

Comenzado el	martes, 14 de mayo de 2024, 21:20
Estado	Finalizado
Finalizado en	martes, 14 de mayo de 2024, 22:18
Tiempo empleado	57 minutos 56 segundos
Calificación	10,000 de 15,000 (66,667%)

Pregunta **1**

Correcta

Se puntúa
1,500 sobre
1,500

1) Descargue los siguientes 2 archivos en un mismo directorio:

padre.c : <https://nube.ingenieria.uncuyo.edu.ar/s/X5qjxH67QPpYaHd>

hijo.o : <https://nube.ingenieria.uncuyo.edu.ar/s/6FgYLJp8rkq6omi>

2) Modifique el archivo padre.c para que envíe al proceso hijo el contenido de la variable tx_buffer mediante la tubería declarada en la variable fd.

3) Ambos archivos se compilan con el siguiente comando en consola:

```
> gcc -m32 -c padre.c && gcc -m32 hijo.o padre.o -o padre
```

4) El binario generado por el comando anterior se ejecuta en consola con:

```
> ./padre
```

5) El proceso hijo cuando reciba el contenido de tx_buffer por la tubería, imprimirá por consola dos líneas:

```
Leido desde tuberia: HOLA HIJA MIA
```

```
El resultado del ejercicio es: XXX
```

6) Copie el número de 3 cifras XXX en la casilla de abajo. Este número es la respuesta del ejercicio.

Respuesta: 

Pregunta **2**

Incorrecta

Se puntúa
0,000 sobre
1,000

Indique características del algoritmo shortest process next.

Seleccione una o más de una:

- ☐ a. Minimiza el tiempo de respuesta promedio.
- ☐ b. El cálculo de envejecimiento se puede hacer con poco costo computacional.
- ☒ c. Se deben tener todos los procesos en tiempo inicial T0. ✖
- ☒ d. Es una mejora a shortest Job First. ✖
- ☐ e. Usa una aproximación basada en registración del comportamiento anterior.

Pregunta **3**

Incorrecta

Se puntúa
0,000 sobre
1,500

1) Descargue los siguientes 2 archivos en un mismo directorio:

padre.c : <https://nube.ingenieria.uncuyo.edu.ar/s/D8YRF2eG3rjxokp>

hijo.o : <https://nube.ingenieria.uncuyo.edu.ar/s/jJnY7woyMDxrQLJ>

2) Modifique el archivo padre.c para que envíe al proceso hijo MENSAJE por la cola de mensajes ya declarada.

3) Ambos archivos se compilan con el siguiente comando en consola:

```
> gcc -c padre.c -lrt && gcc hijo.o padre.o -o padre -lrt
```

4) El binario generado por el comando anterior se ejecuta en consola con:

```
> ./padre
```

5) El proceso hijo a recibir MENSAJE Imprimirá el resultado por consola,

```
Hijo: recibido mensaje esperado
```

```
Hijo: el resultado del ejercicio es XXX
```

6) Copie el número de 3 cifras XXX en la casilla de abajo. Este número es la respuesta del ejercicio.

Respuesta: ✖

Pregunta **4**

Correcta

Se puntúa
1,000 sobre
1,000

Seleccione lo correcto respecto a señales

Seleccione una:

- ☐ a. Las señales se generan exclusivamente con la syscall kill().
- ☐ b. Una señal es la petición por parte del usuario para la terminación de un proceso.
- ☐ c. Las señales tienen persistencia de kernel.
- ☐ d. Se pueden enviar señales entre procesos no relacionados, a excepción de SIGSTOP.
- ☒ e. Una señal se usa generalmente para comunicar estados entre procesos. ✓



Pregunta **5**

Correcta

Se puntúa
1,000 sobre
1,000

Una lo que corresponda:

La persistencia de los datos de una FIFO

es de proceso. ✓

La persistencia de una cola de Message Queue

es de kernel. ✓

La persistencia del nombre de una FIFO

es de sistema de archivos. ✓

La persistencia de las tuberías

es de proceso. ✓

Pregunta **6**

Correcta

Se puntúa
1,000 sobre
1,000

¿Qué es un proceso?

Seleccione una:

- ☐ a. Un proceso es una instancia de un programa en ejecución que solo se puede ejecutar en espacio de usuario.
- ☐ b. Un proceso es un programa listo para ser ejecutado.
- ☒ c. Un proceso es una instancia de un programa en ejecución mas el estado del mismo. ✓
- ☐ d. Ninguna de las respuestas es válida.
- ☐ e. Un proceso es una instancia de un programa en ejecución que solo puede ejecutar invocando el sistema operativo.
- ☐ f. Un proceso es el número que identifica a un programa en ejecución.



Pregunta 7

Incorrecta

Se puntúa
0,000 sobre
1,500

¿Cuál de las siguientes afirmaciones describe correctamente una clase en Python?

- ☐ a. Una clase en Python es una función que permite la repetición de código mediante el uso de bucles.
- ☒ b. Una clase en Python es una estructura de datos que almacena pares de nombre-valor y no permite nombres duplicados. ✖
- ☐ c. Una clase en Python es una colección ordenada de elementos que pueden ser modificados y permite duplicados.
- ☐ d. Una clase en Python es una plantilla para crear objetos, que define atributos y métodos que los objetos creados a partir de la clase tendrán.

Pregunta 8

Correcta

Se puntúa
1,000 sobre
1,000

¿Cuál de las siguientes, es la principal motivación para el surgimiento de sistemas multitarea?

Seleccione una:

- ☐ a. Poder ejecutar más de un proceso en pseudo-paralelo.
- ☐ b. Poder poner mas de un proceso en memoria principal simultáneamente.
- ☒ c. Mejorar el uso de CPU cuando se espera por operaciones de entrada salida. ✔
- ☐ d. Aprovechar la protección de hardware para que los procesos queden aislados entre si.
- ☐ e. Poder aprovechar el tiempo ocioso del los periféricos, y ejecutar otro proceso.

Pregunta 9

Parcialmente
correcta

Se puntúa
0,333 sobre
1,000

Seleccione lo correcto para tipos de estructura de sistemas operativos

- ☐ a. Los OS monolíticos no pueden tener una estructura interna.
- ☒ b. Los OS en Capas se basan en organizar las funciones del OS de manera jerárquica. ✔
- ☒ c. Los OS microkernel son mas robustos que los OS cliente/servidor en cuanto a fallas de drivers. ✖
- ☐ d. Los OS Máquina Virtual se caracterizan por separar multiprogramación de máquina extendida.
- ☒ e. Los OS cliente/servidor son una generalización de OS microkernel. ✔
- ☐ f. Los OS microkernel se ejecutan mas rápido que los OS monolíticos.

Pregunta **10**

Correcta

Se puntúa
1,000 sobre
1,000

Seleccione las afirmaciones correctas respecto a Posix Pipes:

Seleccione una o más de una:

- ☐ a. Poseen un nombre en el sistema de archivos.
- ☐ b. Se pueden abrir en forma bloqueantes y no bloqueantes.
- ☒ c. Se destruyen al cerrar todos los descriptores de lectura y escritura. ✓
- ☒ d. Permiten el intercambio de datos entre procesos relacionados. ✓
- ☐ e. Permiten sincronizar procesos no relacionados.
- ☐ f. Poseen persistencia de kernel.
- ☐ g. Se utilizan para enviar datos entre procesos no relacionados.
- ☐ h. Se destruyen al cerrar los descriptores de escritura.

Pregunta 11

Correcta

Se puntúa
1,500 sobre
1,500

Dos procesos relacionados se comunican por medio de una FIFO.

1. El proceso padre abre la FIFO con permisos de solo lectura, lee la FIFO y muestra lo leído.
2. El proceso padre espera a que el proceso hijo termine para cerrar y eliminar la FIFO y luego termina él.
3. El proceso hijo abre la FIFO con permisos de solo escritura, escribe "0123456789" en la FIFO, cierra la FIFO y luego termina él.

Completar:

```
#define E "0123456789"
#define AA "/tmp/CC"
int a,x,c,f;
char d[10];
int main(){
    mkfifo(AA, 0777);
    //linea en blanco ✓
    x=fork();
    switch (x) {
        case 0:
            c = open(AA, O_WRONLY , 0); ✓
            write(c, E,sizeof(E));
            close(c);
            exit(0);
        break;
        default:
            a = open(AA, O_RDONLY, 0); ✓
            f = read(a, d, sizeof(d));
            write(STDOUT_FILENO, d, f);
        break;
    }
    wait(NULL); ✓
    close(a);
    unlink(AA); ✓
    exit(0);
}
```

close(a);

c = open(AA, O_RDWR, 0);

a = open(AA, O_RDONLY, 0);

a = open(AA, O_RDWR, 0);

wait(NULL);

c = open(AA, O_WRONLY , 0);

read(c, E,sizeof(E));

//linea en blanco

unlink(AA);

write(c, E,sizeof(E));

kill(SIGUSR1,x);

Pregunta **12**

Correcta

Se puntúa
1,000 sobre
1,000

Indique lo que crea correcto sobre la syscall signal()

Seleccione una o más de una:

- ☐ a. Se utiliza para terminar un proceso del cual se conoce el PID.
- ☒ b. Se utiliza para indicar qué hacer cuando el proceso recibe algunas señales. ✓
- ☐ c. Se utiliza para enviar una señal específica a un proceso.
- ☒ d. Se utiliza para ignorar la recepción de algunas señales. ✓
- ☐ e. Se utiliza para especificar qué hacer cuando se recibe la señal SIGSTOP.
- ☐ f. Se utiliza para terminar la ejecución del proceso que recibe la señal.
- ☐ g. Se utiliza para ignorar la recepción de la señal SIGKILL.



Pregunta **13**

Parcialmente
correcta

Se puntúa
0,667 sobre
1,000

ordene los pasos en los que se lleva a cabo la llamada a sistema POSIX read()

- | | | |
|---|--|---|
| 1 | Push de los parámetros de la syscall a la pila en orden inverso. | ✓ |
| 2 | Retorno del programa de usuario al procedimiento de biblioteca. | ✗ |
| 3 | Ejecución de la call gate. | ✓ |
| 4 | Ejecución del procedimiento de la biblioteca. | ✗ |
| 5 | Retorno al procedimiento de biblioteca de espacio usuario. | ✓ |
| 6 | Retorno del procedimiento de biblioteca al programa de usuario. | ✓ |

Comenzado el	martes, 14 de mayo de 2024, 21:20
Estado	Finalizado
Finalizado en	martes, 14 de mayo de 2024, 22:19
Tiempo empleado	58 minutos 36 segundos
Calificación	9,883 de 15,000 (65,889%)

Pregunta **1**

Correcta

Se puntúa 1,000 sobre 1,000

Indique cuál de las siguientes afirmaciones es verdadera para procesos zombies:

Seleccione una o más de una:

- ☐ a. Es un proceso en el cual su padre ha finalizado.
- ☐ b. Es un proceso en estado bloqueado esperando un evento.
- ☒ c. Es sacado de memoria completamente cuando el padre ejecuta la función wait(). ✓
- ☐ d. Solo se conserva su espacio de direcciones reservado por el sistema operativo, hasta que el padre lea su exit status.
- ☐ e. Solo puede pasar a estado listo al recibir una señal.
- ☒ f. Solo se conserva su exit status en la entrada PCB (Process Control Block) de la Tabla de Procesos. ✓
- ☐ g. Es un proceso en estado listo que puede terminar con SIGKILL.

Pregunta **2**

Correcta

Se puntúa 1,500 sobre 1,500

Dos procesos relacionados se comunican por medio de una tubería.

1. El proceso hijo escribe "0123456789" en la tubería y termina.
2. El proceso padre lee de la tubería y muestra lo leído.

Completar:

```
#define E "0123456789"
int a[2], b,e;
char d[10];
int main (){
    pipe(a); ✓
    b = fork(); ✓
    if (b==0){
        //línea en blanco ✓
        close(a[0]);
        strncpy(d, E, sizeof(E) );
        write(a[1], d, sizeof(E)); ✓
        exit(0);
    }
    close(a[1]);
    e = read(a[0], d, sizeof(d));
    write(STDOUT_FILENO, d, e); ✓
    exit(0);
}
```

pipe(b);	write(a[1], d, sizeof(E));	read(a[1], d, sizeof(d));
close(a[1]);	write(STDOUT_FILENO, E, sizeof(E));	wait(NULL);
//línea en blanco	write(a[0], d, sizeof(d));	pipe(a);
write(STDOUT_FILENO, d, e);	close(a[0]);	b = fork();

Pregunta 3

Incorrecta

Se puntúa 0,000 sobre 1,500

¿Qué salida produce el siguiente código en Python?

```
1 fruits = ["orange", "kiwi", "mango", "papaya"]
2 result = ""
3 i = 0
4 while i < len(fruits):
5     if len(fruits[i]) > 4:
6         result += fruits[i][2]
7     else:
8         result += fruits[i][0]
9     i += 1
10 print(result)
11
```

- ☐ a. rkap
- ☐ b. orap
- ☒ c. akpn ✖
- ☐ d. aknp

Pregunta 4

Parcialmente correcta

Se puntúa 0,750 sobre 1,000

Una lo que corresponda:

La persitencia del nombre de una FIFO

es de kernel.



La persistencia de una cola de Message Queue

es de kernel.



La persistencia de las tuberías

es de proceso.



La persitencia de los datos de una FIFO

es de proceso.



Pregunta 5

Parcialmente correcta

Se puntúa 0,300 sobre 1,000

Indique lo que crea correcto sobre la syscall kill()

Seleccione una o más de una:

- ☐ a. Se utiliza para configurar ignorar todas las señales a excepción de SIGKILL y SIGSTOP.
- ☐ b. Siempre se utiliza para terminar un proceso del cual se conoce el PID.
- ☐ c. Se utiliza para especificar qué hacer cuando se recibe la señal SIGKILL.
- ☒ d. Se utiliza para enviar una señal a un proceso en ejecución, a excepción de SIGKILL y SIGSTOP ✗
- ☒ e. Se puede utilizar para que un proceso se envíe una señal a si mismo. ✓
- ☐ f. Se utiliza para enviar una señal específica a un proceso en ejecución.
- ☐ g. Se utiliza para especificar qué hacer cuando el proceso se recibe una señal específica.

Pregunta 6

Parcialmente correcta

Se puntúa 0,833 sobre 1,000

Para la técnica IPC pipe, una lo que corresponda:

Si un proceso intenta escribir en el extremo de escritura de una tubería que cerró todos los descriptores de lectura

da error de escritura y termina. ✗

Si un proceso intenta escribir en el extremo de escritura de una tubería que está vacía

el proceso logra escribir. ✓

Si un proceso intenta leer en el extremo de lectura de una tubería que está llena

el proceso lee datos. ✓

Si un proceso intenta leer en el extremo de lectura de una tubería que está vacía y todos sus descriptores de escritura fueron cerrados

el proceso lee 0 bytes. ✓

Si un proceso intenta leer en el extremo de lectura de una tubería que está vacía

el proceso se bloquea. ✓

Si un proceso intenta escribir en el extremo de escritura de una tubería que está llena

el proceso se bloquea. ✓

Pregunta 7

Correcta

Se puntúa 1,000 sobre 1,000

Indique cuál de las siguientes afirmaciones es correcta referidas a la abstracción Procesos:

Seleccione una o más de una:

- ☒ a. Cuando los procesos son IObounded, se logra una gran mejora en el uso de CPU, usando multiprogramación . ✓
- ☐ b. La multiprogramación es la conmutación entre programas de manera rápida, salvando su estado en las conmutaciones.
- ☐ c. No se justifica hacer multiprogramación cuando los procesos son interactivos.
- ☐ d. Cada programa tiene un espacio de direcciones independiente del resto.
- ☒ e. La abstracción de procesos permite ejecutar en pseudo-paralelo procesos. ✓
- ☐ f. Pueden coexistir dos programas con un mismo proceso.

Pregunta 8

Correcta

Se puntúa 1,000 sobre 1,000

Una lo que corresponda.**El cambio de estado de Running a Blocked es debido a**

la ejecución de syscall bloqueante. ▾

**El cambio de estado de Ready a Running es debido a**

el planificador del sistema operativo. ▾

**El cambio de estado de Blocked a Running es debido a**

no es válida esa transición. ▾

**El cambio de estado de Blocked a Ready es debido a**

interrupción de hardware producida por un periférico. ▾

**El cambio de estado de Running a Ready es debido a**

interrupción de hardware producida por un timer. ▾

**El cambio de estado de Ready a Finished es debido a**

no es válida esa transición. ▾



Pregunta 9

Correcta

Se puntúa 1,500 sobre 1,500

1) Descargue los siguientes 2 archivos en un mismo directorio:

padre.c : <https://nube.ingenieria.uncuyo.edu.ar/s/D8YRF2eG3rxokp>

hijo.o : <https://nube.ingenieria.uncuyo.edu.ar/s/neKxcFbcmYPMf2d>

2) Modifique el archivo padre.c para que envíe al proceso hijo MENSAJE por la cola de mensajes ya declarada.

3) Ambos archivos se compilan con el siguiente comando en consola:

```
> gcc -c padre.c -lrt && gcc hijo.o padre.o -o padre -lrt
```

4) El binario generado por el comando anterior se ejecuta en consola con:

```
> ./padre
```

5) El proceso hijo a recibir MENSAJE Imprimirá el resultado por consola,

```
Hijo: recibido mensaje esperado
```

```
Hijo: el resultado del ejercicio es XXX
```

6) Copie el número de 3 cifras XXX en la casilla de abajo. Este número es la respuesta del ejercicio.

Respuesta: ✓

Pregunta 10

Parcialmente correcta

Se puntúa 0,500 sobre 1,000

Una señal es:

Seleccione una o más de una:

- ☐ a. Una interrupción debida a un evento externo.
- ☐ b. Una forma de enviar estados entre procesos.
- ☐ c. Un buffer mantenido en memoria de kernel de capacidad limitada.
- ☐ d. Una forma de enviar datos entre procesos.
- ☒ e. Una notificación entregada a un proceso debido a un evento asíncrono. ✓
- ☐ f. Una petición por parte del usuario para la creación de un nuevo proceso.

Pregunta 11

Incorrecta

Se puntúa 0,000 sobre 1,000

Seleccione lo correcto para tipos de estructura de sistemas operativos

- ☐ a. Los OS monolíticos normalmente se los separa en distintos procedimientos que interactúan entre sí.
- ☒ b. Los OS monolíticos se caracterizan por separar la multiprogramación de la máquina extendida. ✖
- ☐ c. Los OS cliente/servidor son mas robustos que los OS monolíticos en cuanto a fallas de drivers.
- ☒ d. Los OS cliente/servidor se comunican internamente generalmente con paso de mensajes. ✔
- ☒ e. Los OS con sistemas de capas se ejecutan mas rápido que los OS monolíticos. ✖
- ☐ f. Los OS cliente/servidor se basan en organizar las funciones del OS de manera jerárquica.

Pregunta 12

Correcta

Se puntúa 1,500 sobre 1,500

1) Descargue los siguientes 2 archivos en un mismo directorio:

padre.c : <https://nube.ingenieria.uncuyo.edu.ar/s/X5qjxH67QPpYaHd>hijo.o : <https://nube.ingenieria.uncuyo.edu.ar/s/dyGMgK7xWwbAcPY>

2) Modifique el archivo padre.c para que envíe al proceso hijo el contenido de la variable tx_buffer mediante la tubería declarada en la variable fd.

3) Ambos archivos se compilan con el siguiente comando en consola:

```
> gcc -m32 -c padre.c && gcc -m32 hijo.o padre.o -o padre
```

4) El binario generado por el comando anterior se ejecuta en consola con:

```
> ./padre
```

5) El proceso hijo cuando reciba el contenido de tx_buffer por la tubería, imprimirá por consola dos líneas:

```
Leido desde tuberia: HOLA HIJA MIA
```

```
El resultado del ejercicio es: XXX
```

6) Copie el número de 3 cifras XXX en la casilla de abajo. Este número es la respuesta del ejercicio.

Respuesta: ✔

Pregunta **13**

Incorrecta

Se puntúa 0,000 sobre 1,000

Indique características del algoritmo shortest process next.

Seleccione una o más de una:

- ☐ a. Es facil de implementar, sumando el nuevo valor a la estimación actual y desplazando un bit a la derecha.
- ☐ b. Usa una aproximación basada en registración del comportamiento anterior.
- ☐ c. El cálculo de envejecimiento tiene alto costo computacional.
- ☒ d. Es una mejora al algoritmo de Round-Robin. ✖
- ☐ e. No es necesario tener todos los procesos en tiempo inicial T0.

Comenzado el	martes, 14 de mayo de 2024, 21:21
Estado	Finalizado
Finalizado en	martes, 14 de mayo de 2024, 22:19
Tiempo empleado	58 minutos 11 segundos
Calificación	12,833 de 15,000 (85,556%)

Pregunta **1**

Parcialmente correcta

Se puntúa 0,500 sobre 1,000

Indique lo que crea correcto sobre la syscall signal()

Seleccione una o más de una:

- ☐ a. Se utiliza para terminar un proceso del cual se conoce el PID.
- ☐ b. Se utiliza para enviar una señal específica a un proceso.
- ☐ c. Se utiliza para terminar la ejecución del proceso que recibe la señal.
- ☒ d. Se utiliza para ignorar la recepción de algunas señales. ✓
- ☐ e. Se utiliza para ignorar la recepción de la señal SIGKILL.
- ☐ f. Se utiliza para indicar qué hacer cuando el proceso recibe algunas señales.
- ☐ g. Se utiliza para especificar qué hacer cuando se recibe la señal SIGSTOP.

Pregunta **2**

Correcta

Se puntúa 1,000 sobre 1,000

Indique cuál de las siguientes transiciones de estados de un proceso NO es válida.

Seleccione una o más de una:


- ☒ a. Ready a Bloqued. ✓
- ☐ b. Blocked a Ready.
- ☐ c. Running a Finished.
- ☐ d. New a Ready.
- ☐ e. Running a Ready.
- ☐ f. Running a Blocked.
- ☒ g. New a Running. ✓
- ☐ h. Ready a Running.

Pregunta **3**

Correcta

Se puntúa 1,500 sobre 1,500

¿Cuál de las siguientes afirmaciones describe correctamente una lista en Python?

- ☐ a. Una lista es una colección no ordenada y mutable de elementos, que no permite duplicados.
- ☐ b. Una lista es una colección ordenada y inmutable de elementos, que solo puede contener un tipo de dato.
- ☐ c. Una lista es una colección no ordenada e inmutable de elementos, que permite duplicados.
- ☒ d. Una lista es una colección ordenada y mutable de elementos, que puede contener diferentes tipos de datos y permite elementos duplicados. 


Pregunta **4**

Parcialmente correcta

Se puntúa 0,500 sobre 1,000

Una señal es:

Seleccione una o más de una:

- ☒ a. Una notificación entregada a un proceso debido a un evento asíncrono. 
- ☐ b. Una forma de enviar estados entre procesos.
- ☐ c. Una interrupción debida a un evento externo.
- ☐ d. Una forma de enviar datos entre procesos.
- ☐ e. Un buffer mantenido en memoria de kernel de capacidad limitada.
- ☐ f. Una petición por parte del usuario para la creación de un nuevo proceso.

Pregunta **5**

Correcta

Se puntúa 1,000 sobre 1,000

Indique cuál de las siguientes transiciones de estados de un proceso NO es válida.

Seleccione una o más de una:

- ☒ a. Ready a Finished. ✓
- ☐ b. Running a Finished.
- ☐ c. Blocked a Ready.
- ☒ d. Block a Running. ✓
- ☐ e. Running a Blocked.
- ☐ f. Running a Ready.
- ☐ g. New a Ready.
- ☐ h. Ready a Running.

Pregunta **6**

Correcta

Se puntúa 1,500 sobre 1,500

1) Descargue los siguientes 2 archivos en un mismo directorio:

padre.c : <https://nube.ingenieria.uncuyo.edu.ar/s/D8YRF2eG3rxokp>

hijo.o : <https://nube.ingenieria.uncuyo.edu.ar/s/wnSF4FWijHWAfKx>

2) Modifique el archivo padre.c para que envíe al proceso hijo MENSAJE por la cola de mensajes ya declarada.

3) Ambos archivos se compilan con el siguiente comando en consola:

```
> gcc -c padre.c -lrt && gcc hijo.o padre.o -o padre -lrt
```

4) El binario generado por el comando anterior se ejecuta en consola con:

```
> ./padre
```

5) El proceso hijo al recibir MENSAJE imprimirá el resultado por consola,

```
Hijo: recibido mensaje esperado
```

```
Hijo: el resultado del ejercicio es XXX
```

6) Copie el número de 3 cifras XXX en la casilla de abajo. Este número es la respuesta del ejercicio.

Respuesta: ✓

Pregunta 7

Correcta

Se puntúa 1,000 sobre 1,000

Indique cuál de las siguientes afirmaciones es correcta cuando un proceso hijo (B) termina antes que su padre (A):

Seleccione una o más de una:

- ☒ a. El proceso B se convierte en un proceso zombie. ✓
- ☒ b. Se libera la memoria principal que se usaba en el proceso B. ✓
- ☐ c. El proceso A finaliza al recibir la señal SIGCHLD.
- ☐ d. El proceso A se convierte en un proceso huérfano.
- ☐ e. El proceso A es adoptado por el proceso init
- ☐ f. El proceso B se convierte en un proceso huérfano.
- ☐ g. El proceso A se convierte en un proceso zombie.

Pregunta 8

Correcta

Se puntúa 1,500 sobre 1,500

1) Descargue los siguientes 2 archivos en un mismo directorio:

padre.c : <https://nube.ingenieria.uncuyo.edu.ar/s/X5qjxH67QPpYaHd>

hijo.o : <https://nube.ingenieria.uncuyo.edu.ar/s/9EYKtNRMcSsSqI7>

2) Modifique el archivo padre.c para que envíe al proceso hijo el contenido de la variable tx_buffer mediante la tubería declarada en la variable fd.

3) Ambos archivos se compilan con el siguiente comando en consola:

```
> gcc -m32 -c padre.c && gcc -m32 hijo.o padre.o -o padre
```

4) El binario generado por el comando anterior se ejecuta en consola con:

```
> ./padre
```

5) El proceso hijo cuando reciba el contenido de tx_buffer por la tubería, imprimirá por consola dos líneas:

```
Leido desde tuberia: HOLA HIJA MIA
```

```
El resultado del ejercicio es: XXX
```

6) Copie el número de 3 cifras XXX en la casilla de abajo. Este número es la respuesta del ejercicio.

Respuesta: ✓

Pregunta **9**

Parcialmente correcta

Se puntúa 0,833 sobre 1,000

ordene los pasos en los que se lleva a cabo la llamada a sistema POSIX read()

- | | | |
|---|---|---|
| 1 | hacer push de los parámetros de la syscall a la pila en orden inverso | ✓ |
| 2 | llamada al procedimiento de la biblioteca | ✓ |
| 3 | ejecución de la call gate | ✓ |
| 4 | ejecución del controlador de periférico de espacio kernel | ✓ |
| 5 | ejecución del controlador de periférico de espacio usuario | ✗ |
| 6 | retornar del procedimiento de biblioteca al programa de usuario | ✓ |

Pregunta **10**

Correcta

Se puntúa 1,000 sobre 1,000

Seleccione cuál **NO** es una motivación para utilizar IPC:

Seleccione una:

- ☐ a. Permitir la modularidad.
- ☒ b. Permitir la comunicación entre procesos. ✓
- ☐ c. Permitir la comunicación entre procesos relacionados.
- ☐ d. Compartir información entre procesos cooperativos.
- ☐ e. Ninguna respuesta es válida.
- ☐ f. Acelerar la ejecución de tareas, implementándolas en distintos procesos.

Pregunta **11**

Correcta

Se puntúa 1,000 sobre 1,000

Indique lo correcto respecto a la planificación de procesos.

- ☐ a. Con un planificador no expropiativo se ven beneficiadas las tareas I/O bounded.
- ☐ b. Cuanto menor sea el quantum que utilizan los planificadores, más eficiente será el sistema operativo.
- ☐ c. Los planificadores expropiativos solo se ejecutan cuando el proceso en estado RUN ejecuta una syscall.
- ☐ d. Al crearse hilos en espacio usuario, se deben modificar los planificadores para contemplarlos.
- ☒ e. La necesidad de los planificadores surge para poder tener en memoria más de un proceso a la vez. ✓

Pregunta **12**

Incorrecta

Se puntúa 0,000 sobre 1,000

La función mkfifo():

Seleccione una o más de una:

- ☐ a. Tiene como único argumento el nombre del fifo.
- ☒ b. Permite crear un pipe. ✗
- ☐ c. Permite especificar que sea no bloqueante.
- ☐ d. Se debe usar en conjunto con la llamada fork().
- ☒ e. Permite abrir una fifo en modo bloqueante o no bloqueante. ✗
- ☐ f. Ninguna es correcta
- ☐ g. Retorna el tamaño de la fifo, si no hubo error.
- ☐ h. Retorna el descriptor, si no hubo error.
- ☐ i. Permite borrar una fifo.

Pregunta **13**

Correcta

Se puntúa 1,500 sobre 1,500

Dos procesos relacionados se comunican por medio de una FIFO.

1. El proceso hijo lee de la FIFO, muestra lo leído, cierra la FIFO y luego termina él.
2. El proceso padre escribe "0123456789" en la FIFO, espera a que el proceso hijo termine.
3. El proceso padre para cierra y elimina la FIFO y luego terminar él.

Completar:

```
#define E "0123456789"
#define CC "/tmp/BB"
int e,b,f,g;
char d[10];
int main(){
    mkfifo(CC, 0777);
    b=fork();
    if(b>0){
        //linea en blanco ✓
        f = open(CC, O_WRONLY, 0);
        write(f, E,sizeof(E)); ✓
        wait(NULL);
        close(f);
        unlink(CC); ✓
        exit(0);
    }
    //linea en blanco ✓
    e = open(CC, O_RDONLY, 0);
    g = read(e, d, sizeof(d)); ✓
    write(STDOUT_FILENO, d, g);
    close(e); ✓
    exit(0);
}
```

```
g = write(e, d, sizeof(d)); kill(SIGUSR1,x); pipe(e); read(f, E,sizeof(E));
close(f) close(e); read(e, d, sizeof(d)); //linea en blanco
exit(0); write(f, E,sizeof(E)); unlink(CC); g = read(e, d, sizeof(d));
f = open(CC, O_RDWR, 0);
```

Comenzado el	martes, 14 de mayo de 2024, 21:20
Estado	Finalizado
Finalizado en	martes, 14 de mayo de 2024, 22:19
Tiempo empleado	58 minutos 26 segundos
Calificación	4,567 de 15,000 (30,444%)

Pregunta **1**

Parcialmente correcta

Se puntúa 0,500 sobre 1,000

Una lo que corresponda:

La persistencia de una cola de Message Queue	es de sistema de archivos.	✗
La persitencia de los datos de una FIFO	es de proceso.	✓
La persitencia del nombre de una FIFO	es de kernel.	✗
La persistencia de las tuberías	es de proceso.	✓

Pregunta **2**

Sin contestar

Puntúa como 1,500

1) Descargue los siguientes 2 archivos en un mismo directorio:

padre.c : <https://nube.ingenieria.uncuyo.edu.ar/s/D8YRF2eG3rxokp>

hijo.o : <https://nube.ingenieria.uncuyo.edu.ar/s/neKxcFbcmyPMf2d>

2) Modifique el archivo padre.c para que envíe al proceso hijo MENSAJE por la cola de mensajes ya declarada.

3) Ambos archivos se compilan con el siguiente comando en consola:

```
> gcc -c padre.c -lrt && gcc hijo.o padre.o -o padre -lrt
```

4) El binario generado por el comando anterior se ejecuta en consola con:

```
> ./padre
```

5) El proceso hijo a recibir MENSAJE Imprimirá el resultado por consola,

```
Hijo: recibido mensaje esperado
```

```
Hijo: el resultado del ejercicio es XXX
```

6) Copie el número de 3 cifras XXX en la casilla de abajo. Este número es la respuesta del ejercicio.

Respuesta: **×**

Pregunta **3**

Parcialmente correcta

Se puntúa 0,500 sobre 1,000

Indique lo que crea correcto sobre la syscall signal()

Seleccione una o más de una:

- ☐ a. Se utiliza para ignorar la recepción de la señal SIGKILL.
- ☐ b. Se utiliza para indicar qué hacer cuando el proceso recibe algunas señales.
- ☐ c. Se utiliza para especificar qué hacer cuando se recibe la señal SIGSTOP.
- ☒ d. Se utiliza para ignorar la recepción de algunas señales. ✓
- ☐ e. Se utiliza para terminar un proceso del cual se conoce el PID.
- ☐ f. Se utiliza para terminar la ejecución del proceso que recibe la señal.
- ☐ g. Se utiliza para enviar una señal específica a un proceso.

Pregunta **4**

Parcialmente correcta

Se puntúa 0,833 sobre 1,000

Una lo que corresponda referido a transición entre estados de un proceso.

El cambio de estado de Ready a Running es debido a

el scheduler del sistema operativo.



El cambio de estado de Running a Ready es debido a

interrupción de hardware producida por un timer.



El cambio de estado de Running a Blocked es debido a

la ejecución en el proceso de una syscall bloqueante.



El cambio de estado de New a Ready es debido a

que es exitosa la reserva de recursos para el proceso que hace el sistema operativo.



El cambio de estado de Bloqued a Finished es debido a

la ejecución de la syscall return()



El cambio de estado de Blocked a Ready es debido a

interrupción de hardware producida por un periférico.



Pregunta **5**

Correcta

Se puntúa 1,000 sobre 1,000

Indique la afirmación correcta sobre la syscall signal():

Seleccione una:

- ☐ a. Se utiliza para terminar un proceso del cual se conoce el PID.
- ☐ b. Se utiliza para enviar una señal específica a un proceso.
- ☐ c. Se utiliza para ignorar la recepción de la señal SIGSTOP.
- ☐ d. Se utiliza para terminar a ejecución de un proceso.
- ☒ e. Se utiliza para ignorar algunas señales determinadas. ✓
- ☐ f. Ninguna respuesta es válida.
- ☐ g. Se utiliza para poder ignorar todo tipo de señales.

Pregunta **6**

Incorrecta

Se puntúa 0,000 sobre 1,000

Indique qué afirmación es correcta para el algoritmo de planificación Round-Robin.

Seleccione una o más de una:

- ☐ a. Es un algoritmo no expropiativo.
- ☐ b. Disminuye el tiempo de retorno.
- ☒ c. Es un algoritmo equitativo. ✓
- ☐ d. No es necesario que las tareas esten disponibles en el momento inicial.
- ☒ e. Usa técnicas de envejecimiento. ✗
- ☐ f. Si a una tarea en ejecución se le termina el quantum, va al final de la FIFO.

Pregunta **7**

Incorrecta

Se puntúa 0,000 sobre 1,000

¿Cuál de las siguientes, es la principal motivación para el surgimiento de sistemas multitarea?

Seleccione una:

- ☐ a. Aprovechar la protección de hardware para que los procesos queden aislados entre si.
- ☐ b. Poder poner mas de un proceso en memoria principal simultáneamente.
- ☐ c. Mejorar el uso de CPU cuando se espera por operaciones de entrada salida.
- ☐ d. Poder aprovechar el tiempo ocioso del los periféricos, y ejecutar otro proceso.
- ☒ e. Poder ejecutar más de un proceso en pseudo-paralelo. ✗

Pregunta **8**

Sin contestar

Puntúa como 1,500

1) Descargue los siguientes 2 archivos en un mismo directorio:

padre.c : <https://nube.ingenieria.uncuyo.edu.ar/s/X5qjxH67QPpYaHd>

hijo.o : <https://nube.ingenieria.uncuyo.edu.ar/s/dyGMgK7xWwbAcPY>

2) Modifique el archivo padre.c para que envíe al proceso hijo el contenido de la variable tx_buffer mediante la tubería declarada en la variable fd.

3) Ambos archivos se compilan con el siguiente comando en consola:

```
> gcc -c padre.c && gcc hijo.o padre.o -o padre
```

4) El binario generado por el comando anterior se ejecuta en consola con:

```
> ./padre
```

5) El proceso hijo cuando reciba el contenido de tx_buffer por la tubería, imprimirá por consola dos líneas:

```
Leido desde tuberia: HOLA HIJA MIA
```

```
El resultado del ejercicio es: XXX
```

6) Copie el número de 3 cifras XXX en la casilla de abajo. Este número es la respuesta del ejercicio.

Respuesta: **×**

Pregunta **9**

Incorrecta

Se puntúa 0,000 sobre 1,500

¿Cuál de las siguientes afirmaciones describe correctamente una lista en Python?

- ☒ a. Una lista es una colección ordenada y inmutable de elementos, que solo puede contener un **×** tipo de dato.
- ☐ b. Una lista es una colección no ordenada y mutable de elementos, que no permite duplicados.
- ☐ c. Una lista es una colección no ordenada e inmutable de elementos, que permite duplicados.
- ☐ d. Una lista es una colección ordenada y mutable de elementos, que puede contener diferentes tipos de datos y permite elementos duplicados.

Pregunta **10**

Parcialmente correcta

Se puntúa 0,333 sobre 1,000

Seleccione lo correcto para tipos de estructura de sistemas operativos

- ☒ a. Los OS en Capas se basan en organizar las funciones del OS de manera jerárquica. ✓
- ☐ b. Los OS maquina virtual se ejecutan mas rápido que los OS monolíticos.
- ☒ c. Los OS monolíticos escan compuestos por un solo programa y no poseen una estructura interna. ✗
- ☐ d. Los OS Máquina Virtual el hipervisor da suport esolo de multiprogramación, pero no de máquina extendida.
- ☐ e. Los OS en capas son mas robustos que los OS cliente/servidor en cuanto a fallas de drivers.
- ☒ f. Los OS cliente/servidor normalmente se comunican internamente con paso de mensajes. ✓

Pregunta **11**

Incorrecta

Se puntúa 0,000 sobre 1,000

Indique cuál de las siguientes afirmaciones es correcta cuando un proceso padre (A) termina antes que su hijo (B):

Seleccione una o más de una:

- ☒ a. El proceso A se convierte en un proceso zombie, hasta que B lea su exit status. ✗
- ☐ b. El proceso A es adoptado por un proceso mas jerárquico.
- ☐ c. El proceso B es adoptado por un proceso mas jerárquico.
- ☐ d. El proceso A se convierte en un proceso huérfano.
- ☒ e. El proceso B finaliza al recibir la señal SIGCHLD enviada del proceso A. ✗
- ☐ f. En la entrada PCB del proceso B se modifica el campo PPID.
- ☐ g. El proceso B se convierte en un proceso zombie, hasta que A lea su exit status.

Pregunta **12**

Parcialmente correcta

Se puntúa 0,500 sobre 1,000

Para la técnica IPC pipe, una lo que corresponda:

Si un proceso intenta escribir en el extremo de escritura de una tubería que está llena

da error de escritura y termina.



Si un proceso intenta escribir en el extremo de escritura de una tubería que está vacía

el proceso logra escribir.



Si un proceso intenta leer en el extremo de lectura de una tubería que está vacía y todos sus descriptores de escritura fueron cerrados

el proceso lee un EOF (end of file).



Si un proceso intenta leer en el extremo de lectura de una tubería que está llena

el proceso lee datos.



Si un proceso intenta leer en el extremo de lectura de una tubería que está vacía

el proceso lee 0 bytes.



Si un proceso intenta escribir en el extremo de escritura de una tubería que cerró todos los descriptores de lectura

el proceso recibe un SIGPIPE.



Pregunta **13**

Parcialmente correcta

Se puntúa 0,900 sobre 1,500

Dos procesos relacionados se comunican por medio de una tubería.

1. El proceso hijo escribe "0123456789" en la tubería y termina.
2. El proceso padre lee de la tubería y muestra lo leído.

Completar:

```
#define E "0123456789"
int a[2], b,e;
char d[10];
int main (){
    pipe(a); ✓
    b = fork(); ✓
    if (b==0){
        write(a[1], d, sizeof(E)); ✗
        close(a[0]);
        strncpy(d, E, sizeof(E) );
        wait(NULL); ✗
        exit(0);
    }
    close(a[1]);
    e = read(a[0], d, sizeof(d));
    write(STDOUT_FILENO, d, e); ✓
    exit(0);
}
```

wait(NULL);

write(a[1], d, sizeof(E));

close(a[1]);

write(a[0], d, sizeof(d));

close(a[0]);

//linea en blanco

write(STDOUT_FILENO, E, sizeof(E));

write(STDOUT_FILENO, d, e);

read(a[1], d, sizeof(d));

pipe(b);

pipe(a);

b = fork();

Comenzado el	martes, 14 de mayo de 2024, 20:10
Estado	Finalizado
Finalizado en	martes, 14 de mayo de 2024, 21:07
Tiempo empleado	57 minutos 30 segundos
Calificación	12,833 de 15,000 (85,556%)

Pregunta **1**

Correcta

Se puntúa 1,000 sobre 1,000

Indique la o las afirmaciones correctas respecto de la IPC Posix FIFO:

Seleccione una o más de una:

- ☐ a. Se usan para enviar y recibir mensajes.
- ☐ b. Es opcional el uso de un nombre en el sistema de archivos.
- ☒ c. Se pueden utilizar para enviar datos entre procesos relacionados. ✓
- ☐ d. Poseen dos descriptores, uno de lectura y otro de escritura.
- ☐ e. Se destruyen y eliminan al cerrar los descriptores.
- ☒ f. Se pueden abrir en forma bloqueante y no bloqueantes. ✓

Pregunta **2**

Correcta

Se puntúa 1,000 sobre 1,000

Indique la afirmación correcta sobre la syscall signal():

Seleccione una:

- ☒ a. Ninguna respuesta es válida. ✓
- ☐ b. Se utiliza para terminar a ejecución de un proceso.
- ☐ c. Se utiliza para ignorar la recepción de la señal SIGKILL.
- ☐ d. Se utiliza para enviar una señal específica a un proceso.
- ☐ e. Se utiliza para poder ignorar todo tipo de señales.
- ☐ f. Se utiliza para terminar un proceso del cual se conoce el PID.

Pregunta 3

Parcialmente correcta

Se puntúa 0,833 sobre 1,000

Para la técnica IPC pipe, una lo que corresponda:

Si un proceso intenta escribir en el extremo de escritura de una tubería que cerró todos los descriptores de lectura

el proceso recibe un SIGPIPE.



Si un proceso intenta escribir en el extremo de escritura de una tubería que está vacía

el proceso logra escribir.



Si un proceso intenta leer en el extremo de lectura de una tubería que está llena

el proceso lee datos.



Si un proceso intenta leer en el extremo de lectura de una tubería que está vacía

el proceso se bloquea.



Si un proceso intenta leer en el extremo de lectura de una tubería que está vacía y todos sus descriptores de escritura fueron cerrados

da error de lectura y termina.



Si un proceso intenta escribir en el extremo de escritura de una tubería que está llena

el proceso se bloquea.



Pregunta 4

Correcta

Se puntúa 1,000 sobre 1,000

Indique lo correcto para el algoritmo de planificación Fair-Share.

Seleccione una o más de una:

- ☐ a. disminuye el tiempo de retorno
- ☐ b. Es un sistema proporcional para las tareas.
- ☒ c. Es un algoritmo expropiativo. ✓
- ☒ d. Es un sistema proporcional para los usuarios. ✓
- ☐ e. Usa técnicas de envejecimiento.
- ☐ f. El planificador tiene prioridades preestablecidas.

Pregunta 5

Correcta

Se puntúa 1,500 sobre 1,500

En el programa hay 3 procesos.

1. El proceso padre crea un hijo (hijo1) el cual queda en una espera activa.
2. Luego el proceso padre crea otro hijo (hijo2) el cual queda en una espera activa.
3. El proceso padre debe enviar la señal SIGKILL a sus dos hijos y ser el único proceso en ejecutar la línea `printf ("Mi pid es %d\n",getpid());`

Completar:

```
int a,b=0;
int main (){
    a= fork();
    //línea en blanco ✓
    if (a==0) {
        while(1);
        //línea en blanco ✓
        exit(0);
    }
    b=fork(); ✓
    if (b==0) {
        while(1);
        //línea en blanco ✓
        exit(0);
    }
    kill(a,SIGKILL); ✓
    kill(b,SIGKILL);
    printf ("Mi pid es %d\n", getpid());
    sleep(2);
    exit(0);
}
```

kill(a,SIGKILL); signal(SIGKILL, wait(NULL)); //línea en blanco signal(SIGKILL, b=fork();

c=fork(); kill(b,SIGKILL);

Pregunta 6

Correcta

Se puntúa 1,000 sobre 1,000

Seleccione lo correcto respecto a señales

Seleccione una:

- ☐ a. Una señal es la petición por parte del usuario para la creación de un nuevo proceso.
- ☐ b. Las señales tienen persistencia de proceso.
- ☒ c. Una señal es una notificación entregada a un proceso debido a un evento asíncrono. ✓
- ☐ d. Solo se pueden enviar señales entre procesos relacionados.
- ☐ e. Las señales se generan por una interrupción debida a un evento externo.

Pregunta 7

Correcta

Se puntúa 1,500 sobre 1,500

1) Descargue los siguientes 2 archivos en un mismo directorio:

padre.c : <https://nube.ingenieria.uncuyo.edu.ar/s/ec5xw2enpaNpiW8>

hijo.o : <https://nube.ingenieria.uncuyo.edu.ar/s/RaBxsjtGNa8tPjD>

2) Modifique el archivo padre.c para que envíe al proceso hijo el contenido de la variable tx_buffer mediante la FIFO declarada en la variable myfifo.

3) Ambos archivos se compilan con el siguiente comando en consola:

```
> gcc -m32 -c padre.c && gcc -m32 hijo.o padre.o -o padre
```

4) El binario generado por el comando anterior se ejecuta en consola con:

```
> ./padre
```

5) El proceso hijo cuando reciba el contenido de tx_buffer por la FIFO, imprimirá por consola dos líneas:

```
Hijo recibe desde FIFO: HOLA HIJA MIA
```

```
El resultado del ejercicio es: XXX
```

6) Copie el número de 3 cifras XXX en la casilla de abajo. Este número es la respuesta del ejercicio.

Respuesta: ✓

Pregunta 8

Parcialmente correcta

Se puntúa 0,167 sobre 1,000

Indique que información se guarda en cada entrada de la tabla de procesos (PCB):

Seleccione una o más de una:

- ☒ a. TID de los hilos que está usando. ✗
- ☐ b. IPC que usa actualmente el proceso.
- ☐ c. El PID del proceso padre.
- ☒ d. El PID del proceso. ✓
- ☐ e. El PID del proceso hijo.

Pregunta 9

Parcialmente correcta

Se puntúa 0,500 sobre 1,000

Una lo que corresponda.

El cambio de estado de Ready a Running es debido a

el planificador del sistema operativo.



El cambio de estado de Blocked a Ready es debido a

la ejecución de syscall bloqueante.



El cambio de estado de Ready a Finished es debido a

no es válida esa transición.



El cambio de estado de Running a Blocked es debido a

interrupción de hardware producida por un periférico.



El cambio de estado de Running a Ready es debido a

La llegada de una excepción.



El cambio de estado de Blocked a Running es debido a

no es válida esa transición.



Pregunta 10

Correcta

Se puntúa 1,500 sobre 1,500

¿Cuál de las siguientes afirmaciones describe correctamente un método en una clase en Python?

- ☒ a. Un método es una función definida dentro de una clase que describe el comportamiento de los objetos creados a partir de la clase.
- ☐ b. Un método es una variable global que se puede acceder desde cualquier parte del programa.
- ☐ c. Un método es una función anónima definida con la palabra clave `lambda` dentro de una clase.
- ☐ d. Un método es una lista de atributos que describe las propiedades de los objetos creados a partir de la clase.

Pregunta **11****Correcta**

Se puntúa 1,000 sobre 1,000

Indique cuál de las siguientes afirmaciones es verdadera para procesos zombies:

Seleccione una o más de una:

- ☐ a. Solo se conserva su espacio de direcciones reservado por el sistema operativo, hasta que el padre lea su exit status.
- ☒ b. Solo se conserva su exit status en la entrada PCB (Process Control Block) de la Tabla de Procesos. ✓
- ☒ c. Es sacado de memoria completamente cuando el padre ejecuta la función wait(). ✓
- ☐ d. Es un proceso en el cual su padre ha finalizado.
- ☐ e. Solo puede pasar a estado listo al recibir una señal.
- ☐ f. Es un proceso en estado bloqueado esperando un evento.
- ☐ g. Es un proceso en estado listo que puede terminar con SIGKILL.

Pregunta **12****Parcialmente correcta**

Se puntúa 0,333 sobre 1,000

ordene los pasos en los que se lleva a cabo la llamada a sistema POSIX read()

- | | | |
|---|--|---|
| 1 | Ejecución de la call gate. | ✗ |
| 2 | Ejecución de la task gate. | ✗ |
| 3 | Ejecución del controlador de periférico de espacio usuario. | ✗ |
| 4 | Ejecución en una dirección fija del controlador de periférico de espacio kernel. | ✓ |
| 5 | Ejecución del procedimiento de la biblioteca. | ✗ |
| 6 | Retorno del procedimiento de biblioteca al programa de usuario. | ✓ |

Pregunta **13****Correcta**

Se puntúa 1,500 sobre 1,500

1) Descargue los siguientes 2 archivos en un mismo directorio:

padre.c : <https://nube.ingenieria.uncuyo.edu.ar/s/GdFrdd22bLJ8ttL>

hijo.o : <https://nube.ingenieria.uncuyo.edu.ar/s/o5tGaWKEwfnYXmY>

2) Modifique el archivo padre.c para que envíe las siguientes señales al proceso hijo:

1. SIGUSR1
2. SIGUSR2
3. SIGFPE

Debe enviar las señales de a una a la vez.

3) Ambos archivos se compilan con el siguiente comando en consola:

```
> gcc -m32 -c padre.c && gcc -m32 hijo.o padre.o -o padre
```

4) El binario generado por el comando anterior se ejecuta en consola con:

```
> ./padre
```

5) Cuando el proceso hijo reciba la señal indicada, imprimirá por consola dos líneas:

```
Hijo recibe SIGUSR1
```

```
El resultado del ejercicio es: XXX
```

6) Empareje los resultados según corresponda.

SIGFPE	<input type="text" value="541"/>	✓
SIGUSR2	<input type="text" value="431"/>	✓
SIGUSR1	<input type="text" value="117"/>	✓

Comenzado el	martes, 14 de mayo de 2024, 21:20
Estado	Finalizado
Finalizado en	martes, 14 de mayo de 2024, 22:13
Tiempo empleado	53 minutos 14 segundos
Calificación	11,425 de 15,000 (76,167%)

Pregunta 1

Incorrecta

Se puntúa 0,000 sobre 1,000

Indique cuál de las siguientes afirmaciones es verdadera para procesos zombies:

Seleccione una o más de una:

- ☐ a. Es un proceso en estado bloqueado esperando un evento.
- ☐ b. Solo se conserva su exit status en la entrada PCB (Process Control Block) de la Tabla de Procesos.
- ☐ c. Es sacado de memoria completamente cuando el padre ejecuta la función wait().
- ☒ d. Es un proceso en estado listo que puede terminar con SIGKILL. ✖
- ☐ e. Solo se conserva su espacio de direcciones reservado por el sistema operativo, hasta que el padre lea su exit status.
- ☐ f. Es un proceso en el cual su padre ha finalizado.
- ☐ g. Solo puede pasar a estado listo al recibir una señal.

Pregunta 2

Correcta

Se puntúa 1,000 sobre 1,000

Seleccione lo correcto respecto a señales

Seleccione una o más de una:

- ☒ a. Una señal es una notificación entregada a un proceso debido a un evento asíncrono. ✔
- ☒ b. No todas las señales pueden ser ignoradas por un proceso. ✔
- ☐ c. Un proceso puede bloquear la recepción de todas las señales.
- ☐ d. Solo se pueden enviar señales entre procesos relacionados (padre-hijo-nieto).
- ☐ e. Una señal es la petición por parte del usuario para la creación de un nuevo proceso.

Pregunta 3

Parcialmente correcta

Se puntúa 0,500 sobre 1,000

Seleccione lo correcto respecto a señales

Seleccione una o más de una:

- ☒ a. Una señal se usa generalmente para comunicar estados entre dos procesos. ✓
- ☐ b. Se pueden enviar señales entre procesos no relacionados, a excepción de SIGSTOP y SIGKILL.
- ☐ c. Las señales se pueden generar con la syscall kill(), siempre y cuando se tengan permisos.
- ☐ d. Una señal es la petición por parte del usuario para la terminación de un proceso.
- ☐ e. Un proceso puede bloquear la recepción de todas las señales.

Pregunta 4

Correcta

Se puntúa 1,000 sobre 1,000

¿Qué es un proceso?

Seleccione una:

- ☐ a. Ninguna de las respuestas es válida.
- ☐ b. Un proceso es una instancia de un programa en ejecución que solo se puede ejecutar en espacio de kernel.
- ☐ c. Un proceso es un programa listo para ser ejecutado.
- ☐ d. Un proceso es el número que identifica a un programa en ejecución.
- ☒ e. Un proceso es una instancia de un programa en ejecución mas el estado del mismo. ✓
- ☐ f. Un proceso es una instancia de un programa en ejecución que solo puede ejecutar invocando el sistema operativo.

Pregunta 5

Parcialmente correcta

Se puntúa 0,800 sobre 1,000

Una la que corresponda, respecto a terminación de un proceso.

Si el proceso recibe la señal SIGKILL, realiza una

terminación involuntaria por otro proceso.



Si el proceso hace una llamada a sistema que retorna -1, y luego una llamada exit(status) con status distinto de 0, realiza una

terminación con error voluntaria.



Si el proceso intenta acceder por medio de un puntero a una posición fuera de su espacio de direcciones, realiza una

terminación involuntaria por error fatal.



Si el proceso hace una llamada a sistema return(-0) realiza una

terminación normal voluntaria.



Si el proceso se bloquea esperando entrada salida, realiza una

terminación involuntaria por otro proceso.



Pregunta 6

Correcta

Se puntúa 1,500 sobre 1,500

1) Descargue los siguientes 2 archivos en un mismo directorio:

padre.c : <https://nube.ingenieria.uncuyo.edu.ar/s/D8YRF2eG3rxokp>

hijo.o : <https://nube.ingenieria.uncuyo.edu.ar/s/tdPBeMnExsJNyin>

2) Modifique el archivo padre.c para que envíe al proceso hijo MENSAJE por la cola de mensajes ya declarada.

3) Ambos archivos se compilan con el siguiente comando en consola:

```
> gcc -c padre.c -lrt && gcc hijo.o padre.o -o padre -lrt
```

4) El binario generado por el comando anterior se ejecuta en consola con:

```
> ./padre
```

5) El proceso hijo a recibir MENSAJE Imprimirá el resultado por consola,

```
Hijo: recibido mensaje esperado
```

```
Hijo: el resultado del ejercicio es XXX
```

6) Copie el número de 3 cifras XXX en la casilla de abajo. Este número es la respuesta del ejercicio.

Respuesta:

Pregunta 7

Correcta

Se puntúa 1,500 sobre 1,500

1) Descargue los siguientes 2 archivos en un mismo directorio:

padre.c : <https://nube.ingenieria.uncuyo.edu.ar/s/X5qjxH67QPPYaHd>

hijo.o : <https://nube.ingenieria.uncuyo.edu.ar/s/9EYKtNRMcSsSqi7>

2) Modifique el archivo padre.c para que envíe al proceso hijo el contenido de la variable tx_buffer mediante la tubería declarada en la variable fd.

3) Ambos archivos se compilan con el siguiente comando en consola:

```
> gcc -m32 -c padre.c && gcc -m32 hijo.o padre.o -o padre
```

4) El binario generado por el comando anterior se ejecuta en consola con:

```
> ./padre
```

5) El proceso hijo cuando reciba el contenido de tx_buffer por la tubería, imprimirá por consola dos líneas:

```
Leido desde tuberia: HOLA HIJA MIA
```

```
El resultado del ejercicio es: XXX
```

6) Copie el número de 3 cifras XXX en la casilla de abajo. Este número es la respuesta del ejercicio.

Respuesta: ✓

Pregunta 8

Correcta

Se puntúa 1,000 sobre 1,000

Indique cuál de las siguientes afirmaciones es correcta cuando un proceso padre (A) termina antes que su hijo (B):

Seleccione una o más de una:


- ☐ a. El proceso A se convierte en un proceso huérfano.
- ☒ b. El proceso B se convierte en un proceso huérfano. ✓
- ☐ c. El proceso B finaliza al recibir la señal SIGCHLD del proceso A.
- ☐ d. El proceso A es adoptado por el proceso init.
- ☐ e. El proceso A se convierte en un proceso zombie.
- ☒ f. El proceso B es adoptado por el proceso init. ✓
- ☐ g. El proceso B se convierte en un proceso zombie.

Pregunta 9

Correcta

Se puntúa 1,000 sobre 1,000

Indique qué afirmación es correcta respecto a la planificación de procesos.

- ☐ a. Cuanto mayor sea el quantum que utilizan los planificadores, más interactivo será el sistema operativo.
- ☐ b. Al crearse hilos en espacio usuario, se deben modificar los planificadores para incluirlos.
- ☐ c. Los planificadores expropiativos solo se ejecutan cuando el proceso en estado READY ejecuta una syscall.
- ☒ d. Al poder tener en memoria más de un proceso a la vez, es necesario el uso de planificadores. 
- ☐ e. Con un planificador no expropiativo se ven beneficiadas las tareas I/O bounded.


Pregunta 10

Incorrecta

Se puntúa 0,000 sobre 1,500

¿Qué salida produce el siguiente código en Python?

```
1 x = 7
2 y = 10
3
4 if x > 5 and y < 15:
5     if x % 2 == 0:
6         print("A")
7     else:
8         print("B")
9 elif x <= 5:
10    print("C")
11 else:
12    print("D")
13
```

- ☐ a. A
- ☐ b. C
- ☒ c. D 
- ☐ d. B

Pregunta **11**

Correcta

Se puntúa 1,000 sobre 1,000

!Seleccione las afirmaciones correctas respecto a Posix Pipes:

Seleccione una o más de una:

- ☐ a. Poseen un nombre en el sistema de archivos.
- ☐ b. Se pueden abrir en forma bloqueantes y no bloqueantes.
- ☒ c. Permiten el intercambio de datos entre procesos relacionados. ✓
- ☒ d. Se destruyen al cerrar todos los descriptores de lectura y escritura. ✓
- ☐ e. Poseen persistencia de kernel.
- ☐ f. Permiten sincronizar procesos no relacionados.
- ☐ g. Se destruyen al cerrar los descriptores de escritura.
- ☐ h. Se utilizan para enviar datos entre procesos no relacionados.

Pregunta **12**

Correcta

Se puntúa 1,000 sobre 1,000

Una lo que corresponda:

La persitencia del nombre de una FIFO	es de sistema de archivos.	✓
La persistencia de las tuberías	es de proceso.	✓
La persitencia de los datos de una FIFO	es de proceso.	✓
La persistencia de una cola de Message Queue	es de kernel.	✓

Pregunta 13

Parcialmente correcta

Se puntúa 1,125 sobre 1,500

Dos procesos relacionados se comunican por medio de una cola de mensajes.
 El proceso padre crea la cola de mensajes y escribe un mensaje.
 El proceso hijo lee un mensaje, muestra lo leído y elimina la cola de mensajes. Completar

```
#include <mqueue.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <pthread.h>

#define AAAA "Parcial TD III"
#define MQTD3 "/TD3"

int err, rc,t;
char buff[20];
mqd_t mqd;
struct mq_attr attr;

int main() {
    attr.mq_msgsize = sizeof(buff);
    //linea en blanco ✖
    mqd = mq_open(MQTD3, O_RDWR | O_CREAT, 0777, &attr);
    if (fork() == 0) {
        mq_getattr(mqd, &attr);
        mq_receive(mqd, buff, attr.mq_msgsize, 0);
        printf("%s\n", buff);
        mq_close(mqd);
        exit(0); }
    mq_send(mqd, AAAA, strlen(AAAA), 1);
    wait(NULL);
    mq_close(mqd);
    mq_unlink(MQTD3); ✔
    exit(0); }

mq_close(mqd);

attr.mq_maxmsg = 10;

mq_getattr(mqd, &attr_rcv.mq_msgsize, //linea en blanco

attr.mq_msgsize = sizeof(buff);

mqd = mq_open(MQTD3, O_RDWR, 0);
```