udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA

Solemne 1 **SO (CIT 2003-1)**

Instrucciones. Marque las casillas (◯) completamente sin salirse de ellas (por ejemplo ●). Responda a los siguientes cuestionamientos en las hojas que se le entregan **marcando una única opción**. Se utilizará factor de corrección 4 a 1 (las respuestas en blanco no se toman en cuenta). *Las últimas tres preguntas se evaluarán de manera Completa (4 pts.), Parcial (2 pts.), e Incompleta (0 pts.). No llene las casillas de estas últimas tres preguntas, sino que responda en el espacio indicado.*

◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0
◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1
◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2
◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3
◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4
◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5
◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6
◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7
◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8
◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9

⟵ Marque su RUT sin código verificador (el número después del guión), y escriba sus nombres y apellidos abajo.

Nombre(s) y apellido(s):

.....

.....

1. ¿Qué es una variable de condición?
- ◯ Varialbe de sincronización que implementa `yield`
- ◯ Variable de sincronización que permite la exclusión mutua
- ◯ Variable de sincronización que permite la espera eficiente de un hilo
- ◯ Objeto de sincronización que implementa `yield` en modo usuario

2. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de inicializar el espacio de memoria?
- ◯ Informar al calendarizador que el programa se puede ejecutar
- ◯ Cargar `prog` en la memoria
- ◯ Copiar los argumentos en la memoria
- ◯ Inicializar el hardware para que se ejecute el incio del programa

3. ¿Cuál de las siguientes es una característica de `fork`?
- ◯ Copiar a la memoria el código del programa del proceso padre
- ◯ Copiar el proceso padre completamente
- ◯ Crear una copia del proceso padre, pero no puede ser confiado igual que él
- ◯ Copiar el proceso padre con privilegios distintos

4. Tipo de kernel en el que la funcionalidad mínima está dentro de él, y el resto se encuentra a nivel de usuario.
- ◯ Microkernel
- ◯ Monolítico
- ◯ Híbrido
- ◯ Macrokernel

5. ¿Qué hace la función `wait` de una variable de condición?
- ◯ Duerme el hilo atómicamente mientras despierta al siguiente
- ◯ Atómicamente libera los seguros y mueve el hilo a la lista de espera
- ◯ Mueve el hilo a la lista de espera atómicamente
- ◯ Libera el seguro global, y duerme al hilo

6. ¿Cuál de los siguientes elementos **no** es compartido por los hilos?
- ◯ Heap
- ◯ Código
- ◯ Stack
- ◯ Variables globales

7. ¿Qué es el multi-hilado preventivo?
- ◯ Es cuando los hilos que se ejecutan no pueden descalendarizarse
- ◯ Es cuando los hilos no usan el procesador hasta que se les entrega indefinidamente
- ◯ Es cuando los hilos que se ejecutan pueden ser cambiados indistintamente
- ◯ Es cuando los hilos entregan el procesador voluntariamente, no son interrumpidos

8. ¿Cuál es el principal problema de los hilos a nivel de usuario?
- ◯ No podemos tener varios hilos por proceso
- ◯ El bloqueo del proceso, bloquea todos los hilos
- ◯ No podemos calendarizar hilos dentro del proceso
- ◯ No podemos tener varios procesos por hilo

9. ¿Qué es un sistema operativo?
- ◯ No es software, pero se encarga de administrar recursos y usuarios
- ◯ Es una capa de software que administra recursos y usuarios
- ◯ Es un software que administra recursos
- ◯ Es un software que administra usuarios

10. ¿Cuál es el primer paso en la creación del proceso del programa `prog`?
- ◯ Inicializar la memoria
- ◯ Copiar `prog` a la memoria
- ◯ Inicializar el hardware para que se ejecute el proceso desde el inicio
- ◯ Crear e inicializar el PCB en el kernel

11. ¿Cuál de los siguientes eventos **no** genera una transición de modo kernel a usuario?
- ◯ Continuar después de una interrupción
- ◯ Llamada I/O a un dispositivo
- ◯ Creación de nuevo proceso
- ◯ Cambiar a un proceso diferente

udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA

12. ¿Qué hace la función `yield` de un hilo?

- ◯ Duerme al hilo por un tiempo determinado, y cambia el estado a esperando (*waiting*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a terminado (*finished*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a esperando (*waiting*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a listo (*ready*)

13. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de cargar `prog` en la memoria?

- ◯ Copiar los argumentos en la memoria
- ◯ Informar al calendarizador que el programa se puede ejecutar
- ◯ Inicializar el PCB en el kernel
- ◯ Inicializar el hardware para que se ejecute el incio del programa

14. Modo de ejecución del procesador en el que verifica cada instrucción antes de ejecutarla.

- ◯ Modo dual
- ◯ Modo seguro
- ◯ Modo de usuario
- ◯ Modo kernel

15. Tipo de kernel en el que la mayoría de la funcionalidad de éste se encuentra dentro de él.

- ◯ Microkernel
- ◯ Macrokernel
- ◯ Monolítico
- ◯ Híbrido

16. ¿Qué característica del sistema operativo permite que opere independientemente del hardware en la computadora?

- ◯ Portabilidad
- ◯ Compartimiento de recursos
- ◯ Ilusionista
- ◯ Virtualización

17. ¿Qué es una llamada de sistema?

- ◯ Funciones del kernel que permiten al usuario accesar a recursos restringidos
- ◯ Código del kernel que ejecuta código de usuario
- ◯ Llamada que hace el kernel para realizar una instrucción
- ◯ Funciones de hardware que llama el usuario

18. ¿Cuál de los siguientes elementos **no** pertenece al *TCB*?

- ◯ Estado del hilo
- ◯ Puntero al stack frame
- ◯ Registros salvados
- ◯ Variables locales

19. ¿Qué es la exclusión mutua?

- ◯ Propiedad donde los hilos se bloquean mutuamente
- ◯ Objeto de sincronización que bloquea los hilos
- ◯ Propiedad de la concurrencia para mantener los hilos excluidos
- ◯ Propiedad en la que solamente un hilo puede accesar simultáneamente

20. Es una señal síncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Interrupción
- ◯ Trampa
- ◯ Señal de software
- ◯ Señal de I/O

21. ¿Cuándo se da una condición de carrera?

- ◯ Varios hilos se ejecutan simultáneamente
- ◯ Existe una sección crítica
- ◯ El estado de la ejecución de un programa depende del intercalado de diferentes hilos
- ◯ Varios hilos se pueden intercalar en su ejecución simultánea

22. Región de la memoria reservada por el sistema operativo para mantener el estado de las variables locales durante la llamada a funciones.

- ◯ Heap
- ◯ RAM
- ◯ Stack
- ◯ Swap

23. Modo de ejecución del procesador donde no se ejecuta ninguna verificación.

- ◯ Modo inseguro
- ◯ Modo kernel
- ◯ Modo de usuario
- ◯ Modo dual

24. ¿Qué es una operación atómica?

- ◯ Es una operación indivisible que no puede ser dividida
- ◯ Es una operación de hardware que no puede ser dividida
- ◯ Es una operación del kernel que no puede ser dividida
- ◯ Es una operación que deshabilita las interrupciones

25. ¿Qué son las direcciones de memoria virtuales?

- ◯ Capa de indirección que le da flexibilidad al sistema operativo para administrar la memoria
- ◯ Memoria que reserva el sistema operativo para el uso de máquinas virtuales
- ◯ Memoria adicional que crea el sistema operativo para engañar a las aplicaciones
- ◯ Capa adicional que utiliza el sistema operativo en modo kernel

26. ¿Qué es la ejecución de una aplicación con permisos restringidos?

- ◯ Hilo
- ◯ Proceso
- ◯ Aplicación
- ◯ Programa

27. ¿Qué es un pipe?

- ◯ Es un buffer del kernel entre dos descriptores de archivos
- ◯ Es un buffer de usuario entre dos descriptores de archivos
- ◯ Es un buffer del kernel entre dos procesos
- ◯ Es un buffer de usuario entre dos procesos

28. ¿Donde almacena el sistema operativo la toda la información sobre un proceso en particular?

- ◯ Heap
- ◯ Stack
- ◯ Memoria
- ◯ *Process control block* (PCB)

29. ¿Cuál de las siguientes afirmaciones es verdadera respecto a *spin wait*?

- ◯ Es justificable en monoprocesadores
- ◯ No se debe de usar nunca
- ◯ Produce *deadlocks*
- ◯ Es justificable en multiprocesadores

Dado el siguiente código

```
main(int argc, char** argv){
  int child == fork();
  int x = 5;
  if (child == 0) {
    x += 5;
  } else {
    child = fork();
    x += 10;
    if (child) {
      x+=5;
    }
  }
}
```

30. ¿Cuál de los siguientes **no** es un valor que toma la variable `x` al término de los procesos del programa anterior?

- ◯ 5
- ◯ 10
- ◯ 20
- ◯ 15

31. ¿Cuántas copias distintas de la variable `x` existen en la ejecución del programa anterior?

- ◯ 4
- ◯ 1
- ◯ 2
- ◯ 3

32. ¿Qué es un hilo?

- ◯ Varias secuencias de ejecución que pueden ser calendarizadas dependientemente
- ◯ Una secuencia de ejecución que es calendarizada dependientemente
- ◯ Varias secuencias de ejecución que pueden ser calendarizadas independientemente
- ◯ Una secuencia de ejecución que puede ser calendarizada independientemente

33. ¿Cuál de las siguientes **no** es una característica de un descriptor de archivo de UNIX?

- ◯ Se abren antes de usar
- ◯ Se leen a través de un buffer
- ◯ Se cierran completamente después de usar
- ◯ Se escriben a través de bytes

34. Llamada del sistema de UNIX que le permite a las aplicaciones el comunicarse entre sí para terminarlas, suspenderlas, o resumirlas.

- ◯ `wait`
- ◯ `exec`
- ◯ `signal`
- ◯ `fork`

35. ¿Cuál es el estado de un hilo en el que está listo para ejecutarse pero no está en el procesador?

- ◯ Ejecutandose (*running*)
- ◯ Esperando (*wait*)
- ◯ Inicializado (*init*)
- ◯ Listo (*ready*)

36. ¿Cuántos procesos son creados durante la ejecución del siguiente programa?

```
main(int argc, char** argv){
  forkthem(5);
}
void forkthem(int n){
  if (n > 0){
    fork();
    forkthem(n-1);
  }
}
```

- ◯ 5
- ◯ 16
- ◯ 30
- ◯ 4

37. ¿Cuál de las siguientes **no** es una característica de un hilo?

- ◯ Comparte el código del programa
- ◯ Tiene un stack propio
- ◯ Tiene un contador de programa propio
- ◯ Tiene un segmento de datos propio

38. Es una señal asíncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Interrupción
- ◯ Trampa
- ◯ Señal de software
- ◯ Señal de I/O

39. ¿Qué es concurrencia?

- ◯ Realizar una actividad sin la interrupción de otra
- ◯ Realizar múltiples actividades sin la interrupción de otras
- ◯ Realizar múltiples actividades simultáneamente
- ◯ Realizar múltiples actividades una después de la otra

40. Región de la memoria reservada por el sistema operativo para alojar estructuras de datos que el proceso pueda necesitar.

- ◯ Heap
- ◯ Stack
- ◯ RAM
- ◯ Swap

41. Dado el siguiente código del programa `main`

```
1  char bar(){
2    int b = 2;
3    return 'a';
4  }
5  int foo(int a, char b){
6    int i=0;
7    bar();
8  }
9  main(){
10   int b=2;
11   foo();
12 }
```

Dibuje el stack completo del programa cuando se encuentra ejecutando la linea número 2. Utilice el número de línea para referirse a la dirección de cada instrucción. ◯I ◯P ◯C

El siguiente código implementa una lista circular.

```
const int MAX = 10;

class Queue{
  int items[MAX];
  int front;
  int nextEmpty;
public:
  Queue() { front = nextEmpty = 0;}
  ~Queue();
  void insert(int item);
  int remove();
}

void Queue::insert(int item){
  items[nextEmpty%MAX] = item;
  nextEmpty++;
}
int Queue::remove(){
  int item;
  item = items[front%MAX];
  front++;
  return item;
}
```

Sin embargo, las funciones `Queue::insert` y `Queue::remove` no están sincronizadas. Por lo que más elementos de los que soporta la lista pueden insertarse o removerse.

Responda lo que se le solicita a continuación. Puede declarar los `Lock` y `CV` que considere necesarios. Tenga en cuenta que la implementación debe de considerar que no puede insertar si la lista está llena, por lo que deberá de detener la función que está insertando. De manera similar, cuando elimine si no hay ningún elemento en la lista deberá esperar a que uno exista.

42. Reescriba la función `Queue::insert` para que inserte de manera segura en la lista circular. ◯I ◯P ◯C

43. Reescriba la función `Queue::remove` para que elimine de manera segura en la lista circular. ◯I ◯P ◯C

udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA

Solemne 1 **SO (CIT 2003-1)**

Instrucciones. Marque las casillas (◯) completamente sin salirse de ellas (por ejemplo ●). Responda a los siguientes cuestionamientos en las hojas que se le entregan **marcando una única opción**. Se utilizará factor de corrección 4 a 1 (las respuestas en blanco no se toman en cuenta). *Las últimas tres preguntas se evaluarán de manera Completa (4 pts.), Parcial (2 pts.), e Incompleta (0 pts.). No llene las casillas de estas últimas tres preguntas, sino que responda en el espacio indicado.*

◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0
◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1
◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2
◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3
◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4
◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5
◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6
◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7
◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8
◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9

⟵ Marque su RUT sin código verificador (el número después del guión), y escriba sus nombres y apellidos abajo.

Nombre(s) y apellido(s):

.....

.....

1. ¿Qué es concurrencia?
- ◯ Realizar múltiples actividades sin la interrupción de otras
- ◯ Realizar una actividad sin la interrupción de otra
- ◯ Realizar múltiples actividades una después de la otra
- ◯ Realizar múltiples actividades simultáneamente

2. ¿Qué es el multi-hilado preventivo?
- ◯ Es cuando los hilos no usan el procesador hasta que se les entrega indefinidamente
- ◯ Es cuando los hilos que se ejecutan pueden ser cambiados indistintamente
- ◯ Es cuando los hilos que se ejecutan no pueden descalendarizarse
- ◯ Es cuando los hilos entregan el procesador voluntariamente, no son interrumpidos

3. Tipo de kernel en el que la funcionalidad mínima está dentro de él, y el resto se encuentra a nivel de usuario.
- ◯ Microkernel
- ◯ Macrokernel
- ◯ Monolítico
- ◯ Híbrido

4. Modo de ejecución del procesador en el que verifica cada instrucción antes de ejecutarla.
- ◯ Modo de usuario
- ◯ Modo seguro
- ◯ Modo kernel
- ◯ Modo dual

5. ¿Cuál de los siguientes elementos **no** es compartido por los hilos?
- ◯ Stack
- ◯ Código
- ◯ Variables globales
- ◯ Heap

6. ¿Qué es la ejecución de una aplicación con permisos restringidos?
- ◯ Proceso
- ◯ Programa
- ◯ Hilo
- ◯ Aplicación

7. Es una señal asíncrona al procesador que indica que ocurrió un evento que requiere su atención.
- ◯ Trampa
- ◯ Interrupción
- ◯ Señal de software
- ◯ Señal de I/O

8. ¿Qué son las direcciones de memoria virtuales?
- ◯ Capa de indirección que le da flexibilidad al sistema operativo para administrar la memoria
- ◯ Memoria adicional que crea el sistema operativo para engañar a las aplicaciones
- ◯ Capa adicional que utiliza el sistema operativo en modo kernel
- ◯ Memoria que reserva el sistema operativo para el uso de máquinas virtuales

9. ¿Cuándo se da una condición de carrera?
- ◯ El estado de la ejecución de un programa depende del intercalado de diferentes hilos
- ◯ Varios hilos se ejecutan simultáneamente
- ◯ Existe una sección crítica
- ◯ Varios hilos se pueden intercalar en su ejecución simultánea

10. Modo de ejecución del procesador donde no se ejecuta ninguna verificación.
- ◯ Modo inseguro
- ◯ Modo dual
- ◯ Modo kernel
- ◯ Modo de usuario

11. ¿Cuál de las siguientes afirmaciones es verdadera respecto a *spin wait*?
- ◯ Produce *deadlocks*
- ◯ Es justificable en multiprocesadores
- ◯ Es justificable en monoprocesadores
- ◯ No se debe de usar nunca

udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA

12. ¿Qué es una llamada de sistema?

- ◯ Llamada que hace el kernel para realizar una instrucción
- ◯ Funciones de hardware que llama el usuario
- ◯ Código del kernel que ejecuta código de usuario
- ◯ Funciones del kernel que permiten al usuario accesar a recursos restringidos

13. ¿Qué es un sistema operativo?

- ◯ Es un software que administra recursos
- ◯ Es una capa de software que administra recursos y usuarios
- ◯ No es software, pero se encarga de administrar recursos y usuarios
- ◯ Es un software que administra usuarios

14. ¿Qué hace la función `yield` de un hilo?

- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a esperando (*waiting*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a terminado (*finished*)
- ◯ Duerme al hilo por un tiempo determinado, y cambia el estado a esperando (*waiting*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a listo (*ready*)

15. ¿Qué característica del sistema operativo permite que opere independientemente del hardware en la computadora?

- ◯ Ilusionista
- ◯ Virtualización
- ◯ Compartimiento de recursos
- ◯ Portabilidad

16. ¿Qué hace la función `wait` de una variable de condición?

- ◯ Libera el seguro global, y duerme al hilo
- ◯ Duerme el hilo atómicamente mientras despierta al siguiente
- ◯ Atómicamente libera los seguros y mueve el hilo a la lista de espera
- ◯ Mueve el hilo a la lista de espera atómicamente

17. ¿Cuál es el estado de un hilo en el que está listo para ejecutarse pero no está en el procesador?

- ◯ Inicializado (*init*)
- ◯ Esperando (*wait*)
- ◯ Ejecutandose (*running*)
- ◯ Listo (*ready*)

18. ¿Qué es una operación atómica?

- ◯ Es una operación que deshabilita las interrupciones
- ◯ Es una operación de hardware que no puede ser dividida
- ◯ Es una operación del kernel que no puede ser dividida
- ◯ Es una operación indivisible que no puede ser dividida

19. ¿Qué es un hilo?

- ◯ Varias secuencias de ejecución que pueden ser calendarizadas independientemente
- ◯ Una secuencia de ejecución que es calendarizada dependientemente
- ◯ Varias secuencias de ejecución que pueden ser calendarizadas dependientemente
- ◯ Una secuencia de ejecución que puede ser calendarizada independientemente

20. ¿Cuál es el principal problema de los hilos a nivel de usuario?

- ◯ No podemos calendarizar hilos dentro del proceso
- ◯ El bloqueo del proceso, bloquea todos los hilos
- ◯ No podemos tener varios procesos por hilo
- ◯ No podemos tener varios hilos por proceso

Dado el siguiente código

```
main(int argc, char** argv){
  int child == fork();
  int x = 5;
  if (child == 0) {
    x += 5;
  } else {
    child = fork();
    x += 10;
    if (child) {
      x+=5;
    }
  }
}
```

21. ¿Cuál de los siguientes **no** es un valor que toma la variable `x` al término de los procesos del programa anterior?

- ◯ 10
- ◯ 20
- ◯ 15
- ◯ 5

22. ¿Cuántas copias distintas de la variable `x` existen en la ejecución del programa anterior?

- ◯ 2
- ◯ 3
- ◯ 1
- ◯ 4

23. Es una señal síncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Interrupción
- ◯ Señal de software
- ◯ Trampa
- ◯ Señal de I/O

24. ¿Cuántos procesos son creados durante la ejecución del siguiente programa?

```
main(int argc, char** argv){
  forkthem(5);
}
void forkthem(int n){
  if (n > 0){
    fork();
    forkthem(n-1);
  }
}
```

- ◯ 4
- ◯ 30
- ◯ 16
- ◯ 5

25. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de inicializar el espacio de memoria?

- ◯ Inicializar el hardware para que se ejecute el inicio del programa
- ◯ Cargar `prog` en la memoria
- ◯ Copiar los argumentos en la memoria
- ◯ Informar al calendarizador que el programa se puede ejecutar

udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA

26. ¿Qué es un pipe?

- ◯ Es un buffer del kernel entre dos procesos
- ◯ Es un buffer de usuario entre dos descriptores de archivos
- ◯ Es un buffer del kernel entre dos descriptores de archivos
- ◯ Es un buffer de usuario entre dos procesos

27. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de cargar `prog` en la memoria?

- ◯ Inicializar el PCB en el kernel
- ◯ Copiar los argumentos en la memoria
- ◯ Informar al calendarizador que el programa se puede ejecutar
- ◯ Inicializar el hardware para que se ejecute el incio del programa

28. ¿Donde almacena el sistema operativo la toda la información sobre un proceso en particular?

- ◯ Stack
- ◯ Memoria
- ◯ Heap
- ◯ *Process control block* (PCB)

29. ¿Qué es una variable de condición?

- ◯ Varialbe de sincronización que implementa `yield`
- ◯ Variable de sincronización que permite la espera eficiente de un hilo
- ◯ Objeto de sincronización que implementa `yield` en modo usuario
- ◯ Variable de sincronización que permite la exclusión mutua

30. ¿Cuál es el primer paso en la creación del proceso del programa `prog`?

- ◯ Inicializar el hardware para que se ejecute el proceso desde el inicio
- ◯ Crear e inicializar el PCB en el kernel
- ◯ Copiar `prog` a la memoria
- ◯ Inicializar la memoria

31. Llamada del sistema de UNIX que le permite a las aplicaciones el comunicarse entre sí para terminarlas, suspenderlas, o resumirlas.

- ◯ `signal`
- ◯ `fork`
- ◯ `exec`
- ◯ `wait`

32. Región de la memoria reservada por el sistema operativo para mantener el estado de las variables locales durante la llamada a funciones.

- ◯ RAM
- ◯ Swap
- ◯ Stack
- ◯ Heap

33. ¿Cuál de los siguientes elementos **no** pertenece al *TCB*?

- ◯ Registros salvados
- ◯ Variables locales
- ◯ Estado del hilo
- ◯ Puntero al stack frame

34. ¿Qué es la exclusión mutua?

- ◯ Propiedad en la que solamente un hilo puede accesar simultáneamente
- ◯ Propiedad de la concurrencia para mantener los hilos excluidos
- ◯ Objeto de sincronización que bloquea los hilos
- ◯ Propiedad donde los hilos se bloquean mutuamente

35. ¿Cuál de las siguientes **no** es una característica de un hilo?

- ◯ Tiene un stack propio
- ◯ Comparte el código del programa
- ◯ Tiene un segmento de datos propio
- ◯ Tiene un contador de programa propio

36. ¿Cuál de los siguientes eventos **no** genera una transición de modo kernel a usuario?

- ◯ Creación de nuevo proceso
- ◯ Continuar después de una interrupción
- ◯ Llamada I/O a un dispositivo
- ◯ Cambiar a un proceso diferente

37. ¿Cuál de las siguientes es una característica de `fork`?

- ◯ Copiar a la memoria el código del programa del proceso padre
- ◯ Copiar el proceso padre con privilegios distintos
- ◯ Copiar el proceso padre completamente
- ◯ Crear una copia del proceso padre, pero no puede ser confiado igual que él

38. Tipo de kernel en el que la mayoría de la funcionalidad de éste se encuentra dentro de él.

- ◯ Macrokernel
- ◯ Microkernel
- ◯ Monolítico
- ◯ Híbrido

39. ¿Cuál de las siguientes **no** es una característica de un descriptor de archivo de UNIX?

- ◯ Se escriben a través de bytes
- ◯ Se abren antes de usar
- ◯ Se leen a través de un buffer
- ◯ Se cierran completamente después de usar

40. Región de la memoria reservada por el sistema operativo para alojar estructuras de datos que el proceso pueda necesitar.

- ◯ RAM
- ◯ Stack
- ◯ Swap
- ◯ Heap

41. Dado el siguiente código del programa `main`

```
1  char bar(){
2    int b = 2;
3    return 'a';
4  }
5  int foo(int a, char b){
6    int i=0;
7    bar();
8  }
9  main(){
10   int b=2;
11   foo();
12 }
```

Dibuje el stack completo del programa cuando se encuentra ejecutando la linea número 2. Utilice el número de línea para referirse a la dirección de cada instrucción. ◯I ◯P ◯C

El siguiente código implementa una lista circular.

```
const int MAX = 10;

class Queue{
  int items[MAX];
  int front;
  int nextEmpty;
public:
  Queue() { front = nextEmpty = 0;}
  ~Queue();
  void insert(int item);
  int remove();
}

void Queue::insert(int item){
  items[nextEmpty%MAX] = item;
  nextEmpty++;
}
int Queue::remove(){
  int item;
  item = items[front%MAX];
  front++;
  return item;
}
```

Sin embargo, las funciones `Queue::insert` y `Queue::remove` no están sincronizadas. Por lo que más elementos de los que soporta la lista pueden insertarse o removerse.

Responda lo que se le solicita a continuación. Puede declarar los `Lock` y `CV` que considere necesarios. Tenga en cuenta que la implementación debe de considerar que no puede insertar si la lista está llena, por lo que deberá de detener la función que está insertando. De manera similar, cuando elimine si no hay ningún elemento en la lista deberá esperar a que uno exista.

42. Reescriba la función `Queue::insert` para que inserte de manera segura en la lista circular. ◯I ◯P ◯C

43. Reescriba la función `Queue::remove` para que elimine de manera segura en la lista circular. ◯I ◯P ◯C

udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA

Solemne 1 **SO (CIT 2003-1)**

Instrucciones. Marque las casillas (◯) completamente sin salirse de ellas (por ejemplo ●). Responda a los siguientes cuestionamientos en las hojas que se le entregan **marcando una única opción**. Se utilizará factor de corrección 4 a 1 (las respuestas en blanco no se toman en cuenta). *Las últimas tres preguntas se evaluarán de manera Completa (4 pts.), Parcial (2 pts.), e Incompleta (0 pts.). No llene las casillas de estas últimas tres preguntas, sino que responda en el espacio indicado.*

◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0
◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1
◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2
◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3
◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4
◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5
◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6
◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7
◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8
◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9

⟵ Marque su RUT sin código verificador (el número después del guión), y escriba sus nombres y apellidos abajo.

Nombre(s) y apellido(s):

.....

.....

1. ¿Qué son las direcciones de memoria virtuales?
- ◯ Capa adicional que utiliza el sistema operativo en modo kernel
- ◯ Memoria que reserva el sistema operativo para el uso de máquinas virtuales
- ◯ Capa de indirección que le da flexibilidad al sistema operativo para administrar la memoria
- ◯ Memoria adicional que crea el sistema operativo para engañar a las aplicaciones

2. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de cargar `prog` en la memoria?
- ◯ Informar al calendarizador que el programa se puede ejecutar
- ◯ Inicializar el hardware para que se ejecute el incio del programa
- ◯ Copiar los argumentos en la memoria
- ◯ Inicializar el PCB en el kernel

3. Región de la memoria reservada por el sistema operativo para alojar estructuras de datos que el proceso pueda necesitar.
- ◯ Swap
- ◯ Heap
- ◯ RAM
- ◯ Stack

4. ¿Qué es una operación atómica?
- ◯ Es una operación indivisible que no puede ser dividida
- ◯ Es una operación de hardware que no puede ser dividida
- ◯ Es una operación que deshabilita las interrupciones
- ◯ Es una operación del kernel que no puede ser dividida

5. ¿Qué es concurrencia?
- ◯ Realizar múltiples actividades simultáneamente
- ◯ Realizar múltiples actividades sin la interrupción de otras
- ◯ Realizar una actividad sin la interrupción de otra
- ◯ Realizar múltiples actividades una después de la otra

6. Modo de ejecución del procesador donde no se ejecuta ninguna verificación.
- ◯ Modo inseguro
- ◯ Modo de usuario
- ◯ Modo kernel
- ◯ Modo dual

7. ¿Cuál de las siguientes **no** es una característica de un hilo?
- ◯ Tiene un contador de programa propio
- ◯ Comparte el código del programa
- ◯ Tiene un stack propio
- ◯ Tiene un segmento de datos propio

8. ¿Cuál de las siguientes afirmaciones es verdadera respecto a *spin wait*?
- ◯ No se debe de usar nunca
- ◯ Es justificable en monoprocesadores
- ◯ Es justificable en multiprocesadores
- ◯ Produce *deadlocks*

9. ¿Cuál de los siguientes elementos **no** pertenece al *TCB*?
- ◯ Puntero al stack frame
- ◯ Registros salvados
- ◯ Estado del hilo
- ◯ Variables locales

10. ¿Qué es la exclusión mutua?
- ◯ Propiedad en la que solamente un hilo puede accesar simultáneamente
- ◯ Objeto de sincronización que bloquea los hilos
- ◯ Propiedad de la concurrencia para mantener los hilos excluidos
- ◯ Propiedad donde los hilos se bloquean mutuamente

11. ¿Cuál de los siguientes eventos **no** genera una transición de modo kernel a usuario?
- ◯ Continuar después de una interrupción
- ◯ Cambiar a un proceso diferente
- ◯ Llamada I/O a un dispositivo
- ◯ Creación de nuevo proceso

12. ¿Qué hace la función `yield` de un hilo?

- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a terminado (*finished*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a esperando (*waiting*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a listo (*ready*)
- ◯ Duerme al hilo por un tiempo determinado, y cambia el estado a esperando (*waiting*)

13. Llamada del sistema de UNIX que le permite a las aplicaciones el comunicarse entre sí para terminarlas, suspenderlas, o resumirlas.

- ◯ `exec`
- ◯ `signal`
- ◯ `wait`
- ◯ `fork`

14. ¿Cuál es el estado de un hilo en el que está listo para ejecutarse pero no está en el procesador?

- ◯ Ejecutandose (*running*)
- ◯ Inicializado (*init*)
- ◯ Esperando (*wait*)
- ◯ Listo (*ready*)

15. ¿Qué es un sistema operativo?

- ◯ Es un software que administra recursos
- ◯ Es una capa de software que administra recursos y usuarios
- ◯ Es un software que administra usuarios
- ◯ No es software, pero se encarga de administrar recursos y usuarios

16. Tipo de kernel en el que la mayoría de la funcionalidad de éste se encuentra dentro de él.

- ◯ Macrokernel
- ◯ Híbrido
- ◯ Monolítico
- ◯ Microkernel

17. Es una señal síncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Interrupción
- ◯ Señal de software
- ◯ Señal de I/O
- ◯ Trampa

18. ¿Qué hace la función `wait` de una variable de condición?

- ◯ Atómicamente libera los seguros y mueve el hilo a la lista de espera
- ◯ Mueve el hilo a la lista de espera atómicamente
- ◯ Duerme el hilo atómicamente mientras despierta al siguiente
- ◯ Libera el seguro global, y duerme al hilo

19. ¿Qué es un hilo?

- ◯ Una secuencia de ejecución que es calendarizada dependientemente
- ◯ Una secuencia de ejecución que puede ser calendarizada independientemente
- ◯ Varias secuencias de ejecución que pueden ser calendarizadas dependientemente
- ◯ Varias secuencias de ejecución que pueden ser calendarizadas independientemente

20. ¿Qué es un pipe?

- ◯ Es un buffer de usuario entre dos procesos
- ◯ Es un buffer del kernel entre dos procesos
- ◯ Es un buffer del kernel entre dos descriptores de archivos
- ◯ Es un buffer de usuario entre dos descriptores de archivos

21. Región de la memoria reservada por el sistema operativo para mantener el estado de las variables locales durante la llamada a funciones.

- ◯ Stack
- ◯ Swap
- ◯ RAM
- ◯ Heap

22. ¿Cuál de las siguientes **no** es una característica de un descriptor de archivo de UNIX?

- ◯ Se cierran completamente después de usar
- ◯ Se leen a través de un buffer
- ◯ Se escriben a través de bytes
- ◯ Se abren antes de usar

23. ¿Cuál de los siguientes elementos **no** es compartido por los hilos?

- ◯ Heap
- ◯ Stack
- ◯ Código
- ◯ Variables globales

24. ¿Qué es la ejecución de una aplicación con permisos restringidos?

- ◯ Hilo
- ◯ Programa
- ◯ Aplicación
- ◯ Proceso

25. ¿Cuándo se da una condición de carrera?

- ◯ Varios hilos se ejecutan simultáneamente
- ◯ Existe una sección crítica
- ◯ Varios hilos se pueden intercalar en su ejecución simultánea
- ◯ El estado de la ejecución de un programa depende del intercalado de diferentes hilos

26. ¿Cuántos procesos son creados durante la ejecución del siguiente programa?

```
main(int argc, char** argv){
  forkthem(5);
}
void forkthem(int n){
  if (n > 0){
    fork();
    forkthem(n-1);
  }
}
```

- ◯ 5
- ◯ 30
- ◯ 4
- ◯ 16

27. Modo de ejecución del procesador en el que verifica cada instrucción antes de ejecutarla.

- ◯ Modo de usuario
- ◯ Modo kernel
- ◯ Modo seguro
- ◯ Modo dual

28. ¿Qué es el multi-hilado preventivo?

- ◯ Es cuando los hilos que se ejecutan no pueden descalendarizarse
- ◯ Es cuando los hilos no usan el procesador hasta que se les entrega indefinidamente
- ◯ Es cuando los hilos entregan el procesador voluntariamente, no son interrumpidos
- ◯ Es cuando los hilos que se ejecutan pueden ser cambiados indistintamente

Dado el siguiente código

```
main(int argc, char** argv){
  int child == fork();
  int x = 5;
  if (child == 0) {
    x += 5;
  } else {
    child = fork();
    x += 10;
    if (child) {
      x+=5;
    }
  }
}
```

29. ¿Cuál de los siguientes **no** es un valor que toma la variable `x` al término de los procesos del programa anterior?

- ◯ 5
- ◯ 10
- ◯ 20
- ◯ 15

30. ¿Cuántas copias distintas de la variable `x` existen en la ejecución del programa anterior?

- ◯ 3
- ◯ 1
- ◯ 4
- ◯ 2

31. ¿Donde almacena el sistema operativo la toda la información sobre un proceso en particular?

- ◯ Stack
- ◯ Heap
- ◯ *Process control block* (PCB)
- ◯ Memoria

32. ¿Qué es una llamada de sistema?

- ◯ Llamada que hace el kernel para realizar una instrucción
- ◯ Código del kernel que ejecuta código de usuario
- ◯ Funciones de hardware que llama el usuario
- ◯ Funciones del kernel que permiten al usuario accesar a recursos restringidos

33. ¿Cuál de las siguientes es una característica de `fork`?

- ◯ Copiar el proceso padre completamente
- ◯ Copiar a la memoria el código del programa del proceso padre
- ◯ Copiar el proceso padre con privilegios distintos
- ◯ Crear una copia del proceso padre, pero no puede ser confiado igual que él

34. ¿Cuál es el principal problema de los hilos a nivel de usuario?

- ◯ El bloqueo del proceso, bloquea todos los hilos
- ◯ No podemos calendarizar hilos dentro del proceso
- ◯ No podemos tener varios hilos por proceso
- ◯ No podemos tener varios procesos por hilo

35. ¿Qué es una variable de condición?

- ◯ Variable de sincronización que permite la espera eficiente de un hilo
- ◯ Variable de sincronización que permite la exclusión mutua
- ◯ Varialbe de sincronización que implementa `yield`
- ◯ Objeto de sincronización que implementa `yield` en modo usuario

36. ¿Qué característica del sistema operativo permite que opere independientemente del hardware en la computadora?

- ◯ Ilusionista
- ◯ Compartimiento de recursos
- ◯ Portabilidad
- ◯ Virtualización

37. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de inicializar el espacio de memoria?

- ◯ Informar al calendarizador que el programa se puede ejecutar
- ◯ Inicializar el hardware para que se ejecute el incio del programa
- ◯ Copiar los argumentos en la memoria
- ◯ Cargar `prog` en la memoria

38. Es una señal asíncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Interrupción
- ◯ Señal de I/O
- ◯ Señal de software
- ◯ Trampa

39. Tipo de kernel en el que la funcionalidad mínima está dentro de él, y el resto se encuentra a nivel de usuario.

- ◯ Monolítico
- ◯ Macrokernel
- ◯ Microkernel
- ◯ Híbrido

40. ¿Cuál es el primer paso en la creación del proceso del programa `prog`?

- ◯ Crear e inicializar el PCB en el kernel
- ◯ Inicializar el hardware para que se ejecute el proceso desde el inicio
- ◯ Copiar `prog` a la memoria
- ◯ Inicializar la memoria

41. Dado el siguiente código del programa `main`

```
1  char bar(){
2    int b = 2;
3    return 'a';
4  }
5  int foo(int a, char b){
6    int i=0;
7    bar();
8  }
9  main(){
10   int b=2;
11   foo();
12 }
```

Dibuje el stack completo del programa cuando se encuentra ejecutando la linea número 2. Utilice el número de línea para referirse a la dirección de cada instrucción. ◯I ◯P ◯C

El siguiente código implementa una lista circular.

```
const int MAX = 10;

class Queue{
  int items[MAX];
  int front;
  int nextEmpty;
public:
  Queue() { front = nextEmpty = 0;}
  ~Queue();
  void insert(int item);
  int remove();
}

void Queue::insert(int item){
  items[nextEmpty%MAX] = item;
  nextEmpty++;
}
int Queue::remove(){
  int item;
  item = items[front%MAX];
  front++;
  return item;
}
```

Sin embargo, las funciones `Queue::insert` y `Queue::remove` no están sincronizadas. Por lo que más elementos de los que soporta la lista pueden insertarse o removerse.

Responda lo que se le solicita a continuación. Puede declarar los `Lock` y `CV` que considere necesarios. Tenga en cuenta que la implementación debe de considerar que no puede insertar si la lista está llena, por lo que deberá de detener la función que está insertando. De manera similar, cuando elimine si no hay ningún elemento en la lista deberá esperar a que uno exista.

42. Reescriba la función `Queue::insert` para que inserte de manera segura en la lista circular. ◯I ◯P ◯C

43. Reescriba la función `Queue::remove` para que elimine de manera segura en la lista circular. ◯I ◯P ◯C

udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA

Solemne 1 **SO (CIT 2003-1)**

Instrucciones. Marque las casillas (◯) completamente sin salirse de ellas (por ejemplo ●). Responda a los siguientes cuestionamientos en las hojas que se le entregan **marcando una única opción**. Se utilizará factor de corrección 4 a 1 (las respuestas en blanco no se toman en cuenta). *Las últimas tres preguntas se evaluarán de manera Completa (4 pts.), Parcial (2 pts.), e Incompleta (0 pts.). No llene las casillas de estas últimas tres preguntas, sino que responda en el espacio indicado.*

◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0
◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1
◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2
◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3
◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4
◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5
◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6
◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7
◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8
◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9

⟵ Marque su RUT sin código verificador (el número después del guión), y escriba sus nombres y apellidos abajo.

Nombre(s) y apellido(s):

.....

.....

1. ¿Cuál es el primer paso en la creación del proceso del programa `prog`?

- ◯ Inicializar la memoria
- ◯ Crear e inicializar el PCB en el kernel
- ◯ Inicializar el hardware para que se ejecute el proceso desde el inicio
- ◯ Copiar `prog` a la memoria

2. Modo de ejecución del procesador en el que verifica cada instrucción antes de ejecutarla.

- ◯ Modo dual
- ◯ Modo de usuario
- ◯ Modo kernel
- ◯ Modo seguro

3. Región de la memoria reservada por el sistema operativo para alojar estructuras de datos que el proceso pueda necesitar.

- ◯ Stack
- ◯ Heap
- ◯ RAM
- ◯ Swap

4. ¿Qué característica del sistema operativo permite que opere independientemente del hardware en la computadora?

- ◯ Portabilidad
- ◯ Compartimiento de recursos
- ◯ Ilusionista
- ◯ Virtualización

5. ¿Cuál de los siguientes elementos **no** es compartido por los hilos?

- ◯ Stack
- ◯ Variables globales
- ◯ Código
- ◯ Heap

6. ¿Qué es concurrencia?

- ◯ Realizar múltiples actividades sin la interrupción de otras
- ◯ Realizar múltiples actividades una después de la otra
- ◯ Realizar múltiples actividades simultáneamente
- ◯ Realizar una actividad sin la interrupción de otra

7. ¿Qué es la exclusión mutua?

- ◯ Objeto de sincronización que bloquea los hilos
- ◯ Propiedad de la concurrencia para mantener los hilos excluidos
- ◯ Propiedad en la que solamente un hilo puede acesar simultáneamente
- ◯ Propiedad donde los hilos se bloquean mutuamente

8. ¿Qué es una variable de condición?

- ◯ Variable de sincronización que permite la espera eficiente de un hilo
- ◯ Objeto de sincronización que implementa `yield` en modo usuario
- ◯ Varialbe de sincronización que implementa `yield`
- ◯ Variable de sincronización que permite la exclusión mutua

9. Es una señal síncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Interrupción
- ◯ Señal de I/O
- ◯ Trampa
- ◯ Señal de software

10. ¿Qué es una operación atómica?

- ◯ Es una operación de hardware que no puede ser dividida
- ◯ Es una operación indivisible que no puede ser dividida
- ◯ Es una operación que deshabilita las interrupciones
- ◯ Es una operación del kernel que no puede ser dividida

11. ¿Cuál es el principal problema de los hilos a nivel de usuario?

- ◯ No podemos calendarizar hilos dentro del proceso
- ◯ El bloqueo del proceso, bloquea todos los hilos
- ◯ No podemos tener varios procesos por hilo
- ◯ No podemos tener varios hilos por proceso

12. ¿Cuál de los siguientes elementos **no** pertenece al *TCB*?

- ◯ Registros salvados
- ◯ Variables locales
- ◯ Puntero al stack frame
- ◯ Estado del hilo

13. Modo de ejecución del procesador donde no se ejecuta ninguna verificación.

- ◯ Modo inseguro
- ◯ Modo kernel
- ◯ Modo de usuario
- ◯ Modo dual

14. ¿Cuándo se da una condición de carrera?

- ◯ Varios hilos se ejecutan simultáneamente
- ◯ Varios hilos se pueden intercalar en su ejecución simultánea
- ◯ El estado de la ejecución de un programa depende del intercalado de diferentes hilos
- ◯ Existe una sección crítica

15. ¿Cuál de las siguientes afirmaciones es verdadera respecto a *spin wait*?

- ◯ Produce *deadlocks*
- ◯ Es justificable en multiprocesadores
- ◯ Es justificable en monoprocesadores
- ◯ No se debe de usar nunca

16. ¿Qué son las direcciones de memoria virtuales?

- ◯ Memoria que reserva el sistema operativo para el uso de máquinas virtuales
- ◯ Memoria adicional que crea el sistema operativo para engañar a las aplicaciones
- ◯ Capa de indirección que le da flexibilidad al sistema operativo para administrar la memoria
- ◯ Capa adicional que utiliza el sistema operativo en modo kernel

17. Es una señal asíncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Señal de software
- ◯ Interrupción
- ◯ Señal de I/O
- ◯ Trampa

18. ¿Cuál de las siguientes **no** es una característica de un descriptor de archivo de UNIX?

- ◯ Se cierran completamente después de usar
- ◯ Se abren antes de usar
- ◯ Se escriben a través de bytes
- ◯ Se leen a través de un buffer

19. ¿Qué es un pipe?

- ◯ Es un buffer del kernel entre dos descriptores de archivos
- ◯ Es un buffer de usuario entre dos procesos
- ◯ Es un buffer de usuario entre dos descriptores de archivos
- ◯ Es un buffer del kernel entre dos procesos

20. Tipo de kernel en el que la mayoría de la funcionalidad de éste se encuentra dentro de él.

- ◯ Monolítico
- ◯ Híbrido
- ◯ Microkernel
- ◯ Macrokernel

21. ¿Qué hace la función `yield` de un hilo?

- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a esperando (*waiting*)
- ◯ Duerme al hilo por un tiempo determinado, y cambia el estado a esperando (*waiting*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a terminado (*finished*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a listo (*ready*)

22. ¿Qué es la ejecución de una aplicación con permisos restringidos?

- ◯ Aplicación
- ◯ Hilo
- ◯ Proceso
- ◯ Programa

23. Tipo de kernel en el que la funcionalidad mínima está dentro de él, y el resto se encuentra a nivel de usuario.

- ◯ Híbrido
- ◯ Monolítico
- ◯ Macrokernel
- ◯ Microkernel

24. ¿Qué es el multi-hilado preventivo?

- ◯ Es cuando los hilos que se ejecutan no pueden descalendarizarse
- ◯ Es cuando los hilos que se ejecutan pueden ser cambiados indistintamente
- ◯ Es cuando los hilos entregan el procesador voluntariamente, no son interrumpidos
- ◯ Es cuando los hilos no usan el procesador hasta que se les entrega indefinidamente

25. ¿Cuál es el estado de un hilo en el que está listo para ejecutarse pero no está en el procesador?

- ◯ Esperando (*wait*)
- ◯ Inicializado (*init*)
- ◯ Listo (*ready*)
- ◯ Ejecutandose (*running*)

26. ¿Qué es un hilo?

- ◯ Una secuencia de ejecución que es calendarizada dependientemente
- ◯ Varias secuencias de ejecución que pueden ser calendarizadas independientemente
- ◯ Varias secuencias de ejecución que pueden ser calendarizadas dependientemente
- ◯ Una secuencia de ejecución que puede ser calendarizada independientemente

27. ¿Cuál de las siguientes **no** es una característica de un hilo?

- ◯ Comparte el código del programa
- ◯ Tiene un contador de programa propio
- ◯ Tiene un segmento de datos propio
- ◯ Tiene un stack propio

28. ¿Cuál de las siguientes es una característica de `fork`?

- ◯ Copiar el proceso padre completamente
- ◯ Copiar a la memoria el código del programa del proceso padre
- ◯ Copiar el proceso padre con privilegios distintos
- ◯ Crear una copia del proceso padre, pero no puede ser confiado igual que él

29. ¿Qué es una llamada de sistema?

- ◯ Código del kernel que ejecuta código de usuario
- ◯ Funciones de hardware que llama el usuario
- ◯ Funciones del kernel que permiten al usuario accesar a recursos restringidos
- ◯ Llamada que hace el kernel para realizar una instrucción

30. Llamada del sistema de UNIX que le permite a las aplicaciones el comunicarse entre sí para terminarlas, suspenderlas, o resumirlas.

- ◯ `fork`
- ◯ `exec`
- ◯ `wait`
- ◯ `signal`

31. ¿Cuál de los siguientes eventos **no** genera una transición de modo kernel a usuario?

- ◯ Continuar después de una interrupción
- ◯ Cambiar a un proceso diferente
- ◯ Creación de nuevo proceso
- ◯ Llamada I/O a un dispositivo

32. ¿Qué hace la función `wait` de una variable de condición?

- ◯ Atómicamente libera los seguros y mueve el hilo a la lista de espera
- ◯ Mueve el hilo a la lista de espera atómicamente
- ◯ Libera el seguro global, y duerme al hilo
- ◯ Duerme el hilo atómicamente mientras despierta al siguiente

33. ¿Donde almacena el sistema operativo la toda la información sobre un proceso en particular?

- ◯ Heap
- ◯ Stack
- ◯ *Process control block* (PCB)
- ◯ Memoria

34. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de inicializar el espacio de memoria?

- ◯ Cargar `prog` en la memoria
- ◯ Inicializar el hardware para que se ejecute el incio del programa
- ◯ Copiar los argumentos en la memoria
- ◯ Informar al calendarizador que el programa se puede ejecutar

35. Región de la memoria reservada por el sistema operativo para mantener el estado de las variables locales durante la llamada a funciones.

- ◯ RAM
- ◯ Swap
- ◯ Heap
- ◯ Stack

36. ¿Cuántos procesos son creados durante la ejecución del siguiente programa?

```
main(int argc, char** argv){
  forkthem(5);
}
void forkthem(int n){
  if (n > 0){
    fork();
    forkthem(n-1);
  }
}
```

- ◯ 4
- ◯ 30
- ◯ 5
- ◯ 16

Dado el siguiente código

```
main(int argc, char** argv){
  int child == fork();
  int x = 5;
  if (child == 0) {
    x += 5;
  } else {
    child = fork();
    x += 10;
    if (child) {
      x+=5;
    }
  }
}
```

37. ¿Cuántas copias distintas de la variable `x` existen en la ejecución del programa anterior?

- ◯ 3
- ◯ 4
- ◯ 2
- ◯ 1

38. ¿Cuál de los siguientes **no** es un valor que toma la variable `x` al término de los procesos del programa anterior?

- ◯ 15
- ◯ 20
- ◯ 5
- ◯ 10

39. ¿Qué es un sistema operativo?

- ◯ No es software, pero se encarga de administrar recursos y usuarios
- ◯ Es un software que administra recursos
- ◯ Es una capa de software que administra recursos y usuarios
- ◯ Es un software que administra usuarios

40. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de cargar `prog` en la memoria?

- ◯ Inicializar el PCB en el kernel
- ◯ Copiar los argumentos en la memoria
- ◯ Informar al calendarizador que el programa se puede ejecutar
- ◯ Inicializar el hardware para que se ejecute el incio del programa

41. Dado el siguiente código del programa `main`

```
1  char bar(){
2    int b = 2;
3    return 'a';
4  }
5  int foo(int a, char b){
6    int i=0;
7    bar();
8  }
9  main(){
10   int b=2;
11   foo();
12 }
```

Dibuje el stack completo del programa cuando se encuentra ejecutando la linea número 2. Utilice el número de línea para referirse a la dirección de cada instrucción. ◯I ◯P ◯C

El siguiente código implementa una lista circular.

```
const int MAX = 10;

class Queue{
  int items[MAX];
  int front;
  int nextEmpty;
public:
  Queue() { front = nextEmpty = 0;}
  ~Queue();
  void insert(int item);
  int remove();
}

void Queue::insert(int item){
  items[nextEmpty%MAX] = item;
  nextEmpty++;
}
int Queue::remove(){
  int item;
  item = items[front%MAX];
  front++;
  return item;
}
```

Sin embargo, las funciones `Queue::insert` y `Queue::remove` no están sincronizadas. Por lo que más elementos de los que soporta la lista pueden insertarse o removerse.

Responda lo que se le solicita a continuación. Puede declarar los `Lock` y `CV` que considere necesarios. Tenga en cuenta que la implementación debe de considerar que no puede insertar si la lista está llena, por lo que deberá de detener la función que está insertando. De manera similar, cuando elimine si no hay ningún elemento en la lista deberá esperar a que uno exista.

42. Reescriba la función `Queue::insert` para que inserte de manera segura en la lista circular. ◯I ◯P ◯C

43. Reescriba la función `Queue::remove` para que elimine de manera segura en la lista circular. ◯I ◯P ◯C

udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA

Solemne 1 **SO (CIT 2003-1)**

Instrucciones. Marque las casillas (◯) completamente sin salirse de ellas (por ejemplo ●). Responda a los siguientes cuestionamientos en las hojas que se le entregan **marcando una única opción**. Se utilizará factor de corrección 4 a 1 (las respuestas en blanco no se toman en cuenta). *Las últimas tres preguntas se evaluarán de manera Completa (4 pts.), Parcial (2 pts.), e Incompleta (0 pts.). No llene las casillas de estas últimas tres preguntas, sino que responda en el espacio indicado.*

◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0
◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1
◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2
◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3
◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4
◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5
◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6
◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7
◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8
◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9

⟵ Marque su RUT sin código verificador (el número después del guión), y escriba sus nombres y apellidos abajo.

Nombre(s) y apellido(s):

.....

.....

1. ¿Qué es la exclusión mutua?
- ◯ Propiedad donde los hilos se bloquean mutuamente
- ◯ Propiedad en la que solamente un hilo puede accesar simultáneamente
- ◯ Propiedad de la concurrencia para mantener los hilos excluidos
- ◯ Objeto de sincronización que bloquea los hilos

2. Tipo de kernel en el que la funcionalidad mínima está dentro de él, y el resto se encuentra a nivel de usuario.
- ◯ Macrokernel
- ◯ Monolítico
- ◯ Híbrido
- ◯ Microkernel

3. ¿Cuándo se da una condición de carrera?
- ◯ Varios hilos se pueden intercalar en su ejecución simultánea
- ◯ Varios hilos se ejecutan simultáneamente
- ◯ Existe una sección crítica
- ◯ El estado de la ejecución de un programa depende del intercalado de diferentes hilos

4. Modo de ejecución del procesador donde no se ejecuta ninguna verificación.
- ◯ Modo kernel
- ◯ Modo de usuario
- ◯ Modo dual
- ◯ Modo inseguro

5. ¿Qué es una variable de condición?
- ◯ Variable de sincronización que permite la espera eficiente de un hilo
- ◯ Variable de sincronización que permite la exclusión mutua
- ◯ Varialbe de sincronización que implementa `yield`
- ◯ Objeto de sincronización que implementa `yield` en modo usuario

6. ¿Cuál es el estado de un hilo en el que está listo para ejecutarse pero no está en el procesador?
- ◯ Listo (*ready*)
- ◯ Inicializado (*init*)
- ◯ Esperando (*wait*)
- ◯ Ejecutandose (*running*)

7. Tipo de kernel en el que la mayoría de la funcionalidad de éste se encuentra dentro de él.
- ◯ Microkernel
- ◯ Macrokernel
- ◯ Monolítico
- ◯ Híbrido

8. ¿Qué característica del sistema operativo permite que opere independientemente del hardware en la computadora?
- ◯ Portabilidad
- ◯ Virtualización
- ◯ Ilusionista
- ◯ Compartimiento de recursos

9. ¿Qué es un sistema operativo?
- ◯ No es software, pero se encarga de administrar recursos y usuarios
- ◯ Es un software que administra usuarios
- ◯ Es una capa de software que administra recursos y usuarios
- ◯ Es un software que administra recursos

10. ¿Cuál es el principal problema de los hilos a nivel de usuario?
- ◯ No podemos tener varios procesos por hilo
- ◯ No podemos tener varios hilos por proceso
- ◯ No podemos calendarizar hilos dentro del proceso
- ◯ El bloqueo del proceso, bloquea todos los hilos

11. ¿Qué es concurrencia?
- ◯ Realizar múltiples actividades simultáneamente
- ◯ Realizar una actividad sin la interrupción de otra
- ◯ Realizar múltiples actividades una después de la otra
- ◯ Realizar múltiples actividades sin la interrupción de otras

12. Es una señal asíncrona al procesador que indica que ocurrió un evento que requiere su atención.
- ◯ Señal de I/O
- ◯ Señal de software
- ◯ Interrupción
- ◯ Trampa

13. ¿Donde almacena el sistema operativo la toda la información sobre un proceso en particular?

- ◯ *Process control block* (PCB)
- ◯ Memoria
- ◯ Stack
- ◯ Heap

Dado el siguiente código

```
main(int argc, char** argv){
  int child == fork();
  int x = 5;
  if (child == 0) {
    x += 5;
  } else {
    child = fork();
    x += 10;
    if (child) {
      x+=5;
    }
  }
}
```

14. ¿Cuál de los siguientes **no** es un valor que toma la variable `x` al término de los procesos del programa anterior?

- ◯ 5
- ◯ 10
- ◯ 15
- ◯ 20

15. ¿Cuántas copias distintas de la variable `x` existen en la ejecución del programa anterior?

- ◯ 3
- ◯ 2
- ◯ 1
- ◯ 4

16. ¿Cuál es el primer paso en la creación del proceso del programa `prog`?

- ◯ Inicializar la memoria
- ◯ Inicializar el hardware para que se ejecute el proceso desde el inicio
- ◯ Crear e inicializar el PCB en el kernel
- ◯ Copiar `prog` a la memoria

17. ¿Qué es una llamada de sistema?

- ◯ Funciones de hardware que llama el usuario
- ◯ Funciones del kernel que permiten al usuario acceder a recursos restringidos
- ◯ Llamada que hace el kernel para realizar una instrucción
- ◯ Código del kernel que ejecuta código de usuario

18. Es una señal síncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Señal de software
- ◯ Señal de I/O
- ◯ Trampa
- ◯ Interrupción

19. ¿Cuál de los siguientes elementos **no** pertenece al *TCB*?

- ◯ Estado del hilo
- ◯ Puntero al stack frame
- ◯ Registros salvados
- ◯ Variables locales

20. ¿Qué son las direcciones de memoria virtuales?

- ◯ Memoria que reserva el sistema operativo para el uso de máquinas virtuales
- ◯ Memoria adicional que crea el sistema operativo para engañar a las aplicaciones
- ◯ Capa adicional que utiliza el sistema operativo en modo kernel
- ◯ Capa de indirección que le da flexibilidad al sistema operativo para administrar la memoria

21. ¿Cuál de los siguientes elementos **no** es compartido por los hilos?

- ◯ Stack
- ◯ Variables globales
- ◯ Heap
- ◯ Código

22. ¿Qué es un pipe?

- ◯ Es un buffer del kernel entre dos descriptores de archivos
- ◯ Es un buffer de usuario entre dos descriptores de archivos
- ◯ Es un buffer del kernel entre dos procesos
- ◯ Es un buffer de usuario entre dos procesos

23. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de inicializar el espacio de memoria?

- ◯ Inicializar el hardware para que se ejecute el inicio del programa
- ◯ Copiar los argumentos en la memoria
- ◯ Cargar `prog` en la memoria
- ◯ Informar al calendarizador que el programa se puede ejecutar

24. ¿Qué es un hilo?

- ◯ Una secuencia de ejecución que puede ser calendarizada independientemente
- ◯ Varias secuencias de ejecución que pueden ser calendarizadas independientemente
- ◯ Una secuencia de ejecución que es calendarizada dependientemente
- ◯ Varias secuencias de ejecución que pueden ser calendarizadas dependientemente

25. Modo de ejecución del procesador en el que verifica cada instrucción antes de ejecutarla.

- ◯ Modo de usuario
- ◯ Modo dual
- ◯ Modo kernel
- ◯ Modo seguro

26. ¿Cuál de las siguientes afirmaciones es verdadera respecto a *spin wait*?

- ◯ Es justificable en monoprocesadores
- ◯ Produce *deadlocks*
- ◯ No se debe de usar nunca
- ◯ Es justificable en multiprocesadores

27. ¿Qué hace la función `wait` de una variable de condición?

- ◯ Libera el seguro global, y duerme al hilo
- ◯ Atómicamente libera los seguros y mueve el hilo a la lista de espera
- ◯ Duerme el hilo atómicamente mientras despierta al siguiente
- ◯ Mueve el hilo a la lista de espera atómicamente

28. Llamada del sistema de UNIX que le permite a las aplicaciones el comunicarse entre sí para terminarlas, suspenderlas, o resumirlas.

- ◯ `exec`
- ◯ `signal`
- ◯ `wait`
- ◯ `fork`

29. ¿Cuántos procesos son creados durante la ejecución del siguiente programa?

```
main(int argc, char** argv){
  forkthem(5);
}
void forkthem(int n){
  if (n > 0){
    fork();
    forkthem(n-1);
  }
}
```

- ◯ 30
- ◯ 4
- ◯ 5
- ◯ 16

30. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de cargar `prog` en la memoria?

- ◯ Inicializar el hardware para que se ejecute el incio del programa
- ◯ Inicializar el PCB en el kernel
- ◯ Copiar los argumentos en la memoria
- ◯ Informar al calendarizador que el programa se puede ejecutar

31. ¿Cuál de las siguientes **no** es una característica de un descriptor de archivo de UNIX?

- ◯ Se leen a través de un buffer
- ◯ Se cierran completamente después de usar
- ◯ Se escriben a través de bytes
- ◯ Se abren antes de usar

32. Región de la memoria reservada por el sistema operativo para alojar estructuras de datos que el proceso pueda necesitar.

- ◯ Swap
- ◯ Heap
- ◯ RAM
- ◯ Stack

33. ¿Qué es la ejecución de una aplicación con permisos restringidos?

- ◯ Hilo
- ◯ Programa
- ◯ Aplicación
- ◯ Proceso

34. Región de la memoria reservada por el sistema operativo para mantener el estado de las variables locales durante la llamada a funciones.

- ◯ Heap
- ◯ Swap
- ◯ RAM
- ◯ Stack

35. ¿Qué hace la función `yield` de un hilo?

- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a terminado (*finished*)
- ◯ Duerme al hilo por un tiempo determinado, y cambia el estado a esperando (*waiting*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a listo (*ready*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a esperando (*waiting*)

36. ¿Qué es una operación atómica?

- ◯ Es una operación del kernel que no puede ser dividida
- ◯ Es una operación indivisible que no puede ser dividida
- ◯ Es una operación que deshabilita las interrupciones
- ◯ Es una operación de hardware que no puede ser dividida

37. ¿Cuál de las siguientes es una característica de `fork`?

- ◯ Copiar el proceso padre completamente
- ◯ Copiar a la memoria el código del programa del proceso padre
- ◯ Copiar el proceso padre con privilegios distintos
- ◯ Crear una copia del proceso padre, pero no puede ser confiado igual que él

38. ¿Cuál de las siguientes **no** es una característica de un hilo?

- ◯ Comparte el código del programa
- ◯ Tiene un contador de programa propio
- ◯ Tiene un segmento de datos propio
- ◯ Tiene un stack propio

39. ¿Qué es el multi-hilado preventivo?

- ◯ Es cuando los hilos entregan el procesador voluntariamente, no son interrumpidos
- ◯ Es cuando los hilos no usan el procesador hasta que se les entrega indefinidamente
- ◯ Es cuando los hilos que se ejecutan no pueden descalendarizarse
- ◯ Es cuando los hilos que se ejecutan pueden ser cambiados indistintamente

40. ¿Cuál de los siguientes eventos **no** genera una transición de modo kernel a usuario?

- ◯ Creación de nuevo proceso
- ◯ Cambiar a un proceso diferente
- ◯ Continuar después de una interrupción
- ◯ Llamada I/O a un dispositivo

41. Dado el siguiente código del programa `main`

```
1  char bar(){
2    int b = 2;
3    return 'a';
4  }
5  int foo(int a, char b){
6    int i=0;
7    bar();
8  }
9  main(){
10   int b=2;
11   foo();
12 }
```

Dibuje el stack completo del programa cuando se encuentra ejecutando la linea número 2. Utilice el número de línea para referirse a la dirección de cada instrucción. ◯I ◯P ◯C

El siguiente código implementa una lista circular.

```
const int MAX = 10;

class Queue{
  int items[MAX];
  int front;
  int nextEmpty;
public:
  Queue() { front = nextEmpty = 0;}
  ~Queue();
  void insert(int item);
  int remove();
}

void Queue::insert(int item){
  items[nextEmpty%MAX] = item;
  nextEmpty++;
}
int Queue::remove(){
  int item;
  item = items[front%MAX];
  front++;
  return item;
}
```

Sin embargo, las funciones `Queue::insert` y `Queue::remove` no están sincronizadas. Por lo que más elementos de los que soporta la lista pueden insertarse o removerse.

Responda lo que se le solicita a continuación. Puede declarar los `Lock` y `CV` que considere necesarios. Tenga en cuenta que la implementación debe de considerar que no puede insertar si la lista está llena, por lo que deberá de detener la función que está insertando. De manera similar, cuando elimine si no hay ningún elemento en la lista deberá esperar a que uno exista.

42. Reescriba la función `Queue::insert` para que inserte de manera segura en la lista circular. ◯I ◯P ◯C

43. Reescriba la función `Queue::remove` para que elimine de manera segura en la lista circular. ◯I ◯P ◯C

udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA

Solemne 1 **SO (CIT 2003-1)**

Instrucciones. Marque las casillas (◯) completamente sin salirse de ellas (por ejemplo ●). Responda a los siguientes cuestionamientos en las hojas que se le entregan **marcando una única opción**. Se utilizará factor de corrección 4 a 1 (las respuestas en blanco no se toman en cuenta). *Las últimas tres preguntas se evaluarán de manera Completa (4 pts.), Parcial (2 pts.), e Incompleta (0 pts.). No llene las casillas de estas últimas tres preguntas, sino que responda en el espacio indicado.*

◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0
◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1
◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2
◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3
◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4
◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5
◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6
◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7
◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8
◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9

⟵ Marque su RUT sin código verificador (el número después del guión), y escriba sus nombres y apellidos abajo.

Nombre(s) y apellido(s):

.....

.....

1. Tipo de kernel en el que la funcionalidad mínima está dentro de él, y el resto se encuentra a nivel de usuario.

- ◯ Híbrido
- ◯ Monolítico
- ◯ Macrokernel
- ◯ Microkernel

2. ¿Cuál de las siguientes es una característica de `fork`?

- ◯ Copiar el proceso padre completamente
- ◯ Crear una copia del proceso padre, pero no puede ser confiado igual que él
- ◯ Copiar a la memoria el código del programa del proceso padre
- ◯ Copiar el proceso padre con privilegios distintos

3. Región de la memoria reservada por el sistema operativo para mantener el estado de las variables locales durante la llamada a funciones.

- ◯ RAM
- ◯ Stack
- ◯ Swap
- ◯ Heap

4. Región de la memoria reservada por el sistema operativo para alojar estructuras de datos que el proceso pueda necesitar.

- ◯ Stack
- ◯ RAM
- ◯ Heap
- ◯ Swap

5. ¿Cuál es el principal problema de los hilos a nivel de usuario?

- ◯ El bloqueo del proceso, bloquea todos los hilos
- ◯ No podemos tener varios procesos por hilo
- ◯ No podemos calendarizar hilos dentro del proceso
- ◯ No podemos tener varios hilos por proceso

6. Tipo de kernel en el que la mayoría de la funcionalidad de éste se encuentra dentro de él.

- ◯ Microkernel
- ◯ Monolítico
- ◯ Macrokernel
- ◯ Híbrido

7. ¿Cuál de las siguientes **no** es una característica de un descriptor de archivo de UNIX?

- ◯ Se escriben a través de bytes
- ◯ Se leen a través de un buffer
- ◯ Se abren antes de usar
- ◯ Se cierran completamente después de usar

8. ¿Qué hace la función `yield` de un hilo?

- ◯ Duerme al hilo por un tiempo determinado, y cambia el estado a esperando (*waiting*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a listo (*ready*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a esperando (*waiting*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a terminado (*finished*)

9. ¿Qué es un hilo?

- ◯ Varias secuencias de ejecución que pueden ser calendarizadas dependientemente
- ◯ Una secuencia de ejecución que es calendarizada dependientemente
- ◯ Una secuencia de ejecución que puede ser calendarizada independientemente
- ◯ Varias secuencias de ejecución que pueden ser calendarizadas independientemente

10. ¿Qué característica del sistema operativo permite que opere independientemente del hardware en la computadora?

- ◯ Virtualización
- ◯ Portabilidad
- ◯ Ilusionista
- ◯ Compartimiento de recursos

11. ¿Cuál es el primer paso en la creación del proceso del programa `prog`?

- ◯ Inicializar el hardware para que se ejecute el proceso desde el inicio
- ◯ Crear e inicializar el PCB en el kernel
- ◯ Inicializar la memoria
- ◯ Copiar `prog` a la memoria

12. ¿Qué hace la función `wait` de una variable de condición?
- ◯ Libera el seguro global, y duerme al hilo
- ◯ Mueve el hilo a la lista de espera atómicamente
- ◯ Duerme el hilo atómicamente mientras despierta al siguiente
- ◯ Atómicamente libera los seguros y mueve el hilo a la lista de espera

13. ¿Cuál de los siguientes elementos **no** pertenece al *TCB*?
- ◯ Estado del hilo
- ◯ Registros salvados
- ◯ Variables locales
- ◯ Puntero al stack frame

14. ¿Qué es una variable de condición?
- ◯ Objeto de sincronización que implementa `yield` en modo usuario
- ◯ Varialbe de sincronización que implementa `yield`
- ◯ Variable de sincronización que permite la exclusión mutua
- ◯ Variable de sincronización que permite la espera eficiente de un hilo

15. ¿Qué es concurrencia?
- ◯ Realizar múltiples actividades simultáneamente
- ◯ Realizar una actividad sin la interrupción de otra
- ◯ Realizar múltiples actividades sin la interrupción de otras
- ◯ Realizar múltiples actividades una después de la otra

16. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de inicializar el espacio de memoria?
- ◯ Cargar `prog` en la memoria
- ◯ Informar al calendarizador que el programa se puede ejecutar
- ◯ Copiar los argumentos en la memoria
- ◯ Inicializar el hardware para que se ejecute el incio del programa

17. Modo de ejecución del procesador en el que verifica cada instrucción antes de ejecutarla.
- ◯ Modo seguro
- ◯ Modo dual
- ◯ Modo de usuario
- ◯ Modo kernel

18. ¿Cuántos procesos son creados durante la ejecución del siguiente programa?

```
main(int argc, char** argv){
  forkthem(5);
}
void forkthem(int n){
  if (n > 0){
    fork();
    forkthem(n-1);
  }
}
```

- ◯ 5
- ◯ 30
- ◯ 16
- ◯ 4

19. ¿Cuál es el estado de un hilo en el que está listo para ejecutarse pero no está en el procesador?
- ◯ Inicializado (*init*)
- ◯ Esperando (*wait*)
- ◯ Listo (*ready*)
- ◯ Ejecutandose (*running*)

20. ¿Cuándo se da una condición de carrera?
- ◯ El estado de la ejecución de un programa depende del intercalado de diferentes hilos
- ◯ Varios hilos se pueden intercalar en su ejecución simultánea
- ◯ Varios hilos se ejecutan simultáneamente
- ◯ Existe una sección crítica

21. Modo de ejecución del procesador donde no se ejecuta ninguna verificación.
- ◯ Modo kernel
- ◯ Modo inseguro
- ◯ Modo de usuario
- ◯ Modo dual

22. Es una señal asíncrona al procesador que indica que ocurrió un evento que requiere su atención.
- ◯ Señal de I/O
- ◯ Interrupción
- ◯ Trampa
- ◯ Señal de software

23. Es una señal síncrona al procesador que indica que ocurrió un evento que requiere su atención.
- ◯ Interrupción
- ◯ Trampa
- ◯ Señal de software
- ◯ Señal de I/O

24. ¿Donde almacena el sistema operativo la toda la información sobre un proceso en particular?
- ◯ *Process control block* (PCB)
- ◯ Stack
- ◯ Memoria
- ◯ Heap

25. ¿Cuál de las siguientes afirmaciones es verdadera respecto a *spin wait*?
- ◯ Es justificable en monoprocesadores
- ◯ No se debe de usar nunca
- ◯ Es justificable en multiprocesadores
- ◯ Produce *deadlocks*

26. ¿Cuál de las siguientes **no** es una característica de un hilo?
- ◯ Tiene un contador de programa propio
- ◯ Tiene un segmento de datos propio
- ◯ Comparte el código del programa
- ◯ Tiene un stack propio

27. ¿Cuál de los siguientes eventos **no** genera una transición de modo kernel a usuario?
- ◯ Cambiar a un proceso diferente
- ◯ Creación de nuevo proceso
- ◯ Continuar después de una interrupción
- ◯ Llamada I/O a un dispositivo

28. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de cargar `prog` en la memoria?
- ◯ Informar al calendarizador que el programa se puede ejecutar
- ◯ Inicializar el hardware para que se ejecute el incio del programa
- ◯ Copiar los argumentos en la memoria
- ◯ Inicializar el PCB en el kernel

29. ¿Qué es la exclusión mutua?

- ◯ Propiedad de la concurrencia para mantener los hilos excluidos
- ◯ Propiedad en la que solamente un hilo puede accesar simultáneamente
- ◯ Propiedad donde los hilos se bloquean mutuamente
- ◯ Objeto de sincronización que bloquea los hilos

30. ¿Qué es un pipe?

- ◯ Es un buffer del kernel entre dos descriptores de archivos
- ◯ Es un buffer del kernel entre dos procesos
- ◯ Es un buffer de usuario entre dos procesos
- ◯ Es un buffer de usuario entre dos descriptores de archivos

31. Llamada del sistema de UNIX que le permite a las aplicaciones el comunicarse entre sí para terminarlas, suspenderlas, o resumirlas.

- ◯ `wait`
- ◯ `exec`
- ◯ `signal`
- ◯ `fork`

32. ¿Qué es el multi-hilado preventivo?

- ◯ Es cuando los hilos que se ejecutan no pueden descalendarizarse
- ◯ Es cuando los hilos no usan el procesador hasta que se les entrega indefinidamente
- ◯ Es cuando los hilos entregan el procesador voluntariamente, no son interrumpidos
- ◯ Es cuando los hilos que se ejecutan pueden ser cambiados indistintamente

33. ¿Qué es una operación atómica?

- ◯ Es una operación indivisible que no puede ser dividida
- ◯ Es una operación que deshabilita las interrupciones
- ◯ Es una operación del kernel que no puede ser dividida
- ◯ Es una operación de hardware que no puede ser dividida

34. ¿Qué es un sistema operativo?

- ◯ Es una capa de software que administra recursos y usuarios
- ◯ Es un software que administra usuarios
- ◯ Es un software que administra recursos
- ◯ No es software, pero se encarga de administrar recursos y usuarios

35. ¿Qué es una llamada de sistema?

- ◯ Funciones de hardware que llama el usuario
- ◯ Código del kernel que ejecuta código de usuario
- ◯ Llamada que hace el kernel para realizar una instrucción
- ◯ Funciones del kernel que permiten al usuario accesar a recursos restringidos

Dado el siguiente código

```
main(int argc, char** argv){
  int child == fork();
  int x = 5;
  if (child == 0) {
    x += 5;
  } else {
    child = fork();
    x += 10;
    if (child) {
      x+=5;
    }
  }
}
```

36. ¿Cuál de los siguientes **no** es un valor que toma la variable `x` al término de los procesos del programa anterior?

- ◯ 15
- ◯ 5
- ◯ 10
- ◯ 20

37. ¿Cuántas copias distintas de la variable `x` existen en la ejecución del programa anterior?

- ◯ 2
- ◯ 3
- ◯ 1
- ◯ 4

38. ¿Qué es la ejecución de una aplicación con permisos restringidos?

- ◯ Programa
- ◯ Aplicación
- ◯ Hilo
- ◯ Proceso

39. ¿Cuál de los siguientes elementos **no** es compartido por los hilos?

- ◯ Stack
- ◯ Variables globales
- ◯ Código
- ◯ Heap

40. ¿Qué son las direcciones de memoria virtuales?

- ◯ Memoria adicional que crea el sistema operativo para engañar a las aplicaciones
- ◯ Capa de indirección que le da flexibilidad al sistema operativo para administrar la memoria
- ◯ Capa adicional que utiliza el sistema operativo en modo kernel
- ◯ Memoria que reserva el sistema operativo para el uso de máquinas virtuales

41. Dado el siguiente código del programa `main`

```
1  char bar(){
2    int b = 2;
3    return 'a';
4  }
5  int foo(int a, char b){
6    int i=0;
7    bar();
8  }
9  main(){
10   int b=2;
11   foo();
12 }
```

Dibuje el stack completo del programa cuando se encuentra ejecutando la linea número 2. Utilice el número de línea para referirse a la dirección de cada instrucción. ◯I ◯P ◯C

El siguiente código implementa una lista circular.

```
const int MAX = 10;

class Queue{
  int items[MAX];
  int front;
  int nextEmpty;
public:
  Queue() { front = nextEmpty = 0;}
  ~Queue();
  void insert(int item);
  int remove();
}

void Queue::insert(int item){
  items[nextEmpty%MAX] = item;
  nextEmpty++;
}
int Queue::remove(){
  int item;
  item = items[front%MAX];
  front++;
  return item;
}
```

Sin embargo, las funciones `Queue::insert` y `Queue::remove` no están sincronizadas. Por lo que más elementos de los que soporta la lista pueden insertarse o removerse.

Responda lo que se le solicita a continuación. Puede declarar los `Lock` y `CV` que considere necesarios. Tenga en cuenta que la implementación debe de considerar que no puede insertar si la lista está llena, por lo que deberá de detener la función que está insertando. De manera similar, cuando elimine si no hay ningún elemento en la lista deberá esperar a que uno exista.

42. Reescriba la función `Queue::insert` para que inserte de manera segura en la lista circular. ◯I ◯P ◯C

43. Reescriba la función `Queue::remove` para que elimine de manera segura en la lista circular. ◯I ◯P ◯C

udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA

Solemne 1 **SO (CIT 2003-1)**

Instrucciones. Marque las casillas (◯) completamente sin salirse de ellas (por ejemplo ●). Responda a los siguientes cuestionamientos en las hojas que se le entregan **marcando una única opción**. Se utilizará factor de corrección 4 a 1 (las respuestas en blanco no se toman en cuenta). *Las últimas tres preguntas se evaluarán de manera Completa (4 pts.), Parcial (2 pts.), e Incompleta (0 pts.). No llene las casillas de estas últimas tres preguntas, sino que responda en el espacio indicado.*

◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0
◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1
◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2
◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3
◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4
◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5
◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6
◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7
◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8
◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9

⟵ Marque su RUT sin código verificador (el número después del guión), y escriba sus nombres y apellidos abajo.

Nombre(s) y apellido(s):

.....

.....

Dado el siguiente código

```
main(int argc, char** argv){
  int child == fork();
  int x = 5;
  if (child == 0) {
    x += 5;
  } else {
    child = fork();
    x += 10;
    if (child) {
      x+=5;
    }
  }
}
```

1. ¿Cuántas copias distintas de la variable `x` existen en la ejecución del programa anterior?

◯ 2 ◯ 3
◯ 4 ◯ 1

2. ¿Cuál de los siguientes **no** es un valor que toma la variable `x` al término de los procesos del programa anterior?

◯ 10 ◯ 5
◯ 15 ◯ 20

3. Región de la memoria reservada por el sistema operativo para mantener el estado de las variables locales durante la llamada a funciones.

◯ Heap ◯ RAM
◯ Stack ◯ Swap

4. ¿Qué es una variable de condición?

◯ Varialbe de sincronización que implementa `yield` ◯ Variable de sincronización que permite la exclusión mutua
◯ Variable de sincronización que permite la espera eficiente de un hilo ◯ Objeto de sincronización que implementa `yield` en modo usuario

5. ¿Qué es una llamada de sistema?

◯ Llamada que hace el kernel para realizar una instrucción ◯ Funciones del kernel que permiten al usuario accesar a recursos restringidos
◯ Funciones de hardware que llama el usuario ◯ Código del kernel que ejecuta código de usuario

6. ¿Qué hace la función `yield` de un hilo?

◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a esperando (*waiting*) ◯ Duerme al hilo por un tiempo determinado, y cambia el estado a esperando (*waiting*)
◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a listo (*ready*) ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a terminado (*finished*)

7. ¿Cuántos procesos son creados durante la ejecución del siguiente programa?

```
main(int argc, char** argv){
  forkthem(5);
}
void forkthem(int n){
  if (n > 0){
    fork();
    forkthem(n-1);
  }
}
```

◯ 4 ◯ 5
◯ 16 ◯ 30

8. ¿Cuál de las siguientes afirmaciones es verdadera respecto a *spin wait*?

◯ Produce *deadlocks* ◯ No se debe de usar nunca
◯ Es justificable en multiprocesadores ◯ Es justificable en monoprocesadores

udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA

9. Modo de ejecución del procesador en el que verifica cada instrucción antes de ejecutarla.

- ◯ Modo dual
- ◯ Modo de usuario
- ◯ Modo seguro
- ◯ Modo kernel

10. ¿Cuál es el estado de un hilo en el que está listo para ejecutarse pero no está en el procesador?

- ◯ Listo (*ready*)
- ◯ Inicializado (*init*)
- ◯ Esperando (*wait*)
- ◯ Ejecutandose (*running*)

11. ¿Cuál de las siguientes **no** es una característica de un hilo?

- ◯ Tiene un contador de programa propio
- ◯ Comparte el código del programa
- ◯ Tiene un stack propio
- ◯ Tiene un segmento de datos propio

12. ¿Cuál es el primer paso en la creación del proceso del programa `prog`?

- ◯ Inicializar la memoria
- ◯ Copiar `prog` a la memoria
- ◯ Crear e inicializar el PCB en el kernel
- ◯ Inicializar el hardware para que se ejecute el proceso desde el inicio

13. ¿Qué es un sistema operativo?

- ◯ No es software, pero se encarga de administrar recursos y usuarios
- ◯ Es un software que administra recursos
- ◯ Es una capa de software que administra recursos y usuarios
- ◯ Es un software que administra usuarios

14. ¿Qué es el multi-hilado preventivo?

- ◯ Es cuando los hilos que se ejecutan pueden ser cambiados indistintamente
- ◯ Es cuando los hilos entregan el procesador voluntariamente, no son interrumpidos
- ◯ Es cuando los hilos que se ejecutan no pueden descalendarizarse
- ◯ Es cuando los hilos no usan el procesador hasta que se les entrega indefinidamente

15. ¿Qué es un pipe?

- ◯ Es un buffer del kernel entre dos descriptores de archivos
- ◯ Es un buffer de usuario entre dos descriptores de archivos
- ◯ Es un buffer de usuario entre dos procesos
- ◯ Es un buffer del kernel entre dos procesos

16. ¿Qué son las direcciones de memoria virtuales?

- ◯ Capa adicional que utiliza el sistema operativo en modo kernel
- ◯ Capa de indirección que le da flexibilidad al sistema operativo para administrar la memoria
- ◯ Memoria adicional que crea el sistema operativo para engañar a las aplicaciones
- ◯ Memoria que reserva el sistema operativo para el uso de máquinas virtuales

17. ¿Cuál es el principal problema de los hilos a nivel de usuario?

- ◯ No podemos calendarizar hilos dentro del proceso
- ◯ El bloqueo del proceso, bloquea todos los hilos
- ◯ No podemos tener varios procesos por hilo
- ◯ No podemos tener varios hilos por proceso

18. ¿Qué hace la función `wait` de una variable de condición?

- ◯ Mueve el hilo a la lista de espera atómicamente
- ◯ Duerme el hilo atómicamente mientras despierta al siguiente
- ◯ Atómicamente libera los seguros y mueve el hilo a la lista de espera
- ◯ Libera el seguro global, y duerme al hilo

19. ¿Qué es un hilo?

- ◯ Varias secuencias de ejecución que pueden ser calendarizadas dependientemente
- ◯ Varias secuencias de ejecución que pueden ser calendarizadas independientemente
- ◯ Una secuencia de ejecución que puede ser calendarizada independientemente
- ◯ Una secuencia de ejecución que es calendarizada dependientemente

20. ¿Qué es una operación atómica?

- ◯ Es una operación del kernel que no puede ser dividida
- ◯ Es una operación que deshabilita las interrupciones
- ◯ Es una operación de hardware que no puede ser dividida
- ◯ Es una operación indivisible que no puede ser dividida

21. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de inicializar el espacio de memoria?

- ◯ Cargar `prog` en la memoria
- ◯ Copiar los argumentos en la memoria
- ◯ Informar al calendarizador que el programa se puede ejecutar
- ◯ Inicializar el hardware para que se ejecute el incio del programa

22. ¿Cuál de las siguientes es una característica de `fork`?

- ◯ Copiar a la memoria el código del programa del proceso padre
- ◯ Crear una copia del proceso padre, pero no puede ser confiado igual que él
- ◯ Copiar el proceso padre completamente
- ◯ Copiar el proceso padre con privilegios distintos

23. Modo de ejecución del procesador donde no se ejecuta ninguna verificación.

- ◯ Modo dual
- ◯ Modo kernel
- ◯ Modo inseguro
- ◯ Modo de usuario

24. ¿Qué es la ejecución de una aplicación con permisos restringidos?

- ◯ Proceso
- ◯ Hilo
- ◯ Aplicación
- ◯ Programa

25. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de cargar `prog` en la memoria?

- ◯ Inicializar el hardware para que se ejecute el incio del programa
- ◯ Copiar los argumentos en la memoria
- ◯ Inicializar el PCB en el kernel
- ◯ Informar al calendarizador que el programa se puede ejecutar

26. ¿Qué es concurrencia?

- ◯ Realizar múltiples actividades simultáneamente
- ◯ Realizar múltiples actividades una después de la otra
- ◯ Realizar una actividad sin la interrupción de otra
- ◯ Realizar múltiples actividades sin la interrupción de otras

27. ¿Qué es la exclusión mutua?

- ◯ Objeto de sincronización que bloquea los hilos
- ◯ Propiedad en la que solamente un hilo puede accesar simultáneamente
- ◯ Propiedad de la concurrencia para mantener los hilos excluidos
- ◯ Propiedad donde los hilos se bloquean mutuamente

28. ¿Qué característica del sistema operativo permite que opere independientemente del hardware en la computadora?

- ◯ Portabilidad
- ◯ Compartimiento de recursos
- ◯ Virtualización
- ◯ Ilusionista

29. Tipo de kernel en el que la funcionalidad mínima está dentro de él, y el resto se encuentra a nivel de usuario.

- ◯ Híbrido
- ◯ Macrokernel
- ◯ Microkernel
- ◯ Monolítico

30. Es una señal síncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Interrupción
- ◯ Trampa
- ◯ Señal de software
- ◯ Señal de I/O

31. Región de la memoria reservada por el sistema operativo para alojar estructuras de datos que el proceso pueda necesitar.

- ◯ Swap
- ◯ Stack
- ◯ RAM
- ◯ Heap

32. ¿Cuál de las siguientes **no** es una característica de un descriptor de archivo de UNIX?

- ◯ Se escriben a través de bytes
- ◯ Se abren antes de usar
- ◯ Se cierran completamente después de usar
- ◯ Se leen a través de un buffer

33. ¿Cuándo se da una condición de carrera?

- ◯ El estado de la ejecución de un programa depende del intercalado de diferentes hilos
- ◯ Existe una sección crítica
- ◯ Varios hilos se ejecutan simultáneamente
- ◯ Varios hilos se pueden intercalar en su ejecución simultánea

34. ¿Cuál de los siguientes elementos **no** pertenece al *TCB*?

- ◯ Puntero al stack frame
- ◯ Variables locales
- ◯ Estado del hilo
- ◯ Registros salvados

35. Llamada del sistema de UNIX que le permite a las aplicaciones el comunicarse entre sí para terminarlas, suspenderlas, o resumirlas.

- ◯ `exec`
- ◯ `fork`
- ◯ `wait`
- ◯ `signal`

36. ¿Cuál de los siguientes elementos **no** es compartido por los hilos?

- ◯ Stack
- ◯ Heap
- ◯ Variables globales
- ◯ Código

37. Es una señal asíncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Señal de I/O
- ◯ Señal de software
- ◯ Trampa
- ◯ Interrupción

38. Tipo de kernel en el que la mayoría de la funcionalidad de éste se encuentra dentro de él.

- ◯ Monolítico
- ◯ Macrokernel
- ◯ Microkernel
- ◯ Híbrido

39. ¿Cuál de los siguientes eventos **no** genera una transición de modo kernel a usuario?

- ◯ Cambiar a un proceso diferente
- ◯ Llamada I/O a un dispositivo
- ◯ Creación de nuevo proceso
- ◯ Continuar después de una interrupción

40. ¿Donde almacena el sistema operativo la toda la información sobre un proceso en particular?

- ◯ Heap
- ◯ Memoria
- ◯ *Process control block* (PCB)
- ◯ Stack

41. Dado el siguiente código del programa `main`

```
1  char bar(){
2    int b = 2;
3    return 'a';
4  }
5  int foo(int a, char b){
6    int i=0;
7    bar();
8  }
9  main(){
10   int b=2;
11   foo();
12 }
```

Dibuje el stack completo del programa cuando se encuentra ejecutando la linea número 2. Utilice el número de línea para referirse a la dirección de cada instrucción. ◯I ◯P ◯C

El siguiente código implementa una lista circular.

```
const int MAX = 10;

class Queue{
  int items[MAX];
  int front;
  int nextEmpty;
public:
  Queue() { front = nextEmpty = 0;}
  ~Queue();
  void insert(int item);
  int remove();
}

void Queue::insert(int item){
  items[nextEmpty%MAX] = item;
  nextEmpty++;
}
int Queue::remove(){
  int item;
  item = items[front%MAX];
  front++;
  return item;
}
```

Sin embargo, las funciones `Queue::insert` y `Queue::remove` no están sincronizadas. Por lo que más elementos de los que soporta la lista pueden insertarse o removerse.

Responda lo que se le solicita a continuación. Puede declarar los `Lock` y `CV` que considere necesarios. Tenga en cuenta que la implementación debe de considerar que no puede insertar si la lista está llena, por lo que deberá de detener la función que está insertando. De manera similar, cuando elimine si no hay ningún elemento en la lista deberá esperar a que uno exista.

42. Reescriba la función `Queue::insert` para que inserte de manera segura en la lista circular. ◯I ◯P ◯C

43. Reescriba la función `Queue::remove` para que elimine de manera segura en la lista circular. ◯I ◯P ◯C

udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA

Solemne 1 **SO (CIT 2003-1)**

Instrucciones. Marque las casillas (◯) completamente sin salirse de ellas (por ejemplo ●). Responda a los siguientes cuestionamientos en las hojas que se le entregan **marcando una única opción**. Se utilizará factor de corrección 4 a 1 (las respuestas en blanco no se toman en cuenta). *Las últimas tres preguntas se evaluarán de manera Completa (4 pts.), Parcial (2 pts.), e Incompleta (0 pts.). No llene las casillas de estas últimas tres preguntas, sino que responda en el espacio indicado.*

◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0
◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1
◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2
◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3
◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4
◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5
◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6
◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7
◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8
◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9

⟵ Marque su RUT sin código verificador (el número después del guión), y escriba sus nombres y apellidos abajo.

Nombre(s) y apellido(s):

.....

.....

1. ¿Qué es la ejecución de una aplicación con permisos restringidos?

- ◯ Aplicación
- ◯ Proceso
- ◯ Programa
- ◯ Hilo

2. ¿Qué es una llamada de sistema?

- ◯ Código del kernel que ejecuta código de usuario
- ◯ Funciones de hardware que llama el usuario
- ◯ Llamada que hace el kernel para realizar una instrucción
- ◯ Funciones del kernel que permiten al usuario accesar a recursos restringidos

3. ¿Cuál de los siguientes eventos **no** genera una transición de modo kernel a usuario?

- ◯ Cambiar a un proceso diferente
- ◯ Llamada I/O a un dispositivo
- ◯ Continuar después de una interrupción
- ◯ Creación de nuevo proceso

4. Modo de ejecución del procesador en el que verifica cada instrucción antes de ejecutarla.

- ◯ Modo de usuario
- ◯ Modo kernel
- ◯ Modo dual
- ◯ Modo seguro

5. ¿Cuándo se da una condición de carrera?

- ◯ Existe una sección crítica
- ◯ Varios hilos se pueden intercalar en su ejecución simultánea
- ◯ El estado de la ejecución de un programa depende del intercalado de diferentes hilos
- ◯ Varios hilos se ejecutan simultáneamente

6. ¿Qué característica del sistema operativo permite que opere independientemente del hardware en la computadora?

- ◯ Compartimiento de recursos
- ◯ Ilusionista
- ◯ Portabilidad
- ◯ Virtualización

7. Es una señal síncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Señal de I/O
- ◯ Señal de software
- ◯ Trampa
- ◯ Interrupción

8. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de cargar `prog` en la memoria?

- ◯ Inicializar el PCB en el kernel
- ◯ Inicializar el hardware para que se ejecute el incio del programa
- ◯ Informar al calendarizador que el programa se puede ejecutar
- ◯ Copiar los argumentos en la memoria

9. ¿Qué es la exclusión mutua?

- ◯ Objeto de sincronización que bloquea los hilos
- ◯ Propiedad en la que solamente un hilo puede accesar simultáneamente
- ◯ Propiedad donde los hilos se bloquean mutuamente
- ◯ Propiedad de la concurrencia para mantener los hilos excluidos

10. ¿Qué hace la función `wait` de una variable de condición?

- ◯ Duerme el hilo atómicamente mientras despierta al siguiente
- ◯ Mueve el hilo a la lista de espera atómicamente
- ◯ Atómicamente libera los seguros y mueve el hilo a la lista de espera
- ◯ Libera el seguro global, y duerme al hilo

11. ¿Qué es concurrencia?

- ◯ Realizar múltiples actividades sin la interrupción de otras
- ◯ Realizar múltiples actividades simultáneamente
- ◯ Realizar múltiples actividades una después de la otra
- ◯ Realizar una actividad sin la interrupción de otra

12. ¿Donde almacena el sistema operativo la toda la información sobre un proceso en particular?

- ◯ Stack
- ◯ *Process control block* (PCB)
- ◯ Heap
- ◯ Memoria

13. ¿Cuál de las siguientes es una característica de `fork`?

- ◯ Copiar a la memoria el código del programa del proceso padre
- ◯ Copiar el proceso padre completamente
- ◯ Crear una copia del proceso padre, pero no puede ser confiado igual que él
- ◯ Copiar el proceso padre con privilegios distintos

14. Es una señal asíncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Señal de I/O
- ◯ Trampa
- ◯ Interrupción
- ◯ Señal de software

Dado el siguiente código

```
main(int argc, char** argv){
  int child == fork();
  int x = 5;
  if (child == 0) {
    x += 5;
  } else {
    child = fork();
    x += 10;
    if (child) {
      x+=5;
    }
  }
}
```

15. ¿Cuántas copias distintas de la variable `x` existen en la ejecución del programa anterior?

- ◯ 1
- ◯ 2
- ◯ 4
- ◯ 3

16. ¿Cuál de los siguientes **no** es un valor que toma la variable `x` al término de los procesos del programa anterior?

- ◯ 20
- ◯ 5
- ◯ 15
- ◯ 10

17. Tipo de kernel en el que la mayoría de la funcionalidad de éste se encuentra dentro de él.

- ◯ Monolítico
- ◯ Híbrido
- ◯ Macrokernel
- ◯ Microkernel

18. ¿Qué hace la función `yield` de un hilo?

- ◯ Duerme al hilo por un tiempo determinado, y cambia el estado a esperando (*waiting*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a esperando (*waiting*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a terminado (*finished*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a listo (*ready*)

19. ¿Cuál es el principal problema de los hilos a nivel de usuario?

- ◯ No podemos tener varios procesos por hilo
- ◯ No podemos tener varios hilos por proceso
- ◯ El bloqueo del proceso, bloquea todos los hilos
- ◯ No podemos calendarizar hilos dentro del proceso

20. ¿Cuántos procesos son creados durante la ejecución del siguiente programa?

```
main(int argc, char** argv){
  forkthem(5);
}
void forkthem(int n){
  if (n > 0){
    fork();
    forkthem(n-1);
  }
}
```

- ◯ 16
- ◯ 4
- ◯ 5
- ◯ 30

21. ¿Cuál de los siguientes elementos **no** es compartido por los hilos?

- ◯ Variables globales
- ◯ Heap
- ◯ Stack
- ◯ Código

22. ¿Cuál de las siguientes **no** es una característica de un hilo?

- ◯ Tiene un contador de programa propio
- ◯ Comparte el código del programa
- ◯ Tiene un stack propio
- ◯ Tiene un segmento de datos propio

23. ¿Qué es el multi-hilado preventivo?

- ◯ Es cuando los hilos que se ejecutan no pueden descalendarizarse
- ◯ Es cuando los hilos entregan el procesador voluntariamente, no son interrumpidos
- ◯ Es cuando los hilos que se ejecutan pueden ser cambiados indistintamente
- ◯ Es cuando los hilos no usan el procesador hasta que se les entrega indefinidamente

24. ¿Qué es un hilo?

- ◯ Varias secuencias de ejecución que pueden ser calendarizadas dependientemente
- ◯ Varias secuencias de ejecución que pueden ser calendarizadas independientemente
- ◯ Una secuencia de ejecución que es calendarizada dependientemente
- ◯ Una secuencia de ejecución que puede ser calendarizada independientemente

25. ¿Qué es un sistema operativo?

- ◯ Es una capa de software que administra recursos y usuarios
- ◯ No es software, pero se encarga de administrar recursos y usuarios
- ◯ Es un software que administra usuarios
- ◯ Es un software que administra recursos

26. Región de la memoria reservada por el sistema operativo para mantener el estado de las variables locales durante la llamada a funciones.

- ◯ Stack
- ◯ Heap
- ◯ Swap
- ◯ RAM

udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA

27. ¿Cuál de las siguientes afirmaciones es verdadera respecto a *spin wait*?

- ◯ Es justificable en multiprocesadores
- ◯ Es justificable en monoprocesadores
- ◯ Produce *deadlocks*
- ◯ No se debe de usar nunca

28. ¿Cuál de los siguientes elementos **no** pertenece al *TCB*?

- ◯ Puntero al stack frame
- ◯ Estado del hilo
- ◯ Variables locales
- ◯ Registros salvados

29. Modo de ejecución del procesador donde no se ejecuta ninguna verificación.

- ◯ Modo de usuario
- ◯ Modo inseguro
- ◯ Modo dual
- ◯ Modo kernel

30. ¿Qué es una operación atómica?

- ◯ Es una operación de hardware que no puede ser dividida
- ◯ Es una operación del kernel que no puede ser dividida
- ◯ Es una operación indivisible que no puede ser dividida
- ◯ Es una operación que deshabilita las interrupciones

31. Región de la memoria reservada por el sistema operativo para alojar estructuras de datos que el proceso pueda necesitar.

- ◯ Heap
- ◯ Swap
- ◯ Stack
- ◯ RAM

32. ¿Cuál es el primer paso en la creación del proceso del programa `prog`?

- ◯ Crear e inicializar el PCB en el kernel
- ◯ Inicializar el hardware para que se ejecute el proceso desde el inicio
- ◯ Inicializar la memoria
- ◯ Copiar `prog` a la memoria

33. ¿Qué es una variable de condición?

- ◯ Varialbe de sincronización que implementa `yield`
- ◯ Variable de sincronización que permite la espera eficiente de un hilo
- ◯ Objeto de sincronización que implementa `yield` en modo usuario
- ◯ Variable de sincronización que permite la exclusión mutua

34. ¿Qué es un pipe?

- ◯ Es un buffer del kernel entre dos descriptores de archivos
- ◯ Es un buffer del kernel entre dos procesos
- ◯ Es un buffer de usuario entre dos procesos
- ◯ Es un buffer de usuario entre dos descriptores de archivos

35. ¿Qué son las direcciones de memoria virtuales?

- ◯ Memoria que reserva el sistema operativo para el uso de máquinas virtuales
- ◯ Capa adicional que utiliza el sistema operativo en modo kernel
- ◯ Memoria adicional que crea el sistema operativo para engañar a las aplicaciones
- ◯ Capa de indirección que le da flexibilidad al sistema operativo para administrar la memoria

36. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de inicializar el espacio de memoria?

- ◯ Informar al calendarizador que el programa se puede ejecutar
- ◯ Copiar los argumentos en la memoria
- ◯ Cargar `prog` en la memoria
- ◯ Inicializar el hardware para que se ejecute el incio del programa

37. Llamada del sistema de UNIX que le permite a las aplicaciones el comunicarse entre sí para terminarlas, suspenderlas, o resumirlas.

- ◯ `signal`
- ◯ `exec`
- ◯ `fork`
- ◯ `wait`

38. ¿Cuál es el estado de un hilo en el que está listo para ejecutarse pero no está en el procesador?

- ◯ Esperando (*wait*)
- ◯ Inicializado (*init*)
- ◯ Ejecutandose (*running*)
- ◯ Listo (*ready*)

39. Tipo de kernel en el que la funcionalidad mínima está dentro de él, y el resto se encuentra a nivel de usuario.

- ◯ Macrokernel
- ◯ Híbrido
- ◯ Monolítico
- ◯ Microkernel

40. ¿Cuál de las siguientes **no** es una característica de un descriptor de archivo de UNIX?

- ◯ Se abren antes de usar
- ◯ Se escriben a través de bytes
- ◯ Se leen a través de un buffer
- ◯ Se cierran completamente después de usar

41. Dado el siguiente código del programa `main`

```
1  char bar(){
2    int b = 2;
3    return 'a';
4  }
5  int foo(int a, char b){
6    int i=0;
7    bar();
8  }
9  main(){
10   int b=2;
11   foo();
12 }
```

Dibuje el stack completo del programa cuando se encuentra ejecutando la linea número 2. Utilice el número de línea para referirse a la dirección de cada instrucción. ◯I ◯P ◯C

El siguiente código implementa una lista circular.

```
const int MAX = 10;

class Queue{
  int items[MAX];
  int front;
  int nextEmpty;
public:
  Queue() { front = nextEmpty = 0;}
  ~Queue();
  void insert(int item);
  int remove();
}

void Queue::insert(int item){
  items[nextEmpty%MAX] = item;
  nextEmpty++;
}
int Queue::remove(){
  int item;
  item = items[front%MAX];
  front++;
  return item;
}
```

Sin embargo, las funciones `Queue::insert` y `Queue::remove` no están sincronizadas. Por lo que más elementos de los que soporta la lista pueden insertarse o removerse.

Responda lo que se le solicita a continuación. Puede declarar los `Lock` y `CV` que considere necesarios. Tenga en cuenta que la implementación debe de considerar que no puede insertar si la lista está llena, por lo que deberá de detener la función que está insertando. De manera similar, cuando elimine si no hay ningún elemento en la lista deberá esperar a que uno exista.

42. Reescriba la función `Queue::insert` para que inserte de manera segura en la lista circular. ◯I ◯P ◯C

43. Reescriba la función `Queue::remove` para que elimine de manera segura en la lista circular. ◯I ◯P ◯C

udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA

Solemne 1 **SO (CIT 2003-1)**

Instrucciones. Marque las casillas (◯) completamente sin salirse de ellas (por ejemplo ●). Responda a los siguientes cuestionamientos en las hojas que se le entregan **marcando una única opción**. Se utilizará factor de corrección 4 a 1 (las respuestas en blanco no se toman en cuenta). *Las últimas tres preguntas se evaluarán de manera Completa (4 pts.), Parcial (2 pts.), e Incompleta (0 pts.). No llene las casillas de estas últimas tres preguntas, sino que responda en el espacio indicado.*

◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0
◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1
◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2
◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3
◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4
◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5
◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6
◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7
◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8
◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9

⟵ Marque su RUT sin código verificador (el número después del guión), y escriba sus nombres y apellidos abajo.

Nombre(s) y apellido(s):

.....

.....

1. Es una señal asíncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Señal de software
- ◯ Interrupción
- ◯ Trampa
- ◯ Señal de I/O

2. Modo de ejecución del procesador en el que verifica cada instrucción antes de ejecutarla.

- ◯ Modo de usuario
- ◯ Modo seguro
- ◯ Modo kernel
- ◯ Modo dual

3. Región de la memoria reservada por el sistema operativo para mantener el estado de las variables locales durante la llamada a funciones.

- ◯ Stack
- ◯ Swap
- ◯ Heap
- ◯ RAM

4. ¿Qué es la ejecución de una aplicación con permisos restringidos?

- ◯ Aplicación
- ◯ Proceso
- ◯ Hilo
- ◯ Programa

5. ¿Donde almacena el sistema operativo la toda la información sobre un proceso en particular?

- ◯ Stack
- ◯ *Process control block* (PCB)
- ◯ Heap
- ◯ Memoria

6. ¿Cuántos procesos son creados durante la ejecución del siguiente programa?

```
main(int argc, char** argv){
  forkthem(5);
}
void forkthem(int n){
  if (n > 0){
    fork();
    forkthem(n-1);
  }
}
```

- ◯ 5
- ◯ 4
- ◯ 30
- ◯ 16

7. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de inicializar el espacio de memoria?

- ◯ Informar al calendarizador que el programa se puede ejecutar
- ◯ Inicializar el hardware para que se ejecute el incio del programa
- ◯ Copiar los argumentos en la memoria
- ◯ Cargar `prog` en la memoria

8. ¿Qué son las direcciones de memoria virtuales?

- ◯ Memoria que reserva el sistema operativo para el uso de máquinas virtuales
- ◯ Capa de indirección que le da flexibilidad al sistema operativo para administrar la memoria
- ◯ Capa adicional que utiliza el sistema operativo en modo kernel
- ◯ Memoria adicional que crea el sistema operativo para engañar a las aplicaciones

9. ¿Qué es un hilo?

- ◯ Varias secuencias de ejecución que pueden ser calendarizadas independientemente
- ◯ Una secuencia de ejecución que puede ser calendarizada independientemente
- ◯ Una secuencia de ejecución que es calendarizada dependientemente
- ◯ Varias secuencias de ejecución que pueden ser calendarizadas dependientemente

udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA

10. ¿Cuándo se da una condición de carrera?

- ◯ Varios hilos se pueden intercalar en su ejecución simultánea
- ◯ Varios hilos se ejecutan simultáneamente
- ◯ El estado de la ejecución de un programa depende del intercalado de diferentes hilos
- ◯ Existe una sección crítica

11. Región de la memoria reservada por el sistema operativo para alojar estructuras de datos que el proceso pueda necesitar.

- ◯ Swap
- ◯ RAM
- ◯ Stack
- ◯ Heap

12. ¿Qué hace la función `wait` de una variable de condición?

- ◯ Atómicamente libera los seguros y mueve el hilo a la lista de espera
- ◯ Duerme el hilo atómicamente mientras despierta al siguiente
- ◯ Libera el seguro global, y duerme al hilo
- ◯ Mueve el hilo a la lista de espera atómicamente

13. ¿Qué es una operación atómica?

- ◯ Es una operación indivisible que no puede ser dividida
- ◯ Es una operación que deshabilita las interrupciones
- ◯ Es una operación de hardware que no puede ser dividida
- ◯ Es una operación del kernel que no puede ser dividida

14. ¿Cuál de los siguientes eventos **no** genera una transición de modo kernel a usuario?

- ◯ Cambiar a un proceso diferente
- ◯ Llamada I/O a un dispositivo
- ◯ Continuar después de una interrupción
- ◯ Creación de nuevo proceso

15. ¿Qué característica del sistema operativo permite que opere independientemente del hardware en la computadora?

- ◯ Portabilidad
- ◯ Compartimiento de recursos
- ◯ Ilusionista
- ◯ Virtualización

16. Es una señal síncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Señal de software
- ◯ Señal de I/O
- ◯ Interrupción
- ◯ Trampa

17. ¿Qué es la exclusión mutua?

- ◯ Propiedad de la concurrencia para mantener los hilos excluidos
- ◯ Objeto de sincronización que bloquea los hilos
- ◯ Propiedad en la que solamente un hilo puede accesar simultáneamente
- ◯ Propiedad donde los hilos se bloquean mutuamente

18. Tipo de kernel en el que la funcionalidad mínima está dentro de él, y el resto se encuentra a nivel de usuario.

- ◯ Híbrido
- ◯ Macrokernel
- ◯ Monolítico
- ◯ Microkernel

Dado el siguiente código

```
main(int argc, char** argv){
  int child == fork();
  int x = 5;
  if (child == 0) {
    x += 5;
  } else {
    child = fork();
    x += 10;
    if (child) {
      x+=5;
    }
  }
}
```

19. ¿Cuántas copias distintas de la variable `x` existen en la ejecución del programa anterior?

- ◯ 4
- ◯ 2
- ◯ 3
- ◯ 1

20. ¿Cuál de los siguientes **no** es un valor que toma la variable `x` al término de los procesos del programa anterior?

- ◯ 5
- ◯ 20
- ◯ 15
- ◯ 10

21. ¿Cuál de las siguientes es una característica de `fork`?

- ◯ Copiar a la memoria el código del programa del proceso padre
- ◯ Crear una copia del proceso padre, pero no puede ser confiado igual que él
- ◯ Copiar el proceso padre completamente
- ◯ Copiar el proceso padre con privilegios distintos

22. ¿Qué es una llamada de sistema?

- ◯ Llamada que hace el kernel para realizar una instrucción
- ◯ Funciones de hardware que llama el usuario
- ◯ Código del kernel que ejecuta código de usuario
- ◯ Funciones del kernel que permiten al usuario accesar a recursos restringidos

23. ¿Qué es el multi-hilado preventivo?

- ◯ Es cuando los hilos que se ejecutan pueden ser cambiados indistintamente
- ◯ Es cuando los hilos que se ejecutan no pueden descalendarizarse
- ◯ Es cuando los hilos no usan el procesador hasta que se les entrega indefinidamente
- ◯ Es cuando los hilos entregan el procesador voluntariamente, no son interrumpidos

24. ¿Qué es un sistema operativo?

- ◯ Es un software que administra usuarios
- ◯ No es software, pero se encarga de administrar recursos y usuarios
- ◯ Es una capa de software que administra recursos y usuarios
- ◯ Es un software que administra recursos

25. ¿Qué es concurrencia?

- ◯ Realizar múltiples actividades simultáneamente
- ◯ Realizar una actividad sin la interrupción de otra
- ◯ Realizar múltiples actividades una después de la otra
- ◯ Realizar múltiples actividades sin la interrupción de otras

26. ¿Qué hace la función `yield` de un hilo?

- ◯ Duerme al hilo por un tiempo determinado, y cambia el estado a esperando (*waiting*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a listo (*ready*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a terminado (*finished*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a esperando (*waiting*)

27. ¿Cuál de las siguientes **no** es una característica de un hilo?

- ◯ Tiene un contador de programa propio
- ◯ Tiene un segmento de datos propio
- ◯ Tiene un stack propio
- ◯ Comparte el código del programa

28. ¿Cuál es el principal problema de los hilos a nivel de usuario?

- ◯ No podemos tener varios hilos por proceso
- ◯ El bloqueo del proceso, bloquea todos los hilos
- ◯ No podemos calendarizar hilos dentro del proceso
- ◯ No podemos tener varios procesos por hilo

29. ¿Cuál de las siguientes **no** es una característica de un descriptor de archivo de UNIX?

- ◯ Se abren antes de usar
- ◯ Se escriben a través de bytes
- ◯ Se cierran completamente después de usar
- ◯ Se leen a través de un buffer

30. Tipo de kernel en el que la mayoría de la funcionalidad de éste se encuentra dentro de él.

- ◯ Híbrido
- ◯ Macrokernel
- ◯ Microkernel
- ◯ Monolítico

31. ¿Qué es una variable de condición?

- ◯ Objeto de sincronización que implementa `yield` en modo usuario
- ◯ Variable de sincronización que permite la espera eficiente de un hilo
- ◯ Varialbe de sincronización que implementa `yield`
- ◯ Variable de sincronización que permite la exclusión mutua

32. ¿Cuál es el estado de un hilo en el que está listo para ejecutarse pero no está en el procesador?

- ◯ Esperando (*wait*)
- ◯ Ejecutandose (*running*)
- ◯ Listo (*ready*)
- ◯ Inicializado (*init*)

33. ¿Cuál es el primer paso en la creación del proceso del programa `prog`?

- ◯ Inicializar el hardware para que se ejecute el proceso desde el inicio
- ◯ Inicializar la memoria
- ◯ Copiar `prog` a la memoria
- ◯ Crear e inicializar el PCB en el kernel

34. Modo de ejecución del procesador donde no se ejecuta ninguna verificación.

- ◯ Modo de usuario
- ◯ Modo kernel
- ◯ Modo inseguro
- ◯ Modo dual

35. Llamada del sistema de UNIX que le permite a las aplicaciones el comunicarse entre sí para terminarlas, suspenderlas, o resumirlas.

- ◯ `signal`
- ◯ `wait`
- ◯ `fork`
- ◯ `exec`

36. ¿Cuál de los siguientes elementos **no** es compartido por los hilos?

- ◯ Código
- ◯ Variables globales
- ◯ Heap
- ◯ Stack

37. ¿Cuál de las siguientes afirmaciones es verdadera respecto a *spin wait*?

- ◯ Es justificable en monoprocesadores
- ◯ No se debe de usar nunca
- ◯ Produce *deadlocks*
- ◯ Es justificable en multiprocesadores

38. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de cargar `prog` en la memoria?

- ◯ Inicializar el hardware para que se ejecute el incio del programa
- ◯ Informar al calendarizador que el programa se puede ejecutar
- ◯ Inicializar el PCB en el kernel
- ◯ Copiar los argumentos en la memoria

39. ¿Qué es un pipe?

- ◯ Es un buffer de usuario entre dos descriptores de archivos
- ◯ Es un buffer de usuario entre dos procesos
- ◯ Es un buffer del kernel entre dos procesos
- ◯ Es un buffer del kernel entre dos descriptores de archivos

40. ¿Cuál de los siguientes elementos **no** pertenece al *TCB*?

- ◯ Puntero al stack frame
- ◯ Registros salvados
- ◯ Estado del hilo
- ◯ Variables locales

41. Dado el siguiente código del programa `main`

```
1  char bar(){
2    int b = 2;
3    return 'a';
4  }
5  int foo(int a, char b){
6    int i=0;
7    bar();
8  }
9  main(){
10   int b=2;
11   foo();
12 }
```

Dibuje el stack completo del programa cuando se encuentra ejecutando la linea número 2. Utilice el número de línea para referirse a la dirección de cada instrucción. ◯I ◯P ◯C

El siguiente código implementa una lista circular.

```
const int MAX = 10;

class Queue{
  int items[MAX];
  int front;
  int nextEmpty;
public:
  Queue() { front = nextEmpty = 0;}
  ~Queue();
  void insert(int item);
  int remove();
}

void Queue::insert(int item){
  items[nextEmpty%MAX] = item;
  nextEmpty++;
}
int Queue::remove(){
  int item;
  item = items[front%MAX];
  front++;
  return item;
}
```

Sin embargo, las funciones `Queue::insert` y `Queue::remove` no están sincronizadas. Por lo que más elementos de los que soporta la lista pueden insertarse o removerse.

Responda lo que se le solicita a continuación. Puede declarar los `Lock` y `CV` que considere necesarios. Tenga en cuenta que la implementación debe de considerar que no puede insertar si la lista está llena, por lo que deberá de detener la función que está insertando. De manera similar, cuando elimine si no hay ningún elemento en la lista deberá esperar a que uno exista.

42. Reescriba la función `Queue::insert` para que inserte de manera segura en la lista circular. ◯I ◯P ◯C

43. Reescriba la función `Queue::remove` para que elimine de manera segura en la lista circular. ◯I ◯P ◯C

udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA

Solemne 1 **SO (CIT 2003-1)**

Instrucciones. Marque las casillas (◯) completamente sin salirse de ellas (por ejemplo ●). Responda a los siguientes cuestionamientos en las hojas que se le entregan **marcando una única opción**. Se utilizará factor de corrección 4 a 1 (las respuestas en blanco no se toman en cuenta). *Las últimas tres preguntas se evaluarán de manera Completa (4 pts.), Parcial (2 pts.), e Incompleta (0 pts.). No llene las casillas de estas últimas tres preguntas, sino que responda en el espacio indicado.*

◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0
◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1
◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2
◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3
◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4
◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5
◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6
◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7
◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8
◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9

⟵ Marque su RUT sin código verificador (el número después del guión), y escriba sus nombres y apellidos abajo.

Nombre(s) y apellido(s):

.....

.....

1. ¿Cuál de las siguientes es una característica de `fork`?

- ◯ Crear una copia del proceso padre, pero no puede ser confiado igual que él
- ◯ Copiar a la memoria el código del programa del proceso padre
- ◯ Copiar el proceso padre completamente
- ◯ Copiar el proceso padre con privilegios distintos

2. ¿Cuál de las siguientes **no** es una característica de un hilo?

- ◯ Tiene un segmento de datos propio
- ◯ Tiene un contador de programa propio
- ◯ Tiene un stack propio
- ◯ Comparte el código del programa

3. ¿Cuál es el estado de un hilo en el que está listo para ejecutarse pero no está en el procesador?

- ◯ Listo (*ready*)
- ◯ Esperando (*wait*)
- ◯ Ejecutandose (*running*)
- ◯ Inicializado (*init*)

4. ¿Cuál de las siguientes afirmaciones es verdadera respecto a *spin wait*?

- ◯ Es justificable en monoprocesadores
- ◯ Produce *deadlocks*
- ◯ No se debe de usar nunca
- ◯ Es justificable en multiprocesadores

5. ¿Cuántos procesos son creados durante la ejecución del siguiente programa?

```
main(int argc, char** argv){
  forkthem(5);
}
void forkthem(int n){
  if (n > 0){
    fork();
    forkthem(n-1);
  }
}
```

- ◯ 5
- ◯ 4
- ◯ 16
- ◯ 30

6. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de cargar `prog` en la memoria?

- ◯ Inicializar el PCB en el kernel
- ◯ Inicializar el hardware para que se ejecute el incio del programa
- ◯ Copiar los argumentos en la memoria
- ◯ Informar al calendarizador que el programa se puede ejecutar

7. ¿Qué es la ejecución de una aplicación con permisos restringidos?

- ◯ Programa
- ◯ Aplicación
- ◯ Hilo
- ◯ Proceso

8. ¿Cuál es el principal problema de los hilos a nivel de usuario?

- ◯ No podemos tener varios hilos por proceso
- ◯ El bloqueo del proceso, bloquea todos los hilos
- ◯ No podemos calendarizar hilos dentro del proceso
- ◯ No podemos tener varios procesos por hilo

9. Modo de ejecución del procesador donde no se ejecuta ninguna verificación.

- ◯ Modo inseguro
- ◯ Modo dual
- ◯ Modo kernel
- ◯ Modo de usuario

10. ¿Donde almacena el sistema operativo la toda la información sobre un proceso en particular?

- ◯ Heap
- ◯ Stack
- ◯ *Process control block* (PCB)
- ◯ Memoria

11. Es una señal síncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Interrupción
- ◯ Señal de software
- ◯ Señal de I/O
- ◯ Trampa

12. ¿Cuándo se da una condición de carrera?

- ◯ Varios hilos se pueden intercalar en su ejecución simultánea
- ◯ Existe una sección crítica
- ◯ Varios hilos se ejecutan simultáneamente
- ◯ El estado de la ejecución de un programa depende del intercalado de diferentes hilos

13. Tipo de kernel en el que la mayoría de la funcionalidad de éste se encuentra dentro de él.

- ◯ Macrokernel
- ◯ Híbrido
- ◯ Monolítico
- ◯ Microkernel

14. Llamada del sistema de UNIX que le permite a las aplicaciones el comunicarse entre sí para terminarlas, suspenderlas, o resumirlas.

- ◯ `fork`
- ◯ `signal`
- ◯ `wait`
- ◯ `exec`

15. Región de la memoria reservada por el sistema operativo para alojar estructuras de datos que el proceso pueda necesitar.

- ◯ Swap
- ◯ Stack
- ◯ RAM
- ◯ Heap

16. ¿Cuál es el primer paso en la creación del proceso del programa `prog`?

- ◯ Crear e inicializar el PCB en el kernel
- ◯ Copiar `prog` a la memoria
- ◯ Inicializar el hardware para que se ejecute el proceso desde el inicio
- ◯ Inicializar la memoria

17. ¿Qué característica del sistema operativo permite que opere independientemente del hardware en la computadora?

- ◯ Portabilidad
- ◯ Ilusionista
- ◯ Virtualización
- ◯ Compartimiento de recursos

18. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de inicializar el espacio de memoria?

- ◯ Copiar los argumentos en la memoria
- ◯ Inicializar el hardware para que se ejecute el incio del programa
- ◯ Informar al calendarizador que el programa se puede ejecutar
- ◯ Cargar `prog` en la memoria

19. Región de la memoria reservada por el sistema operativo para mantener el estado de las variables locales durante la llamada a funciones.

- ◯ Heap
- ◯ RAM
- ◯ Swap
- ◯ Stack

20. ¿Qué es concurrencia?

- ◯ Realizar múltiples actividades sin la interrupción de otras
- ◯ Realizar múltiples actividades una después de la otra
- ◯ Realizar una actividad sin la interrupción de otra
- ◯ Realizar múltiples actividades simultáneamente

Dado el siguiente código

```
main(int argc, char** argv){
  int child == fork();
  int x = 5;
  if (child == 0) {
    x += 5;
  } else {
    child = fork();
    x += 10;
    if (child) {
      x+=5;
    }
  }
}
```

21. ¿Cuál de los siguientes **no** es un valor que toma la variable `x` al término de los procesos del programa anterior?

- ◯ 5
- ◯ 20
- ◯ 15
- ◯ 10

22. ¿Cuántas copias distintas de la variable `x` existen en la ejecución del programa anterior?

- ◯ 1
- ◯ 3
- ◯ 4
- ◯ 2

23. ¿Qué es una operación atómica?

- ◯ Es una operación que deshabilita las interrupciones
- ◯ Es una operación indivisible que no puede ser dividida
- ◯ Es una operación del kernel que no puede ser dividida
- ◯ Es una operación de hardware que no puede ser dividida

24. ¿Qué hace la función `wait` de una variable de condición?

- ◯ Atómicamente libera los seguros y mueve el hilo a la lista de espera
- ◯ Duerme el hilo atómicamente mientras despierta al siguiente
- ◯ Libera el seguro global, y duerme al hilo
- ◯ Mueve el hilo a la lista de espera atómicamente

25. ¿Qué es una llamada de sistema?

- ◯ Llamada que hace el kernel para realizar una instrucción
- ◯ Funciones de hardware que llama el usuario
- ◯ Funciones del kernel que permiten al usuario accesar a recursos restringidos
- ◯ Código del kernel que ejecuta código de usuario

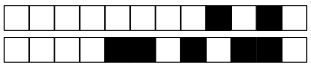
26. ¿Qué son las direcciones de memoria virtuales?

- ◯ Capa de indirección que le da flexibilidad al sistema operativo para administrar la memoria
- ◯ Capa adicional que utiliza el sistema operativo en modo kernel
- ◯ Memoria que reserva el sistema operativo para el uso de máquinas virtuales
- ◯ Memoria adicional que crea el sistema operativo para engañar a las aplicaciones

27. ¿Cuál de las siguientes **no** es una característica de un descriptor de archivo de UNIX?

- ◯ Se abren antes de usar
- ◯ Se escriben a través de bytes
- ◯ Se cierran completamente después de usar
- ◯ Se leen a través de un buffer

udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA



28. Modo de ejecución del procesador en el que verifica cada instrucción antes de ejecutarla.

- ◯ Modo seguro
- ◯ Modo kernel
- ◯ Modo de usuario
- ◯ Modo dual

29. ¿Cuál de los siguientes eventos **no** genera una transición de modo kernel a usuario?

- ◯ Llamada I/O a un dispositivo
- ◯ Creación de nuevo proceso
- ◯ Cambiar a un proceso diferente
- ◯ Continuar después de una interrupción

30. Es una señal asíncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Señal de I/O
- ◯ Trampa
- ◯ Señal de software
- ◯ Interrupción

31. ¿Qué es la exclusión mutua?

- ◯ Propiedad de la concurrencia para mantener los hilos excluidos
- ◯ Propiedad en la que solamente un hilo puede accesar simultáneamente
- ◯ Propiedad donde los hilos se bloquean mutuamente
- ◯ Objeto de sincronización que bloquea los hilos

32. ¿Qué es una variable de condición?

- ◯ Objeto de sincronización que implementa `yield` en modo usuario
- ◯ Varialbe de sincronización que implementa `yield`
- ◯ Variable de sincronización que permite la exclusión mutua
- ◯ Variable de sincronización que permite la espera eficiente de un hilo

33. ¿Cuál de los siguientes elementos **no** es compartido por los hilos?

- ◯ Código
- ◯ Stack
- ◯ Variables globales
- ◯ Heap

34. ¿Qué es un sistema operativo?

- ◯ Es un software que administra recursos
- ◯ Es un software que administra usuarios
- ◯ Es una capa de software que administra recursos y usuarios
- ◯ No es software, pero se encarga de administrar recursos y usuarios

35. ¿Qué es un pipe?

- ◯ Es un buffer del kernel entre dos procesos
- ◯ Es un buffer de usuario entre dos descriptores de archivos
- ◯ Es un buffer del kernel entre dos descriptores de archivos
- ◯ Es un buffer de usuario entre dos procesos

36. ¿Qué hace la función `yield` de un hilo?

- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a terminado (*finished*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a esperando (*waiting*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a listo (*ready*)
- ◯ Duerme al hilo por un tiempo determinado, y cambia el estado a esperando (*waiting*)

37. ¿Cuál de los siguientes elementos **no** pertenece al *TCB*?

- ◯ Estado del hilo
- ◯ Puntero al stack frame
- ◯ Registros salvados
- ◯ Variables locales

38. Tipo de kernel en el que la funcionalidad mínima está dentro de él, y el resto se encuentra a nivel de usuario.

- ◯ Híbrido
- ◯ Microkernel
- ◯ Monolítico
- ◯ Macrokernel

39. ¿Qué es un hilo?

- ◯ Una secuencia de ejecución que es calendarizada dependientemente
- ◯ Una secuencia de ejecución que puede ser calendarizada independientemente
- ◯ Varias secuencias de ejecución que pueden ser calendarizadas independientemente
- ◯ Varias secuencias de ejecución que pueden ser calendarizadas dependientemente

40. ¿Qué es el multi-hilado preventivo?

- ◯ Es cuando los hilos que se ejecutan pueden ser cambiados indistintamente
- ◯ Es cuando los hilos no usan el procesador hasta que se les entrega indefinidamente
- ◯ Es cuando los hilos entregan el procesador voluntariamente, no son interrumpidos
- ◯ Es cuando los hilos que se ejecutan no pueden descalendarizarse

41. Dado el siguiente código del programa `main`

```
1  char bar(){
2    int b = 2;
3    return 'a';
4  }
5  int foo(int a, char b){
6    int i=0;
7    bar();
8  }
9  main(){
10   int b=2;
11   foo();
12 }
```

Dibuje el stack completo del programa cuando se encuentra ejecutando la linea número 2. Utilice el número de línea para referirse a la dirección de cada instrucción. ◯I ◯P ◯C

El siguiente código implementa una lista circular.

```
const int MAX = 10;

class Queue{
  int items[MAX];
  int front;
  int nextEmpty;
public:
  Queue() { front = nextEmpty = 0;}
  ~Queue();
  void insert(int item);
  int remove();
}

void Queue::insert(int item){
  items[nextEmpty%MAX] = item;
  nextEmpty++;
}
int Queue::remove(){
  int item;
  item = items[front%MAX];
  front++;
  return item;
}
```

Sin embargo, las funciones `Queue::insert` y `Queue::remove` no están sincronizadas. Por lo que más elementos de los que soporta la lista pueden insertarse o removerse.

Responda lo que se le solicita a continuación. Puede declarar los `Lock` y `CV` que considere necesarios. Tenga en cuenta que la implementación debe de considerar que no puede insertar si la lista está llena, por lo que deberá de detener la función que está insertando. De manera similar, cuando elimine si no hay ningún elemento en la lista deberá esperar a que uno exista.

42. Reescriba la función `Queue::insert` para que inserte de manera segura en la lista circular. ◯I ◯P ◯C

43. Reescriba la función `Queue::remove` para que elimine de manera segura en la lista circular. ◯I ◯P ◯C

udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA

Solemne 1 **SO (CIT 2003-1)**

Instrucciones. Marque las casillas (◯) completamente sin salirse de ellas (por ejemplo ●). Responda a los siguientes cuestionamientos en las hojas que se le entregan **marcando una única opción**. Se utilizará factor de corrección 4 a 1 (las respuestas en blanco no se toman en cuenta). *Las últimas tres preguntas se evaluarán de manera Completa (4 pts.), Parcial (2 pts.), e Incompleta (0 pts.). No llene las casillas de estas últimas tres preguntas, sino que responda en el espacio indicado.*

◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0
◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1
◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2
◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3
◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4
◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5
◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6
◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7
◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8
◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9

⟵ Marque su RUT sin código verificador (el número después del guión), y escriba sus nombres y apellidos abajo.

Nombre(s) y apellido(s):

.....

.....

1. ¿Cuál es el estado de un hilo en el que está listo para ejecutarse pero no está en el procesador?

- ◯ Ejecutandose (*running*)
- ◯ Esperando (*wait*)
- ◯ Listo (*ready*)
- ◯ Inicializado (*init*)

2. ¿Qué es un hilo?

- ◯ Varias secuencias de ejecución que pueden ser calendarizadas independientemente
- ◯ Varias secuencias de ejecución que pueden ser calendarizadas dependientemente
- ◯ Una secuencia de ejecución que puede ser calendarizada independientemente
- ◯ Una secuencia de ejecución que es calendarizada dependientemente

3. ¿Cuál de las siguientes **no** es una característica de un descriptor de archivo de UNIX?

- ◯ Se abren antes de usar
- ◯ Se cierran completamente después de usar
- ◯ Se escriben a través de bytes
- ◯ Se leen a través de un buffer

4. Es una señal asíncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Señal de software
- ◯ Interrupción
- ◯ Señal de I/O
- ◯ Trampa

5. Modo de ejecución del procesador donde no se ejecuta ninguna verificación.

- ◯ Modo de usuario
- ◯ Modo kernel
- ◯ Modo inseguro
- ◯ Modo dual

6. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de inicializar el espacio de memoria?

- ◯ Copiar los argumentos en la memoria
- ◯ Inicializar el hardware para que se ejecute el incio del programa
- ◯ Cargar `prog` en la memoria
- ◯ Informar al calendarizador que el programa se puede ejecutar

7. ¿Cuál de las siguientes afirmaciones es verdadera respecto a *spin wait*?

- ◯ Es justificable en monoprocesadores
- ◯ Es justificable en multiprocesadores
- ◯ Produce *deadlocks*
- ◯ No se debe de usar nunca

8. ¿Qué es el multi-hilado preventivo?

- ◯ Es cuando los hilos no usan el procesador hasta que se les entrega indefinidamente
- ◯ Es cuando los hilos entregan el procesador voluntariamente, no son interrumpidos
- ◯ Es cuando los hilos que se ejecutan pueden ser cambiados indistintamente
- ◯ Es cuando los hilos que se ejecutan no pueden descalendarizarse

9. ¿Qué característica del sistema operativo permite que opere independientemente del hardware en la computadora?

- ◯ Virtualización
- ◯ Portabilidad
- ◯ Compartimiento de recursos
- ◯ Ilusionista

10. ¿Qué es la ejecución de una aplicación con permisos restringidos?

- ◯ Aplicación
- ◯ Proceso
- ◯ Hilo
- ◯ Programa

11. ¿Cuál de los siguientes elementos **no** es compartido por los hilos?

- ◯ Código
- ◯ Heap
- ◯ Variables globales
- ◯ Stack

12. Región de la memoria reservada por el sistema operativo para mantener el estado de las variables locales durante la llamada a funciones.

- ◯ Swap
- ◯ Heap
- ◯ Stack
- ◯ RAM

13. ¿Cuál es el primer paso en la creación del proceso del programa `prog`?

- ◯ Crear e inicializar el PCB en el kernel
- ◯ Copiar `prog` a la memoria
- ◯ Inicializar la memoria
- ◯ Inicializar el hardware para que se ejecute el proceso desde el inicio

14. ¿Cuántos procesos son creados durante la ejecución del siguiente programa?

```
main(int argc, char** argv){
  forkthem(5);
}
void forkthem(int n){
  if (n > 0){
    fork();
    forkthem(n-1);
  }
}
```

- ◯ 4
- ◯ 30
- ◯ 5
- ◯ 16

15. ¿Qué es concurrencia?

- ◯ Realizar múltiples actividades simultáneamente
- ◯ Realizar una actividad sin la interrupción de otra
- ◯ Realizar múltiples actividades sin la interrupción de otras
- ◯ Realizar múltiples actividades una después de la otra

16. ¿Donde almacena el sistema operativo la toda la información sobre un proceso en particular?

- ◯ Memoria
- ◯ Stack
- ◯ *Process control block* (PCB)
- ◯ Heap

17. Tipo de kernel en el que la mayoría de la funcionalidad de éste se encuentra dentro de él.

- ◯ Monolítico
- ◯ Híbrido
- ◯ Macrokernel
- ◯ Microkernel

18. Es una señal síncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Interrupción
- ◯ Señal de I/O
- ◯ Señal de software
- ◯ Trampa

19. ¿Cuál de las siguientes es una característica de `fork`?

- ◯ Crear una copia del proceso padre, pero no puede ser confiado igual que él
- ◯ Copiar el proceso padre completamente
- ◯ Copiar el proceso padre con privilegios distintos
- ◯ Copiar a la memoria el código del programa del proceso padre

20. ¿Cuál es el principal problema de los hilos a nivel de usuario?

- ◯ No podemos calendarizar hilos dentro del proceso
- ◯ El bloqueo del proceso, bloquea todos los hilos
- ◯ No podemos tener varios hilos por proceso
- ◯ No podemos tener varios procesos por hilo

21. ¿Qué es la exclusión mutua?

- ◯ Propiedad en la que solamente un hilo puede acesar simultáneamente
- ◯ Propiedad donde los hilos se bloquean mutuamente
- ◯ Propiedad de la concurrencia para mantener los hilos excluidos
- ◯ Objeto de sincronización que bloquea los hilos

22. ¿Cuál de las siguientes **no** es una característica de un hilo?

- ◯ Tiene un stack propio
- ◯ Tiene un contador de programa propio
- ◯ Comparte el código del programa
- ◯ Tiene un segmento de datos propio

23. ¿Cuándo se da una condición de carrera?

- ◯ Existe una sección crítica
- ◯ Varios hilos se ejecutan simultáneamente
- ◯ Varios hilos se pueden intercalar en su ejecución simultánea
- ◯ El estado de la ejecución de un programa depende del intercalado de diferentes hilos

Dado el siguiente código

```
main(int argc, char** argv){
  int child == fork();
  int x = 5;
  if (child == 0) {
    x += 5;
  } else {
    child = fork();
    x += 10;
    if (child) {
      x+=5;
    }
  }
}
```

24. ¿Cuántas copias distintas de la variable `x` existen en la ejecución del programa anterior?

- ◯ 4
- ◯ 2
- ◯ 1
- ◯ 3

25. ¿Cuál de los siguientes **no** es un valor que toma la variable `x` al término de los procesos del programa anterior?

- ◯ 10
- ◯ 20
- ◯ 15
- ◯ 5

26. ¿Qué es una variable de condición?

- ◯ Variable de sincronización que permite la espera eficiente de un hilo
- ◯ Varialbe de sincronización que implementa `yield`
- ◯ Variable de sincronización que permite la exclusión mutua
- ◯ Objeto de sincronización que implementa `yield` en modo usuario

27. ¿Qué hace la función `wait` de una variable de condición?

- ◯ Mueve el hilo a la lista de espera atómicamente
- ◯ Libera el seguro global, y duerme al hilo
- ◯ Duerme el hilo atómicamente mientras despierta al siguiente
- ◯ Atómicamente libera los seguros y mueve el hilo a la lista de espera

28. Región de la memoria reservada por el sistema operativo para alojar estructuras de datos que el proceso pueda necesitar.

- ◯ Swap
- ◯ Stack
- ◯ Heap
- ◯ RAM

29. ¿Cuál de los siguientes eventos **no** genera una transición de modo kernel a usuario?

- ◯ Llamada I/O a un dispositivo
- ◯ Continuar después de una interrupción
- ◯ Cambiar a un proceso diferente
- ◯ Creación de nuevo proceso

30. Modo de ejecución del procesador en el que verifica cada instrucción antes de ejecutarla.

- ◯ Modo seguro
- ◯ Modo dual
- ◯ Modo de usuario
- ◯ Modo kernel

31. ¿Qué es una operación atómica?

- ◯ Es una operación del kernel que no puede ser dividida
- ◯ Es una operación que deshabilita las interrupciones
- ◯ Es una operación de hardware que no puede ser dividida
- ◯ Es una operación indivisible que no puede ser dividida

32. ¿Qué es una llamada de sistema?

- ◯ Funciones del kernel que permiten al usuario accesar a recursos restringidos
- ◯ Llamada que hace el kernel para realizar una instrucción
- ◯ Código del kernel que ejecuta código de usuario
- ◯ Funciones de hardware que llama el usuario

33. ¿Qué es un sistema operativo?

- ◯ Es un software que administra usuarios
- ◯ No es software, pero se encarga de administrar recursos y usuarios
- ◯ Es una capa de software que administra recursos y usuarios
- ◯ Es un software que administra recursos

34. ¿Qué es un pipe?

- ◯ Es un buffer de usuario entre dos procesos
- ◯ Es un buffer del kernel entre dos procesos
- ◯ Es un buffer de usuario entre dos descriptores de archivos
- ◯ Es un buffer del kernel entre dos descriptores de archivos

35. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de cargar `prog` en la memoria?

- ◯ Inicializar el hardware para que se ejecute el incio del programa
- ◯ Inicializar el PCB en el kernel
- ◯ Copiar los argumentos en la memoria
- ◯ Informar al calendarizador que el programa se puede ejecutar

36. Tipo de kernel en el que la funcionalidad mínima está dentro de él, y el resto se encuentra a nivel de usuario.

- ◯ Híbrido
- ◯ Microkernel
- ◯ Monolítico
- ◯ Macrokernel

37. Llamada del sistema de UNIX que le permite a las aplicaciones el comunicarse entre sí para terminarlas, suspenderlas, o resumirlas.

- ◯ `wait`
- ◯ `fork`
- ◯ `signal`
- ◯ `exec`

38. ¿Qué hace la función `yield` de un hilo?

- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a esperando (*waiting*)
- ◯ Duerme al hilo por un tiempo determinado, y cambia el estado a esperando (*waiting*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a terminado (*finished*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a listo (*ready*)

39. ¿Cuál de los siguientes elementos **no** pertenece al *TCB*?

- ◯ Variables locales
- ◯ Estado del hilo
- ◯ Registros salvados
- ◯ Puntero al stack frame

40. ¿Qué son las direcciones de memoria virtuales?

- ◯ Capa adicional que utiliza el sistema operativo en modo kernel
- ◯ Memoria adicional que crea el sistema operativo para engañar a las aplicaciones
- ◯ Memoria que reserva el sistema operativo para el uso de máquinas virtuales
- ◯ Capa de indirección que le da flexibilidad al sistema operativo para administrar la memoria

41. Dado el siguiente código del programa `main`

```
1  char bar(){
2    int b = 2;
3    return 'a';
4  }
5  int foo(int a, char b){
6    int i=0;
7    bar();
8  }
9  main(){
10   int b=2;
11   foo();
12 }
```

Dibuje el stack completo del programa cuando se encuentra ejecutando la linea número 2. Utilice el número de línea para referirse a la dirección de cada instrucción. ◯I ◯P ◯C

El siguiente código implementa una lista circular.

```
const int MAX = 10;

class Queue{
  int items[MAX];
  int front;
  int nextEmpty;
public:
  Queue() { front = nextEmpty = 0;}
  ~Queue();
  void insert(int item);
  int remove();
}

void Queue::insert(int item){
  items[nextEmpty%MAX] = item;
  nextEmpty++;
}
int Queue::remove(){
  int item;
  item = items[front%MAX];
  front++;
  return item;
}
```

Sin embargo, las funciones `Queue::insert` y `Queue::remove` no están sincronizadas. Por lo que más elementos de los que soporta la lista pueden insertarse o removerse.

Responda lo que se le solicita a continuación. Puede declarar los `Lock` y `CV` que considere necesarios. Tenga en cuenta que la implementación debe de considerar que no puede insertar si la lista está llena, por lo que deberá de detener la función que está insertando. De manera similar, cuando elimine si no hay ningún elemento en la lista deberá esperar a que uno exista.

42. Reescriba la función `Queue::insert` para que inserte de manera segura en la lista circular. ◯I ◯P ◯C

43. Reescriba la función `Queue::remove` para que elimine de manera segura en la lista circular. ◯I ◯P ◯C

udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA

Solemne 1 **SO (CIT 2003-1)**

Instrucciones. Marque las casillas (◯) completamente sin salirse de ellas (por ejemplo ●). Responda a los siguientes cuestionamientos en las hojas que se le entregan **marcando una única opción**. Se utilizará factor de corrección 4 a 1 (las respuestas en blanco no se toman en cuenta). *Las últimas tres preguntas se evaluarán de manera Completa (4 pts.), Parcial (2 pts.), e Incompleta (0 pts.). No llene las casillas de estas últimas tres preguntas, sino que responda en el espacio indicado.*

◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0
◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1
◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2
◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3
◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4
◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5
◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6
◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7
◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8
◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9

⟵ Marque su RUT sin código verificador (el número después del guión), y escriba sus nombres y apellidos abajo.

Nombre(s) y apellido(s):

.....

.....

1. ¿Qué es la ejecución de una aplicación con permisos restringidos?

- ◯ Aplicación
- ◯ Hilo
- ◯ Proceso
- ◯ Programa

2. ¿Cuál de las siguientes **no** es una característica de un descriptor de archivo de UNIX?

- ◯ Se escriben a través de bytes
- ◯ Se leen a través de un buffer
- ◯ Se cierran completamente después de usar
- ◯ Se abren antes de usar

3. Tipo de kernel en el que la mayoría de la funcionalidad de éste se encuentra dentro de él.

- ◯ Monolítico
- ◯ Macrokernel
- ◯ Microkernel
- ◯ Híbrido

4. ¿Qué es la exclusión mutua?

- ◯ Propiedad en la que solamente un hilo puede acesar simultáneamente
- ◯ Propiedad donde los hilos se bloquean mutuamente
- ◯ Objeto de sincronización que bloquea los hilos
- ◯ Propiedad de la concurrencia para mantener los hilos excluidos

5. ¿Cuál es el estado de un hilo en el que está listo para ejecutarse pero no está en el procesador?

- ◯ Inicializado (*init*)
- ◯ Ejecutandose (*running*)
- ◯ Esperando (*wait*)
- ◯ Listo (*ready*)

6. Modo de ejecución del procesador en el que verifica cada instrucción antes de ejecutarla.

- ◯ Modo dual
- ◯ Modo kernel
- ◯ Modo de usuario
- ◯ Modo seguro

7. ¿Cuándo se da una condición de carrera?

- ◯ Existe una sección crítica
- ◯ Varios hilos se pueden intercalar en su ejecución simultánea
- ◯ Varios hilos se ejecutan simultáneamente
- ◯ El estado de la ejecución de un programa depende del intercalado de diferentes hilos

8. ¿Donde almacena el sistema operativo la toda la información sobre un proceso en particular?

- ◯ Stack
- ◯ Memoria
- ◯ Heap
- ◯ *Process control block* (PCB)

9. Llamada del sistema de UNIX que le permite a las aplicaciones el comunicarse entre sí para terminarlas, suspenderlas, o resumirlas.

- ◯ `exec`
- ◯ `wait`
- ◯ `signal`
- ◯ `fork`

10. ¿Cuál de las siguientes afirmaciones es verdadera respecto a *spin wait*?

- ◯ Es justificable en monoprocesadores
- ◯ Produce *deadlocks*
- ◯ Es justificable en multiprocesadores
- ◯ No se debe de usar nunca

11. Modo de ejecución del procesador donde no se ejecuta ninguna verificación.

- ◯ Modo kernel
- ◯ Modo de usuario
- ◯ Modo inseguro
- ◯ Modo dual

12. ¿Cuál es el principal problema de los hilos a nivel de usuario?

- ◯ No podemos tener varios procesos por hilo
- ◯ El bloqueo del proceso, bloquea todos los hilos
- ◯ No podemos calendarizar hilos dentro del proceso
- ◯ No podemos tener varios hilos por proceso

13. ¿Qué son las direcciones de memoria virtuales?

- ◯ Capa de indirección que le da flexibilidad al sistema operativo para administrar la memoria
- ◯ Memoria adicional que crea el sistema operativo para engañar a las aplicaciones
- ◯ Memoria que reserva el sistema operativo para el uso de máquinas virtuales
- ◯ Capa adicional que utiliza el sistema operativo en modo kernel

14. ¿Qué es una llamada de sistema?

- ◯ Código del kernel que ejecuta código de usuario
- ◯ Funciones de hardware que llama el usuario
- ◯ Llamada que hace el kernel para realizar una instrucción
- ◯ Funciones del kernel que permiten al usuario accesar a recursos restringidos

15. Región de la memoria reservada por el sistema operativo para alojar estructuras de datos que el proceso pueda necesitar.

- ◯ Stack
- ◯ Swap
- ◯ Heap
- ◯ RAM

16. ¿Cuál de las siguientes **no** es una característica de un hilo?

- ◯ Tiene un stack propio
- ◯ Tiene un contador de programa propio
- ◯ Comparte el código del programa
- ◯ Tiene un segmento de datos propio

17. Tipo de kernel en el que la funcionalidad mínima está dentro de él, y el resto se encuentra a nivel de usuario.

- ◯ Microkernel
- ◯ Macrokernel
- ◯ Híbrido
- ◯ Monolítico

18. ¿Qué hace la función `yield` de un hilo?

- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a esperando (*waiting*)
- ◯ Duerme al hilo por un tiempo determinado, y cambia el estado a esperando (*waiting*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a terminado (*finished*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a listo (*ready*)

19. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de cargar `prog` en la memoria?

- ◯ Informar al calendarizador que el programa se puede ejecutar
- ◯ Inicializar el hardware para que se ejecute el incio del programa
- ◯ Copiar los argumentos en la memoria
- ◯ Inicializar el PCB en el kernel

20. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de inicializar el espacio de memoria?

- ◯ Informar al calendarizador que el programa se puede ejecutar
- ◯ Inicializar el hardware para que se ejecute el incio del programa
- ◯ Cargar `prog` en la memoria
- ◯ Copiar los argumentos en la memoria

21. Es una señal síncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Interrupción
- ◯ Señal de software
- ◯ Trampa
- ◯ Señal de I/O

22. ¿Cuántos procesos son creados durante la ejecución del siguiente programa?

```
main(int argc, char** argv){
  forkthem(5);
}
void forkthem(int n){
  if (n > 0){
    fork();
    forkthem(n-1);
  }
}
```

- ◯ 30
- ◯ 16
- ◯ 5
- ◯ 4

23. ¿Qué es un sistema operativo?

- ◯ Es un software que administra usuarios
- ◯ Es una capa de software que administra recursos y usuarios
- ◯ No es software, pero se encarga de administrar recursos y usuarios
- ◯ Es un software que administra recursos

24. ¿Qué hace la función `wait` de una variable de condición?

- ◯ Mueve el hilo a la lista de espera atómicamente
- ◯ Atómicamente libera los seguros y mueve el hilo a la lista de espera
- ◯ Libera el seguro global, y duerme al hilo
- ◯ Duerme el hilo atómicamente mientras despierta al siguiente

Dado el siguiente código

```
main(int argc, char** argv){
  int child == fork();
  int x = 5;
  if (child == 0) {
    x += 5;
  } else {
    child = fork();
    x += 10;
    if (child) {
      x+=5;
    }
  }
}
```

25. ¿Cuál de los siguientes **no** es un valor que toma la variable `x` al término de los procesos del programa anterior?

- ◯ 5
- ◯ 15
- ◯ 20
- ◯ 10

26. ¿Cuántas copias distintas de la variable `x` existen en la ejecución del programa anterior?

- ◯ 2
- ◯ 3
- ◯ 1
- ◯ 4

27. ¿Cuál de los siguientes elementos **no** es compartido por los hilos?

- ◯ Stack
- ◯ Variables globales
- ◯ Código
- ◯ Heap

28. Es una señal asíncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Señal de software
- ◯ Señal de I/O
- ◯ Trampa
- ◯ Interrupción

29. ¿Qué es una operación atómica?

- ◯ Es una operación de hardware que no puede ser dividida
- ◯ Es una operación del kernel que no puede ser dividida
- ◯ Es una operación que deshabilita las interrupciones
- ◯ Es una operación indivisible que no puede ser dividida

30. ¿Qué es el multi-hilado preventivo?

- ◯ Es cuando los hilos entregan el procesador voluntariamente, no son interrumpidos
- ◯ Es cuando los hilos no usan el procesador hasta que se les entrega indefinidamente
- ◯ Es cuando los hilos que se ejecutan no pueden descalendarizarse
- ◯ Es cuando los hilos que se ejecutan pueden ser cambiados indistintamente

31. ¿Cuál de los siguientes eventos **no** genera una transición de modo kernel a usuario?

- ◯ Llamada I/O a un dispositivo
- ◯ Continuar después de una interrupción
- ◯ Creación de nuevo proceso
- ◯ Cambiar a un proceso diferente

32. ¿Cuál de las siguientes es una característica de `fork`?

- ◯ Copiar a la memoria el código del programa del proceso padre
- ◯ Copiar el proceso padre con privilegios distintos
- ◯ Copiar el proceso padre completamente
- ◯ Crear una copia del proceso padre, pero no puede ser confiado igual que él

33. ¿Qué es un hilo?

- ◯ Una secuencia de ejecución que puede ser calendarizada independientemente
- ◯ Varias secuencias de ejecución que pueden ser calendarizadas dependientemente
- ◯ Varias secuencias de ejecución que pueden ser calendarizadas independientemente
- ◯ Una secuencia de ejecución que es calendarizada dependientemente

34. ¿Qué es concurrencia?

- ◯ Realizar múltiples actividades sin la interrupción de otras
- ◯ Realizar una actividad sin la interrupción de otra
- ◯ Realizar múltiples actividades una después de la otra
- ◯ Realizar múltiples actividades simultáneamente

35. Región de la memoria reservada por el sistema operativo para mantener el estado de las variables locales durante la llamada a funciones.

- ◯ Stack
- ◯ RAM
- ◯ Heap
- ◯ Swap

36. ¿Qué característica del sistema operativo permite que opere independientemente del hardware en la computadora?

- ◯ Compartimiento de recursos
- ◯ Virtualización
- ◯ Ilusionista
- ◯ Portabilidad

37. ¿Cuál es el primer paso en la creación del proceso del programa `prog`?

- ◯ Crear e inicializar el PCB en el kernel
- ◯ Inicializar la memoria
- ◯ Copiar `prog` a la memoria
- ◯ Inicializar el hardware para que se ejecute el proceso desde el inicio

38. ¿Cuál de los siguientes elementos **no** pertenece al *TCB*?

- ◯ Variables locales
- ◯ Puntero al stack frame
- ◯ Estado del hilo
- ◯ Registros salvados

39. ¿Qué es una variable de condición?

- ◯ Objeto de sincronización que implementa `yield` en modo usuario
- ◯ Variable de sincronización que permite la exclusión mutua
- ◯ Varialbe de sincronización que implementa `yield`
- ◯ Variable de sincronización que permite la espera eficiente de un hilo

40. ¿Qué es un pipe?

- ◯ Es un buffer del kernel entre dos procesos
- ◯ Es un buffer del kernel entre dos descriptores de archivos
- ◯ Es un buffer de usuario entre dos procesos
- ◯ Es un buffer de usuario entre dos descriptores de archivos

41. Dado el siguiente código del programa `main`

```
1  char bar(){
2    int b = 2;
3    return 'a';
4  }
5  int foo(int a, char b){
6    int i=0;
7    bar();
8  }
9  main(){
10   int b=2;
11   foo();
12 }
```

Dibuje el stack completo del programa cuando se encuentra ejecutando la linea número 2. Utilice el número de línea para referirse a la dirección de cada instrucción. ◯I ◯P ◯C

El siguiente código implementa una lista circular.

```
const int MAX = 10;

class Queue{
  int items[MAX];
  int front;
  int nextEmpty;
public:
  Queue() { front = nextEmpty = 0;}
  ~Queue();
  void insert(int item);
  int remove();
}

void Queue::insert(int item){
  items[nextEmpty%MAX] = item;
  nextEmpty++;
}
int Queue::remove(){
  int item;
  item = items[front%MAX];
  front++;
  return item;
}
```

Sin embargo, las funciones `Queue::insert` y `Queue::remove` no están sincronizadas. Por lo que más elementos de los que soporta la lista pueden insertarse o removerse.

Responda lo que se le solicita a continuación. Puede declarar los `Lock` y `CV` que considere necesarios. Tenga en cuenta que la implementación debe de considerar que no puede insertar si la lista está llena, por lo que deberá de detener la función que está insertando. De manera similar, cuando elimine si no hay ningún elemento en la lista deberá esperar a que uno exista.

42. Reescriba la función `Queue::insert` para que inserte de manera segura en la lista circular. ◯I ◯P ◯C

43. Reescriba la función `Queue::remove` para que elimine de manera segura en la lista circular. ◯I ◯P ◯C

udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA

Solemne 1 **SO (CIT 2003-1)**

Instrucciones. Marque las casillas (◯) completamente sin salirse de ellas (por ejemplo ●). Responda a los siguientes cuestionamientos en las hojas que se le entregan **marcando una única opción**. Se utilizará factor de corrección 4 a 1 (las respuestas en blanco no se toman en cuenta). *Las últimas tres preguntas se evaluarán de manera Completa (4 pts.), Parcial (2 pts.), e Incompleta (0 pts.). No llene las casillas de estas últimas tres preguntas, sino que responda en el espacio indicado.*

◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0
◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1
◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2
◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3
◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4
◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5
◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6
◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7
◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8
◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9

← Marque su RUT sin código verificador (el número después del guión), y escriba sus nombres y apellidos abajo.

Nombre(s) y apellido(s):

.....

.....

1. ¿Qué es un hilo?

- ◯ Varias secuencias de ejecución que pueden ser calendarizadas dependientemente
- ◯ Una secuencia de ejecución que puede ser calendarizada independientemente
- ◯ Una secuencia de ejecución que es calendarizada dependientemente
- ◯ Varias secuencias de ejecución que pueden ser calendarizadas independientemente

2. ¿Cuál es el estado de un hilo en el que está listo para ejecutarse pero no está en el procesador?

- ◯ Ejecutandose (*running*)
- ◯ Listo (*ready*)
- ◯ Esperando (*wait*)
- ◯ Inicializado (*init*)

3. ¿Qué es concurrencia?

- ◯ Realizar múltiples actividades simultáneamente
- ◯ Realizar múltiples actividades una después de la otra
- ◯ Realizar una actividad sin la interrupción de otra
- ◯ Realizar múltiples actividades sin la interrupción de otras

4. Región de la memoria reservada por el sistema operativo para mantener el estado de las variables locales durante la llamada a funciones.

- ◯ Swap
- ◯ RAM
- ◯ Stack
- ◯ Heap

5. ¿Cuál de los siguientes elementos **no** es compartido por los hilos?

- ◯ Stack
- ◯ Código
- ◯ Heap
- ◯ Variables globales

Dado el siguiente código

```
main(int argc, char** argv){
  int child == fork();
  int x = 5;
  if (child == 0) {
    x += 5;
  } else {
    child = fork();
    x += 10;
    if (child) {
      x+=5;
    }
  }
}
```

6. ¿Cuántas copias distintas de la variable `x` existen en la ejecución del programa anterior?

- ◯ 1
- ◯ 2
- ◯ 4
- ◯ 3

7. ¿Cuál de los siguientes **no** es un valor que toma la variable `x` al término de los procesos del programa anterior?

- ◯ 5
- ◯ 20
- ◯ 10
- ◯ 15

8. ¿Qué característica del sistema operativo permite que opere independientemente del hardware en la computadora?

- ◯ Portabilidad
- ◯ Compartimiento de recursos
- ◯ Ilusionista
- ◯ Virtualización

9. ¿Cuál de los siguientes eventos **no** genera una transición de modo kernel a usuario?

- ◯ Llamada I/O a un dispositivo
- ◯ Creación de nuevo proceso
- ◯ Continuar después de una interrupción
- ◯ Cambiar a un proceso diferente

10. ¿Cuál de los siguientes elementos **no** pertenece al *TCB*?

- ◯ Variables locales
- ◯ Puntero al stack frame
- ◯ Estado del hilo
- ◯ Registros salvados

11. ¿Cuál es el primer paso en la creación del proceso del programa `prog`?

- ◯ Copiar `prog` a la memoria
- ◯ Crear e inicializar el PCB en el kernel
- ◯ Inicializar el hardware para que se ejecute el proceso desde el inicio
- ◯ Inicializar la memoria

12. ¿Cuál de las siguientes **no** es una característica de un hilo?

- ◯ Tiene un segmento de datos propio
- ◯ Tiene un contador de programa propio
- ◯ Comparte el código del programa
- ◯ Tiene un stack propio

13. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de cargar `prog` en la memoria?

- ◯ Inicializar el hardware para que se ejecute el incio del programa
- ◯ Informar al calendarizador que el programa se puede ejecutar
- ◯ Inicializar el PCB en el kernel
- ◯ Copiar los argumentos en la memoria

14. Tipo de kernel en el que la funcionalidad mínima está dentro de él, y el resto se encuentra a nivel de usuario.

- ◯ Monolítico
- ◯ Híbrido
- ◯ Microkernel
- ◯ Macrokernel

15. ¿Donde almacena el sistema operativo la toda la información sobre un proceso en particular?

- ◯ Heap
- ◯ Stack
- ◯ Memoria
- ◯ *Process control block* (PCB)

16. ¿Qué es una llamada de sistema?

- ◯ Código del kernel que ejecuta código de usuario
- ◯ Llamada que hace el kernel para realizar una instrucción
- ◯ Funciones del kernel que permiten al usuario accesar a recursos restringidos
- ◯ Funciones de hardware que llama el usuario

17. ¿Cuál de las siguientes **no** es una característica de un descriptor de archivo de UNIX?

- ◯ Se escriben a través de bytes
- ◯ Se cierran completamente después de usar
- ◯ Se abren antes de usar
- ◯ Se leen a través de un buffer

18. ¿Cuál de las siguientes es una característica de `fork`?

- ◯ Copiar el proceso padre con privilegios distintos
- ◯ Copiar el proceso padre completamente
- ◯ Crear una copia del proceso padre, pero no puede ser confiado igual que él
- ◯ Copiar a la memoria el código del programa del proceso padre

19. Región de la memoria reservada por el sistema operativo para alojar estructuras de datos que el proceso pueda necesitar.

- ◯ RAM
- ◯ Stack
- ◯ Heap
- ◯ Swap

20. ¿Qué es la exclusión mutua?

- ◯ Propiedad donde los hilos se bloquean mutuamente
- ◯ Objeto de sincronización que bloquea los hilos
- ◯ Propiedad en la que solamente un hilo puede accesar simultáneamente
- ◯ Propiedad de la concurrencia para mantener los hilos excluidos

21. ¿Cuál de las siguientes afirmaciones es verdadera respecto a *spin wait*?

- ◯ Produce *deadlocks*
- ◯ Es justificable en monoprocesadores
- ◯ Es justificable en multiprocesadores
- ◯ No se debe de usar nunca

22. ¿Cuántos procesos son creados durante la ejecución del siguiente programa?

```
main(int argc, char** argv){
  forkthem(5);
}
void forkthem(int n){
  if (n > 0){
    fork();
    forkthem(n-1);
  }
}
```

- ◯ 16
- ◯ 5
- ◯ 30
- ◯ 4

23. Es una señal síncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Trampa
- ◯ Señal de software
- ◯ Interrupción
- ◯ Señal de I/O

24. ¿Qué es una operación atómica?

- ◯ Es una operación del kernel que no puede ser dividida
- ◯ Es una operación indivisible que no puede ser dividida
- ◯ Es una operación de hardware que no puede ser dividida
- ◯ Es una operación que deshabilita las interrupciones

25. ¿Qué son las direcciones de memoria virtuales?

- ◯ Capa de indirección que le da flexibilidad al sistema operativo para administrar la memoria
- ◯ Capa adicional que utiliza el sistema operativo en modo kernel
- ◯ Memoria adicional que crea el sistema operativo para engañar a las aplicaciones
- ◯ Memoria que reserva el sistema operativo para el uso de máquinas virtuales

26. Modo de ejecución del procesador donde no se ejecuta ninguna verificación.

- ◯ Modo dual
- ◯ Modo kernel
- ◯ Modo inseguro
- ◯ Modo de usuario

27. ¿Qué es un sistema operativo?

- ◯ Es una capa de software que administra recursos y usuarios
- ◯ Es un software que administra recursos
- ◯ Es un software que administra usuarios
- ◯ No es software, pero se encarga de administrar recursos y usuarios

28. ¿Qué es un pipe?

- ◯ Es un buffer del kernel entre dos descriptores de archivos
- ◯ Es un buffer del kernel entre dos procesos
- ◯ Es un buffer de usuario entre dos procesos
- ◯ Es un buffer de usuario entre dos descriptores de archivos

29. Llamada del sistema de UNIX que le permite a las aplicaciones el comunicarse entre sí para terminarlas, suspenderlas, o resumirlas.

- ◯ `wait`
- ◯ `signal`
- ◯ `fork`
- ◯ `exec`

30. Tipo de kernel en el que la mayoría de la funcionalidad de éste se encuentra dentro de él.

- ◯ Híbrido
- ◯ Microkernel
- ◯ Monolítico
- ◯ Macrokernel

31. ¿Qué es el multi-hilado preventivo?

- ◯ Es cuando los hilos entregan el procesador voluntariamente, no son interrumpidos
- ◯ Es cuando los hilos no usan el procesador hasta que se les entrega indefinidamente
- ◯ Es cuando los hilos que se ejecutan no pueden descalendarizarse
- ◯ Es cuando los hilos que se ejecutan pueden ser cambiados indistintamente

32. ¿Qué es la ejecución de una aplicación con permisos restringidos?

- ◯ Hilo
- ◯ Proceso
- ◯ Programa
- ◯ Aplicación

33. ¿Cuál es el principal problema de los hilos a nivel de usuario?

- ◯ El bloqueo del proceso, bloquea todos los hilos
- ◯ No podemos tener varios hilos por proceso
- ◯ No podemos tener varios procesos por hilo
- ◯ No podemos calendarizar hilos dentro del proceso

34. Modo de ejecución del procesador en el que verifica cada instrucción antes de ejecutarla.

- ◯ Modo dual
- ◯ Modo de usuario
- ◯ Modo seguro
- ◯ Modo kernel

35. ¿Qué es una variable de condición?

- ◯ Variable de sincronización que permite la exclusión mutua
- ◯ Objeto de sincronización que implementa `yield` en modo usuario
- ◯ Variable de sincronización que permite la espera eficiente de un hilo
- ◯ Varialbe de sincronización que implementa `yield`

36. ¿Qué hace la función `yield` de un hilo?

- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a terminado (*finished*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a esperando (*waiting*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a listo (*ready*)
- ◯ Duerme al hilo por un tiempo determinado, y cambia el estado a esperando (*waiting*)

37. ¿Qué hace la función `wait` de una variable de condición?

- ◯ Libera el seguro global, y duerme al hilo
- ◯ Mueve el hilo a la lista de espera atómicamente
- ◯ Atómicamente libera los seguros y mueve el hilo a la lista de espera
- ◯ Duerme el hilo atómicamente mientras despierta al siguiente

38. ¿Cuándo se da una condición de carrera?

- ◯ Varios hilos se pueden intercalar en su ejecución simultánea
- ◯ Varios hilos se ejecutan simultáneamente
- ◯ El estado de la ejecución de un programa depende del intercalado de diferentes hilos
- ◯ Existe una sección crítica

39. Es una señal asíncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Interrupción
- ◯ Trampa
- ◯ Señal de I/O
- ◯ Señal de software

40. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de inicializar el espacio de memoria?

- ◯ Cargar `prog` en la memoria
- ◯ Informar al calendarizador que el programa se puede ejecutar
- ◯ Copiar los argumentos en la memoria
- ◯ Inicializar el hardware para que se ejecute el incio del programa

41. Dado el siguiente código del programa `main`

```
1  char bar(){
2    int b = 2;
3    return 'a';
4  }
5  int foo(int a, char b){
6    int i=0;
7    bar();
8  }
9  main(){
10   int b=2;
11   foo();
12 }
```

Dibuje el stack completo del programa cuando se encuentra ejecutando la linea número 2. Utilice el número de línea para referirse a la dirección de cada instrucción. ◯I ◯P ◯C

El siguiente código implementa una lista circular.

```
const int MAX = 10;

class Queue{
  int items[MAX];
  int front;
  int nextEmpty;
public:
  Queue() { front = nextEmpty = 0;}
  ~Queue();
  void insert(int item);
  int remove();
}

void Queue::insert(int item){
  items[nextEmpty%MAX] = item;
  nextEmpty++;
}
int Queue::remove(){
  int item;
  item = items[front%MAX];
  front++;
  return item;
}
```

Sin embargo, las funciones `Queue::insert` y `Queue::remove` no están sincronizadas. Por lo que más elementos de los que soporta la lista pueden insertarse o removerse.

Responda lo que se le solicita a continuación. Puede declarar los `Lock` y `CV` que considere necesarios. Tenga en cuenta que la implementación debe de considerar que no puede insertar si la lista está llena, por lo que deberá de detener la función que está insertando. De manera similar, cuando elimine si no hay ningún elemento en la lista deberá esperar a que uno exista.

42. Reescriba la función `Queue::insert` para que inserte de manera segura en la lista circular. ◯I ◯P ◯C

43. Reescriba la función `Queue::remove` para que elimine de manera segura en la lista circular. ◯I ◯P ◯C

udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA

Solemne 1 **SO (CIT 2003-1)**

Instrucciones. Marque las casillas (◯) completamente sin salirse de ellas (por ejemplo ●). Responda a los siguientes cuestionamientos en las hojas que se le entregan **marcando una única opción**. Se utilizará factor de corrección 4 a 1 (las respuestas en blanco no se toman en cuenta). *Las últimas tres preguntas se evaluarán de manera Completa (4 pts.), Parcial (2 pts.), e Incompleta (0 pts.). No llene las casillas de estas últimas tres preguntas, sino que responda en el espacio indicado.*

◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0
◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1
◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2
◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3
◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4
◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5
◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6
◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7
◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8
◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9

⟵ Marque su RUT sin código verificador (el número después del guión), y escriba sus nombres y apellidos abajo.

Nombre(s) y apellido(s):

.....

.....

1. ¿Qué es un sistema operativo?
- ◯ Es una capa de software que administra recursos y usuarios
- ◯ No es software, pero se encarga de administrar recursos y usuarios
- ◯ Es un software que administra usuarios
- ◯ Es un software que administra recursos

2. ¿Qué es una operación atómica?
- ◯ Es una operación de hardware que no puede ser dividida
- ◯ Es una operación del kernel que no puede ser dividida
- ◯ Es una operación indivisible que no puede ser dividida
- ◯ Es una operación que deshabilita las interrupciones

3. ¿Cuál de las siguientes **no** es una característica de un descriptor de archivo de UNIX?
- ◯ Se abren antes de usar
- ◯ Se leen a través de un buffer
- ◯ Se cierran completamente después de usar
- ◯ Se escriben a través de bytes

4. ¿Cuál de las siguientes **no** es una característica de un hilo?
- ◯ Tiene un segmento de datos propio
- ◯ Tiene un stack propio
- ◯ Comparte el código del programa
- ◯ Tiene un contador de programa propio

5. Modo de ejecución del procesador donde no se ejecuta ninguna verificación.
- ◯ Modo de usuario
- ◯ Modo inseguro
- ◯ Modo dual
- ◯ Modo kernel

6. ¿Cuál de las siguientes afirmaciones es verdadera respecto a *spin wait*?
- ◯ Produce *deadlocks*
- ◯ Es justificable en monoprocesadores
- ◯ Es justificable en multiprocesadores
- ◯ No se debe de usar nunca

7. Tipo de kernel en el que la funcionalidad mínima está dentro de él, y el resto se encuentra a nivel de usuario.
- ◯ Microkernel
- ◯ Monolítico
- ◯ Híbrido
- ◯ Macrokernel

8. ¿Qué es un pipe?
- ◯ Es un buffer de usuario entre dos descriptores de archivos
- ◯ Es un buffer del kernel entre dos descriptores de archivos
- ◯ Es un buffer de usuario entre dos procesos
- ◯ Es un buffer del kernel entre dos procesos

9. Región de la memoria reservada por el sistema operativo para mantener el estado de las variables locales durante la llamada a funciones.
- ◯ Heap
- ◯ Swap
- ◯ Stack
- ◯ RAM

10. ¿Qué es la exclusión mutua?
- ◯ Propiedad de la concurrencia para mantener los hilos excluidos
- ◯ Propiedad donde los hilos se bloquean mutuamente
- ◯ Propiedad en la que solamente un hilo puede acesar simultáneamente
- ◯ Objeto de sincronización que bloquea los hilos

11. Modo de ejecución del procesador en el que verifica cada instrucción antes de ejecutarla.
- ◯ Modo de usuario
- ◯ Modo kernel
- ◯ Modo dual
- ◯ Modo seguro

12. Es una señal síncrona al procesador que indica que ocurrió un evento que requiere su atención.
- ◯ Señal de software
- ◯ Trampa
- ◯ Señal de I/O
- ◯ Interrupción

13. ¿Cuál de los siguientes elementos **no** es compartido por los hilos?
- ◯ Variables globales
- ◯ Heap
- ◯ Código
- ◯ Stack

14. ¿Qué son las direcciones de memoria virtuales?

- ◯ Memoria que reserva el sistema operativo para el uso de máquinas virtuales
- ◯ Capa de indirección que le da flexibilidad al sistema operativo para administrar la memoria
- ◯ Capa adicional que utiliza el sistema operativo en modo kernel
- ◯ Memoria adicional que crea el sistema operativo para engañar a las aplicaciones

15. ¿Qué característica del sistema operativo permite que opere independientemente del hardware en la computadora?

- ◯ Virtualización
- ◯ Portabilidad
- ◯ Compartimiento de recursos
- ◯ Ilusionista

16. ¿Cuál de los siguientes eventos **no** genera una transición de modo kernel a usuario?

- ◯ Continuar después de una interrupción
- ◯ Creación de nuevo proceso
- ◯ Llamada I/O a un dispositivo
- ◯ Cambiar a un proceso diferente

17. ¿Cuál de las siguientes es una característica de `fork`?

- ◯ Copiar a la memoria el código del programa del proceso padre
- ◯ Crear una copia del proceso padre, pero no puede ser confiado igual que él
- ◯ Copiar el proceso padre completamente
- ◯ Copiar el proceso padre con privilegios distintos

18. ¿Donde almacena el sistema operativo la toda la información sobre un proceso en particular?

- ◯ *Process control block* (PCB)
- ◯ Memoria
- ◯ Heap
- ◯ Stack

19. ¿Cuál es el estado de un hilo en el que está listo para ejecutarse pero no está en el procesador?

- ◯ Listo (*ready*)
- ◯ Esperando (*wait*)
- ◯ Ejecutandose (*running*)
- ◯ Inicializado (*init*)

20. ¿Qué hace la función `wait` de una variable de condición?

- ◯ Duerme el hilo atómicamente mientras despierta al siguiente
- ◯ Libera el seguro global, y duerme al hilo
- ◯ Mueve el hilo a la lista de espera atómicamente
- ◯ Atómicamente libera los seguros y mueve el hilo a la lista de espera

21. ¿Cuándo se da una condición de carrera?

- ◯ Varios hilos se ejecutan simultáneamente
- ◯ Existe una sección crítica
- ◯ Varios hilos se pueden intercalar en su ejecución simultánea
- ◯ El estado de la ejecución de un programa depende del intercalado de diferentes hilos

22. Llamada del sistema de UNIX que le permite a las aplicaciones el comunicarse entre sí para terminarlas, suspenderlas, o resumirlas.

- ◯ `fork`
- ◯ `wait`
- ◯ `signal`
- ◯ `exec`

23. ¿Qué es concurrencia?

- ◯ Realizar múltiples actividades una después de la otra
- ◯ Realizar múltiples actividades simultáneamente
- ◯ Realizar una actividad sin la interrupción de otra
- ◯ Realizar múltiples actividades sin la interrupción de otras

24. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de cargar `prog` en la memoria?

- ◯ Informar al calendarizador que el programa se puede ejecutar
- ◯ Inicializar el PCB en el kernel
- ◯ Inicializar el hardware para que se ejecute el incio del programa
- ◯ Copiar los argumentos en la memoria

25. ¿Qué es el multi-hilado preventivo?

- ◯ Es cuando los hilos que se ejecutan no pueden descalendarizarse
- ◯ Es cuando los hilos no usan el procesador hasta que se les entrega indefinidamente
- ◯ Es cuando los hilos entregan el procesador voluntariamente, no son interrumpidos
- ◯ Es cuando los hilos que se ejecutan pueden ser cambiados indistintamente

26. ¿Qué es una variable de condición?

- ◯ Variable de sincronización que permite la exclusión mutua
- ◯ Varialbe de sincronización que implementa `yield`
- ◯ Variable de sincronización que permite la espera eficiente de un hilo
- ◯ Objeto de sincronización que implementa `yield` en modo usuario

27. ¿Cuántos procesos son creados durante la ejecución del siguiente programa?

```
main(int argc, char** argv){
  forkthem(5);
}
void forkthem(int n){
  if (n > 0){
    fork();
    forkthem(n-1);
  }
}
```

- ◯ 16
- ◯ 4
- ◯ 30
- ◯ 5

28. ¿Cuál es el principal problema de los hilos a nivel de usuario?

- ◯ El bloqueo del proceso, bloquea todos los hilos
- ◯ No podemos tener varios procesos por hilo
- ◯ No podemos calendarizar hilos dentro del proceso
- ◯ No podemos tener varios hilos por proceso

29. ¿Qué es una llamada de sistema?

- ◯ Funciones del kernel que permiten al usuario accesar a recursos restringidos
- ◯ Código del kernel que ejecuta código de usuario
- ◯ Llamada que hace el kernel para realizar una instrucción
- ◯ Funciones de hardware que llama el usuario

30. Tipo de kernel en el que la mayoría de la funcionalidad de éste se encuentra dentro de él.

- ◯ Híbrido
- ◯ Macrokernel
- ◯ Monolítico
- ◯ Microkernel

31. ¿Cuál de los siguientes elementos **no** pertenece al *TCB*?

- ◯ Variables locales
- ◯ Puntero al stack frame
- ◯ Estado del hilo
- ◯ Registros salvados

32. Región de la memoria reservada por el sistema operativo para alojar estructuras de datos que el proceso pueda necesitar.

- ◯ Swap
- ◯ Heap
- ◯ RAM
- ◯ Stack

33. ¿Qué hace la función `yield` de un hilo?

- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a terminado (*finished*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a esperando (*waiting*)
- ◯ Duerme al hilo por un tiempo determinado, y cambia el estado a esperando (*waiting*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a listo (*ready*)

Dado el siguiente código

```
main(int argc, char** argv){
  int child == fork();
  int x = 5;
  if (child == 0) {
    x += 5;
  } else {
    child = fork();
    x += 10;
    if (child) {
      x+=5;
    }
  }
}
```

34. ¿Cuál de los siguientes **no** es un valor que toma la variable `x` al término de los procesos del programa anterior?

- ◯ 15
- ◯ 20
- ◯ 5
- ◯ 10

35. ¿Cuántas copias distintas de la variable `x` existen en la ejecución del programa anterior?

- ◯ 4
- ◯ 1
- ◯ 2
- ◯ 3

36. ¿Qué es un hilo?

- ◯ Una secuencia de ejecución que puede ser calendarizada independientemente
- ◯ Varias secuencias de ejecución que pueden ser calendarizadas dependientemente
- ◯ Varias secuencias de ejecución que pueden ser calendarizadas independientemente
- ◯ Una secuencia de ejecución que es calendarizada dependientemente

37. Es una señal asíncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Interrupción
- ◯ Señal de software
- ◯ Señal de I/O
- ◯ Trampa

38. ¿Qué es la ejecución de una aplicación con permisos restringidos?

- ◯ Programa
- ◯ Proceso
- ◯ Hilo
- ◯ Aplicación

39. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de inicializar el espacio de memoria?

- ◯ Informar al calendarizador que el programa se puede ejecutar
- ◯ Copiar los argumentos en la memoria
- ◯ Cargar `prog` en la memoria
- ◯ Inicializar el hardware para que se ejecute el inicio del programa

40. ¿Cuál es el primer paso en la creación del proceso del programa `prog`?

- ◯ Crear e inicializar el PCB en el kernel
- ◯ Inicializar la memoria
- ◯ Inicializar el hardware para que se ejecute el proceso desde el inicio
- ◯ Copiar `prog` a la memoria

41. Dado el siguiente código del programa `main`

```
1  char bar(){
2    int b = 2;
3    return 'a';
4  }
5  int foo(int a, char b){
6    int i=0;
7    bar();
8  }
9  main(){
10   int b=2;
11   foo();
12 }
```

Dibuje el stack completo del programa cuando se encuentra ejecutando la linea número 2. Utilice el número de línea para referirse a la dirección de cada instrucción. ◯I ◯P ◯C

El siguiente código implementa una lista circular.

```
const int MAX = 10;

class Queue{
  int items[MAX];
  int front;
  int nextEmpty;
public:
  Queue() { front = nextEmpty = 0;}
  ~Queue();
  void insert(int item);
  int remove();
}

void Queue::insert(int item){
  items[nextEmpty%MAX] = item;
  nextEmpty++;
}
int Queue::remove(){
  int item;
  item = items[front%MAX];
  front++;
  return item;
}
```

Sin embargo, las funciones `Queue::insert` y `Queue::remove` no están sincronizadas. Por lo que más elementos de los que soporta la lista pueden insertarse o removerse.

Responda lo que se le solicita a continuación. Puede declarar los `Lock` y `CV` que considere necesarios. Tenga en cuenta que la implementación debe de considerar que no puede insertar si la lista está llena, por lo que deberá de detener la función que está insertando. De manera similar, cuando elimine si no hay ningún elemento en la lista deberá esperar a que uno exista.

42. Reescriba la función `Queue::insert` para que inserte de manera segura en la lista circular. ◯I ◯P ◯C

43. Reescriba la función `Queue::remove` para que elimine de manera segura en la lista circular. ◯I ◯P ◯C

udp Escuela de Informática y Telecomunicaciones
FACULTAD DE INGENIERÍA

Solemne 1 **SO (CIT 2003-1)**

Instrucciones. Marque las casillas (◯) completamente sin salirse de ellas (por ejemplo ●). Responda a los siguientes cuestionamientos en las hojas que se le entregan **marcando una única opción**. Se utilizará factor de corrección 4 a 1 (las respuestas en blanco no se toman en cuenta). *Las últimas tres preguntas se evaluarán de manera Completa (4 pts.), Parcial (2 pts.), e Incompleta (0 pts.). No llene las casillas de estas últimas tres preguntas, sino que responda en el espacio indicado.*

◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0 ◯0
◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1 ◯1
◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2 ◯2
◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3 ◯3
◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4 ◯4
◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5 ◯5
◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6 ◯6
◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7 ◯7
◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8 ◯8
◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9 ◯9

⟵ Marque su RUT sin código verificador (el número después del guión), y escriba sus nombres y apellidos abajo.

Nombre(s) y apellido(s):

.....

.....

1. ¿Donde almacena el sistema operativo la toda la información sobre un proceso en particular?

- ◯ Heap
- ◯ Memoria
- ◯ Stack
- ◯ *Process control block* (PCB)

2. Tipo de kernel en el que la mayoría de la funcionalidad de éste se encuentra dentro de él.

- ◯ Macrokernel
- ◯ Microkernel
- ◯ Monolítico
- ◯ Híbrido

3. ¿Qué es una llamada de sistema?

- ◯ Llamada que hace el kernel para realizar una instrucción
- ◯ Funciones del kernel que permiten al usuario accesar a recursos restringidos
- ◯ Código del kernel que ejecuta código de usuario
- ◯ Funciones de hardware que llama el usuario

4. ¿Qué es un hilo?

- ◯ Una secuencia de ejecución que es calendarizada dependientemente
- ◯ Varias secuencias de ejecución que pueden ser calendarizadas independientemente
- ◯ Una secuencia de ejecución que puede ser calendarizada independientemente
- ◯ Varias secuencias de ejecución que pueden ser calendarizadas dependientemente

5. Modo de ejecución del procesador en el que verifica cada instrucción antes de ejecutarla.

- ◯ Modo de usuario
- ◯ Modo seguro
- ◯ Modo dual
- ◯ Modo kernel

6. ¿Qué es una variable de condición?

- ◯ Variable de sincronización que permite la espera eficiente de un hilo
- ◯ Objeto de sincronización que implementa `yield` en modo usuario
- ◯ Variable de sincronización que permite la exclusión mutua
- ◯ Varialbe de sincronización que implementa `yield`

7. ¿Cuál es el principal problema de los hilos a nivel de usuario?

- ◯ No podemos calendarizar hilos dentro del proceso
- ◯ No podemos tener varios procesos por hilo
- ◯ El bloqueo del proceso, bloquea todos los hilos
- ◯ No podemos tener varios hilos por proceso

8. ¿Cuál de las siguientes afirmaciones es verdadera respecto a *spin wait*?

- ◯ Es justificable en multiprocesadores
- ◯ No se debe de usar nunca
- ◯ Produce *deadlocks*
- ◯ Es justificable en monoprocesadores

9. ¿Qué es el multi-hilado preventivo?

- ◯ Es cuando los hilos entregan el procesador voluntariamente, no son interrumpidos
- ◯ Es cuando los hilos no usan el procesador hasta que se les entrega indefinidamente
- ◯ Es cuando los hilos que se ejecutan pueden ser cambiados indistintamente
- ◯ Es cuando los hilos que se ejecutan no pueden descalendarizarse

10. Región de la memoria reservada por el sistema operativo para alojar estructuras de datos que el proceso pueda necesitar.

- ◯ Swap
- ◯ RAM
- ◯ Stack
- ◯ Heap

11. ¿Cuál de los siguientes elementos **no** pertenece al *TCB*?

- ◯ Variables locales
- ◯ Puntero al stack frame
- ◯ Estado del hilo
- ◯ Registros salvados

12. ¿Cuál de las siguientes es una característica de `fork`?

- ◯ Copiar el proceso padre con privilegios distintos
- ◯ Copiar el proceso padre completamente
- ◯ Copiar a la memoria el código del programa del proceso padre
- ◯ Crear una copia del proceso padre, pero no puede ser confiado igual que él

13. ¿Qué característica del sistema operativo permite que opere independientemente del hardware en la computadora?

- ◯ Portabilidad
- ◯ Virtualización
- ◯ Compartimiento de recursos
- ◯ Ilusionista

14. ¿Cuál de las siguientes **no** es una característica de un hilo?

- ◯ Tiene un stack propio
- ◯ Tiene un contador de programa propio
- ◯ Comparte el código del programa
- ◯ Tiene un segmento de datos propio

15. ¿Qué es concurrencia?

- ◯ Realizar múltiples actividades una después de la otra
- ◯ Realizar múltiples actividades simultáneamente
- ◯ Realizar una actividad sin la interrupción de otra
- ◯ Realizar múltiples actividades sin la interrupción de otras

16. ¿Cuántos procesos son creados durante la ejecución del siguiente programa?

```
main(int argc, char** argv){
  forkthem(5);
}
void forkthem(int n){
  if (n > 0){
    fork();
    forkthem(n-1);
  }
}
```

- ◯ 16
- ◯ 5
- ◯ 4
- ◯ 30

17. Es una señal síncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Señal de I/O
- ◯ Señal de software
- ◯ Interrupción
- ◯ Trampa

18. ¿Qué es la exclusión mutua?

- ◯ Objeto de sincronización que bloquea los hilos
- ◯ Propiedad de la concurrencia para mantener los hilos excluidos
- ◯ Propiedad en la que solamente un hilo puede acesar simultáneamente
- ◯ Propiedad donde los hilos se bloquean mutuamente

19. ¿Qué es una operación atómica?

- ◯ Es una operación de hardware que no puede ser dividida
- ◯ Es una operación indivisible que no puede ser dividida
- ◯ Es una operación del kernel que no puede ser dividida
- ◯ Es una operación que deshabilita las interrupciones

20. ¿Qué hace la función `wait` de una variable de condición?

- ◯ Libera el seguro global, y duerme al hilo
- ◯ Mueve el hilo a la lista de espera atómicamente
- ◯ Duerme el hilo atómicamente mientras despierta al siguiente
- ◯ Atómicamente libera los seguros y mueve el hilo a la lista de espera

21. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de cargar `prog` en la memoria?

- ◯ Informar al calendarizador que el programa se puede ejecutar
- ◯ Inicializar el PCB en el kernel
- ◯ Copiar los argumentos en la memoria
- ◯ Inicializar el hardware para que se ejecute el incio del programa

22. ¿Qué es un pipe?

- ◯ Es un buffer del kernel entre dos procesos
- ◯ Es un buffer de usuario entre dos descriptores de archivos
- ◯ Es un buffer del kernel entre dos descriptores de archivos
- ◯ Es un buffer de usuario entre dos procesos

23. En la creación del proceso del programa `prog`, ¿qué paso es el siguiente después de inicializar el espacio de memoria?

- ◯ Copiar los argumentos en la memoria
- ◯ Inicializar el hardware para que se ejecute el incio del programa
- ◯ Cargar `prog` en la memoria
- ◯ Informar al calendarizador que el programa se puede ejecutar

24. ¿Cuál de los siguientes eventos **no** genera una transición de modo kernel a usuario?

- ◯ Continuar después de una interrupción
- ◯ Llamada I/O a un dispositivo
- ◯ Cambiar a un proceso diferente
- ◯ Creación de nuevo proceso

25. Región de la memoria reservada por el sistema operativo para mantener el estado de las variables locales durante la llamada a funciones.

- ◯ Stack
- ◯ Swap
- ◯ RAM
- ◯ Heap

26. ¿Qué es la ejecución de una aplicación con permisos restringidos?

- ◯ Programa
- ◯ Hilo
- ◯ Aplicación
- ◯ Proceso

Dado el siguiente código

```
main(int argc, char** argv){
  int child == fork();
  int x = 5;
  if (child == 0) {
    x += 5;
  } else {
    child = fork();
    x += 10;
    if (child) {
      x+=5;
    }
  }
}
```

27. ¿Cuántas copias distintas de la variable `x` existen en la ejecución del programa anterior?

- ◯ 3
- ◯ 4
- ◯ 1
- ◯ 2

28. ¿Cuál de los siguientes **no** es un valor que toma la variable `x` al término de los procesos del programa anterior?

- ◯ 10
- ◯ 5
- ◯ 20
- ◯ 15

29. Tipo de kernel en el que la funcionalidad mínima está dentro de él, y el resto se encuentra a nivel de usuario.

- ◯ Híbrido
- ◯ Microkernel
- ◯ Macrokernel
- ◯ Monolítico

30. ¿Qué es un sistema operativo?

- ◯ Es un software que administra usuarios
- ◯ Es un software que administra recursos
- ◯ No es software, pero se encarga de administrar recursos y usuarios
- ◯ Es una capa de software que administra recursos y usuarios

31. Modo de ejecución del procesador donde no se ejecuta ninguna verificación.

- ◯ Modo de usuario
- ◯ Modo dual
- ◯ Modo kernel
- ◯ Modo inseguro

32. ¿Cuál es el estado de un hilo en el que está listo para ejecutarse pero no está en el procesador?

- ◯ Inicializado (*init*)
- ◯ Ejecutandose (*running*)
- ◯ Listo (*ready*)
- ◯ Esperando (*wait*)

33. Llamada del sistema de UNIX que le permite a las aplicaciones el comunicarse entre sí para terminarlas, suspenderlas, o resumirlas.

- ◯ `signal`
- ◯ `fork`
- ◯ `exec`
- ◯ `wait`

34. ¿Cuál es el primer paso en la creación del proceso del programa `prog`?

- ◯ Inicializar la memoria
- ◯ Inicializar el hardware para que se ejecute el proceso desde el inicio
- ◯ Copiar `prog` a la memoria
- ◯ Crear e inicializar el PCB en el kernel

35. ¿Qué son las direcciones de memoria virtuales?

- ◯ Memoria que reserva el sistema operativo para el uso de máquinas virtuales
- ◯ Capa adicional que utiliza el sistema operativo en modo kernel
- ◯ Memoria adicional que crea el sistema operativo para engañar a las aplicaciones
- ◯ Capa de indirección que le da flexibilidad al sistema operativo para administrar la memoria

36. ¿Cuál de las siguientes **no** es una característica de un descriptor de archivo de UNIX?

- ◯ Se cierran completamente después de usar
- ◯ Se abren antes de usar
- ◯ Se escriben a través de bytes
- ◯ Se leen a través de un buffer

37. Es una señal asíncrona al procesador que indica que ocurrió un evento que requiere su atención.

- ◯ Trampa
- ◯ Interrupción
- ◯ Señal de software
- ◯ Señal de I/O

38. ¿Cuándo se da una condición de carrera?

- ◯ El estado de la ejecución de un programa depende del intercalado de diferentes hilos
- ◯ Varios hilos se ejecutan simultáneamente
- ◯ Varios hilos se pueden intercalar en su ejecución simultánea
- ◯ Existe una sección crítica

39. ¿Cuál de los siguientes elementos **no** es compartido por los hilos?

- ◯ Heap
- ◯ Código
- ◯ Stack
- ◯ Variables globales

40. ¿Qué hace la función `yield` de un hilo?

- ◯ Duerme al hilo por un tiempo determinado, y cambia el estado a esperando (*waiting*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a listo (*ready*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a terminado (*finished*)
- ◯ Entrega voluntariamente el resto del ciclo en el procesador, y cambia el estado a esperando (*waiting*)

41. Dado el siguiente código del programa `main`

```
1  char bar(){
2    int b = 2;
3    return 'a';
4  }
5  int foo(int a, char b){
6    int i=0;
7    bar();
8  }
9  main(){
10   int b=2;
11   foo();
12 }
```

Dibuje el stack completo del programa cuando se encuentra ejecutando la linea número 2. Utilice el número de línea para referirse a la dirección de cada instrucción. ◯I ◯P ◯C

El siguiente código implementa una lista circular.

```
const int MAX = 10;

class Queue{
  int items[MAX];
  int front;
  int nextEmpty;
public:
  Queue() { front = nextEmpty = 0;}
  ~Queue();
  void insert(int item);
  int remove();
}

void Queue::insert(int item){
  items[nextEmpty%MAX] = item;
  nextEmpty++;
}
int Queue::remove(){
  int item;
  item = items[front%MAX];
  front++;
  return item;
}
```

Sin embargo, las funciones `Queue::insert` y `Queue::remove` no están sincronizadas. Por lo que más elementos de los que soporta la lista pueden insertarse o removerse.

Responda lo que se le solicita a continuación. Puede declarar los `Lock` y `CV` que considere necesarios. Tenga en cuenta que la implementación debe de considerar que no puede insertar si la lista está llena, por lo que deberá de detener la función que está insertando. De manera similar, cuando elimine si no hay ningún elemento en la lista deberá esperar a que uno exista.

42. Reescriba la función `Queue::insert` para que inserte de manera segura en la lista circular. ◯I ◯P ◯C

43. Reescriba la función `Queue::remove` para que elimine de manera segura en la lista circular. ◯I ◯P ◯C