1. Clúster Computing generalmente se caracteriza por tener poca dispersión geográfica, sistemas homogéneos y un carácter estático. (1 P)

1 a. Verdadero (✓)
b. Falso ( )

2. Las aplicaciones distribuidas deben ejecutarse independientemente de la arquitectura de hardware. En este sentido, un Sistema Operativo Distribuido (SOD) es una capa de software que actúa entre el hardware y la aplicación distribuida. (1 P)

1 a. Verdadero (✓)
b. Falso ( )

3. Dos ejecuciones distribuidas F y E son equivalentes cuando tienen el mismo conjunto de eventos y se mantiene el orden causal de los mismos (1 P).

1 a. Verdadero (✓)
b. Falso ( )

4. Según la Taxonomía de Flynn, un sistema distribuido tipo ..... se caracteriza por: (1) Cada unidad ejecuta una instrucción distinta, (2) Cada unidad procesa un dato distinto y (3) Todas las unidades operan simultáneamente (1 P).

a. MIMD (✓)
1 b. SIMD ( )
c. MISD ( )
d. SISD ( )

5. En los ordenadores con memoria compartida, la memoria local de una CPU no es visible por otras CPUs (1 P).

1 a. Verdadero ( )
b. Falso (✓)

6. Granularidad es una medida cualitativa del cociente entre el tiempo dedicado al a la computación y el tiempo dedicado a la comunicación. Cuando se observa un alto grado de computación entre comunicaciones sucesivas, se considera: (1 P)

1 a. Granularidad fina ( )
b. Granularidad gruesa (✓)

7. El segmento de código que solicita permiso para acceder a la sección crítica se conoce como: (1 P)
   a. Sección de entrada (X)
   b. Sección de salida ( )
   c. Condición de carrera ( )

8. Un programa contiene 5 procesos que trabajan de manera concurrente. Todos ellos desean acceder a una región de memoria compartida por lo que incluyen una región crítica en su código. Se ha comprobado que uno de los procesos está en sección de salida, sin embargo también participa en la decisión de quién entra en sección crítica. Dicho programa no cumple con el principio de : (1 P)
   a. Exclusión mutua ( )
   b. Progreso (X)
   c. Espera limitada ( )

9. Un semáforo de tipo contador es inicializado con el número de recursos existentes S. En este sentido, el semáforo gestiona los recursos por medio de 2 operaciones P(S) y V(S). En donde: (1 P) (X)
   a. P(S) decrementa S en 1 y V(S) incrementa S en 1 ( )
   b. V(S) decrementa S en 1 y P(S) incrementa S en 1

10. Sean dos procesos concurrentes P1 y P2 que se encuentran corriendo: (2 P)
   - P1 con enunciado S1
   - P2 con enunciado S2
   - Se desea que S2 sea ejecutado después de que S1 haya terminado. La solución utilizando semáforos (sincro – inicializado en 0) es:

a) ( )

P1:_____
_____
P(sincro)
S1
_____
_____

P2:_____
_____
V(sincro)
S2
_____
_____

b) ( )

P1:_____
_____
V(sincro)
S1
_____
_____

P2:_____
_____
S2
P(sincro)
_____
_____

c) (X)

P1:_____
_____
S1
V(sincro)

P2:_____
_____
P(sincro)
S2

11. Se garantiza que al iniciar una operación con un semáforo, ningún otro proceso puede tener acceso al semáforo hasta que la operación termine o se bloquee. Dicho principio se conoce como: (1 P)

1

a. Integridad ( )
b. Deadlock ( )
c. Atomicidad (✓)
d. Cola de tareas ( )

12. En un computador, múltiples procesos y threads están ejecutándose simultáneamente. Sin embargo...(1 P)

1

a. Cada proceso tiene su propio registro (register) y stack ( )
b. Cada thread tiene su propio registro (register) y stack (✓)

13. Multiples threads dentro de un proceso comparten: (2 P)

2

a. Heap storage, static storage and code (✓)
b. Static storage, register and stack ( )
c. Heap storage, register and code ( )
d. Register, stack and code ( )

14. Un código se considera "thread safe" cuando: (2 P)

2

a. Múltiples threads pueden ejecutarse y escribir en el mismo espacio de memoria compartida simultáneamente ( )
b. Múltiples threads pueden ejecutarse simultáneamente, sin embargo sólo un thread puede escribir en el mismo espacio de memoria compartida. (✓)
c. Múltiples threads pueden ejecutarse simultáneamente, sin embargo sólo un thread puede enviar mensajes de broadcast ( )

15. Una conjunto de 3 Procesos (P0,P1,P2) se encuentra en estado "deadlock" cuando: (2P)

2

a. Un proceso P0 espera un recurso R0 que ya ha sido asignado a P1. Sin embargo, P1 al mismo tiempo espera un recurso R1 que ya ha sido asignado a P2. Los procesos P0 y P2 no han sido sincronizados adecuadamente, por lo que el resultado obtenido es diferente a una aplicación en serie ( )
b. Un proceso P0 espera un recurso R0 que ya ha sido asignado a P1. Sin embargo, P1 al mismo tiempo espera un recurso R1 que ya ha sido asignado a P2. P2 espera un recurso R2 que ya ha sido asignado a P0 (✓)
c. Un proceso P0 espera un recurso R0 que ya ha sido asignado a P1. Sin embargo, P1 al mismo tiempo espera un recurso R1 que ya ha sido asignado a P2. P2 tienen un tiempo de ejecución alto y tarda en liberar R1, tiempo en el cual los otros procesos están bloqueados ( )

16. En el método de paso de mensajes, el uso de **buzones** (puertos) corresponde con un enlace de .... (1 P)

1

a. Comunicación directa ( )
b. Comunicación indirecta (✓)
c. Comunicación binaria ( )
d. Comunicación multidimensional ( )

17. En el método de paso de mensajes, en una comunicación en donde las operaciones recibir y enviar tienen el mismo formato se la conoce como: (1 P)

1

a. Comunicación simétrica (✓)
b. Comunicación asimétrica ( )
c. Comunicación binaria ( )
d. Comunicación multidimensional ( )

18. En una región crítica (S) de tipo condicional se declara una variable v de tipo T compartida (*variable v: compartida T*). El enunciado de región crítica condicional sería: (*región* **v** *cuando* **B** *hacer S*). Es decir: (2 P)

2

a. B es una expresión booleana y S se ejecuta cuando B es falso. ( )
b. B es una expresión booleana y S se ejecuta cuando B es verdadero. (✓)
c. S se ejecuta independientemente del tipo o valor de B ( )

19. Una opción para la sincronización de procesos son los monitores. En un monitor los procesos no tienen acceso directo a las estructuras de datos. (1 P)

0

a. Verdadero ( )
b. Falso (✓)

20. En programas que utilizan OpenMP es posible experimentar problemas de rendimiento ocasionados por "false sharing". Este fenómeno sucede cuando: (2 P)

a. Varios threads operan sobre datos independientes pero que se encuentran en una misma región de una dirección memoria (líneas caché). Los protocolos de caché sincronizada obligan a actualizar las líneas en todos los threads ocasionando retrasos. (✓)

2

b. El usuario ha implementado un algoritmo que tiene una granularidad gruesa, el mismo que ocasiona que el número de mensajes entre threads sea elevado. Este comportamiento ocasiona retrasos considerables en el programa ( )
c. La aplicación requiere que varios threads intenten acceder a una misma región de memoria, sin embargo no se ha protegido correctamente la región crítica. Esto ha ocasionado que cada vez que se ejecuta la aplicación, se tiene resultados diferentes ( )

21. Una aplicación requiere que cada thread espere hasta que todos los otros threads hayan llegado. Este comportamiento se puede lograr en OpenMP con el uso de: (1 P)

1

a. #pragma omp critical ( )
b. #pragma omp parallel ( )
c. #pragma omp barrier (✓)

22. En OpenMP, la expresión **#pragma omp atomic** provee exclusión mutua pero sólo aplica para la actualización de una posición de memoria (1 P)

1

a. Verdadero (✓)
b. Falso ( )

23. En la arquitectura básica de una programación paralela basada en paso de mensajes todos los procesadores tienen una memoria compartida y los procesos que se ejecutan en cada procesador operan sobre datos globales (1 P)

0

a. Verdadero (✓)
b. Falso ( )

24. En una comunicación punto a punto basado en paso de mensajes, las operaciones que corresponden a un modelo de comunicación **bloqueante** son: (1 P)

1

a. MPI_Send() y MPI_Recv() (✓)
b. MPI_Isend() y MPI_Irecv() ( )

25. En una comunicación punto a punto basado en paso de mensajes, la operación **MPI_Ssend()** se da por finalizada sólo cuando el mensaje ha sido recibido en el destino, es decir espera al receptor (1 P)

1

a. Verdadero (✓)
b. Falso ( )

26. En una comunicación punto a punto basado en paso de mensajes, la operación **MPI_Bsend()** sólo tiene éxito si se ha iniciado ya la recepción correspondiente (el receptor está preparado) (1 P)

1

a. Verdadero ( )
b. Falso (✓)

27. En una comunicación basada en paso de mensajes, las operaciones de comunicación colectiva implican un punto de sincronización y deben ser invocadas por todos los procesos (mismos parámetros). (1 P)

1

a. Verdadero (✓)
b. Falso ( )

28. En una comunicación basada en paso de mensajes MPI, la operación **MPI_Barrier** ( ) envía los mismos datos desde un proceso (root) a todos los demás procesos. (1 P)

1

a. Verdadero ( )
b. Falso (✓)

29. En una comunicación basada en paso de mensajes MPI, se desea que cada uno de los procesos reciba una parte de los datos que se encuentran en un buffer del root. La operación que permite esta funcionalidad es: (1 P)

1

a. MPI_Scatter (✓)
b. MPI_Gather ( )
c. MPI_Reduce ( )

30. En el modelo cliente servidor se asignan roles diferentes a los procesos que comunican. En este sentido: (2 P)

a. El servidor es un elemento activo (siempre debe estar pendiente de las peticiones) y el cliente un elemento pasivo (no siempre debe estar encendido, sólo cuando invoca peticiones (✓)

0 b. El servidor es un elemento pasivo (espera la llegada de peticiones) y el cliente un elemento activo (invoca peticiones) ( )

c. El servidor y cliente pueden tener un rol activo o pasivo dependiendo de la aplicación desarrollada ( )

31. En un modelo de servidor secuencial, el servidor crea un hijo (thread) que atiende la petición y envía la respuesta al cliente. (1 P)

1 a. Verdadero ( )
b. Falso (✓)

32. El diseño de software de un servidor puede realizarse utilizando diferentes arquitecturas. En la arquitectura concurrente tipo ..... cada trabajo se divide en una serie de fases, en cada una de ellas se procesa por un proceso ligero especializado: (2 P)

2 a. Segmentación (✓)
b. Bidireccional ( )
c. Distribuidor ( )
d. Unidireccional ( )

33. Un servidor **HTTP** es un ejemplo de servidor **con estado** ya que mantiene información de estado y cada petición/respuesta puede depender de otras anteriores. (1 P)

1 a. Verdadero ( )
b. Falso (✓)

34. ..... es un mecanismo de IPC y representa un extremo de una comunicación bidireccional con una dirección asociada (1 P)

a. Dirección IP ( )
1 b. Puerto TCP ( )
c. Socket (✓)
d. Dirección MAC ( )

35. Un socket tipo **AF_INET** puede comunicarse con otro socket tipo **AF_INET6** siempre y cuando los equipos tengan conectividad a nivel de red (ping funciona correctamente). (1 P)

0 a. Verdadero (✓)
b. Falso ( )

36. Un servidor requiere proveer un servicio utilizando TCP. El socket que debe utilizarse es: (1 P)

a. SOCK_STREAM (✓)
1 b. SOCK_DGRAM ( )
c. SOCK_RAW ( )

37. La longitud máxima de un datagrama (datos y cabeceras) es 64 KB. La cabera IP + cabecera UDP tienen 28 bytes. (1 P)

1
a. Verdadero (✓)
b. Falso ( )

38. Para que dos procesos puedan comunicarse por medio de sockets es necesario que los números de puertos en cada extremo sean iguales. (1 P)

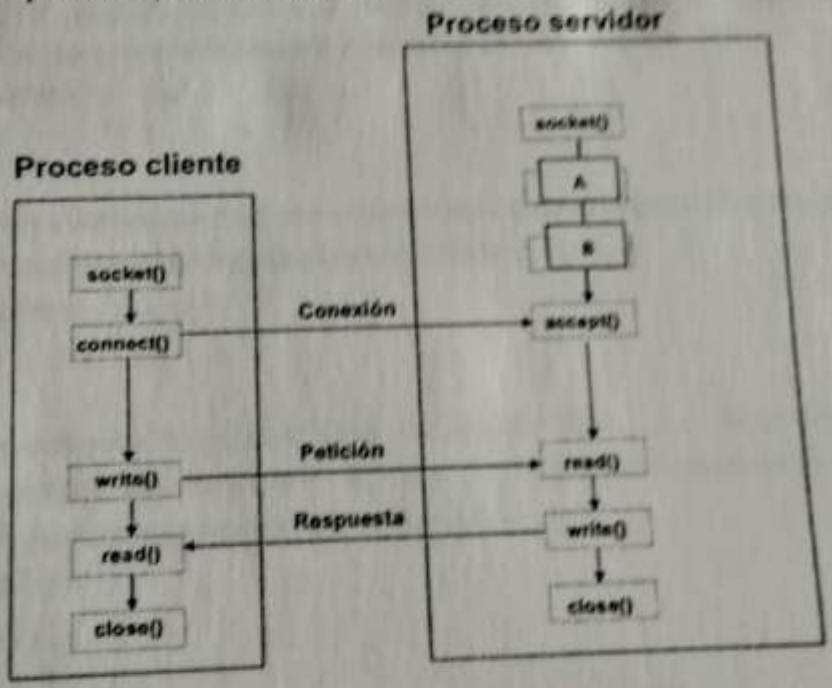
0
a. Verdadero (✓)
b. Falso ( )

39. Los protocolos que tienen límite entre mensajes son: (2 P)

2
a. IP, UDP (✓)
b. IP, TCP ( )
c. UDP, TCP ( )
d. IP, TCP, UDP ( )

40. El siguiente gráfico representa el modelo de comunicación con sockets tipo stream, las primitivas A y B corresponden con: (2 P)

2



a. A→open(), B→ready() ( )
b. A→ready(), B→open() ( )
c. A→listen(), B→bind() ( )
d. A → bind(), B→listen() (✓)