last time

page table review
- virtual to physical translation
- two-level page tables

how xv6 manages page tables
- `walkpgdir`, `mappages`, etc.
- x86-32 page table format

xv6 memory layout
- high memory for the kernel, mapping everything
- virtual-to-physical/phyiscal-to-virtual utility functions
- sbrk to determine end of user memory

page fault handling

xv6 page faults (now)

fault from accessing page table entry marked ‘not-present’

xv6: prints an error and kills process:

```
*((int*) 0x800444) = 1;
...
/* in trap.c: */
    cprintf("pid %d %s: trap %d err %d on cpu %d "
            "eip 0x%x addr 0x%x--kill proc\n",
            myproc()->pid, myproc()->name, tf->trapno,
            tf->err, cpuid(), tf->eip, rcr2());
    myproc()->killed = 1;
```

```
pid 4 processname: trap 14 err 6 on cpu 0 eip 0x1a addr 0x800444--k
```

14 = T_PGFLT

special register CR2 contains faulting address

xv6 page faults (now)

fault from accessing page table entry marked ‘not-present’

xv6: prints an error and kills process:

```
*((int*) 0x800444) = 1;
...
/* in trap.c: */
    cprintf("pid␣%d␣%s:␣trap␣%d␣err␣%d␣on␣cpu␣%d␣"
            "eip␣0x%x␣addr␣0x%x--kill␣proc\n",
            myproc()->pid, myproc()->name, tf->trapno,
            tf->err, cpuid(), tf->eip, rcr2());
    myproc()->killed = 1;
```

```
pid 4 processname: trap 14 err 6 on cpu 0 eip 0x1a addr 0x800444--k
```

14 = T_PGFLT

special register CR2 contains faulting address

xv6 page faults (now)

fault from accessing page table entry marked 'not-present'

xv6: prints an error and kills process:

```
*((int*) 0x800444) = 1;
...
/* in trap.c: */
    cprintf("pid␣%d␣%s:␣trap␣%d␣err␣%d␣on␣cpu␣%d␣"
            "eip␣0x%x␣addr␣0x%x--kill␣proc\n",
            myproc()->pid, myproc()->name, tf->trapno,
            tf->err, cpuid(), tf->eip, rcr2());
    myproc()->killed = 1;
```

```
pid 4 processname: trap 14 err 6 on cpu 0 eip 0x1a addr 0x800444--k
```

14 = T_PGFLT

special register CR2 contains faulting address

xv6: if one handled page faults

returning from page fault handler without killing process

…retries the failing instruction

can use to update the page table — "just in time"

```
if (tf->trapno == T_PGFLT) {
  void *address = (void *) rcr2();
  if (is_address_okay(myproc(), address)) {
      setup_page_table_entry_for(myproc(), address);
      // return from fault, retry access
  } else {
      // actual segfault, kill process
      cprintf("...");
      myproc()->killed = 1;
  }
}
```

xv6: if one handled page faults

check *process control block* to see if access okay

returning from page fault handler without killing process

…retries the failing instruction

can use to update the page table — "just in time"

```
if (tf->trapno == T_PGFLT) {
  void *address = (void *) rcr2();
  if (is_address_okay(myproc(), address)) {
      setup_page_table_entry_for(myproc(), address);
      // return from fault, retry access
  } else {
      // actual segfault, kill process
      cprintf("...");
      myproc()->killed = 1;
  }
}
```

xv6: if one handled page faults

if so, setup the page table so it works next time
i.e. immediately after returning from fault

returning from page

…retries the failing instruction

can use to update the page table — "just in time"

```
if (tf->trapno == T_PGFLT) {
  void *address = (void *) rcr2();
  if (is_address_okay(myproc(), address)) {
      setup_page_table_entry_for(myproc(), address);
      // return from fault, retry access
  } else {
      // actual segfault, kill process
      cprintf("...");
      myproc()->killed = 1;
  }
}
```

extra data structures needed

OSs can do all sorts of tricks with page tables

...but more bookkeeping is required

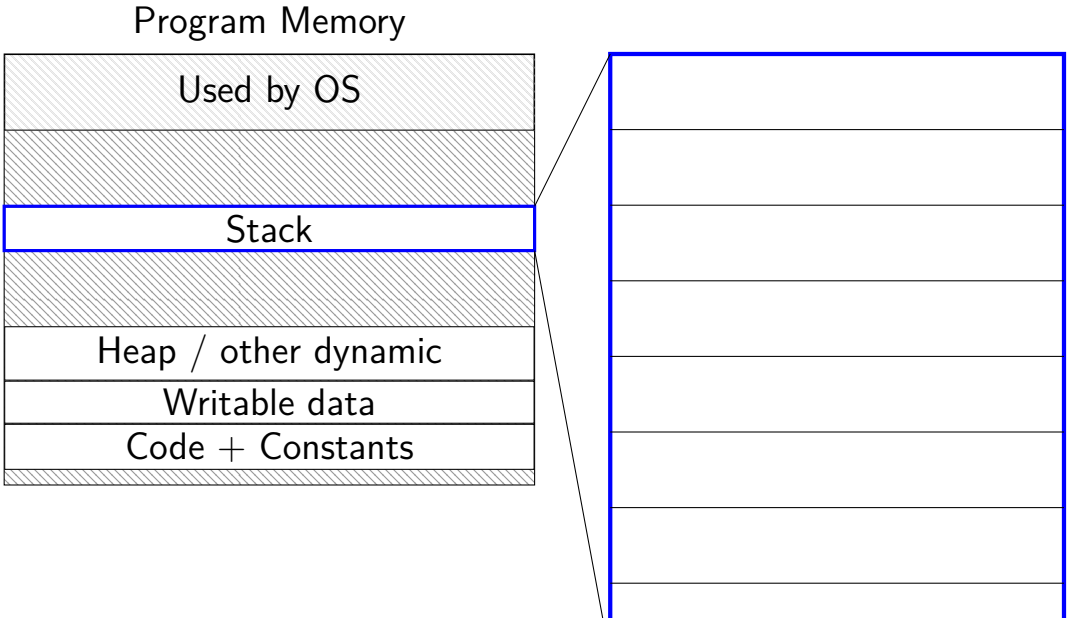
tracking what processes think they have in memory

- since page table won't tell the whole story
- OS will change page table

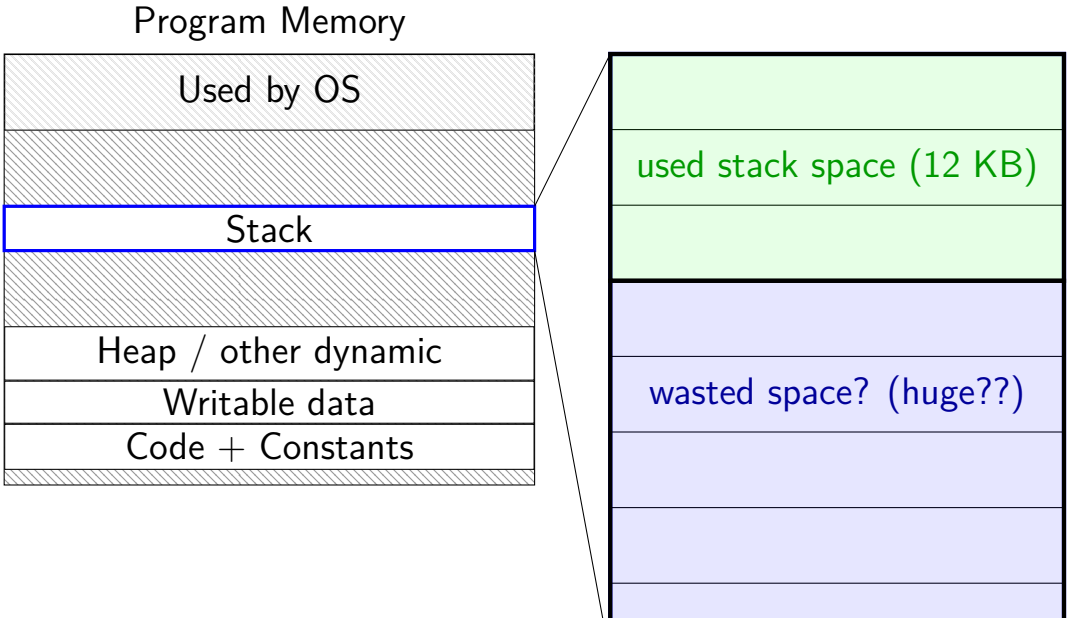
tracking how physical pages are used in page tables

- multiple processes might want same data = same page

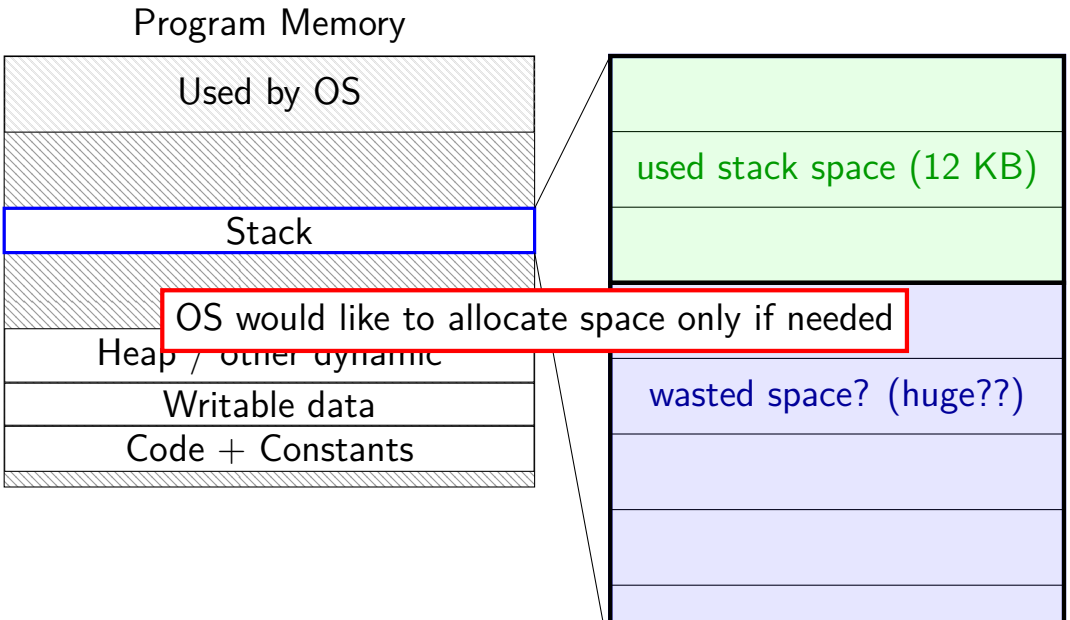
space on demand



space on demand



space on demand

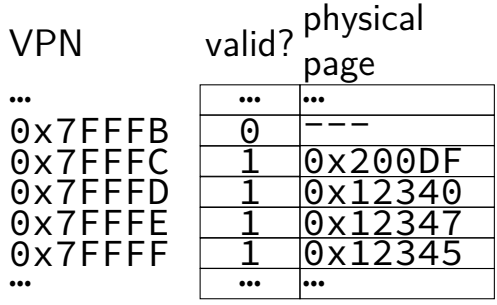


allocating space on demand

`%rsp = 0x7FFFC000`

```
...
// requires more stack space
A: pushq %rbx

B: movq 8(%rcx), %rbx
C: addq %rbx, %rax
...
```



VPN	valid?	physical page
…	…	…
0x7FFFB	0	---
0x7FFFC	1	0x200DF
0x7FFFD	1	0x12340
0x7FFFE	1	0x12347
0x7FFFF	1	0x12345
…	…	…

allocating space on demand

`%rsp = 0x7FFFC000`

```
...
// requires more stack space
A: pushq %rbx
B: movq 8(%rcx), %rbx
C: addq %rbx, %rax
...
```

page fault!

VPN	valid?	physical page
…	…	…
0x7FFFB	0	---
0x7FFFC	1	0x200DF
0x7FFFD	1	0x12340
0x7FFFE	1	0x12347
0x7FFFF	1	0x12345
…	…	…

pushq triggers exception
hardware says "accessing address `0x7FFFBFF8`"
OS looks up what's should be there — "stack"

allocating space on demand

`%rsp = 0x7FFFC000`

```
...
// requires more stack space
A: pushq %rbx
B: movq 8(%rcx), %rbx
C: addq %rbx, %rax
...
```

restarted

VPN	valid?	physical page
…	…	…
0x7FFFB	1	0x200D8
0x7FFFC	1	0x200DF
0x7FFFD	1	0x12340
0x7FFFE	1	0x12347
0x7FFFF	1	0x12345
…	…	…

in exception handler, OS allocates more stack space
OS updates the page table
then returns to retry the instruction

xv6: adding space on demand

```
struct proc {
  uint sz;     // Size of process memory (bytes)
  ...
};
```

adding allocate on demand logic:

on page fault: if address $\geq$ sz

- kill process — out of bounds

on page fault: if address $<$ sz

- find virtual page number of address
- allocate page of memory, add to page table
- return from interrupt

versus more complicated OSes

range of valid addresses is not just 0 to maximum

need some more complicated data structure to represent

will get to that later

fast copies

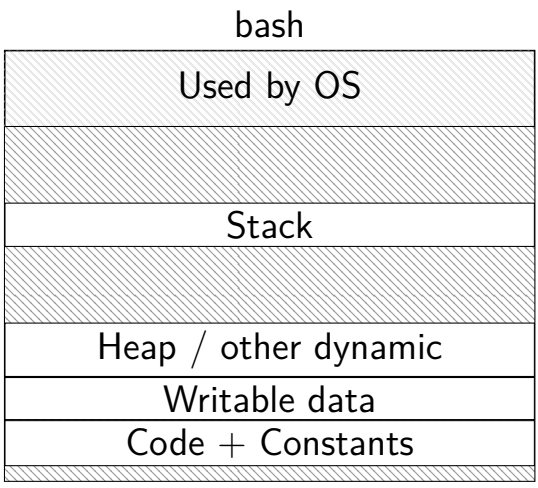
recall : `fork()`

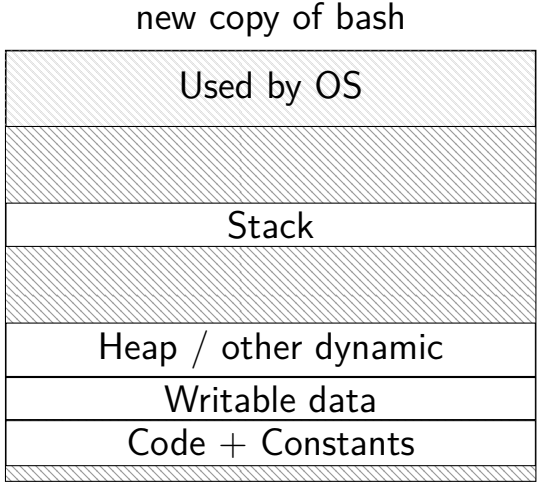
creates a copy of an entire program!

(usually, the copy then calls `execve` — replaces itself with another program)

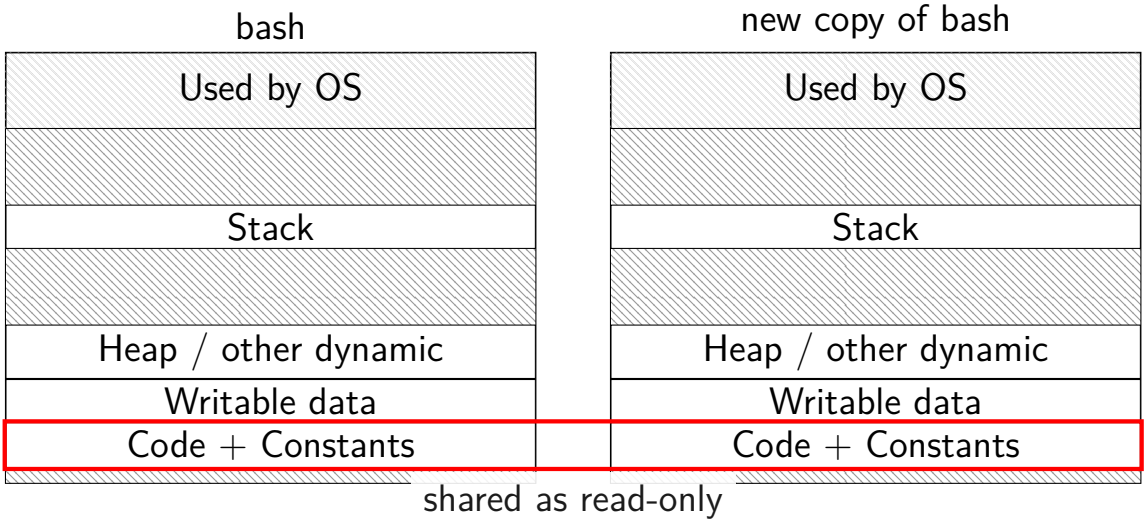
how isn't this really slow?

do we really need a complete copy?

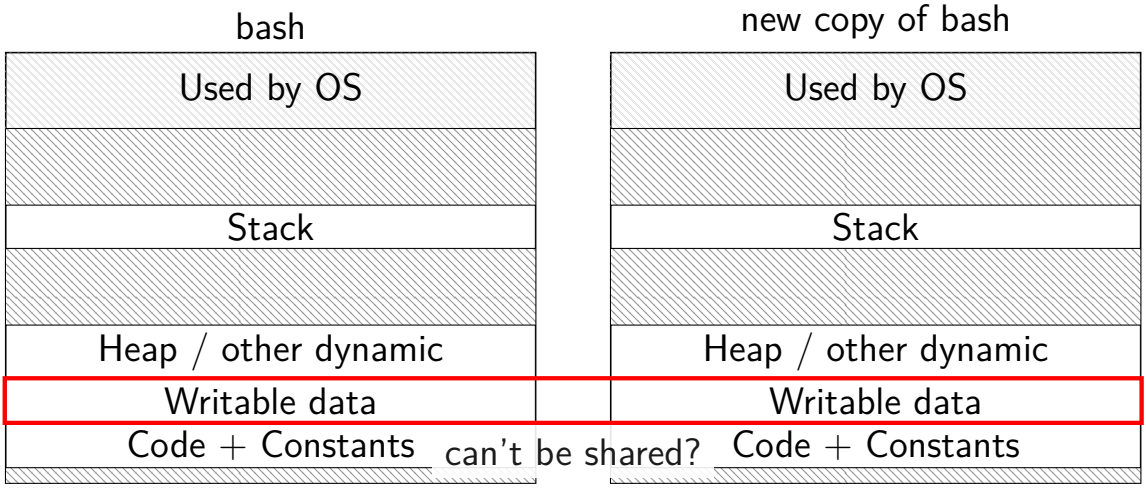




do we really need a complete copy?



do we really need a complete copy?



trick for extra sharing

sharing writeable data is fine — until either process modifies the copy

can we detect modifications?

trick: tell CPU (via page table) shared part is read-only

processor will trigger a fault when it's written

copy-on-write and page tables

VPN	valid?	write?	physical page
…	…	…	…
0x00601	1	1	0x12345
0x00602	1	1	0x12347
0x00603	1	1	0x12340
0x00604	1	1	0x200DF
0x00605	1	1	0x200AF
…	…	…	…

copy-on-write and page tables

VPN	valid?	write?	physical page
…	…	…	…
0x00601	1	0	0x12345
0x00602	1	0	0x12347
0x00603	1	0	0x12340
0x00604	1	0	0x200DF
0x00605	1	0	0x200AF
…	…	…	…

VPN	valid?	write?	physical page
…	…	…	…
0x00601	1	0	0x12345
0x00602	1	0	0x12347
0x00603	1	0	0x12340
0x00604	1	0	0x200DF
0x00605	1	0	0x200AF
…	…	…	…

copy operation actually duplicates page table
both processes share all physical pages
but marks pages in both copies as read-only

copy-on-write and page tables

VPN	valid?	write?	physical page
…	…	…	…
0x00601	1	0	0x12345
0x00602	1	0	0x12347
0x00603	1	0	0x12340
0x00604	1	0	0x200DF
0x00605	1	0	0x200AF
…	…	…	…

VPN	valid?	write?	physical page
…	…	…	…
0x00601	1	0	0x12345
0x00602	1	0	0x12347
0x00603	1	0	0x12340
0x00604	1	0	0x200DF
0x00605	1	0	0x200AF
…	…	…	…

when either process tries to write read-only page triggers a fault — OS actually copies the page

copy-on-write and page tables

VPN	valid?	write?	physical page
…	…	…	…
0x00601	1	0	0x12345
0x00602	1	0	0x12347
0x00603	1	0	0x12340
0x00604	1	0	0x200DF
0x00605	1	0	0x200AF
…	…	…	…

VPN	valid?	write?	physical page
…	…	…	…
0x00601	1	0	0x12345
0x00602	1	0	0x12347
0x00603	1	0	0x12340
0x00604	1	0	0x200DF
0x00605	1	1	0x300FD
…	…	…	…

after allocating a copy, OS reruns the write instruction

copy-on write cases

trying to write forbidden page (e.g. kernel memory)

- kill program instead of making it writable

trying to write read-only page and...

only one page table entry refers to it

- make it writeable
- return from fault

multiple process's page table entries refer to it

- copy the page
- replace read-only page table entry to point to copy
- return from fault

mmap

Linux/Unix has a function to “map” a file to memory

```
int file = open("somefile.dat", O_RDWR);

    // data is region of memory that represents file
char *data = mmap(..., file, 0);

    // read byte 6 from somefile.dat
char seventh_char = data[6];

   // modifies byte 100 of somefile.dat
data[100] = 'x';
    // can continue to use 'data' like an array
```

mmap options (1)

```
#include <sys/mman.h>
void *mmap(void *addr, size_t length, int prot, int flags,
           int fd, off_t offset);
```

`length` bytes from open file `fd` starting at byte `offset`

protection flags `prot`, bitwise or together 1 or more of:

- PROT_READ
- PROT_WRITE
- PROT_EXEC
- PROT_NONE (for forcing segfaults)

mmap options (1)

```
#include <sys/mman.h>
void *mmap(void *addr, size_t length, int prot, int flags,
           int fd, off_t offset);
```

`length` bytes from open file `fd` starting at byte `offset`

protection flags `prot`, bitwise or together 1 or more of:

- PROT_READ
- PROT_WRITE
- PROT_EXEC
- PROT_NONE (for forcing segfaults)

mmap options (1)

```
#include <sys/mman.h>
void *mmap(void *addr, size_t length, int prot, int flags,
           int fd, off_t offset);
```

`length` bytes from open file `fd` starting at byte `offset`

protection flags `prot`, bitwise or together 1 or more of:

- PROT_READ
- PROT_WRITE
- PROT_EXEC
- PROT_NONE (for forcing segfaults)

mmap options (2)

```
#include <sys/mman.h>
void *mmap(void *addr, size_t length, int prot, int flags,
           int fd, off_t offset);
```

`flags`, choose at least

- MAP_SHARED — changing memory changes file and vice-versa
- MAP_PRIVATE — make a copy of data in file (using copy-on-write)

…along with additional flags:

- MAP_ANONYMOUS (not POSIX) — ignore `fd`, just allocate space
- … (and more not shown)

`addr`, suggestion about where to put mapping (may be ignored)

- can pass NULL — "choose for me"
- address chosen will be returned

mmap options (2)

```
#include <sys/mman.h>
void *mmap(void *addr, size_t length, int prot, int flags,
           int fd, off_t offset);
```

`flags`, choose at least

- MAP_SHARED — changing memory changes file and vice-versa
- MAP_PRIVATE — make a copy of data in file (using copy-on-write)

…along with additional flags:

- MAP_ANONYMOUS (not POSIX) — ignore `fd`, just allocate space
- … (and more not shown)

`addr`, suggestion about where to put mapping (may be ignored)

- can pass NULL — "choose for me"
- address chosen will be returned

Linux maps

```
$ cat /proc/self/maps
00400000-0040b000 r-xp 00000000 08:01 48328831                           /bin/cat
0060a000-0060b000 r--p 0000a000 08:01 48328831                           /bin/cat
0060b000-0060c000 rw-p 0000b000 08:01 48328831                           /bin/cat
01974000-01995000 rw-p 00000000 00:00 0                                  [heap]
7f60c718b000-7f60c7490000 r--p 00000000 08:01 77483660                   /usr/lib/locale/locale-archive
7f60c7490000-7f60c764e000 r-xp 00000000 08:01 96659129                   /lib/x86_64-linux-gnu/libc-2.19
7f60c764e000-7f60c784e000 ---p 001be000 08:01 96659129                   /lib/x86_64-linux-gnu/libc-2.19
7f60c784e000-7f60c7852000 r--p 001be000 08:01 96659129                   /lib/x86_64-linux-gnu/libc-2.19
7f60c7852000-7f60c7854000 rw-p 001c2000 08:01 96659129                   /lib/x86_64-linux-gnu/libc-2.19
7f60c7854000-7f60c7859000 rw-p 00000000 00:00 0
7f60c7859000-7f60c787c000 r-xp 00000000 08:01 96659109                   /lib/x86_64-linux-gnu/ld-2.19.so
7f60c7a39000-7f60c7a3b000 rw-p 00000000 00:00 0
7f60c7a7a000-7f60c7a7b000 rw-p 00000000 00:00 0
7f60c7a7b000-7f60c7a7c000 r--p 00022000 08:01 96659109                   /lib/x86_64-linux-gnu/ld-2.19.so
7f60c7a7c000-7f60c7a7d000 rw-p 00023000 08:01 96659109                   /lib/x86_64-linux-gnu/ld-2.19.so
7f60c7a7d000-7f60c7a7e000 rw-p 00000000 00:00 0
7ffc5d2b2000-7ffc5d2d3000 rw-p 00000000 00:00 0                          [stack]
7ffc5d3b0000-7ffc5d3b3000 r--p 00000000 00:00 0                          [vvar]
7ffc5d3b3000-7ffc5d3b5000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
```

Linux maps

```
$ cat /proc/self/maps
00400000-0040b000 r-xp 00000000 08:01 48328831                           /bin/cat
0060a000-0060b000 r--p 0000a000 08:01 48328831                           /bin/cat
0060b000-0060c000 rw-p 0000b000 08:01 48328831                           /bin/cat
01974000-01995000 rw-p 00000000 00:00 0                                  [heap]
7f60c718b000-7f60c7490000 r--p 00000000 08:01 77483660                   /usr/lib/locale/locale-archive
7f60c7490000-7f60c764e000 r
7f60c764e000-7f60c784e000 -
7f60c784e000-7f60c7852000 r
7f60c7852000-7f60c7854000 rw-p 001c2000 08:01 96659129                   /lib/x86_64-linux-gnu/libc-2.19
7f60c7854000-7f60c7859000 rw-p 00000000 00:00 0
7f60c7859000-7f60c787c000 r-xp 00000000 08:01 96659109                   /lib/x86_64-linux-gnu/ld-2.19.so
7f60c7a39000-7f60c7a3b000 rw-p 00000000 00:00 0
7f60c7a7a000-7f60c7a7b000 rw-p 00000000 00:00 0
7f60c7a7b000-7f60c7a7c000 r--p 00022000 08:01 96659109                   /lib/x86_64-linux-gnu/ld-2.19.so
7f60c7a7c000-7f60c7a7d000 rw-p 00023000 08:01 96659109                   /lib/x86_64-linux-gnu/ld-2.19.so
7f60c7a7d000-7f60c7a7e000 rw-p 00000000 00:00 0
7ffc5d2b2000-7ffc5d2d3000 rw-p 00000000 00:00 0                          [stack]
7ffc5d3b0000-7ffc5d3b3000 r--p 00000000 00:00 0                          [vvar]
7ffc5d3b3000-7ffc5d3b5000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
```

at virtual addresses 0x400000–0x40b000

Linux maps

```
$ cat /proc/self/maps
00400000-0040b000 r-xp 00000000 08:01 48328831                           /bin/cat
0060a000-0060b000 r--p 0000a000 08:01 48328831                           /bin/cat
0060b000-0060c000 rw-p 0000b000 08:01 48328831                           /bin/cat
01974000-01995000 rw-p 00000000 00:00 0                                  [heap]
7f60c718b000-7f60c7490000 r--p 00000000 08:01 77483660                   /usr/lib/locale/locale-archive
7f60c7490000-7f60c764e000 r-xp 000000[illegible]                                                  -2.19
7f60c764e000-7f60c784e000 ---p 001be0[illegible]                                                  -2.19
7f60c784e000-7f60c7852000 r--p 001be0[illegible]                                                  -2.19
7f60c7852000-7f60c7854000 rw-p 001c20[illegible]                                                  -2.19
7f60c7854000-7f60c7859000 rw-p 000000[illegible]
7f60c7859000-7f60c787c000 r-xp 00000000 08:01 96659109                   /lib/x86_64-linux-gnu/ld-2.19.so
7f60c7a39000-7f60c7a3b000 rw-p 00000000 00:00 0
7f60c7a7a000-7f60c7a7b000 rw-p 00000000 00:00 0
7f60c7a7b000-7f60c7a7c000 r--p 00022000 08:01 96659109                   /lib/x86_64-linux-gnu/ld-2.19.so
7f60c7a7c000-7f60c7a7d000 rw-p 00023000 08:01 96659109                   /lib/x86_64-linux-gnu/ld-2.19.so
7f60c7a7d000-7f60c7a7e000 rw-p 00000000 00:00 0
7ffc5d2b2000-7ffc5d2d3000 rw-p 00000000 00:00 0                          [stack]
7ffc5d3b0000-7ffc5d3b3000 r--p 00000000 00:00 0                          [vvar]
7ffc5d3b3000-7ffc5d3b5000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
```

read, not write, execute, private

private = copy-on-write (if writeable)

Linux maps

```
$ cat /proc/self/maps
00400000-0040b000 r-xp 00000000 08:01 48328831                   /bin/cat
0060a000-0060b000 r--p 0000a000 08:01 48328831                   /bin/cat
0060b000-0060c000 rw-p 0000b000 08:01 48328831                   /bin/cat
01974000-01995000 rw-p 00000000 00:00 0                          [heap]
7f60c718b000-7f60c7490000 r--p 00000000 08:01 77483660           /usr/lib/locale/locale-archive
7f60c7490000-7f60c764e000 r-xp                                                              -2.19
7f60c764e000-7f60c784e000 ---p                                                              -2.19
7f60c784e000-7f60c7852000 r--p                                                              -2.19
7f60c7852000-7f60c7854000 rw-p 001c2000 08:01 96659129           /lib/x86_64-linux-gnu/libc-2.19
7f60c7854000-7f60c7859000 rw-p 00000000 00:00 0
7f60c7859000-7f60c787c000 r-xp 00000000 08:01 96659109           /lib/x86_64-linux-gnu/ld-2.19.so
7f60c7a39000-7f60c7a3b000 rw-p 00000000 00:00 0
7f60c7a7a000-7f60c7a7b000 rw-p 00000000 00:00 0
7f60c7a7b000-7f60c7a7c000 r--p 00022000 08:01 96659109           /lib/x86_64-linux-gnu/ld-2.19.so
7f60c7a7c000-7f60c7a7d000 rw-p 00023000 08:01 96659109           /lib/x86_64-linux-gnu/ld-2.19.so
7f60c7a7d000-7f60c7a7e000 rw-p 00000000 00:00 0
7ffc5d2b2000-7ffc5d2d3000 rw-p 00000000 00:00 0                  [stack]
7ffc5d3b0000-7ffc5d3b3000 r--p 00000000 00:00 0                  [vvar]
7ffc5d3b3000-7ffc5d3b5000 r-xp 00000000 00:00 0                  [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0 [vsyscall]
```

starting at offset 0 of the file `/bin/cat`

Linux maps

```
$ cat /proc/self/maps
00400000-0040b000 r-xp 00000000 08:01 48328831                           /bin/cat
0060a000-0060b000 r--p 0000a000 08:01 48328831                           /bin/cat
0060b000-0060c000 rw-p 0000b000 08:01 48328831                           /bin/cat
01974000-01995000 rw-p 00000000 00:00 0                                  [heap]
7f60c718b000-7f60c7490000 r--p 00000000 08:01 77483660                   /usr/lib/locale/locale-archive
7f60c7490000-7
7f60c764e000-7
7f60c784e000-7
7f60c7852000-7
7f60c7854000-7
7f60c7859000-7
7f60c7a39000-7
7f60c7a7a000-7
7f60c7a7b000-7f60c7a7c000 r--p 00022000 08:01 96659109                   /lib/x86_64-linux-gnu/ld-2.19.so
7f60c7a7c000-7f60c7a7d000 rw-p 00023000 08:01 96659109                   /lib/x86_64-linux-gnu/ld-2.19.so
7f60c7a7d000-7f60c7a7e000 rw-p 00000000 00:00 0
7ffc5d2b2000-7ffc5d2d3000 rw-p 00000000 00:00 0                          [stack]
7ffc5d3b0000-7ffc5d3b3000 r--p 00000000 00:00 0                          [vvar]
7ffc5d3b3000-7ffc5d3b5000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
```

device major number 8

device minor number 1

inode 48328831

more on what this means when we talk about filesystems

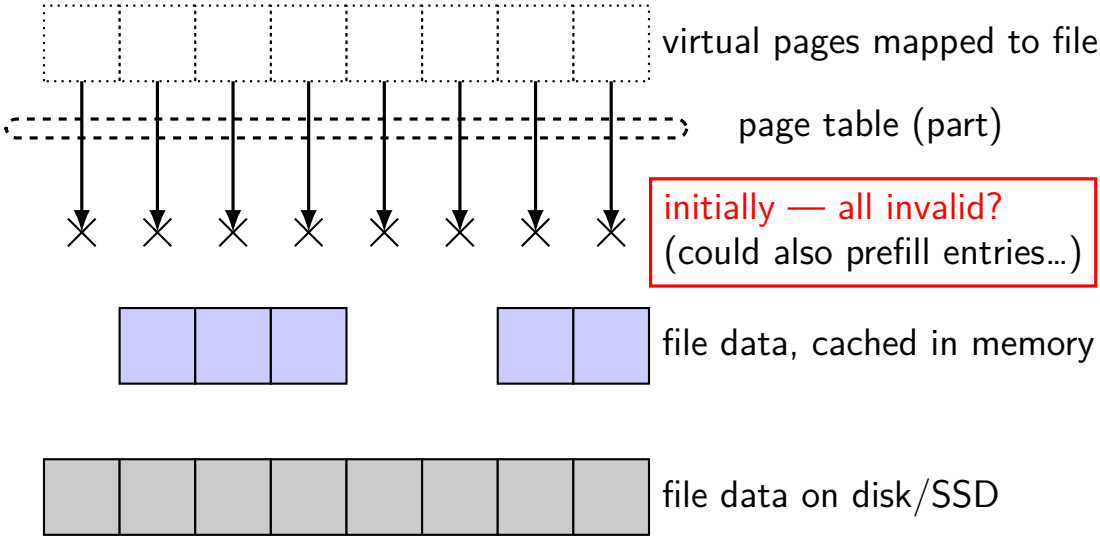
Linux maps

```
$ cat /proc/self/maps
00400000-0040b000 r-xp 00000000 08:01 48328831                           /bin/cat
0060a000-0060b000 r--p 0000a000 08:01 48328831                           /bin/cat
0060b000-0060c000 rw-p 0000b000 08:01 48328831                           /bin/cat
01974000-01995000 rw-p 00000000 00:00 0                                  [heap]
7f60c718b000-7f60c7490000 r--p 00000000 08:01 77483660                   /usr/lib/locale/locale-archive
7f60c7490000-7f6                                                                                    -2.19
7f60c764e000-7f6                                                                                    -2.19
7f60c784e000-7f6                                                                                    -2.19
7f60c7852000-7f6                                                                                    -2.19
7f60c7854000-7f6
7f60c7859000-7f6                                                                                    2.19.so
7f60c7a39000-7f6
7f60c7a7a000-7f60c7a7b000 rw-p 00000000 00:00 0
7f60c7a7b000-7f60c7a7c000 r--p 00022000 08:01 96659109                   /lib/x86_64-linux-gnu/ld-2.19.so
7f60c7a7c000-7f60c7a7d000 rw-p 00023000 08:01 96659109                   /lib/x86_64-linux-gnu/ld-2.19.so
7f60c7a7d000-7f60c7a7e000 rw-p 00000000 00:00 0
7ffc5d2b2000-7ffc5d2d3000 rw-p 00000000 00:00 0                          [stack]
7ffc5d3b0000-7ffc5d3b3000 r--p 00000000 00:00 0                          [vvar]
7ffc5d3b3000-7ffc5d3b5000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
```

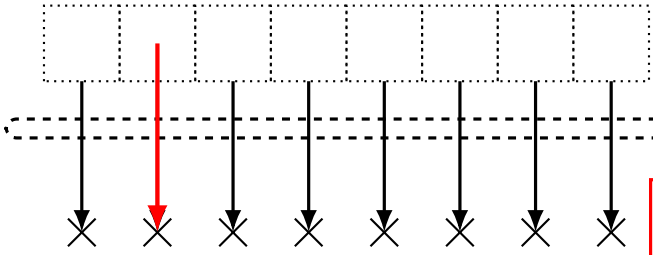
as if:

```
int fd = open("/bin/cat", O_RDONLY);
mmap(0x400000, 0x1000, PROT_READ | PROT_EXEC,
     MAP_PRIVATE, fd, 0xb000);
```

mapped pages (read-only)



mapped pages (read-only)



virtual pages mapped to file

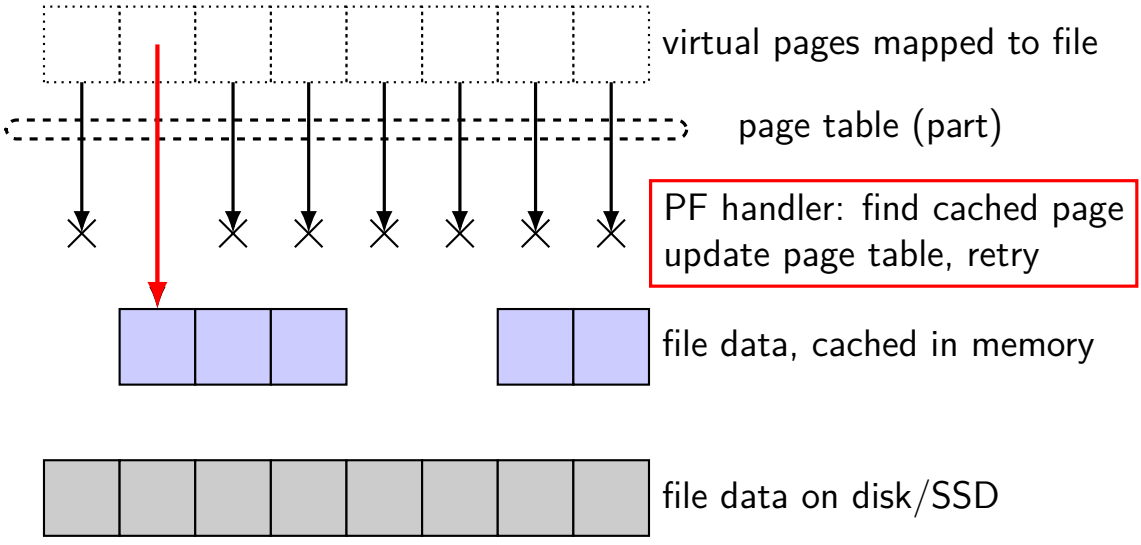
page table (part)

read from second page?
page fault

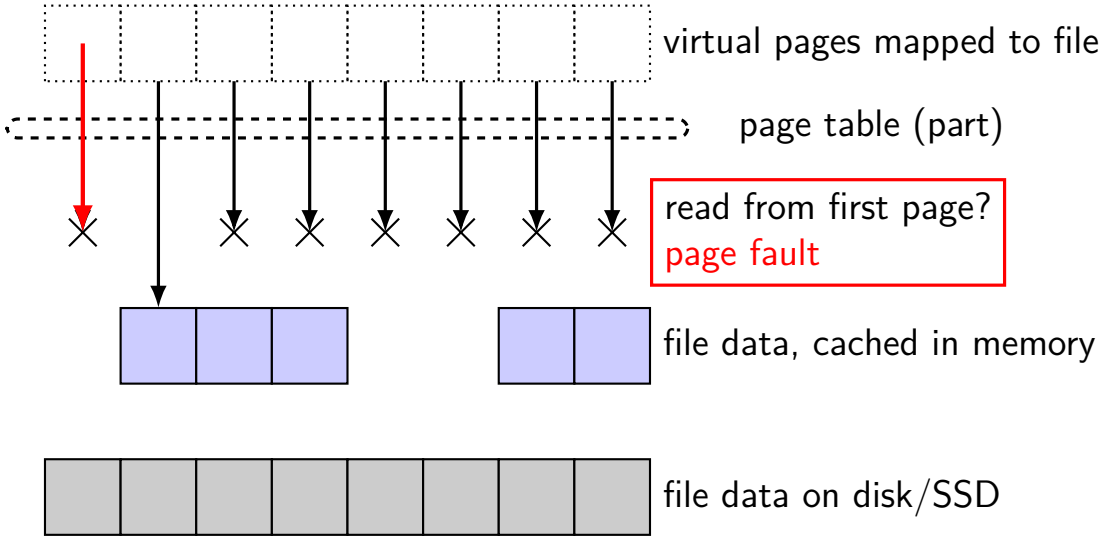
file data, cached in memory

file data on disk/SSD

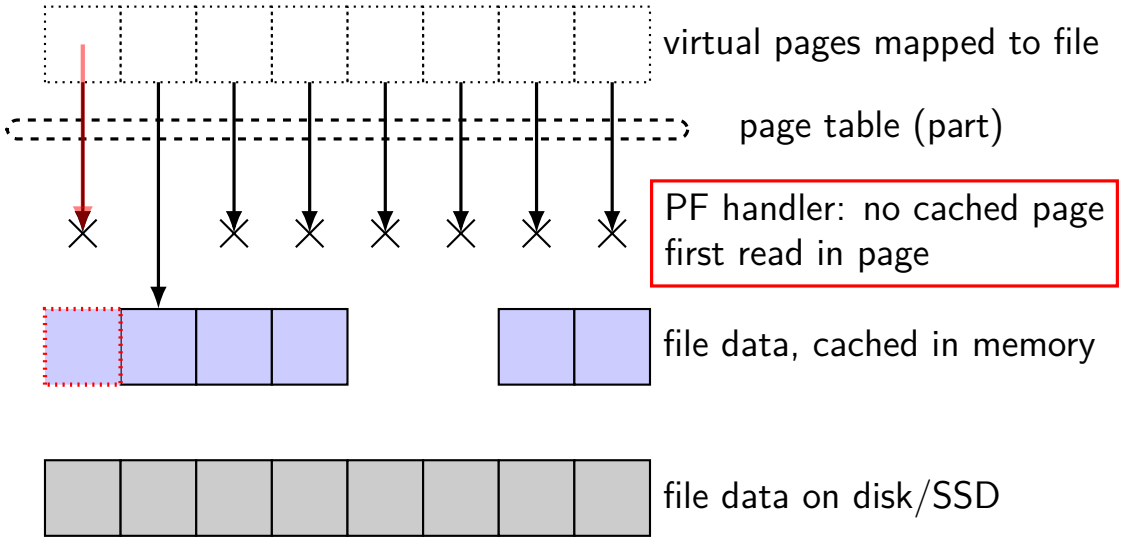
mapped pages (read-only)



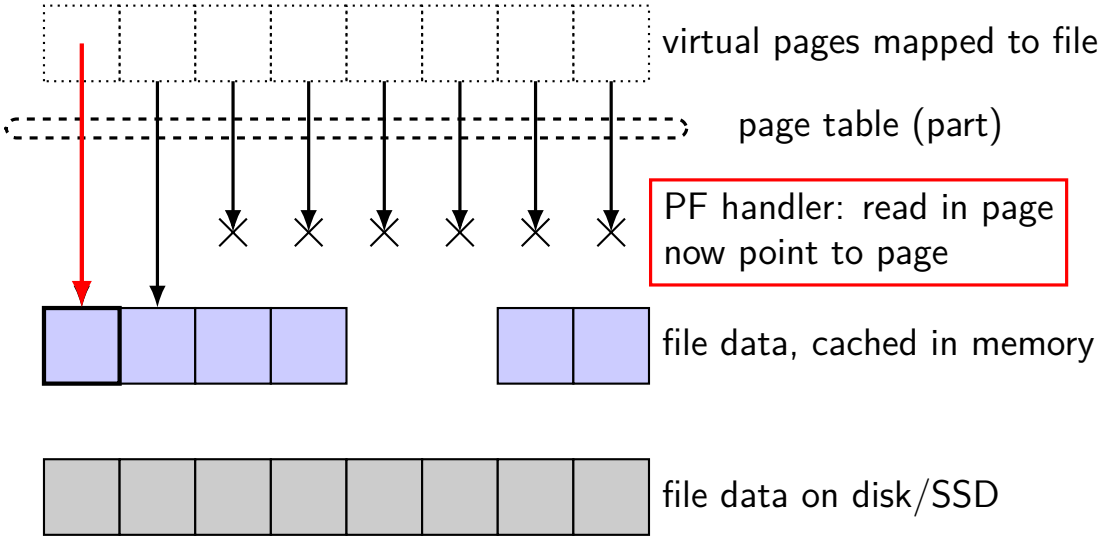
mapped pages (read-only)



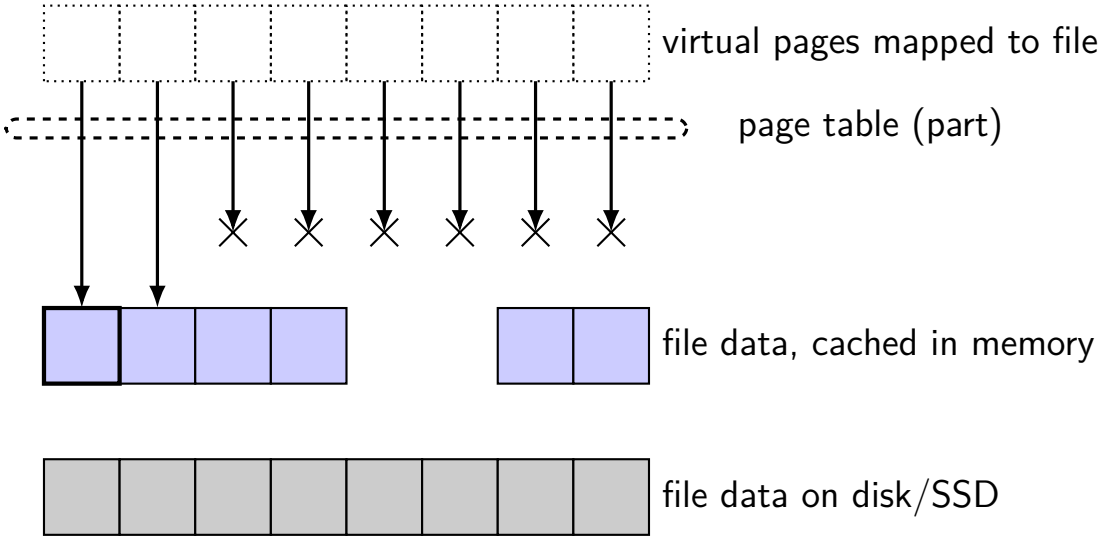
mapped pages (read-only)



mapped pages (read-only)



mapped pages (read-only)



shared mmap

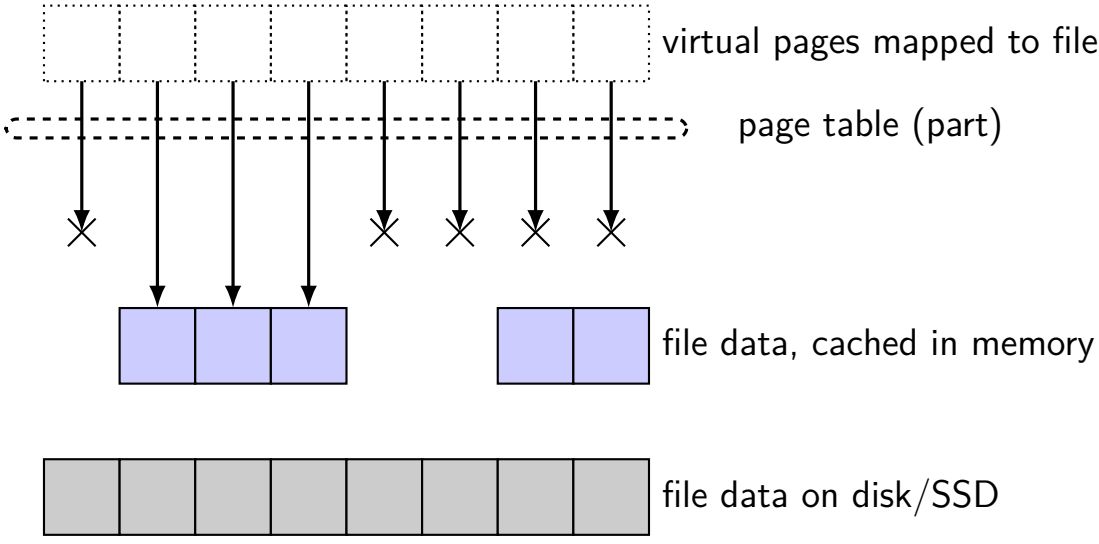
```
int fd  = open("/tmp/somefile.dat", O_RDWR);
mmap(0, 64 * 1024, PROT_READ | PROT_WRITE,
     MAP_SHARED, fd, 0);
```

---

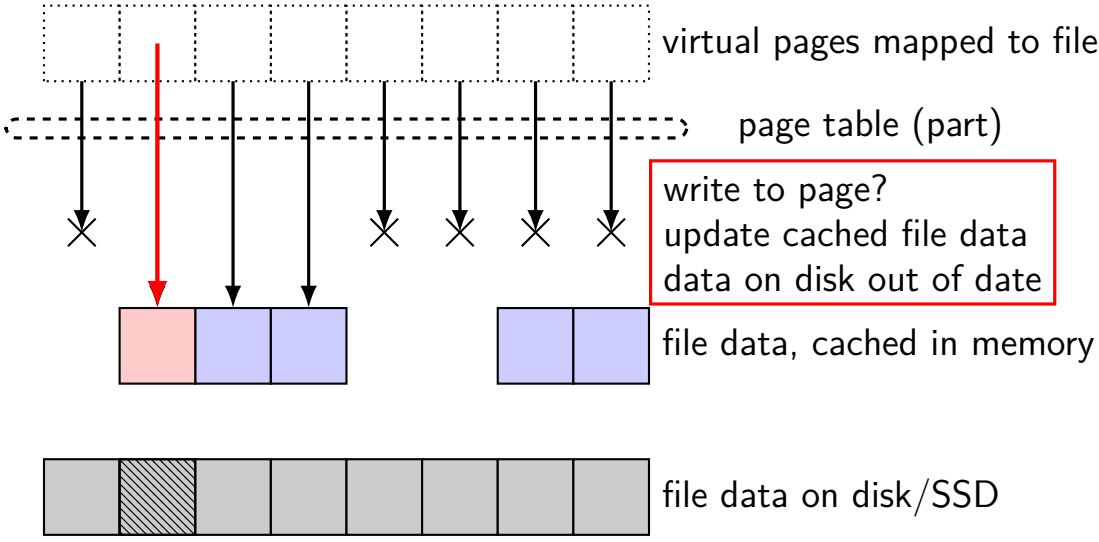
from /proc/PID/maps for this program:

```
7f93ad877000-7f93ad887000 rw-s 00000000 08:01 1839758 /tmp/somefile.dat
```

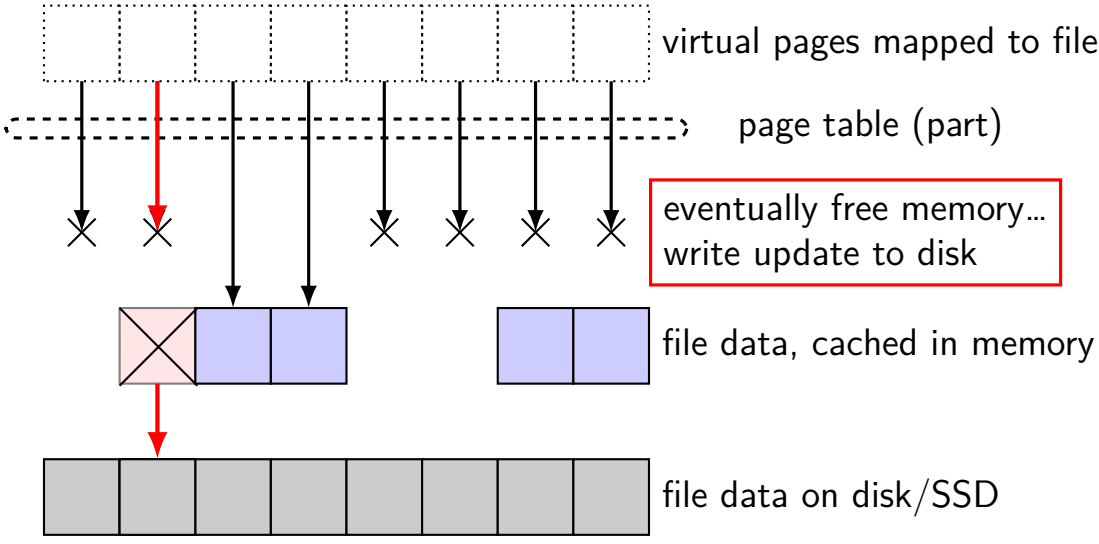
mapped pages (read/write, shared)



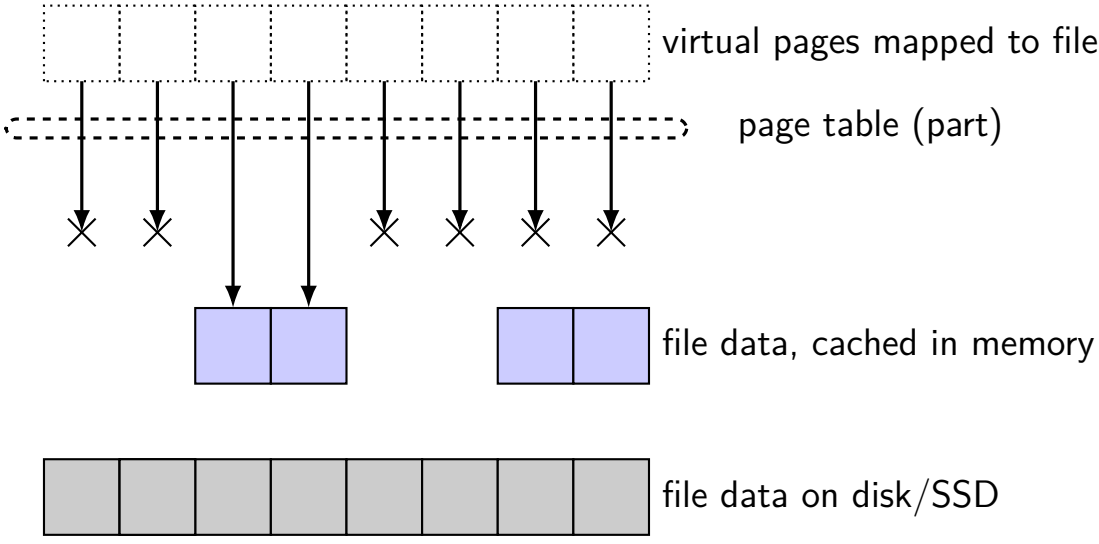
mapped pages (read/write, shared)



mapped pages (read/write, shared)



mapped pages (read/write, shared)



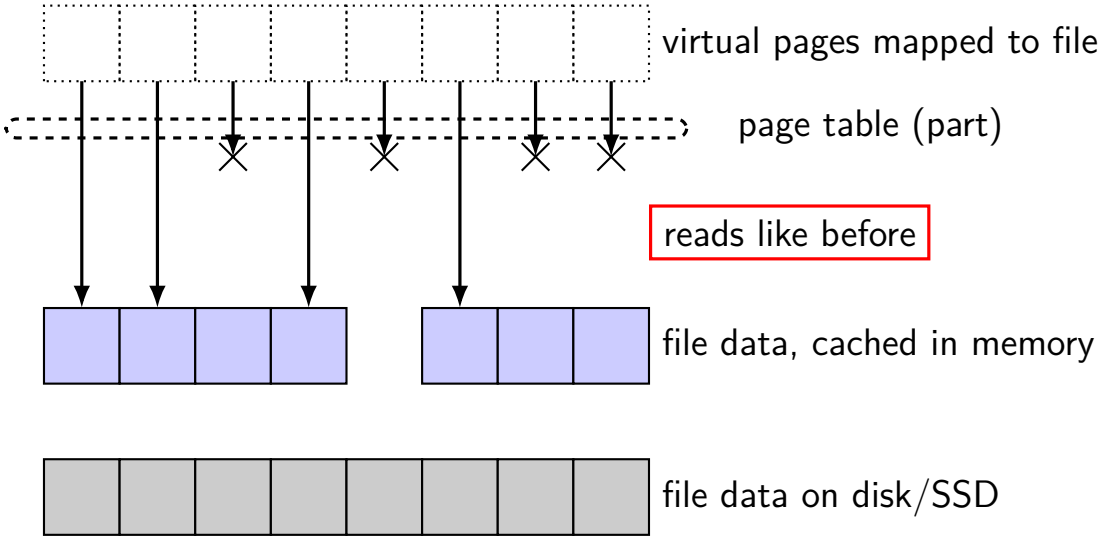
Linux maps

```
$ cat /proc/self/maps
00400000-0040b000 r-xp 00000000 08:01 48328831                           /bin/cat
0060a000-0060b000 r--p 0000a000 08:01 48328831                           /bin/cat
0060b000-0060c000 rw-p 0000b000 08:01 48328831                           /bin/cat
01974000-01995000 rw-p 00000000 00:00 0                                  [heap]
7f60c718b000-7f60c7490000 r--p 00000000 08:01 77483660                   /usr/lib/locale/locale-archive
7f60c7490000-7f                                                                                    -2.19
7f60c764e000-7f                                                                                    -2.19
7f60c784e000-7f                                                                                    -2.19
7f60c7852000-7f                                                                                    -2.19
7f60c7854000-7f
7f60c7859000-7f                                                                                    2.19.so
7f60c7a39000-7f
7f60c7a7a000-7f60c7a7b000 rw-p 00000000 00:00 0
7f60c7a7b000-7f60c7a7c000 r--p 00022000 08:01 96659109                   /lib/x86_64-linux-gnu/ld-2.19.so
7f60c7a7c000-7f60c7a7d000 rw-p 00023000 08:01 96659109                   /lib/x86_64-linux-gnu/ld-2.19.so
7f60c7a7d000-7f60c7a7e000 rw-p 00000000 00:00 0
7ffc5d2b2000-7ffc5d2d3000 rw-p 00000000 00:00 0                          [stack]
7ffc5d3b0000-7ffc5d3b3000 r--p 00000000 00:00 0                          [vvar]
7ffc5d3b3000-7ffc5d3b5000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
```

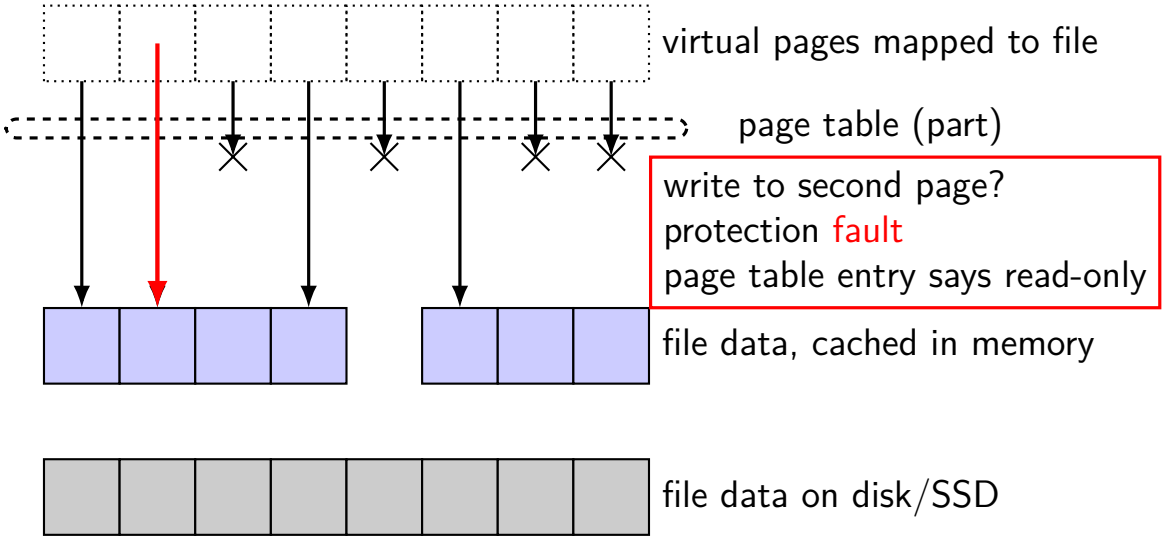
read/write, copy-on-write (private) mapping

```
int fd = open("/bin/cat", O_RDONLY);
mmap(0x60b000, 0x1000, PROT_READ | PROT_WRITE,
     MAP_PRIVATE, fd, 0xb000);
```

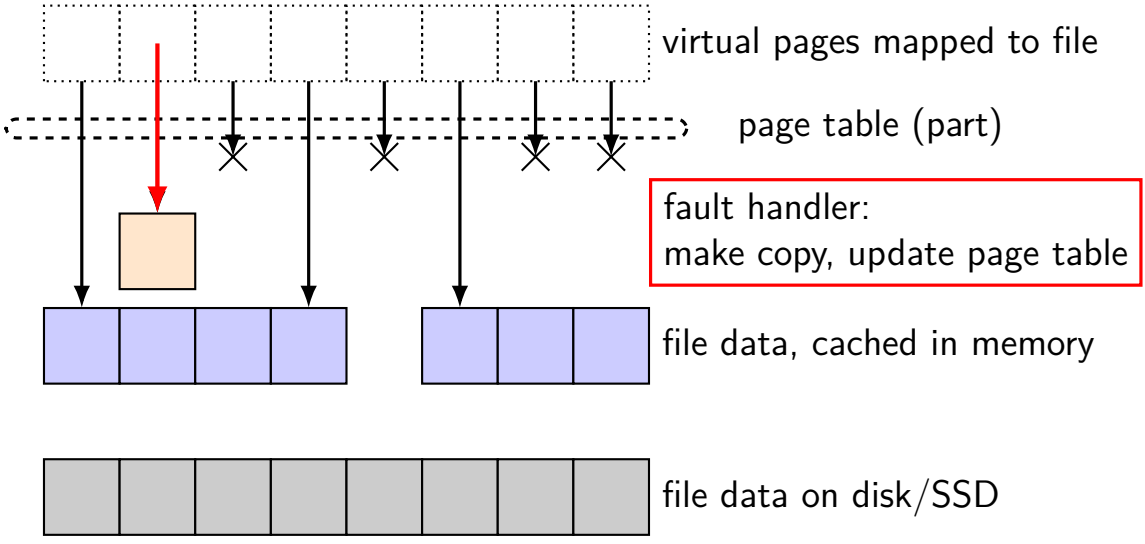
mapped pages (copy-on-write)



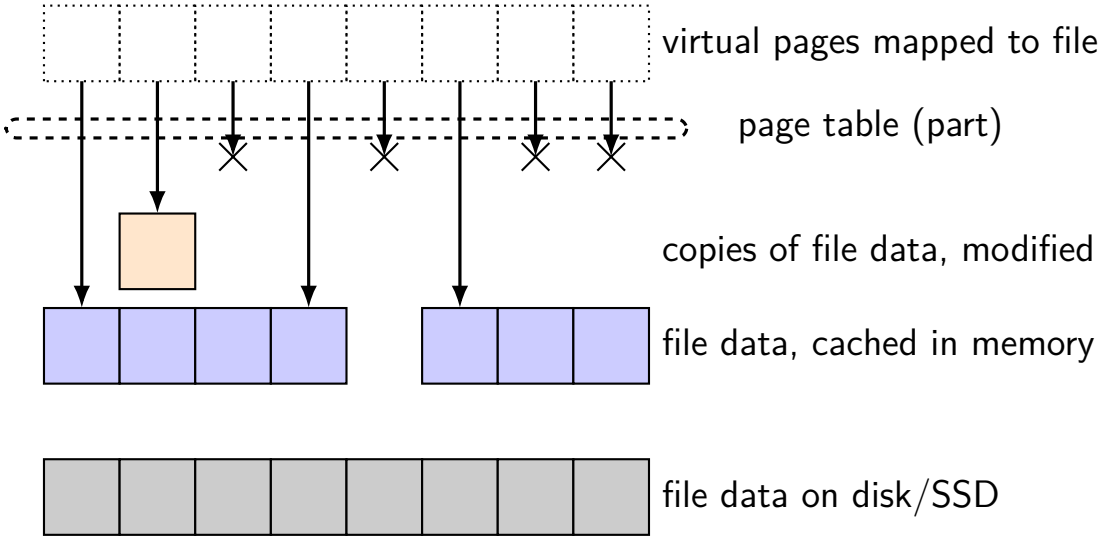
mapped pages (copy-on-write)



mapped pages (copy-on-write)



mapped pages (copy-on-write)

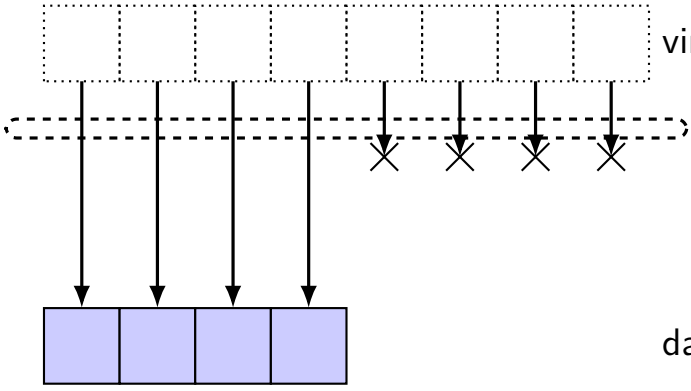


Linux maps

```
$ cat /proc/self/maps
00400000-0040b000 r-xp 00000000 08:01 48328831                           /bin/cat
0060a000-0060b000 r--p 0000a000 08:01 48328831                           /bin/cat
0060b000-0060c000 rw-p 0000b000 08:01 48328831                           /bin/cat
01974000-01995000 rw-p 00000000 00:00 0                                  [heap]
7f60c718b000-7f60c7490000 r--p 00000000 08:01 77483660                   /usr/lib/locale/locale-archive
7f60c7490000-7f60c764e000 r-xp 00000000 08:01                                                    -2.19
7f60c764e000-7f60c784e000 ---p 001be000 08:01                                                    -2.19
7f60c784e000-7f60c7852000 r--p 001be000 08:01                                                    -2.19
7f60c7852000-7f60c7854000 rw-p 001c2000 08:01                                                    -2.19
7f60c7854000-7f60c7859000 rw-p 00000000 00:00 0
7f60c7859000-7f60c787c000 r-xp 00000000 08:01 96659109                   /lib/x86_64-linux-gnu/ld-2.19.so
7f60c7a39000-7f60c7a3b000 rw-p 00000000 00:00 0
7f60c7a7a000-7f60c7a7b000 rw-p 00000000 00:00 0
7f60c7a7b000-7f60c7a7c000 r--p 00022000 08:01 96659109                   /lib/x86_64-linux-gnu/ld-2.19.so
7f60c7a7c000-7f60c7a7d000 rw-p 00023000 08:01 96659109                   /lib/x86_64-linux-gnu/ld-2.19.so
7f60c7a7d000-7f60c7a7e000 rw-p 00000000 00:00 0
7ffc5d2b2000-7ffc5d2d3000 rw-p 00000000 00:00 0                          [stack]
7ffc5d3b0000-7ffc5d3b3000 r--p 00000000 00:00 0                          [vvar]
7ffc5d3b3000-7ffc5d3b5000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
```

heap — no corresponding file
just read/write memory

mapped pages (no backing file)



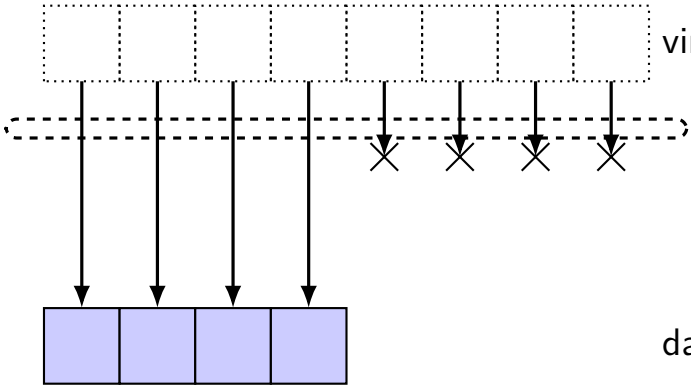
virtual pages w/o backing file

page table (part)

data in memory

swapped out data (if any)

mapped pages (no backing file)



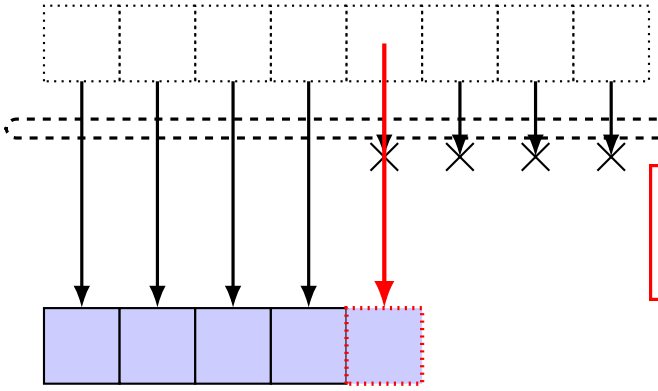
virtual pages w/o backing file

page table (part)

data in memory

swapped out data (if any)

mapped pages (no backing file)



virtual pages w/o backing file

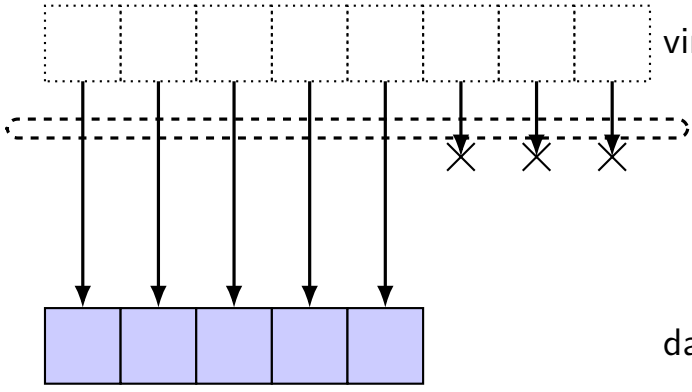
page table (part)

access new page
page fault handler
allocates on demand

data in memory

swapped out data (if any)

mapped pages (no backing file)



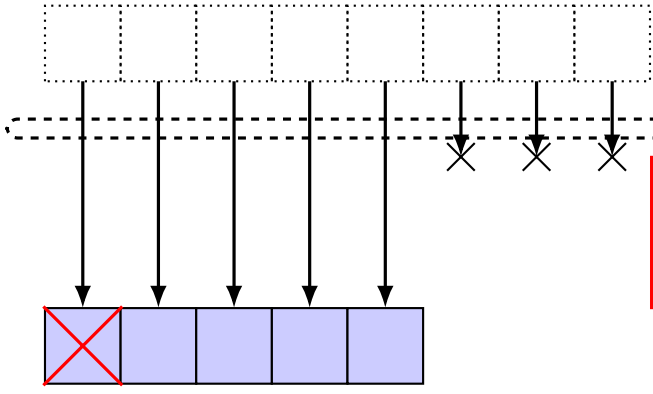
virtual pages w/o backing file

page table (part)

data in memory

swapped out data (if any)

mapped pages (no backing file)



virtual pages w/o backing file

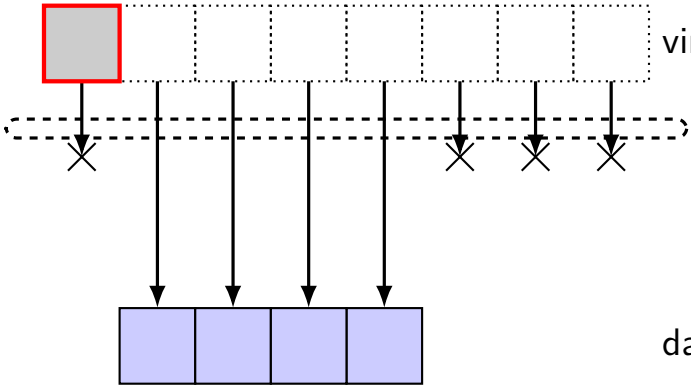
page table (part)

need more memory
save page to disk
temporarily

data in memory

swapped out data (if any)

mapped pages (no backing file)



virtual pages w/o backing file

page table (part)

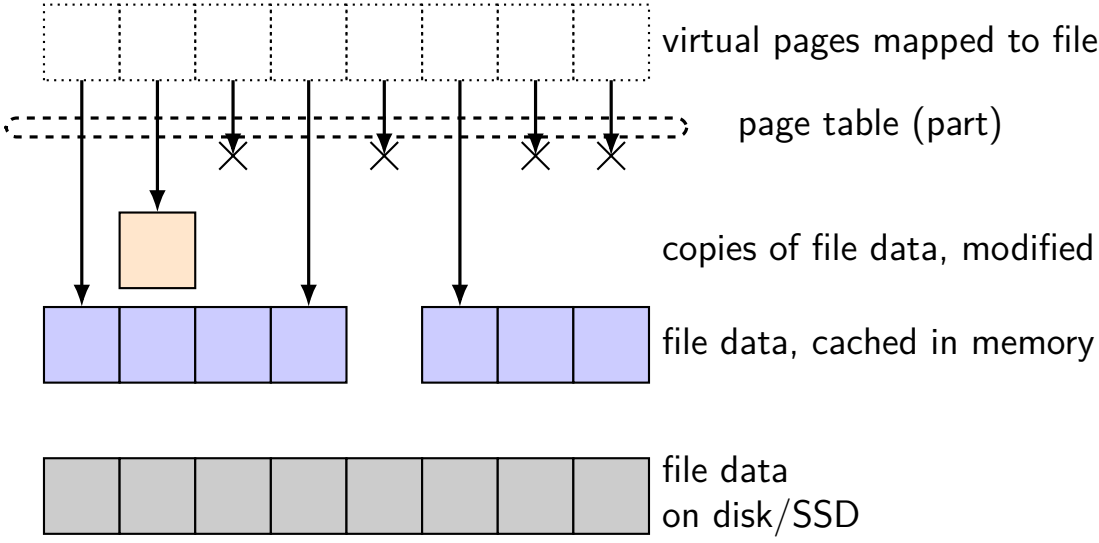
data in memory

swapped out data (if any)

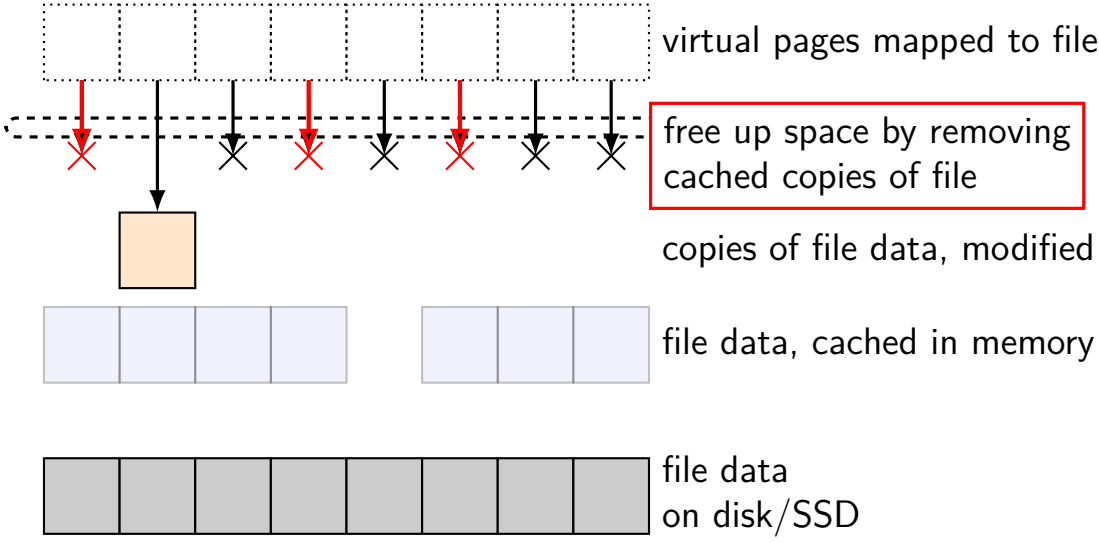
Linux maps

```
$ cat /proc/self/maps
00400000-0040b000 r-xp 00000000 08:01 48328831                           /bin/cat
0060a000-0060b000 r--p 0000a000 08:01 48328831                           /bin/cat
0060b000-0060c000 rw-p 0000b000 08:01 48328831                           /bin/cat
01974000-01995000 rw-p 00000000 00:00 0                                  [heap]
7f60c718b000-7f60c7490000 r--p 00000000 08:01 77483660                   /usr/lib/locale/locale-archive
7f60c7490000-7f60c764e000 r-xp 00000000 08:01 96659129                   /lib/x86_64-linux-gnu/libc-2.19
7f60c764e000-7f60c784e000 ---p 001be000 08:01 96659129                   /lib/x86_64-linux-gnu/libc-2.19
7f60c784e000-7f60c7852000 r--p 001be000 08:01 96659129                   /lib/x86_64-linux-gnu/libc-2.19
7f60c7852000-7f60c7854000 rw-p 001c2000 08:01 96659129                   /lib/x86_64-linux-gnu/libc-2.19
7f60c7854000-7f60c7859000 rw-p 00000000 00:00 0
7f60c7859000-7f60c787c000 r-xp 00000000 08:01 96659109                   /lib/x86_64-linux-gnu/ld-2.19.so
7f60c7a39000-7f60c7a3b000 rw-p 00000000 00:00 0
7f60c7a7a000-7f60c7a7b000 rw-p 00000000 00:00 0
7f60c7a7b000-7f60c7a7c000 r--p 00022000 08:01 96659109                   /lib/x86_64-linux-gnu/ld-2.19.so
7f60c7a7c000-7f60c7a7d000 rw-p 00023000 08:01 96659109                   /lib/x86_64-linux-gnu/ld-2.19.so
7f60c7a7d000-7f60c7a7e000 rw-p 00000000 00:00 0
7ffc5d2b2000-7ffc5d2d3000 rw-p 00000000 00:00 0                          [stack]
7ffc5d3b0000-7ffc5d3b3000 r--p 00000000 00:00 0                          [vvar]
7ffc5d3b3000-7ffc5d3b5000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
```

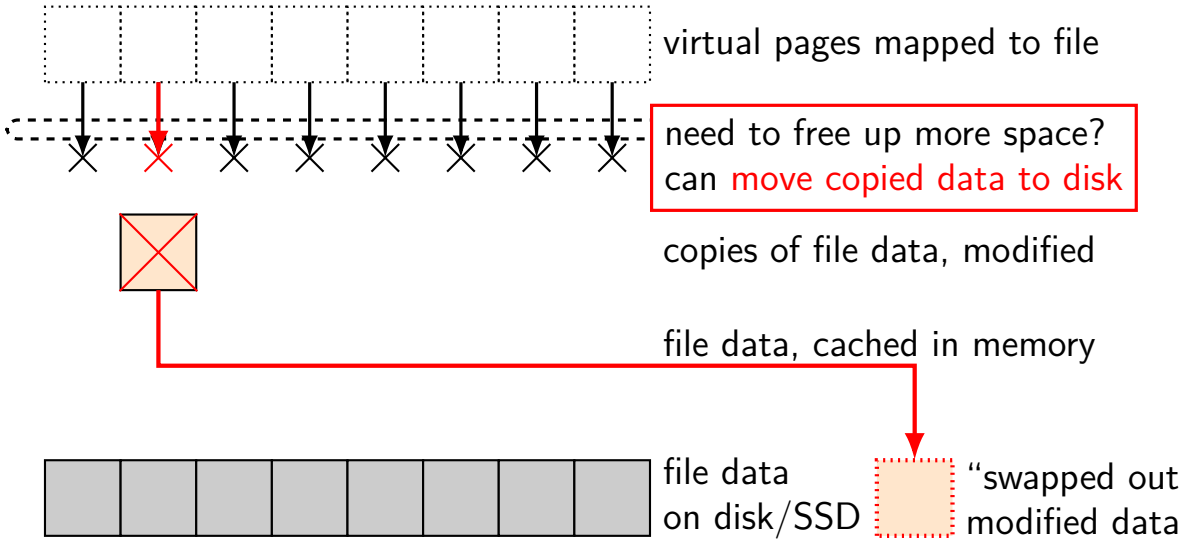
swapping with copy-on-write



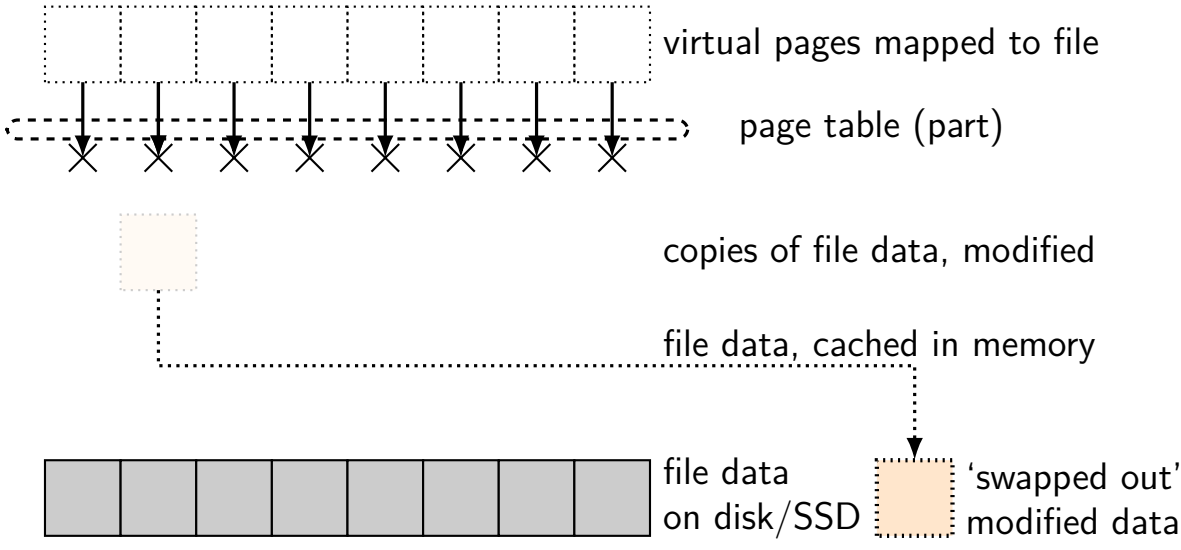
swapping with copy-on-write



swapping with copy-on-write



swapping with copy-on-write



swapping

historical major use of virtual memory is supporting "swapping"

using disk (or SSD, ...) as the next level of the memory hierarchy

process is allocated space on disk/SSD

memory is a cache for disk/SSD

- only need keep 'currently active' pages in physical memory

swapping

historical major use of virtual memory is supporting "swapping"

using disk (or SSD, ...) as the next level of the memory hierarchy

process is allocated space on disk/SSD

memory is a cache for disk/SSD

- only need keep 'currently active' pages in physical memory

swapping $\approx$ mmap with "default" files to use

HDD/SDDs are slow

HDD reads and writes: milliseconds to tens of milliseconds

- minimum size: 512 bytes
- writing tens of kilobytes basically as fast as writing 512 bytes

SSD writes and writes: hundreds of microseconds

- designed for writes/reads of kilobytes (not much smaller)

HDD/SDDs are slow

HDD reads and writes: milliseconds to tens of milliseconds

- minimum size: 512 bytes
- writing tens of kilobytes basically as fast as writing 512 bytes

SSD writes and writes: hundreds of microseconds

- designed for writes/reads of kilobytes (not much smaller)

HDD/SDDs are slow

HDD reads and writes: milliseconds to tens of milliseconds

- minimum size: 512 bytes
- writing tens of kilobytes basically as fast as writing 512 bytes

SSD writes and writes: hundreds of microseconds

- designed for writes/reads of kilobytes (not much smaller)

the page cache

memory is a cache for disk

files, program memory has a place on disk

- running low on memory? always have room on disk
- assumption: disk space approximately infinite

physical memory pages: disk 'temporarily' kept in faster storage

- possibly being used by one or more processes?
- possibly part of a file on disk?
- possibly both

goal: manage this cache intelligently

the page cache

memory is a cache for disk

files, program memory has a place on disk

- running low on memory? always have room on disk
- assumption: disk space approximately infinite

physical memory pages: disk 'temporarily' kept in faster storage

- possibly being used by one or more processes?
- possibly part of a file on disk?
- possibly both

goal: manage this cache intelligently

memory as a cache for disk

"cache block" $\approx$ physical page

fully associative

- any virtual address/file part can be stored in any physical page

replacement is managed by the OS

normal cache hits happen without OS

- common case that needs to be fast

page cache components [text]

mapping: virtual address or file+offset → physical page

- handle cache hits

find backing location based on virtual address/file+offset

- handle cache misses

track information about each physical page

- handle page allocation
- handle cache eviction

page cache components