Unit 6

Q. 6.1

Demand paging:

- Machines whose memory architecture is based on ~~to~~ pages & whose CPU has restartable instruction can support a kernel that implements a demand paging algorithm, swapping pages of memory bet^n main memory and swap device.

Machine supports

- Demand paging systems free processes from size limitations until imposed by physical memory available on machine.

Size limitation

eg: Machines that contain 1 or 2 MB of physical memory can execute process of size are 4 or 5 MB

Virtual size limit

- But Kernel still imposes a limit on virtual size of process, dependent on the amount of virtual memory the machine can address

- Processes tends to execute instructions in small portions of their text space, such as program loops & frequently called subroutines & their data references tend to cluster in small subsets of total data space of process. This is known as principle of 'locality'

principle of locality

Working set & window of working set

- The working set of process is set of pages that the process has referenced in its last 'n' memory references; the number 'n' is called 'window' of working set

- When a process address a page that isn't in working set, it incurs page fault; in handling the fault, the kernel updates the working set, reading in pages from secondary device if necessary

- Data Structures for Demand Paging.
  - Kernel contains 4 major DS to support low-level memory management f^n's & demand paging.

//Shortcut: PPDS

① Page frame data table (pfdata table)
- Describes each page of physical memory. indexed by page no.
- Kernel allocates space for pfdata table once, for lifetime of system but allocates memory pages for other structures dynamically.

② Page table entries:
- Each entry contains physical address of page, protection bits (which indicate read, write or execute from page) and following bits to support demand paging:
  - Valid
  - Reference
  - Modify
  - Age
  - Copy on write

③ Disk Block Descriptors: (DBD)
- Each page table entry is associated with DBD, which describes disk copy of virtual page.
- Indicates 'demand fill' or 'demand zero'.

④ Swap Use Table:

- This table contains an entry for every page on swap device.

- The entry consists of reference count of how many page table entries point to pages on swap device.

- The hw supplies the kernel with virtual address that was accessed to cause the memory fault, & kernel finds the page table entry & disk block descriptor of the page.

The page that caused the fault is in one of five states:
1. On a swap device & not in memory,
2. On the free page list in memory.
3. In an executable file,
4. Marked "demand zero".
5. Marked "demand fill".

Algorithm: vfault /* handler for validity fault
input: address where the process faulted.
output: none.
{
find region, page table entry, disk block descriptor corresponding to faulted address, lock region;
if (address outside virtual space) {
send signal (: segmentation violation) to process;
goto OUT;
}
if (address now valid) goto OUT;
if (page in cache) {
remove page from cache; adjust page table entry;
while (page contents not valid)
sleep (event contents become valid);
}
else { // page not in cache
assign new page to region;
put new page in cache, update pfdata entry;
if (page not previously loaded & page "demand zero")
clear assigned page to 0;
else {read virtual page from swap dev / exec file;
sleep (event I/O done);
}

recalculate process priority,

OUT: unlock region;

}

G.4

Q4 Allocation of ~~of~~ space on swap device?

what is swap device

- Swap device is block device in configurable section of disk.

Cont. block allocation

- The kernel allocates space for files one block at a time. It allocates space on swap device in group of continuous blocks without any fragmentation.

swap differs from file system DS

- As allocation of swap device differs from allocation of scheme for file system, the DS that catalog the free space differ too.

defn: map

- Kernel maintains free-space for swap-device in an "in-core table" called a map.

map's other resource use

- Maps, used for other resources besides the swap device, allows first-fit allocation of contiguous "blocks" of a resource.

map exp

- A map is an array where each entry consists of address of an allocatble resource & no. of resource available there. The kernel interprets address & unit according to type of map.

map's initial

- Initially a map contains one entry that indicates the address & total no. of resources.

a) Address Units
[1 10000]
Initial swap map

=> b) Address Units
[101 9900]
After allocating 100 units

c) Address Units
[151 9850]
After allocating 50 units

- As the kernel allocates the free resources, it updates the map so that it ~~can~~ continues to contain accurate information about resources.

Algorithm: malloc
Input: i> map address, i> Requested no. of units.
Output: Address (if successful), else 0.

```
{
  for (every map entry) {
      if (current map entry can fit requested units)
        {
          if (requested units == no. of units in entry)
              delete entry from map;
          else adjust start address of entry;
          return (original address of entry);
        }
  }
  return (0);
}
```

6.5

Q5. Explain f^n's of line discipline & dists.

» ∘ Terminal drivers:

- Has same f^n like other drivers: control transmission of data 'to' & 'from' terminals.
- They are special because they are UI to system.
- They contain internal interface to line discipline module, which interprets input & output.

* - In canonical mode, the line discipline converts raw data seq typed at keyboard to canonical form (what user really meant) before sending data to receiving process.
* - Also converts raw output seq written by process to format that user expects.

- In 'raw mode', the line discipline passes data bet^n processes & terminals without such conversion.

The functions of Line Discipline are:

input 1. ∘ to pass input strings into lines
erase 2. ∘ to process erase characters.
kill 3. ∘ to process 'kill' character that invalidates all chars typed so far in current line.
write to terminal 4. ∘ to write received chars to terminal.
expand 5. ∘ to expand output (such as tab chars to seq. of blank spaces)
signal 6. ∘ to generate signals to process for terminal hangups, line breaks, or in response to user hitting delete key.
raw 7. ∘ to allow raw mode, that doesn't interpret special characters such as erase, kill or carriage return.

6.5

Q5- Explain f°s of line discipline of dists

» ◦ Terminal drivers:

- Has same f° like other drivers: control transmission of data 'to' & 'from' terminals.
- They are special because they are UI to system.
- They contain internal interface to line discipline module, which interprets input & output.

* - In canonical mode, the line discipline converts raw data seq typed at keyboard to canonical form (what user really meant) before sending data to receiving process.
* - Also converts raw output seq written by process to format that user expects.

- In 'raw mode', the line discipline passes data bet° process & terminals without such conversion.

The functions of Line Discipline are:

input 1 ◦ pass input strings into lines
erase 2 ◦ to process erase characters.
kill 3 ◦ to process 'kill' character that invalidates all chars typed so far in current line.
write 4 ◦ to write received chars to terminal.
expand 5 ◦ to expand output (such as tab chars to seq. of blank spaces)
signal 6 ◦ to generate signals to process for terminal hangups, line breaks, or in response to user hitting delete key.
raw 7 ◦ to allow raw mode, that doesn't interpret special characters such as erase, kill or carriage return.

- Line discipline could be required by not only terminal but other processes as well.
- ∴ Kernel puts it in correct place.

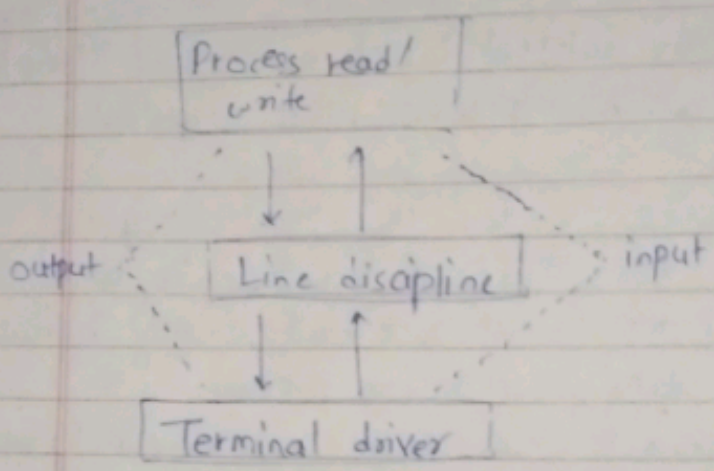

fig: Logical flow of data.

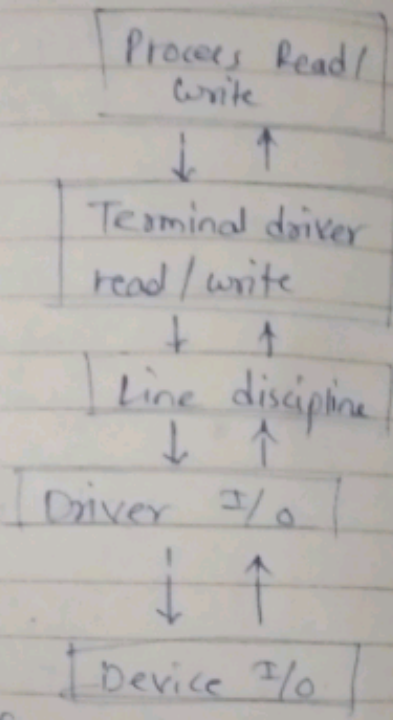

fig: flow of control.

* CLists

- Line discipline manipulates data on clists
- clist (character list) is variable length linked list of 'cblocks' with a count of no. of characters on list.

- Cblock contains pointer to next cblock, small char array for data, & set of offsets indicating the position of valid data in the cblock.