Examen de Sistemas Operativos

Junio 2010

(2 puntos por cuestión)

1) Escribir un programa en C que sea capaz de ejecutar los dos programas (con sus respectivos argumentos) 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 el programa `prog1` con el argumento `arg_p1`, y el programa `prog2` con el argumento `arg_p2`:

```
[xxxxx]$ ejecuta prog1 prog2 arg_p1 arg_p2
```

2) Escribir una función que sirva para truncar un fichero al tamaño indicado manteniendo la información que contenía al final del mismo. La función recibe como parámetros el nombre del fichero y el tamaño al que se debe truncar. Si se produce un error la función retorna su código, en caso contrario devuelve un 0. La función se utilizará para ficheros cuya longitud es menor de 10000 bytes. Tratar lo errores que se puedan producir.

```
int trunca(char *f, int longitud);
```

3) Escribir un programa que cree dos threads. Cada uno debe llamar respectivamente a las funciones `uno`, y `dos`, que acceden a una variable global sin protección para exclusión mutua del tipo `variable_global` (definida como `typedef struct`). El programa principal debe esperar a que los threads que ejecutan `uno` y `dos` acaben, y luego debe ejecutar la función `tres`. Utilizar los mecanismos de sincronización que se consideren más adecuados. Las funciones `uno`(), `dos`() y `tres`() están en el fichero de cabeceras `funciones.h` y presentan los siguientes prototipos:

```
void uno(variable_global *v);
void dos(variable_global *v);
void tres(variable_global *v);
```

4) Escribir una función para un thread que ejecute periódicamente la función `actividad`. A la función del thread se le pasa como parámetro el periodo en microsegundos representado por un número entero.

```
void actividad ();
```

5) El comando "`mast_analysis offset_based_optimized -c -p ejemplo.txt`" realiza el análisis de planificabilidad del sistema descrito en el fichero `ejemplo.txt` y genera infomación que muestra por la salida estándar. Las opciones `-c` y `-p` permiten realizar respectivamente el cálculo de techos de prioridad y la asignación de prioridades. Escribir un *script* para una *shell* de Unix que permita analizar todos los sistemas descritos en los ficheros que acaban con "`.txt`" del directorio que se pasa como primer parámetro con las opciones que se pasan en los parámetros siguientes (hasta el máximo de las dos que admite el comando de análisis). Realizar el tratamiento de errores. Ejemplo de llamada al *script* "`analiza`" para el directorio `origen` con las opciones `-c` y `-p`:

```
[xxxxx]$ analiza origen -c -p
```