Examen final Sistemas Operativos 07/02/2023

Cadena de supermercados (2.5 puntos)

Una cadena de supermercados tiene sucursales en todo el país. Debido a que la actualización de precios es una tarea propensa a errores, la delega a cada proveedor. A su vez, cada computadora de cada sucursal actualiza su lista de precios localmente al iniciar.

A continuación se presentan las funciones que ejecutan las sucursales y los proveedores.

```
1.    sem_t sem_a = 1; // Asumimos que set_t funciona como un int
2.    sem_t sem_b = 1;
3.    int sc = 0;
4.
5.    void sucursal() {
6.      while (1) {
7.        sem_wait(&sem_a);
8.        if(++sc == 1) sem_wait(&sem_b);
9.        sem_post(&sem_a);
10.
11.       consultar_precios_en_casa_central();
12.
13.       sem_wait(&sem_a);
14.       if(--sc == 0) sem_post(&sem_b);
15.       sem_post(&sem_a);
16.
17.       actualizar_precios_localmente();
18.     }
19.   }
20.
21.   void proveedor() {
22.     while (1) {
23.       calcular_aumento_de_precios();
24.
25.       sem_wait(&sem_b);
26.
27.       actualizar_precios_en_casa_central();
28.
29.       sem_post(&sem_b);
30.     }
31.   }
```

Desde que se usa este sistema, los proveedores reportan demoras de varios minutos para poder actualizar los precios, sobre todo en el horario de apertura de las sucursales.

Explique por qué sucede esto y modifique la solución presentada para evitar estas demoras a los proveedores.

Shell (1.5 puntos)

Un desarrollador realiza varias modificaciones al kernel de Linux y al compilar nota que hay muchos warnings, así que decide armar una lista de los mismos para analizarlos uno a uno. Sin embargo, el compilador suele repetir el mismo warning una y otra vez. Para obtener una lista de los warnings sin repetidos, utiliza el comando ***nauniq***, el cual es similar a ***uniq***, solo que detecta líneas repetidas incluso si no son adyacentes (*). A continuación se presenta el script ***bash*** que ejecuta.

```
make 2>&1 > output
grep “warning” output > warnings
rm output
nauniq warnings
rm warnings
```

Los warnings usualmente se envían a stderr por este motivo redirige stderr a stdout: ***2>&1***, para poder guardar todo el output en el archivo ***output***. Luego filtra las líneas que contengan la palabra “warning” y las guarda en el archivo ***warnings***. Finalmente, obtiene la lista de warnings sin repetidos. Además, elimina los archivos temporales utilizados.

La compilación del kernel de Linux puede demorar varios minutos, incluso horas, dependiendo de las especificaciones del hardware. Con este script el desarrollador debe esperar a que la compilación termine para poder empezar a analizar los warnings o resolver alguno más sencillo, si existiera.

Sugiera un script que resuelva este problema utilizando los mismos programas y enumere todas las ventajas que presenta su solución respecto a la de este desarrollador.

(*)

Entrada	Salida de ***uniq***	Salida de ***nauniq***
A B B A C	A B A C	A B C

Server (1.5 puntos)

El siguiente es un fragmento de código de un servidor que recibe información de 2 sockets diferentes y la procesa. Identifique y explique los problemas que presenta esta implementación. Modifique el código con las soluciones a estos problemas.

```
1.    void server (){
2.
3.      // ...
4.
5.      while (1){
6.        memset(buf1, 0, sizeof(buf1));
7.        read(socket1_fd, buf1, 1);
8.        procesar_informacion(buf1, sizeof(buf1))
9.
10.       memset(buf2, 0, sizeof(buf2));
11.       read(socket2_fd, buf2, 1);
12.       procesar_informacion(buf2, sizeof(buf2))
13.     }
14.
15.     // ...
16.
17.   }
```

Semáforos (2.5 puntos)

La siguiente es una implementación simplificada de semáforos que utiliza la instrucción en assembler ***xchg*** dentro de la función ***_xchg***, la cual intercambia los valores de ***sem_value*** y ***new_value*** atómicamente y retorna el valor de ***sem_value*** previo al intercambio.

```
1.    extern int _xchg(int *sem_value, int new_value);
2.
3.    void sem_wait(sem *s){
4.      while(_xchg(&(s->value), s->value - 1) <= 0)
5.        _xchg(&(s->value), s->value + 1);
6.    }
7.
8.    void sem_post(sem *s){
9.      _xchg(&(s->value), s->value + 1);
10.   }
```

Determine si esta solución es correcta. En caso de considerarla incorrecta de un ejemplo que ilustre el problema y sugiera una solución. Para esto puede crear funciones y variables auxiliares, pero deberá utilizar la función ***_xchg*** provista. La espera activa se asume correcta.

TLB (2 puntos)

Considere que el siguiente programa se ejecuta en una computadora con un tamaño de página de 2^{12} bytes y un TLB de 2^{6} entradas.

```
1. int arr[ARR_SIZE];
2. int i = 0;
3. while(i < ARR_SIZE){
4.   arr[i] = arr[i] + 1;
5.   i += I_INCREMENT;
6. }
```

Indique qué valores de ARR_SIZE y I_INCREMENT causarían un *miss* en el TLB por cada iteración del ciclo. Justifique.

Si este ciclo se repitiera múltiples veces, ¿su respuesta seguiría siendo la misma? Justifique