Apellido y Nombres	Legajo	Profesor de Cursada	# de Hojas

Parte teórica

Dibuje conceptualmente el contenido del stack del programa en ejecución, si al correrlo con un debugger colocamos un *breakpoint* en la línea indicada. Asuma un sistema operativo de **32 bits**, y que al arrancar el programa se ha asignado la dirección de memoria **0xbfff0000** como comienzo del stack.

```
#include <stdio.h>

int local (int a, int b);

int main (void) {
   printf("%d\n", local(5,10));
   return 0;
}

int local(int a, int b) {
   int r;
   r = a+b;
>> return r;
}
```

Parte Práctica

Participamos en la implementación de un sistema de distribución de mensajes (conocido como Pub/Sub), donde recibimos mensajes desde un **"publicador"** para un cierto canal de comunicación, y los distribuye entre los **"subscriptores"** que estén colgados de ese canal. Se cuenta con la siguiente estructura y definiciones:

```
#define OK 0
#define ERR_MEMORY  -1
#define ERR_ITEM -2

typedef struct sub_s {
  char id[100];           // Identificador único del subscriptor
  unsigned int *canales;  // Lista de canales a los que está subscripto.
  int (*enviar)(char *);  // Puntero a función para enviar mensajes
  struct sub_s *sig;
} sub_t;
```

Se pide implementar las siguientes tres funciones:

```
int suscribir(sub_t **s, char *id, unsigned int *canales, int (*enviar)(char *));
int desuscribir(sub_t **s, char *id);
int publicar(sub_t *s, char *msg, unsigned int canal);
```

suscribir agrega un suscriptor a la lista. Devuelve OK en caso de éxito, ERR_MEM en caso de algún error de memoria, o ERR_ITEM si el suscriptor ya se encuentra en la lista (*por ende no agrega duplicados*).

desuscribir quita un suscriptor de la lista, liberando memoria según corresponda. Devuelve OK en caso de éxito, ERR_MEM en caso de error de memoria, o ERR_ITEM si el suscriptor no se encuentra en la lista.

publicar busca los suscriptores conocidos que se encuentren colgados del canal dado, y de ser así les envía el mensaje especificado. Devuelve OK en caso de éxito, o ERR_ITEM si no hay subscriptores para dicho canal.

Cada uno de los subscriptores provee una función propia para enviarle los mensajes publicados a través del puntero a función **enviar(...)**, que devuelve OK en caso de éxito, y ERR_ITEM en caso de error. Dichas funciones ya se encuentran implementadas, sólo conocemos su prototipo, y simplemente deben ser utilizadas en el momento oportuno de enviarle mensajes a cada subscriptor particular.

Consideraciones

- El identificador es simplemente un nombre único para el subscriptor.
- El canal es siempre un número entero positivo.
- Los subscriptores pueden estar colgados de uno o más canales. La lista de canales para cada subscriptor es un vector de enteros positivos, de longitud variable, **terminado en cero**.
- **No puede asumir que el id y la lista de canales recibidos en suscribir() son eternos. Debe copiar su contenido al nodo de la lista, no simplemente asignarlo.**
- Puede agregar las funciones auxiliares que crea necesarias (sugerencia: función auxiliar que cuente la cantidad de canales recibidos en el vector).
- La lista de subscriptores puede estar vacía (o no) tanto al momento de suscribir, desuscribir y publicar.

No se pide:

- Ninguna implementación del main ni las funciones de enviar mensajes.
- Ninguna lógica más allá de la necesaria para las tres funciones pedidas.
- Ninguna reinvención de la rueda. Utilice funciones de la librería estándar de C siempre que sea posible. Puede preguntar los prototipos si no los recuerda exactamente.

Condición de aprobación

- Implementación de las funciones pedidas sin errores conceptuales, sin que produzcan *segmentation faults* ni *memory leaks*.
- Demostración de conocimientos teóricos.