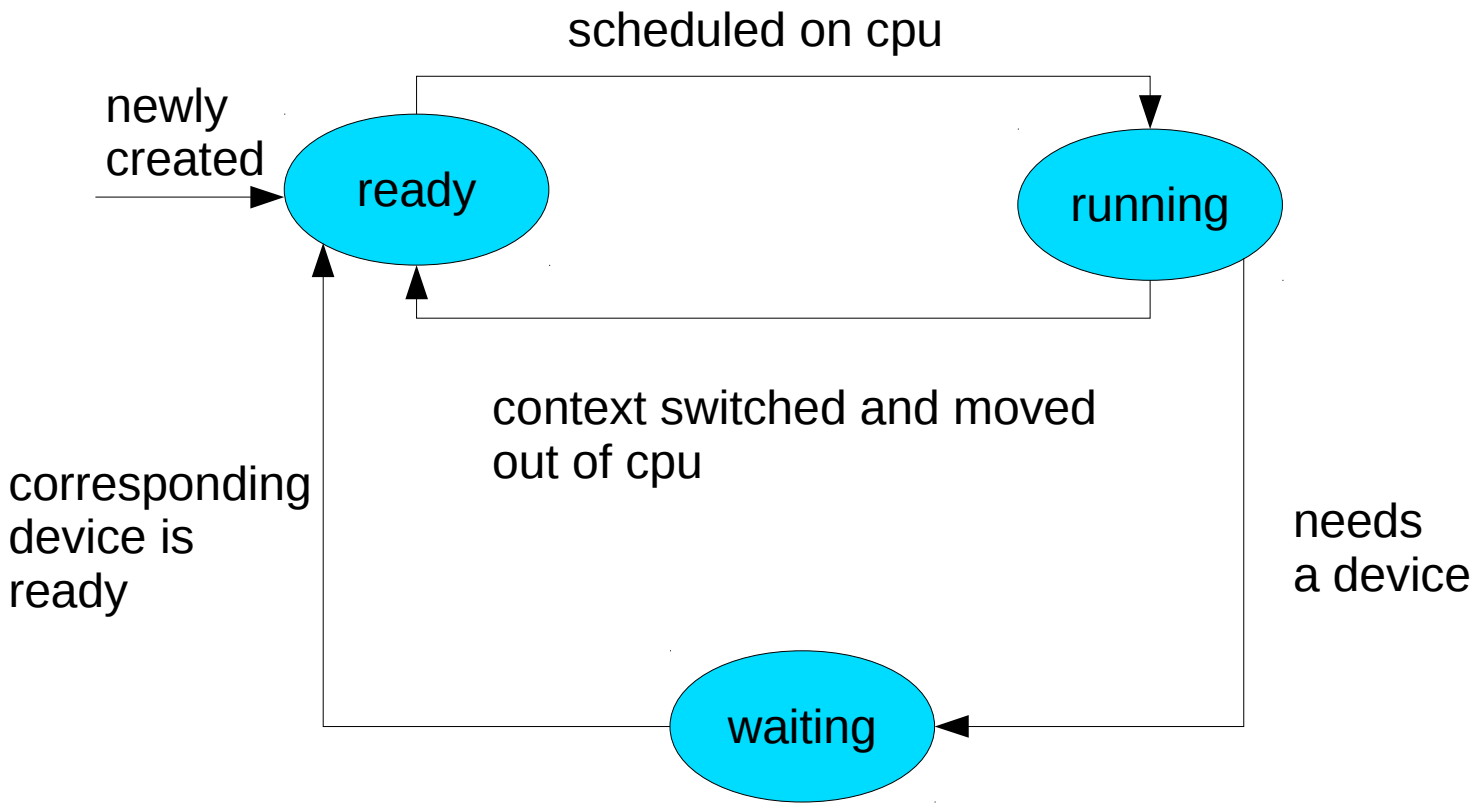
Compute Cycles and I/O cycles

```
a=a+1;

a=0;

while (c>0) { c=c-1;}

cin >> x;


fopen (“myfile”)
```

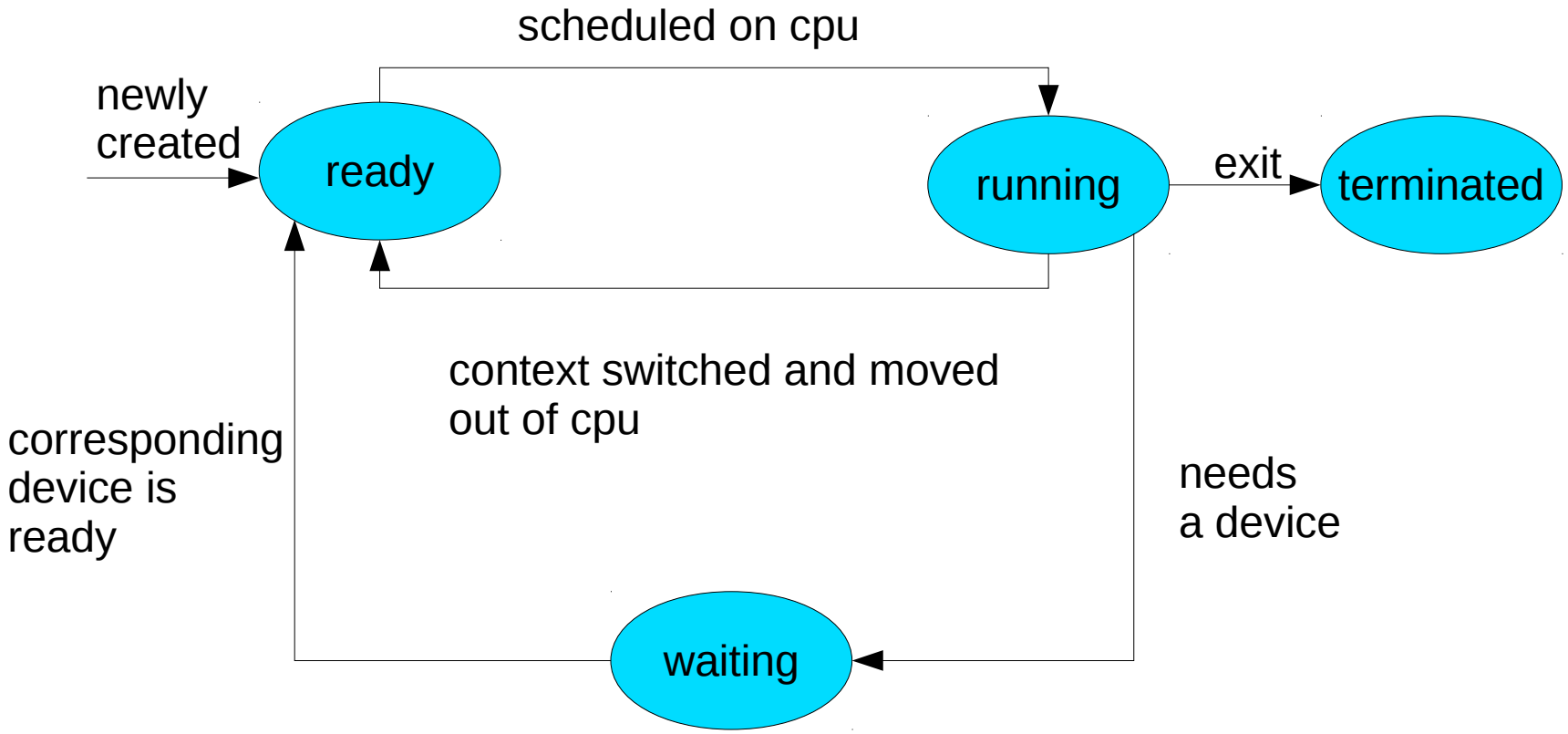
Computations on the cpu and involving main memory are very fast as compared to interactions with slow devices, and devices that involve human interaction

If the process is kept 'running' during this time, CPU is wasted. and other users/other programs feel the slow response from cpu

A process undergoes state changes
So we have a State machine to represent the states and state transitions



A process undergoes state changes
So we have a State machine to represent the states and state transitions



There are other possible states which we will explore eventually

CPU Scheduling Concerns

Sizes of the tasks
- small sized task: should they keep waiting?
- large tasks: should they always give in to small tasks?

Fairness of scheduling:
- should some types of tasks suffer always due to priority user tasks?

response time
- how long will it take to schedule a process?

cpu utilization
- are cpu cycles wasted?

scheduling efficiency
- is scheduling algorithm itself taking a lot of time?