# Senior Software Engineer. Optical Networking Area

2008

## Introducción

El objetivo del presente ejercicio es que el candidato diseñe, desarrolle / implemente y valide una serie de aplicaciones simples, que permitan evaluar:

- Los conocimientos de lenguajes C / C++ y programación de sistemas del candidato.
- Su diseño y su metodología.
- Capacidad de solucionar nuevos problemas.
- ”Coding Style”: estructura en directorios, Makefiles, estilo de C/C++ etc.

El candidato deberá entregar, en formato electrónico, el código fuente correspondiente a los ejercicios propuestos y toda documentación o ficheros auxiliares (e.g. ficheros de entrada a modo de ejemplo, Makefiles, etc..) que considere necesarios. Se recomienda un directorio por aplicación.

## Comentarios

- En todo momento, el candidato puede proponer soluciones alternativas o, en su defecto, proporcionar la idea general de la solución propuesta, criterios generales de diseño o incluso bocetos de solución (e.g. el pseudo código equivalente).
- En caso de duda referente a la especificación funcional, o si el candidato considera que varias interpretaciones o soluciones son posibles, se deberá escoger una, justificando su elección.
- No es necesaria una implementación completamente operativa, pero como referencia, el sistema en que la aplicación se ejecutaría es un sistema GNU/Linux con kernel 2.6 (testing / unstable), compilado con gcc / g++ 4.2.
- Se valorará todo tipo de documentación asociada, desde cómo compilar y usar el programa, hasta documentación de la implementación misma, especificando los diferentes (mini)módulos y las entradas / salidas de cada módulo y/o función.
- No se supone conocimiento de la librería Glib, Boost o específicos de Linux y se espera (y recomienda) que el candidato utilice recursos (Internet, buscadores, foros, libros, ejemplos) para aprender el uso de librerías o ”Howtos”.
- Responda a los ejercicios que considere oportunos, priorice según su propio criterio.

# Formulación del problema

Considere un archivo de texto ”input.txt” con un número variable de líneas, cada línea conteniendo 3 enteros, separados por un espacio.

```
123 545 645
234 323 454
...
```

El primer entero identifica el número de registro, y los dos siguientes son atributos del mismo, atributo ”a” y atributo ”b”. Tanto el número de registro como los atributos se codifican con un entero sin signo de 32 bits. El programa final debe:

1. Al ser ejecutado, abrir el archivo en modo lectura, recorrer cada una de las líneas y construir en memoria una estructura de datos (e.g. mapa, o tabla hash, en su defecto una lista doblemente enlazada) que contenga todos los registros leídos y que permita la búsqueda rápida por identificador de registro.
2. Una vez leído el archivo, iterar cada uno de los registros desde la estructura de datos propuesta y para cada registro enviar un datagrama UDP a la dirección IP 10.10.1.1, puerto 2020, conteniendo el identificador de registro y los dos atributos.

# Ejercicio I: Implementación en C con ayuda de Glib

En este caso, se desea que la implementación utilice en la medida de lo posible la librería Glib `http://library.gnome.org/devel/glib/unstable/index.html` : *(From Wikipedia) GLib is a cross-platform software utility library. It started life as part of the GTK+ project, but is now used by other applications. While it was originally a convenient library to collect low-level code in, it has since expanded into offering wrapper functions for functionality that is typically different across platforms. GLib is a cross-platform component that makes applications portable across different operating systems.*

## Palabras clave

`g_malloc, g_malloc0,... GTree g_tree_new, g_tree_foreach... GList g_list_new,... socket, bind, sendto, htonl...`

# Ejercicio II: Implementación (opcional) en C / Glib con diseño multithread

En este caso, se mantiene la idea básica del programa, pero se pide un diseño multithread, en la que un thread es responsable de la lectura del fichero, y el segundo thread se ocupa de transmitir los datagramas UDP a la red. Ambos threads se comunican mediante un modelo ”productor/consumidor”.

## Palabras clave

`GThread g_thread_init, g_thread_create,... GAsyncQueue g_async_queue_push g_async_queue_pop`

# Ejercicio III: Implementación (opcional) en C++ con STL

## Básicos STL

- a) Desarrolle su propia versión (en el namespace ”ext”) de copy_if. El prototipo podría ser algo como:

```
template<typename I, typename O, typename Pred>
O copy_if ( I first, I last, O res, Pred p)
```

- b) Desarrolle un programa que lea una serie enteros desde STDIN, los guarde en una contenedor / secuencia y elimine los elementos cuyo valor sea par, mostrando por pantalla el resultado. No utilice bucles explícitos (for, while, do while, break, etc.)

### Palabras clave

`std::vector, algorithm, std::copy, functors, function objects.`

Mismo diseño que en el ejercicio I, pero la solución debe ser desarrollada en C++ usando la ”Standard Template Library” (STL), sobre todo los contenedores estándar, los iteradores y, si necesario, los algoritmos estándar. Esta sección es a implementar exclusivamente si el ejercicio I ha sido desarrollado en su totalidad.

## Específico 2008 - I

Desarrolle un programa que obtenga los interfaces de red de un sistema GNU/Linux 2.6, y muestre por pantalla:

- el nombre e índice del interfaz.
- La dirección IP si la tiene
- El tipo (e.g. ethernet, gre, ...)

### Palabras clave

`ifreq, netlink, rtnl_open, ...`

## Específico 2008 - II

Desarrolle un programa que obtenga la tabla de rutas del kernel GNU/Linux 2.6, y muestre por pantalla:

- el prefijo correspondiente a la entrada.
- el gateway
- todo atributo (e.g. coste) necesario.

### Palabras clave

`netlink, rtnetlink, route(8), ip(8), ifconfig(8)`

## Específico 2008 - III

Explique que es OSPF, y diferencias con OSPF-TE

## Ejercicio IV: Implementación (opcional) en C++ / STL y multi-thread

Misma idea que en el ejercicio II pero utilizando C++. El candidato es libre de utilizar cualquier librería abierta a su discreción que considere oportuna. Este ejercicio debe implementarse si el resto de ejercicios han sido desarrollados en su totalidad.