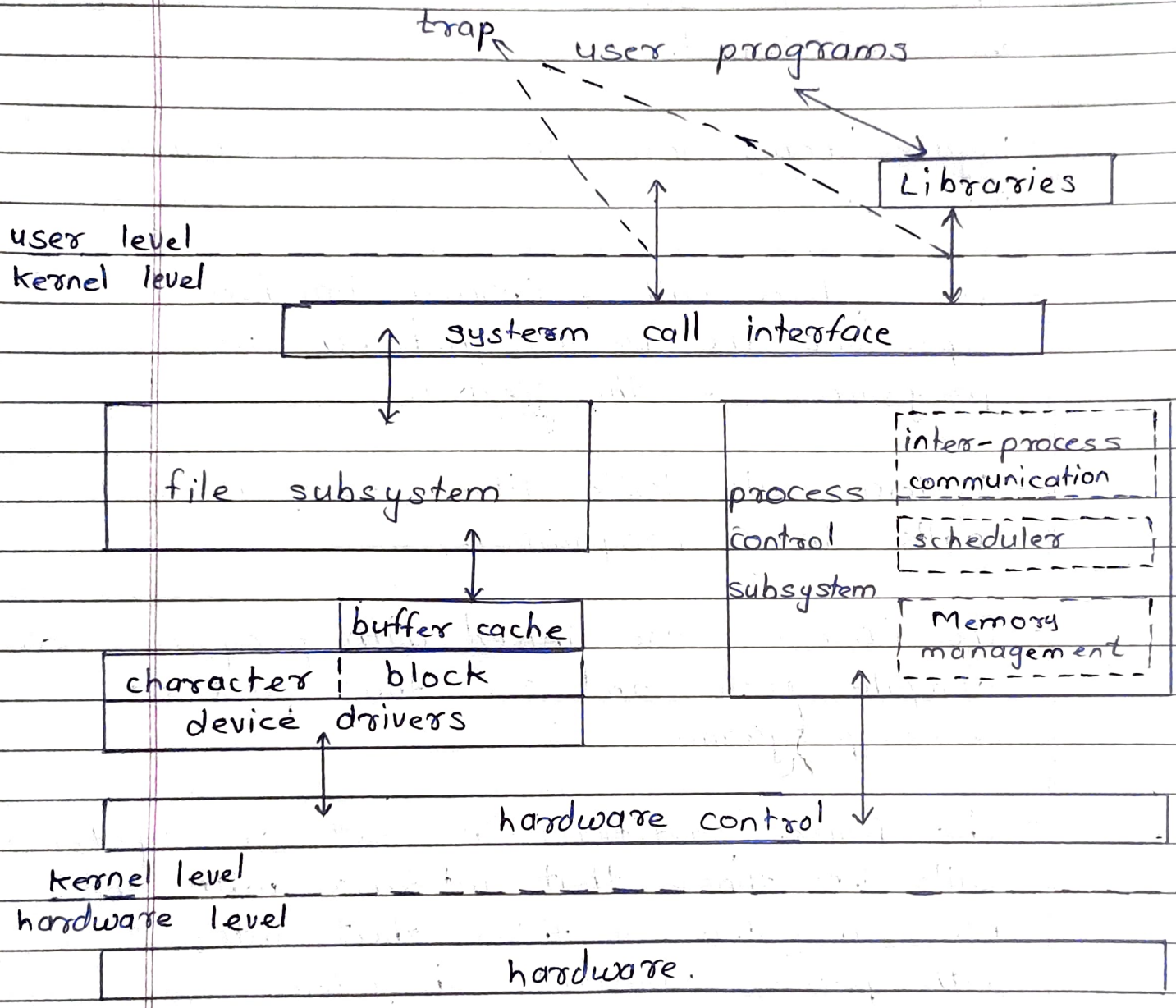
Page:
Date: / /

Q.1. Draw & explain the architecture of unix operating system.

→



- The two entities, files & processes, are two central concepts in the UNIX system model.

- Figure shows block diagram of kernel, various modules & their relationships to each other.

- It shows the file subsystem on the left

and the process control subsystem on right, the two major components of kernel.

- The diagram serves as a useful logical view of kernel.

- Figure shows three levels: user, kernel and hardware.

- The system call & library interface represent the border between user programs & the kernel.

- System calls look like ordinary function calls in C programs.

- The libraries are linked with programs at compile time and are thus part of user program.

- The figure partitions the set of system calls into those that interact with the file subsystem & those that interact with process control subsystem.

- The file subsystem manages files, allocating file space, administering free space, controlling access to files and retrieving data for users.

- Processes interact with file subsystem via a specific set of system calls such as open, close, read, write, state, chown

chmod.

- The process control subsystem is responsible for process synchronization, interprocess communication, memory management & process scheduling.

- The memory management module controls the allocation of memory.

- The scheduler module allocates the CPU to processes. It schedules them to run.

- The hardware control is responsible for handling interrupts and for communicating with machine.

Q.2. Describe briefly user perspective of UNIX system.

→ i] The file system -

-The UNIX file system is characterized by -

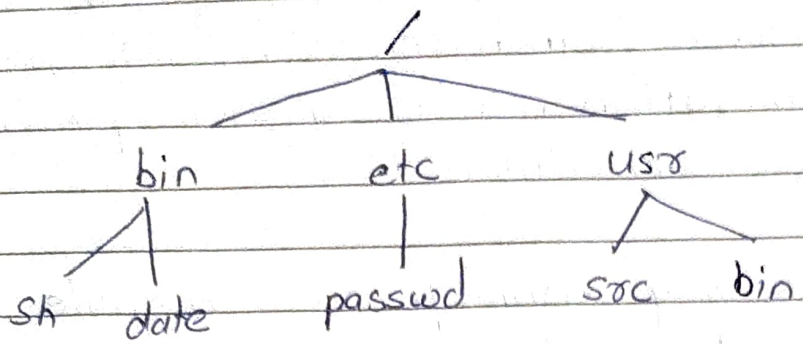
① a hierarchial structure
② consistent treatment of file data.
③ the ability to create & delete files
④ dynamic growth of files
⑤ the protection of file data.

- The file system is organized as a tree with a single root node called root (written "/")

- Every non-leaf node of file system structure is a directory of files and files at

leaf nodes of tree are either directories, regular files or special device files.

-



- A path name is sequence of component names separated by slash characters.

- A full path name starts with a slash character and specifies a file that can be found by starting at the file system root and traversing the file/tree.

- Permission to access a file is controlled by access permissions associated with the file.

- Access permissions can be set independently to control read, write & execute permission for three classes of users:- the file owner, a file group & everyone else.

2] Processing environment:

- A program is an executable file and a process is an instance of the program in execution.

- Many processes can execute simultaneous

on UNIX systems with no logical limit to their number and many instances of a program can exist simultaneously in the system.

- Various system calls allow processes to create new processes, terminate processes, synchronize stages of process execution, and control reaction to various events according to appropriate system call.

3) Building block primitives -
- The philosophy of UNIX system is to provide operating system primitives that enable users to write small, modular programs that can be used as building blocks to build more complex programs.

- Processes conventionally have access to three files -
① they read from their standard input file
② they write to their standard output file
③ write error messages to their standard error file.

- Processes executing at a terminal typically use the terminal for these three files but each may be redirected independently.

Page :
Date : / /

Q. 4. Describe briefly file subsystem of UNIX system.

→ - The set of system calls is partitioned into those that interact with the file subsystem and those that interact with process control subsystem.

- The file subsystem manages files, allocating file space, administering free space, controlling access to files and retrieving data for users.

- Processes interact with the file subsystem via a specific set of system calls open (to open a file for reading or writing), close, read, write, stat (query the attributes of a file), chown (change the ~~reo~~ record of who owns the file), and chmod (change the access permissions of a file).

- The file subsystem accesses file data using a buffering mechanism that regulates data flow between the kernel and secondary storage devices.

- The buffering mechanism interacts with I/O devices drivers.

- The file subsystem also interacts with raw I/O device drivers without buffering mechanism.

Q. 5. Draw and explain data structure for processes

→ - Every process has an entry in the kernel process table, and each process is allocated a u area (user area) that contains private data manipulated only by the kernel.

- The process table contains a per process region table, whose entries point to entries in a region table.

- A region is a contiguous area of process's address space, such as text, data & stack.

- Region table entries describe the attributes of the region, such as whether it contains text or data, whether it is shared or private and where the data of the region is located in memory.

- The extra level of indirection (from per process region table to region table) allows independent processes to share regions.

- When a process invokes the exec system call, the kernel allocates regions for its text, data & stack after freeing the old regions the process had been using.

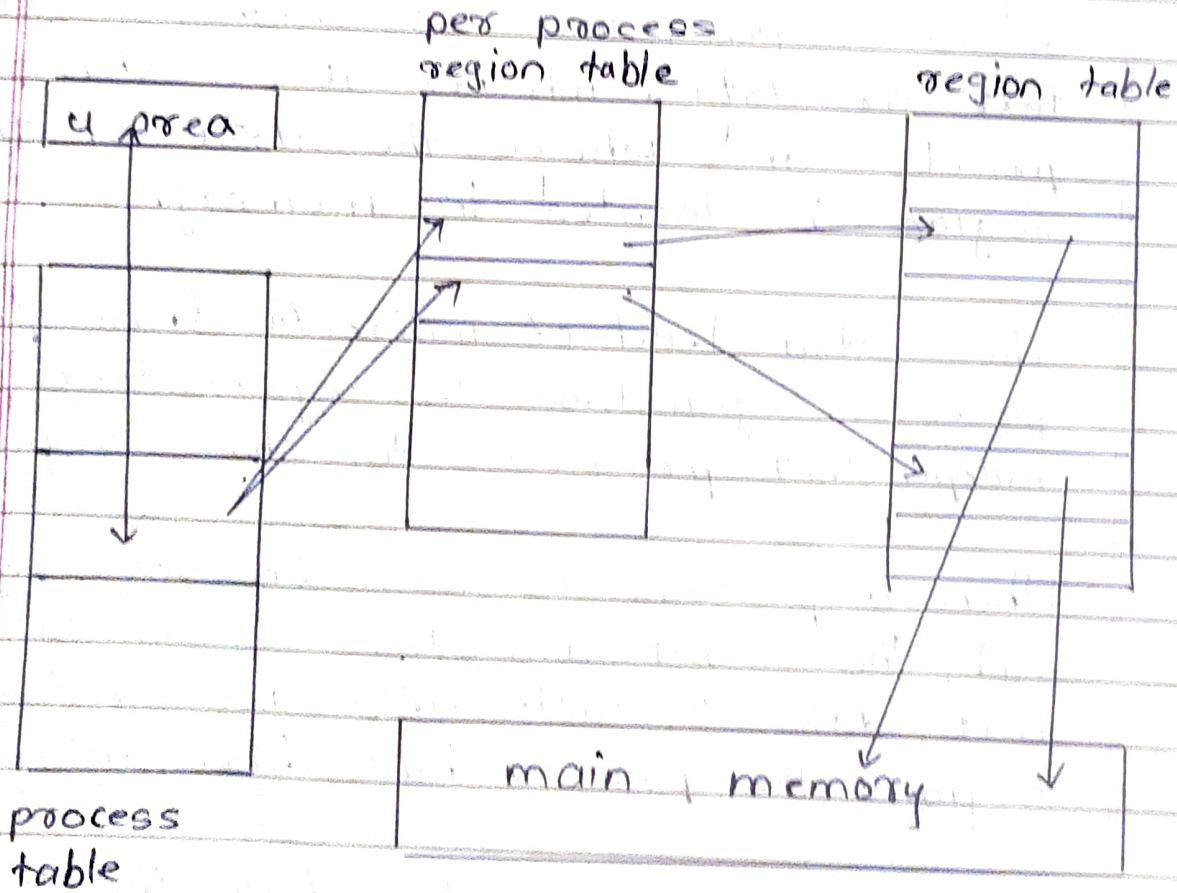
- When a process invokes fork, the kernel duplicates the address space of the old process, allowing processes to share regions

when possible & making a physical copy otherwise.

- When a process invokes exit, the kernel frees the regions the process had used.

- The process table points to a per process region table with pointers to the region table entries for the text, data & stack regions of process.

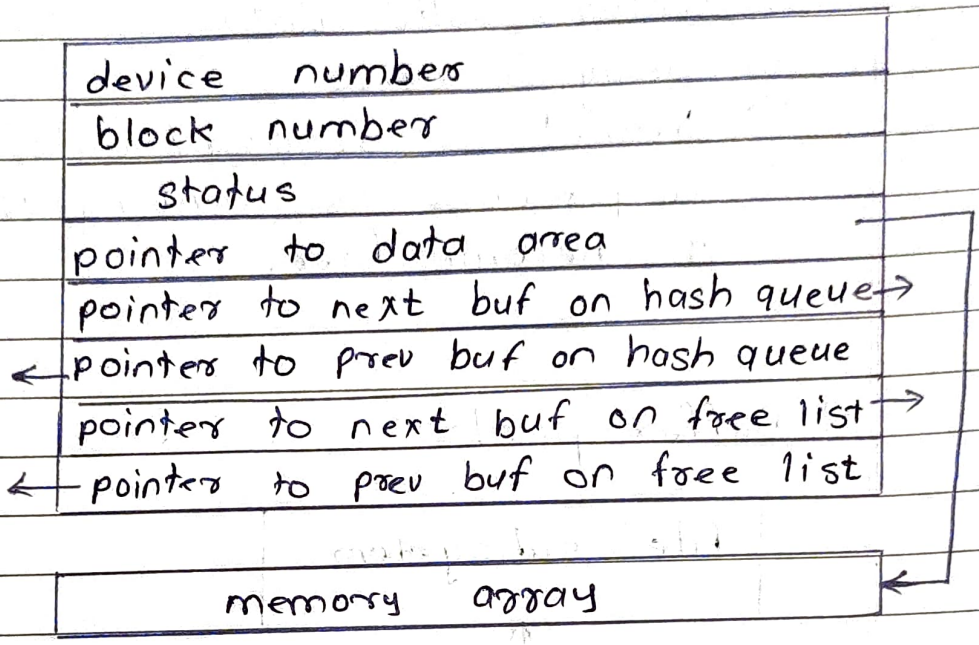
- The process table entry and the u area contain control & status information about the process.



Q.6. Draw & explain the structure of buffer pool & buffer header.

→

i] Buffer headers.



- A buffer consists of two parts: a memory array that contains data from disk and a buffer header that identifies the buffer.

- The data in a buffer corresponds to data in a logical disk block on a file system, and the kernel identifies the buffer contents by examining identifier fields in the buffer header.

- The contents of the disk block map into the buffer, but the mapping is temporary.

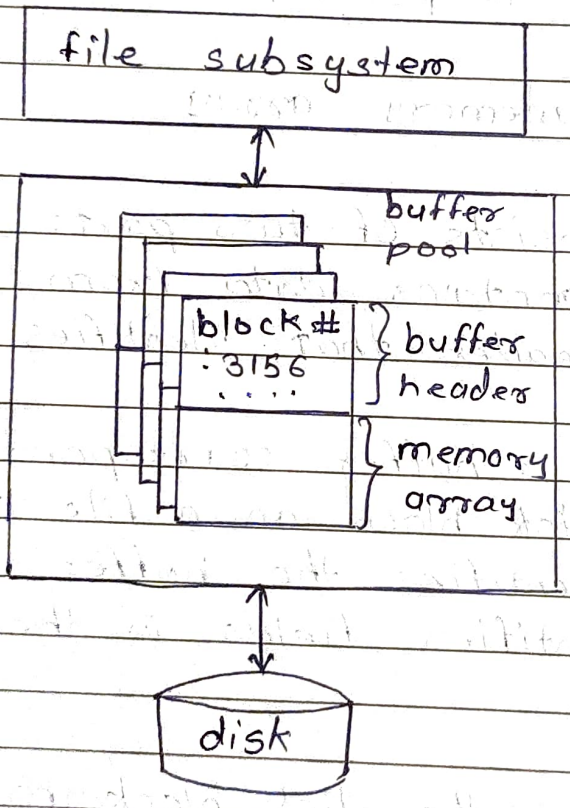
- A disk block can never map into more than one buffer at a time.

- The device number is the logical file system

number.

- The status of buffer is combination of :-
  1. locked
  2. contains valid data
  3. delayed-write
  4. kernel is reading / writing
  5. a process is waiting for the buffer to become free.

2] Buffer pool -



- The kernel catches data in the buffer pool according to a least recently used (LRU) algorithm.

- The kernel maintains a free list of buffers

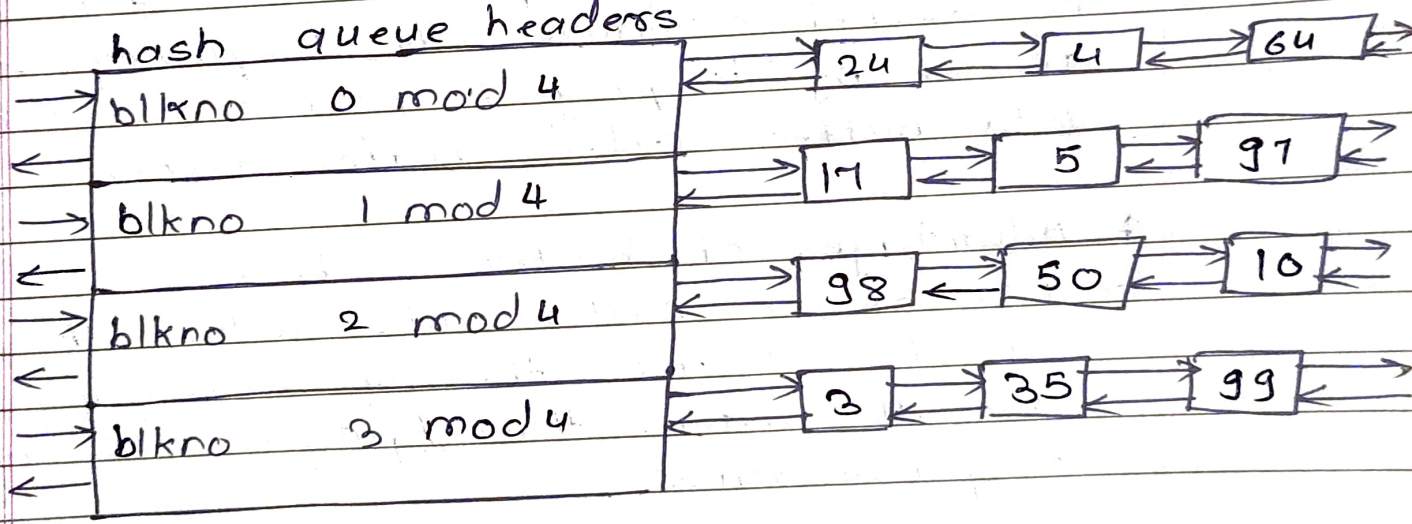
that preserve

- The kernel takes a buffer from the head of the free list for any free buffer.

- The kernel returns a buffer to the tail of the free list.

- The kernel links the buffers on a hash queue into a circular, doubly linked list, similar to structure of free list.

- Ex - f(blkno) = blkno mod 4;



- Each buffer always exists on a hash queue.

- A buffer may be simultaneously on a hash queue and on the free list if its status is free.

- The kernel has two ways to find buffer :-
① It searches the hash queue if it is

looking for a particular buffer.
② It removes a buffer from the free list if it is looking for any free buffer.

Q.7. Define buffer cache. Give advantages & disadvantages.

→ *buffer cache -

- When a process wants to access data from a file, the kernel brings the data into main memory, alters it and then request to save in the file system.

- To increase the response time & throughput, the kernel minimizes the frequency of disk access by keeping a pool of internal data buffer called buffer cache.

* Advantages of buffer cache -

- use of the buffer cache reduce the amount of disk traffic, thereby increasing overall system throughput & decreasing response time.

- The buffer algorithms help insure file system integrity.

* Disadvantages of buffer cache -

- For delayed write, the system is vulnerable to crashes that leave disk data in an incorrect state.

- use of buffer cache requires an extra

data copy when reading & writing to & from user processes.

Q.8. Explain algorithm for reading a disk block or explain bread() algorithm.

→ algorithm bread /* block read */
input : file system block number
output : buffer containing data.
{
get buffer for block (getblk);
if (buffer data valid)
return buffer;
initiate disk read;
sleep (event disk read complete);
return buffer;
}

To read a disk block, a process uses algorithm getblk to search for it first in buffer cache. If it is available in the cache the kernel can return it immediately instead of physically reading the block from disk.

If it is not present in cache, the kernel calls the disk drive to schedule a read request & goes to sleep awaiting the event that the I/o completes.

The disk driver informs the disk controller hardware that it wants to read data & the disk controller then transmits the data to buffer.

The disk interrupt handler awakens the sleeping process and the contents of disk block are now in buffer.

Q. 9. Explain algorithm for writing a disk block or explain bwrite() algorithm.

→ algorithm bwrite /* block write */
input: buffer
output: none
{
initiate disk write;
if (I/O synchronous) {
sleep (event I/O complete);
release buffer (brelse);
} else if (buffer marked for delayed write)
mark buffer to put at head of free list;
}

- If the write is synchronous, the calling process goes to sleep awaiting I/O completion and release the buffer when it awakens.

- If the write is asynchronous, the kernel starts the disk write but does not wait for the write to complete.

- If the kernel does delayed write, it ~~makes~~ marks the buffer accordingly, releases the buffer using algorithm brelse, and continues without scheduling

Page:
Date: / /

I/O.

- The kernel writes the block to disk before another process can reallocate the buffer to another block.

Q.10. List the scenarios ~~of~~ for retrieval of buffers and explain the getblk() algorithm & brelse() algorithm.

→

* scenarios for retrieval of buffer.

① The kernel finds the block on its hash queue and its buffer is free.

② The kernel cannot find the block on the hash queue, so it allocates a buffer from the free list.

③ The kernel connot find the block on the hash queue and in attempting to allocate a buffer from the free list, finds a buffer marked "delayed write".

④ The kernel cannot find the block on the hash queue, and the free list is empty.

⑤ The kernel finds the block on the hash queue, but its buffer is currently busy.

Page:
Date: / /

Q. * algorithm for buffer allocation

```
algorithm getblk
input : file system number
        block number
output : locked buffer
{
while (buffer not found) {
    if (block in hash queue) {
        if (buffer busy) { /* scenario 5 */
            sleep (event buffer becomes free);
            continue;
        } }
    mark buffer busy; /* scenario 1 */
    remove buffer from free list;
    return buffer;
    } else {
        if (there are no buffers on free
            list) { /* scenario 4 */
            sleep (event any buffer becomes
                free);
            continue;
        }
    remove buffer from free list;
    if (buffer marked for delayed write)
        { */scenario 3 */
        asynchronous write buffer to disk;
        continue;
    }
    /* scenario 2 */
    remove buffer from old hash queue;
    put buffer onto new hash queue;
    return buffer;
} }
}
```

- The kernel leaves the buffer marked busy; no other process can access it.
- When the kernel finishes using the buffer, it releases the buffer according to algorithm brelse.
- It wakes up processes that had fallen asleep.
- The kernel places the buffer at the end of the free list.

* Algorithm for releasing a buffer.

```
algorithm brelse
input: locked buffer
output: none
{
wakeup all procs : event, waiting for any
                   buffer to become free;
wakeup all procs : event, waiting for this
                   buffer to become free;
raise processor execution level to block
      interrupts;
if (buffer contents valid & buffer not old)
    enqueue buffer at end of free list;
else
    enqueue buffer at beginning of free list;
lower processor execution level to allow
interrupts;
unlock (buffer);
}
```

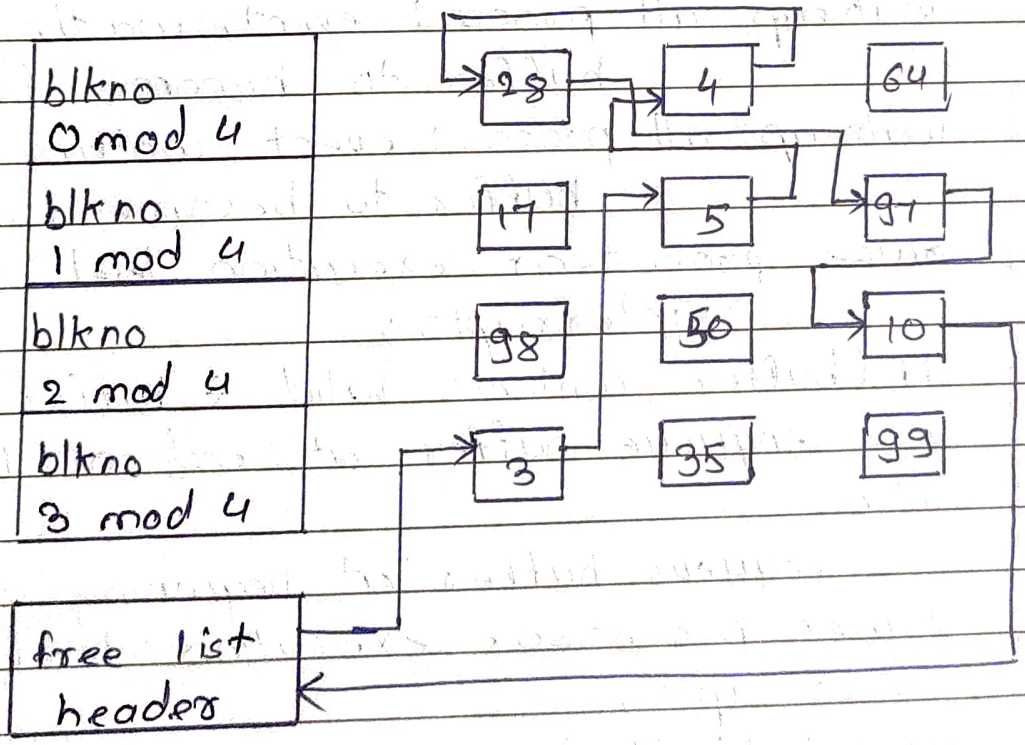
- The kernel also invokes brelse when a handling a disk interrupt to release buffers used for asynchronous I/O.

- The kernel raises the processor execution level to prevent disk interrupts when manipulating the free list.

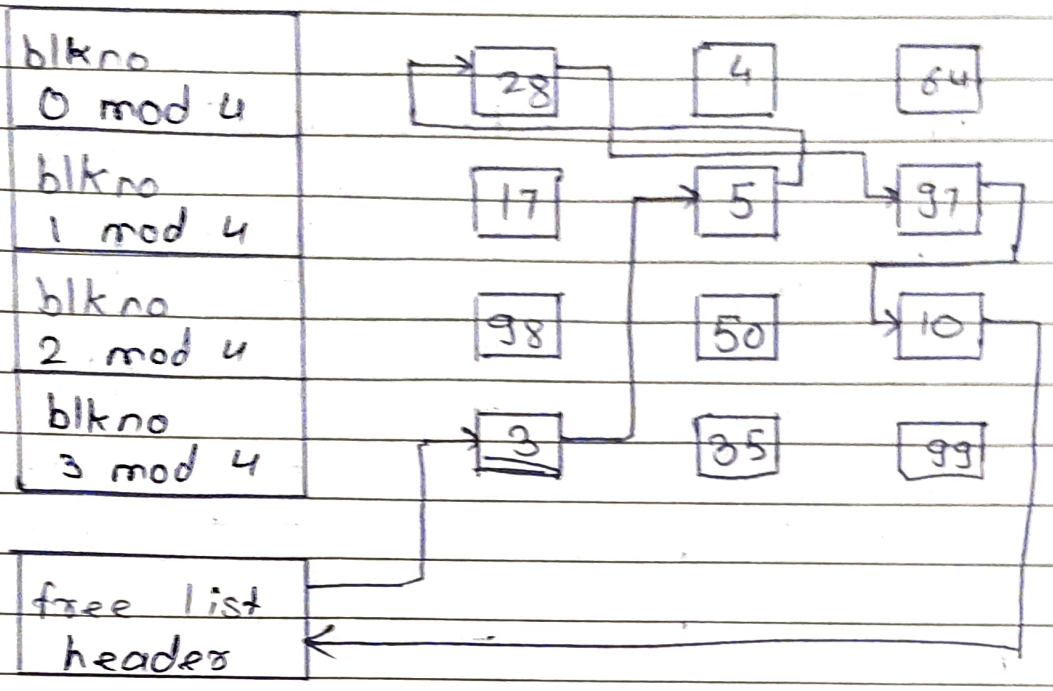
Q.11. Describe scenarios for retrieval of buffer with diagram.

→ i] The kernel finds the block on its hash queue and its buffer is free.

search for block 4 on first hash queue

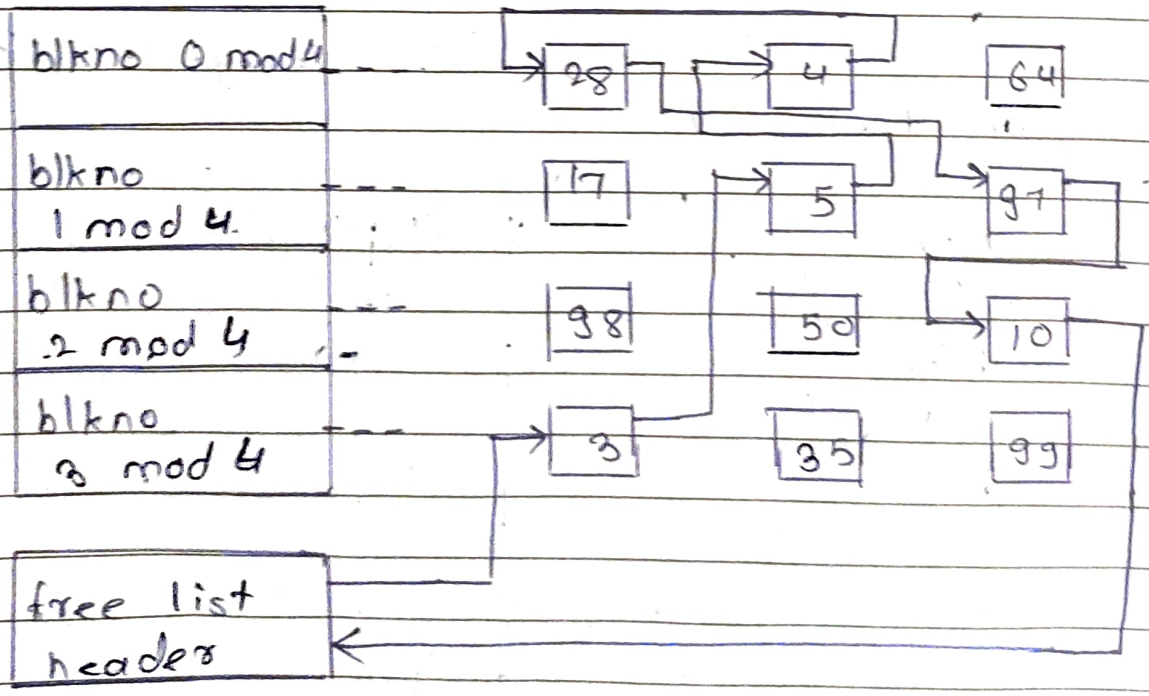


remove block 4 from free list

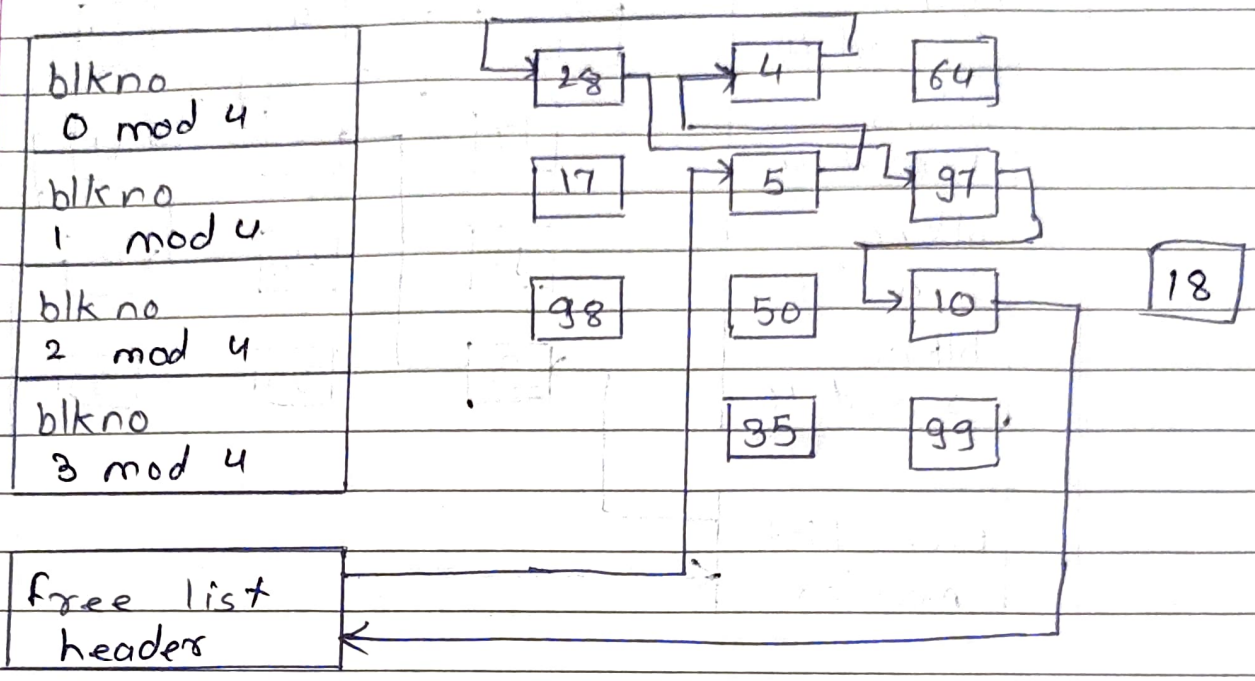


2] The kernel cannot find the block on the hash queue, so it allocates a buffer from the free list.

- search for block 18 - not in cache

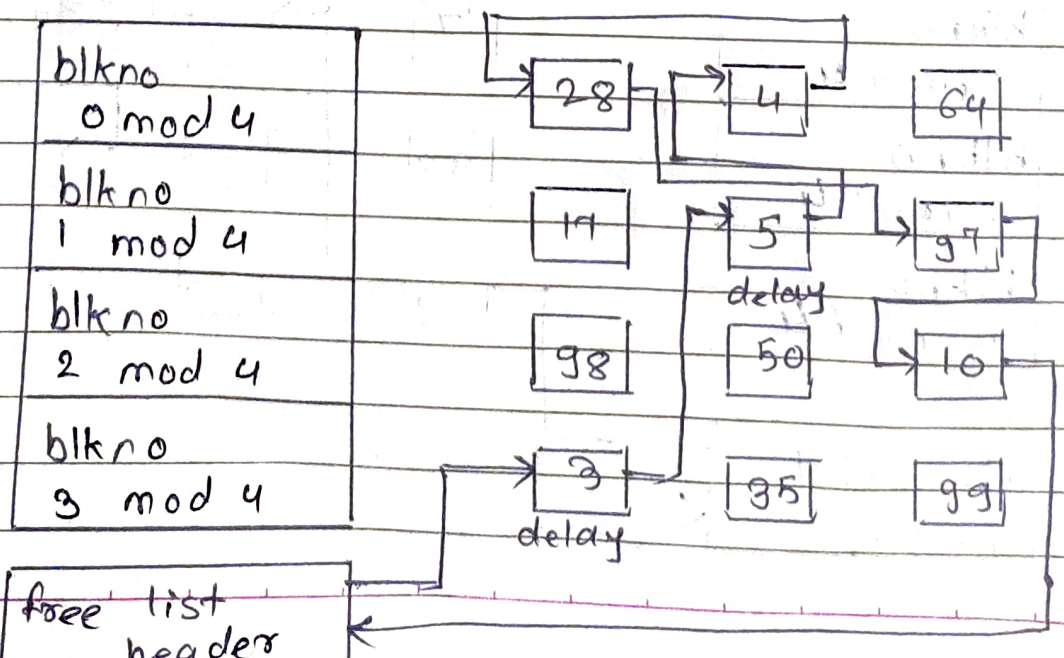


remove first block from free list, assign 18. to 18.

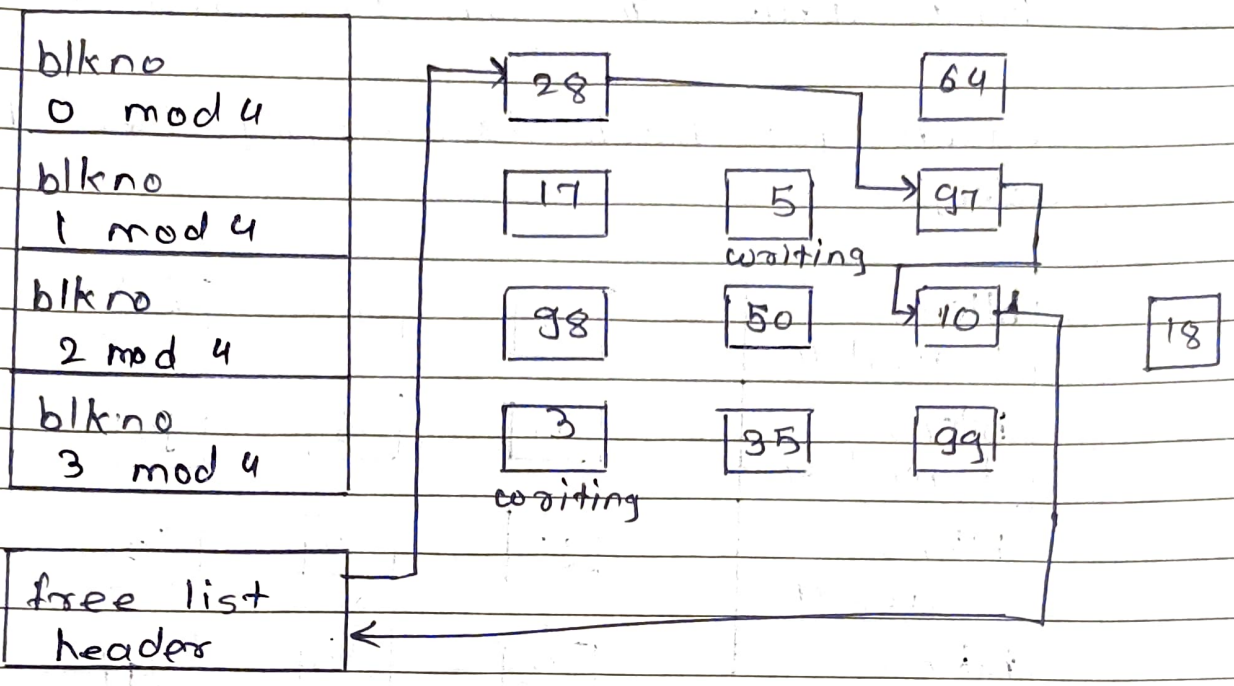


3] The kernel cannot find the block on the hash queue and in attempting to allocate a buffer from the free list, finds a buffer marked "delayed write".

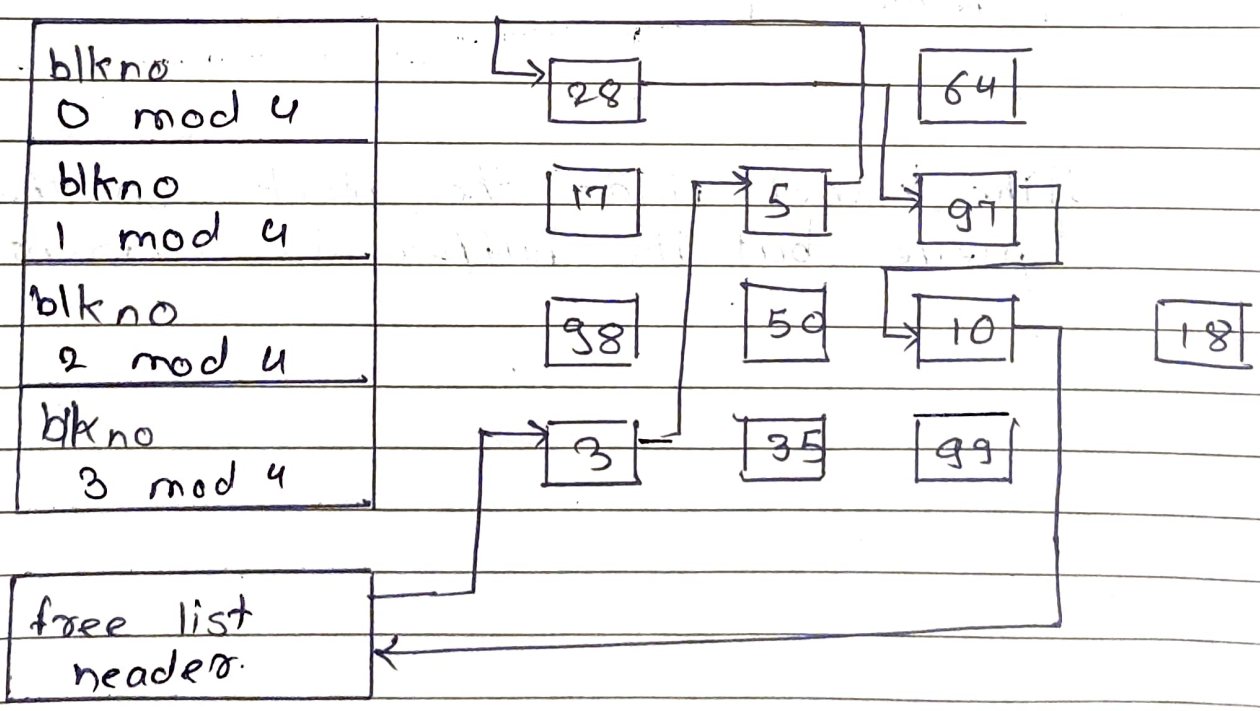
search for block 18 - delayed write block on free list:



waiting blocks 3 & 5, reassign 4 to 18.

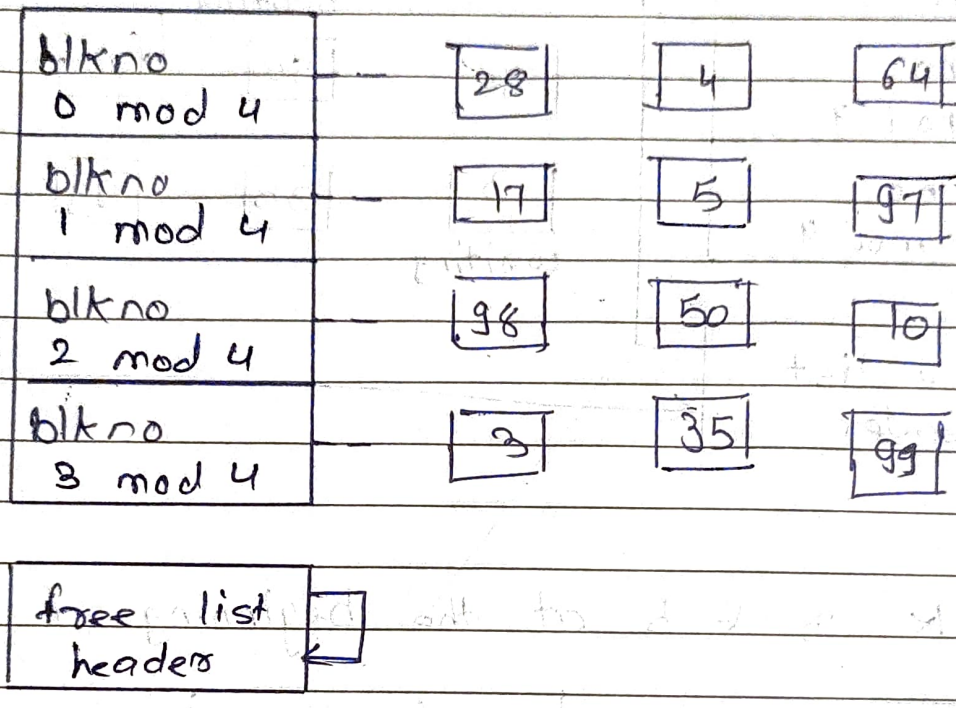


block 3 & 5 at the beginning of free list.



4] The kernel cannot find the block on the hash queue, and the free list is empty.

search for block 18, empty free list → asleep



5] The kernel finds the block on the hash queue, but its buffer is currently busy.

Page :
Date : / /

Q.12. Draw the structure of buffer pool & free list header in which mod 5 hash function is used.

Q.13. Explain race condition for free buffer & locked buffer.

Q.14. Draw block diagram of Unix system kernel. Explain file subsystem and process control subsystem.

or

with neat diagram of system kernel, explain architecture of Unix operating system.