Today we will talk about:
- page fault handling
- page replacement

1) Page fault

On Linux, the things done on a page fault are:
- error: different errors depending on their number might call different hooks
- allocate: the next thing we can do is fig 2
- page in
- copy on write

**Fig 1:**

```plaintext
mm->fault_handler():

1. init/alloc, pde + pde = alloc -> pde(addr)
2. page table, pte + pte = alloc -> pte(page)
3. for the hardware view, make_pte(page)
```

- copy on write: it is the next one in term of difficulty

**Fig 2:**

- different copy of the same page accessed by different processes
- to the process P2 want to write on the copies, it will write on the copies that are not write protected only

So the fault will happen if we have a copy on write then:

If RW
- new page = alloc
- copy & make_pte(new, RW)
2) Page Replacement

The OS will have to be adaptable, being able to upgrade its information in case of memory increase. The OS goal is also to use the memory address at its full advantage, with the help of the LRU (Least Recently Used) algorithm.

If you have no other choice the simplest replacement policy is random (used by VMware). The next one is FIFO. These 2 methods are equivalents, because randomness is related to the number of addresses present at a given time. The 2 others are LRU and Belady’s OPT.
LRU works fine for disk buffer performance increase with memory access. FIFO became easier to implement.

Belly's OPI is an offline algorithm (opposite to an online one) which

chooses algorithm.

It goes through tables the first time checks those that are 0, then when it comes back, the ones equal to 0 are the most probable to be discarded.