Operating Systems (CS3510) Final Exam

SURAJ TELUGU

CS20BTECH11050

1.sol

Earlier the computers were having a single CPU core which handles all the data which is read and written. The systems were single processing i.e only one process can run at time, all other processes had to wait. If the current process has an I/O request it takes much time and makes the system less efficient to use. With Multiprogramming we can overcome the above difficulties.

The main objective of multiprogramming is to have some process running at all times, to maximize CPU utilization. When a process waits for I/O request in the mean time CPU carries out other processes which are stored in cache (for small processes) and secondary memory (for large processes). Though the secondary memory takes more time we have no choice since our computer does not have DMA. Since Secondary memory takes few milliseconds to give the process to the CPU in the mean time small processes in cache can be implemented by scheduling properly.

2.sol

Initially the word is in the disk to calculate the average access time we need to calculate the average time taken from disk to main memory , main memory to cache and then time to access cache

Disk -> Main memory -> Cache -> Access

Average access time = ∑ hit ratio X (time to access memory element)

Hit ratio for Disk is always 100% since it is slow

Average access time = 1x10ms + 0.95 x 10ns + 0.99 x 1ns

= 10ms + 10.49ns = **10.00001049 ms**

3.sol

Given,

The application has a serial portion (S) = 30% = 0.3

We know that, speed up ≤ 1/ (S + (1-S)/N) (as per Amahl's law)

Maximum speed up (as per Amahl's law) = **1/ (S + (1-S)/N)**

= 1/ (0.3 + 0.7/N)

a) For Four processors:
   Maximum speed up = 1/ (0.3 + 0.7/4) = 40/19
   Maximum speed up = **2.1053**

b) For Eight processors:
   Maximum speed up = 1/ (0.3 + 0.7/8) = 80/31
   Maximum speed up = **2.5806**

4.sol

Given,

For Servicing the request, it takes 12ms. If a disk operation is needed, as is the case one-third of the time, an additional 75 msec is required, during disk operation thread sleeps.

Request/sec server can handle:

(a) Single threaded server

Time for Disk -> cache -> access(request) = 75 + 12 = 87
Avg Time for each request = 2/3(Request servicing) + 1/3(Disk operation)
= 2/3(12) + 1/3(87) = 37ms

For each request it takes average time of 37ms

Request/sec = 1/37ms = 1000/37 = 27.027

Therefore, a single threaded server can handle **27 requests per second**

(b) Multi-threaded server

Since there are many threads their disk operation is done in other thread. So, we only need cache access time for the request the proceed

For each request it takes average time of 12ms

Request/sec = 1/12ms = 1000/12 = 83.3333

Therefore, a multi-threaded server can handle **83 requests per second**

5.sol

(a)

Asynchronous cancellation: One thread immediately terminates the target thread

The difficulty with asynchronous cancellation occurs when resources like data has been allotted to a cancelled thread or when a target thread is cancelled in the midst of updating data sharing with other threads. Though operating system reclaims system resources it cannot reclaim all the resources. Cancelling a thread asynchronously may not free a necessary system – wide resource which might cause problems in execution of other threads.

Example:

Let there be two threads in which one collects a password string from I/O and the other thread checks whether it is a strong or weak password. Let thread 1 be cancelled asynchronously in another thread then the shared memory of thread 1 and thread 2 gets cancelled which may interrupt the execution of thread 2 and also the operating system may or may not had reclaimed the data from thread 1.

(b)

Deferred cancellation: The target thread periodically checks whether it should terminate, allowing it an opportunity to terminate itself in an orderly fashion.

The cancellation point invoke the pthread_testcancel() function. Once we reach a cancellation point the call to the above function will not return and thread will terminate, if its not a cancellation point the call to this function will return and thread will continue to run. Additionally, pthreads allows a function known as a cleanup handler to be invoked if a thread is cancelled. This function allows any resources a thread may have acquired to be released before the thread is terminated.

Pseudo Code:

```
while (1)
 {  /* do some work for awhile */ ...
    if(!cancellation point) {
      pthread_testcancel();
       return;
    }
    else pthread_testcancel();
}
```

6.sol

(a)

A spawned process is destroyed automatically when its parent is destroyed;

Beneficial Situation: If a spawned process depends on the data altered by parent through pipes after invoking the fork call, then once the parent is destroyed the spawned process must be destroyed because it cannot execute without the altered data.

A spawned processes proceed independently of their parents, and the destruction;

Beneficial Situation: If the spawned process uses exec() system call in it then all the data related to the parent are erased so even if the parent is destroyed the spawned process is now and independent process and executes independently

(b)

Situation in which destroying a parent should specifically not result in the destruction of its children

If child process uses exec() system call all the parent copied data is erased making the child a new and independent process so this process should not be deleted even if the parent is deleted since it does not depend on its parent.

7.sol

Single threaded server is better when same service should be performed for any request i.e if the server is using a many to one threading, then it can do the same service for different client requests.

Example of such server:

If the server is an online – Library server whose only action is to provide the link of the book if it is present and “Not found” message if it is not present then, for different client requests i.e for different request to find different books same thread is executed and the book link is returned if it is present.

Using multi-threading in such server would waste memory.