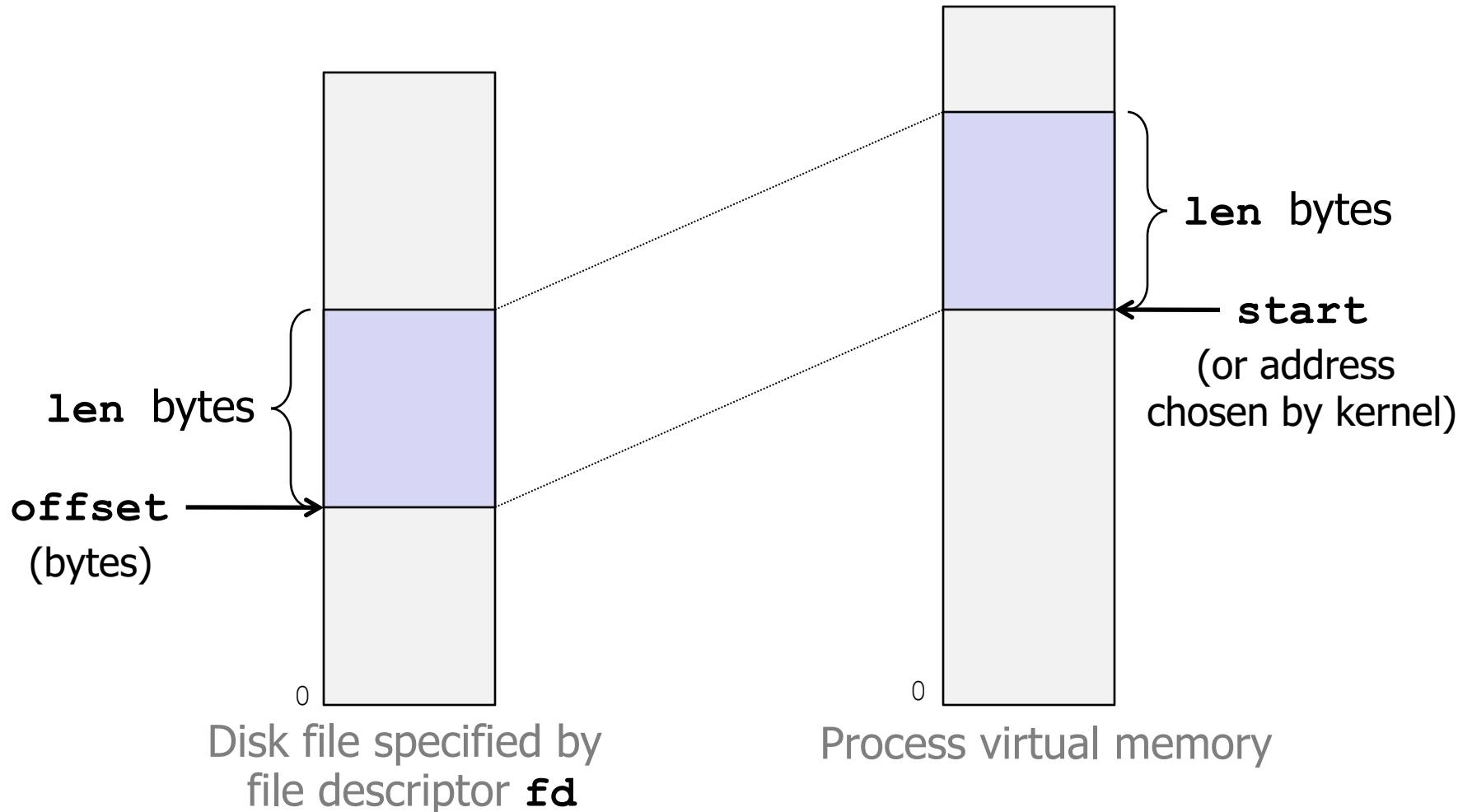


User-Level Memory Mapping

```
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
42     /* Ptr to memory mapped area */
43     char *bufp;
44
45     bufp = mmap(NULL, size, PROT_READ, MAP_PRIVATE, fd, 0);
46
47     /* first argument is 1 */
48     write(STDOUT_FILENO, bufp, size);
49
50     return;
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 }

```