

☒ 1. Last time

☒ 2. Weensy OS

☒ 3. gdb

☒ 4. mmap()

lab 4 work here

↪ : thinks it is interacting with hardware

↪ : uses syscall interface (and possibly virtualization extensions)

↪ : paravirtualized (in between hardware and syscall)

p-fork p-allocator

* WeensyOS

QEMU

Linux (devbox)

Docker

Your OS (Windows, Mac OS, ...)

Physical Hardware

cpu, h/w
process
cpu, h/w
process

Hints:

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

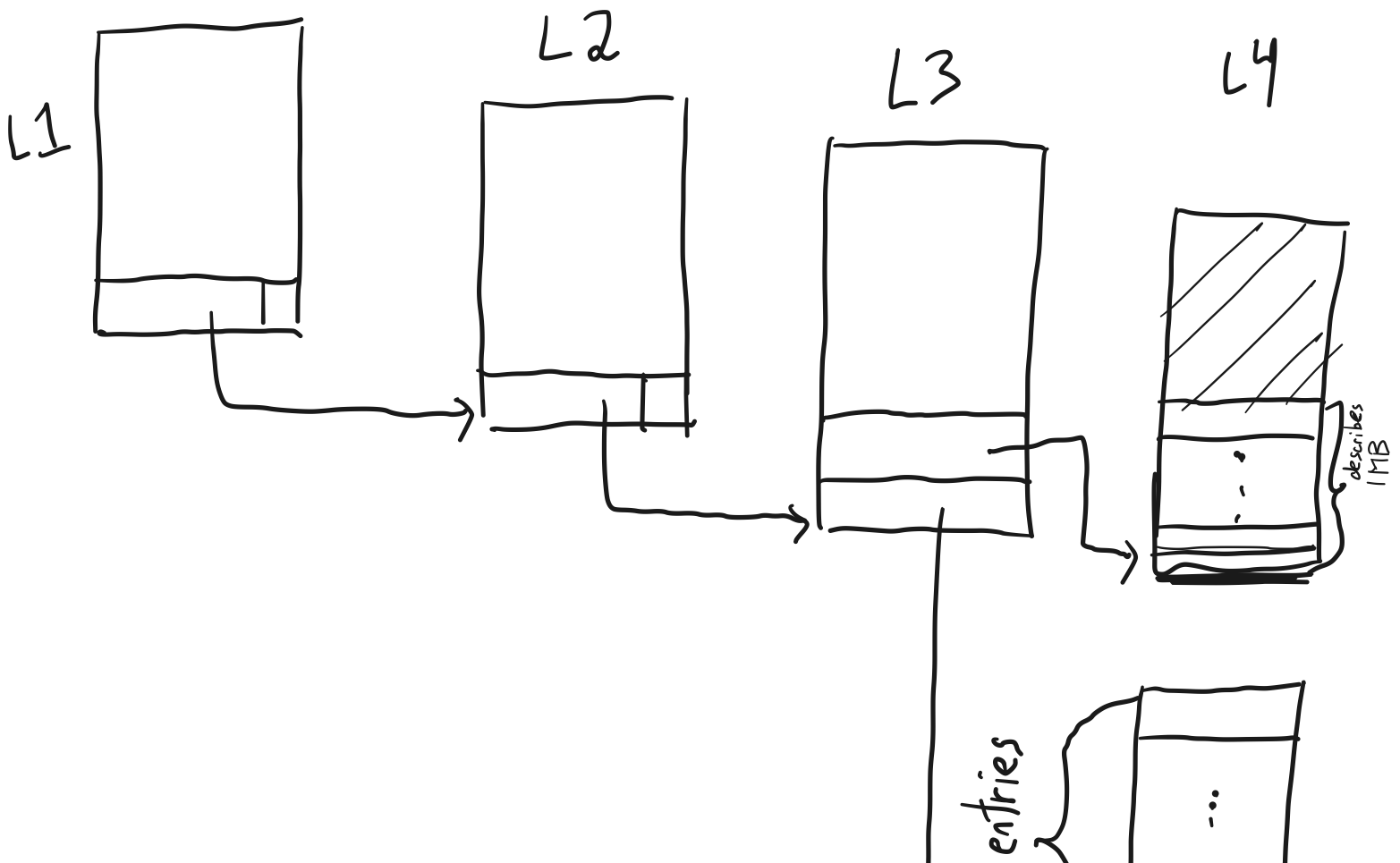
sys_page_alloc, lookin process.h

kernel returns: `exception_return()`;
%rax contains return value of system call
(errno, 0)

%rdi:
arg to
syscall

- you'll use `virtual-memory-map()` = (NULL vs. non-NULL)
- pay attention to the "allocator" argument
- 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?

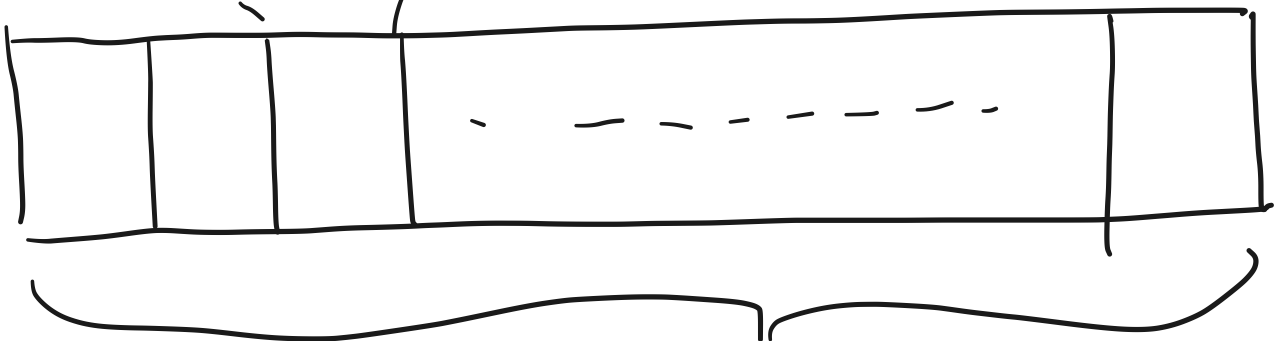




each entry
describes a mapping
for one page
(4KB)

PCB $\equiv$ struct proc (in kernel.h)

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

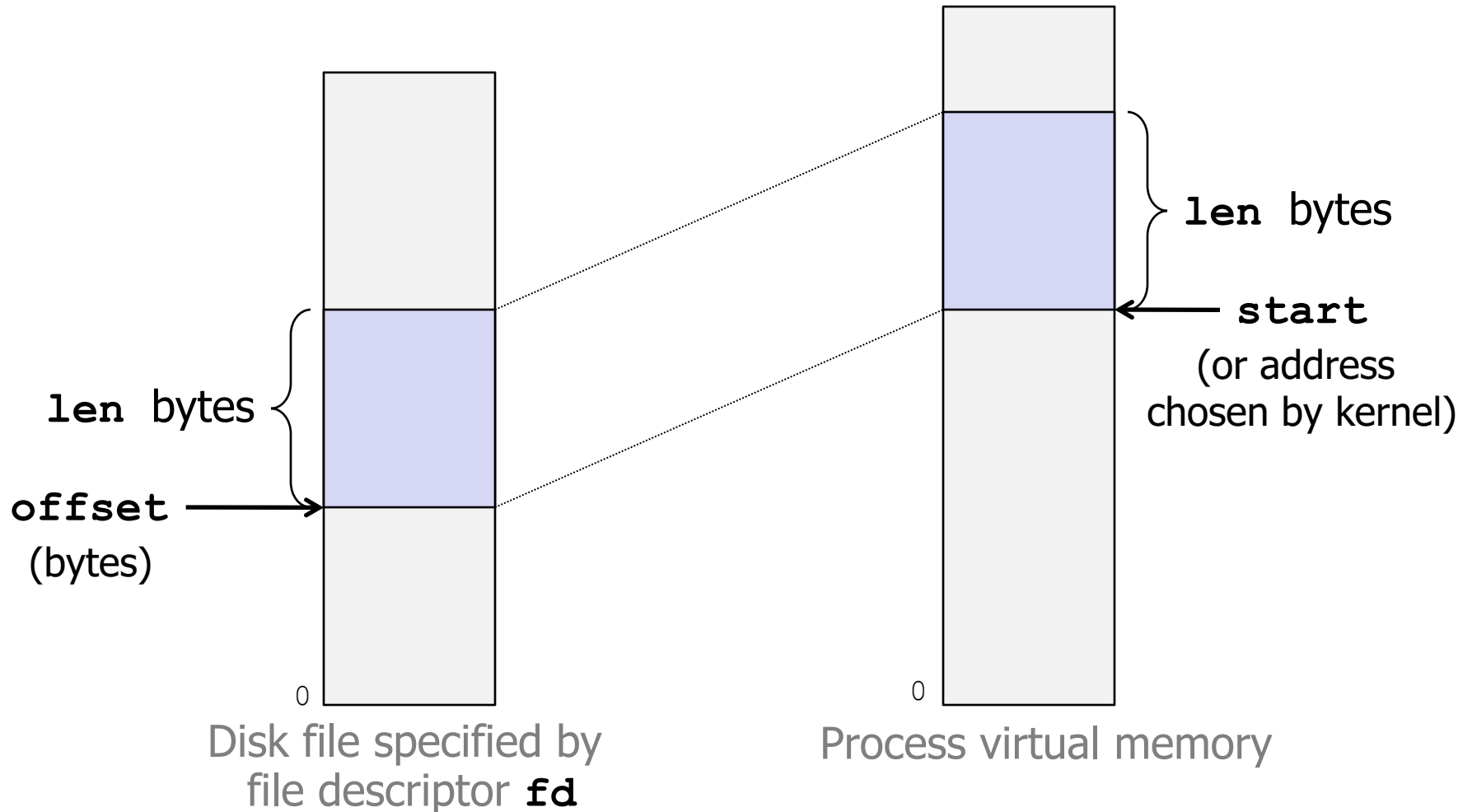


one per physical page

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)
```



Mar 19, 25 12:50

cat-mmap.c

Page 1/1

```

1  /*
2  * Takes a filename as input, and writes the contents of the file
3  * to stdout.
4  *
5  * Uses mmap()
6  */
7
8  #include <fcntl.h>
9  #include <stdio.h>
10 #include <stdlib.h>
11 #include <sys/mman.h>
12 #include <sys/stat.h>
13 #include <sys/types.h>
14 #include <unistd.h>
15
16 void mmapcopy(int fd, int size);
17
18 int main(int argc, char **argv) {
19     struct stat stat;
20     int fd;
21
22     /* Check for required cmd line arg */
23     if (argc != 2) {
24         printf("usage: %s <filename>\n", argv[0]);
25         exit(0);
26     }
27
28     /* Copy input file to stdout */
29     if ((fd = open(argv[1], O_RDONLY, 0)) < 0)
30         perror("open");
31
32     fstat(fd, &stat);
33     mmapcopy(fd, stat.st_size);
34
35     close(fd);
36
37     return 0;
38 }
39
40 void mmapcopy(int fd, int size) {
41     /* Ptr to memory mapped area */
42     char *bufp;
43
44     bufp = mmap(NULL, size, PROT_READ, MAP_PRIVATE, fd, 0);
45
46     /* first argument is 1 */
47     write(STDOUT_FILENO, bufp, size);
48
49     return;
50 }
51
52

```

Mar 19, 25 12:50

cat-naive.c

Page 1/1

```

1  /*
2  * Takes a filename as input, and writes the contents of the file
3  * to stdout.
4  *
5  * This is a naive version, and should be contrasted with the version
6  * that uses mmap().
7  */
8
9  #include <stdio.h>
10 #include <stdlib.h>
11 #include <unistd.h>
12 #include <fcntl.h>
13
14 int main(int argc, char **argv) {
15     int fd;
16     int rc;
17     char buf[256];
18
19     /* Check for required cmd line arg */
20     if (argc != 2) {
21         printf("usage: %s <filename>\n", argv[0]);
22         exit(0);
23     }
24
25     /* Copy input file to stdout */
26     if ((fd = open(argv[1], O_RDONLY, 0)) < 0)
27         perror("open");
28
29     while ((rc = read(fd, buf, sizeof(buf))) > 0)
30         /* first argument is 1 */
31         write(STDOUT_FILENO, buf, rc);
32
33     close(fd);
34
35     return 0;
36 }

```