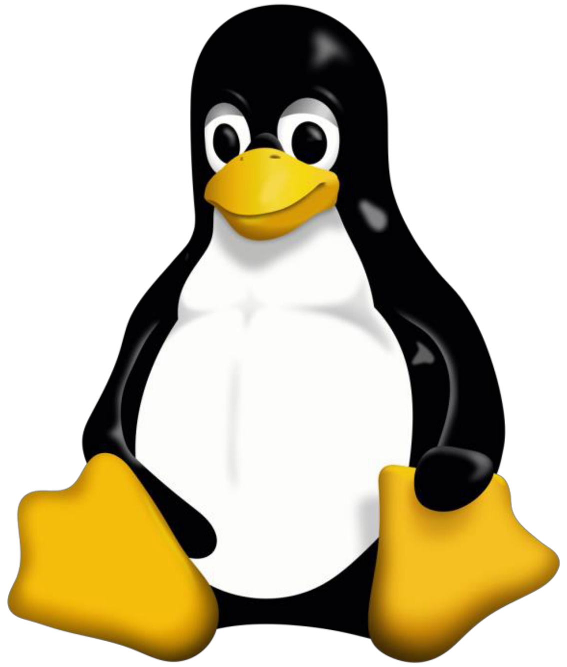


Linux Character Device Driver

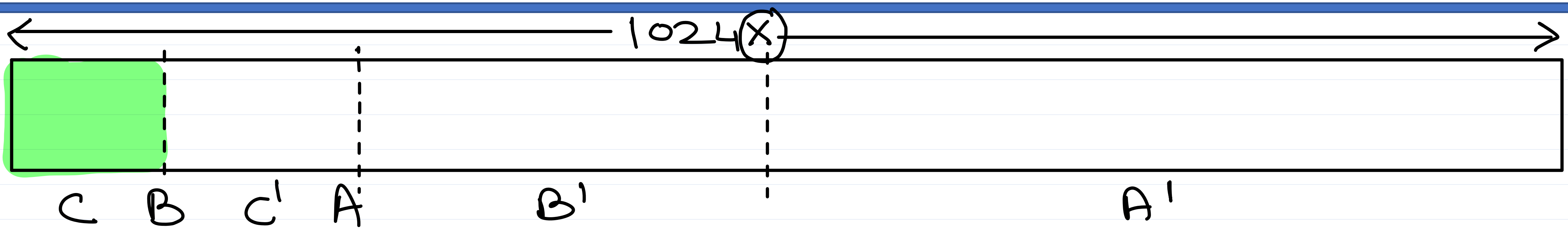
Sunbeam Infotech

Linux mem mgmt

① Paging
- ↳ MMU - TLB hw
- ↳ page table - multi-level paging
  - - 3 level paging

② page fault handling
- ↳ CPU req logical page - but page not in RAM.
  - ↓ Page fault ex.
  - OS page fault handler ← Page fault ex.
  - ↳ ✓ check if addr is valid & perms are valid.
    - ✓ allocate empty frame → ?
      - - if not empty frame, page replacement.
    - ✓ if page on disk/swap, load it in RAM (in the frame)
    - ✓ update page table & restart instn.

Buddy allocator



alloc req

① 128 → C

$1024 \to \cancel{x}$

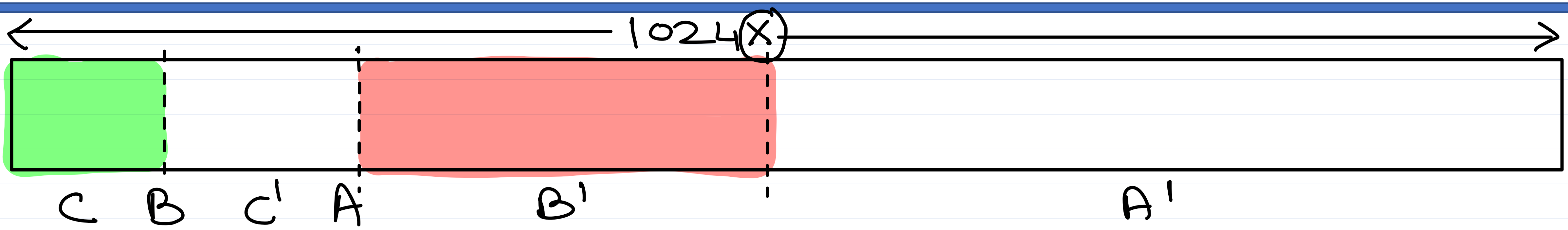
$512 \to A', \cancel{A}$

$256 \to B', \cancel{B}$

$128 \to C', \cancel{C}$

$64 \to$

Buddy allocator



alloc req

① 128 → C

② 256 → B'

$1024 \rightarrow \cancel{x}$

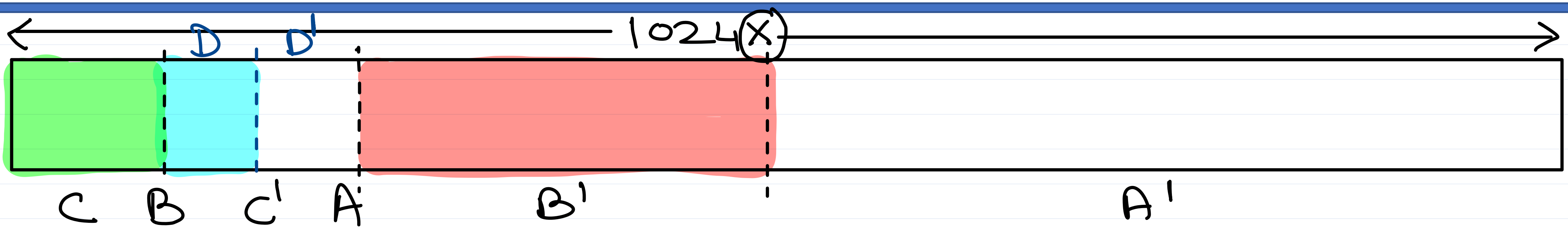
$512 \rightarrow A', \cancel{A}$

$256 \rightarrow \cancel{B'}, \cancel{B}$

$128 \rightarrow C', \cancel{C}$

$64 \rightarrow$

Buddy allocator



alloc req

1. 128 → C
2. 256 → B'
3. 64 → D

1024 → ~~X~~

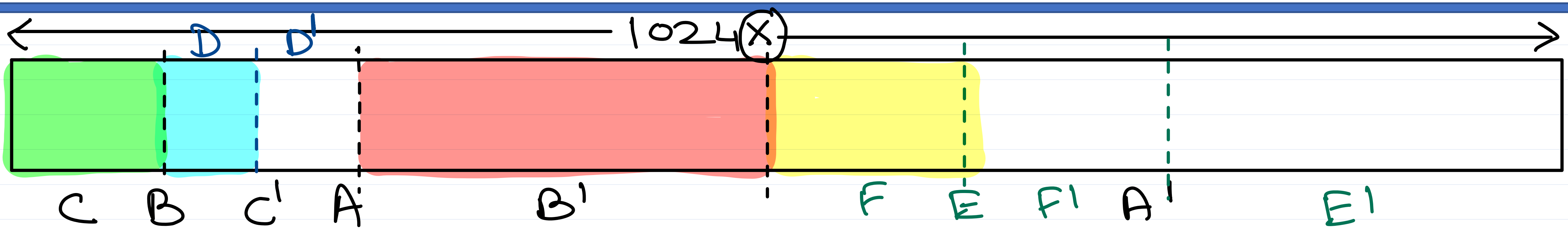
512 → A', ~~A~~

256 → ~~B'~~, ~~B~~

128 → ~~C'~~, ~~C~~

64 → D', ~~D~~

Buddy allocator



alloc req

1	128	→	C
2	256	→	B'
3	64	→	D
4	128	→	F

1024 → ~~X~~

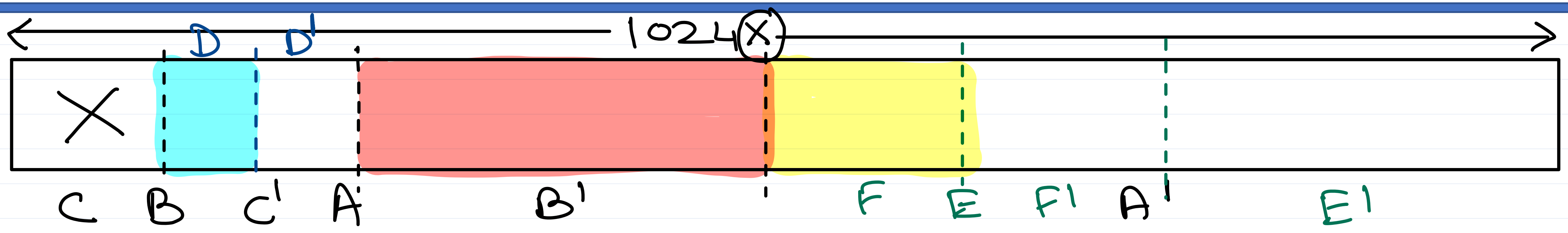
512 → ~~A'~~, ~~A~~

256 → ~~B~~, ~~B~~ E', ~~E~~

128 → ~~C'~~, ~~C~~ F', ~~F~~

64 → D', ~~D~~

Buddy allocator



alloc req

1. 128 → C
2. 256 → B'
3. 64 → D
4. 128 → F

dealloc req

1. C ~~X~~

1024 → ~~X~~

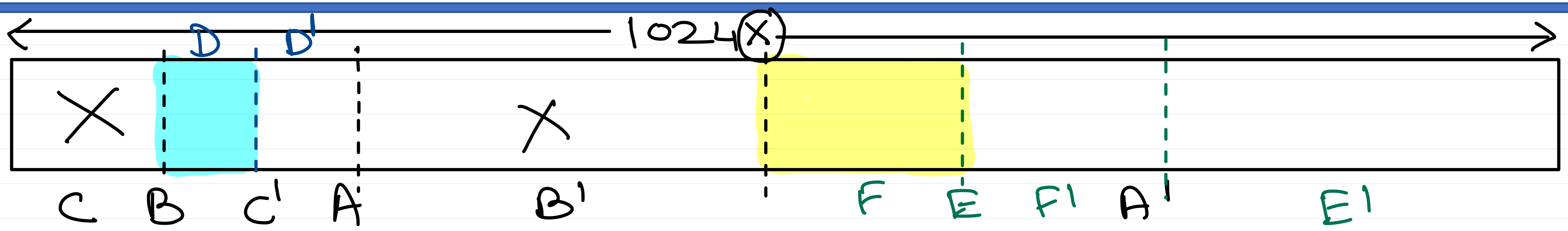
512 → ~~A'~~ , ~~A~~

256 → ~~B~~ , ~~B~~ E' , ~~E~~

128 → ~~C'~~ , ~~C~~ F' , ~~F~~ , C

64 → D' , ~~D~~

Buddy allocator



alloc req

① 128 → C

② 256 → B'

③ 64 → D

④ 128 → F

dealloc req

① C ✗

② B' ✗

1024 → ~~X~~

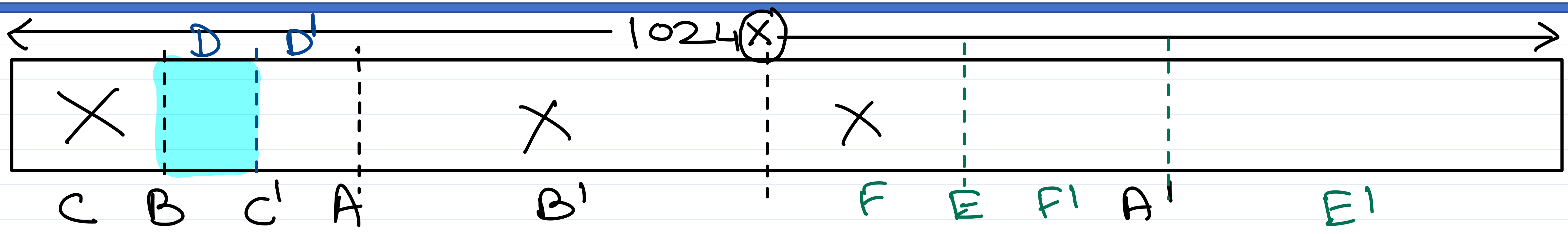
512 → ~~A'~~, ~~A~~

256 → ~~B~~, ~~B~~, E', ~~E~~, B'

128 → ~~C'~~, ~~C~~, F', ~~F~~, C

64 → D', ~~D~~

Buddy allocator

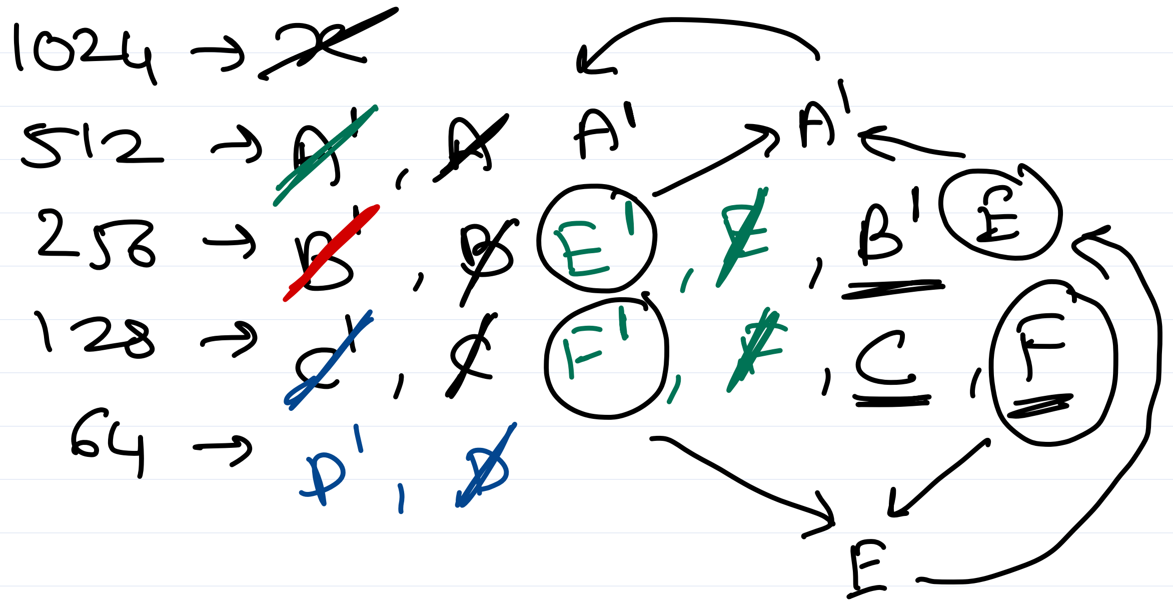


alloc req

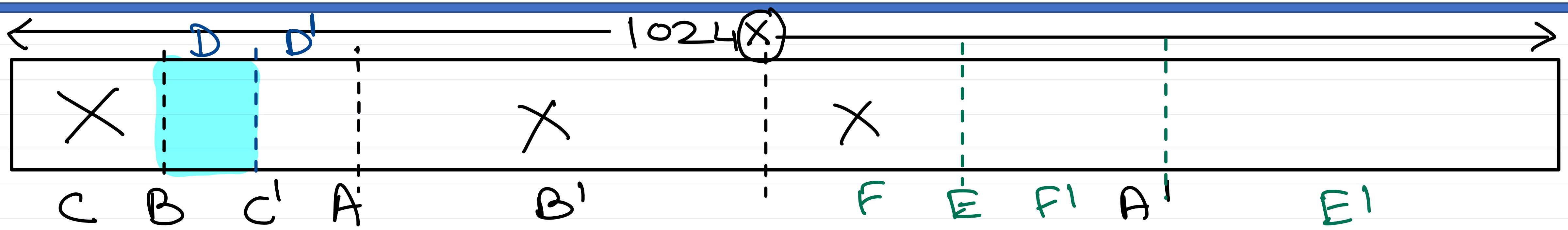
1. 128 → C
2. 256 → B'
3. 64 → D
4. 128 → F

dealloc req

1. C ✗
2. B' ✗
3. F ✗


1024 → ~~X~~
512 → ~~A'~~, ~~A~~ A' A'
256 → ~~B~~, ~~B~~ E', ~~E'~~, B' E
128 → ~~C'~~, ~~C~~ F', ~~F'~~, C, F
64 → D', ~~D~~
E

Buddy allocator



alloc req

① 128 → C

② 256 → B'

③ 64 → D

④ 128 → F

dealloc req

① C ✗

② B' ✗

③ F ✗

④ D ✗

↳ home work

1024 → ~~X~~

512 → ~~A'~~, ~~A~~ A' A'

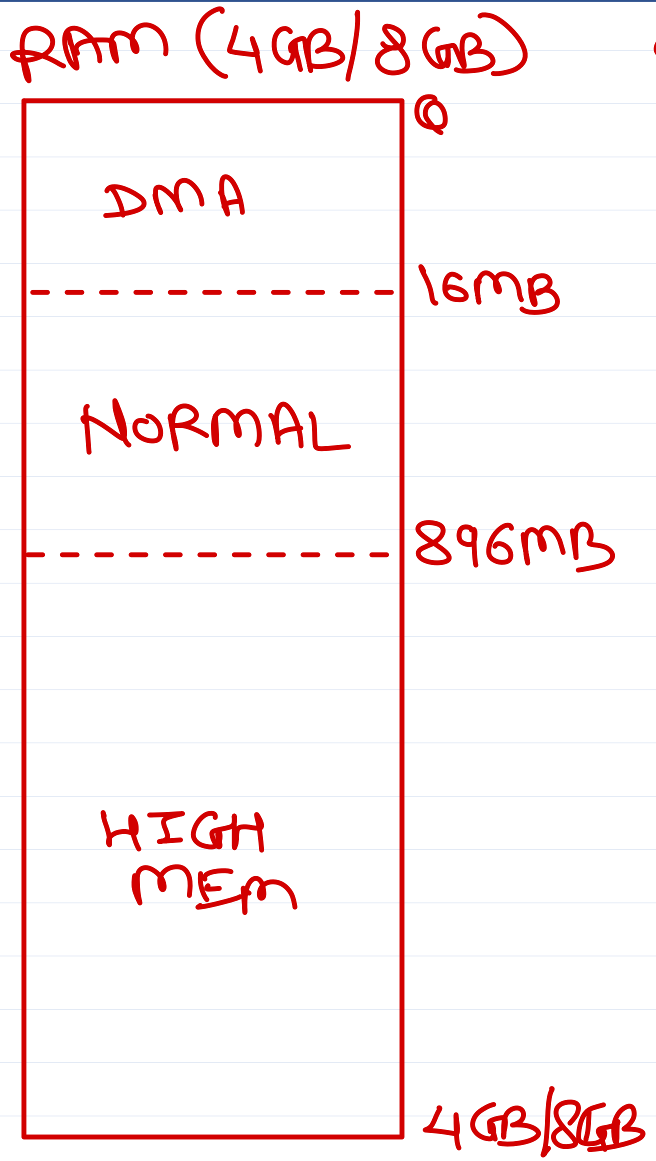
256 → ~~B~~, ~~B~~ E', ~~E~~, B' E

128 → ~~C~~, ~~C~~ F', ~~F~~, C, F

64 → D', ~~D~~

E

Buddy allocator in Linux



on x86-32 arch → Three Zones

① DMA → 0 - 16 MB → Buddy1

② NORMAL → 16MB - 896 MB → Buddy2

③ HIGHMEM → 896 MB and above → Buddy3

```
cmd> cat /proc/buddyinfo
```

$2^0 \rightarrow$

$2^1 \rightarrow$

$2^2 \rightarrow$

$2^3 \rightarrow$

$\vdots$

$2^{10} \rightarrow$

to allocate mem from buddy allocator:

① get_free_pages(mask, order);

- base addr.
- mask: allocator behaviour
- order: num of consecutive pages - power 2

② free_pages(addr);

Linux Buddy allocator

cat /proc/buddyinfo

x86_32

addr = __get_free_pages(mask, order);

free_pages(addr, order);

0 →	1 Page
1 →	2 Pages
2 →	4 "
3 →	8 "
4 →	16 "
⋮	
10 →	1024 "

Buddy Alloc
free list

$2^{10} \rightarrow$

$2^{9} \rightarrow$

$2^{8} \rightarrow$

$2^{7} \rightarrow$

⋮

$2^{3} \rightarrow$

$2^{2} \rightarrow$

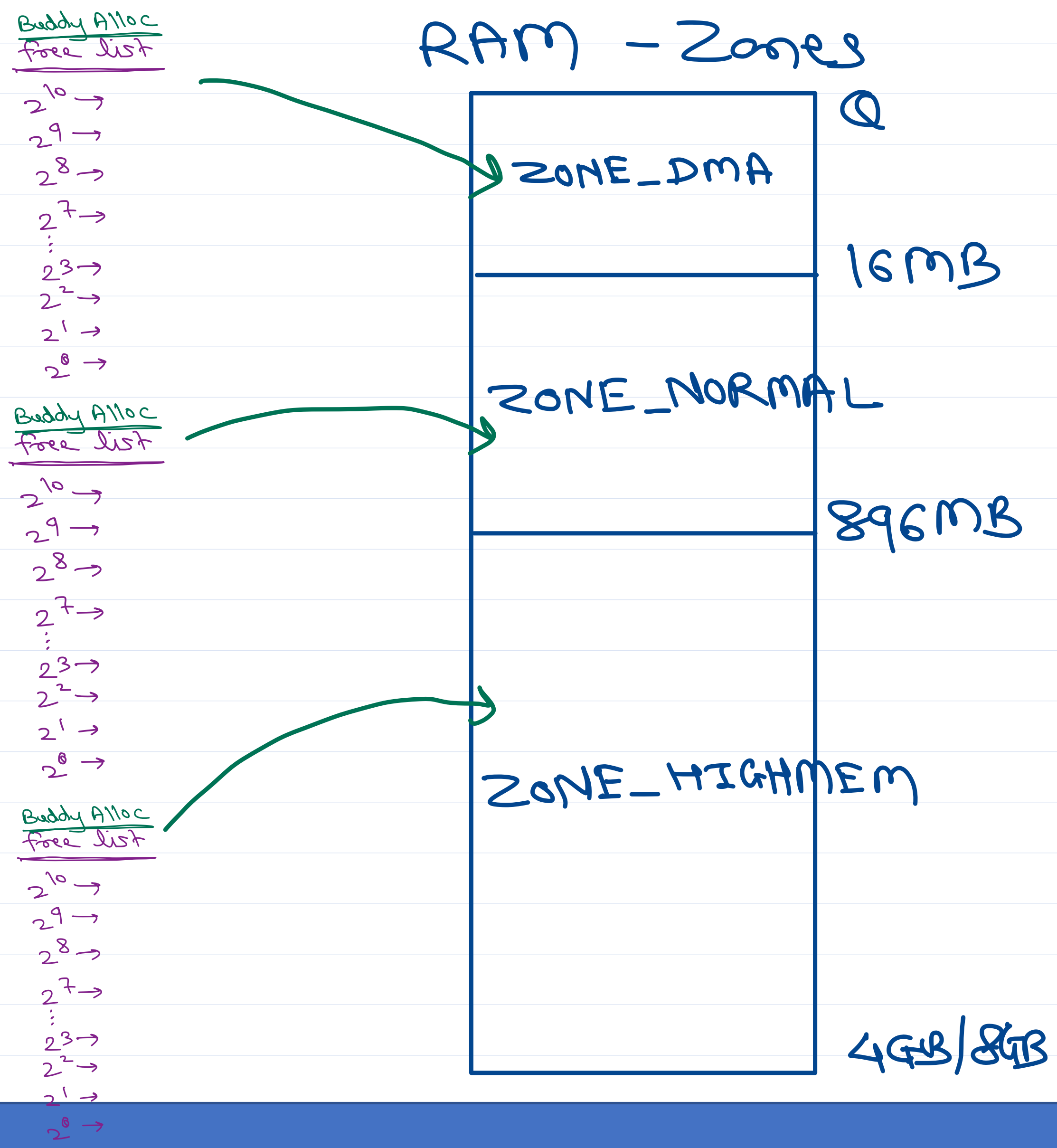
$2^{1} \rightarrow$

$2^{0} \rightarrow$

Combo of action mod + zone mod = Type Flag

↳ e.g. GFP_KERNEL = __GFP_WAIT + __GFP_IO + __GFP_FS

↳ e.g. GFP_ATOMIC = __GFP_HIGH



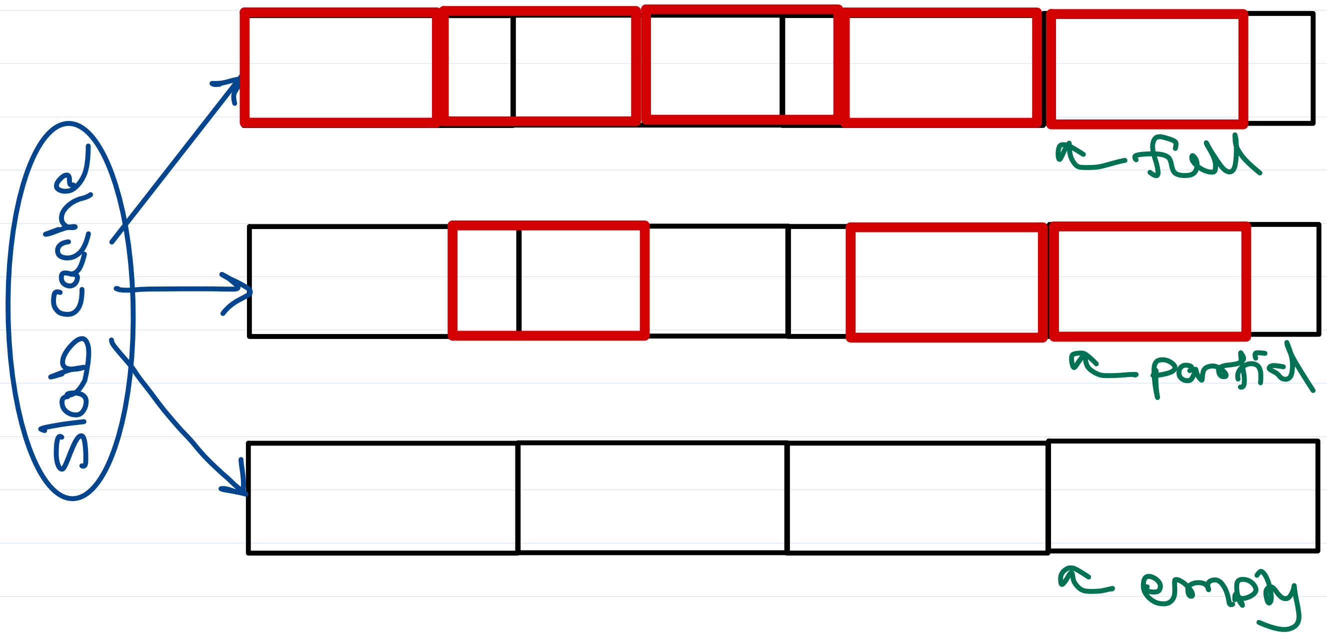
Slab allocator

cat /proc/slabinfo

3KB

```
ptr = kmalloc(size, mask);
        :                └→ GFP_XYZ
kfree(ptr);
```

- ✓ allocates smaller contiguous block from slab cache.
- ✓ lesser internal fragmentation.
- ✓ faster allocation for commonly required (well-known) objects.
  e.g. task_struct, mm_struct, inode, ...



Example:
- ✓ object size = 3KB
- ✓ pages per slab = 4
- ✓ objects per slab = 5
- ✓ slabs per cache = 3
- ✓ num of objs = 15
- ✓ active obj = 8

- ✓ A slab is set of contiguous pages.
- ✓ Slab allocator allocates slab - using buddy allocator.

vmalloc()

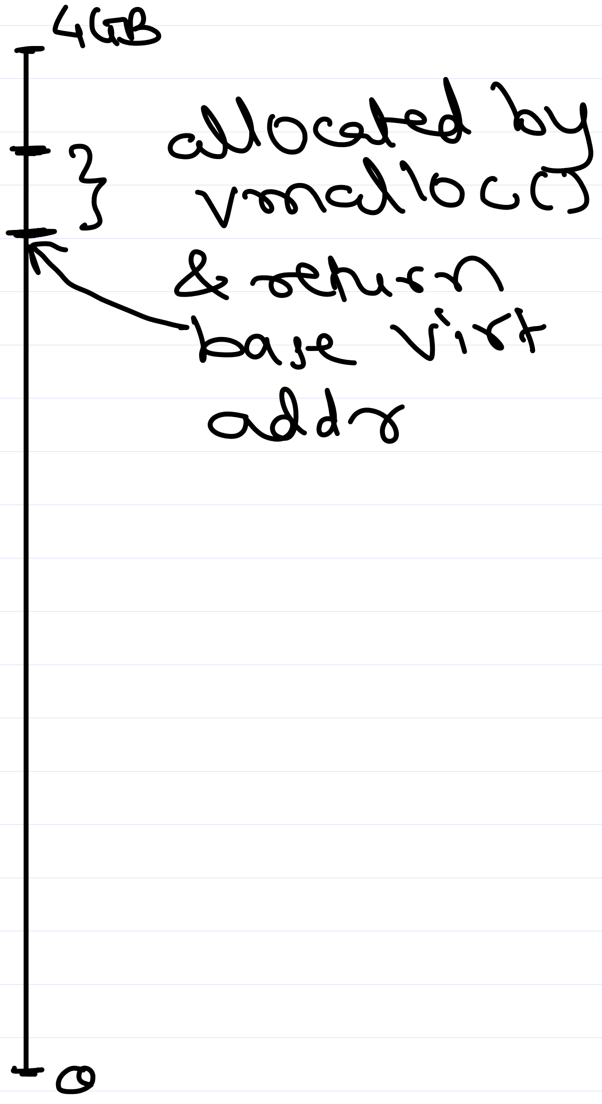
In user space, contiguous virtual addresses are allocated by mmap(). It internally calls vmalloc().

To allocate contiguous virtual memory in kernel modules/code, use vmalloc().

```
ptr = vmalloc(size);
  :
vfree(ptr);
```

✓ Allocates contiguous range of virtual address.
- ✓ allocates a new VAD (vm_area_struct).
- ✓ allocates corresponding page table entries.

✓ Actual physical pages will be allocated, when pages are accessed by CPU (read/write).



Page replacement

If RAM is full, existing page(s) needs to be swapped out - to make space available for new pages.

Local Replacement:

- To make space avail for process swap out a few pages of that process itself.

Global Replacement:

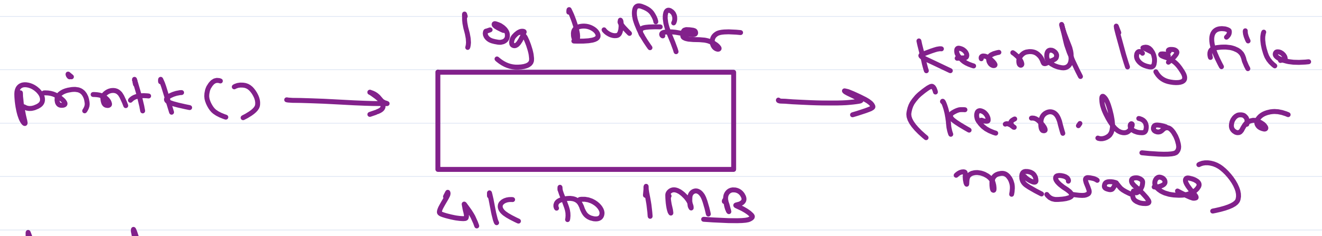
- To make space avail for process swap out a few pages of any process.

Kernel runs a daemon process which ensures that avail memory is not going below a threshold level. If needed, it will swap out pages from RAM (as per page replacement algo). This process is called as "page stealing process". In Linux Kernel, this is done by "kswapd" daemon.

Kernel debugging techniques

① debugging by printing

printk() → log levels
0 7
EMERG ... DEBUG



kernel log daemon → syslogd or klogd.

② debugging by querying

① ioctl() operation ② procfs entry.

③ debugging by watching

strace command → shows which syscalls are called

④ System faults/hangs

① Kernel OOPs message → register values + stack trace.

② Kernel panic - crash - scheduler stopped

⑤ Kernel debuggers

① gdb ② kdb

SUNBEAM

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>