1.a. A portable Operating System is one that can be ported from one system architecture to another with little modification. Explain, giving examples, why it is infeasible to build an OS that is portable without any modification at all.

(Exam question 2020-21)

Answer:

By definition an OS has to interact with the hardware directly. It therefore contains code that depends on the specifics of the processor architecture such as its instruction set and the memory management system.

1.b. Briefly describe two ways in which deadlocks can be prevented.

(Exam question 2020-21)

Answer:

Prevent one of the four deadlock conditions from happening – mutual exclusion (share resource), hold and wait (ask for all resources at start), no preemption (allow for it), circular wait (force single resources or order resources).

1.c. A multi-threaded process uses a kernel semaphore S to synchronise between its threads. What issues, if any, arise if the threads are user-level or kernel-level threads? You can assume no threads in other processes have access to S.

(Exam question 2020-21)

Answer:

With kernel threads, a thread can block on a semaphore and the kernel can run another thread in the same process. With user-level threads, when one thread blocks on a semaphore, the kernel thinks the entire process is blocked and does not run it ever again. Consequently, the process fails.

1.d. List three reasons why scheduling processes and threads on a multiprocessor system is more compli-cated than scheduling them on a uniprocessor system.

(Exam question 2020-21)

Answer:

Access to shared data for concurrently executing processes, load-balancing across processors, difficulty of debugging due to non-determinism, difficult to exploit hardware level features in user-level programming.

1.e. In a system that allows multi-programming, how many processes need to be in main memory to only waste 10% of CPU? Assume each process spends around 80% waiting for I/O. Show your working clearly.

(Exam question 2020-21)

Answer:

n = total number of processes

p = fraction of time a process is waiting for I/O

pn = probability that all processes are waiting for I/O

CPU Utilisation = 1 – pn ⇒ 1 – 0.8n = 90% (from question)

⇒ 0.9 = 1 – 0.8n ⇒ 0.8n = 0.1 ⇒ n = log0.8 0.1 ≈ 10

1.f. Consider the following scenario with three processes P1, P2, P3 and two buffers B1 and B2.

• P1 takes a character from Input and writes it to B1

• P2 takes a character from B1 and writes it to B2

• P3 takes a character from B2 and writes it to Output

Each process runs in a never ending loop and can read/write one character at a time (reads are destructive) and each buffer can only hold one character. Given the concurrent nature of access, write a programme (pseudo-code) to coordinate these three processes using semaphores. Please state your assumptions.

(Exam question 2020-21)

Answer:

Text, letter

Description automatically generated

2.a. How does the page size defined on a particular architecture affect the working set size of processes on that architecture?

Answer: The working set size is the physical memory allocated to a process, which will be composed of pages. If pages are larger there is likely to be more internal fragmentation and therefore a larger working set size.

2.b. An alternative to using interrupts for I/O is polling (where you keep checking the status). List the circumstances in which using polling is a better choice.

(Exam question 2020-21)

Answer:

Polling is a simple mechanism and thus easier to implement than interrupt-driven I/O. This is big advantage for embedded devices that are resource-constraint in terms of their OS footprint. Also, in an embedded system, the CPU may not have anything else to do while waiting for I/O to finish. It may also be fine to use I/O polling when the I/O operation is supposed to finish quickly, making the overhead of a context switch in interrupt-driven I/O unnecessary

2.c. A file system uses inodes with single-indirect and double-indirect blocks and a block size of 1 KB. Approximately by what factor does the maximum possible file size increase if the block size of the file system is doubled.

(Exam question 2020-21)

Answer:

For very large files, almost all data blocks are accessed through the double indirect pointer, so we consider only these. For a block size of B and a pointer size of P, total size of the double-indirect data blocks is:

A picture containing text

Description automatically generated

If B is doubled, maximum file size will increase approximately by a factor of 2^3 or 8.

2.d. Consider a scenario where processes are communicating with each other by writing and reading blocks to disk. Would your choice of disk scheduling algorithm influence the synchronisation between the processes. Briefly discuss your answer.

(Exam question 2020-21)

Answer:

Each disk scheduling algorithm has its own way of ordering the I/O requests. Hence, the I/O requests will not be performed in the order in which they are produced. Therefore, irrespective of the disk scheduling algorithm used, the synchronisation between the processes will be affected

2.e. In a file system that uses I-nodes briefly describe (using an example) the advantages or disadvantages of using only triple indirect pointers to locate all file blocks.

Answer: For medium/small files using triple indirect pointers would be slower, as the system has to dereference 4 pointers to get to the actual memory block. For medium/small files, direct pointers, or less severely indirect points could have been used, which would require fewer dereferences and therefore be faster.

2.f. Consider a virtual memory system with a page size of 8 KB and a page table entry of 4 bytes. Please answer the questions below stating any assumptions.

1. Assuming that every page table fits into a single page, how many levels of page tables would be required to map a 46-bit virtual address?
2. Assuming the presence of a Translation Lookaside Buffer (TLB), how many memory operations are required to read or write a single 32-bit word? Discuss your answer.

Answer:

i)

Page size = 2^13 Bytes.

Virtual address = 46-bits.

Page Size/virtual address = (2^16)/46 = 1424 virtual addresses can fit on each page.

11 bits required to address 1424 (2^10 = 1024 is too few).

13 bits required for offset.

46 – 13 = 33 bits for various page levels.

Each page level requires 11 bits.

33/11 = 3 page levels.

ii) \*\*\*PLEASE CORRECT\*\*\*

Depends on how many TLB misses and page faults.

Assuming 46-bit address and 3-level paging;

No TLB misses => 1 memory access.

Each TLB miss adds one memory access.

Each page fault adds another memory access.

Diagram

Description automatically generated