Exercicis de comandes de Linux relacionades amb el Nivell 7 del model TCP/IP (HTTP)

1.-a) Obre el Firefox de la màquina real i vés a una pàgina web qualsevol ¿Què veus si pulses llavors CTRL+SHIFT+E (o F12 i després cliques sobre la pestanya titolada "Xarxa" que hi apareixerà a la part inferior de la finestra) i pulses tot seguit F5 (per repetir la petició)? Digues el significat de les següents columnes que apareixen: "Estat", "Mètode", "Domini", "Fitxer", "Tipus", "Transferits" i "Mida"

aII) Clica amb el botó dret sobre el títol de qualsevol de les columnes mostrades i sel·lecciona les opcions "IP remota", "Protocol", "Galetes" i "Temps"->"Durada". ¿Què veus en cada cas?

b) Fixa't que a la zona superior esquerra d'aquesta nova "secció" apareguda a la finestra del navegador es mostra una barra on es té la possibilitat de filtrar la llista mostrada de peticions per alguna part de la seva URL (part del domini, o de la ruta del recurs, etc). Prova-la

bII) A la banda esquerra d'aquesta barra superior hi apareixen uns quants botons interessants que també es demana que els provis. Concretament, els botons que permeten...:

*Pausar/Reanudar el registre de les peticions mostrades en aquesta secció
*Crear i enviar una nova petició "ad-hoc" (indicant mètode, URL, "querystring", capçaleres i cos)
*Bloquejar peticions alguna part de la URL de les quals concordi amb el valor indicat
*Filtrar la llista de peticions segons el tipus de contingut demanat (HTML,CSS,Javascript,imatges...)
*Deshabilitar la catxé del navegador (per tal de realitzar les peticions sempre al destí real)
*Simular límits en l'ample de banda (per veure com es comporta el navegador en baixa connectivitat)
*(A la roda dentada): guardar la informació mostrada en un fitxer per analitzar-la posteriorment

c) Si sel.lecciones una petició qualsevol concreta de les que t'hagin aparegut, digues el significat de les seccions "Capçaleres de resposta" i "Capçaleres de sol.licitud" que apareixen sota la pestanya "Capçaleres" de la zona dreta de la pantalla

cII) ¿Què passa si, tenint sel·leccionada una petició qualsevol concreta, cliques sobre el botó "Edita i torna a enviar" que apareix a la part dreta sota la pestanya "Capçaleres" de la zona dreta de la pantalla?

cIII) ¿Quin és el significat de la informació que apareix sota la pestanya "Temps" corresponent a la petició sel·leccionada? ¿I el de la pestanya "Memoria cau"? ¿I el de la pestanya "Seguretat"? ¿I el de la pestanya "Resposta"? ¿I el de la pestanya "Galetes"?

d) Sel.lecciona una petició que sigui de tipus imatge ("png", "jpg", etc) i fes-hi clic amb el botó dret del ratolí. Al menú contextual que apareix, per a què serviria l'opció "Anomena i desa la imatge"? ¿I l'opció "Copia->Copiar com a curl"?

NOTA: Consulteu https://firefox-source-docs.mozilla.org/devtools-user/network_monitor
per més informació sobre aquesta secció

1BIS.-Vés a https://radar.cloudflare.com i respón:

a) ¿Quin percentatge del total de tràfic total a Internet és de tipus HTTP? (panell "Overview")

b) ¿Quin percentatge del total d'IPs a Internet són de tipus "v4" i quantes són "v6"? (panell "Overview")

c) ¿Quin percentatge del tràfic HTTP és de versió "1.1", quin és "2.0" i quin és "3.0" (panell "Overview")

d) ¿Quins són els dominis DNS més populars d'Internet? (panell "Overview")

e) ¿Quins són els navegadors més populars d'Internet? (panell "Overview")

Una buena manera de comprender el funcionamiento interno del protocolo HTTP es utilizando un cliente TCP simple como *ncat* porque con él deberemos generar las peticiones HTTP "a mano" y recibiremos las respuestas HTTP "tal cual". Así, pues, si abrimos un terminal y, por ejemplo, ejecutamos el comando *ncat www.hola.com 80* estaremos conectando con el servidor web de la revista del corazón, pero tendremos que indicarle qué queremos de él: el cursor nos estará indicando que debemos escribir la petición deseada (hay que ir rápido: si nos entretenemos el servidor cortará la conexión). Un ejemplo de petición podría ser éste:

```
GET / HTTP/1.1
Host:www.hola.com
```

donde el path "/" indica que queremos obtener la página inicial por defecto. Tras pulsar dos veces Enter (para dejar una línea en blanco indicando que la petición no tiene más cabecera ni cuerpo), *ncat* enviará dicha petición y al instante deberíamos ver por pantalla cómo obtenemos dos cosas: las líneas de cabecera devueltas por el servidor y tras una línea en blanco, el código de la página web solicitada (esto es, el de la página inicial del sitio "www.hola.com"). Más concretamente, las líneas de cabecera más relevantes (hay más) obtenidas con la petición de ejemplo anterior son:

HTTP/1.1 301 Moved Permanently
Date: Sun, 23 Aug 2023 18:27:08 GMT
Server: Apache
Expires: Thu, 19 Nov 2024 08:52:00 GMT
Location: https://www.hola.com
Content-Type: text/html; charset=UTF-8
Content-Length: 0

De las líneas anteriores se puede deducir que el servidor web Apache en realidad no nos está ofreciendo ningún contenido -es decir, no obtenemos ningún código de ninguna página en el cuerpo de la respuesta-; tan solo nos indica que la página inicial de Hola ("www.hola.com") ya no se encuentra disponible a través del protocol HTTP (puerto 80) sino que deberemos usar HTTPS (puerto 443). Si probamos entonces de escribir *ncat –ssl www.hola.com 443* y en la consola que se nos ofrece volvemos a probar la misma petición...:

```
GET / HTTP/1.1
Host:www.hola.com
```

...obtendremos, ahora sí, el código fuente de la página principal de Hola.

> NOTA: El parámetro *--ssl* es necesario para conseguir que *ncat* pueda manejar correctament conexiones cifradas con certificados TLS de servidor, que son el tipo de conexiones que se realizan mediante HTTP**S** (la "S" es de "Secure")

Notar que *ncat* es un simple cliente TCP que no entiende el contenido del recurso obtenido al realizar una petición GET (lo que normalmente es el código de una página web); así que lo que hace es simplemente "vomitarlo" por pantalla. Si la obtención de esta página la hubiera realizado un navegador, éste sí que habría procedido seguidamente a interpretar dicho contenido y el resultado de dicha interpretación lo habría "renderizado" gráficamente, mostrando finalmente la página web tal como la visualiza un usuario estándar.

Usar *ncat* para estudiar tráfico HTTP puede resultar un poco tedioso y, además, solo funciona para la versión 1.1 (porque a partir de la dos el protocolo es binario en vez de basarse en texto). Para una interacción con servidores web más cómoda y rápida, es más adecuado emplear un cliente HTTP propiamente dicho, tal como el comando *curl*, aplicación de terminal libre, gratuita y multiplataforma. Este programa es capaz, como ya sabemos, de realizar peticiones HTTP (y HTTPS y HTTP/2) de cualquier tipo a un servidor dado, mostrando la respuesta obtenida por pantalla.

2.-a) Arrenca una màquina virtual qualsevol i utilitza la comanda *ncat* (si no està instal.lada, instal.la-la) com a client HTTP per tal d'obtenir la pàgina http://www.redhat.com/en/open-source Llegeix els paràgrafs de teoria per saber com fer-ho i també pots mirar la pista de sota. ¿Quin codi de resposta i codi HTML veus? Per què? Fixa't en el valor de la capçalera de resposta anomenada "Location".

PISTA: Executa la comanda així: *ncat www.redhat.com 80* i, escriu, al terminal que apareix:

GET /en/open-source HTTP/1.1
Host:www.redhat.com
(línia en blanc)

NOTA: Si en comptes de fer la petició interactivament la volguessir fer mitjançant la comanda *echo* "entubant-la" al *ncat* directament, recorda d'afegir llavors a l'*echo* el paràmetre -e per a què els símbols \r\n s'interpretin correctament com a salts de línia, així: *echo -ne "GET /en/open-source HTTP/1.1\r\nHost:www.redhat.com\r\n\r\n" | ncat www.redhat.com 80*

aII) Utilitza el mateix *ncat* pero ara com a client HTTPS (és a dir, HTTP + TLS) per tal d'obtenir la pàgina https://www.redhat.com/en/open-source Mira la pista de sota per saber com fer-ho. ¿Quin codi de resposta i codi HTML veus ara? Per què?

PISTA: L'únic canvi que has de fer respecte l'apartat anterior és afegir el paràmetre *--ssl* a la comanda *ncat* i canviar el port de destí per a què sigui 443 en comptes de 80

b) Fes una petició POST contra la URL http://httpbin.org/post enviant-hi les parelles clau<->valor "a*=3"* i *"b=hi"*. Comprova que aquestes dades "hagin arribat bé" observant la resposta obtinguda (concretament, el que apareix dins de la secció anomenada "form")

PISTA:Executa la comanda així: *ncat httpbin.org 80* i, escriu, al terminal que apareix:

POST /post HTTP/1.1
Host:httpbin.org
Content-Length:8
Content-Type:application/x-www-form-urlencoded
(línia en blanc)
a=3&b=hi

NOTA: La capçalera "Content-Length" és necessària enviar-la per dir-li al servidor la mida de les dades que se li enviaran (la cadena "a=3&b=76" són 8 caràcters d'un byte cadascun). La capçalera "Content-Type" és necessària per dir-li al servidor el format en el qual s'envien aquestes dades i així aquest les pugui interpretar correctament (en aquest sentit, l'estructura "clau1=valor2&clau2=valor2..." és la que utilitzen per defecte els formularis web i aquest format és l'anomenat "application/x-www-form-urlencoded")

bII) Repeteix l'apartat anterior però aquest cop enviant les dades via HTTPS (és a dir, a https://httpbin.org/post)

bIII) Repeteix l'apartat anterior però aquest cop enviant les dades en format JSON (llegeix la pista següent):

PISTA:Executa la comanda així: *ncat --ssl httpbin.org 443* i, escriu, al terminal que apareix:

POST /post HTTP/1.1
Host:httpbin.org
Content-Length:16
Content-Type:application/json
(línia en blanc)
{"a":3,"b":"hi"}

c) OPCIONAL: Utilitza ara la mateixa comanda *ncat* pero ara com a servidor HTTP. Concretament, primer crea un fitxer anomenat "pag.html" amb el següent contingut...:

```
<html><body><h1>Benvingut</h1><p>M'agraden els macarrons</p></body></html>
```

...i després crea un script anomenat "servWeb.sh" amb el següent contingut:

```
#!/bin/bash
while [[ true ]] ; do
        { echo -ne "HTTP/1.1 200 OK\r\nContent-Length: $(wc -c pag.html)\r\n\r\n"; cat pag.html; } | ncat -l -p 8080
done
```

NOTA: La capçalera "Content-Length" és necessària enviar-la per dir-li al client el tamany de la pàgina web oferida i així fer-li saber quan l'ha rebut tota (i per tant quan no cal continuar tenint oberta la conexió): si no es digués aquest tamany el client es quedaria esperant a rebre més dades fins que finalment arribés el temps de tallar la conexió (timeout), però aquest comportament no és gens òptim.
NOTA: Les claus { } són importants perquè fan que la canonada rebi la sortida no només de la comanda *cat* sinó també de la comanda *echo* prèvia (i en el mateix ordre)

Executa l'script anterior i seguidament obre un navegador per escriure la direcció http://127.0.0.1:8080 a la seva barra de direccions. ¿Què veus? ¿Què apareix al terminal on s'està executant l'script? ¿Per què creus que no s'ha fet servir a l'script el paràmetre *-k* de *ncat*?

NOTA: Si al navegador obtens un error de conexió és possible que tinguis un tallafocs activat al teu sistema. Deshabilita'l temporalment amb la comanda *sudo systemctl stop firewalld* o *systemctl stop ufw* i torna-ho a intentar.
NOTA: Si es volgués fer servir *ncat* com a servidor HTTPS, a més dels paràmetres *-l* i *-p* caldria indicar també el paràmetre *--ssl* ja conegut però, a més, dos paràmetres extra més: *--ssl-cert (*per indicar la ruta del certificat a usar) i *--ssl-key* (per indicar la ruta de la clau privada a usar); el tema dels certificats+claus privades ho estudiarem més endavant.

Tots els apartats de l'exercici anterior s'han realitzat fent servir el protocol HTTP/1.1 en lloc del protocol HTTP/2 perquè aquest darrer fa tota la transmissió de dades entre els extrems en format binari (mentre que el primer la fa en text). Això fa que HTTP/2 sigui molt més eficient però, per contra, no sigui tan senzill d'escriure una petició HTTP/2 (ni interpretar la resposta) amb la comanda *ncat* o similar

3.-a) Després d'haver fet els primers apartats de l'exercici anterior, ¿per què creus que la comanda *curl https://www.redhat.com/en/open-source* no genera cap sortida a pantalla? Pista: afegeix-li el paràmetre *-v* (o millor, el paràmetre *-I*) per veure les capçaleres de resposta rebudes i dedueix-ne llavors la resposta. ¿Què passarà, llavors, si li afegeixes el paràmetre *-L*?

b) ¿A quin sol paràmetre de *curl* és equivalent escriure el conjunt dels paràmetres *-s*, *-D -* i *-o /dev/null* (així per exemple: *curl -s -D - -o /dev/null https://www.hola.com*)? Consulta la referència de teoria si cal.

Per defecte, si a *curl* se li indica una URL de tipus HTTP, emprarà per defecte la versió 1.1 del protocol però si se li indica una URL de tipus HTTPS, llavors emprarà per defecte la versió 2.0. Si es volgués forçar a *curl* a utilitzar la versió 2.0 en una URL de tipus HTTP (cosa molt estranya, d'altra banda), es pot afegir el paràmetre *--http2* En tot cas, dependrà del servidor si es pot utilitzar HTTP 2.0 per la comunicació o caldrà, en canvi, "rebaixar" a la versió anterior. Per saber si un determinat servidor implementa o no la versió 2.0 del protocol HTTP, es pot executar la següent comanda: *curl -vI --stderr - https://www.lecturas.com | grep ALPN* Aquesta comanda mostrarà dos missatges, un similar a *"ALPN: offers h2,http/1.1"*, enviat pel navegador al servidor informant-li de les versions amb les que és capaç de treballar i l'altre, el que ens interessa, que és "*ALPN: server accepted h2*", enviat pel servidor confirmant que, efectivament, utilitzarà la versió 2.0 del protocol; si aquest darrer missatge no és present, llavors podem deduir que el servidor no utilitzar HTTP/2

NOTA: El paràmetre *–stderr* – serveix per redireccionar la sortida d'error (que és on *curl* escriu totes les capçaleres mostrades gràcies a *-v*) a la sortida estàndard (que és dón agafa les dades la canonada "|"). Això és necessari per a què el text que "treu" *curl* (per defecte, ja ho hem dit, a la sortida d'error) pugui ser reencaminat (gràcies a la canonada) a l'entrada estàndard de la comanda *grep* (que és on aquesta comanda treballa, no pas d'una hipotètica entrada d'error)
NOTA: Una altra manera de fer la mateixa comprobació seria executar aquesta altra comanda alternativa: *openssl s_client -alpn h2 -connect www.lecturas.com:443 -status*

c) Fes la prova explicada al paràgraf blau anterior contra el servidor web del cole ("elpuig.xeill.net") i confirma si pot comunicar-se amb HTTP/2 o no

NOTA: Curl també disposa del paràmetre *--http3* per tal de prova de connectar amb el servidor directament en HTTP/3 (i si el servidor no ho soportés, passar llavors a la versió anterior, HTTP/2). Aquesta opció evita, doncs, l'ús de la capçalera Alt-Svc (per fer el recorregut invers, de HTTP/2 a HTTP/3, que és avui dia l'estàndard)

d) Explica per què mostren el que mostren les següents comandes (la pots provar al terminal de la teva màquina real o al d'un màquina virtual qualsevol si instal.les abans els paquet "curl"):

curl -sIL https://www.twitter.com | grep "HTTP/"

NOTA: ¿Per què creus que aquesta comanda es podria fer servir per saber si un determinat lloc web està caigut?

curl -sIL https://buff.ly/1lTcZSM | grep "^Location"

NOTA: Aquí la clau està en el valor de la capçalera de servidor Location

NOTA: ¿Què veus si proves la comanda *curl -s http://tinyurl.com/api-create.php?url=http://elpuig.xeill.net* ?

curl -s https://www.merriam-webster.com/word-of-the-day | grep -Eo "word-of-the-day/.-$(date +%Y-%m-%d)" | uniq | cut -f2 -d"/" | cut -f1 -d"-"*

NOTA: La comanda anterior és un exemple molt bàsic d'una tècnica més general anomenada "web scraping"

NOTA: Fixa't sobre tot en l'expressió regular utilitzada al *grep*. Pots consultar com funcionen les expressions regulars a http://www.regular-expressions.info i/o provar-les a http://regexr.com o https://regex101.com

curl -s http://artscene.textfiles.com/vt100/wineglas.vt | pv -L4800 -q

NOTA: El contingut dels arxius "vt" (hi ha molts més a http://artscene.textfiles.com/vt100) són seqüències de control del terminal per posicionar el cursor a la pantalla, fer-lo desaparèixer, etc; es pot trobar una llista completa a http://www.termsys.demon.co.uk/vtansi.htm . La comanda *pv* serveix per mostrar el progrés de les dades que rep per la canonada (encara que amb *-q* no mostra res de fet, perquè el que realment es vol és fer servir la funcionalitat del seu paràmetre *-L*, el qual serveix per modificar la ràtio de transferència de dades en bytes/s (per tant, si l'animació va massa ràpida o massa lenta, es pot canviar aquest valor).

e) Prova aquestes URL curioses al terminal i digues què es veu:

curl https://wttr.in/Barcelona
curl -A "Mozilla/5.0 (Windows NT 10.0; Win64; x64)" https://wttr.in/Barcelona

NOTA: En aquest sentit, és molt interessant aquest article: https://csvbase.com/blog/2

curl https://ipinfo.io
curl -A "Mozilla/5.0 (Windows NT 10.0; Win64; x64)" https://ipinfo.io

NOTA: Més URLs que mostren la teva IP pública (i més dades relacionades) en format JSON de forma similar a l'anterior són: http://httpbin.org/ip o http://wtfismyip.com/json Més URLs que mostren (només) la teva IP pública (sense salt de línia al final) són http://whatismyip.akamai.com, http://ifconfig.me , http://l2.io/ip , http://ipecho.net/plain , http://ident.me , http://tnx.nl/ip , http://wgetip.com, http://ip.tyk.nu o http://curlmyip.net Més URLs que mostren (només) la teva IP pública (amb salt de línia al final) són http://icanhazip.com, http://ipaddr.site, http://ifconfig.co, http://ifconfig.pro, http://ipinfo.io/ip, http://checkip.amazonaws.com o http://eth0.me .També pots provar amb *curl* aquests serveis indicant altres direccions IP pública que no siguin la teva per obtenir-ne informació relacionada (geolocalització, operador telefònic, etc). Per exemple: http://ipinfo.io/207.46.13.41 (o http://ipinfo.io/207.46.13.41/loc), http://ifconfig.me/207.46.13.41, http://ifconfig.co/country (o http://ifconfig.co/city , http://ifconfig.co/country-iso , http://ifconfig.co/json)

curl http://artscene.textfiles.com/asciiart/panda

NOTA: La imatge mostrada per la comanda anterior és un exemple concret d'"Ascii Art" (https://www.asciiart.eu)

curl -T gat.jpg https://free.keep.sh

Consulta el manual de *curl* (o la teoria de classe) per saber per a què serveix el seu paràmetre ***-T*** i quin mètode HTTP és el que implementa. Consulta també la documentació del servei "Keep.sh" (https://keep.sh/docs) per saber què ofereix, com funciona i altres exemples d'ús.

NOTA: Una altra URL que serveix pel mateix és http://filepush.co/upload (es faria servir així: *curl -T gat.jpg http://filepush.co/upload/gat.jpg*). Altres similar són https://transfersh.com o https://transfer.sh

NOTA: Una altra URL que serveix, però, només per pujar-hi text és http://termbin.com La gràcia és que en aquest cas el codi font del servidor es troba disponible a https://github.com/solusipse/fiche (això permet poder instal·lar-lo a una màquina pròpia)

f) Fes una petició POST contra la URL http://httpbin.org/post amb les següents dades (estan en format JSON, compte amb indicar el valor correcte a la capçalera de client "Content-Type"): *{"nom":"ana","edat":26}* Comprova que aquestes dades "hagin arribat bé" observant la resposta obtinguda (concretament, el que apareix dins de la secció anomenada "json")

PISTA: Recorda: per realitzar una petició POST (com és el cas) en comptes del tipus per defecte, que és GET, cal escriure al *curl* el paràmetre ***-X POST.*** No obstant, aquest paràmetre ha de venir acompanyat sempre d'un altre on s'especifiqui exactament les dades concretes a enviar al servidor: aquest paràmetre té la forma general *-d "dades"*, així que en aquest cas, doncs, haurem d'afegir això també: ***-d '{"nom":"ana","edat":26}'*** No obstant, el format per defecte de les dades que s'envien en una petició POST estàndard està estipulat que sigui "application/x-www-form-urlencoded" , així que per indicar al servidor que a la petició POST s'envien dades en format JSON cal afegir un altre paràmetre que serveix per informar d'aquest fet, concretament ***-H 'Content-Type:application/json'*** (aquest paràmetre el que fa en realitat simplement és afegir a la petició la capçalera de client indicada, la qual pot ser qualsevol però en aquest cas concret és "Content-Type" , la qual serveix justament per informar del tipus MIME de les dades que acompanya a la petició

NOTA: Tal com hem dit, si no modifiquéssim explícitament el valor per defecte de la capçalera de petició "Content-Type" el format de les dades enviades via POST seria "application/x-www-from-urlencoded". Això implica que el paràmetre *-d* hauria d'indicar-se llavors així: ***-d '?nom=ana&edat=26'*** En qualsevol cas, es faci servir un format o un altre, si les dades estiguessin guardades en un fitxer, llavors hauríem d'utilitzar el paràmetre *-d* així: ***-d @/ruta/fitxer.txt***

4.-a) ¿Què fa la línia *curl* ***-O*** *-C -* *http://www.linuxvoice.com/issues/016/Linux-Voice-Issue-016.pdf*? ¿Quina diferència hi ha amb *curl* ***-o magazine16.pdf*** *-C - http://www.linuxvoice.com/issues/016/Linux-Voice-Issue-016.pdf* ? ¿I amb *curl* ***-o /opt/magazine16.pdf*** *-C - http://www.linuxvoice.com/issues/016/Linux-Voice-Issue-016.pdf ?*

b) OPCIONAL Per descarregar tots els Pdfs que hi ha penjats a la web anterior, gràcies a què les seves URLs segueixen una estructura amb un patró clar i definit, és fàcil realitzar un script que vagi descarregant-los un a un, com el següent. Prova'l i raona el per què de la condició que hi apareix:

```
#/bin/bash
for i in {1..50}
do
    if [[ $i -lt 10 ]] ;  then
        curl -O -C - http://www.linuxvoice.com/issues/00$i/Linux-Voice-Issue-00$i.pdf
    else
        curl -O -C - http://www.linuxvoice.com/issues/0$i/Linux-Voice-Issue-0$i.pdf
    fi
done
```

NOTA: Els enllaços a la web de LinuxVoice també podrien ser de tipus "https" en comptes de "http" però llavors caldria afegir el paràmetre *-k* a la comanda *curl* per a què aquesta els acceptés. Això és degut a què aquesta web no implementa correctament la seguretat proporcionada per "https" (o, en altres paraules, ofereix un certificat "de mala qualitat" perquè és autosignat) i per defecte curl no accepta aquest tipus d'enllaços (a no ser que se l'indiqui el contrari amb el mencionat paràmetre *-k*)

NOTA: En realitat, amb els expansors de Bash (és a dir, els claudàtors i/o les claus, entre d'altres símbols) es poden indicar sèries senzilles per aconseguir descarregar varis fitxers amb una sola comanda *curl*, com així mostren els següents exemples hipotètics (on el paràmetre *-Z* de *curl* serveix, amés, per fer múltiples descàrregues en paral·lel en lloc de seqüencialment). Cal dir que per a què això funcioni, les URLs no han d'escriure's entre cometes (si fos així el Bash no expandiria els símbols):

```
curl -Z -O https://example.com/section[A-Z].html      : Descarrega "sectionA.html", "sectionB.html", etc
curl -Z -O https://example.com/index-[01-10].html     : Descarrega "index01.html", "index02.html", etc
curl -Z -O https://example.com/index-[01-10:2].html   : Descarrega "index01.html", "index03.html", etc
curl -Z -O https://example.com/{one,beta,plus}.html   : Descarrega "one.html", "beta.html" i "plus.html"
```

c) OPCIONAL ¿Què fa la línia *curl -s https://elpais.com | grep -Eo '[^"]*\.(png|jpg)'* ? A partir d'aquí, ¿què faria el següent script?

```
#!/bin/bash
for foto in $(curl -s https://elpais.com | grep -Eo '[^"]*\.(png|jpg)' ) ;  do
        curl -O "$foto"
done
```

NOTA: Per realitzar descàrregues massives de fitxers vía HTTP/S és més còmode, no obstant, utilitzar la comanda *wget*, més especialitzada en aquest tipus de tasques (tal com veurem al proper exercici)

d) **OPCIONAL** Què fa aquest script? Estudia tots els detalls (el que retorna *curl*, els paràmetres del *grep*, la sintaxi *${#nomArray[@]}*, etc)

```
#!/bin/bash
ipPublica=$(curl -s ipecho.net/plain)
#Els parèntesis exteriors són per generar un array amb els valors obtinguts de $(nmap ... | grep ...)
arrayPorts=($(nmap ${ipPublica} | grep -Eo "^[0-9]{1,5}"))
for (( i = 0; i < ${#arrayPorts[@]}; i++ )) ;  do
        echo -e "PORT: ${arrayPorts[$i]}"
done
```

e) I aquest? (fixa't en el paràmetre *-w* de *curl*; consulta el manual si ho necessites)

```
#!/bin/bash
for i in {1..3} ;  do
        curl -s -w "Temps de càrrega: %{time_total}\n" -o /dev/null https://www.google.com
done
```

NOTA: Altres variables interessants que es poden indicar al missatge indicat amb *-w* són, per exemple: ***%{response_code}***, ***%{content_type}, %{num_redirects}, %{url_effective}, %{remote_ip}, %{remote_port}, %{size_download}***, ***%{speed_download}, %{time_connect}, %{time_appconnect}, %{time_namelookup}***,... Per més informació, consultar el manual de *curl* o també https://blog.kenweiner.com/2014/11/http-request-timings-with-curl.html

eII) OPCIONAL I aquest? (on es fa servir també el paràmetre *-w* de *curl*)

```
#!/bin/bash
if [[ "$#" -lt 1 ]] ; then
    echo "Exemple d'ús: $0 https://google.com" ; exit 1
fi
declare -A R=( [000]="Connection Failed"  [200]="Successful" [301]="Moved Permanently"
     [302]="Moved Temporarily"  [400]="Bad Request" [401]="Unauthorized" [403]="Forbidden"
    [404]="Not found"  [500]="Internal Server Error" [502]="Bad Gateway" [503]="Service Unavailable"  )
for ((i=1; i<=3; i++)) ; do
    HTTPCODE=$(curl --connect-timeout 1 -s -o /dev/null -w "%{http_code}" ${1})
    if [[ ${HTTPCODE} -eq 200 ]] ; then
        echo "OK" ; exit
    else
        echo "ERROR:$(date +'%Y-%m-%d %H:%M:%S') SITE:${1} RETURN:${R[$HTTPCODE]} COUNT:${i}"
    fi
    sleep 2
done
```

La comanda de terminal *wget* és un descarregador de fitxers a través dels protocols HTTP/S, FTP/S i més. És una eina més especialitzada que *curl*: mentre que aquest és un client HTTP/S genèric que, entre altres coses, permet descarregar fitxers, *wget* és un client HTTP/S específicament dissenyat per aquesta tasca concreta: descarregar (i, per tant, ofereix més opcions específiques que *curl* no dóna, com ara la possibilitat de fer descàrregues recursives, entre d'altres). Tot seguit es mostra una "xuleta" de les opcions més útils:

wget https://url/dun/fitxer	Descarrega al disc dur el fitxer indicat
-c	Continua la descàrrega (si anteriorment va fallar) des d'allà on es va interrompre
-O nom	Indica el nom que tindrà el fitxer un cop descarregat
-r	Realitza una descàrrega recursiva si la URL indicada és la d'una carpeta en comptes de la d'un fitxer. Combinat amb el paràmetre *-l* *nº* serveix per indicar fins a quin nivell (1=una subcarpeta, 2= dues subcarpetes) es vol descarregar...si no s'indica s'entén "infinit"
-p	Realitza la descàrrega de tots els recursos (CSS, Javascript, imatges...) referenciades per la pàgina descarregada, si la URL indicada és la d'una pàgina individual, per tal de què aquesta es pugui visualitzar adientment. Se sol combinar amb *-H* per tal de permetre connectar a altres servidors més enllà de l'indicat per si cal descarregar-hi de més enllà aquests recursos

-N	Descarrega només els arxius més nous que els locals
-A "ext1","ext2",...	Descarrega només els arxius que trobi amb l'extensió indicada El paràmetre contrari (descarrega tot menys els arxius indicats) és *-R*
-np	No descarrega contingut anterior a la URL indicada. Per a què funcioni correctament aquest paràmetre, però, cal que la URL indicada finalitzi amb el símbol "/"
-nd	Tot ho descarrega a la mateixa carpeta local (sense respectar, doncs, la jerarquia de carpetes del lloc remot)
-U valorUserAgent	Permet indicar el valor desitjat per la capçalera e petició "User-Agent". Aquest paràmetre és útil quan el lloc a accedir té prohibit l'entrada específicament a *wget* (i cal llavors enganyar-lo fent-li creure que som un navegador normal)
-k	Un cop feta la descàrrega, transforma els enllaços per tal de què tot el contingut es pugui visitar offline (canvia les rutes absolutes per relatives i als recursos no descarregats els referencia amb la URL completa). Se sol combinar amb *-E*, el qual afegeix l'extensió adient a cada recurs descarregat (".css" als arxius CSS, ".js" als arxius Javascript, etc) ja que per defecte *wget* això no ho fa i això pot provocar que els links (que sí incorporen l'extensió) no apuntin al nom correcte del recurs.
-e robots=off	Permet saltar-se la restricció de fer cas a l'arxiu "robots.txt". Aquest paràmetre és útil per poder accedir a totes les subcarpetes on es troben els recursos a descarregar, ja que pot ser que existeixi al servidor en qüestió un arxiu "robots.txt" que no ho permeti (consulta https://www.robotstxt.org/robotstxt.html per + informació)

5.- Consulta la llista de paràmetres de la comanda *wget* indicada al paràgraf anterior i digues quina combinació d'aquests utilitzaries per descarregar-te en una mateixa carpeta de la teva màquina real tots (i només) els arxius PDF que penjen recursivament sota la URL http://www.linuxvoice.com/issues Prova-ho