1.What’s a process control block?

The operating system keeps a record of each process in the system, in the form of a process control block (PCB).

process ID number,

pointer,

process state,

program counter,

Contents of CPU registers,

memory management information,

I/O status information,

Accounting information.

2.What’s a heavyweight process?

Operating systems frequently support a special kind of processes called a thread to allow more complex data-sharing. The code section, data section, and O/S housekeeping info of a process is collectively known as a task. A task with just one thread is called a heavyweight process.

3.What’s a lightweight process (thread)?

A thread is also known as a lightweight process.

4.What’s context-switching?

When the operating system switches between processes, it has a fair amount of housekeeping to do. This housekeeping is known as context switching.

5.Concepts such as zombie & orphan processes, spooling, OS kernel, system calls, time-sharing system, etc

Orphan process: a process whose parent is terminated.

Zombie process: one which terminated, but its live parent process is not WAITING for it. Since no process is receiving its exit status, it stays in the process table.

A solution to the I/O bottleneck is to perform I/O for one job J1 at the same time as the CPU is executing another job J2. This is known as spooling. A disk is present here to store the jobs being input to the system and store the output from the jobs in the system.

OS kernel:

Processor's time which is shared among multiple users simultaneously is termed as time-sharing. An interactive program is constantly checking for user input while it is running. If several interactive programs are multi-programmed, and the CPU witches fast between them they behave as though they are being run independently, on several slower CPUs. It is called a time-sharing or multitasking system.

Processes issue requests to the kernel by making system calls.

6.How can heavyweight processes communicate with each other?

One final, general way of communicating between processes is via a message-passing system. To pass messages between processes, a communication link must be established. send to and receive from sockets.

7.How can lightweight processes communicate with each other?

8.User-level threads and kernel-level threads, their advantages and disadvantages.

Kernel-level threads: implemented via system calls. -The kernel knows about individual threads. But switching between threads is slower.

User-level threads: implemented in user-level libraries, without the need for system calls. These are the fastest kind of thread to switch between. But then the kernel doesn’t know about individual threads within a process. Why should that matter?

1. UNFAIRNESS: ONE PROCESS MIGHT HAVE 100 THREADS, ANOTHER PROCESS JUST ONE. THE KERNEL GIVES THEM EQUAL TIME IN THE CPU. (II) SYSTEM CALLS. IF A THREAD MAKES A SYSTEM CALL, THEN THE WHOLE (HEAVYWEIGHT) PROCESS IS SUSPENDED UNTIL IT’S COMPLETED; THAT INCLUDES ALL THE OTHER THREADS.
2. Kernel-level threads: implemented via system calls. In this case, the kernel knows about individual threads. But switching between threads is slower.

9.Process management such as process creation and termination.

Termination:When a process has executed its last statement, it executes a special function: the exit system call, at which point, it may return its status to its parent process (the one waiting for it to terminate); all its resources (main memory, open files etc) are de-allocated by the operating system.

10.What are the criteria for evaluating CPU scheduling algorithms?

CPU utilisation: the percentage of time that the CPU is busy.

Throughput: the number of processes that are completed per time unit.

Turnaround time (for a single process): the length of time from when the process was submitted (arrived) to when it is completed.

Waiting time (for a single process): the total amount of time the process spends waiting for the CPU. Response time (for a single process): the average time from the submission of a request to a process until the first response is produced.

11.What is a preemptive scheduling algorithm?

13.What are the different scheduling algorithms, and what are their advantages and disadvantages?

FCFS: Advantages:

easy to implement. easy to understand. Disadvantages:

waiting time not likely to be minimal. convoy effect: lots of small processes can get stuck behind one big one.

SJF: Advantages:

provably optimal average waiting time. Disadvantages:

you never know in advance what the length of the next cpu burst is going to be. possibility of long processes never getting executed.

RR: If time quantum is big, same as FCFS. If time quantum is small, we get faster response time, but slower throughput.

14.Concepts such as race condition, critical section, etc.

A race condition is an undesirable situation that occurs when a device or system attempts to perform two or more operations at the same time, but because of the nature of the device or system, the operations must be done in the proper sequence to be done correctly.

To prevent race conditions, we must guarantee that certain sections of the cooperating programs are not interleaved. These special protected sections are termed critical sections.

15.The three requirements of a critical section.

-Mutual Exclusion: only one process can be executing its critical section at a time.

-Bounded Waiting: when a process P1 becomes ready to enter its critical section, there must be a bound on the number of times other processes can enter their critical sections before P1 can do so.

 -Progress: if a process wants to enter its critical section, then:

No process executing its non-critical section, should be allowed to stop it. The decision about whether it can enter its critical section can’t be postponed indefinitely.

16.2-process software synchronisation solutions, e.g. Peterson’s solution.

17.Atomic instructions by hardware such as testAndSet, CAS.

18.Semaphores Busy waiting vs. queueing semaphores

19.The producer-consumer and single-writer-multiple-reader problems using semaphores

20.What’s a deadlock state?

A deadlock is a situation in which two computer programs sharing the same resource are effectively preventing each other from accessing the resource, resulting in both programs ceasing to function.

21.What’s a safe state?

A system is in a safe state if there exists a sequence of processes P1, . . . , Pn such that the system can allocate all the resources still to be demanded by each process Pi , using only when the resources currently free, and  the collection of resources requested by all Pj (j < i).

22.Resource allocation graphs Graph-based deadlock detection algorithm

23.Address binding methods: compile-time, load-time, run-time. . .

Address Binding: Symbols can be converted into addresses at several points:

Compile time. Compilers can create:

absolute code: if it’s known where the process will reside in memory.

relocatable code: if this is not known. Link time. Programs are often designed as separate modules which are compiled separately, but make reference to each other. These must be linked, so that their variables are correctly bound. Load time. When the process is loaded into main memory. Run time. If a process can be moved during execution from one place in memory to another.

24.Concepts such as logical/virtual and physical addresses, page table, swapping. . .

\*A logical address is one referred to in a piece of executable code. It is used by the compiler and the CPU.

\*A physical address is the address which is actually sent to the memory unit.

\*Logical address to physical address: The logical address generated by the CPU is divided into two parts: a page number and an offset. The page number is used as an index into a page table. The page table contains the base address of each page in physical memory. This base address is combined with the offset to define the physical memory address that is sent to the memory unit.

25.Memory allocation methods: contiguous (best-fit, worst-fit), noncontiguous. . .

26.Paging mechanism and segmentation, and their benefits and drawbacks. . .

\*Pros/Cons of Paging:

Pros: eliminates external fragmentation, efficient memory usage

Cons: Internal fragmentation, need special hardware for address translation

27.Virtual memory with demand paging

28.Quantitative questions:

Page replacement algorithms (FIFO, optimal, LRU)

Memory access time

29.Thrashing, its causes and its remedies

Thrashing: If a process has not been allocated sufficient frames, it thrashes:

-A page fault occurs, requiring a page to be replaced.

-The replaced page is itself needed shortly after. A process is thrashing if it’s spending more time paging than executing. The solution can be local page replacement where a process can only be allocated pages in its own region in memory.

30.The working set and locality models

Locality of reference: is a term for the phenomenon in which the same values, or related storage locations, are frequently accessed, depending on the memory access pattern.

The Working Set model: The working set is an approximation of the program’s current locality.

31.Concepts like inode, file, directory, DMA, I/O ports, etc.

32.Methods of file access (sequential, random)

File Access Methods: Sequential access: most common method.

-A file pointer identifies a record within the file. -The pointer can be moved incrementally forwards or backwards. -The hardware metaphor is a tape.

Direct access: -A file is viewed as a numbered sequence of records. Operations (e.g. read, write) can be

carried out on any record in any order. -The hardware metaphor is a disk.

33.Semantic for system calls for file like read() and write().

34.Memory-mapped files

35.Handshaking between CPU and I/O devices

36.Disk allocation methods (contiguous, linked, indexed, FAT-based; multilevel indexing) plus their advantages and disadvantages

Disk Allocation Methods: Contiguous allocation: Each file occupies a set of contiguous blocks on the disk. Pros:

-Good sequential access.

-Good direct access. Cons:

-External fragmentation.

\*Linked Allocation: -Each file is a linked list of disk blocks. -Each block of the file contains a pointer to the next block. Pros:

-No external fragmentation.

-Files can be arbitrarily big. Cons:

-Internal fragmentation. -Pointers take up some space. -Can only be used effectively for sequential-access files.

\*Indexed Allocation: Each file has an index block, containing a table specifying the physical block for each logical block. Pros:

-No external fragmentation

-Very direct access Cons:

-Internal fragmentation. -Lots of head seeks.

37.I/O device types and their characteristics

38.Disk scheduling algorithms, their advantages and drawbacks.