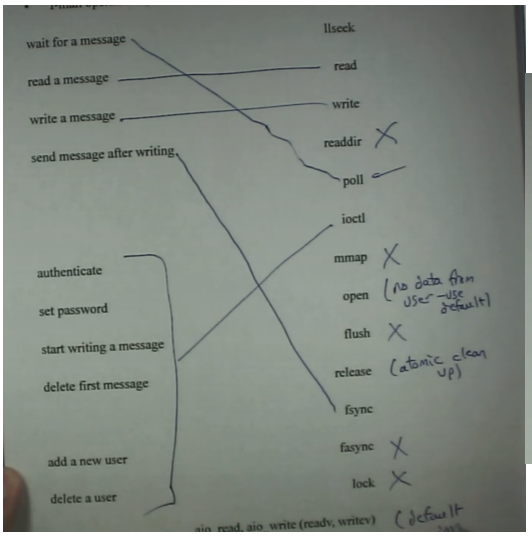
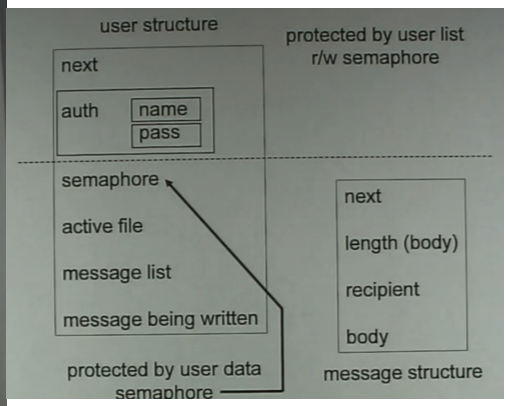
- Block devices (underlies file systems)
  - data accessible only in blocks of fixed size, with size determined by device
  - can be addressed randomly
  - transfers to/from device are usually buffered (to) and cached (from) for performance
  - examples: disks, CD ROM, DVD

- Character device:
  - contiguous space (or spaces) of bytes
  - some allow random access (e.g., magnetic tape driver)
  - others available only sequentially (e.g., sound card)
  - examples: keyboard, terminal, printers



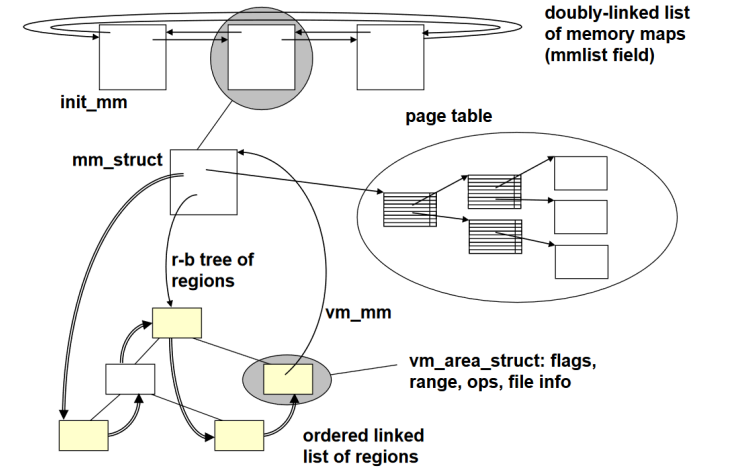


- What can block?
  - read — wait for a new message
  - poll — wait for a new message
  - write — typically needs to wait for the device, but not in I-mail
  - (so only readers/pollers block in I-mail)

- When are readers (or pollers) awoken?
  - new message delivered
  - user is deleted (nothing is coming now!)

- Protocol for sleeping
  - release all locks
  - check conditions without locks
  - go to sleep
  - when awoken
    - reacquire locks
    - recheck all validity requirements for waking (otherwise go back to sleep; false alarm)

- a few small items → kmalloc
- a lot of items, repeatedly → slab cache
- a big, physically contiguous region → free pages
- a big area of virtual memory → vmalloc
  (not necessarily physically contiguous)

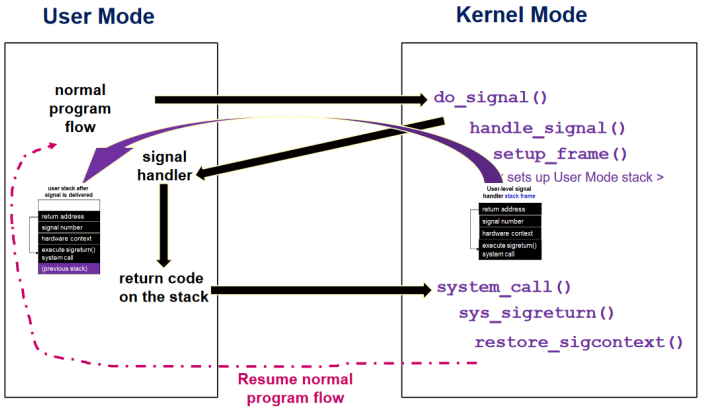


Similarities
- asynchronous w.r.t. program execution
- not queued – like a physical line; sending twice may cause handler to execute once or twice
- can be ignored, blocked, or caught
- but has some NMIs (SIGKILL, SIGSTOP)
- handler can be changed from default
- only data given to handler is signal # (traditional model)
- signal is (by default) blocked while handler executes

Differences
- generated by software (kernel or a program via a system call)
- no "device" associated with signal; only software with permission can send a signal (single permission allows any signal to be sent)
- further differences with POSIX (Portable Operating System Interface) model (as opposed to traditional model)

When are signals delivered? (see entry.S)

- check **`sigpending`** (in task structure) when returning from
  - any interrupt
  - any exception
  - any system call
- only deliver to currently executing process



Permission check for signal generation

- sysadmin (or kernel, or privileged call) can always send
- process with same user id can always send
- process with same login session can send SIGCONT

If generated signal is not being ignored

- it is added to pending signals
- a sleeping recipient is then woken up
  (kicked out of wait queues, for example)