1.- Desarrolle un programa Shell que permita enviar tarjetas navideñas "personalizadas". Se tiene un archivo "cuerpo.txt" con el texto sin el saludos. Se tiene otro archivo "destinatarios.txt" con dos columnas, la primera contiene el nombre y la segunda la dirección de correo del destinatario.
Su programa enviará correos electrónicos a cada destinatario similares a:
Hola Domingo, /* éste es el saludo que debe cambiar según destinatario.txt */
Los mejores deseos en esta Navidad. /* de aquí hacia abajo de cuerpo.txt */
Saludos,

Agustín

Respuesta:

```
#! /bin/bash
# Pregunta 1 Certamen 1, 1 Sem.2003
# destinatarios.txt contiene lista de pares organizados por
# línea, cada una contiene el Nombre y la segunda el correo
# electrónico del destinatario.
# cuerpo.txt contiene el contenido del mensaje sin el saludo inicial.
# sintaxis: p1.sh < destinatarios.txt

while read line
do
    set `echo $line`
    echo "Hola $1," > /tmp/$2$$
    cat cuerpo.txt >> /tmp/$2$$
    mail $2 < /tmp/$2$$
    rm -f /tmp/$2$$
done
```

Otra solución:

```
while read nombre email
do
    echo "Hola $nombre," > /tmp/t$$
    cat cuerpo.txt >> /tmp/t$$
    mail $email < /tmp/t$$
    rm -f /tmp/t$$
done < destinatarios.txt
```

Otra: En este caso se ejecuta p1.sh sin parámetros.

```
while read nombre correo
do
    echo "Hola $nombre,
    `cat cuerpo.txt`" | mail $correo
done < destinatarios.txt
```

2.-
a. ¿Bajo qué secuencia se crea un proceso Zombie?

Un proceso Zombie se crea cuando un proceso hijo termina y su proceso padre no toma del sistema operativo el estado de retorno del hijo.
La secuencia sería:

1.- Se crea proceso hijo.

2.- Proceso hijo termina. /* esto da origen a la aparición de un Zombie. */

3.- Después de un rato el proceso padre decide lIer es estado de término del hijo. /* esta operación hace que el proceso zombie termine*/

b. ¿Qué función tiene la peculiaridad de ser invocada una vez pero retornar "dos veces"?
La función fork(). un proceso la llama y retorna en dos procesos distintos.

c. ¿Qué función tiene la peculiaridad de ser invocada y posiblemente no retornar?
La función exec en sus distintas versiones. Esta función no retorna cuando su invocación es exitosa. En otras palabras el código que le sigue a exec no se ejecuta si el llamado es exitoso.

d. ¿En qué consiste el ataque inundación SYN?
Consiste en iniciar el establecimiento de conexión con un servidor TCP y no responder con un acuse de recibo del mensaje SYN/ACK recibo devuelta. Con esto se termina por copar el número de procesos definidos por listen() y el servicio queda bloqueado para todo requerimiento futuro.

e. Mencione una ventaja de "fifos" frente a "pipes".
Los fifos permiten la comunicación entre procesos independientemente si éstos están o no relacionados (padre/hijo, o hermanos, etc). Los pipes tiene que poseer un antecesor común. Los fifos incluso se pueden establecer entre usuarios distintos.

f. ¿Qué pasa si en la especificación de la dirección IP destino del servidor a ser contactado por un cliente vía TCP, el programador del cliente omite la conversión htonl()?
Si esto no se hace la conexión sólo se establecerá si la arquitectura de la máquina donde se ejecuta el cliente es usa orden "big-endian" (arquitectura motorola, SPARC)

3.- Haga un programa en C que tomando como argumento otra dirección IP, verifique que es multicast y luego indique si generaría "colisión" a nivel de dirección MAC Ethernet con 226.1.1.178.

```
int main(int argc, char * argv[])
{
  unsigned long int IPfija, IPparametro;
  IPfija=inet_addr("226.1.1.178");
  IPparametro=inet_addr(argv[1]);
  if ( (IPparametro & htonl(0xF0000000)) != htonl(0xE0000000)) {
    printf("Su dirección no es multicast\n");
    exit(0);
```

```
  }

  if ( (IPfija & htonl(0x007FFFFF)) == (IPparametro & htonl(0x007FFFFF)))
    printf("Colisión con 226.1.1.178\n");
  else
    printf ("NO Colisión con 226.1.1.178\n");
}
```

4.- Se desea imponer un "timeout" a la lectura de un dato. Haga un programa espere por el ingreso de una línea de texto. Si el usuario lo hace dentro de 10 segundos, el programa retornará el número de caracteres leídos y termina; sino, el programa escribe "adiós" y termina.

```
#include <netdb.h>
int main(void)
{
  fd_set readfd;
  int n;
  struct timeval tv;
  char entrada[256];

  tv.tv_sec = 5;
  tv.tv_usec=0;
  FD_ZERO(&readfd);
  FD_SET(0,&readfd);
  n=select(FD_SETSIZE, &readfd, NULL, NULL, &tv);
  if ( n >0)
    printf("Usted ingresó %i caracteres \n",read(0, entrada, 256));
  else
    printf("Adiós\n");
}
```

5.- Se presenta la siguiente situación. Máquina A y B están en la misma LAN y se usa TTL=2:
Máquina A: Tiene dos aplicaciones

- Una tiene un servidor UDP corriendo en el puerto 2222.
- Otra es un receptor Multicast para recibir del grupo 234.5.5.5 puerto 3333.

Maquina B: Envía un datagrama a la dirección 234.5.5.5 puerto 2222.

a) ¿Habría alguna forma para que alguna de las aplicaciones de A reciba el datagrama enviado?
Sí.
b) Dependiendo de su respuesta anterior indique:
Si no, explique ¿por qué?.
Si sí, ¿Bajo qué condiciones la máquina A recibe el datagrama? ¿Qué receptor lo recibe el unicast o el multicast?.

Lo recibe si en la programación del servidor UDP se definió que aceptaba paquetes arribando a cualquier interfaz (INADDR_ANY).

Como la aplicación multicast debe incorporarse al grupo multicast, la capa IP queda programa como si tuviera una interfaz más. Por otro lado por tratarse aplicaciones UDP, las aplicaciones unicast y multicast comparten los puertos. Luego, en estas condiciones el servidor UDP recibirá el paquete enviado al grupo multicast.

No hay forma para que el receptor Multicast reciba el paquete.