Instructions:

1. **Please write the answer for each question ONLY in the space provided below the question.**
2. **Please note, merely writing the answer without justification/ reason (wherever explicitly asked to give reason) will not fetch FULL marks.**

[2] 1. Given five memory partitions of 100 KB, 500 KB, 200 KB, 300 KB, and 600 KB (in order), how would each of the first-fit, best-fit, and worst-fit algorithms place processes of 212 KB, 417 KB, 112 KB, and 426 KB (in order)? Which algorithm makes the most efficient use of memory?

2

Solution:

head

Initial: (100kB) → (500kB) → (200kB) → (300kB) → (600kB) → (NULL).

FirstFit: Final free space (along w/ allocations):-

(100) free → (212) (alloc) → (112) (alloc) → (176) free → (200) free → (300) free → (417) (alloc) → (183) free

Here, 426kB can't be allocated a contiguous chunk without coalescing partitions.

BestFit: 212kB process gets space from 300kB partition
417kB process gets space from 500kB partition
112kB process gets space from 200kB partition
426kB process gets space from 600kB partition

⇒ Free space: (100) → (83) → (88) → (88) → (174) → NULL

WorstFit: Final free space (along w/ allocations):-

(100) free → (417) (alloc) → (83) free → (200) free → (300) free → (212) (alloc) → (112) (alloc) → (276) free

Here, 426kB can't be allocated a contiguous chunk without coalescing partitions.

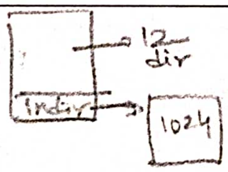
Note that $\boxed{\text{Best Fit}}$ makes most efficient use of memory.

[1] 2. An inode has a fixed number of direct pointers (12), and a single indirect pointer. If a file grows large enough, an indirect block is allocated (from the data-block region of the disk), and the inode's slot for an indirect pointer is set to point to it. Assuming each slot can point to a 4-KB block, and that disk addresses are 4 bytes, the file can grow to be __________.

1

Solution:

Number of disk addresses per block $= \frac{4kB}{4B} = \boxed{1024}$



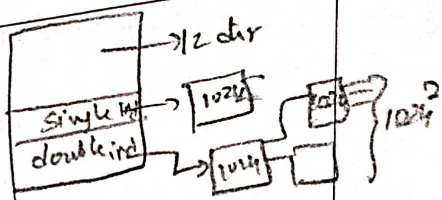
So, single-indirect pointer can point to $\boxed{1024}$ disk blocks, while we also have $\boxed{12}$ direct pointers.

So, Total file size (max) $= (12+1024) \times 4KB = (1036) \times 4KB = \boxed{4144kB}$ Ans

$\simeq \boxed{4MB + 48kB}$

$1036 \times 4 = 4144$

1 3. An example files system with twelve direct pointers, as well as both a single and a double indirect block. Assuming a block size of 4 KB, and 4-byte disk addresses, such a file system can have a max file size of____________(approximately).

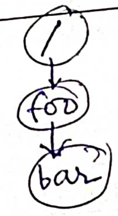


Solution: Similar to Question(3) of this quiz,
#Blocks pointed to directly = 12
#Blocks pointed to by single indirect = 1024
#Blocks pointed to by double indirect = 1024^2
$\Rightarrow$ Total (Max) file size = $(12+1024+1024^2)\times 4kB$ = $\boxed{48kB + 4MB + 4GB}$
$\approx \boxed{4198448kB}$

1

1 4. Let us assume that you want to simply open a file /foo/bar, read it, and then close it. In doing so, the file system will read ____________inodes. Explain your answer.

Solution:
open: read root inode, read foo inode, read bar inode
read: read bar inode
close: read bar inode



$\Rightarrow$ 5 inodes ~~are read~~

-0

1 5. Once /foo/bar is opened, assume a process appends a data block to it three times. The following things will be written to during each append: __________, __________, and __________. [assume no buffering of writes]

Solution:
~~→ read inode of bar~~
→ write to Data block of bar
→ write to inode of bar (last accessed time)
→ write to ~~inode,~~ datablock bitmap

1

1 6. (a) What are the advantages of an inode-based file system design compared to FAT?
(b) Why do we have direct blocks? Why not just have indirect blocks?

Solution:
(a) - Inode-based file system can easily accommodate hard and soft linking.
- ~~FAT~~ Inode-based file system is less bulky. ~~with substantially~~

(b) Indirect blocks are not always required for files, as they are too huge. ~~Single indirection alone, increases~~ Each level of indirection leads to an almost-exponential increase in size of file. Simply put, most files are small. Only a small proportion of files are large enough to require indirection. (Direct blocks immediately point ~~out~~ to the data of a block ~~at a disk address~~.)
Hence, for small ~~blocks~~ files, indirect blocks would be overkill.

1

7200 revol → 1 min = 60s
120 revol → 1 s = 1000ms

1000
3.00

8.3
12) 100
96
40

[3] 7. Assume we have a hard drive with the following specifications:

- An average seek time of 8 ms $t_{seek} = 8\,ms$
- A rotational speed of 7200 revolutions per minute (RPM) $t_{rot} = \frac{60 \times 1000\,ms}{7200\,revol} = \frac{600}{72}\,ms/revol = \frac{100}{12}\,ms/revol$
- A controller that can transfer data at a maximum rate of 50 MB/s

(a) What is the expected throughput of the hard drive when reading 4 KB sectors from a random location on disk?

Solution: $t_{seek} = 8ms$, $t'_{rot} = 8.3ms \rightarrow 1$ revol. Transfer rate $= 50MB/s$.

For 4kB sectors, (random) Transfer-time $= \frac{4kB}{50\,MB/s} = \frac{4}{50} \times \frac{1}{1024}\,s = 8 \times 10^{-2} \times 10^{-3}\,s \simeq 8 \times 10^{-5}\,s \simeq 0.08s$

$\simeq 0.08s$.

~~Throughput~~ on average, $\frac{1}{2}$ rotation is required to reach destination $\Rightarrow t_{rot} = 4.16ms$

So, Throughput $= \frac{4KB}{8 + 4.16 + 0.08\,(ms)} = \frac{4}{12.24}\,kB\,ms^{-1} = \frac{4000}{12.24}\,kB/s$

$\simeq \boxed{333\,kB/s}$ (approx) ±10%

Ans.

(b) What is the expected throughput of the hard drive when reading 4 KB sectors from the same track on disk (i.e., the read/write head is already positioned over the correct track when the operation starts)?

Solution: For same track on disk (with correct initial positioning)

$\hookrightarrow t_{seek} = 0$, $t_{rot} = 4.16\,ms$ (on average, half rotation to reach desired sector)

$t_{transfer} \simeq 0.08s$ (from 7(a))

$\Rightarrow$ Throughput $= \frac{4kB}{0 + 4.16 + 0.08}\,kB\,ms^{-1} = \frac{4000}{4.24}\,kB/s = \frac{1000}{1.06}\,kB/s$

$\sim \boxed{998\,kB/s}$ (approx) ±10%

Ans

−2.5
+0.5

(c) What is the expected throughput of the hard drive when reading the very next 4 KB sector (i.e. the read/write head is immediately over the proper track and sector at the start of the operation)?

Solution: when reading next 4kB sector, ~~all in~~ here,

$t_{seek} = 0$, $t_{rot} = 0$, $t_{transfer} \doteq \frac{4kB}{50MB/s} \simeq 0.08s$

$\Rightarrow$ Throughput = Transfer Rate $= \boxed{50MB/s}$ Ans.