Final de Sistemas Operativos

28/07/2015

Nota:

Apellido y Nombre	*Profesor*	*Tomé conocimiento de la nota: (Sólo aplazos)*

Preguntas teóricas					**Ejercicios**	
1	**2**	**3**	**4**	**5**	**1**	**2**

A) Teoría: Explícitamente defina como VERDADERA o FALSA cada una de estas afirmaciones justificando brevemente.

Peter revisa la implementación de Memoria Virtual de un Sistema Operativo de código abierto, y descubre que utiliza una variante del algoritmo Clock Mejorado y una partición de swapp. Luego de varios días de trabajo, implementa Journaling sobre la partición y ofrece sus cambios a la comunidad, pero estos son rechazados con la siguiente respuesta:

1) Una partición de swapp no necesita mecanismos que garanticen la integridad de los datos

Algo deprimido, continúa revisando el FileSystem (de tipo EXT2) que se está utilizando. Para su sorpresa, descubre que dentro de los inodos ningún campo hace referencia al nombre. Después de algunas horas de trabajo, lo agrega y ofrece nuevamente sus cambios, pero también son rechazados con la siguiente respuesta:

2) El SO no usa el nombre del archivo para identificarlo unívocamente en el sistema

Al realizar una prueba de carga, con una cantidad alta de procesos, la PC que utiliza Peter se vuelve bastante lenta. Siguiendo los consejos de su madre (quién no sabe mucho del tema), le duplica la memoria. Para sorpresa de todos, esto mejora notablemente el funcionamiento, ante el mismo conjunto de procesos.

3) Cuando la cantidad de procesos en ejecución es alta, la cantidad de memoria disponible puede afectar el rendimiento del Sistema Operativo

Al finalizar la prueba, revisa los resultados y nota que los procesos IO bound tardaron mucho más de lo esperado en ejecutar sus ráfagas de CPU. El algoritmo utilizado es Virtual Round Robin.

4) Este resultado puede darse sólo si el quantum utilizado es demasiado grande

Por último, revisando otro set de procesos, se da cuenta de que las entradas-salidas nunca bloquearon a los procesos.

5) Las operaciones de lectura y escritura de este sistema se están realizando siempre de forma no bloqueante

B) Práctica: Resuelva los ejercicios justificando las respuestas

1) Uno de los procesos que se ejecuta en la PC utiliza el siguiente código. Al correrlo, nota algunos problemas, si se le asignan 3 frames, con asignación fija y sustitución local, y el algoritmo de sustitución LRU.

```
int GLOBAL = 10;
int main (int argc, char *argv[]) {
   int suma = sumar(GLOBAL, 1);
   GLOBAL = suma;
   printf(suma);
}
int sumar(int op1, int op2) {
   return op1 + op2;
}
```

a. Sabiendo que este Sistema coloca en diferentes páginas (una por cada elemento de su imagen) el stack, el heap, la sección de datos, el código del proceso y las bibliotecas que hacen llamadas al sistema, **indique el estado de los frames** luego de ejecutarlo, sabiendo que siempre se carga primero la instrucción y luego los operandos.
Tip: descomponga cada sentencia en páginas

b. **Reordene las sentencias** de código del proceso para que ocasione menos fallos de página, indicando que problemas/puntos de mejora encontró.

2) En otra prueba, se ejecuta nuevamente el código anterior, pero modificando la función main para que soporte múltiples KLTs. La función crear_hilos(N) crea N hilos (KLTs) que correrán en código posterior, y esperar_hilos() bloquea al hilo KLT en ejecución, hasta que todos los otros se bloqueen y así terminen.

```
int main (int argc, char *argv[]) {
   crear_hilos(5);
   int suma = sumar(GLOBAL, 1);
   GLOBAL = suma;
   printf(suma);
   esperar_hilos();
}
```

a. Sincronice correctamente la ejecución utilizando semáforos sólo donde sea necesario, evitando inconsistencias.

b. Indique si es posible que los procesos queden en deadlock.

c. ¿Qué sucedería con la ejecución si fuesen ULTs?

El tiempo de duración del examen final será de 90’ a contar desde el momento de comienzo del mismo. Si el alumno por algún motivo comenzara más tarde sólo podrá utilizar el tiempo remanente. Utilice hojas separadas para la teoría / ejercicios.

Final de Sistemas Operativos

28/07/2015

Nota:

Teoría:

1) Verdadero, no necesita integridad porque la información volcada sólo se utiliza sólo cuando el SO está activo.

2) Verdadero, usa el número de inodo

3) Verdadero, se puede caer en thrashing, y esto se soluciona con más memoria

4) Falso, si el quantum es demasiado chico, los procesos IO bound volverán demasiado rápido a la cola de menor prioridad, con un resultado similar.

5) Falso. Podrían también ser asíncronas ó ser bloqueantes pero no darse las condiciones para el bloqueo (por ejemplo estando disponibles los dispositivos).

Practica:

***1. a)** Lo primero es dividir el proceso en páginas. Llamaremos a las páginas según sus iniciales S, H, D, C y B. El heap nunca se usa. La instrucción siempre se pide primero y puede estar en la página de código o en la de sistema.*

```
int GLOBAL = 10; → C y D
int main (int argc, char *argv[]) { → C y S
    int suma = sumar(GLOBAL, 1); → C,D y S
    GLOBAL = suma; → C, S y D
    printf(suma); → C, S y B
}
int sumar(int op1, int op2) {
    return op1 + op2;
}
```

Se ejecuta en orden y tenemos los siguientes pedidos (el cambio de color muestra el cambio de línea en la ejecución)

C	D	C	S	C	D	S	C	S	D	C	S	B
C	C	C	C	C	C	C	C	C	C	C	C	B
-	D	D	D	D	D	D	D	D	D	D	D	D
-	-	-	S	S	S	S	S	S	S	S	S	S
FP	FP	-	FP	-	-	-	-	-	-	-	-	FP

***1. b)** El código, el stack y la biblioteca no pueden eliminarse. Pero sí la página de datos, haciendo que GLOBAL sea local.*

```
int main (int argc, char *argv[]) { → C y S
    int GLOBAL = 10; → C y S
    int suma = sumar(GLOBAL, 1); → C y S
    GLOBAL = suma; → C y S
    printf(suma); → C, S y B
}
int sumar(int op1, int op2) {
    return op1 + op2;
}
```

Se ejecuta en orden y tenemos los siguientes pedidos (el cambio de color muestra el cambio de línea en la ejecución)

C	S	C	S	C	S	C	S	C	S	B
C	C	C	C	C	C	C	C	C	C	C
-	S	S	S	S	S	S	S	S	S	S
-	-	-	-	-	-	-	-	-	-	B
FP	FP	-	-	-	-	-	-	-	-	FP

2.

```
int main (int argc, char *argv[]) {
    crear_hilos(5);
    wait(MUTEX);
    int suma = sumar(GLOBAL, 1);
    GLOBAL = suma;
    signal(MUTEX);
    printf(suma);
    esperar_hilos();
}
```

a. Sólo es necesario el semáforo MUTEX = 1. Poner semáforos en suma no es necesario porque es declarada después de que se crean los hilos, por lo que NO SE COMPARTE.

b. Dado que sólo comparten GLOBAL, y que es una única instancia, no pueden quedar en deadlock.

c. Si fuesen ULTs, la función “esperar_hilos” no podría bloquearlos, porque de ser así se bloquearían todos.

El tiempo de duración del examen final será de 90’ a contar desde el momento de comienzo del mismo. Si el alumno por algún motivo comenzara más tarde sólo podrá utilizar el tiempo remanente. Utilice hojas separadas para la teoría / ejercicios.