Name: Anish Mishra
Batch: f-s
Enrollment number: 9920103138

PAGE 01
DATE 29-11-2022

Tutorial-9

File Structure

Sol-1) The main problem if a system allowed a file system to be mounted at more than one location is the existence of multiple paths to the same file. Therfore there would be different ways to similar records, which could confuse users. This situation can also lead to emergence of unecassary errors.

Sol-2) Contiguous: If the file is usally accessed sequentially, if the file is Relatively Small.

Linked: If the file is large and accessed sequentially.

Indexed: If the file is large and accessed randomly.

Sol-3) Dynamic tables allows more flexibility in system use growth - tables are never exceeded, avoiding artificial use limits. Unfortunately, kernel structure and code are more complicated so there is more potential for bugs. The use of one resource can take away more system resources (by growing to accomodate the requests) than with static tables.

Sol-4) VFS layer introduces a layer of indirection in the file system implementation. In many ways, it is similar to object oriented programming techniques. System calls can be made generically (independent of file system type). Each file system type provide its function calls and data structures to the VFS layer. A system call is translated into the proper specific functions for the target file system at the VFS layer. The calling program

has no file system specific code, and the upper levels of system call structures likewise are file system-independent. The translation at the VFS layer turns these generic calls into file system specific operations.

S.1-5)

	Contiguous	Linked	Indexed
A	201	1	1
B	101	52	1
C	1	1	1
D	198	1	0
E	98	52	0
F	0	100	0

S.1-6) Caches allows components of differing speed to communicate more efficiently by storing data from slower device temporarily, in a faster device (cache). Caches, almost by definition, more expensive than device they are caching for, so increasing the number of size of caches would increase system cost.

S.1-7) This method requires more overhead than that of standard contiguous allocation. It requires less overhead than the standard linked allocation.

S.1-8) (a) The block is added in the middle:

Contiguous: Assume that in the middle means after block 75 and before block 76. We move the last 75 blocks down

one position and then write in the new block.

$75(1r+1w)+1w = 75r+76w = 151$ I/O operations

Linked: We cannot find block 75 without traversing the linked list stored in the first 74 data blocks. So, we first read through these 74 blocks. Then, we read block 75, copy it's link into the new block (in main memory), update block 75's link to point to the new block, write out block 75, write new block.

$74r+1r+1w+1w = 75r+2w = 77$ I/O operations.

Indexed: Update the index in main memory. Write the new block.

$1w = 1$ I/O operation.

(b) The block is removed from the beginning.

Contiguous: Simply change the starting address to 2.

0 I/O operations.

Linked: Read in block 1 and change the starting address to the link stored in this block.

$1r = 1$ I/O operation

Indexed: Simply remove the block address from the linked list in the index block.

0 I/O operations

Sol-9) Let z be the starting file address (Block number)

(a) Contiguous. Divide the logical address by 512 with X and Y the resulting quotient and remainder respectively.

i) Add X to Z to obtain the physical ~~memory~~ block number. Y is the displacement into that block.

ii) 1

(b) The number of blocks that must be read from the disk is,

⇒ Contiguous : 1