Presentación

Esta PEC plantea una serie de actividades con el objetivo que el estudiante se familiarice con la temática de los últimos módulos de la asignatura. Cada pregunta indica un peso de la pregunta respecto a la evaluación final de esta entrega.

Competencias

Transversales:

- Capacidad para adaptarse a las tecnologías y a los futuros entornos actualizando las competencias profesionales

Específicas:

- Capacidad de analizar un problema en el nivel de abstracción adecuado a cada situación y aplicar las habilidades y conocimientos adquiridos para abordarlo y resolverlo
- Capacidad de diseñar y construir aplicaciones informáticas mediante técnicas de desarrollo, integración y reutilización

Enunciado

1. **Módulo 5**: El Sistema de Ficheros. (Peso 15%+15%) .

   1.1 .Explica la diferencia entre un enlace duro(hard link) y un enlace simbólico (symbolic link).

   Solución: Un hard-link es una copia total o parcial de la entrada del directorio asociada al archivo o subdirectorio que se desea compartir. Por su parte, un symbolic-link es un tipo especial de archivo que contiene el nombre de ruta del archivo o subdirectorio compartido, es decir, la ubicación del archivo dentro de la estructura de directorios.

   1.2 Selecciona la respuesta correcta a las siguientes preguntas. Justificar la respuesta. (Solución en **negrita**):

   - ¿ Qué información NO se almacena en un fichero ejecutable?
     a. El valor inicial del contador de programa.
     b. Las variables globales con valor inicial.
     c. **Las variables locales con valor inicial.**

     Justif: En un fichero ejecutable nunca se almacena información sobre las variables locales, como tampoco sus valores. En general, un fichero ejecutable incluye información como el valor inicial del contador de programa, las variables globales con valor inicial y otra información necesaria para ejecutar el programa

   - Una partición en un sistema de Ficheros
     a. Permite tener a varios nombres asociados a un mismo archivo.
     b. Es una división lógica de las unidades de almacenamiento de memoria secundaria que no requieren de ningún soporte de hardware adicional

     Justif: Es su propia definición. Es decir una división lógica de las unidades de almacenamiento de memoria secundaria. Nunca puede tener archivos de tamaño ilimitado , ya que está limitado al de la partición. Las particiones nunca tienen soporte de hardware adicional.

     c. Permite tener archivos de tamaño ilimitado.

   - Al intentar ejecutar un programa con un sistema operativo Unix, el intérprete de comandos muestra el siguiente error: “acceso denegado”. El fichero se llama test, sus permisos son: -rw - - - - - - - , y pertenece al usuario que lo ha intentado ejecutar. ¿Qué debe ejecutar el usuario para poder ejecutar este fichero?

     a. Modificar sus permisos mediante el comando *chmod u+x test.*

Justif: Con esta Instrucción se puede añadir el permiso de ejecución. Con esto el usuario añadirá permisos de ejecución a los de lectura y escritura que ya posee. Inicialmente no tenía permisos de ejecución.

b. Modificar sus permisos mediante el comando *chmod u-x test.*
c. Añadir el directorio actual a la variable de entorno *PATH.*
d. Cambiarle la extensión ejecutando *mv test test.exe.*

2. **Módulo 6**: La Gestión de Procesos. (Peso 10%+20%)

2.1. Selecciona la respuesta correcta a las siguientes preguntas (Solución en **negrita).**Justifica las respuestas**:**

- ¿Cuál de las siguientes sentencias es cierta? Justificar la respuesta.
  a. Un proceso en estado ready es cuando está ejecutándose.
  b. La transición de ready a run se dará siempre que se desbloquee un proceso.
  c. **Un proceso puede iniciar por sí mismo la transición de estado run a bloqueado.**

     Justif: Por ejemplo en la finalización de una operación de E/S, esper esperando la recepción señal específica, esperando un recurso compartido o bien que esperando que el usuario proporcione una entrada. Esto ocurre cuando el proceso necesita acceder a un recurso que no está disponible de inmediato, como un archivo o una impresora.

- La transición del estado ready a bloqueado se dará:
  a. Cuando se expire un Quantum.
  b. Si enviamos una signal SIGSTOP.

     Justif: Con este Signal obligamos al SO a bloquear un proceso.

  c. Cuando termine una operación de Entrada/Salida
  d. Cuando entre un proceso nuevo en estado ready

- En un sistema Operativo existe:
  a. Solo un PCB que corresponde al proceso en estado Run en cada momento.
  b. PCB's solo los procesos en estado blocked.
  c. Un PCB por cada proceso existente.

     Justif: Cada proceso sólo puede representar un PCB. Este PCB contendrá información específica del proceso: identificador, estado entre otros.

  d. PCB's solo los procesos en estado ready.

- ¿Es posible que un programa esté asociado simultáneamente a dos procesos?
  a. **Si**

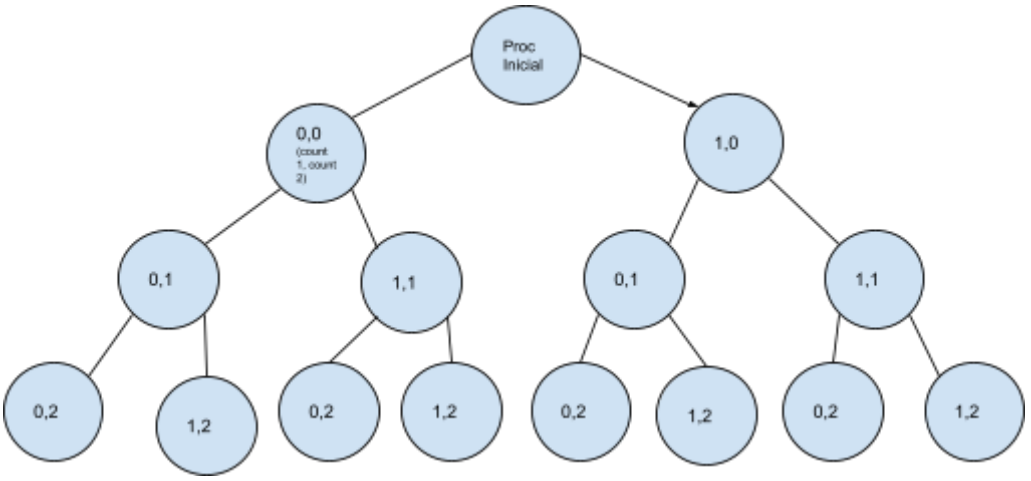
     Justif: Con la ejecución de varias instancias se generan nuevos procesos del mismo programa. Por ejemplo con la instrucción fork, creando una copia exacta. Recordamos que un proceso es una instancia de un programa. Varias instancias corresponden a varios procesos.

  b. Solo es posible en los sistemas operativos que soportan múltiples hilos
  c. Si pero el programa debe estar diseñado para la utilización simultánea de dos procesos
  d. Si pero el sistema debería tener dos procesadores

2.2. Estudia el siguiente código y dibuja la jerarquía de procesos resultante.

```
#include <sys/types.h>
 #include <sys/wait.h>
 #include <unistd.h>
#include  <stdio,h>
#include  <stdlib.h>
#define L1 2
#define L2 3

int main (int argc, char *argv[]) {
  int cont1, cont2;
  pid_t pid;
  for (cont2= 0; cont2< L2; cont2++) {
     for (cont1= 0; cont1< L1; cont1++) {
        pid= fork();
        if (pid== 0)
          break;
        }
       if (pid!= 0)
        break;
       } return 0;
}
```



3. **Módulo 7**:La Concurrencia y la Comunicación. (Peso 10%+15%+15%)

3.1 En un sistema en el que X procesos compiten por Y recursos de memoria, uno de los procesos necesita utilizar de forma simultánea 2, pero el resto de procesos los usa de uno en uno. Se asume que todos los procesos usan recursos durante un tiempo limitado. Selecciona la opción correcta (Solución en **negrita).** Justificar la respuesta:

a. Puede producirse interbloqueo si la reserva de los recursos no se realiza en una operación indivisible.
b. **No se puede producir interbloqueo en ningún caso**.

Justifi: Esto es debido al hecho de que el proceso que usa dos recursos de manera simultánea puede siempre esperar a que se liberen los recursos; y los que usan recursos de uno en uno no requieren, en principio, que otros procesos liberen recursos añadidos.

c. Puede producirse interbloqueo si X > Y.
d. En cualquier momento se puede puede producir un interbloqueo.

3.2 En un túnel de lavado hay tres tipos de servicios: Básico, Completo y Deluxe. Los clientes acceden al negocio, y esperan a que les atienda el encargado para que les indique donde dejar el coche dependiendo del servicio contratado. De forma que el servicio Básico tiene una zona con capacidad para 20 coches, Completo tiene otra zona con capacidad de 6 coches y Deluxe No espera. Realizar un pseudocódigo de forma que utilizando semáforos coordine las tareas de los clientes. En este pseudocódigo tendrás que definir el proceso Cliente() y la declaración de los semáforos como variables globales. Utilizar las instrucciones de semáforos sem_init, sem_wait y sem_signal.

Solución:

Programa Tunel_lavado;

```
/* Declaración de los Semaforos */
 encargado,  zona_basica, zona_completo: semaforo;

process cliente ()

        while true do
                sem_wait(encargado)
                indica_servicio
                sem_signal(encargado);
                if servicio = Basico then
                        sem_wait(zona_basica);
                        {Limpiar_Coche_Basico}
                        sem_signal(zona_basica);
                end;
                if servicio = completo then
                        sem_wait(zona_completo);
                        {Limpiar_Coche_Completo}
                        sem_signal(zona_completo);
                end;
                if servicio = deluxe then
                        {Limpiar_Coche_Deluxe}
                end;
        end;
end;

/* Inicializar Semaforos */
sem_init(encargado, 1);
sem_init(zona_basica, 20);
sem_init(zona_completo, 6);
/* ejecucion de procesos clienes */
cliente;
end;
```

3.3 En una oficina municipal de atención al ciudadano existen N ventanillas. Cuando un ciudadano entra en la oficina para realizar alguna gestión debe guardar una única cola hasta que alguna ventanilla queda libre. Rellenar el siguiente pseudocódigo de un programa basado en C que usando semáforos binarios coordine la actividad de los ciudadanos en la oficina. El programa debe tener dos partes: pseudocódigo del proceso ciudadano y pseudocódigo de la función principal para inicializar los semáforos y lanzar la ejecución concurrente de los procesos.
Recordar que un semáforo binario únicamente soporta las operaciones:
–sem_init(S,valor), donde valor puede tomar los valores 0 o 1.
–sem_wait(S)
–sem_signal(S)

Solución (en negrita):

```
int contador=0; /*Definición e inicialización de la variable global contador de ciudadanos
en cola o bien en ventanilla */
semáforo_binario S1, S2,mutex_entrada; /*Definición semáforos binarios */
void ciudadano() /*Proceso ciudadano */
{
  sem_wait(mutex_entrada);
  sem_wait(S1);
  contador = contador + 1;
  if (contador > N){
   sem_signal(S1);
   sem_wait(S2); /*Esperar a que haya una ventanilla libre */
   }
  else sem_signal(S1);
  sem_signal(mutex_entrada);

  realizar_gestión();

  sem_wait(S1);
  if (contador > N) sem_signal(S2); /*Queda una ventanilla libre */

  contador = contador - 1;
 sem_signal(S1);
}
main() /*Inicialización semáforos y ejecución concurrente */
{
sem_init(S1,1);
sem_init(S2,0);
ejecución_concurrente(ciudadano,...,ciudadano);
}
```

Recursos

- Módulos 5, 6 y 7 de la asignatura.
- Documento "Introducción a la programación de Unix" (disponible en el aula) o cualquier otro manual similar.
- El aula "Laboratorio de Sistemas Operativos" (podéis plantear vuestras dudas relativas al entorno UNIX, programación,...).

Criterios de evaluación

Se valorará la justificación de las respuestas presentadas.

El peso de cada pregunta está indicado en el enunciado.

Formato y fecha de entrega

Se entregará un fichero zip que contendrá un fichero pdf con la respuesta a las preguntas y, si es preciso, los ficheros adicionales que queráis entregar.

Fecha límite de entrega: 24:00 del 09 de Enero de 2022.