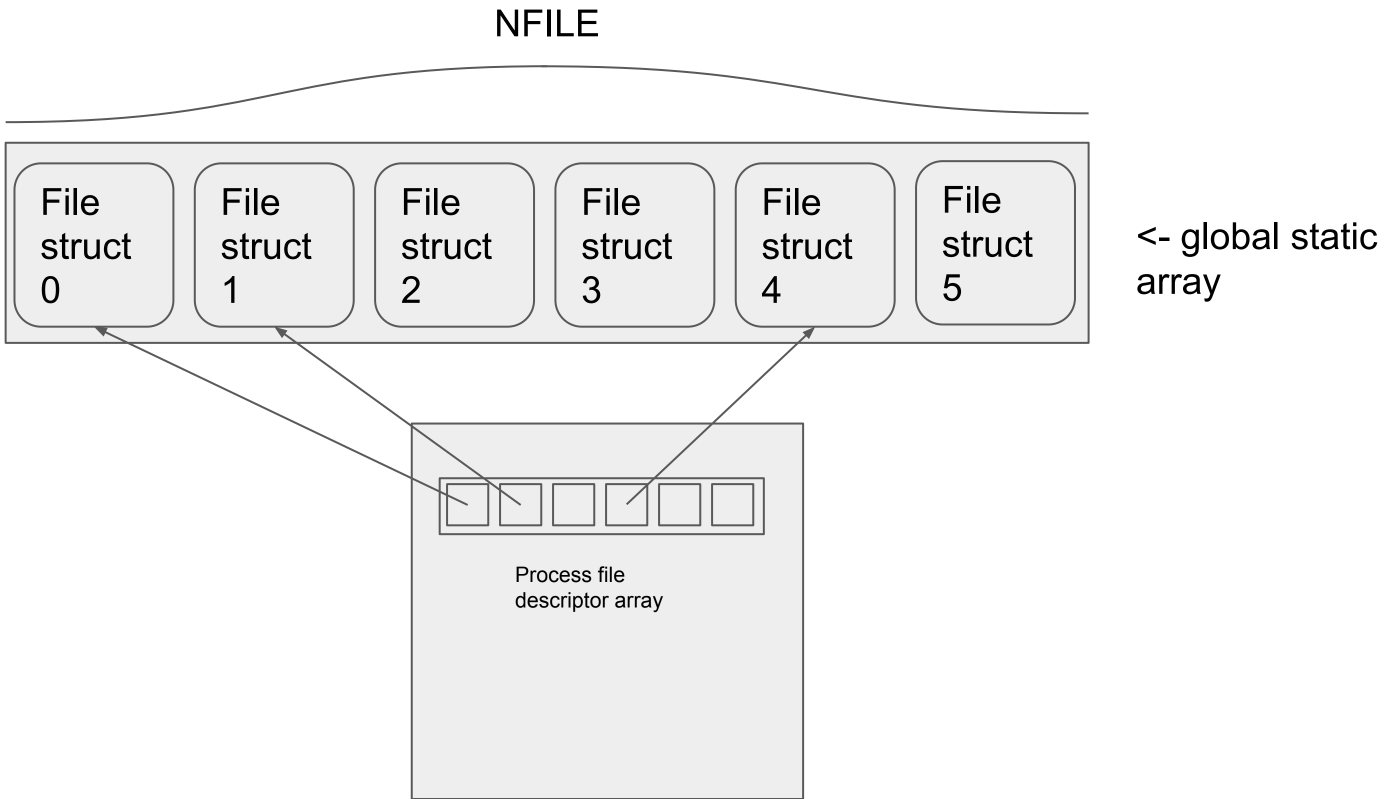
CSE 451

Lab 2: Alloc

How files are (currently) stored in xv6 [kernel/file.c]



What’s wrong with this?

- Fixed, max number of files for entire system (NFILE)
- Resource usage fixed (memory does not scale with file usage)
- Kind of boring

Solution?

Dynamically allocate memory for files at runtime!

Brief memory overview

- Kernel hands us virtual memory pages (kalloc)
- Large chunks of contiguous memory (4KiB)

- Using this instead of a static file table is a start, but still has issues
- Way too much space for just one file struct (32 bytes)
- Can be smart about it, but difficult and tedious to do over and over
    - Many kernel libraries that may want dynamically allocated memory

Slab allocator

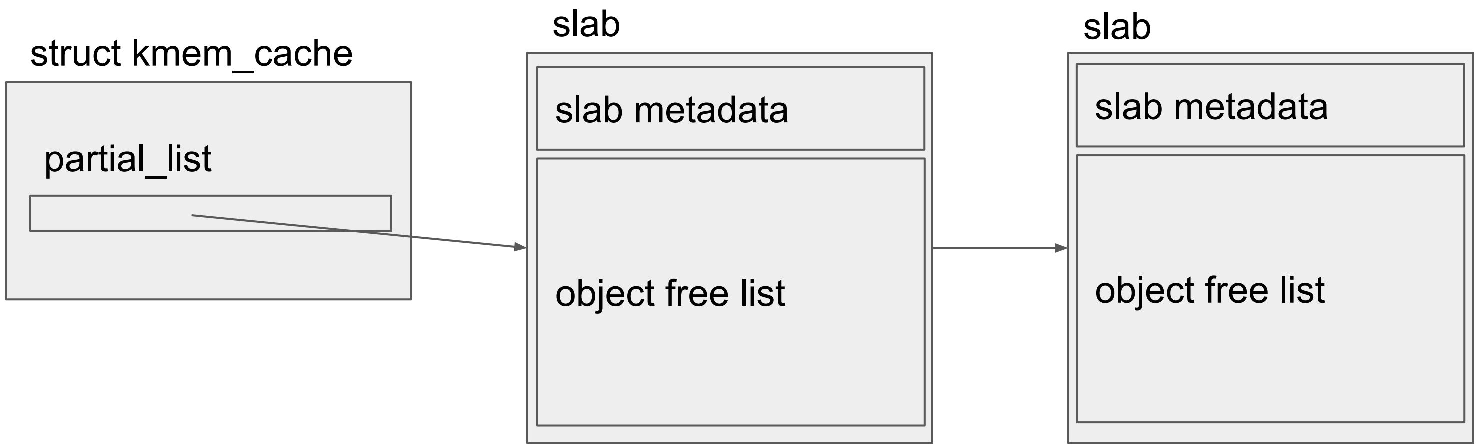
In essence:

- Create allocator struct that will hand out smaller objects instead of entire pages
- Ask kernel for entire pages (4096 bytes) by kalloc()ing
- Break pages up into smaller objects
- Allocate “objects” when asked for them
    - For the purpose of this lab, can all be same size + only need to be bigger than a file struct
- Keep track of allocated pages and objects and free them when necessary

Slub allocator

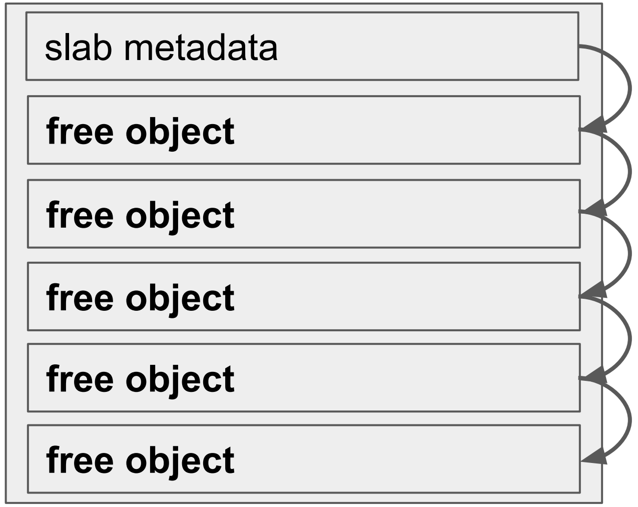
- Current implementation of slab allocator in linux kernel
  - Third link on lab page
- Gives a brief description of the high level implementation details
- Recommend that you follow this
- Take struct definitions with a grain of salt
  - Overall design is pretty similar to what you might want to implement, but the nuances are up to you

Slub allocator

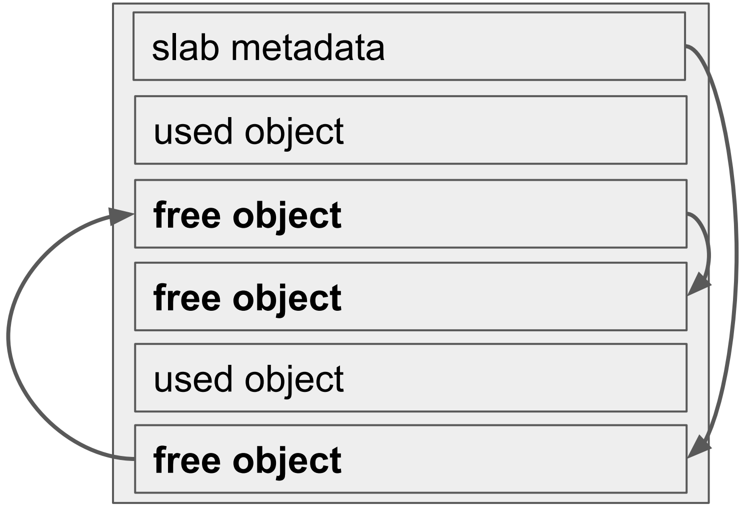


This is not a comprehensive design! You may require other metadata in the allocator, slabs, objects, etc.

Zooming in on a slab



When newly allocated



After some allocation/deallocation

(In terms of the bare minimum for slab metadata, you'll probably need a pointer to the object free list & a way of keeping track of how many objects are in use)

Slub allocator rules

- If a slab is filled (all objects are in use), remove it from the cache partial-list
    - Should go back in the list if it has an opening
    - How to recover entire slab when an object becomes free again?
- If a slab is empty (all objects are freed), entire slab should be freed
    - Use kfree (a slab is just a page, for this lab)

Functions/macros/resources you may find useful when implementing:

- PGROUNDDOWN()/PGROUNDUP()/PGSIZE (riscv.h)
- List operations (list.c/list.h)

GDB Demo

Good luck!

Any questions?