



REDES DE COMPUTADORAS

PRÁCTICA 2:

"Checksum"

NOMBRE DEL ESTUDIANTE:

HERNÁNDEZ RODRÍGUEZ ARMANDO GIOVANNI

GRUPO: 2CM15

PROFESORA: CORTEZ DUARTE NIDIA ASUNCIÓN



Nombres: Hernández Rodríguez Armando Giovanni



Práctica 2.-Checksum

Objetivo: El alumno(s) implementará en lenguaje C una función que reciba una trama (arreglo de caracteres sin signo) y que imprima el checksum correspondiente.

Utilizado para detectar errores en la transmisión de datos a nivel de software. Usado por los protocolos de TCP/IP.

Vamos a trabajar con la estructura de la cabecera IP; por ahora solo basta recordar en que posición del arreglo se encuentra el Checksum. Si consideramos unicamente el paquete IP, entonces el checksum se encuentra en T[10] y T[11]

Prueba 1. En esta trama los valores del Checksum estan en 0000 ya que apenas se va a calcular (ALICIA)

Unsigned char $T[]=\{0x45, 0x00, 0x01, 0xe2, 0xd7, 0xdb, 0x40, 0x00, 0x80, 0x06, 0x00, 0x00, 0xc0, 0xa8, 0x01, 0x43, 0x94, 0xcc, 0x3a, 0xdd\}$

Prueba 2: Para esta trama ya se tiene el valor del checksum (es la que recibió BETITO) verificar si el checksum es correcto y en caso de no serlo mostrar el correcto.

45 00 01 9c d7 de 40 00 80 06 88 9d c0 a8 01 43 94 cc 3a dd

Instrucciones.

- -Abrir un block de notas y escribir el código correspondiente en C.
- -Las tramas se inicializarán en hexadecimal en un arreglo de caracteres sin signo
- -Compilar en consola haciendo uso de gcc (en caso de no tenerlo, instalarlo).
- -Una vez terminado deberán probar con los dos ejercicios vistos en clase e incluir aquí las capturas de pantalla

Se entrega este documento, incluir:

Todo tu Código, sugiero utilizar alguna herramienta para dar formato como:

http://www.planetb.ca/syntax-highlight-word

Nota: Para darle formato al código se utilizó carbon:

https://carbon.now.sh/



M. en C. Nidia A. Cortez Duarte



Nombres: Hernández Rodríguez Armando Giovanni



Práctica 2.-Checksum

Capturas de pantalla del correcto funcionamiento del programa

La primera imagen muestra la compilación exitosa de la practica

argio@DESKTOP-C3J0CAB MINGW64
\$ gcc checksum.c -o checksum

Luego al ejecutar la <u>prueba</u> 1 se tiene obtiene que el checksum correspondiente es **8fa5**

argio@DESKTOP-C3J0CAB MINGW64 \$./checksum Prueba1 Checksum de la trama1: 8fa5

Ahora con la prueba 2 se despliega el respectivo mensaje de que el checksum es incorrecto y se procede a obtener el checksum correcto que es **8fe8**

argio@DESKTOP-C3J0CAB MINGW64
\$./checksum

Prueba2
Checksum incorrecto
El checksum correcto es: 8fe8





Nombres: Hernández Rodríguez Armando Giovanni



Práctica 2.-Checksum

Mapa de memoria

Deberán crear y mostrar **el mapa de memoria** utilizado en su programa (considerar registros de 8 bits) En el mapa de memoria NO se debe considerar el arreglo de la trama.

El mapa de memoria utilizado para esta práctica es el siguiente:

Mapa de memoria del Programa

unsigned short int checksum	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
unsigned char i	0	0	0	0	0	0	0	0
unsigned char aux	0	0	0	0	0	0	0	0

Conclusiones

Mediante el desarrollo de la práctica dos pude comprobar la importancia de funciones para comprobar la trama, como es el caso de la suma de comprobación o checksum, valor que permite comprobar la integridad de la información transmitida. Aunque al seguir cada uno de los pasos establecidos por el RFC 791 se puede determinar si una suma de comprobación es correcta o no, se ahorra una serie de pasos si primero se calcula el valor del checksum y luego se compara con los valores de una trama establecida, ya que en dado caso que el checksum sea incorrecto se proporciona el valor correcto inmediatamente. Anteriormente no había escuchado sobre funciones de comprobación de trama, sin embargo con las clases de redes ahora sé que existen tres formas principales para calcular el número de secuencia de verificación de trama y son las siguientes: verificación por redundancia cíclica (CRC), paridad bidimensional y checksum o suma de comprobación.

Finalmente, considero que la integridad de datos es muy importante en redes porque nos garantiza que los datos no han sido modificados por partes no autorizadas durante su transmisión.

Criterio	Valor	Tu evaluación
El programa se escribe en un block de notas y se compila con gcc	1	1
Se incluye todo el código	1	1
El mapa de memoria refleja todas las variables utilizadas en el programa y han sido seleccionadas de forma consciente.	1	1
Las imágenes de la ejecución son claras	1	1
Se incluyen las conclusiones solicitadas	1	1
TOTAL	5	5



M. en C. Nidia A. Cortez Duarte



Nombres: Hernández Rodríguez Armando Giovanni



Práctica 2.-Checksum

Anexo Código Fuente

```
unsigned short int checksum = 0x00000; // 2 bytes = 16 bits
unsigned short int getChecksum(unsigned char frame[] ){
     unsigned char i = 0x00;
unsigned char aux = 0x00;
checksum = 0x0000; // 16 bits = 2 bytes
            //Acarreo fuera de los 16 bit
if(i==17){
void correctChecksum(unsigned char frame[]){
      frame[10]= 0x00;
frame[11] = 0x00;
       if((checksum >> 8) == frame[10] & (checksum & 0x00ff)== frame[11]){
  printf("\nChecksum correcto");
            printf("\nChecksum incorrecto");
printf("\nEl checksum correcto es: %.4x", checksum);
int main(){
     unsigned char frame1[] = {0x45, 0x00, 0x01, 0xe2, 0xd7, 0xdb, 0x40, 0x00, 0x80, 0x06, 0x00, 0x00,
0, 0xa8, 0x01, 0x43, 0x94, 0xcc, 0x3a, 0xdd};
unsigned char frame2[] = {0x45, 0x00, 0x01, 0x9c, 0xd7, 0xde, 0x40, 0x00, 0x80, 0x06, 0x88, 0x9d,
0, 0xa8, 0x01, 0x43, 0x94, 0xcc, 0x3a, 0xdd};
checksum = 0x0000;
     checksum = getChecksum(frame1);
printf("\nPruebal\nChecksum de la trama1: %.4x\n",checksum);
      return 0;
```