Implement a system call that prints a given message (follow these instructions: [Add System Call to Minix Tutorial - Updated.pdf](https://online.upr.edu/pluginfile.php/2526214/question/questiontext/3143063/1/25752323/Add%20System%20Call%20to%20Minix%20Tutorial%20-%20Updated.pdf))

# Exercise 1

## 25/25

Analiza el archivo /usr/src/include/minix/ipc.h (Struct message)

Modifica el recién creado system call para recibir e imprimir un parámetro de tipo long a través de la estructura “message” (el archivo [System\_Calls\_in\_Minix.pdf](https://online.upr.edu/pluginfile.php/2526214/question/questiontext/3143063/2/25603401/System%20Calls%20in%20Minix.pdf) puede ser de ayuda). Desde el archivo de test se debe enviar el parámetro a la librería de usuario creada en el ejercicio anterior está a su vez envía el parámetro al syscall.  Tu librería debe utilizar el mensaje tipo M4.

El system call debe imprimir el número arbitrario que le envía el código de usuario por medio de la librería.  El formato de la salida debe ser:

Hola! El número es: <número>

donde <número> es el valor del parámetro enviado a la librería por medio del mensaje.

Incluye en la contestación las definiciones de las funciones

int do\_mycall() {  
}  
  
int mycall(long integer) {  
}

en ese orden, según las definiste y las probaste en tu copia de minix.

# Exercise 2

## 25/25

La variable global who\_p contiene el index del proceso que invocó el syscall en la tabla de procesos del PM (array mproc[NR\_PROCS]). La variable global mp representa al proceso que invocó el system call. El array mproc[] puede ser accedido globalmente. Este contiene información de todos los procesos de usuario. Vea el archivo forkexit.c en el servidor pm para saber cómo puede recorrerlo.

La estructura de un proceso en el servidor PM contiene una estructura “message” que puede utilizarse para retornar valores a la función que invocó el syscall. Algunos de sus campos son renombrados en el archivo param.h del PM. El loop principal del PM se encuentra en main.c

Teniendo en cuenta que la información de procesos del PM está contenida en /usr/src/servers/pm/mproc.h:

¿Puedes identificar 5 atributos de los procesos?  
¿Qué flag que puede utilizarse para saber si un proceso está en uso.?  
  
No hace falta contestar algo en este ejercicio.  El conocimiento adquirido en él será útil para resolver los siguientes.

# Exercise 3

## 25/25

* **Investiga cómo funcionan los syscalls gepid y getppid.**

Ayuda: Busca los archivos getpid.c y getppid.c. Utiliza lo aprendido en el laboratorio para encontrar las demás partes de la implementación de estos syscalls a partir del nombre de las mismas.

# Exercise 4

## 15/25

Implementa un syscall que devuelva el id de un proceso o el de su parent en función de un parámetro recibido por la librería de usuario. Para ello define dos constantes en dicha librería (CURRENT = 1, PARENT = 2), dependiendo de la constante reciba retorne el valor correspondiente.  La llamada debe usar el tipo de mensaje M4.

Incluye en la contestación las definiciones de las funciones

int my\_getpid() { /\* system call \*/  
}  
  
int mygetpid(long integer) { /\* user library \*/  
}

en ese orden, según las definiste y las probaste en tu copia de minix.  Incluye también las definiciones (#define) de CURRENT y PARENT y úsalas en tu código.  No debes setear mp\_reply.