dce
ACTIVIDADES T2

PROBLEMA 2.1 Se sabe que la sobrecarga (overhead) de un monitor software sobre un computador es del 4 %. Si el monitor se activa cada 2 segundos, ¿cuánto tiempo tarda el monitor en ejecutarse por cada activación?

$$sobrecarga = \frac{x}{2s} = 4\% \rightarrow x = 0.08s\ (80\ milisegundos)$$

PROBLEMA 2.2 En un sistema Linux se ha ejecutado la orden uptime tres veces en momentos diferentes. El resultado, de forma resumida, es el siguiente:

```
... load average:  6.85,  7.37, 7.83
... load average:  8.50, 10.93, 8.61
... load average: 37.34,  9.47, 3.30
```

Indique si la carga crece, decrece, se mantiene estacionaria o bien no puede decidir sobre ello

No se puede decidir sobre la evolución porque no hay una tendencia clara en los valores de las medidas

PROBLEMA 2.3 En un sistema Linux se ha ejecutado la siguiente orden:

```
$ time quicksort
real 0m40.2s
user 0m17.1s
sys  0m3.2s
```

Indique si el sistema está soportando mucha o poca carga. Razone la respuesta.

El tiempo de respuesta es el real (usado x el sistema). Está soportando mucha carga porque t_{espera} = 40,2 - (17,1 + 3.2) = 19,7s, mucho más de lo que usa el usuario.

PROBLEMA 2.4 Después de conectarse a un sistema informático, un usuario ejecuta las dos órdenes siguientes con el resultado que se muestra:

```
% uptime
  9:50am  up 173 days, 23:02,  1 user,  load average: 0.00, 0.00, 0.00
% time simulador
real    8m0.70s
user    3m5.20s
sys     0m4.01s
```

1. ¿En qué condición de carga se encuentra el computador (baja, media o alta) en el momento de conexión del usuario?

Baja (0)

2. ¿Cuál es el tiempo de ejecución del programa simulador?

real = 480,7

3. ¿Encuentra alguna incoherencia en los resultados anteriores? Justifique la respuesta con argumentos sólidos.

Si, si la carga es baja (0), ¿cómo es posible que esté en espera tantos segundos? (480,7 – (185,2 + 4,01) = 291,49s)

PROBLEMA 2.5 Indíquese una orden (u órdenes) que se podría emplear para monitorizar los aspectos siguientes de la actividad en un computador que trabaja con el sistema operativo Linux:

1. Capacidad de memoria física ocupada por un proceso.	top
2. Número de cambios de contexto por segundo.	vmstat | sar
3. Carga media del sistema.	uptime | sar
4. Número de interrupciones por segundo.	vmstat | sar
5. Capacidad libre de la unidad de disco magnético.	df
6. Usuarios conectados a la máquina.	who
7. Utilización del procesador en modo usuario.	top | vmstat | sar
8. Tiempo que lleva ejecutándose un proceso.	top | ps
9. Tiempo que tarda un proceso en ejecutarse.	time

PROBLEMA 2.6 Considere las órdenes siguientes ejecutadas en un sistema Linux:

```
$ time simulador_original
real 0m24.2s
user 0m15.1s
sys  0m1.6s
```

```
$ time simulador_mejorado
real 0m32.8s
user 0m10.7s
sys  0m2.1s
```

1. ¿Cuál es el tiempo de ejecución de ambos simuladores?
2. Calcule, si es el caso, la mejora en el tiempo de ejecución del simulador mejorado respecto del original.

PROBLEMA 2.7 Un informático desea evaluar el rendimiento de un computador por medio del benchmark SPEC CPU2006. Una vez compilados todos los programas del paquete y lanzado su ejecución monitoriza el sistema con la orden vmstat 1 5. El resultado de las medidas de este monitor es el siguiente:

```
procs -----------memory---------- ---swap-- -----io---- ---system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs  us sy id wa
 0  0      8  14916  92292 833828    0    0     0     3    0    7   3  1 96  0
 1  0      8  14916  92292 833828    0    0     0     0 1022   40 100  0  0  0
 3  0      8  14916  92292 833828    2    1    16     3 1016   34  99  1  0  0
 1  0      8  14916  92292 833828    0    4     0     8 1035   36  98  2  0  0
 2  0      8  14916  92292 833828    1    5     4    28 1035   36  99  1  0  0
```

Indique si, a la vista de los datos anteriores, los resultados obtenidos en la prueba evaluación serán correctos o no. Justifique la respuesta.

Los resultados serían incorrectos porque el sistema operativo presenta actividad de intercambio con el disco magnético.

PROBLEMA 2.8 Considere la siguiente secuencia de órdenes en un sistema informático donde está instalado el monitor sar:

```
% sadc resultado
% ls -l resultado
-rw-r--r--  1 usuario grupo 712 2006-01-16 10:55 resultado
```

1. Indique qué contiene el fichero resultado y cómo se codifica esta información.

Los datos estadísticos y un registro en formato binario (sadc)

2. ¿Qué orden habría que emplear para visualizar en formato ASCII toda la información capturada por el monitor de actividad en la activación anterior?

Orden sar -A f resultado

3. Si el monitor sar está instalado en la máquina para ejecutarse cada tres minutos y disponemos de 50 MB de espacio en el disco duro para almacenar la información de actividad, ¿cuántos ficheros históricos diarios podremos mantener en esta instalación?

El tamaño del archivo es 712B. Si el monitor sar se ejecuta cada 3 minutos, 1440 / 3 = 480 ejecuciones por día. Si cada ejecución genera un archivo de 712B, el tamaño total de los archivos al día será 0,342MB al día. Como el tamaño del disco es 50MB, dividimos éste por el tamaño diario diario de cada archivo: 50 / 0,342 = 146 archivos.

PROBLEMA 2.9 El monitor sar (system activity reporter) de un computador se activa cada 15 minutos y tarda 750 ms en ejecutarse por cada activación. Se pide:

1. Calcular la sobrecarga que genera este monitor sobre el sistema informático.

$$sobrecarga = \frac{0{,}750}{15 * 60} = 0{,}083\%$$

2. Si la información generada en cada activación ocupa 8192 bytes, ¿cuántos ficheros históricos del tipo saDD se pueden almacenar en el directorio /var/log/sa si se dispone únicamente de 200 MB de capacidad libre?

1440 / 15 = 96 ficheros históricos al día. 96 * 8192 = 0,75MB por día.

200MB / 0,75 Mbdia = 266 ficheros.

PROBLEMA 2.10 El día 8 de octubre se ha ejecutado la siguiente orden en un sistema Linux:

```
% ls /var/log/sar
-rw-r--r--   1 root     root      3049952 Oct  6 23:50 sa06
-rw-r--r--   1 root     root      3049952 Oct  7 23:50 sa07
-rw-r--r--   1 root     root      2372320 Oct  8 18:40 sa08
```

¿Cada cuánto tiempo se activa el monitor sar instalado en el sistema? ¿Cuánto ocupa el registro de información almacenada cada vez que se activa el monitor?

Cada 10 minutos, ya que la última llamada se hace a las 23:50. Si se producen activaciones cada 10 minutos, eso significa que al día se van a dar 144. Dividimos lo que ocupa el fichero histórico entre las activaciones para obtener el tamaño por activación: 3049952 / 144 = 21180,2 = 21KB.

PROBLEMA 2.11 Indique el resultado que produce la ejecución de las siguientes órdenes sobre un sistema Linux con el monitor sar instalado:

1. sar

Utilización del procesador en el día actual en que se está consultando.

2. sar -A

Toda la información en el día actual.

3. sar -u 1 30

Utilización del procesador. 30 hace referencia al nº de muestras que quieren ser tomadas y 1 el intervalo entre ellas en segundos.

4. sar -uB -f /var/log/sa/08

Utilización del procesador y paginación de la MV en el día 8 del mes.

5. sar -d -s 12:30:00 -e 18:15:00 -f /var/log/sa/08

Transferencias para cada disco desde las 12:30 y 18:15 del día 8.

6. sadc

recoge los datos estadíscos y construye un registro en formato binario (back-end)

7. sadc 2 4

Lo mismo de antes pero tomando 4 muestras en intervalos de 2s.

8. sadc 2 4 fichero

Lo mismo de antes pero almacenando los resultados en el fichero llamado *fichero*.

PROBLEMA 2.12 Indique cómo se lleva a cabo la instrumentalización de un programa para que sea posible monitorizarlo utilizando la herramienta gprof. ¿Dónde se almacena la información recogida por la monitorización? ¿Cómo se puede visualizar en un formato legible?

La instrumentación del programa se hace durante su compilación. Típicamente se usa el parámetro -pg en la orden de compilación: gprof programa.c -o programa -pg. La información recogida por el monitor durante la ejecución del programa se guarda en el fichero gmon.out. Finalmente, para ver en pantalla los resultados ofrecidos por la ejecución del programa se ejecuta la orden gprof programa.

PROBLEMA 2.13 Después de instrumentar un programa con la herramienta gprof el resultado obtenido ha sido el siguiente:

```
Flat profile:
Each sample counts as 0.01 seconds.

  %   cumulative   self              self     total
 time   seconds   seconds    calls  s/call   s/call  name
 59.36     27.72    27.72        3     9.24    14.39  reduce
 33.08     43.17    15.45        6     2.57     2.57  invierte
  7.56     46.70     3.53        2     1.76     1.76  calcula
```

El grafo de dependencias muestra que invierte() es llamado desde el procedimiento reduce().

1. ¿Cuánto tarda en ejecutarse el código propio del procedimiento reduce()? 9,24s

2. ¿Cuál es el procedimiento más lento del programa? ¿Y el más rápido?

Más lento: reduce, más rápido: calcula.

3. Si el procedimiento más lento de todos se sustituye por otro tres veces más rápido, ¿cuánto tiempo tardará en ejecutarse el programa?

$$T_{mejorado} = \frac{27{,}72}{3} + 15{,}45 + 3{,}53 = 28{,}22s$$

4. Si el procedimiento invierte() se sustituye por una nueva versión cuatro veces más rápida, ¿qué mejora se obtendrá en el tiempo de ejecución?

$$T_{mejorado} = 27{,}72 + \frac{15{,}45}{4} + 3{,}53 = 35{,}11s$$

$$A = \frac{46{,}70}{35{,}11} = 1{,}33 = 33\%\ más\ rápido$$

5. Calcule cuál es la aceleración máxima que se podría conseguir en el tiempo de ejecución mediante la optimización del código del procedimiento invierte().

Optimización de invierte = tiempo=0.

$$T_{mejorado} = 27{,}72 + 0 + 3{,}53 = 31{,}25s$$

$$A = \frac{46{,}70}{31{,}25} = 1{,}49 = 49\%\ más\ rápido$$

PROBLEMA 2.14 La monitorización de un programa de dibujo en tres dimensiones mediante la herramienta gprof ha proporcionado la siguiente información (por errores en la trasmisión hay valores que no están disponibles):

```
Flat profile:

  %    cumulative     self              self    total
 time    seconds   seconds  calls   s/call   s/call  name
xxxxx     xxxxx     15.47      3     5.16     5.16  colorea
xxxxx     xxxxx      1.89      5     0.38     0.38  interpola
xxxxx     xxxxx      1.76      1     1.76     3.65  traza
xxxxx     xxxxx      0.46                          main

Call graph:

index  % time    self  children     called     name
[1]    100.0    0.46    19.12                 main      [1]
                15.47     0.00        3/3       colorea   [2]
                 1.76     1.89        1/1       traza     [3]
-----------------------------------------------------------
                15.47     0.00        3/3       main      [1]
[2]     79.0   15.47     0.00          3       colorea   [2]
-----------------------------------------------------------
                 1.76     1.89        1/1       main      [1]
[3]     18.6    1.76     1.89          1       traza     [3]
                 1.89     0.00        5/5       interpola [4]
-----------------------------------------------------------
                 1.89     0.00        5/5       traza     [3]
[4]      9.7    1.89     0.00          5       interpola [4]
-----------------------------------------------------------
```

1. ¿En cuánto tiempo se ejecuta el programa de dibujo?

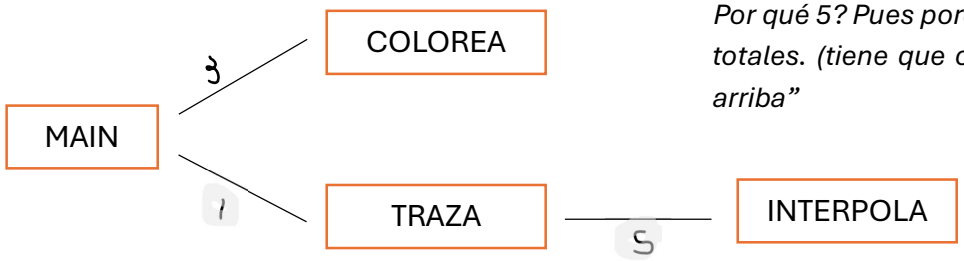
15,47 + 1,89 + 1,76 + 0,46 = 19,58s

0,46 + 19,12 = 19,58s *(se pueden hacer ambos y tienen que dar lo mismo obv)*

2. Indique cuánto tiempo tarda en ejecutarse el código propio de main().

0,46s

3. Establezca la relación de llamadas entre los procedimientos del programa así como el número de veces que se ejecuta cada uno de ellos.



*Por qué 5? Pues porque 5 * 1 = 5 y son las llamadas totales. (tiene que concordar al multiplicar "hacia arriba"*

4. Calcule el nuevo tiempo de ejecución del programa si se elimina el código propio de main() y se reduce a la mitad el tiempo de ejecución del código propio del procedimiento traza().

$$T_{mejorado} = 15{,}47 + 1{,}89 + \frac{1{,}76}{2} + 0 = 18{,}245s$$

5. Proponga y justifique numéricamente una acción sobre el programa original que no afecte el procedimiento colorea() (ni su código ni el número de veces que es ejecutado) con el fin de conseguir que el programa se ejecute en 10 segundos.

No se puede porque colorea ya son +10 segundos. Para poder llegar a 10s se debería modificar colorea.

PROBLEMA 2.15 El resultado de la monitorización de una aplicación informática dedicada al análisis de modelos atmosféricos se muestra a continuación (nótese que hay información no disponible):

Flat profile:

% time	cumulative seconds	self seconds	calls	self s/call	total s/call	name
xxxxx	xxxxx	30.16	52	0.58	0.58	nimbo
xxxxx	xxxxx	5.13	2	2.56	2.56	borrasca
xxxxx	xxxxx	3.51	2	1.75	1.75	lluvia
xxxxx	xxxxx	1.76	1	1.76	34.17	nube

1. Indique cuánto tiempo tarda en ejecutarse el programa.

30,16 + 5,13 + 3,51 + 1,76 = 40,56s

2. Determine el porcentaje del tiempo de ejecución que consume el procedimiento lluvia().

Regla de tres:

40, 56 → 100

3,51 → ¿?? → 351/40,56 = 8,65%

3. ¿Cuál es el procedimiento más lento de todo el programa?

Nube (total s/call) o Borrasca (self s/call) → *en la solución nos dice esta última. Mejor hacer caso a lo que dicen los de la asignatura y cada vez que nos pregunten esto contestar el tiempo de self s/call.*

4. ¿Cuánto tiempo tarda en ejecutarse el código propio de borrasca()?

2,56s

5. Calcule el nuevo tiempo de ejecución del programa si el procedimiento nimbo() se rediseña y mejora 3 veces.

$$T_{mejorado} = \frac{30{,}16}{3} + 5{,}13 + 3{,}51 + 1{,}76 = 20{,}45s$$

6. Proponga y justifique numéricamente alguna manera de reducir el tiempo de ejecución del programa original hasta los 20 segundos.

Mejorando 4 veces el proceso nimbo se puede (ya que si con 3 hemos llegado a 20,45s, nos bastaría con hacerlo 1 vez más rápido para bajar de los 20s).

Una explicación más matemática:

$$T_{mejorado} = 20s = \frac{30{,}16}{x} + 5{,}13 + 3{,}51 + 1{,}76$$

$$x = 3{,}14 \rightarrow 4\ veces$$