Sistemas Operativos
1° Recuperatorio Parcial 1C2017 - TT - Resolución

Aclaración: La mayoría de las preguntas o ejercicios no tienen una única solución. Por lo tanto, si una solución particular no es similar a la expuesta aquí, no significa necesariamente que la misma sea incorrecta. Ante cualquier duda, consultar con el profesor del curso.

TEORÍA:

1) Consiste en las etapas de fetch (traer instrucción), decode (decodificar la instrucción) y execute (ejecutar la instrucción). Al final se realiza un chequeo de la existencia de interrupciones, y en caso de que exista alguna se llama al sistema operativo.
Sí, podría ocurrir. Por ejemplo, al ejecutar una instrucción de división por cero.

2) Un proceso padre y el proceso hijo no comparten nada. El proceso hijo conoce el identificador del proceso padre (PPID) como única referencia. Entre hilos si comparten distinta informacion: PCB, codigo, datos (variables globales), archivos abiertos, etc.

3)

	FIFO	Round Robin	SJF
Equidad (tiempo cpu)	No presenta mucha debido a usar el orden de llegada	Presenta bastante equidad, debido al límite de tiempo por cpu (quantum)	Su criterio de equidad se basa en priorizar procesos cortos. No es muy justo.
Overhead	Poco overhead debido a ser algoritmo sencillo	Cierto overhead por interrumpir los procesos cada tanto	Mucho overhead por tener que comparar rafagas y potencialmente desalojar
Starvation			

4)
- Mutua exclusión obligatoria
- Tiempo de espera finito (no starvation)

- Tiempo de espera reducido
- Ingreso inmediato a región crítica si la misma está libre
- No interferencia de un proceso hacia otros si el mismo no está en ninguna región crítica

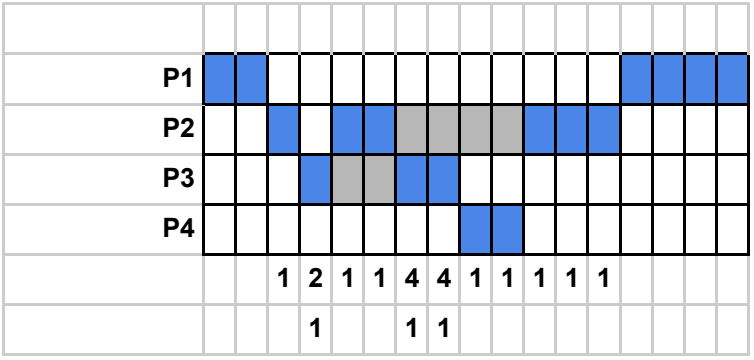
5)

1. Exclusión Mutua: sólo un proceso puede usar un recurso cada vez.
2. Retención y Espera: un proceso puede retener unos recursos asignados mientras espera que se le asignen otros
3. No Expropiación: ningún proceso puede ser forzado a abandonar un recurso que retenga
4. Espera Circular: Cadena cerrada de procesos, retiene al menos un recurso que necesita el siguiente proceso de la cadena.

PRACTICA

Ejercicio 1:

a)



b) Corto plazo: En todos los instantes donde se haya producido un process switch, o haya ingresado un proceso nuevo y vaya directo a ejecutar.
Largo plazo: todos los instantes donde haya llegado un proceso al sistema o finalizado.

c)

- new: solo podría haber sido utilizado durante la creación de un proceso (no para frenar su ingreso a Ready, dado que no esta especificado algun nivel maximo de multiprogramacion)
- ready/running/blocked: cuando el proceso está listo, ejecutando o bloqueado
- suspendido/bloqueado - suspendido/listo: no se usa en ningún momento
- exit: no hay indicios de que se use (se usaría cuando el proceso finaliza pero se desean recolectar su valor de retorno o estadísticas)

Ejercicio 2:

Nota: se elige el uso de la instrucción de hardware test_and_set como modo para evitar la condición de carrera al manipular el semáforo. Podría haberse usado también alguna solución de software o la deshabilitación de las interrupciones.

```
struct {
       int valor;
       t_queue bloqueados;
       int ocupado; // para la región
crítica del semáforo en si mismo
} t_sem;



void signal(struct t_sem sem, int qty)
{
  while(!test_and_set(&ocupado,1));
  sem.valor += qty;
  if(sem.valor <= 0){
    ocupado = 0;
    desbloquear(sem, algun_proceso());
  } else {
    ocupado = 0;
  }
}
```

```
void wait(struct t_sem sem, int qty) {
  while(!test_and_set(&ocupado,1));
  sem.valor -= qty;
  if(sem.valor < 0){
    ocupado = 0;
    bloquear(sem, proceso_actual());
  } else {
    ocupado = 0;
  }
}

int try_wait(struct t_sem sem, int qty)
{
  while(!test_and_set(&ocupado,1));
  sem.valor -= qty;
  if(sem.valor < 0){
    sem.valor += qty;
    ocupado = 0;
    return REINTENTAR_MAS_TARDE;
  } else {
    ocupado = 0;
    return OPERACION_EXITOSA;
  }
}
```

Ejercicio 3

a) Recursos Totales = 5,3,4,3 ; Recursos Asignados = 5,2,3,3 => RD = 0,1,1,0

No puede avanzar ningún proceso. P5 está en starvation (no retiene recurso alguno, solo esta esperando lo que intervienen en el deadlock). P1, P2, P3, P4 están en deadlock.

b) La mejor secuencia de finalización es matando: P2. La peor secuencia es P4 -> P1 -> P2 . P5 no es contemplado, ya que no tiene recursos asignados.

Mejor secuencia: RD = 0,1,1,0 ; P2 -> RD' = 2,2,1,1
Ahora cuento con los recursos disponibles como para que P4, P1 y P3 finalicen:

P4 -> RA = 1,0,1,1
RD'' = 3,2,2,2

P1 -> RA = 1,1,2,0
RD''' = 4,3,4,2

P3 -> RA = 1,0,0,1

RD’’’’ = 5,3,4,3 = RT

P5 ->RA = 0,0,0,0

RD’’’’’ = 5,3,4,3 = RT

Peor secuencia: RD = 0,1,1,0 ; P4 -> RD’ = 1,1,2,1 ; P1 -> RD’’ = 2,2,4,1 ;
P2 -> RD’’’ = 4,3,4,2 Ahora cuento con los recursos disponibles como para que P3 finalice:

P3 -> RA = 1,0,0,1

RD’’’’ = 5,3,4,3 = RT

P5 ->RA = 0,0,0,0

RD’’’’’ = 5,3,4,3 = RT