PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS OPERATIVOS

1ra práctica (tipo a)
(Primer semestre de 2020)

Horario 0781: prof. V. Khlebnikov
Horario 0782: prof. A. Bello R.

Duración: 1 h. 50 min.
Nota: **La presentación, la ortografía y la gramática influirán en la calificación.**
Puntaje total: 20 puntos

Pregunta 1 (6 puntos – 30 min.) Su respuesta debe estar en el Campus Virtual (Documentos del curso, Prácticas, Práctica 1, 0781/0782) **antes de las 11:40**. Por cada 3 minutos de retardo son -2 puntos.
El nombre de su archivo debe ser <*su_código_de_8_dígitos*>`_11.txt`. Por ejemplo, `20171903_11.txt`.

Si en un programa está la siguiente sentencia:

```
            while (*p++ && (fork() || fork()));
```

donde

```
char  c[] = "hi!";
char* p = c;
```

¿cuántos procesos van a ejecutar este programa?

Tome en cuenta que, en este lenguaje de programación, la evaluación de expresiones booleanas, por defecto, es de corto circuito (*short-circuit boolean-expression evaluation*), o sea, si el valor de la expresión ya está determinado, entonces el resto de la expresión no se evalúa. También, considerando la evaluación de la expresión booleana, no olvide que el proceso hijo hereda el resultado de la parte ya evaluada por su padre.

Si le ayuda construcción del árbol de procesos, lo puede presentar de la siguiente forma:

```
70 --- 71 ---
    |      |-
    |
    |- 72 ---
```

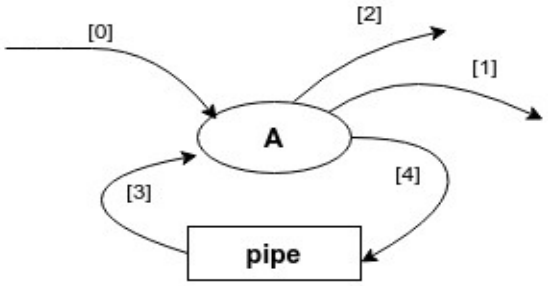
Pregunta 2 (6 puntos – 30 min.) Su respuesta debe estar en el Campus Virtual (Documentos del curso, Prácticas, Práctica 1, 0781/0782) **antes de las 12:20**. Por cada 3 minutos de retardo son -2 puntos.
El nombre de su archivo debe ser <*su_código_de_8_dígitos*>`_12.txt`. Por ejemplo, `20171903_12.txt`.

Dado el siguiente segmento de código

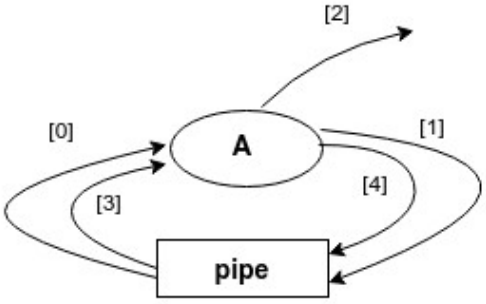
```
01 int fd[2];
02 pid_t haschild;
03
04 pipe(fd);
05 dup2(fd[0], STDIN_FILENO);
06 dup2(fd[1], STDOUT_FILENO);
07 close(fd[0]);
08 close(fd[1]);
09 pipe(fd);
10 haschild = fork();
11 if (haschild > 0)
12    dup2(fd[1], STDOUT_FILENO);
13 else if (!haschild)
14    dup2(fd[0], STDIN_FILENO);
15 close(fd[0]);
16 close(fd[1]);
17 pipe(fd);
18 haschild = fork();
```

a continuación se hace una descripción gráfica de dicho segmento de código.

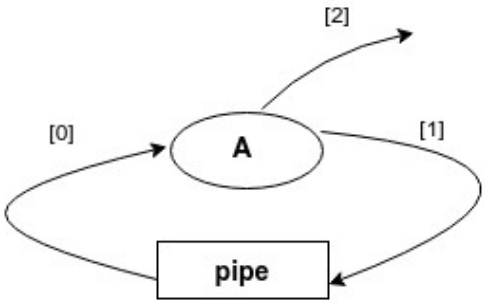
- Estado del proceso después de ejecutar la línea: `04 pipe(fd);`



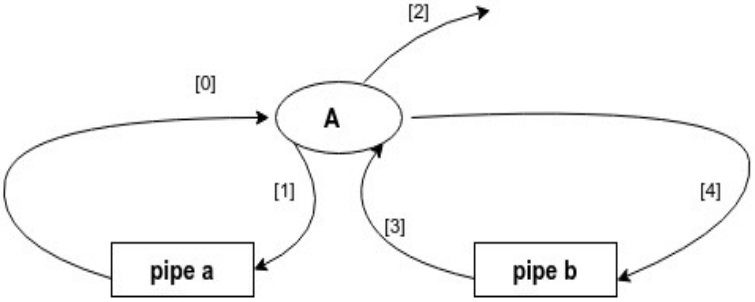
- Estado del proceso después de ejecutar las líneas:
```
05 dup2(fd[0], STDIN_FILENO);
06 dup2(fd[1], STDOUT_FILENO);
```



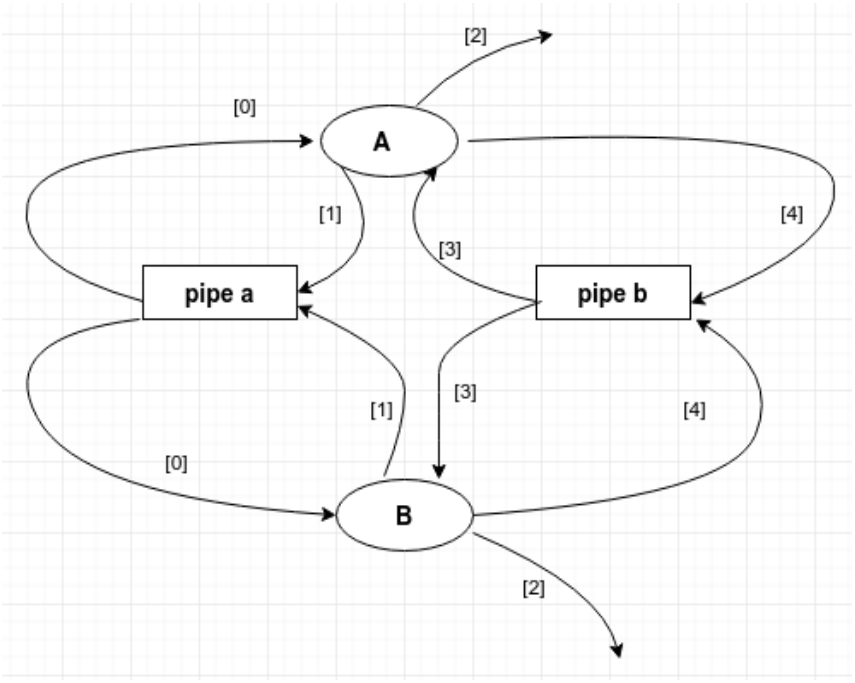
- Estado del proceso después de ejecutar las líneas:
```
07 close(fd[0]);
08 close(fd[1]);
```



- Estado del proceso después de ejecutar la línea: `09 pipe(fd);`

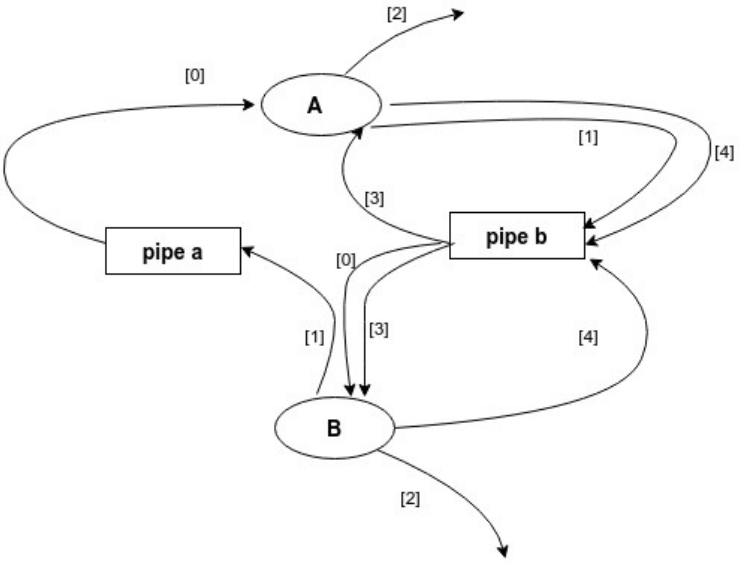


- Estado del proceso después de ejecutar la línea: `10 haschild = fork();`

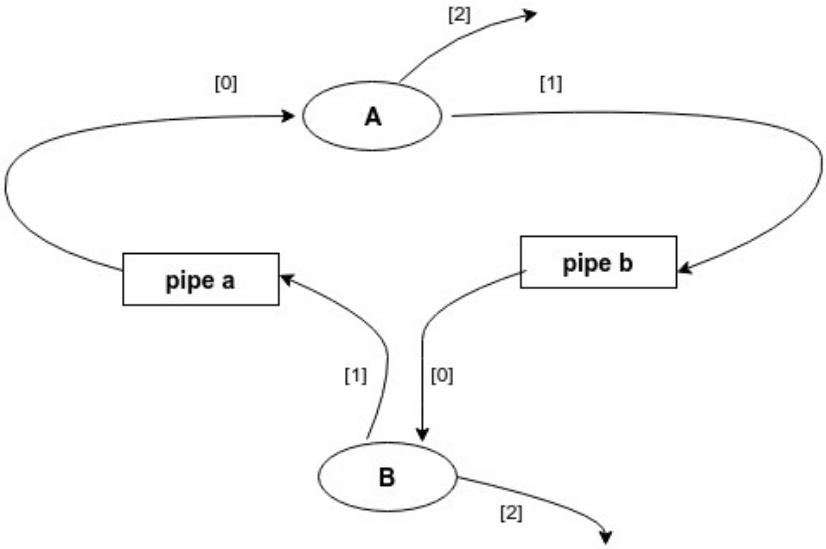


- Estado del proceso después de ejecutar las líneas:

```
11 if (haschild > 0)
12    dup2(fd[1],STDOUT_FILENO);
13 else if (!haschild)
14    dup2(fd[0], STDIN_FILENO);
```



- Estado del proceso después de ejecutar las líneas: `15 close(fd[0]);  16 close(fd[1]);`



Después de ejecutarse las líneas 17 y 18:

a) (1 punto) ¿Cuántos procesos se han creado?

b) (1 punto) ¿Cuántos pipes se han creado?

c) (4 puntos) Para cada uno de los *pipes* creados, mencione qué proceso escribe o lee sobre cada *pipe*, indicando el correspondiente descriptor. Por ejemplo, en la figura anterior:

> **`pipe a`**: el proceso **`A`** lee del *pipe* a través de su entrada estándar ([0]) y el proceso **`B`** escribe al *pipe* a través de su salida estándar ([1]).

<u>Pregunta 3</u> (8 puntos – 30 min.) Su respuesta debe estar en el Campus Virtual (Documentos del curso, Prácticas, Práctica 1, 0781/0782) **antes de las 13:00**. Por cada 3 minutos de retardo son -2 puntos.
El nombre de su archivo debe ser <*su_código_de_8_dígitos*>`_13.txt`. Por ejemplo, `20171903_13.txt`.

a) (4 puntos) En el siguiente cuadro se asume que cada instrucción $\mathbf{p}_i$ y $\mathbf{q}_i$ son atómicas:

Instrucción de asignación con una referencia global	
`integer n ← 0`	
`p`	**`q`**

`integer temp`
`p1: temp ← n`
`p2: n ← temp + 1`
	`q1: n ← n + 1`
	`q2: n ← n + 1`
	`q3: n ← n + 1`

A partir de las instrucciones (p_i y q_i) de los hilos **`p`** y **`q`**, se pueden obtener distintos escenarios, dependiendo de la secuencia de ejecución de las instrucciones. Al final de cada secuencia el valor de la variable *n* podría ser diferente.

Por ejemplo, dos posibles secuencias son:

```
p1, p2, q1, q2, q3 → n = 4
q1, q2, q3, p1, p2 → n = 4
```

Tenga en cuenta que algunas secuencias no son válidas, por ejemplo:

`p2, p1, q1, q2, q3` no es válida porque no se puede ejecutar la instrucción `p2` antes que `p1`.

Se le solicita proporcionar todas las secuencias posibles de ejecución, indicando después de cada secuencia, el valor final de la variable *n*. No considere las proporcionadas en el ejemplo.

b) (4 puntos) Se quiere medir el tiempo de ejecución de varios hilos que implementan diferentes algoritmos para resolver el mismo problema. A todos los hilos (que son 5) se obliga ejecutar la siguiente sección de entrada para sincronizar los tiempos de inicio de ejecución de sus algoritmos:

```
#define         N       5;      /* cantidad total de hilos */

semaphore       mutex = 1;
semaphore       go = 0;
int             n = 0;          /* cantidad de hilos listos */

void
enter_region(void)
{
        down(&mutex);
        n++;
        up(&mutex);

        if (n == N) up(&go);

        down(&go);
}
```

¿En qué caso de la secuencia de ejecución se obtiene un *deadlock* (2 puntos)?
¿En qué caso de la secuencia de ejecución NO se obtiene un *deadlock* (2 puntos)?

La práctica ha sido preparada por AB (2,3a) y VK (1,3b)
con LibreOffice Writer en Linux Mint 19.3 Tricia.

Profesores del curso: (0781) V. Khlebnikov
(0782) A. Bello R.

Pando, 14 de mayo de 2020