PAGE NO.:
DATE: / /

OS Tutorial - 8.

Q.1. Explain algorithm for allocating space from maps (malloc) with example.

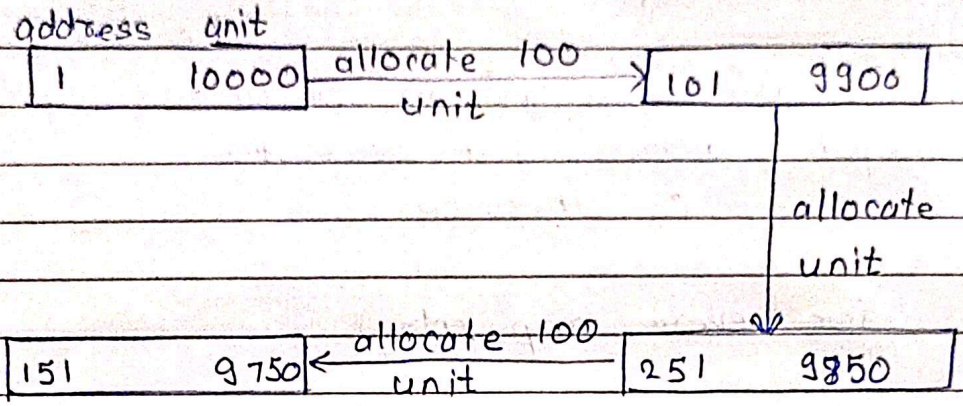
→

```
algorithm malloc /* to allocate map space */
input: (1) map address    /* indicates which map
                              to use */
       (2) requested no. of units.
output: address, if successful
        0, otherwise
{
  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);
}
```

The kernel maintains the space for swap device in an in-core table, called map. Initially a map contains one entry that indicates the address & total no. of resources. The kernel searches the map for the first entry that contains enough space to accommodate the request.
If the request consumes all the resources of map entry, the kernel removes the entry from.

array & compresses the map.
Otherwise, it adjusts the address & unit fields of the entry ~~acc~~ according to amount of resources allocated.

Ex –



Q.2. Explain freeing swap space with example.

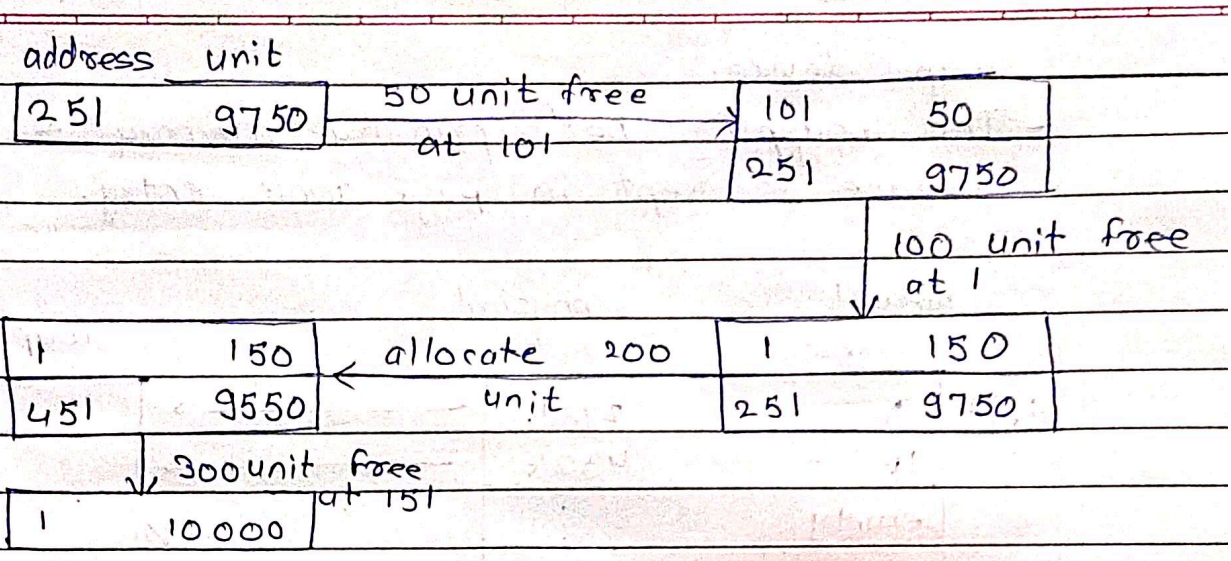
→ For freeing swap space ~~three~~ cases are possible:-

① The freed resources completely fill a hole in the map. The kernel combines the newly freed resources and the existing entries into one entry in ~~table~~ map.

② The freed resources partially ~~feel~~ fill a hole in map. The kernel adjusts the address & unit fields of appropriate entry. The no. of entries in map remains same.

③ The freed resources partially fill a hole but are not contiguous to any resources in the map. The kernel creates new entry for the map.

Example –



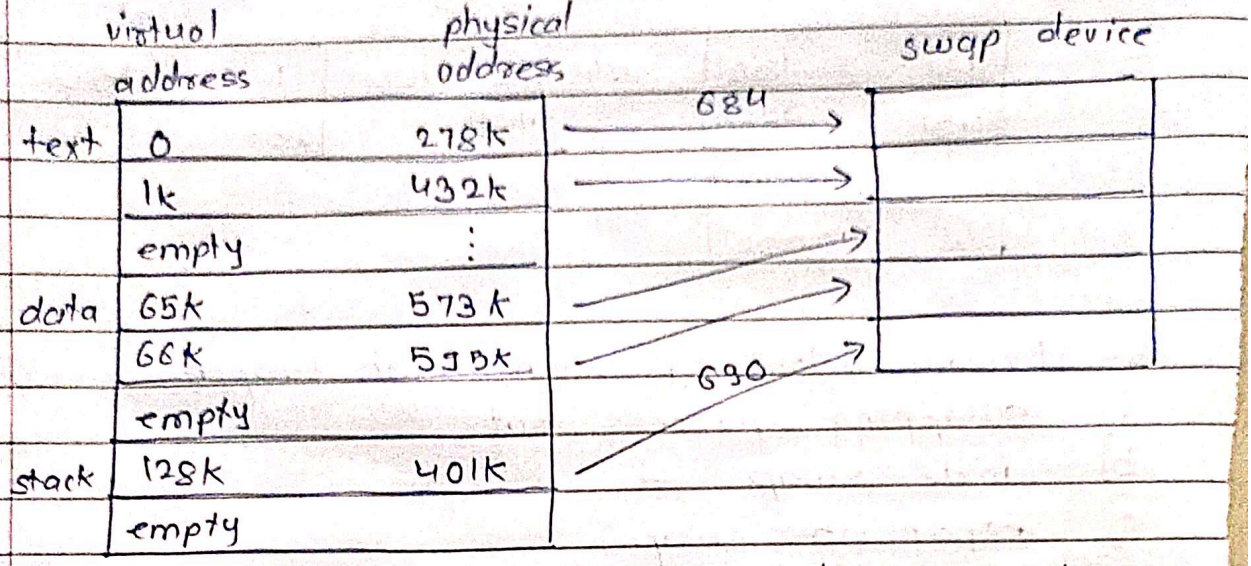
Q.3. Explain below operations with swapper algorithm.
a] swapping process out
b] fork swap
c] expansion swap
d] swapping process in.

→ a] swapping process out

IMP - The kernel swaps a process out if it needs space in memory, which may result in:
① The fork system call must allocate space for a child process
② The brk system call increases the size of process
③ A process becomes larger by the natural growth of its stack
④ The kernel wants to free space in memory for processes it had previously swapped out & should now swap in.

- The kernel must gather the page addresses of data at primary memory to be swapped out.
- Kernel copies the physical memory assigned to a process to the allocated space on the

swap device.
- The mapping bet" primary memory & swap device is kept in page table entry.



Mapping process space onto swap device.

b] fork swap.

→ - The fork system call assumes that parent process found enough enough memory to create the child context. Otherwise, the kernel swaps the process out.
- When swap is complete, the child process exists on the swap device; the parent places the child in ready-to-run state.
- When the kernel schedules, the child will complete its part of fork system call & return to user mode.

c] expansion swap

→ - It reserves enough space on the swap device to contain the memory space of the process, including the newly requested space.

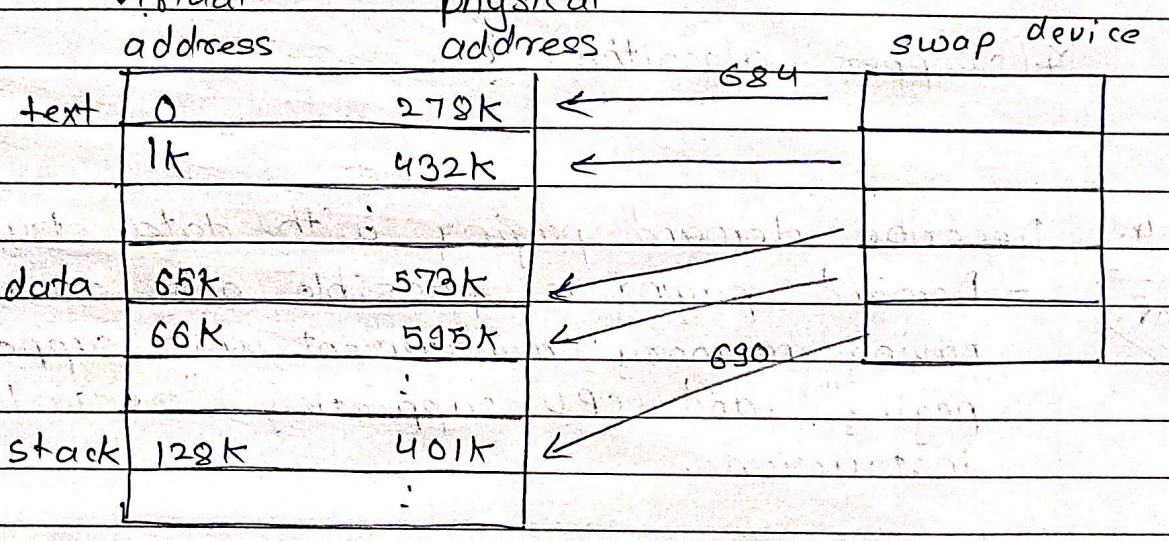
PAGE NO.:
DATE: / /

- Then it adjusts address translation mapping of the process.

- Finally, it swaps the process out on newly allocated space in swapping device.

- When the process swaps the process into memory, it will allocate physical memory according to new address translation map.

d] IMP → swapping process in



swapping a process into memory.

- Process 0, the swapper, is the only process that swaps processes into memory from swap devices.
- The swapper sleeps if there is no work for it to do & the kernel periodically wakes it up.
- The kernel schedules the swapper to execute just as it schedules other processes, but the

PAGE NO.:
DATE : / /

swapper executes only in kernel mode.
- When the swapper wakes up to swap processes in, it examines all processes that are in the state "ready to run but swapped out" & selects one that has been swapped out the longest.
- If there is enough free memory available, the swapper swaps the process in.
- If a process successfully swaps in a process, it searches the set of "ready to run but swapped out" processes for others to swap in & repeats the above procedure.

& (swapper algorithm) ?

Q.4. Describe demand paging with data structure.

IMP →

- Demand paging is possible only on machines having memory management unit supporting "paging" and CPU supporting "restartable" instructions.

- Swapping pages of memory reside bet^n main memory & swap device.

- Demand paging systems free processes from size limitation. For ex - a machine having 1M of physical memory can run process of size more than 1M.

- The kernel still imposes a limit on virtual size of process.

PAGE NO.:
DATE: / /

- When a process accesses a page that is not part of its working set, it incurs a page fault.

- The kernel suspends the execution of the process until it reads the page into memory.

- Process tend to execute instr^n in small portion of text space, their data references tends to cluster in small subset of total data space of process. This is known as locality.

- The working set of process is the set of pages that process has referred in its last "n" memory references. The number "n" is called as window of working set.

- A larger working set demands more pages in main memory.

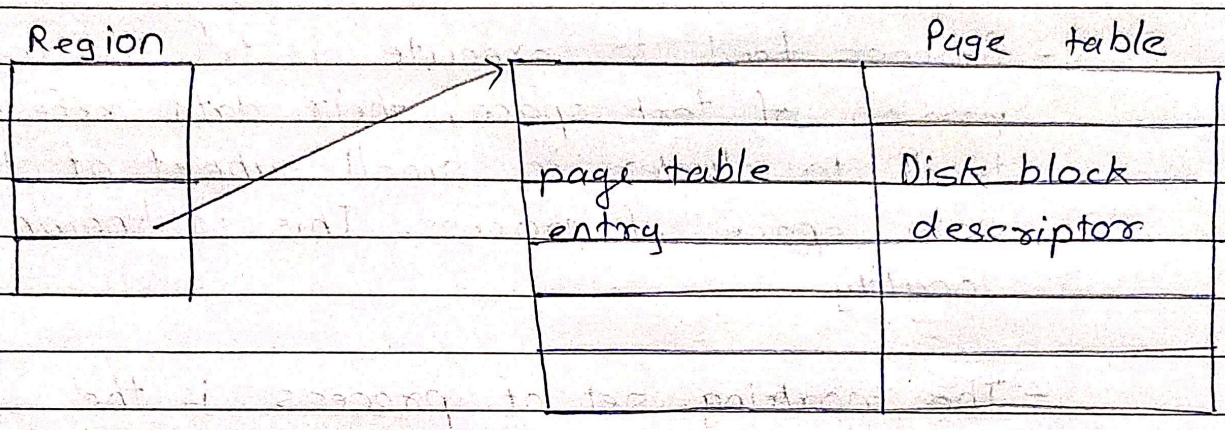
* Data structure

The kernel contains 4 major data structures to support low-level mm & demand paging -

① page table entries
② disk block descriptor
③ page frame data table
④ swap use table.

- The kernel allocates space for the page frame data table once for the lifetime of system.

- Each page table entry contains physical address of page.



Page table entry

Page address	age	cp	wrt	mod	ref	val	prot

Disk block descriptor

swap device	block num	Type

- Page table entry contains the physical address of page & following bits:
  1. valid - whether page content is valid
  2. reference - whether the page is referenced recently
  3. modify - whether page content is modified
  4. age - age of page
  5. protection - read/write permission
  6. copy of write

PAGE NO.:
DATE : / /

- The pfdata table describes each page of physical memory & is indexed by page number. The fields of entry are:-
  ① The page state indicating that the page is on swap device.
  ② No. of processes that reference the page.
  ③ logical device & block no.
  ④ pointers to other pfdata table entries.

Q.5. Write a short note on:
a) driver interface
b) terminal driver
c) streams.