Conditions that make dead lock possible

1. mutual exclusion
2. lock & wait
3. non-preemption
4. circular wait

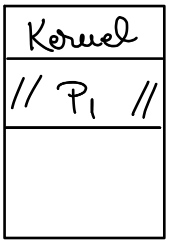
ALWAYS LOCK THE RESOURCES IN THE SAME ORDER

↑ to prevent deadlock

Memory management

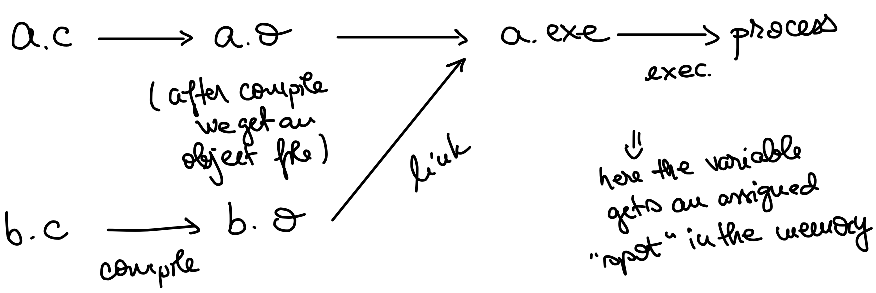
- real → single tasking OS Ⓐ
  → multitasking OS
  - fixed partition
    - * absolute Ⓑ
    - * relocatable Ⓒ
  - variable partitions Ⓓ
- virtual → paged Ⓔ
  → segmented Ⓕ
  → paged - segmented Ⓖ

A. Single tasking OS

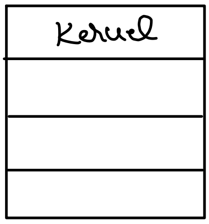


-the compiler hard codes physical memory addresses in the executable

Address calculation :

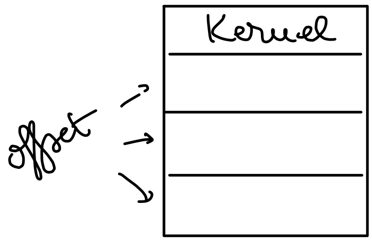

a.c
a.o
(after compile we get an object file)
a.exe
exec.
process
b.c
compile
b.o
link
⇓ here the variable gets an assigned "spot" in the memory

B.



→ split into memory portions
→ compile into a partition

C. (change compiler and improve address computation)



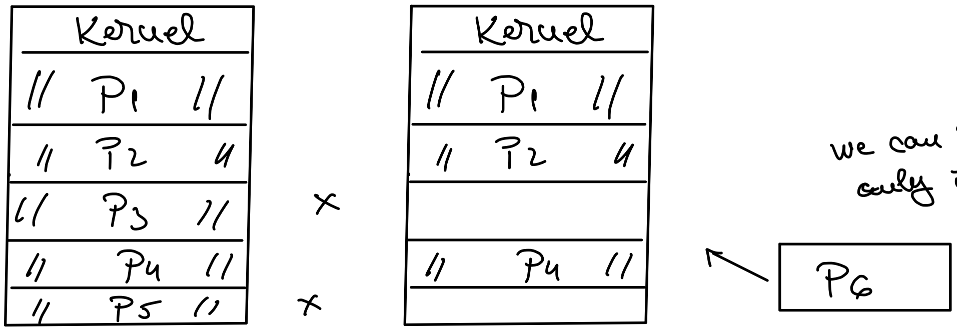
→ no hardcoding of physical addresses

→ every partition has an offset

→ relative addresses (relative to the beginning of the partition)

→ the program can run in any partition (that is free)

D. partitions are not predefined, we define them based on the size of the program that has to start

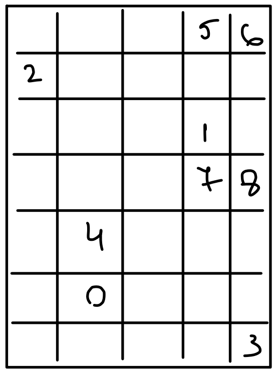


we can run P6 only if we have the continuous memory necessary

We have the necessary memory but it is SPLIT ⇒ fragmentation

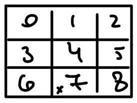
E.

RAM



↑ real pages (fixed size of a page)

Prog



↑ virtual pages

→ when the process stops, the real pages are freed

→ fragmentation solved with more fragmentation

→ more complex address calculation
↓
needs a search

⇒ we need a table for each virtual page position in the real pages ⇒ slow
BUT the processor help with hardware

F. segments do not solve fragmentation, segments only group sections to be protected
↓
they do not have a fixed size

G. TOP but waste a bit more memory than just paged

Loading policies:

→ when should pages be loaded?

- load all of them at process start
  (slower start and wasted RAM)
- load when needed
  (even slower start? and slower execution)
- load the requested page and a few neighbouring pages
  (chances are they will be needed)

Unloading policies:

RAM

	0		
3			1
	2		

Prog:

0	1
2	3