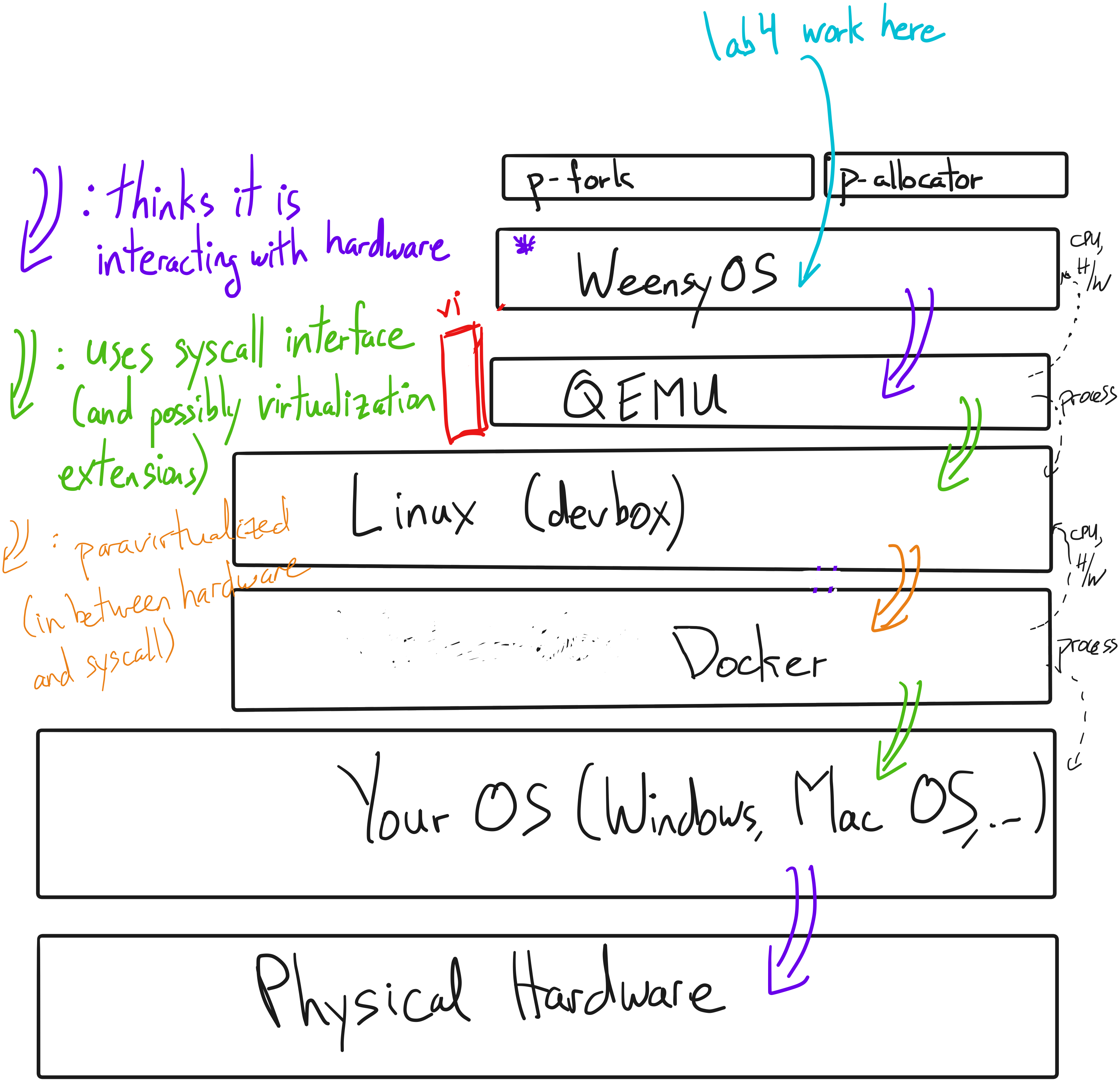
- [x] 1. Last time
- [x] 2. Weensy OS
- [x] 3. gdb
- [x] 4. mmap()



Hints:

- processes: files matching p-*.c
- kernel code: files matching k-*.c , k-*.s
- system calls and returns || (): /* cousin of mmap() */

sys_page_alloc();

look in process.h

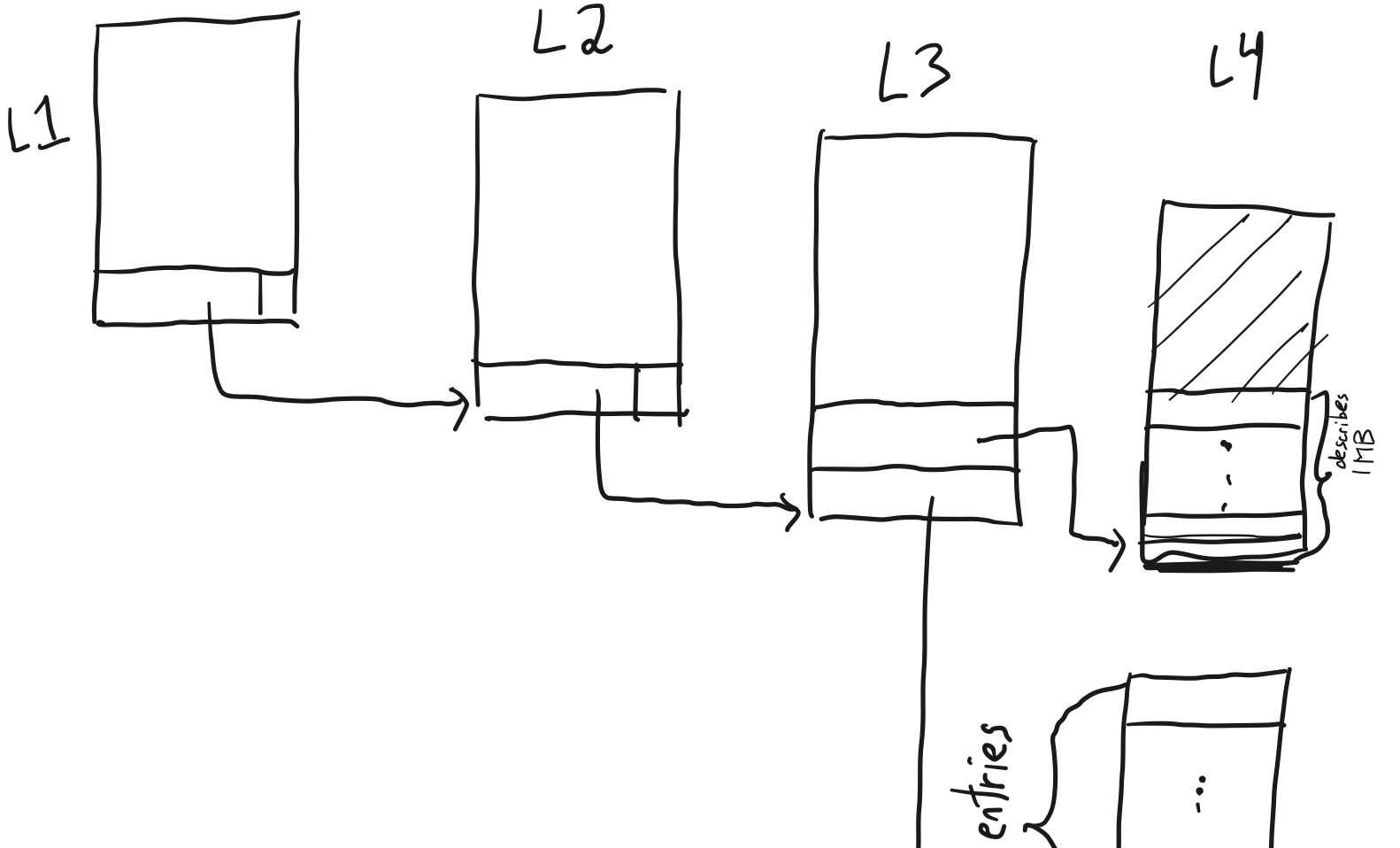
kernel returns: exception_return();
%rax contains return value of system call

%rdi: arg to syscall

<VPN, PPN>

- you'll use virtual_memory_map() =
  - pay attention to the allocator argument (NULL vs. non-NULL)
  - make sure your allocator initializes the page table
    memset(addr, 0, len);

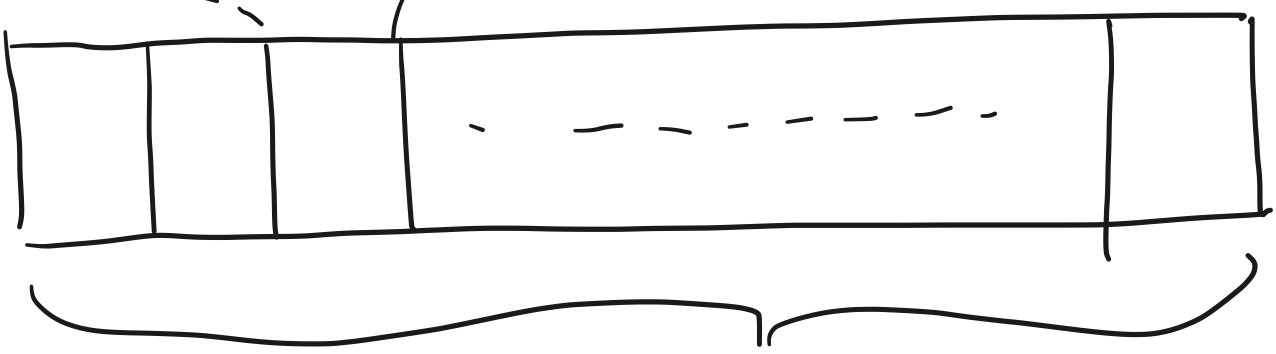
- a process's virtual address space: 3 MB. What's the page table structure?



each entry
describes a mapping
for one page
(4KB)

PCB ≡ struct proc (in kernel.h)

```
struct physical_pageinfo {
    int8_t owner;
    int8_t refcount;
}
```

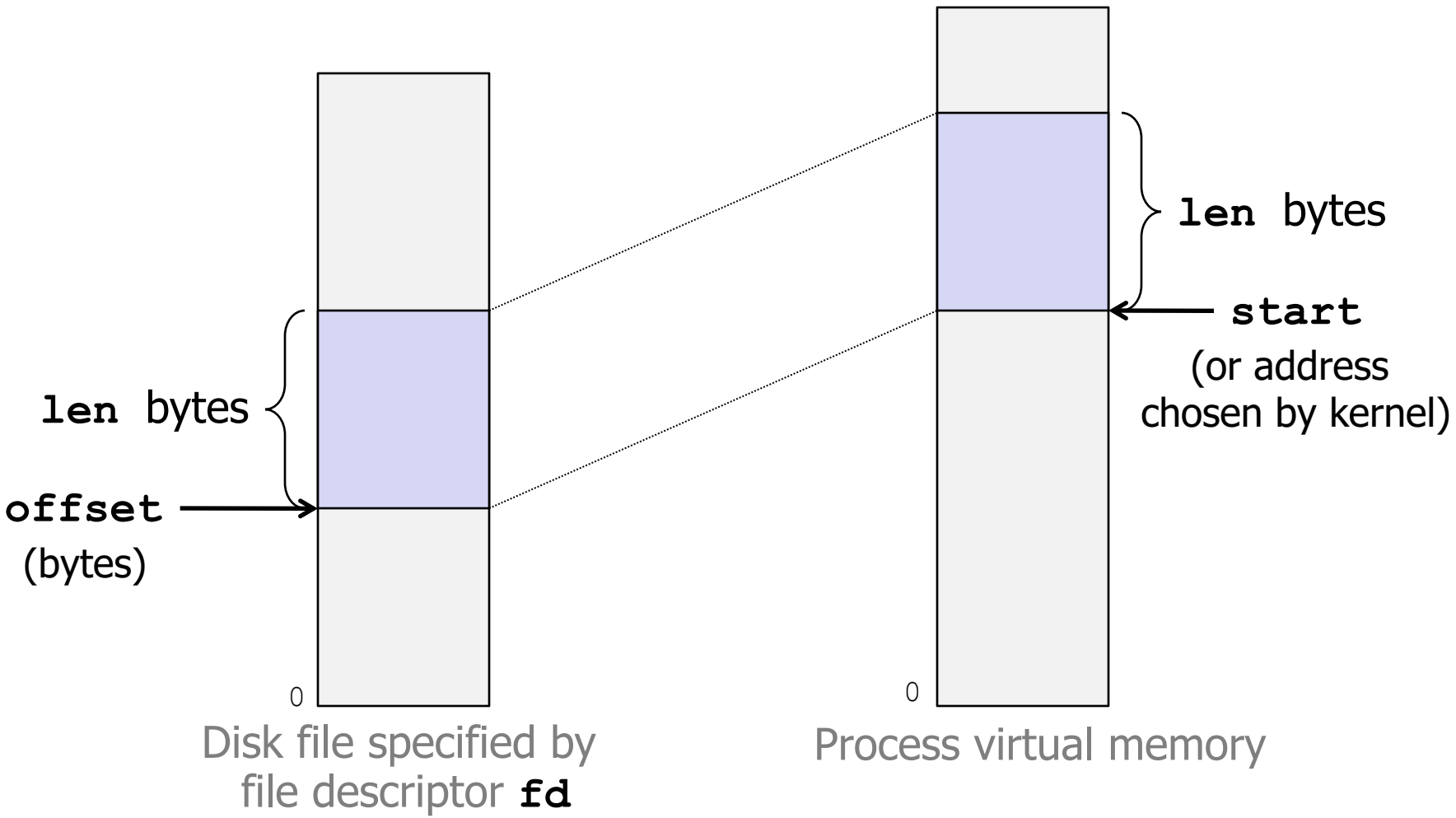


one per physical page

this is not a page table; it is bookkeeping.

User-Level Memory Mapping

```
void *mmap(void *start, int len,
           int prot, int flags, int fd, int offset)
```



```c
/*
 * Takes a filename as input, and writes the contents of the file
 * to stdout.
 *
 * Uses mmap()
 */

#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

void mmapcopy(int fd, int size);

int main(int argc, char **argv) {
  struct stat stat;
  int fd;

  /* Check for required cmd line arg */
  if (argc != 2) {
    printf("usage: %s <filename>\n", argv[0]);
    exit(0);
  }

  /* Copy input file to stdout */
  if ((fd = open(argv[1], O_RDONLY, 0)) < 0)
    perror("open");

  fstat(fd, &stat);
  mmapcopy(fd, stat.st_size);

  close(fd);

  return 0;
}

void mmapcopy(int fd, int size) {

    /* Ptr to memory mapped area */
    char *bufp;

    bufp = mmap(NULL, size, PROT_READ, MAP_PRIVATE, fd, 0);

    /* first argument is 1 */
    write(STDOUT_FILENO, bufp, size);

    return;
}

```



```c
/*
 * Takes a filename as input, and writes the contents of the file
 * to stdout.
 *
 * This is a naive version, and should be contrasted with the version
 * that uses mmap().
 */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

int main(int argc, char **argv) {
  int fd;
  int rc;
  char buf[256];

  /* Check for required cmd line arg */
  if (argc != 2) {
    printf("usage: %s <filename>\n", argv[0]);
    exit(0);
  }

  /* Copy input file to stdout */
  if ((fd = open(argv[1], O_RDONLY, 0)) < 0)
    perror("open");

  while ((rc = read(fd, buf, sizeof(buf))) > 0)
     /* first argument is 1 */
     write(STDOUT_FILENO, buf, rc);

  close(fd);

  return 0;
}
```