PROGRAMACIÓN DE SERVICIOS Y PROCESOS

Unidad Didáctica 01: Programación multiproceso

Actividad: Ataque a fichero encriptado con multiproceso en C

Resumen de planificación de actividad

Objetivos	Contenidos que se trabajan
• Poner en práctica la creación y sincronización de procesos en sistemas Linux mediante el lenguaje de programación C	• Ejecutables. Procesos. Servicios. • Programación concurrente. • Creación de procesos. • Comunicación entre procesos. • Gestión de procesos. • Comandos para la gestión de procesos en sistemas libres y propietarios. • Sincronización entre procesos. • Programación de aplicaciones multiproceso. • Documentación. • Depuración.
Objetivos didácticos	**Criterios de evaluación**
RA1: Desarrolla aplicaciones compuestas por varios procesos reconociendo y aplicando principios de programación paralela.	c) Se han analizado las características de los procesos y de su ejecución por el sistema operativo. e) Se han utilizado clases para programar aplicaciones que crean subprocesos. f) Se han utilizado mecanismos para sincronizar y obtener el valor devuelto por los subprocesos iniciados. g) Se han desarrollado aplicaciones que gestionen y utilicen procesos para la ejecución de varias tareas en paralelo. h) Se han depurado y documentado las aplicaciones desarrolladas.

Recursos necesarios	Agrupamiento	Duración estimada
• Ordenador Ubuntu con acceso a Internet	Grupo clase	1 sesión

Secuencia de desarrollo de la actividad

1. Planteamiento y objetivos.
2. Resolución de dudas

INSTRUCCIONES

Esta actividad se hará de forma individual.

Entrega

Los ficheros de código fuente necesarios.

ACTIVIADAD: ATAQUE CONTRA FICHERO ZIP PROTEGIDO CON PROGRAMA MULTIPROCESO EN C

Hemos de desarrollar un pequeño programa en **C bajo Linux** que realice un ataque contra un fichero zip protegido con contraseña. Para ello, tendremos un proceso principal que tendrá definido un array con el alfabeto de caracteres a usar para componer las claves del ataque. Este alfabeto de caracteres contendrá los números del 0 al 9, y las letras mayúsculas del alfabeto (sin incluir la Ñ).

Para simplificar, todas las claves que compongamos para el ataque con este alfabeto tendrán 3 caracteres de longitud. Esté será el alfabeto a usar para el ataque:

```
char alfabeto[] = {
   'A','B','C','D','E','F','G','H','I','J','K','L',
   'M','N','O','P','Q','R','S','T','U','V','W','X',
   'Y','Z','0','1','2','3','4','5','6','7','8','9'};
```

Para realizar el ataque, el proceso padre **creará tres procesos hijos** e intentará repartir la carga del ataque entre los tres. A cada proceso hijo le indicará, con algún mecanismo de comunicación de procesos, el rango de claves del ataque. Por ejemplo al primer proceso le puede ordenar que use desde el primer carácter del alfabeto al 12 con lo que el proceso probará sucesivamente las claves AAA, AAB, AAC, hasta llegar a L99.

Cada proceso hijo irá componiendo claves y con cada clave intentará desencriptar el fichero mediante la herramienta *unzip*. Por ejemplo para atacar al fichero *secretos.zip* usará el siguiente comando del sistema operativo:

unzip -oq -P clave secretos.zip

Si un proceso hijo encuentra la clave, la imprimirá por pantalla.

Al finalizar cada proceso hijo le devolverá al padre (aunque este no use el valor devuelto):

- -1: si no ha conseguido encontrar la clave o si ha ocurrido algún error
- 0: si ha conseguido encontrar la clave

El proceso padre esperará a que finalicen todos hijos y devolverá:

- -1: si ha ocurrido algún error en la creación de los procesos hijos
- 0: si ha realizado el ataque

Mejoras (opcional):

1. Que el nombre del fichero zip a atacar se le suministre por línea de comandos al ejecutable.
2. Que los procesos hijos notifiquen al padre que se ha encontrado la contraseña para que este finalice su ejecución y la del resto de hijos.