Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Escuela de Ciencias Exactas y Naturales
Departamento de Ciencias de la Computación
R-322 — Sistemas Operativos I

Nombre y Apellido: [redacted] Aprobado +

Examen Parcial

Ej. 1. Responda verdadero o falso y justifique:

a) Los semáforos tienen menos expresividad que los mutex.

b) Los semáforos tienen más expresividad que las variables de condición.

c) Los procesos creados con fork() comparten la memoria correspondiente a todas las variables definidas antes de la invocación al fork().

d) En un sistema multiprocesador, el algoritmo de Peterson es una solución correcta al problema de exclusión mutua para 2 procesos.

Ej. 2. ¿Qué es un deadlock?

De las condiciones que se listan a continuación, ¿cuáles deben ocurrir en simultáneo para que se produzca un deadlock en un programa/sistema?

a) La existencia de región crítica con acceso mutuamente exclusivo.

b) La existencia de región crítica.

c) Hold and wait: Un proceso puede bloquearse esperando un recurso teniendo otros recursos.

d) No preemption: Un proceso no puede sacarle un lock a otro, el unlock debe ser voluntario.

e) Espera circular: Que se forme un ciclo en el grafo de recurso tenido / esperado.

f) Ejecución paralela de al menos 2 procesos.

g) Otra no especificada (decir cuál).

Razone: Si todas las que eligió deben ocurrir en simultáneo, bastaría con eliminar una condición para prevenir un deadlock. Entonces, siendo usted un programador de aplicaciones concurrente, ¿cúal o cuáles de ellas podría eliminar para que su aplicación / programa no sufra de eventuales deadlocks?

Ej. 3. Supongamos que tenemos 2 threads o procesos que se ejecutan concurrentemente y tienen la siguiente pinta:

```
Thread A
{
a1;
a2;
}

Thread B
{
b1;
```

b2;
}

Donde $a1, a2, b1, b2$ son líneas de código que no nos interesan. Lo único importante es que se necesita garantizar que $a1$ pase antes que $b2$ y que $b1$ pase antes que $a2$. Implemente usando semáforos una solución adecuada a este problema.

Ej. 4. Completar las líneas en blanco de forma tal que el programa imprima siempre "Yeet Haw. Asumir que el sistema se comporta con semántica Mesa.

```
int ben = 0; eq_1=0;
pthread_mutex_t m;
pthread_cond_t ben_eq_1;
void *helper(void *arg) {
  pthread_mutex_lock(&m);
  ben += 1;
  eq_1=1;
  pthread_cond_signal(&ben_eq_1);
  pthread_exit(NULL); pthread_mutex_unlock(&m);
}

void main() {
  pthread_t thread;
  pthread_create(&thread, NULL, &helper, NULL);
  pthread_yield();
  pthread_mutex_lock(&m);
  for(eq_1==0)
  pthread_cond_wait(&ben_eq_1, &m);
  if (ben == 1)
    printf("Yeet Haw\n");
  else
    printf("Yee Howdy\n");
  pthread_mutex_unlock(&m);
}
```

(for ?) → Debe ir un condicional if o while

↑ Mesa

Ej. 5. ¿Qué son y cómo se relacionan entre sí los conceptos de región crítica, u *race condition* y exclusión mutua?