(1) In memory management,

(a) What is meant by relocation and protection? Why are they needed?

(b) What is external fragmentation? How can it be resolved?

Answer:

1. Relocation refers to shifting all addresses of code in main memory by a BASE bytes, so that the code can be loaded into, and executed from, any location in main memory.

Protection means checking each memory reference by the code to verify that it is below a maximum value and that it doesn’t try to access memory belonging to others.

Relocation and protection together allow a programmer to write code using relative addressing (0-MAX) without having to know the final memory address where the code and data may actually reside.

1. External fragmentation occurs when all free memory regions are small and distributed at different non-contiguous locations in the main memory. As a result, an allocation request for a large contiguous memory region cannot be satisfied, even though enough free memory exists.

External fragmentation is resolved by **compaction**, i.e. moving all allocated memory regions to one end of physical memory so that all free memory is contiguous at the other end.

(2) I/O Models

(a) ﻿Explain the steps along the ﻿﻿﻿path that data travels from (say) a﻿﻿ hard disk to a ﻿user space application﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿?

(b) ﻿Compare the blocking I/O model with the I/O Multiplexing model? How are they similar and how are they different?

Answer:

1. First stage is the reception of data from the hard disk into the kernel buffers. The second stage is copying the data from the kernel buffers to user space.
2. In the blocking I/O model, a process blocks in both stages of data reception, but only on a single I/O descriptor. The OS receives the data to kernel buffers, copies the data to user space and then returns back to user process.

I/O multiplexing is is similar to blocking model. However, during the first stage, the process can block on I/O on multiple file descriptors, typically using the select() system call. If I/O event occurs on any descriptor, the OS returns to the process from the select() call. The process then tests each file descriptor and handles the I/O on ready descriptors in a blocking manner during the second stage. The first stage blocks on multiple descriptors simultaneously. The second stage blocks on each ready descriptor one by one.

Additionally, I/O multiplexing using select system call can acquire a range of behaviors in the first stage from fully blocking, to fully non-blocking, depending upon the timeout value specified in the select call.

(3) For a RAID system **with M disks, including data and parity,** compare the level of parallelism provided by RAID 1, RAID 3, RAID 4, and RAID 5 for multiple simultaneous (i) read I/O operations, (ii) write I/O operations, and (iii) combination of read and write I/O operations? Explain your answers.

Answer:

(i) Read I/O operations:

RAID 1: Allows up to M reads to be processed at a time

RAID 3: Allows only one read I/O operation to be processed at a time

RAID 4: Allows up to M-1 parallel reads to be processed at a time

RAID 5: Allows up to M parallel reads to be processed at a time

(ii) Write I/O operations:

RAID 1: Allows up to M/2 writes to be processed at a time.

RAID 3: Allows only one write I/O operation to be processed at a time

RAID 4: Allows only one write I/O operation to be processed at a time because a single parity disk becomes a bottleneck.

RAID 5: Allows up to M/2 parallel write I/O operations to be processed at a time

(iii) Combination of read and write I/O operations:

RAID 1: Allows X parallel writes Y parallel reads where (2X+Y = M).

RAID 3: Allows only one I/O operation (read or write) to be processed at a time

RAID 4: Allows EITHER only one write OR M-1 reads to be processed.

RAID 5: Allows X parallel writes Y parallel reads where (2X+Y = M).

A correct answer could also be as follows without the formula: RAID 1 and 5 allows multiple reads and writes simultaneously.

(4) Assume that the

•   Size of each disk block is B bytes.

•   Address of each disk block is A bytes long.

•   The top level of a UNIX i-node contains D direct block addresses,  one single-indirect block address, one double-indirect block address,  one triple-indirect block address, and one quadruple-indirect block address.

* 1. What is the size of the ***largest “small”***file that can be addressed through direct block addresses?
  2. What is the size of the ***largest***file that can be supported by a UNIX inode?

Explain your answers.

Answer:

1. Number of direct block addresses = D

Size of file addressable through only direct block addresses = D x B bytes

1. Number of block addresses that can be stored in a disk block = size of disk block/size of each address = B/A

Size of the largest file supported =

Size addressable through direct block addresses +

Size addressable through single indirect block addresses +

Size addressable through double indirect block addresses +

Size addressable through triple indirect block addresses +

Size addressable through quadruple indirect block addresses

=

DB + (B/A)B + (B/A)(B/A)B + (B/A)(B/A)(B/A)B + (B/A)(B/A)(B/A)(B/A)B bytes

(5)

a) Explain therole of *file system cache* during (a) read I/O operations and (b) write I/O operations.

(b) How to Log-structured file systems improve I/O throughput?

Answer: