PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS OPERATIVOS

2da práctica (tipo a)
(Primer semestre de 2017)

Horario 0781: prof. V. Khlebnikov

Duración: 1 h. 50 min.
Nota: No se puede usar ningún material de consulta.
La presentación, la ortografía y la gramática influirán en la calificación.
Puntaje total: 20 puntos

Supongamos que existen dos procesos, **P** y **Q**. Cada proceso consiste en varios hilos concurrentes. Cada hilo incluye un bucle infinito en cual se realiza un trabajo de preparación de un dato (aquí será un entero) que se envía a un hilo de otro proceso que realiza su parte de trabajo sobre el dato. El intercambio de los datos se realiza por una memoria compartida entre los procesos. Se exigen 2 requerimientos:

1. Después de que un hilo P1 del proceso **P** proporcione el dato a un hilo Q1 en **Q**, el hilo P1 puede proceder solamente después de recibir un dato de Q1. Simétricamente, Q1, al proporcionar un dato a P1, puede proceder solamente después de recibir un dato de P1.

2. El hilo P1, al colocar un dato a la memoria compartida, debe asegurar que ningún otro hilo en **P** sobrescriba el dato en la memoria compartida antes que lo obtenga un hilo en **Q**.

Caso 1. (**5 puntos – 25 min.**) Consideremos que los procesos **P** y **Q**, tienen solo un hilo cada uno:

```
 1      semaphore semP = 0, semQ = 0;
 2      int bufP, bufQ;   /* en la memoria compartida */

 3      thread P1()                     thread Q1()
 4      {                               {
 5          int itemP;                      int itemQ;

 6          while (TRUE) {                  while (TRUE) {
 7              ...                             ...
 8              itemP = ...;                    itemQ = ...;    /* producir el dato parcial */
 9              up(&semQ);                      up(&semP);
10              down(&semP);                    down(&semQ);
11              bufP = itemP;                   bufQ = itemQ;   /* colocar el dato para otro hilo */
12              itemP = bufQ;                   itemQ = bufP;   /* obtener el dato procesado por otro hilo */
13              ...                             ...
14          }                               }
15      }                               }
```

Indique la secuencia de ejecución que produce un resultado no deseado. Lo que significa que el código no es correcto. Use el siguiente formato para la secuencia de ejecución solicitada: P1:8,9; Q1:8,9, … , lo que quiere decir que el hilo P1 ejecutó las líneas 8, 9 y después el hilo Q1 también ejecutó las líneas 8 y 9, por ejemplo.

Caso 2. (**5 puntos – 25 min.**) Consideremos que el proceso **P** tiene 2 hilos y el proceso **Q** tiene un solo hilo:

```
 1      semaphore semP = 0, semQ = 0, mutex = 1;
 2      int bufP, bufQ;

 3      thread Pi()                     thread Q1()
 4      {                               {
 5          int itemP;                      int itemQ;

 6          while (TRUE) {                  while (TRUE) {
 7              ...                             ...
 8              itemP = ...;                    itemQ = ...;
 9              up(&semQ);                      up(&semP);
10              down(&semP);                    down(&semQ);
11                  down(&mutex);                   down(&mutex);
12                      bufP = itemP;                   bufQ = itemQ;
13                  up(&mutex);                     up(&mutex);
14              up(&semQ);                      up(&semP);
```

```
15              down(&semP);                    down(&semQ);
16                  down(&mutex);                   down(&mutex);
17                      itemP = bufQ;                   itemQ = bufP;
18                  up(&mutex);                     up(&mutex);
19              ...                             ...
20          }                               }
21      }                               }
```

Ahora imagine que un hilo en **P** y el hilo en **Q** ya "establecieron un contacto" y en este momento aparece el segundo hilo en **P**. Indique la secuencia de ejecución que produce un resultado no deseado. Lo que significa que este código tampoco es correcto.

Caso 3. (**5 puntos – 25 min.**) Para encontrar el error en el siguiente código suficiente considerar un hilo en cada proceso (a pesar que el código fue preparado para múltiples hilos en cada proceso), pero ahora se necesita prestar atención a la semántica de sentencia **`while`** y sus consecuencias:

```
1       semaphore readyP=1, doneP=0, readyQ=1, doneQ=0;
2       int bufP, bufQ;

3       thread Pi()                     thread Qj()
4       {                               {
5           int itemP;                      int itemQ;

6           while (TRUE) {                  while (TRUE) {
7               ...                             ...
8               itemP = ...;                    itemQ = ...;
9               down(&readyP);                  down(&readyQ);
10                  bufP = itemP;                   bufQ = itemQ;
11                  up(&doneP);                     up(&doneQ);
12                  down(&doneQ);                   down(&doneP);
13                  itemP = bufQ;                   itemQ = bufP;
14              up(&readyP);                    up(&readyQ);
15              ...                             ...
16          }                               }
17      }                               }
```

Indique la secuencia de ejecución que produce un resultado no deseado. Lo que significa que este código tampoco es correcto.

Caso 4. (**5 puntos – 25 min.**) Dicen que en el siguiente código un 2do hilo puede "robar" el dato "predestinado" al 1er hilo en el mismo proceso:

```
1       semaphore readyP=1, doneP=0, readyQ=1, doneQ=0;
2       int bufP, bufQ;

3       thread Pi()                     thread Qj()
4       {                               {
5           int itemP;                      int itemQ;

6           while (TRUE) {                  while (TRUE) {
7               ...                             ...
8               itemP = ...;                    itemQ = ...;
9               down(&readyQ);                  down(&readyP);
10                  bufP = itemP;                   bufQ = itemQ;
11                  up(&doneP);                     up(&doneQ);
12                  down(&doneQ);                   down(&doneP);
13                  itemP = bufQ;                   itemQ = bufP;
14              up(&readyP);                    up(&readyQ);
15              ...                             ...
16          }                               }
17      }                               }
```

Indique la secuencia de ejecución que refleja el hecho del "robo" descrito.

La práctica ha sido preparada por VK
con LibreOffice Writer en Linux Mint 18.1 Serena.

Profesor del curso: (0781) V. Khlebnikov

Pando, 12 de mayo de 2017