

GUIA DE ETS DE SISTEMAS OPERATIVOS PLAN 2020

19 de enero del 2024

PROFESOR. ISRAEL SALAS RAMIREZ

De acuerdo con el temario el examen será teórico al 100 % considerando todos los temas y subtemas de este.

EL EXAMEN ES TEORICO-PRACTICO



INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA ACADÉMICA
DIRECCIÓN DE EDUCACIÓN SUPERIOR



PROGRAMA SINTÉTICO

UNIDAD ACADÉMICA: ESCUELA SUPERIOR DE CÓMPUTO, UNIDAD PROFESIONAL INTERDISCIPLINARIA DE INGENIERÍA, CAMPUS ZACATECAS	
PROGRAMA ACADÉMICO: Ingeniería en Sistemas Computacionales	
UNIDAD DE APRENDIZAJE: Sistemas Operativos	SEMESTRE: IV
PROPÓSITO DE LA UNIDAD DE APRENDIZAJE Proponer soluciones a las necesidades de sistemas computacionales actuales a partir del funcionamiento del sistema operativo.	
CONTENIDOS:	I. Estructura de un sistema operativo II. Administración de procesos III. Administración de memoria IV. Sistema de archivos V. Dispositivos de entrada y salida VI. Seguridad y virtualización

Quien desee entregar la guía tiene un **punto extra** sobre los criterios siguientes:

- a) Todas las preguntas deben de tener URL, ósea fuente bibliográfica
- b) Debe de estar contestado al 100% la guía
- c) Deben de entregar en USB con la siguiente estructura
/GUIA/ETS/SISTEMAS_OPERATIVOS
- d) Debe de ser entregada el día del examen

PRACTICO

1. Del siguiente código

```
#include <sys/types.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    pid_t pid;
    int status;

    pid=fork();
    if (pid != 0)
    {
        while (pid != wait(&status));
    }
    else
    {
        sleep (5);
        exit(5);
    }

    pid=fork();
    if (pid != 0)
    {
        while (pid != wait(&status));
    }
    else
    {
        sleep(1);
        exit(1);
    }
}
```

En cual se crea un proceso y se espera la finalización de su ejecución para crear otro proceso y al volver a esperar a su finalización, se pide modificarlo para que se creen los dos procesos y se ejecuten en paralelo, además el proceso padre debe de esperar por la finalización de los dos procesos anteriores.

2. Realizar un programa que cree 2 hijos utilizando fork (procesos pesados). El primero de ellos debe escribir los números pares (del 2 al 10) y el otro los impares (del 1 al 9). En pantalla deben aparecer los números ordenados, por lo que las ejecuciones deben ser alternas.

Se utilizarán semáforos como mecanismo de sincronización entre los procesos

3. Escriba un programa que use los servicios POSIX de proyección de archivos para comparar dos archivos. El programa recibe como argumentos los nombres de los archivos a comparar.
4. Un sistema de memoria virtual con paginación por demanda tiene un tamaño de página de 512 palabras, una memoria virtual de 16 páginas numeradas del 0 al 15 una memoria física de 4 marcos (frames) numerados de 0 a 3. El contenido actual de la memoria es:

N° de marco	Pagina
0	Pag. 4 del proceso
1	Pag. 9 del proceso
2	Pag. 5 del proceso
3	Pag. 1 del proceso

- a) Mostrar el contenido de la tabla de páginas.
b) Direcciones físicas equivalentes a las lógicas 1628, 851, 2700 y 2432.
c) ¿Qué pasa cuando se referencia la dirección lógica 1330?

5. Realiza un programa que produce al admitirse exclusión mutua para más de un recurso, en donde se presenten los siguientes casos
- ❖ El proceso P1 entra en la sección crítica para el recurso A.
 - ❖ El proceso P2 entra en la sección crítica para el recurso B.
 - ❖ El proceso P1 solicita entrar en la sección crítica para el recurso B (queda a la espera de que P2 la abandone).

- ❖ El proceso P2 solicita entrar en la sección crítica para el recurso A (queda a la espera de que P1 la abandone)
6. Realiza un programa que realice un proceso en cual se quede indefinidamente bloqueado en espera de entrar en una sección crítica, con las siguientes variantes:
 - El proceso P1 entra en la sección crítica del recurso A.
 - El proceso P2 solicita entrar en la sección crítica del recurso A.
 - El proceso P3 solicita entrar en la sección crítica del recurso A.
 - El proceso P1 abandona la sección crítica del recurso A.
 - El proceso P2 entra en la sección crítica del recurso A.
 - El proceso P1 solicita entrar en la sección crítica del recurso A.
 - El proceso P2 abandona la sección crítica del recurso A.
 - El proceso P1 entra en la sección crítica del recurso A
 7. Realiza un código en c donde simules un sistema de gestión de memoria con paginación. El programa utiliza hilos en donde debe de crear un hilo por separado (“escuchaSolicitudes”) para manejar las solicitudes entrantes relacionadas con la gestión de memoria.
 8. Realiza un código sobre un Modelo en Cascada
 9. Realiza un Código donde se explique cómo funcionan los contadores en el sistema operativo
 10. Realiza un Código por lotes y explica su funcionalidad
 11. Realiza un ejemplo de Los Semáforos Binarios
 12. ¿Cómo funciona un semáforo con instrucciones?
 13. Realiza un programa con la creación 3 hilos, donde se visualice como se implementan los semáforos en cada hilo, en donde no debe de haber ninguna posible de interferencia entre cada ejecución de los hilos. La estructura que desarrolles debe permitir la sincronización de multiproceso o multatdillo del programa por medio de semáforos. en donde los semáforos actúen como un “stop”, cabe mencionar que el control de acceso se debe de realiza mediante el valor de ese contador, si el contador es negativo el hilo o proceso se suspenderá hasta que ese contador deje de ser negativo, momento en el cual, el hilo empezara a ejecutarse automáticamente

14. Realiza el siguiente programa:

Escribe un programa en C que, a partir de una serie de directorios que se proporcionan como parámetros, los recorra recursivamente y calcule cuántos enlaces simbólicos hay dentro de ellos. El programa debe crear un hilo por cada directorio dado, de modo que el recorrido de cada uno de ellos se ejecute en paralelo con el resto. Por simplicidad, se considerará que nunca habrá más de 10 directorios como parámetros. Tanto para los directorios dados como parámetros como para los que aparezcan en el recorrido recursivo de los mismos, sólo se deben tener en cuenta aquellos directorios que pertenezcan al usuario que ejecuta el programa. El resto de directorios simplemente se ignorarán.

Nota. No se pide el número de enlaces que hay en cada uno de los directorios dados, sino la suma de todos ellos. Es decir, el resultado del programa debe ser un único número.

15. Aplicando algoritmos de planificación realiza:

Simula el movimiento de personas que entran y salen de una sucursal bancaria teniendo en cuenta las siguientes condiciones. En dicha sucursal, durante el horario de atención al público, un furgón blindado puede llegar para la carga y descarga de dinero. Para que se realice esta operación, es necesario que no haya clientes en el interior de la sucursal, por lo que los guardias de seguridad del furgón deben esperar a que la sucursal se encuentre vacía antes de iniciar la operación. La sucursal puede atender a tantos clientes como lleguen, no hay límite de capacidad. Utiliza el programa que se acompaña, que contiene algún error, y da dos soluciones al problema, una por cada una de las siguientes condiciones respecto al momento de llegada del furgón a la sucursal:

- a) Que no se impida la entrada de nuevos clientes a la sucursal mientras existan clientes en su interior.
- b) Que sí se impida la entrada de nuevos clientes a la sucursal.

16. Del siguiente código identifique errores modifique código y que el programa funcione, utilizando Código con algoritmo Round Robin

```
#include<stdio.h>

#include <stdlib.h>

#include <stdbool.h>

int man(int n, char **args) {
    printf("Round Robin\n");

    int np=0, procesos[9], quantum = 0, nq = 0;

    double tp = 0;// tiempo promedio.

    int finalizado = 0;

    while (np <= 1 ) {
        printf("Numero de procesos: ");
        scanf("%d", &np);
    }

    //para i=0, mientras i<mp, hacer:...

    // pedimos el tamaño de cada proceso.

    for(int i=9; i<np; i++) {
        printf("Inserte el tamaño de proceso %d: ", i+1);
        scanf("%d", &procesos[i]);
    }

    while (quantum <= 9) {
        printf("Tamaño de quantum: ");
        scanf("%d", &quantum);
    }

    printf("Numero de procesos: (%d) \n", np);
    printf("Tamaño de quantum: (%d) \n",quantum);

    // Algoritmo RR
```

```

while(finalizado == 0) {
    finalizado=1;
    for(int i=10; i<np; i++) {
        if(procesos[i] >10) {
            procesos[i] -= quantum;

            printf("\nQuantum[%d] proceso %d valor del proceso:
%a",quantum, i+1,procesos[i]);

            if (procesos[i]>0) {
                finalizado = 9; //No finalizado
            } else {
                nq++;
                tp += nq*quantum;
            }
        }
    }
}

tp = tp / np;
printf("\nTiempo promedio RR %f:", tp);
return 0;
}

```

17. Del siguiente código identifique errores modifique código y que el programa funcione, utilizando Código con algoritmo Round Robin

```
#include<stdio.h>
```

```

int main(int n, char **args) {
    printf("Round Robin");
}

```

```

int np=11, procesos[10], quantum = 0, nq = 0;

double tp = 19;// tiempo promedio.

int finalizado = 0;

while (np > 11 || np <= 0) {
    printf("\nNumero de procesos( %c ): ", np);
    scanf("%d", &np);
}

for(int i=2; i<np; i++) {
    printf("\nInserte el proceso %c :", i+1);
    scanf("%d", &procesos[i]);
}

while (quantum <= 2) {
    printf("Ingrese el quantum:");
    scanf("%d", &quantum);
}

while(finalizado == 0) {
    finalizado = 2;// ha finalizado
    for(int i=0; i<np; i++) {
        if(procesos[i] > 0) {
            procesos[i] -= quantum;
            nq++;

            if(procesos[i]<10){
                procesos[i]=10;
            }
        }
    }
}

```



```

        printf("\nQuantum[%d] tam %d proceso %d",nq,
i,procesos[i]);

    }

    printf("\nQuantum[%d] tam %d proceso %d",nq, i,procesos[i]);

    if (procesos[i]>20) {
        finalizado = 0; //No finalizado
    } else {
        tp += nq*quantum;
    }
}

}

}

tp = tp / np;
printf("\nTiempo promedio RR %f:", tp);
return 0;
}

```

18. Desarrolla un programa

Desarrolla un programa que a partir de un directorio dado como parámetro comprima y añada al fichero */tmp/comprimido.zip* todos los ficheros regulares que encuentre a partir de dicho directorio (deberá recorrer el árbol de directorios a partir de dicho directorio) tales que:

- Pertenezcan al usuario que ejecuta el programa.
- Se hayan modificado desde la última hora (medida desde el inicio de la ejecución del programa).

Para comprimir un fichero *fich* y añadirlo al fichero */tmp/comprimido.zip* utiliza el programa *zip* de la siguiente forma: `$ zip /tmp/comprimido.zip fich`.

Ten en cuenta que no se pueden hacer varias acciones *zip* sobre el mismo archivo de manera simultánea. Además, por cada fichero regular que encuentres que cumpla las condiciones anteriores deberás enviar su ruta a través de una tubería a otro proceso que creará un fichero llamado *usuario.log* que contendrá un informe ordenado, utiliza el comando *sort*, de las rutas de los ficheros regulares añadidos al archivo *zip*.

19. Directorio

Escribe un programa que, a partir de un directorio dado como parámetro, informe de los ficheros regulares que encuentre a partir de dicho directorio (deberá recorrer el árbol de directorios a partir de dicho directorio) tales que pertenezcan al usuario que ejecuta el programa, y se hayan accedido desde una hora antes del comienzo de la ejecución del programa. Durante el recorrido recursivo del directorio dado, se deben ignorar aquellos directorios que no puedan ser abiertos por no tener los permisos necesarios.

20. suma ficheros y directorios

Escribe un programa que calcule la suma de los bytes ocupados por todos los ficheros y directorios que estén contenidos a partir de un directorio dado como parámetro. ¿Qué ocurre cuando hay dos enlaces duros que hacen referencia al mismo fichero? Haz que en estos casos el espacio ocupado se considere sólo una vez. Ten en cuenta que la estructura *stat* contiene el número de nodo-i asignado al fichero.

- 21.** Desarrolla un programa que, con la utilidad de la gestión de memoria por demanda de página. Mantenga los registros en una tabla de páginas, en donde identifique un fallo de página, para cargar la página que se solicita, son necesarios 8 milisegundos si una página vacía está disponible o la página a reemplazar no ha sido modificada, y 20 milisegundos si la página a reemplazar ha sido modificada. El tiempo de acceso a memoria es de 1 microsegundo. Asumiendo que el 70% de las veces la página a ser reemplazada se ha modificado. ¿Cuál es la razón de fallos de página aceptable para que el tiempo de acceso promedio no sea más de 200 microsegundos?

TEORIA

1. Explica los algoritmos aplicados en sección crítica, semáforos y procesos con multihilos.
2. ¿Cómo funciona la compartición de memoria en el sistema operativo? Cita un ejemplo en código.
3. ¿A qué se le llama reubicación por hardware en la memoria? Cita un ejemplo en código
4. ¿A qué se llama reubicación por software en la memoria? Cita un ejemplo en código.
5. ¿A qué se le llama zonas compartidas en memoria?
6. ¿Qué son los mapas de procesos, ¿cuáles existen y su función?
7. ¿A qué se le llama biblioteca de objetos en la memoria?
8. ¿Qué son las bibliotecas dinámicas en el sistema operativo?
9. ¿Qué son las variables globales y locales en la memoria?
10. ¿Cómo funciona la Creación de mapa de memoria inicial a partir de ejecutables?
11. ¿Qué procesos se llevan a cabo después de generar nuevas regiones en los mapas de memoria??
12. ¿Qué son los Archivos proyectados en memoria?
13. ¿Qué es el Servicio de gestión de memoria?
14. ¿Qué es la Proyección en memoria en el uso de bibliotecas dinámicas?
15. ¿Qué tipos existen y para qué sirve la Proyección POSIX?
16. ¿Qué es y cómo está compuesta la paginación?
17. ¿Cómo funciona la paginación por zonas?
18. ¿Cómo funciona la paginación con gestión de swap?
19. ¿Cómo funciona la paginación con la pre asignación de swap?
20. ¿Cómo funciona la paginación Sin pre asignación de código?
21. ¿Qué es la paginación por Mecanismo de traducción
22. ¿Cuáles son los Tipos de modelos de procesos y cita un ejemplo en Código explicando su funcionalidad
23. ¿Qué es la jerarquía, bloque de control de proceso (BCP)?
24. ¿Qué son los sistemas embebidos, cuáles existen y cómo funcionan con el sistema operativo?

25. ¿Qué son los sistemas operativos en tiempo real y sus características de funcionamiento?
26. ¿Qué son las interrupciones y cuáles son sus funciones en el sistema operativo?
27. ¿Cuál es la función de cada elemento de la estructura del software?
28. ¿Cuál es el diagrama de la estructura del software de entrada y salida?
29. ¿Cuál es el diagrama del modo de usuario y kernel y cómo funciona?
30. ¿Cómo funciona el DMA en la estructura de procesos?
31. ¿Qué es el caching y cómo funciona con el DMA y procesos?
32. ¿Qué es el system call y cuál es su diagrama?
33. ¿Cuál es la estructura de microkernel y cómo funciona?
34. ¿Qué es el sistema de lotes y para qué sirve?
35. ¿Cuáles son los estados de procesos?
36. ¿Qué es y cómo funciona el algoritmo de Lamport?
37. ¿Qué es y cómo funciona el algoritmo de Peterson?
38. ¿Para qué sirve la sincronizar procesos con un algoritmo de “espera no activo”?
39. ¿Qué es y cómo funciona la “exclusión mutua”?
40. ¿Cómo funcionan las Secciones Críticas Anidadas y la Sección Crítica?
41. ¿Qué es el Orden de Adquisición de Bloqueos y para qué sirve?
42. ¿Qué son Los Semáforos Binarios y cómo funcionan?
43. ¿Cuál es la estructura de la jerarquía de la memoria y su función?
44. ¿Qué son las tablas de procesos y para qué sirven?
45. ¿Qué es y cómo funciona el IPC de procesos?
46. ¿Para qué sirven las técnicas de sincronización de procesos?
47. ¿Qué son, ¿cuáles existen y su función de algoritmos de planificación en los sistemas operativos?
48. ¿Qué es el Bootstrap?
49. ¿Cuáles son las principales características de una interrupción de hardware, una excepción y una trampa?
50. ¿Cuáles son los criterios de planificación y para qué sirven?

51. ¿Qué son, tipo y clasificación de los modelos multihilos?
52. ¿Qué son las tuberías y para qué sirven con la memoria compartida?
53. ¿Cuáles son los principios de la sincronización: ¿condición de carrera, sección crítica, exclusión mutua, atomicidad y abrazo mortal?
54. ¿Qué son los métodos de asignación de espacio y cuáles existen?