TIPOS DE LIBRERIAS Y SU RELACION CON LOS SISTEMAS OPERATIVOS

1.- Que es una libreria de programacion

Tal vez alguna vez te habrás preguntado: ¿Qué es una librería de programación? Bueno, debes saber que una librería de programación es un conjunto de implementaciones codificadas (Archivos o ficheros) por medio de un lenguaje de programación (C/c++,Phyton, Java Script) que ofrece una interfaz bien definida según la funcionalidad para la que fue diseñada.

Para entenderlo bien, puede decirse que se trata de un conjunto de instrucciones que pueden ser utilizadas por diferentes programas. La misma librería, en teoría, es usada en varios programas, dándole el mismo uso o uno diferente.

Por ejemplo, si estás desarrollando una aplicación para automatizar el envío de correos, cuando estés desarrollando las fechas de configuración puedes utilizar una biblioteca que te ayude con los códigos y archivos que necesitas, después continúas desarrollando el programa.

Se trata de una herramienta que te permite optimizar tiempo y recursos. Podría parecer algo complicado, pero no es así y al final del blog te darás cuenta.

Antes de comenzar, es importante tener en cuenta que las librerías se vinculan a un programa o a otras librerías en diferentes pasos del desarrollo o cuando se ejecutan. Esto depende del tipo de vínculo que se quiera establecer. Es decir, las librerías se pueden vincular en diferentes pasos de la creación del programa.

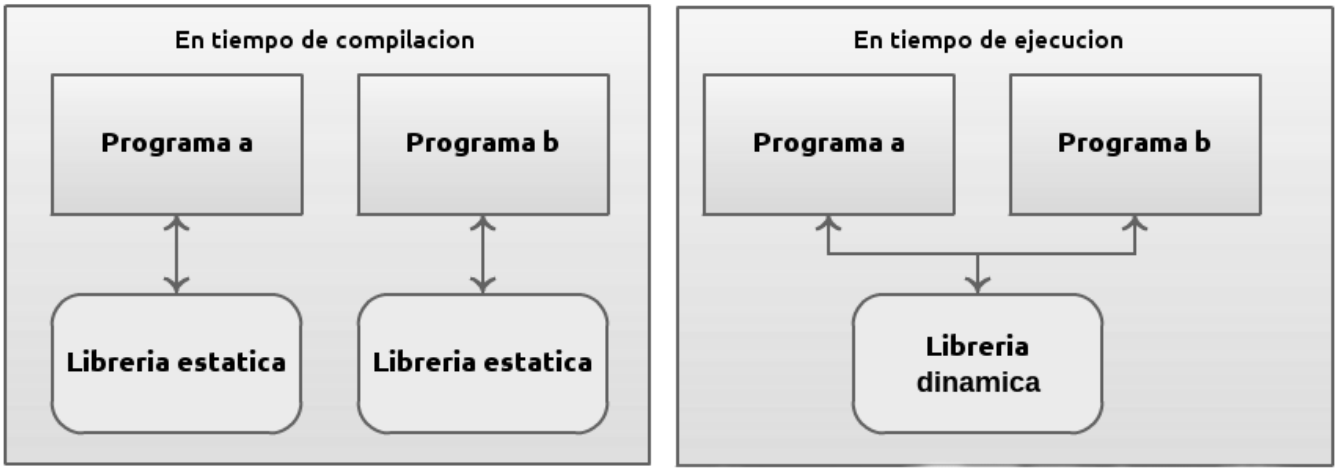
En las practicas que estamos realizando , las librerias se usan como ficheros externos , donde para obtener usarlas necesitamos incluir (include) una cabecera en nuestro programa main donde se hace referencia a las funciones disponibles . Estos ficheros se compilan con los programas que hemos escrito y se linkan resolviendo todas las direcciones que utilizan en la memoria, de forma que el programa final incluye tanto el programa que nosotros hemos escrito como las librerias que utilizamos.

2.- Tipos de librerias

Dentro de la clasificacion de las librerias , hay dos tipos principales si bien hay algun subtipo que tambien se puede considerar.

a.- Librerias Estaticas

Son las librerias que disponen de informacion de procedimientos utiles para nuestro programa ; pero que no son especificos del mismo , tienen procedimientos de utilidad general y pueden ser realizadas por nosotros mismos o extraidas de internet o de cualquier otra fuente .
Estas librerias tienen que incluirse con nuestro programa en el momento de compilacion y por tanto aumenta el tamaño de nuestro codigo



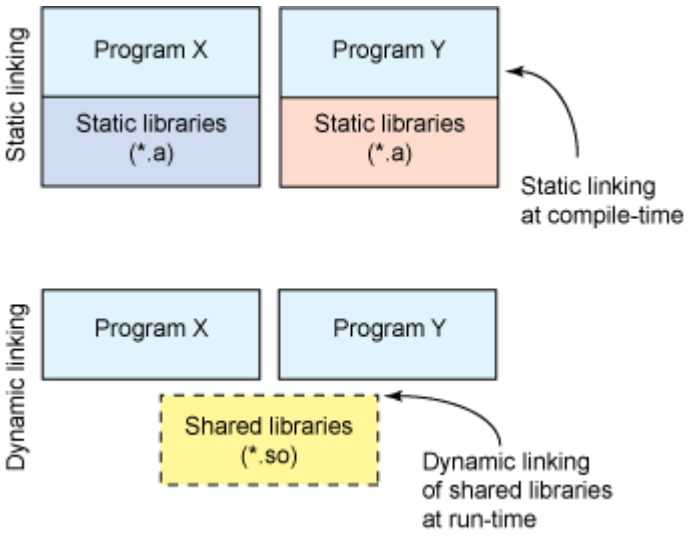
Como podemos ver en la figuran anterior la libreria estatica se utiliza en muchos programas y nos permite realizar aplicaciones sin saber el detalle interno de cada cosa .En nuestro caso manejaremos wifi , bt , i2c ,... sin tener el detalle del bajo nivel y solo conoceremos como funcionan los objetos asociados a la libreria .

Son todos los ficheros con el siguiente formato de nombre y extensión lib*.a. Se incluyen dentro del ejecutable de la aplicación que las usa.

Para el uso de una libreria no se necesita el codigo fuente sino solo una cabecera y un codigo objeto . Por tanto se puede pasar un programa sin que la persona que lo usa tenga acceso al contenido del mismo .

b.- Librerias Dinamicas

Las librerias dinamocas tienen el mismo comportamiento conceptual de una libreria estatica ; es decir , nos permiten utilizar recursos sin tener el detalle de como estan hechos ; pero tienen una diferencia fundamental se cargan y descargan en tiempo de ejecucion



Son todos los ficheros con el siguiente formato de nombre y extensión lib*.so*. Estos ficheros no se incluyen dentro de los ejecutables de las aplicaciones y, por tanto, los ejecutables ocupan menos. A cambio, debemos disponer de los ficheros de biblioteca dinámica junto con el ejecutable para que este funcione. Los ejecutables cargan las bibliotecas en tiempo de ejecución usando ld.so o ld-linux.so. Como nota, en entornos Ms Windows, son las DLL.

En Linux (y en Unix) las librerías compartidas se llaman Shared Objects y llevan el sufijo .so (su equivalente en Windows son las .dll).

Como ya hemos mencionado, las librerías compartidas se vinculan a un programa en tiempo de ejecución, permitiendo que el código de la librería se cargue en memoria una única vez y pueda ser usado por varios programas, de esta forma se consigue que el tamaño del código sea menor con su correspondiente ahorro de espacio en memoria.

Ademas de esto con las librerías compartidas se cumple el principio de modularidad (programación modular o orientada a objetos), de forma que si necesitamos modificar alguna funcionalidad nos bastara con editar la librería que la contiene, dejando el programa que las utiliza sin modificar. Los nombres de las librerías compartidas Por convenio las librerías compartidas pueden tener varios tipos de nombre : El nombre usado por el enlazador (‘lib’ + nombre de la libreria + ‘.so’) Nombre completo (‘lib’ + nombre de la libreria + ‘.so’ + ‘.’ + numero de versión) // Link hacia la librería con el nombre real Nombre real (‘lib’ + nombre de la libreria + ‘.so’ + ‘.’ + numero de versión + ‘.’ + numero de subversión + ‘.’ + revisión) // La revisión es opcional

Para el cambio de versiones hay que tener dos factores en cuenta, el numero de la subversión se cambia cuando se realizan cambios en la librería y esta no pierde compatibilidad con la version anterior, pero si se pierde compatibilidad el cambio tiene que ser de versión y no de subversión. Grac## 2.- Tipos de librerias ias a este convenio de nombres es posible que múltiples versiones de una librería compartida coexistan en el sistema.

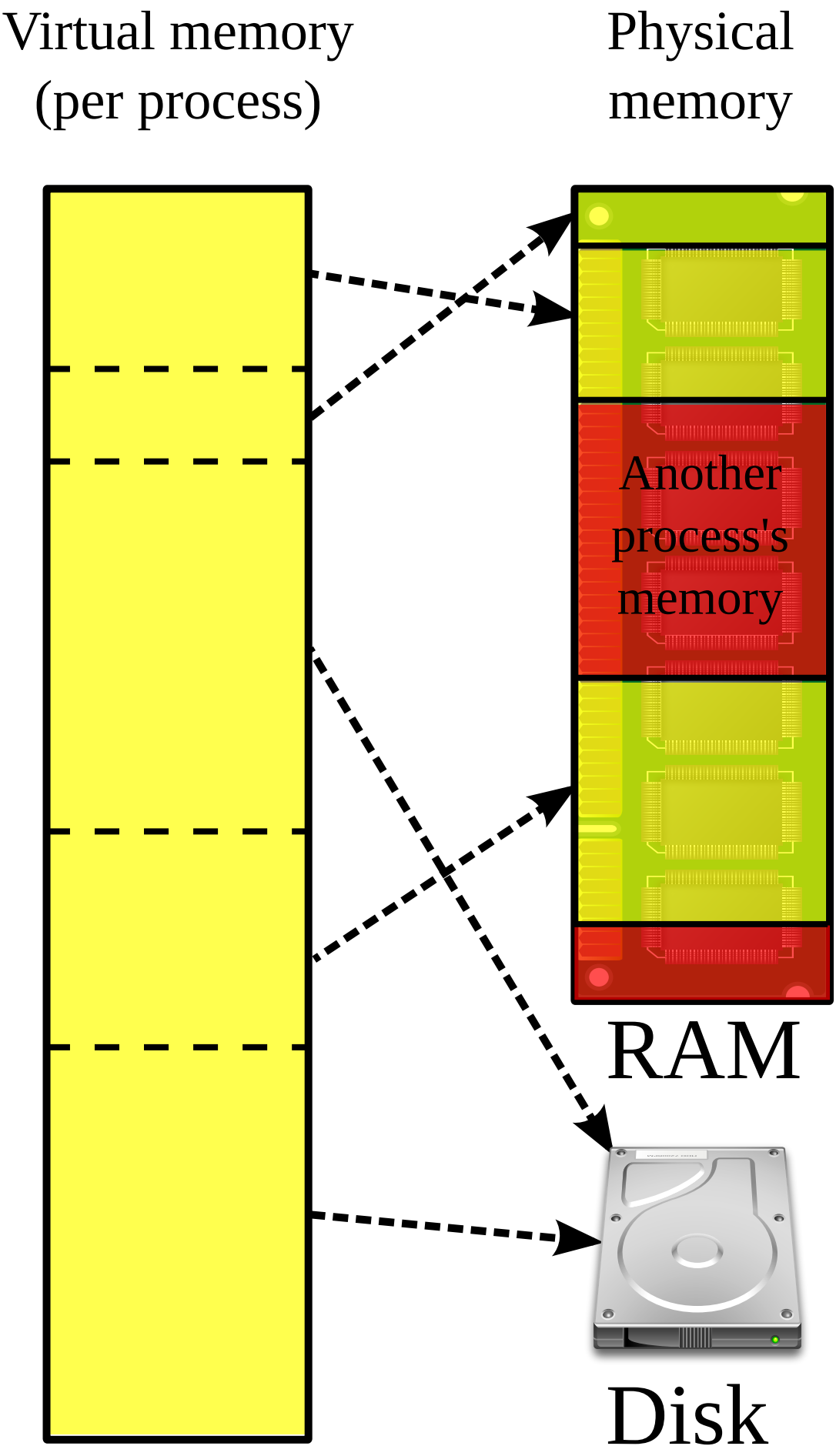
/lib : Librerías de sistema, vitales /usr/lib : Librerías de usuario básicas, no se necesitan en el boot /usr/local/lib : Librerías que no forman parte de la distribución estándar

3.- Manejo de la memoria y sistemas operativos

Recomendamos el enlace administracion de memoria para tener una informacion mas completa .

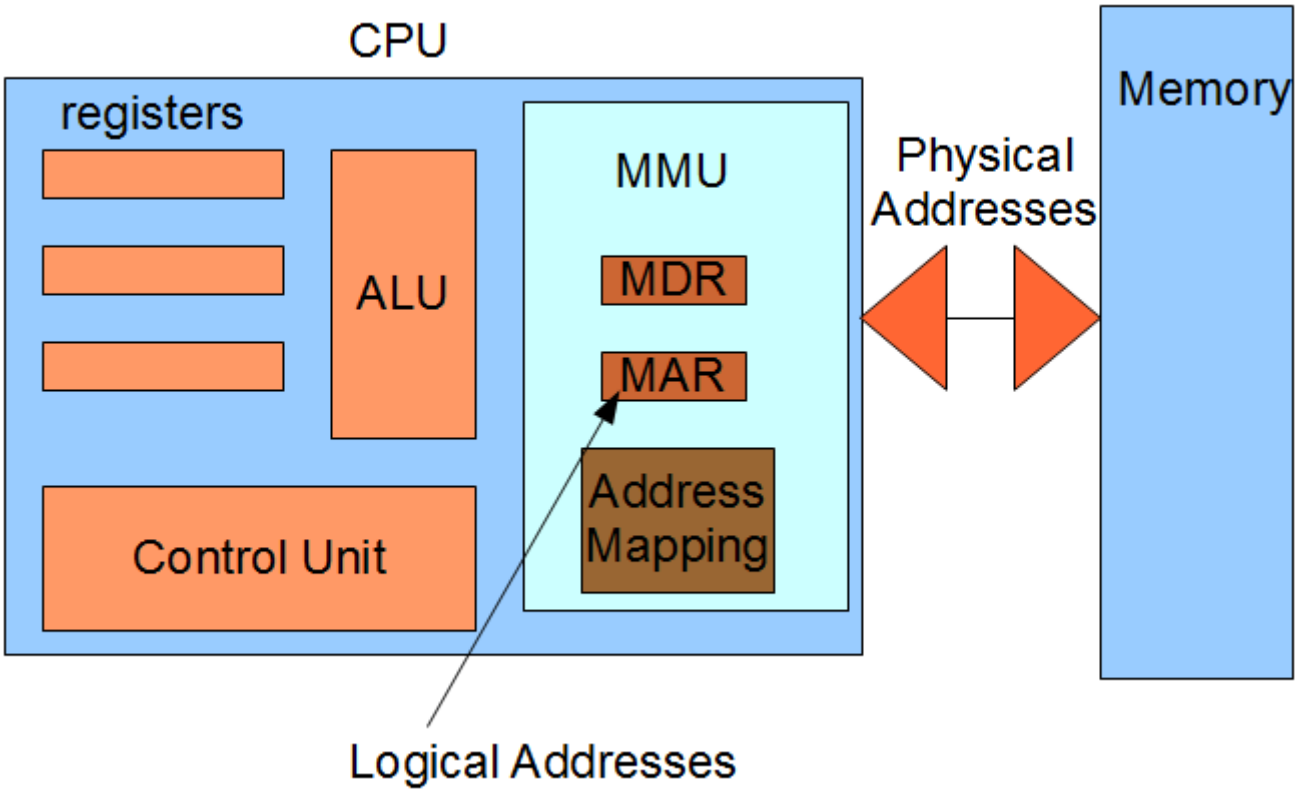
En los procesadores pequeños como los que estamos utilizando en la asignatura la memoria esta ubicada de forma secuencial ; esto significa que nuesra memoria empieza en una direccion y apartir de alli se ubica nuestro programa (solo con alguna excepcion como los vectores de interrupciones).

Pero cuando la memoria aumenta de tamaño es fisicamente muy costoso tener una rango de direcciones lineal especialmente porque la memoria se distribuye en multiples dispositivos .


Virtual memory
(per process)
Physical
memory
Another
process's
memory
RAM
Disk

A partir de aqui en los procesadores mas complejos de utiliza el concepto de memoria virtual , dicha memoria permite al programa ejecutarse secuencialmente y a las librerias dinamicas cargarse uy descargarse en las mismas direcciones aunque fisicamente esten ubucadas en lugares distintos .

De hecho dentro del procesador debe existir una unidad de administracion de memoria MMU



Para utilizar un sistema operativo tipo windows , linux , google, mac ,... necesitamos que nuestro procesador disponga de MMU y acceso a memoria dinamica .

Sin embargo podemos utilizar un sistema operativo de bajo nivel FreeRTOS .

Este sistema operativo no puede cargar y descargar procesos como se hace en linux o windows ; pero si nos permite facilitar nuestro tranbajo de programacion , gestionando tareas de una forma sencilla y protegiendo el uso de estas tareas para que no exista conflicto entre las mismas cuando se quieren utilizar recursos comunes .

Revisar una explicacion detallada en https://www.electrosoftcloud.com/freertos-en-esp32-esp8266-multi-tarea/