Examen de Sistemas Operativos

Junio 2009

(2 puntos por cuestión)

1) Escribir un programa en C que sea capaz de ejecutar los dos programas cuyos nombres se le pasan como argumentos desde la línea de comandos, y que pinte un mensaje de aviso cuando los dos programas hayan finalizado. Si el programa se llama `ejecuta` el siguiente ejemplo ejecutaría los programas `prog1`, y `prog2`:

   ```
   [xxxxx]$ ejecuta prog1 prog2
   ```

2) Escribir una función que sirva para concatenar dos ficheros. La función recibe los nombres de los dos ficheros como parámetros y debe añadir al fichero que se pasa como primer parámetro el fichero que se pasa en segundo lugar. La función retorna el error producido o un 0 si va bien.

   ```
   int concatena(char *f1, char*f2);
   ```

3) Escribir un programa que cree tres threads. El primero debe llamar a la función `calcula_uno` y depositar el resultado en un variable global. El segundo debe llamar a la función `calcula_dos` y depositar el resultado en otra variable global. El tercero debe esperar a tener la información producida por los otros threads en las llamadas a `calcula_uno` y `calcula_dos`, después ejecutar `calcula_tres` (con los datos producidos por los otros dos threads) y depositar el resultado en otra variable gobal. El programa principal debe esperar a que el tercer thread realice su cálculo y pintar el resultado. Utilizar los mecanismos de sincronización que se consideren más adecuados. Las funciones `calcula_uno`(), `calcula_dos`() y `calcula_tres`() están en el fichero de cabeceras `calcula.h` de acuerdo con los prototipos:

   ```
   float calcula_uno();
   float calcula_dos();
   float calcula_tres(const float *a, const float *b);
   ```

4) Escribir una función a la que se le pasa un entero que representa tiempo en milisegundos, y que se duerma utilizando la función `clock_nanosleep`() en modo absoluto hasta un instante igual a la hora actual más el tiempo indicado. Todo según el reloj `CLOCK_REALTIME`.

5) El comando "`mast_analysis offset_based_optimized -c -p -s ejemplo.txt`" realiza el análisis de planificabilidad del sistema descrito en el fichero `ejemplo.txt` y genera una serie de infomación que muestra por la salida estándar. Escribir un *script* para una *shell* de Unix que permita analizar todos los sistemas descritos en los ficheros "`.txt`" del directorio que se pasa como primer parámetro, y genere en el directorio que se pasa como segundo parámetro un conjunto de ficheros con las salidas producidas por las ejecuciones de `mast_analysis`. El nombre de estos ficheros se construye anteponiendo al nombre del fichero analizado el prefijo que se pasa como tercer parámetro. Realizar el tratamiento de errores. Ejemplo de llamada al *script* "`analiza_con_mast`":

   ```
   [xxxxx]$ analiza_con_mast dir_origen dir_resultados salida_
   ```