SISTEMAS OPERATIVOS
2º Curso – Dobles Grados
Ejercicios del Tema 1

Tema 1:

1. Uno de los mecanismos de virtualización ligera que suministra el kernel de Linux es el de *namespaces*. Indicar cual es la función básica del citado mecanismo.

   Los nombres para referencias ciertos recursos/servicios del sistema se pueden aislar en espacios separados al objeto de que desde un espacio de nombre no se puedan referenciar recursos/servicios de otro espacio de cara a aislarlos.

2. En relación a la arquitectura monolítica:
   a) Cuál es la principal razón para que esta arquitectura sea la "arquitectura de ejecución" más utilizada en los sistemas operativos más comúnmente utilizados, y cuál es el principal inconveniente.

      Ventaja: Eficiencia - el coste de un servicio del sistema supone una única llamada al sistema y su retorno y no hay cambio de proceso para obtener ese servicio.

      Inconveniente: no aislamiento de errores

   b) Las estructuras monolíticas actuales, como ocurre en Linux y Mac OS X, permiten algún mecanismo de adaptación de la funcionalidad del kernel en tiempo de ejecución Indicar qué mecanismo es y en qué consiste.

      Módulo de carga dinámica o LKM. Se puede añadir funcionalidad en ejecución construyendo un fragmento de código objeto que se enlaza en tiempo de carga (`insmod`) con el resto del kernel para añadir funcionalidad.

      En iOS, los extend.

3. Explicar cuales son las diferencias entre *hipervisores nativos* (Tipo I) e *hipervisores anfitrión* (Tipo II). ¿Cómo se hace para que un SO que esta diseñado para ejecutarse en modo kernel/supervisor se ejecute en otro nivel de privilegio?
   Tipo I: se ejecutan sobre el hardware; Tipo II: se ejecutan sobre un sistema anfitrión. Paravirtualización, traducción binaria de código o soporte hardware

4. Los kernel actuales de Linux incluyen un mecanismo denominado *espacios de nombres (namespaces)*. Indicar cual es la función básica del citado mecanismo y en concreto la del *PID namespaces*.
   Función, en pregunta 1. Como en virtualización deseamos que los procesos de una MV no vean los procesos de otra, cada MV tiene su espacio de nombres. De esta forma un proceso en la MV1 no puede afectar al procesos de MV2 pues no conoce/sabe cuales son PIDs.

5. Explicar cual es el principal inconveniente de una arquitectura monolítica.
   El kernel del SO es un único ejecutable por lo que no existe aislamiento de errores dentro de él; si un componente falla, este error puede propagarse a otros componente y

provocar que el sistema se 'cuelgue'.

6. Los kernel actuales de Linux soportan dos mecanismos básicos para la virtualización:
   a) Indicar que mecanismos son y qué responsabilidad tienen cada uno desde el punto de vista de la gestión de recursos.
      Namespaces: aislar
      Grupos de control: asignar/limitar recursos
   b) Cuál es el objetivo de los *espacios de nombres de identificadores de procesos* (*pid namespaces*), y cómo se consigue.
      Visto en ejercicio anterior

7. Respecto a los tipos de sistemas operativos, indicar:
   a) Cómo se implementaría un sistema de tiempo compartido.
      Visto en teoría – transparencia 29
   b) Qué caracteriza a un sistema operativo SMP (Symmetric Multi-Processing).
      Cualquier procesador/nucleo/hiper-hebra puede ejecutar cualquier programa o función del sistema operativo.

8. La mayoría de los sistemas operativos de producción actualmente en uso tienen una estructura de ejecución monolítica, soportan multiprocesamiento simétrico y tienen soporte para tiempo-real. Rellenar la Tabla que se muestra a continuación indicando en cada celda cuales son las razones de hacerlo así y cuales son las implicaciones en su construcción.

	Arquitectura de ejecución monolítica	Multiprocesamiento simétrico (SMP)	Soporte para tiempo-real
Por qué ...	**Eficiencia**	**Mayor productividad, todas las CPU puede ejecutar cualquier cosa (no hay cuellos de botella).**	**Dar soporte a aplicaciones donde el tiempo de respuesta esta acotado**
Implicaciones en su construcción ...	**La implementación presenta cierta complejidad debido a las dependencias entre funciones/servicios. El SO es grande y siempre tiene que estar cargado en memoria.**	**Más complejos de construir ya que debemos asegurarnos que si dos/más procesos se ejecutan diferentes CPUs y ejecutan la misma función, sus ejecuciones no interfieran (evitar condiciones de carrera)**	**Más complejos de construir: los sistemas convencionales no están pensados para tener un *deadline* asociado a las tareas sino que el sistema los ejecuta de cara a tener una mayor productividad y un tiempo de respuesta más o menos aceptable (no acotado). Esto afecta al planificador, a la propia construcción del kernel como el tratamiento de interrupciones interrupciones, etc.**