



Facultad de Ingeniería

Escuela de Ingeniería en Bioinformática

Informe Proyecto

Administración de Sistemas

Alumno:

Nicolás Rojas Poblete

Matrícula:

2016430002

E-mail:

nicrojas16@alumnos.utalca.cl

Profesor:

Alejandro Valdés

Módulo:

Administración de Sistemas

Minor:

Desarrollo de Software y
Administración de Sistemas

Índice

1. Introducción	5
2. Compilando programas, librerías, kernel	6
2.1. Compilando el compilador GCC	6
2.1.1. Prerequisitos GCC 9.3	6
2.1.2. Descarga y Configuración GCC 9.3	7
2.1.3. Instalación GCC 9.3	8
2.2. Compilando Librería KD-tree	8
2.2.1. Creando Programa con la librería KD-tree	9
2.2.2. Compilación y Prueba del programa	11
2.3. Compilando el Kernel	12
2.3.1. Descarga y Configuración	12
2.3.2. Screen y Compilación	14
2.3.3. Instalación de la imagen	15
3. Acceso Remoto	16
3.1. Telnet	16
3.2. SSH	16
3.2.1. Configuración SSH	17
3.2.2. Conexión por túnel	18

3.2.3. SSHFS: Montar archivos remotos	18
4. Transferencia de Datos	19
4.1. TFTP	19
4.2. FTP	20
4.2.1. SSL/TLS: Cifrar el tráfico de FTP	23
5. Resolución de Nombres - DNS	25
6. Sistema de Archivos por Red: NFS	27
6.1. Cambiar permisos de la carpeta compartida	28
6.2. Montar automáticamente	29
7. Conclusión	31

Índice de figuras

1.	Makefile importando librería KD-Tree	11
2.	Ejecución del programa con KD-tree	11
3.	Menú de Configuración del Kernel	13
4.	Configuración de la familia del CPU	13
5.	Archivo de Configuración del Kernel	13
6.	Archivo config de ssh para establecer conexión remota mediante un túnel . .	18
7.	Puertos de red TCP/UDP en uso	19
8.	Configuración de proftpd para usuarios anónimos	21
9.	Configuración de archivo db.nicrojas.cl	25
10.	Configuración de archivo /etc/fstab	30

1. Introducción

El aumento del acceso a Internet en las últimas décadas ha significado un aumento exponencial en la cantidad de servidores encargados de alojar diferentes servicios como transferencia de archivos, servicios web, resolución de nombres de dominio, etc. Por otro lado el aumento en la capacidad de computo ha permitido tener disponible una mayor cantidad de supercomputadores para lo que se suele usar algún servicio de acceso remoto para correr diferentes tipos de cálculos, estos cluster de computadores suelen contar con acceder con un sistema de archivos compartidas en red para almacenar toda la información generada.

El objetivo de este proyecto:

- Comprender el proceso de compilación de programas y librerías, realizando diferentes pruebas y versiones del compilador.
- Configurar el acceso remoto usando diferentes protocolos como telnet y ssh.
- Evaluar los diferentes protocolos para la transferencia de archivos.
- Montar un sistema de archivos en la red, configurando el acceso y permisos.

2. Compilando programas, librerías, kernel

2.1. Compilando el compilador GCC

El compilador es muy importante dentro de los sistemas informáticos, se encargan de traducir el código fuente de los programas a un lenguaje que pueda ejecutar la máquina. En sistemas Linux por lo general viene incluido el compilador GCC en una versión defecto. Para ver la versión por defecto se puede ejecutar el siguiente comando.

```
$ gcc --version
gcc (Debian 8.3.0-6) 8.3.0
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

En caso de que no se encuentre instalado, se puede instalar desde los repositorios de su distribución por apt u otro gestor de paquetes.

```
$ sudo apt install gcc
```

2.1.1. Prerequisitos GCC 9.3

En este caso se procederá a instalar la versión 9.3 de GCC a partir de la versión 8.3 que viene en nuestro sistema instalada.

Según el sitio web de manuallinux.eu^[1] los requisitos para compilar gcc 9.3 son los siguientes:

```
Gawk - (5.0.1)
M4 - (1.4.18)
Libtool - (2.4.6)
Make - (4.3)
Bison - (3.5.3)
Flex - (2.6.4)
Automake - (1.16.1)
Autoconf - (2.69)
Gettext - (0.20.1)
Gperf - (3.1)
Texinfo - (6.7)
```

Librerías en Desarrollo

```
Gmp - (6.2.0)
Mpfr - (4.0.2)
Mpc - (1.1.0)
ISL - (0.18)
```

2.1.2. Descarga y Configuración GCC 9.3

Una vez revisadas las dependencias se procede a descargar el código fuente de GCC 9.3, esto en el ftp oficial.

```
$ wget ftp://ftp.mirrorservice.org/sites/sourceware.org/pub/gcc/releases/gcc-9.3.0/gcc-9.3.0.tar.gz
```

Para descomprimir

```
$ tar -xzf gcc-9.3.0.tar.gz
```

Luego es necesario entrar en la carpeta y dentro de los archivos viene incluido un script que descarga las principales dependencias(Gmp,Mpfr,Mpc,ISL).

```
$ cd gcc-9.3.0/
$ contrib/download_prerequisites
```

Luego se puede crear una carpeta aparte en dónde se configure y construya el nuevo compilador.

```
$ cd ~
$ mkdir build-gcc9.3
$ cd build-gcc9.3/
```

Ahora se procede a ejecutar el ./configure con los siguiente parámetros.

```
$ ../gcc-9.3.0/configure --build=x86_64-linux-gnu --host=x86_64-linux-gnu \
--target=x86_64-linux-gnu --enable-shared --enable-threads=posix \
--enable-__cxa_atexit --enable-clocale=gnu --enable-languages=c,c++,fortran \
--prefix=/usr/local/gcc-9.3 --disable-multilib --program-suffix=-9.3
```

2.1.3. Instalación GCC 9.3

Una vez terminada la configuración y creado el archivo make file se procede a compilar

```
$ make -j 16
```

En este caso -j 16 indica la cantidad de CPUs disponibles en el sistema para la compilación. Dependiendo de la potencia puede tardar unos pocos minutos hasta un par de horas.

En este caso un CPU con 8 núcleos/16 hilos a 5.0 GHz la compilación solo tardó 15 minutos aproximadamente.

```
real    15m41.040s
user    140m26.060s
sys     3m52.618s
```

Luego como superusuario se copian los binarios al prefix indicado anteriormente en la configuración

```
$ sudo make install-strip
```

Por último es necesario exportar el directorio a la variable de entorno PATH

```
$ export PATH=$PATH:/usr/local/gcc-9.3/bin/
$ export export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/gcc-9.3/lib64/
```

Ahora se puede comprobar la version instalada

```
$ gcc-9.3 --version
gcc-9.3 (GCC) 9.3.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

2.2. Compilando Librería KD-tree

El primer paso es descargar el código fuente desde el repositorio oficial.

```
$ wget http://nuclear.mutantstargoat.com/sw/kdtree/files/kdtree-0.5.7.tar.gz
$ tar -xzvf kdtree-0.5.7
$ cd kdtree-0.5.7/
```


Una vez listo se procede a configurar la librería y hacer la compilación.

```
$ ./configure
$ make
```

2.2.1. Creando Programa con la librería KD-tree

Se usa como base el ejemplo test2 incluido en el código fuente y se modifica para que solicite un punto inicial y el radio de búsqueda en el árbol. Quedando un programa como el siguiente.

```
#include <assert.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <time.h>
#include "kdtree.h"

#define DEF_NUM_PTS 10

/* entrega la distancia euclidiana entre 2 puntos */
static double dist_sq( double *a1, double *a2, int dims );

/* numero random entre -10 y 10 */
static double rd( void );

int main(int argc, char **argv) {
    printf("Kd-tree_de_3_dimensiones\n");
    int i, num_pts = DEF_NUM_PTS;
    void *ptree;
    char *data, *pch;
    struct kdres *presults;
    double pos[3], dist;
    double pt[3];
    printf("Ingrese_un_numero_de_3_dimensiones:(Separado_por_un_espacio)_");
    scanf("%lf_%lf_%lf", &pt[0], &pt[1], &pt[2]);
    double radius;
    printf("Ingrese_el_radio_de_búsqueda:_");
    scanf("%lf", &radius);
    if(argc > 1 && isdigit(argv[1][0])) {
        num_pts = atoi(argv[1]);
    }
    if(!(data = malloc(num_pts))) {
        perror("malloc_failed");
        return 1;
    }
}
```

```

srand( time(0) );
/* Se crea el arbol kd-tree con 3 dimensiones */
ptree = kd_create( 3 );
/* Se agregan nodos con datos random al arbol */
for( i=0; i<num_pts; i++ ) {
    data[i] = 'a' + i;
    assert( 0 == kd_insert3( ptree, rd(), rd(), rd(), &data[i] ) );
}
/* busca los nodos mas cercanos al punto entregado en el radio entregado
   por e

   l usuario */
presults = kd_nearest_range( ptree, pt, radius );
/* imprime los resultados */
printf( "found_%d_results:\n", kd_res_size(presults) );

while( !kd_res_end( presults ) ) {
    /* se obtiene las coordenadas de cada nodo con resultados */
    pch = (char*)kd_res_item( presults, pos );

    /* calcula la distancia euclidiana entre el resultado del arbol y punto
       ingr

       esado por el usuario */
    dist = sqrt( dist_sq( pt, pos, 3 ) );

    /* Se imprimen los datos calculados */
    printf( "node_at_(%.3f,_%%.3f,_%%.3f)_is_%.3f_away_and_has_data=%c\n",
           pos[0], pos[1], pos[2], dist, *pch );
    /* Se recorre el siguiente resultado */
    kd_res_next( presults );
}
/* libera la memoria utilizada de las estructuras definidas anteriormente
   */
free( data );
kd_res_free( presults );
kd_free( ptree );

return 0;
}

static double dist_sq( double *a1, double *a2, int dims ) {
    double dist_sq = 0, diff;
    while( --dims >= 0 ) {
        diff = (a1[dims] - a2[dims]);
        dist_sq += diff*diff;
    }
    return dist_sq;
}

static double rd( void ) {
    return (double)rand()/RAND_MAX * 20.0 - 10.0;
}

```

Con el código listo se crea el archivo Makefile para hacer su compilación, en dónde se debe considerar la ruta en donde se encuentra la librería kd-tree. Quedando como se muestra en la siguiente imagen.

```
prefix=/usr/local
CC = gcc

CFLAGS = -std=c89 -pedantic -Wall -g -I..
SRC = test2.c
OBJ = test2.o
APP = test2
LDFLAGS= ../libkdtree.a -lm

all: $(OBJ)
    $(CC) $(CFLAGS) -o $(APP) $(OBJ) $(LDFLAGS)

clean:
    $(RM) $(OBJ) $(APP)

install: $(APP)
    install -m 0755 $(APP) $(prefix)/bin

uninstall: $(APP)
    $(RM) $(prefix)/bin/$(APP)

.PHONY: install
```

Figura 1: Makefile importando librería KD-Tree

2.2.2. Compilación y Prueba del programa

Se procede a compilar con make y ejecutar el el programa

```
$ make
$ ./test2
```

Como se mencionó anteriormente el programa solicita datos al usuario y luego realiza la búsqueda en un árbol kd-tree generado con números al azar. En la siguiente imagen se muestra su ejecución.

```
Kd-tree de 3 dimensiones
Ingrese un numero de 3 dimensiones: (Separado por un espacio) 1 1 3
Ingrese el radio de busqueda: 7
found 4 results:
node at (0.338, -4.283, 3.609) is 5.359 away and has data=j
node at (-0.699, 3.482, 2.057) is 3.152 away and has data=e
node at (-2.216, -3.809, -0.008) is 6.520 away and has data=c
node at (1.476, 7.077, 6.068) is 6.824 away and has data=b
```

Figura 2: Ejecución del programa con KD-tree

2.3. Compilando el Kernel

2.3.1. Descarga y Configuración

Primero es necesario saber que versión del kernel viene instalada en nuestro sistema, para ello usamos el siguiente comando.

```
$ uname -a
Linux admSist6 4.19.0-10-amd64 #1 SMP Debian 4.19.132-1 (2020-07-24) x86_64
GNU/Linux
```

Ahora se procede a descargar la última versión estable del kernel

```
$ wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.8.14.tar.xz
```

Se descomprime el paquete

```
$ tar axvf linux-5.8.14.tar.xz
```

En caso de haber realizado otra configuración en el actual directorio, es necesario limpiar con los siguientes comandos.

```
$ make distclean
$ make clean
```

Luego se debe copiar la configuración con la cuál se configuró el kernel instalado en nuestra máquina.

```
$ cp /boot/config-4.19.0-10-amd64 .config
```

Ahora se carga esta configuración anterior para el nuevo kernel

```
$ yes "" | make oldconfig
```

En este caso puede arrojar algunos errores de dependencias, en este caso fue necesario instalar flex y bison

```
$ apt install flex
$ apt install bison
```

También se puede desplegar un menú en la consola para ver todas las opciones de configuración

```
$ make menuconfig
```

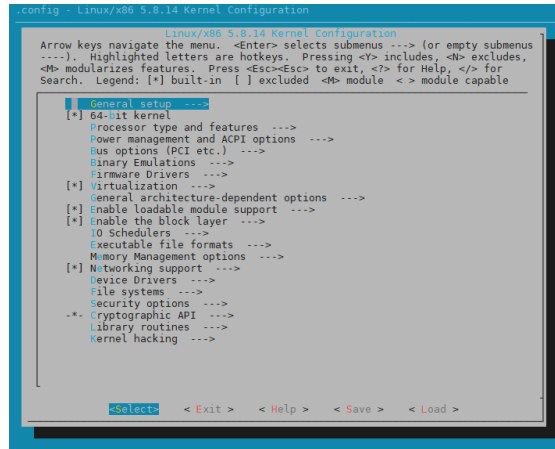


Figura 3: Menú de Configuración del Kernel

Se deben seleccionar las opciones dependiendo del cada sistema y los requerimientos. Para este caso se cambio la configuración específica al tipo de CPU y se desactivó drivers de audio que no serán utilizados.

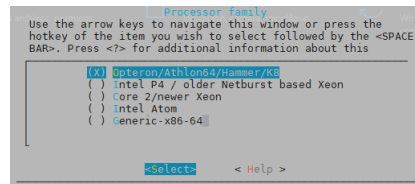


Figura 4: Configuración de la familia del CPU

Una vez configurado se guarda todo y si vemos el contenido del archivo .config se encuentran todos los parámetros de configuración para instalar el nuevo kernel.

```
# Automatically generated file; DO NOT EDIT.
# Linux/x86 5.8.14 Kernel Configuration
#
CONFIG_CC_VERSION_TEXT="gcc (Debian 8.3.0-6) 8.3.0"
CONFIG_CC_IS_GCC=y
CONFIG_GCC_VERSION=80300
CONFIG_LTO_VERSION=231010000
CONFIG_CLANG_VERSION=0
CONFIG_CC_CAN_LINK=y
CONFIG_CC_CAN_LINK_STATIC=y
CONFIG_CC_HAS_ASM_GOTO=y
CONFIG_CC_HAS_ASM_INLINE=y
CONFIG_IRQ_WORK=y
CONFIG_BUILDTIME_TABLE_SORT=y
CONFIG_THREAD_INFO_IN_TASK=y
#
# General setup
#
CONFIG_INIT_ENV_ARG_LIMIT=32
# CONFIG_COMPILE_TEST is not set
CONFIG_LOCALVERSION=""
# CONFIG_LOCALVERSION_AUTO is not set
CONFIG_BUILD_SALT="4.19.0-10-amd64"
CONFIG_HAVE_KERNEL_GZIP=y
CONFIG_HAVE_KERNEL_BZIP2=y
CONFIG_HAVE_KERNEL_LZMA=y
CONFIG_HAVE_KERNEL_XZ=y
CONFIG_HAVE_KERNEL_LZO=y
CONFIG_HAVE_KERNEL_LZ4=y
# CONFIG_KERNEL_GZIP is not set
# CONFIG_KERNEL_BZIP2 is not set
# CONFIG_KERNEL_LZMA is not set
CONFIG_KERNEL_XZ=y
# CONFIG_KERNEL_LZO is not set
# CONFIG_KERNEL_LZ4 is not set
.config 94361, 229152C
```

Figura 5: Archivo de Configuración del Kernel

Dentro de este archivo es importante modificar la línea que hace referencia al certificado del kernel anterior.

```
$ nano .config
CONFIG_SYSTEM_TRUSTED_KEYS=""
```

Además se puede deshabilitar la línea que crea una imagen dbg con la información del debug del kernel, esto permite que la imagen sea mucho más pequeña y así no quedarse sin espacio durante la compilación.

```
$ nano .config
CONFIG_DEBUG_INFO=n
```

2.3.2. Screen y Compilación

Ahora quedaría compilar el kernel, para lo cual se hizo una prueba en dos máquinas , una alojada en un servidor remoto y otra localmente.

- La máquina remota cuenta con 2 CPUs con 2 hilos de AMD a 2.7 GHz y 2 GB RAM
- La máquina local cuenta con 8 CPUs con 16 hilos de Intel a 5.0 GHz y 16 GB RAM

Para la máquina remota se configuró el uso de screen que es un software de terminales virtuales que permiten dejar el proceso de compilación corriendo sin necesidad de estar conectado todo el tiempo al servidor remoto.

Instalación de screen

```
$ apt install screen
```

Se crea una terminal en screen con nombre

```
$ screen -S kernel
```

Esto nos deja lista la terminal para lanzar un proceso de larga duración en nuestro servidor remoto. Para salir de esta terminal se debe apretar CTR + A + D. Para volver a la terminal creada se pueden listar las terminales de screen creadas

```
$ screen -ls
There is a screen on:
      30906.kernel      (10/10/2020 07:47:05 PM)      (Detached)
1 Socket in /run/screen/S-root.
```

Con esto podemos volver a conectarnos a esa terminal

```
$ screen -r 30906.kernel
```

Finalmente dejamos corriendo la compilación del kernel

```
$ make -j2 deb-pkg
```

Para la máquina local se usó

```
$ make -j16 deb-pkg
```

Los tiempos de ejecución en la máquina remota fueron los siguientes

```
real    96m49.182s
user    168m48.511s
sys     20m3.484s
```

Mientras que en la máquina local

```
real    11m7.837s
user    129m13.261s
sys     16m43.976s
```

2.3.3. Instalación de la imagen

Una vez terminado el proceso de compilación solo quedaría instalar la imagen .deb generada

```
$ cd ..
$ dpkg -i dpkg -i linux-image-5.8.14_5.8.14-1_amd64.deb
```

Terminada la instalación se reinicia la máquina para que arranque con el nuevo kernel.

```
$ shutdown -r now
```

Si la máquina vuelve arrancar se puede verificar el kernel instalado

```
$ uname -a
Linux admSist6 5.8.14 #1 SMP Sun Oct 11 04:00:29 -03 2020 x86_64 GNU/Linux
```

3. Acceso Remoto

3.1. Telnet

Primero es necesario la instalación de telnet

```
$ sudo apt install telnetd
```

Una vez instalado se debe correr el servicio para poder utilizarlo

```
$ sudo systemctl start inetd
```

Antes de establecer una conexión a otra máquina de forma remota, es necesario crear un nuevo usuario y asignándole un directorio y una contraseña.

Para crear un usuario y asignarle un directorio en /home usamos el siguiente comando.

```
$ sudo useradd -m -s /bin/bash nirojas
```

Luego para asignarle una contraseña

```
$ sudo passwd nirojas
New password:
Retype new password:
passwd: password updated successfully
```

Ahora ya podremos conectarnos mediante telnet

```
$ telnet 10.1.1.33 -l nirojas
Trying 10.1.1.33...
Connected to 10.1.1.33.
Escape character is '^]'.
Password:
```

3.2. SSH

SSH es un protocolo que permite al igual que telnet permite la conexión remota a otra máquina, la diferencia es que SSH utiliza un canal seguro, usando un sistema de cifrado para enviar y recibir la información.[\[2\]](#)

3.2.1. Configuración SSH

Para conectarse por ssh a un host remoto es necesario especificar el usuario seguido de la dirección ip o hostname de la máquina remota.

```
$ ssh nirojas@bioinfo.utalca.cl
nirojas@bioinfo.utalca.cl's password:
```

Para facilitar el ingreso se puede modificar el archivo config (crear si no existe), para crear un alias a la conexión remota.

```
$ vim .ssh/config

Host bioinfo
user nirojas
hostname bioinfo.utalca.cl
```

Además es posible generar una llave encriptada rsa para acceder al host remoto sin indicar la contraseña cada vez que se intente conectar.

Primero se genera una llave rsa que permitirá identificar a nuestro usuario y máquina local.

```
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/yanrri/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/yanrri/.ssh/id_rsa.
Your public key has been saved in /home/yanrri/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:OUkuYjpu7B/wQDdGJoWH84L0P+oBUfTHCprmp6UMTlQ yanrri@CEBOYAN
The key's randomart image is:
+---[RSA 2048]-----+
|  o*+                |
|  .+=+o .            |
|  .ooE+. o.          |
|  ..*+o.oo o         |
|  *o.+.. S           |
|+  .* + . .          |
|  .+o++ .            |
|=.Bo..               |
|  B+o.               |
+-----[SHA256]-----+
```

Luego es necesario crear el directorio .ssh/ en el host remoto en caso de que no haya sido creado con anterioridad.

```
$ ssh bioinfo mkdir -p .ssh
```

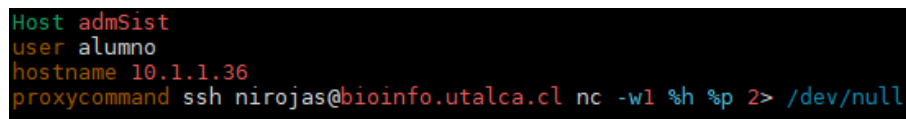
Finalmente se copia la llave generada anteriormente en el archivo que almacena las claves autorizadas para realizar conexión sin password.

```
$ cat ~/.ssh/id_rsa.pub | ssh bioinfo 'cat >> .ssh/authorized_keys'
```

3.2.2. Conexión por túnel

En algunos casos para conectarse a un equipo, se requiere un host remoto intermedio que haga un túnel al host remoto final.

El host intermedio se puede agregar a la configuración ssh quedando como la siguiente imagen.



```
Host admSist
  user alumno
  hostname 10.1.1.36
  proxycommand ssh nirojas@bioinfo.utalca.cl nc -w1 %h %p 2> /dev/null
```

Figura 6: Archivo config de ssh para establecer conexión remota mediante un túnel

La imagen muestra en 'proxycommand' el host que actúa como túnel para establecer la conexión con el host remoto final. En este caso se usa el servidor bioinfo.utalca.cl para poder conectarse a la máquina con dirección IP 10.1.1.36 con el usuario alumno.

Finalmente si se desea conectar a través de un túnel sin indicar contraseña por cada conexión, es necesario copiar la clave rsa tanto en el host intermedio como el host final.

3.2.3. SSHFS: Montar archivos remotos

Otro aspecto interesante de las conexiones remotas es la posibilidad de acceder a los archivos de un host remoto directamente desde nuestra máquina local, para ello existen varias opciones una de ellas es usar el protocolo seguro de SSH con el software SSHFS que permite montar un directorio remoto como se muestra en el siguiente comando.

```
sshfs alumno@10.1.1.36:RemoteFolder/ ~/Documents/ -o ssh_command='ssh -J nirojas@bioinfo.utalca.cl'
```

En este caso se agregó la opción *ssh_command* para lograr la conexión mediante un túnel.

4. Transferencia de Datos

4.1. TFTP

Corresponde a un protocolo de transferencia de archivos más sencillo y rápido que el clásico FTP ya que realiza un menor número de validaciones y no cuenta con un sistema de cifrado o autenticado. Trabaja sobre el puerto 69 en UDP.[3]

Primero será necesario instalar un software que trabaje con este protocolo por el lado del servidor, en este caso se escogió *tftpd-hpa*

```
$ sudo apt install tftpd-hpa
```

Una vez instalado se ejecuta el servicio.

```
$ sudo systemctl start tftpd-hpa
```

Para comprobar el puerto en el que se está ejecutando se puede ejecutar el comando `netstat`

```
root@admSist6:~# netstat -putna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22             0.0.0.0:*               LISTEN      348/sshd
tcp        0      0 0.0.0.0:23             0.0.0.0:*               LISTEN      18504/inetd
tcp        0      0 127.0.0.1:6010         0.0.0.0:*               LISTEN      13189/sshd: alumno@
tcp        0      0 10.1.1.36:22           10.1.1.71:39984         ESTABLISHED 13176/sshd: alumno
tcp        0      0 10.1.1.36:22           10.1.1.71:39982         ESTABLISHED 13169/sshd: alumno
tcp6       0      0 :::21                  :::*                    LISTEN      12924/proftpd: (acc
tcp6       0      0 :::22                  :::*                    LISTEN      348/sshd
tcp6       0      0 :::1:6010              :::*                    LISTEN      13189/sshd: alumno@
udp        0      0 0.0.0.0:69             0.0.0.0:*               LISTEN      589/in.tftpd
root@admSist6:~#
```

Figura 7: Puertos de red TCP/UDP en uso

Se puede ver que en la última línea el programa *tftpd* está usando el puerto UDP 69

Una vez arriba el servidor es necesario instalar un software por el lado del cliente para subir o bajar archivos del servidor. En este caso se instalará *tftp*.

```
$ sudo apt install tftp
```

Para conectarnos al servidor simplemente se debe indicar la dirección IP del servidor *tftp*

```
$ tftp 10.1.1.35
tftp>
```

Se abre una consola en donde se puede digitar el símbolo ? para ver los comandos disponibles.

```
tftp>?  
Commands may be abbreviated.  Commands are:  
  
connect      connect to remote tftp  
mode         set file transfer mode  
put          send file  
get          receive file  
quit         exit tftp  
verbose      toggle verbose mode  
trace        toggle packet tracing  
status       show current status  
binary       set mode to octet  
ascii        set mode to netascii  
rexmt        set per-packet retransmission timeout  
timeout      set total retransmission timeout  
?            print help information
```

Para enviar un archivo al servidor

```
tftp> put top_secret.txt  
Received 15 bytes in 0.0 seconds  
tftp>
```

Para obtener un archivo que se encuentre en el servidor

```
tftp> get top_secret.txt  
Received 15 bytes in 0.0 seconds  
tftp>
```

4.2. FTP

Es el protocolo estándar para la transferencia de archivos. Usa normalmente el puerto 20 y 21 en TCP.

Al igual que con TFTP existen diferentes softwares que implementan este protocolo del lado del servidor y del cliente. Incluso algunas opciones con interfaz gráfica como FileZilla y gFTP.

En este caso para el servidor se instalará el programa *proftpd*

```
$ sudo apt install proftpd-basic
```

Una vez instalado se puede levantar el servicio

```
$ sudo systemctl start proftpd.service
```

En la Figura 7 se puede ver también que el servicio proftpd esta trabajando en el puerto 21 en TCP.

Con el servidor funcionando podemos instalar el programa *ftp* del lado del cliente.

```
$ sudo apt install ftp
```

Para conectarse simplemente se debe indicar la dirección IP del servidor.

```
ftp 10.1.1.35
Connected to 10.1.1.35.
220 ProFTPD Server (Debian) [::1]
Name (10.1.1.35:alumno): nico
331 Password required for alumno
Password:
230 User nico logged in
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

En este caso pide una autenticación a través de un usuario y contraseña. Se puede configurar el servidor para permitir conexiones anónimas. Para esto es necesario configurar el archivo de configuración ubicado en */etc/proftpd/proftpd.conf*



```
<Anonymous ~ftp>
  User ftp
  Group nogroup
  # We want clients to be able to login with "anonymous" as well as "ftp"
  UserAlias anonymous ftp
  # Cosmetic changes, all files belongs to ftp user
  DirFakeUser on ftp
  DirFakeGroup on ftp

  RequireValidShell off

  # Limit the maximum number of anonymous logins
  MaxClients 10

  # We want 'welcome.msg' displayed at login, and '.message' displayed
  # in each newly chdir'd directory.
  DisplayLogin welcome.msg
  DisplayChdir .message

  # Limit WRITE everywhere in the anonymous chroot
  <Directory *>
    <Limit WRITE>
      DenyAll
    </Limit>
  </Directory>

  # Uncomment this if you're brave.
  # <Directory incoming>
  # # Umask 022 is a good standard umask to prevent new files and dirs
  # # (second param) from being group and world writable.
  # Umask <Limit READ WRITE> 022 022
  #
  # DenyAll
  # </Limit>
  # <Limit STOP>
  # AllowAll
  # </Limit>
  # </Directory>
</Anonymous>
```

Figura 8: Configuración de proftpd para usuarios anónimos

Estas líneas estarán comentadas por defecto en el archivo de configuración, al modificar esto será necesario reiniciar el servicio para aplicar los cambios.

```
$ sudo systemctl restart proftpd.service
```

Al conectarse simplemente debe especificarse el usuario ftp y omitir la contraseña.

```
$ ftp 10.1.1.35
Connected to 10.1.1.35.
220 ProFTPD Server (Debian) [::ffff:10.1.1.36]
Name (10.1.1.36:alumno): ftp
331 Anonymous login ok, send your complete email address as your password
Password:
230-Welcome, archive user ftp@stgo-10.1.1.35.in.utralca.cl !
230-
230-The local time is: Mon Nov 16 00:22:33 2020
230-
230-This is an experimental FTP server. If you have any unusual problems,
230-please report them via e-mail to <root@admSist6>.
230-
230 Anonymous access granted, restrictions apply
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

En este caso ftp tiene más opciones que tftp como por ejemplo navegar entre directorios o listar los archivos. Para ver todos los comandos disponibles se puede ver de la misma forma que tftp.

```
>ftp ?
Commands may be abbreviated.  Commands are:

!            dir            mdelete      qc           site
$            disconnect     mdir         sendport    size
account      exit            mget         put          status
append       form           mkdir        pwd          struct
ascii        get            mls          quit         system
bell         glob           mode         quote        sunique
binary       hash           modtime     recv         tenex
bye          help           mput        reget        tick
case         idle           newer        rstatus     trace
cd           image          nmap         rhelp        type
cdup         ipany          nlist        rename       user
chmod        ipv4           ntrans       reset        umask
close        ipv6           open         restart     verbose
cr           lcd            prompt       rmdir        ?
delete       ls             passive      runique
debug        macdef         proxy        send
```

Ejemplo para descargar un archivo del servidor

```
>ftp get welcome.msg
local: welcome.msg remote: welcome.msg
200 PORT command successful
150 Opening BINARY mode data connection for welcome.msg (170 bytes)
226 Transfer complete
170 bytes received in 0.00 secs (3.3776 MB/s)
ftp>
```

4.2.1. SSL/TLS: Cifrar el tráfico de FTP

Otra de las grandes diferencias que tiene FTP con respecto a TFTP es que permite configurar conexiones seguras a través de un certificado SSL (capa de sockets seguros). Esto permite que la información que se transmite esta cifrada por lo que cualquier persona que intente interceptar el tráfico no podrá acceder ni modificar los archivos. TLS corresponde a una versión actualizada de SSL con mayor seguridad.

Lo primero es incluir el archivo de configuración para TLS descomentando la siguiente línea.

```
Include /etc/proftpd/tls.conf
```

Si es necesario se debe crear el archivo `tls.conf` en el directorio `/etc/proftpd/`. Dentro debe estar lo siguiente:

```
# vim tls.conf
<IfModule mod_tls.c>
    TLSEngine on
    TLSLog /var/log/proftpd/tls.log
    TLSProtocol SSLv23
    TLSRSACertificateFile /etc/ssl/certs/proftpd.crt
    TLSRSACertificateKeyFile /etc/ssl/private/proftpd.key
    TLSRequired on
</IfModule>
```

Ahora hace falta generar los certificados público y privado.

```
# proftpd-gencert
Generating a RSA private key
.....+++++
.....+++++
writing new private key to '/etc/ssl/private/proftpd.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:CL
State or Province Name (full name) [Some-State]:Maule
Locality Name (eg, city) []:Curico
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Utalca
Organizational Unit Name (eg, section) []:Bioinformatica
Common Name (e.g. server FQDN or YOUR name) []:Nicolas
Email Address []:nicrojas16@alumnos.utralca.cl

Use the following information in your ProFTPD configuration:

TLRSRSCertificateFile      /etc/ssl/certs/proftpd.crt
TLRSRSCertificateKeyFile   /etc/ssl/private/proftpd.key

See /etc/proftpd/tls.conf for suggested TLS related configuration
items and include that file in your /etc/proftpd/proftpd.conf file.
```

Por último se reinicia el servicio de proftpd.

```
# systemctl restart proftpd.service
```

Para probar la conexión es necesario un cliente con soporte para conexiones seguras como lo es lftp.

Para instalar lftp

```
$ sudo apt install lftp
```

Para conectarnos al servidor

```
$ lftp sftp://alumno@10.1.1.35
Password:
lftp alumno@10.1.1.35:~> get hola.txt
13 bytes transferred
```


5. Resolución de Nombres - DNS

Con el crecimiento exponencial de las redes en internet han surgido algunos problemas para ubicar y memorizar las direcciones IP de cada uno de los servicios que vistamos en la web. Para ello la solución fue asignar nombres a cada uno de los servicios web para que sea mucho más fácil de recordar y promocionar que una dirección IP, con ello nacen nombres de dominio como google.com, facebook.com, twitter.com, etc. Estos nombres son traducidos a la dirección IP correspondiente mediante un software alojado en un servidor. Por lo tanto cuando nos conectamos a google.com por ejemplo, lo primero que se realiza es mandar la solicitud al servidor DNS para que traduzca este nombre, este nos devuelve la dirección IP del sitio web y a través de esta nos conectamos.

El software que se utiliza para hacer esta traducción es *bind9*. A continuación se muestra su instalación junto con herramientas para comprobar su funcionamiento.

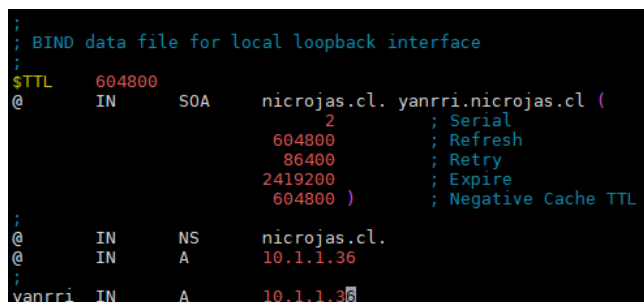
```
$ sudo apt install bind9 dnsutils dnstracer
```

Una vez instalado se debe configurar y crear los archivos en la carpeta */etc/bind/* para añadir un nuevo dominio y subdominios.

Para añadir se puede copiar el archivo db.local a otro con el nombre del dominio que vamos a crear por ejemplo db.nicrojas.cl

```
# cd /etc/bind/  
# cp db.local db.nicrojas.cl  
# vim db.nicrojas.cl
```

Ahora se debe editar el archivo y se debe cambiar el nombre localhost por el nombre de dominio y subdominio a crear. En la parte final del archivo se debe especificar la IP a la que hace referencia estos nombres, primero el dominio y luego los subdominios como se muestra en la Figura 9.



```
;  
; BIND data file for local loopback interface  
;  
$TTL 604800  
@ IN SOA nicrojas.cl. yanrri.nicrojas.cl (  
        2      ; Serial  
        604800 ; Refresh  
        86400  ; Retry  
        2419200 ; Expire  
        604800 ) ; Negative Cache TTL  
;  
@ IN NS nicrojas.cl.  
@ IN A 10.1.1.36  
;  
yanrri IN A 10.1.1.36
```

Figura 9: Configuración de archivo db.nicrojas.cl

En este caso se configuró que el subdominio este alojado a la misma máquina que el dominio principal pero eventualmente podríamos tener cada subdominio en alguna máquina dedicada para este.

El siguiente archivo a crear es el de `named.conf.nicrojas.cl` el cual indica la zona con nuestro nombre de dominio e incluye la ruta del archivo `db.nicrojas.cl`

La estructura de este nuevo archivo es de la siguiente forma:

```
# vim named.conf.nicrojas.cl
zone "nicrojas.cl" {
    type master;
    file "/etc/bind/db.nicrojas.cl";
};
```

Por último en el archivo `named.conf` se encuentran todas las zonas de nuestro servidor DNS, por lo que hay que agregar la siguiente línea para agregar el dominio creado.

```
# echo "include \"/etc/bind/named.conf.nicrojas.cl\";" >> /etc/bind/named.conf
```

Una vez realizado esto se debe reiniciar el servicio para actualizar las configuraciones.

```
# systemctl restart bind9
```

Para comprobar el funcionamiento del dominio podemos conectarnos a la máquina de Jessica(10.1.1.35) y probar el nombre de dominio y subdominio creado.

```
$ ssh nico@10.1.1.35
```

Dentro de esta máquina debemos indicar la dirección IP del servidor DNS a utilizar, para ello se debe modificar el archivo `/etc/resolv.conf` y agregar la siguiente línea.

```
# vim /etc/resolv.conf
nameserver 10.1.1.36
```

La línea debe agregarse arriba del resto de `nameserver`, para que funcione correctamente.

Por último probamos su funcionamiento con la funcionalidad `nslookup` obteniendo lo siguiente.

```
$ nslookup nicrojas.cl
Server:          10.1.1.36
Address:         10.1.1.36#53

Name:   nicrojas.cl
Address: 10.1.1.36
```

Se prueba también el subdominio.

```
$ nslookup yanrri.nicrojas.cl
Server:          10.1.1.36
Address:         10.1.1.36#53

Name:   yanrri.nicrojas.cl
Address: 10.1.1.36
```

Como se puede observar la consulta de nombre la hace al servidor DNS alojado en nuestra máquina remota (10.1.1.36), usando el puerto 53 que es el predeterminado para este servicio. Por último entrega la dirección a la que hace referencia el dominio/subdominio que en este caso corresponde a la misma del servidor DNS.

Otra prueba interesante es poder hacer ping usando el nombre de dominio/subdominio.

```
$ ping yanrri.nicrojas.cl -c 4
PING yanrri.nicrojas.cl (10.1.1.36) 56(84) bytes of data.
64 bytes from 10.1.1.36 (10.1.1.36): icmp_seq=1 ttl=64 time=0.106 ms
64 bytes from 10.1.1.36 (10.1.1.36): icmp_seq=2 ttl=64 time=0.193 ms
64 bytes from 10.1.1.36 (10.1.1.36): icmp_seq=3 ttl=64 time=0.192 ms
64 bytes from 10.1.1.36 (10.1.1.36): icmp_seq=4 ttl=64 time=0.195 ms

--- yanrri.nicrojas.cl ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 54ms
rtt min/avg/max/mdev = 0.106/0.171/0.195/0.040 ms
```

6. Sistema de Archivos por Red: NFS

NFS es otro protocolo, mediante el cual se pueden compartir archivos de forma distribuida, rápida y sencilla mediante el uso de la red. Sin embargo, no es un protocolo con cifrado ya que busca la mayor rapidez posible en el acceso a los archivos. Debido a esto suele implementarse en entornos de red privados para evitar problemas de seguridad. [4]

Primero se debe configurar el servidor principal que alojará la carpeta compartida en la red. Para ello es necesario instalar *nfs-kernel-server*

```
$ sudo apt install nfs-kernel-server
```

Luego como root se puede crear una carpeta la que será configurada para ser compartida en la red.

```
# mkdir /NFS_Folder
```

Luego en el archivo */etc/exports* se debe agregar la carpeta creada junto con la dirección IP que tendrá acceso a esta con los permisos.

```
# vim /etc/exports
/NFS_Folder          10.1.1.0/24(rw,sync,no_subtree_check)
```

En este caso se ha configurado para que se pueda compartir con toda la red privada especificando una máscara de subred de /24 bits.

Solo queda reiniciar el servicio para guardar las configuraciones.

```
# systemctl restart nfs-kernel-server.service
```

Ahora por parte del cliente es necesario instalar *nfs-common* para montar la carpeta compartida en red.

```
$ sudo apt install nfs-common
```

Se crea una carpeta en donde se montará la carpeta compartida

```
$ mkdir shared/
```

Como root se monta la carpeta compartida creada en el servidor, sobre la carpeta creada en el cliente.

```
# mount -t nfs4 10.1.1.35:/shared_Nico /home/alumno/shared/
```

Para esto se usó como servidor la máquina de Jessica(10.1.1.35) en dónde se creo la carpeta compartida *shared_Nico* la cual contiene un archivo para probar su funcionamiento. Una vez montado en el cliente(10.1.1.36) se ven los archivos en ambos lados.

```
root@admSist5:/shared_Nico# ls
soy_un_archivo.txt
```

```
alumno@admSist6:~/shared$ ls
soy_un_archivo.txt
```

6.1. Cambiar permisos de la carpeta compartida

Con las configuraciones anterior solamente el servidor podrá editar/modificar los archivos, mientras que el cliente solo tendrá acceso de lectura, esto pese a que en la configuración

de */etc/exports* tenga permisos para escribir también. Para revertir esto será necesario cambiar permisos también a la carpeta creada en el servidor y también se puede agregar a la configuración que los archivos queden con propietario de un usuario.

Para asignar los permisos a la carpeta compartida

```
# chmod 777 /shared_Nico/  
# chmod u+rwX /shared_Nico/
```

Cambiar propietario a la carpeta compartida y sus archivos

```
# chown alumno:alumno -R shared_Nico/
```

Asignar el propietario desde configuración NFS.

```
# vim /etc/exports  
/home/alumno/shared_Nico 10.1.1.36(rw,sync,no_subtree_check,anonuid=1000,  
    anongid=1000)
```

En este caso se agrega el uid del usuario que quedará como propietario de los archivos que se agreguen a la carpeta compartida.

Para ver la información e id de los usuarios de la máquina.

```
$ cat /etc/passwd
```

6.2. Montar automáticamente

Una vez todo configurado pareciera estar listo una vez que se realiza el montaje, sin embargo al apagar o reiniciar la máquina es necesario volver a montar la carpeta compartida en la red. Esto se puede cambiar configurando el automontaje.

Por ejemplo al reiniciar y luego listar los archivos de la carpeta en dónde se había realizado anteriormente el montaje, se puede ver que esta se encuentra vacía.

```
# shutdown -r now  
$ cd ~/shared  
$ ls -ll  
total 0
```

Para montar de forma automática es necesario editar como root el archivo */etc/fstab* .

```
# vim /etc/fstab
```

Dentro del archivo se agrega la información del montaje primero la IP del servidor con la ruta de la carpeta y luego la carpeta local en dónde se hará el montaje. Luego una serie de parámetros como se ve en la siguiente figura.

```
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/vdal during installation
UUID=11201891-c858-4cc1-842e-7d7f2ddea053 / ext4 errors=remount-ro 0 1
# swap was on /dev/vda5 during installation
UUID=394675db-16cc-43d6-bdf5-3bcc20a59fe3 none swap sw 0 0
/dev/sr0 /media/cdrom0 udf,iso9660 user,noauto 0 0
10.1.1.35:/shared_Nico /home/alumno/shared nfs4 defaults,user,exec,auto 0 0
```

Figura 10: Configuración de archivo */etc/fstab*

El parámetro que permite el montaje automático es el de *auto*. Con esto guardamos el contenido del archivo y podemos comprobar reiniciando nuevamente y debería estar montado con los archivos compartidos en red.

```
# shutdown -r now
$ cd ~/shared
$ ls -ll
total 4
-rw-r--r-- 1 root root 49 Nov 22 05:14 soy_un_archivo.txt
```

Para desmontar la unidad simplemente con el comando *umount*.

```
# umount /home/alumno/shared
```

Sin embargo para que no se vuelva a montar cuando se reinicie la máquina será necesario modificar el archivo */etc/fstab* en el parámetro *auto* por *noauto*.

7. Conclusión

A modo de síntesis, se ha demostrado la importancia que tiene cada uno de los servicios y recursos que se pueden configurar en un sistema computacional. Muchos de estos servicios los usamos día a día en Internet.

Además cabe destacar que el uso de un protocolo u otro dependerá de las necesidades específicas de cada implementación para obtener el mejor rendimiento o la mayor seguridad según corresponda. Cada uno de los servicios permite configurar diferentes aspectos como acceso, permisos, seguridad, entre otros y siempre será necesario adaptarlo a los requerimientos.

Referencias

- [1] Manual Linux <https://manuallinux.eu/gcc.html>
- [2] SSH https://es.wikipedia.org/wiki/Secure_Shell
- [3] TFTP <https://es.wikipedia.org/wiki/TFTP>
- [4] NFS <https://www.extrahop.com/resources/protocols/nfs/>