PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS OPERATIVOS
Examen 1
(Segundo semestre de 2023)

Horarios 0781, 0782: prof. V. Khlebnikov

Duración: 3 horas
Notas: No está permitido el uso de ningún material o equipo electrónico.
La presentación, la ortografía y la gramática influirán en la calificación.
Puntaje total: 20 puntos

Pregunta 1 (5 puntos – 45 min.)

a) (0,5 puntos) Un programa tiene código y datos. Un proceso además tiene pila. ¿Qué se almacena en ella?

b) (0,5 puntos) ¿Por qué en un proceso se almacenan los registros de la CPU y no se almacenas los registros de otros dispositivos como la tarjeta de red, por ejemplo, o la tarjeta de video, o el disco duro?

c) (0,5 puntos) ¿Por qué pueden haber los estados de procesos *Ready Suspend*, *Blocked Suspend* pero no puede haber el estado *Running Suspend*?

d) (0,5 puntos) ¿Cuáles son los recursos privados de un hilo (*thread*)?

e) (2 puntos)

```
 1     $ date +%F --date=tomorrow
 2     2023-10-11
 3     $ >foo
 4     $ >foo date +%F --date=tomorrow
 5     $ >>foo date +%F --date=tomorrow
 6     $ cat foo
 7     2023-10-11
 8     2023-10-11
 9     $ <foo cat
10     2023-10-11
11     2023-10-11
12     $
```

¿Qué programa se ejecuta con la orden dada en la línea 3 y cuál será el resultado de ejecución?

¿Cuál es la diferencia en el comportamiento de la ejecución del programa **`cat`** en las líneas 6 y 9?

f) (1 punto) ¿En qué consiste el trabajo del servidor de reencarnación?

Pregunta 2 **(5 puntos – 45 min.)**

a) (2 puntos) Usted sabe que el proceso hijo hereda las copias de las variables del proceso padre y por eso los cambios de los valores de las variables, los que suceden después de la creación del proceso hijo en uno de los procesos, no afectan a las variables del otro proceso. Pero si el proceso, antes de crear al proceso hijo, además de una variable tiene la otra variable de tipo puntero cuyo valor es justamente la dirección de la primera variable. Se crea el proceso hijo que hereda ambas variables. Si el proceso hijo modifica la primera variable, se modificará su propia copia de la variable y nada más. Pero si el proceso hijo modificará el valor de la primera variable por el puntero, usando su dirección guardada en el puntero heredado, ¿cuál será el resultado? Explíquelo.

b) (3 puntos) El programa `M` se ejecuta con la siguiente orden que le pasa 4 parámetros (`argc=5`, `argv[0]='M'`, `argv[1]='O'`, `argv[2]='S'`, etc.):

```
$ ./M O S 4 E
```

El programa verifica si la cantidad de argumentos (`argc`) es 4.
Solamente cuando esto es cierto, hace la llamada al sistema `fork` y nada más.
Si `argv[1]` y `argv[2]` coinciden, hace la llamada al sistema `exit`.
Incrementa en 1 el valor de `argv[1]` y hace la llamada al sistema `exec` de la siguiente manera:

```
execl(argv[0],argv[0],argv[1],argv[2],argv[3],(char *) NULL);
```

¿Cuántos procesos ejecutarán este programa? Describa el árbol de procesos e indica su altura y la cantidad de hojas de este árbol en su apariencia máxima.

Pregunta 3 **(5 puntos – 45 min.)** Se presenta el siguiente algoritmo:

```
int   in[1:n]=([n] 0),
    last[1:n]=([n] 0);    /* 2 vectores de n elem. Inicialmente 0 */

process CS[i=1 to n]
{
    while (true)
    {
        for [j=1 to n]
        {
            /* entry protocol */
            /* remember process i is in stage j and is last */
            last[j]=i; in[i]=j;
            for [k=1 to n such_that i!=k]       /* todos k de 1 a n pero diferente de i */
            {
                /* wait if process k is in higher numbered stage
                   and process i was the last to enter stage j */
                while (in[k]>=in[i] and last[j]==i);
            }
        }

        critical section;
        in[i]=0;              /* exit protocol */
        noncritical section;
    }
}
```

The value of `in[i]` indicates which stage `CS[i]` is executing; the values of `last[j]` indicates which process was the last to begin stage j. If `CS[i]` was the last process to enter stage j, it waits for another process to enter stage j and only then can enter stage j. Thus, at most n-1 processes can be past the first stage, n-2 past the second stage, and so on

Si, para `n = 3`, llegan a la vez tres procesos `CS[1]`, `CS[2]`, `CS[3]`, uno tras otro en este orden, y durante su *quantum* de tiempo con el algoritmo de planificación *Round-Robin*, cada uno logra entrar en la sección crítica o quedarse en `while`, ¿de qué manera cada uno de ellos ejecutará el protocolo de entrada? Presente la traza de ejecución de estos 3 procesos, indicando los valores de las variables que se modifican, hasta el momento cuando el 2do proceso entre en su sección crítica.

Pregunta 4 **(5 puntos – 45 min.)** El problema clásico de los lectores y los escritores puede ser resuelto de la siguiente manera:

```
int nr=0, nw=0;
semaphore m1=1, m2=1, m3=1;
semaphore read=1, write=1;

Procesos de los lectores (Ri):

1    down(&m3);
2      down(&read);
3        down(&m1);
4          nr = nr + 1;
6          if (nr == 1) down(&write);
8        up(&m1);
9      up(&read);
10   up(&m3);
11   read_data_base();
19   down(&m1);
20     nr = nr - 1;
22     if (nr == 0) up(&write);
24   up(&m1);
```

```
Procesos de los escritores (Wi):

1    down(&m2);
2      nw = nw + 1;
4      if (nw == 1) down(&read);
6    up(&m2);
7    down(&write);
8      write_data_base();
18   up(&write);
19   down(&m2);
20     nw = nw - 1;
22     if (nw == 0) up(&read);
24   up(&m2);
```

Las **operaciones** que ejecuta cada proceso están numeradas. Algunas líneas contienen dos operaciones, por ejemplo, la operación 4 del lector es el incremento de la variable `nr` y la operación 5 es la asignación del valor incrementado a la variable `nr`. La lectura en base de datos dura 8 operaciones, mientras que la escritura dura 10 operaciones. Si el resultado de la operación 6 del lector es negativo, entonces la operación 7 no se ejecuta y la siguiente operación a ejecutarse será la 8. La planificación de procesos sigue la política *round-robin* con el *quantum* de 4 operaciones. El despachador mismo no consume tiempo para el cambio de contexto. Y la última regla muy importante: si hay varios procesos bloqueados en el mismo semáforo, la operación `up()` de este semáforo libera los procesos según FIFO.

Si al momento 0 en la cola de procesos listos tenemos R1, R2, W1, R3 y W2 que llegaron en este orden, presente sus ejecuciones, según el algoritmo presentado antes, de la siguiente manera:

Tiempo
0 Listos: R1, R2, W1, R3, W2. Se ejecuta R1 (1 – 4).
4 R2 (1), se bloquea en m3
5 W1 (1 – 4), true
9 R3 (1), se bloquea en m3 (R2,R3)
10 …

Preparado por VK
con LibreOffice Writer en Linux Mint 21.2 “Victoria”

Profesor del curso: V. Khlebnikov

Lima, 11 de octubre de 2023