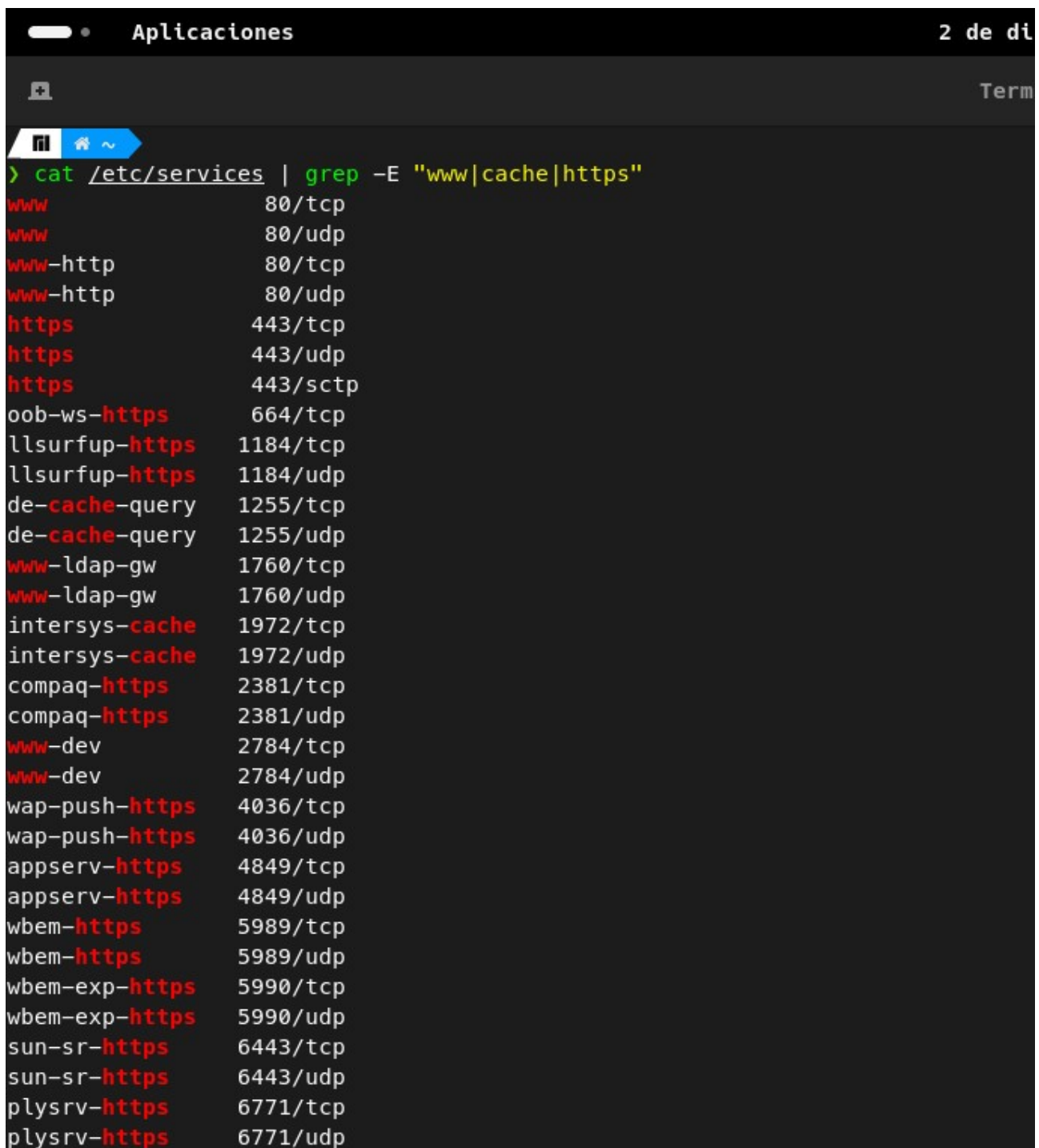


TEMA 4: SERVICIO HTTP

Ejercicio 1: Usando el comando `cat /etc/services` busca que puertos usa `www`, `caché` y `https` (recuerda que había un comando que te permitía buscar concretamente una palabra o fragmento de palabra, en este caso recomiendo `http`).

He usado el comando `cat /etc/services | grep -E "www|cache|https"`

He hecho dos capturas para poder ver todos los puertos que se usan.



```
> cat /etc/services | grep -E "www|cache|https"
www            80/tcp
www            80/udp
www-http       80/tcp
www-http       80/udp
https          443/tcp
https          443/udp
https          443/sctp
oob-ws-https   664/tcp
llsurfup-https 1184/tcp
llsurfup-https 1184/udp
de-cache-query 1255/tcp
de-cache-query 1255/udp
www-ldap-gw    1760/tcp
www-ldap-gw    1760/udp
intersys-cache 1972/tcp
intersys-cache 1972/udp
compaq-https   2381/tcp
compaq-https   2381/udp
www-dev        2784/tcp
www-dev        2784/udp
wap-push-https 4036/tcp
wap-push-https 4036/udp
appserv-https  4849/tcp
appserv-https  4849/udp
wbem-https     5989/tcp
wbem-https     5989/udp
wbem-exp-https 5990/tcp
wbem-exp-https 5990/udp
sun-sr-https   6443/tcp
sun-sr-https   6443/udp
plysrv-https   6771/tcp
plysrv-https   6771/udp
```

Aplicaciones		2 de di
		Term
appserv-https	4849/tcp	
appserv-https	4849/udp	
wbem-https	5989/tcp	
wbem-https	5989/udp	
wbem-exp-https	5990/tcp	
wbem-exp-https	5990/udp	
sun-sr-https	6443/tcp	
sun-sr-https	6443/udp	
plysrv-https	6771/tcp	
plysrv-https	6771/udp	
oracleas-https	7443/tcp	
oracleas-https	7443/udp	
sun-user-https	7677/tcp	
sun-user-https	7677/udp	
synapse-nhttps	8243/tcp	
synapse-nhttps	8243/udp	
pcsync-https	8443/tcp	
pcsync-https	8443/udp	
https-wmap	8991/tcp	
https-wmap	8991/udp	
armcenterhttps	9295/tcp	
armcenterhttps	9295/udp	
tungsten-https	9443/tcp	
tungsten-https	9443/udp	
memcache	11211/tcp	
memcache	11211/udp	
emc-xsw-dcache	11723/tcp	
emc-xsw-dcache	11723/udp	
amt-soap-https	16993/tcp	
amt-soap-https	16993/udp	
commtact-https	20003/tcp	
commtact-https	20003/udp	

Ejercicio 2: Busca y realiza una comparativa entre la versión HTTP 1.0 y la versión

HTTP 1.1. Haz lo mismo entre las versiones HTTP1.X, la HTTP2.0 y la HTTP3.0.

HTTP/1.0 – Desarrollando expansibilidad

- La versión del protocolo se envía con cada petición: HTTP/1.0 se añade a la línea de la petición GET.
- Se envía también un código de estado al comienzo de la respuesta, permitiendo así que el navegador pueda responder al éxito o fracaso de la petición realizada, y actuar en consecuencia (como actualizar el archivo o usar el caché local de algún modo).

- El concepto de cabeceras de HTTP, se presentó tanto para las peticiones como para las respuestas, permitiendo la transmisión de metadatos y conformando un protocolo muy versátil y ampliable.
- Con el uso de las cabeceras de HTTP, se pudieron transmitir otros documentos además de HTML, mediante la cabecera.

HTTP/1.1 – El protocolo estándar, aclaró ambigüedades y agregó numerosas mejoras

- Una conexión podía ser reutilizada, ahorrando así el tiempo de reabrir la repetidas veces para mostrar los recursos empotrados dentro del documento original pedido.
- Enrutamiento ('Pipelining' en inglés) se agregó a la especificación, permitiendo realizar una segunda petición de datos, antes de que fuera respondida la primera, disminuyendo de este modo la latencia de la comunicación.
- Se permitió que las respuestas a peticiones, pudieran ser divididas en subpartes.
- Se agregaron controles adicionales a los mecanismos de gestión de la caché.
- La negociación de contenido, incluyendo el lenguaje, el tipo de codificación, o tipos, se añadieron a la específica, permitiendo que servidor y cliente, acordasen el contenido más adecuado a intercambiarse.
- Gracias a la cabecera, Host pudo ser posible alojar varios dominios en la misma dirección IP.

HTTP/2 – Un protocolo para un mayor rendimiento, tiene notables diferencias fundamentales respecto a la versión anterior HTTP/1.1

- Es un protocolo binario, en contraposición a estar formado por cadenas de texto, tal y como están basados en sus protocolos anteriores. Así pues no se puede leer directamente, ni crear manualmente. A pesar de este inconveniente, gracias a este cambio es posible utilizar en él técnicas de optimización.
- Es un protocolo multiplexado. Peticiones paralelas pueden hacerse sobre la misma conexión, no está sujeto pues a mantener el orden de los mensajes, ni otras restricciones que tenían los protocolos anteriores HTTP/1.x
- Comprime las cabeceras, ya que estas, normalmente son similares en un grupo de peticiones. Esto elimina la duplicación y retardo en los datos a transmitir.
- Esto permite al servidor almacenar datos en la caché del cliente, previamente a que estos sean pedidos, mediante un mecanismo denominado 'server push'.

HTTP/3 – todavía no está en uso ampliamente y el estándar aún no es el final.

- Es básicamente las mejoras propuestas por HTTP/2 sobre una nueva capa de transporte que no es TCP: QUIC. QUIC está basado en UDP, pero establece la forma de crear canales con control de flujo, cifrado y multiplexación para poder servir mejor a la web moderna.
- QUIC permite usar TLS para establecer los parámetros de cifrado e incluso permite adelantar el intercambio de información antes de negociar completamente los parámetros

de cifrado, haciendo un poco más débil el cifrado pero incrementando la velocidad de descarga.

Ejercicio 3: Mediante telnet intenta conectarte a www.apache.org y a www.microsoft.com y mira que versión de HTTP tienen. Para conectarte necesitas emplear la siguiente orden: [apache.org](http://www.apache.org) la 1.1 y [microsoft.com](http://www.microsoft.com) la 1.0

```
Aplicaciones 2 de dic 23:54
Terminal
> telnet www.apache.org 80
Trying 151.101.2.132...
Connected to www.apache.org.
Escape character is '^]'.
HEAD/
HTTP/1.1 400 Bad Request
Connection: close
Content-Length: 11
content-type: text/plain; charset=utf-8
x-served-by: cache-mad2200147

Bad RequestConnection closed by foreign host.
```

```
Aplicaciones 2 de dic 23:53
Terminal
> telnet www.microsoft.com 80
Trying 92.123.57.133...
Connected to www.microsoft.com.
Escape character is '^]'.
HEAD/
HTTP/1.0 408 Request Time-out
Server: AkamaiGHost
Mime-Version: 1.0
Date: Sat, 02 Dec 2023 22:53:20 GMT
Content-Type: text/html
Content-Length: 218
Expires: Sat, 02 Dec 2023 22:53:20 GMT

<HTML><HEAD>
<TITLE>Request Timeout</TITLE>
</HEAD><BODY>
<H1>Request Timeout</H1>
The server timed out while waiting for the browser's request.<P>
Reference&#32;&#35;2&#46;1a79dd58&#46;1701557600&#46;0
</BODY></HTML>
Connection closed by foreign host.
```