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

Tema 2:

1. Si invocamos la función `clone()` con los argumentos que se muestran a continuación, indicar que tipo de hebra/proceso estamos creando. Justificarlo mostrando como quedarían los descriptores de las tareas.

```
clone(funcion,...,CLONE_VM|CLONE_FILES|CLONE_FS|CLONE_TRHEAD, ...)
```

2. La llamada al sistema `clone()` en Linux se utiliza tanto para crear procesos como hilos (hebras). Indicar cómo la utilizaríamos, y dibujar los descriptores de procesos, cuando:
   a) Un proceso crea otro proceso independiente.
   b) Un proceso crea un hilo de la misma tarea.

3. El algoritmo de planificación *CFS* (*Completely Fair Scheduler*) de Linux reparte imparcialmente la CPU entre los procesos de la clase correspondiente. Justificar como es posible que un proceso que realiza muchas entradas-salidas (por tanto, tiene ráfagas de CPU muy cortas) obtenga un trato imparcial respecto de un proceso acotado por computo (pocas entradas-salidas y que consume las ráfagas asignadas completamente). Para el razonamiento, podemos suponer que solo hay un proceso de cada tipo y que la prioridad base de ambos es 120.

4. Explicar cómo se lleva a cabo la finalización de un proceso/hebra por parte del kernel de Linux tras una invocación explícita de la llamada al sistema exit() hasta la liberación de todos los recursos usados por el mismo.

5. En un sistema Linux pueden existir procesos que pertenecen a diferentes clases de planificación. Indicar:
   a) ¿Qué clases son estas y que algoritmo implementan?
   b) Si en un momento dado, hay en el sistema al menos un proceso que pertenece a cada una de las clases, indicar en que orden se planificarán.
   c) Si en la clase de tiempo compartido tenemos tres procesos: P1 y P2 con prioridad 120 y P3 con prioridad 110 ¿qué porcentaje de CPU la asignará en planificador a cada uno sabiendo que la prioridad 120 tiene un peso de 1024 y a la prioridad 110 le corresponde un peso de 110?

6. Entre los diferentes pasos que sigue el planificador de Linux, función `schedule()`, está el de seleccionar la siguiente tarea a ejecutar (función `pick_next_task()`). A la vista de lo estudiado, esbozar los pasos que debe seguir esta función para seleccionar el proceso teniendo en cuenta que existen tres clases de planificación (`SCHED_FIFO`, `SCHED_RR` y `SCHES_NORMAL`) y de las características/propiedades de los procesos dentro de cada clase.

7. Suponga un sistema Linux con dos procesos: uno de tiempo-real de la clase SCHED_RR que está actualmente bloqueado; otro de la clase CFS que esta en ejecución usando un recurso compartido del sistema. Cuando el proceso de tiempo-real se desbloquea, indicar la secuencia de eventos que se producen hasta que pasa a ejecutarse sabiendo que debe usar el

recurso que esta usando ahora el proceso de tiempo compartido.

8. ¿Qué son los *Gobernadores* (*Governors*) en un sistema Linux? Indicar cuales son los dos gobernadores de usuario y su función.

9. El planificador a medio plazo de Linux implementa una política apropiativa denominada "*apropiatividad mediante puntos de apropiación*".
   a) ¿En qué consiste y/o cómo se implementa?
   b) ¿Qué ventajas e inconvenientes presenta respecto a una política totalmente apropiativa?

10. En el algoritmo de planificación de la clase CFS siempre se elige como siguiente proceso para ejecución al proceso cuyo `vruntime` es menor. Indicar que representa este parámetro y como se calcula.

11. Justificar por qué el kernel de Linux no es apto para la ejecución de aplicaciones de tiempo real duras, es decir que tienen plazos estrictos de ejecución. Observación, recordad la definición de latencia de apropiación y la implementación que hace de esta el kernel.

12. ¿Qué mecanismos utiliza el kernel de Linux para gestionar el consumo de energía de los procesadores?

13. Sobre los grupos de control (*cgroups*) en Linux:
   a) ¿Qué utilidades tiene esta construcción?
   b) ¿Cómo dan soporte a la virtualización?
   c) Indicar al menos tres subsistemas de grupos de control.