Punteros

Definimos a un puntero como una variable que se encarga de almacenar direcciones de memoria, generalmente representada en Hexadecimal. Los punteros miden 4 bytes en memoria, ósea una longitud de 32 bits, lo que significa que puede albergar 232 direcciones distintas (4 GB).

El sistema operativo se encarga de asignarnos memoria para nuestro proceso, y el resto será destinada a otros usuarios, a otros procesos o al mismo sistema operativo. Si intentamos acceder a memoria no asignada, tendremos un error. Con los punteros, podemos salirnos de nuestra área asignada.

Operadore de tipo puntero

\* => Lo apuntado por …

& => La dirección de …

Ejemplos:

Int \* PUNT

Float \* Q

¿De que tipo es PUNT? PUNT es de tipo int \*, es decir que es un puntero que apunta a un entero; mientras que Q es de tipo float \*, que significa que es un puntero que apunta a un flotante.

¿Qué es lo apuntado por PUNT? Un entero

¿Qué es lo apuntado por Q? Un flotante

Si se quiere cargar un puntero con el contenido del otro, lo tenemos que hacer mediante un casteo al tipo que corresponda.

Un puntero que no sabemos que contiene, es decir, cuando se lo declara, como cualquier variable recién declarada, contiene basura. Esto es llamado un puntero descontrolado. Cuando querramos acceder a una dirección de memoria basura, seguramente vamos a querer acceder fuera de nuestro segmento y vamos a obtener un error.

Así como %d se utiliza para mostrar en pantalla un entero, o %f para lo mismo con un flotante, tenemos también un formato de tipo puntero %p que lo muestra como una dirección de memoria, con sus 4 bytes.

A través de los punteros, no solo podemos ver memoria, sino que además podemos modificar memoria

**Archivos de Disco**

**Concepto de Flujo (Stream)**

Se introduce un intermediario entre los programas en C y los archivos, conocido como flujo (stream), y lo que fluye es información.

Un driver es un programita que va a manejar el dispositivo en bajo nivel.

**Flujos Standard**

1. stdin = Standard Input (Teclado)

Es el flujo standard para el teclado, y es un flujo de entrada.

1. stdout = Standard Output (Pantalla)

Es el flujo por el cual se canaliza todo lo que enviamos con printf, putchar, putc …

1. stderr = Standard Error (Pantalla)

Canaliza los mensajes de error.

**Flujos no Standard**

Cuando se trabaja con algún archivo en algún medio de almacenamiento, voy a tener que crear el flujo, entonces el nombre lo asigno yo.

Una vez nombrado y asignado el flujo, dejo de trabajar con el nombre del archivo físico, sino que trabajo con el nombre del flujo.

**Tipos de Acceso**

**Acceso en bajo nivel**

Son los drivers que trabajan a bajo nivel para poder manejar nuestro programa o dispositivo. Gracias al concepto de flujo, no trabajamos en este acceso.

**Acceso de tipo Unix**

Son accesos de tipo “no buffereados”, es decir que la información que mando al disco, va directamente al disco cada vez que la enviamos.

**Acceso de tipo ANSI (Buffereado)**

Estos accesos intercalan entre nuestro programa y el disco un área de memoria denominada “Buffer”. Este “Buffer” es un separador que se encuentra alojado en la memoria, en la dinámica, la local, global … tiene restricciones de acceso y funciona como una memoria de tipo FIFO (First In First Out / El primero que llega es el primero en salir).

Se las conoce como “colas” a este tipo de memoria, asociándola a las colas que se hacen en los bancos (El primero en llegar, es el primero en ser atendido y por lo tanto, el primero en salir).

**Ventajas de buffer**

Un buffer se utiliza para compatibilizar dispositivos que trabajan en distintos tiempos o con distintas velocidades.

Cuando se quiere leer algo del disco, lo que se hace es acceder al disco, bajar esa información al buffer una única vez, pagando el tiempo de acceso al disco. Luego se lee la información o los bytes directamente del buffer, con la velocidad que nos permite la memoria.

Para poder tener una noción clara sobre esto, deberíamos tener una idea mas acertada de cual es el tiempo de acceso a las memorias y la del disco.

Generalmente, el tiempo de acceso a las memorias esta expresado en unidad de frecuencia (GHz), pero si lo quisiéramos expresar en tiempo, estaría en el orden de los nanosegundos (Mil millonésima parte del segundo).

El tiempo de acceso a los discos es mucho mayor, ya que depende de mecanismos (Es el caso de los Discos Rígidos). Tomando el caso de un disco de 7200 RPM, podemos estimar haciendo el cálculo de 60s / 7200rpm = 8.3 x 10-3 (Milésimas de segundo).

Teniendo en cuenta esto, podemos decir que la memoria es 1 millón de veces más rápida que el disco rígido.

La única desventaja de esto, es que la memoria del buffer es volátil, por lo que si se interrumpe la energía, la información se pierde.

**Tipos de Flujos**

**Flujos de Texto**

Lo que fluye son caracteres, en esos caracteres existen desdobles, el caso mas significativo es el del “\n”.

En el “\n” la n viene de “new line”, pero además el cursor vuelve al margen izquierdo. Este efecto ocurre porque el “\n” se reemplaza cuando va al flujo stdout (Recordemos que es un flujo de texto / Muestra texto en pantalla), desdoblándose en “\r””\n”.

Cuando mando esta información al disco, cuando mando un byte, se están guardando dos y cuando luego leo, estos dos se arman en uno solo, recibiendo así lo mismo que mande.

Es por esto que no puedo saber con precisión donde van a estar los caracteres por mucho que los cuente.

Los accesos de tipo “Random” no van a ser confiables y por lo tanto, no vamos a manejar la información que tengamos que acceder de forma directa en flujos de texto, para eso están los flujos binarios.

**Flujos Binarios**

Lo que fluyen son bytes.

**Apertura de un Archivo**

1. Crear el flujo
2. Vincular el flujo con el archivo físico alojado en el disco.
3. Asignarle un nombre.