SISTEMAS OPERATIVOS AVANZADOS (TEORÍA)[1]

VIERNES 4 FEBRERO 2022

PROCESOS (3 PUNTOS)

a) ¿En un servidor empleado en aplicaciones científicas (p.ej, modelando de clima) que algoritmo de planificación emplearías? ¿Y en un teléfono móvil?
b) ¿Por qué el *trap-frame* no se puede guardar en el Bloque de Control de Proceso y se ha de guardar en el *stack* de supervisor (u otra estructura flexible en memoria)?
c) ¿Cómo se evita que un planificador MLFQ los procesos con uso intenso de CPU sufran de inanición?
d) ¿El código de una llamada al sistema puede hacer uso de otra llamada al sistema?
e) En el planificador CFS, ¿Cuál es la utilidad del Red-black-tree?
f) ¿Qué ocurre si durante el código de una llamada al sistema finaliza la rodaja de tiempo en RR (es decir, llega una interrupción del *timer*)?

MEMORIA (3 PUNTOS)

a) Cuando se ejecuta el siguiente código en un sistema operativo actual, los valores de A, B, C y D cambian entre ejecución y ejecución (con el mismo ejecutable). ¿Por qué?

```
#include <stdio.h>
#include <stdlib.h>
int y=1;
int main(int argc, char *argv[])
{
   int x = 3*y;
   printf("location of A  : %p\n", (void *) &y);
   printf("location of B  : %p\n", (void *) main);
   printf("location of C  : %p\n", (void *) &x);
   printf("location of D  : %p\n", (void *) malloc(1));
   return x;
}
```

b) ¿Cuál es la diferencia entre un fallo de página y un fallo de TLB? ¿En cuál de ellos es interesante disponer de ayuda del hardware y por qué?
c) ¿En que condiciones un algoritmo de reemplazo de páginas aleatorio es mejor que uno aproximado LRU (como el del reloj)? ¿Cuándo se actualiza el bit de acceso a la página en la tabla de páginas para el algoritmo del reloj?
d) ¿Por qué no se puede emplear un *bitmap* en la gestión de espacio libre del *heap*?
e) En sistema de memoria que emplee segmentación (con código, *heap*, *stack*, etc…) ¿Quién y cuando decide el número de segmento que aparece en cada dirección virtual?
f) ¿Cómo se previene que se produzca un hit en el TLB si el marco que contiene la página implicada ha sido *swapeado* al disco?

(…)

[1] Todas las respuestas han de ser justificadas adecuadamente

CONCURRENCIA (2.5 PUNTOS)

a) ¿Qué consecuencia tendría que varios *threads* empleasen el mismo *stack* de *kernel*?

b) El siguiente código, correspondiente a la suma de dos vectores, no es *thread-safe* (pues puede dar lugar a *deadlock*). Proponer una implementación alternativa que si lo sea.

```
void vector_add(vector_t *v_dst, vector_t *v_src) {
   Pthread_mutex_lock(&v_dst->lock);
   Pthread_mutex_lock(&v_src->lock);
   int i;
   for (i = 0; i < VECTOR_SIZE; i++) {
      v_dst->values[i] = v_dst->values[i] + v_src->values[i];
   }
   Pthread_mutex_unlock(&v_dst->lock);
   Pthread_mutex_unlock(&v_src->lock);
}
```

a) La siguiente función corresponde a la ejecución de un *thread*. El resultado de dicho thread no se retorna correctamente. Proponer una corrección al problema/s.

```
void *mythread(void *arg) {
   myarg_t *m = (myarg_t *) arg;
   myret_t r;
   r.x = 1;
   r.y = 2;
   return (void *) &r;
}
```

b) ¿Por qué se requiere soporte hardware para implementar los *locks*? ¿Que aproximación usa x86?

c) ¿Por qué con una sola variable condicional no se puede resolver el problema del productor-consumidor?

PERSISTENCIA (1.5 PUNTOS)

a) ¿De qué tamaño tiene que ser el segmento en un sistema LFS para lograr al menos una tasa de utilización del 95% de al ancho de banda de un disco sata2 (300MB/s), si el tiempo de posicionamiento de las cabezas es de 10ms?

b) Suponiendo que el tamaño de bloque del sistema de ficheros es 4KB, ¿Cuanto es el tamaño de fichero máximo que soportaría si el sistema de ficheros es realmente simple, donde cada i-nodo tiene 10 punteros directos y 2 indirectos? Cada puntero tiene 32 bits (4 bytes).

c) Si un proceso crea un fichero y posteriormente realiza un *fork()*, ¿El hijo será capaz de ver los cambios que haga el padre en el fichero (después de haber realizado el *fork*)?

SISTEMAS OPERATIVOS AVANZADOS (PRÁCTICAS)

VIERNES 4 FEBRERO 2022

Para cada una de las prácticas hay que explicar que comprueba el test indicado y explicar de que modo lo resuelve **tú implementación** de la práctica (fichero/s y líneas de código). Indicar el *commit* que incluye el cambio (8 primeras cifras del hash). Ser recomienda resolver el examen empleado exclusivamente el interfaz de ***gitlab***.

PRACTICA 1

test/P1/ctest/procs5.c

PRÁCTICA 2

test/P1/ctest/sleep_high_priority.c
test/P1/ctest/getpinfo.c

PRÁCTICA 3

tests/P3/ctests/shmem_access_persistent.c
tests/P3/ctests/shmem_count_fork.c

PRÁCTICA 4

tests/P4/ctests/badclone.c
tests/P4/ctests/thread2.c

PRÁCTICA 5

tests/P5/ctests/write4.c