Network Administration and System Administration

Midterm Examination

Time: 2022/04/11 09:10 - 12:10

Instructions and Announcements

- 考試時間共三小時，三人一組考試。分組連結為 https://docs.google.com/spreadsheets/d/1wBNaP24xZRQy4pbS7FG9YUayLDsY7P8xBJxpK9GbkOc。
- 請加入 gather town 進行考試，連結: https://app.gather.town/invite?token=e9CFAVo03tBFu9hMfd-ReMKCJARCkzkx。萬一 gather town 有問題無法使用，NTU COOL 的置頂區有 YouTube 直播連結備用。
- 考試期間禁止使用手機、電話、任何通訊軟體等與同組成員外任何人聯繫，也禁止組與組之間**一切討論與合作**，如被發現視為作弊行為，**期中考 0 分**，並依校規懲處。
- 作答過程中請自行斟酌備份，避免電腦發生意外，損失過多進度，可考慮準備隨身碟或雲端空間備份進度。
- 為避免發生重大意外，請自行注意 VM 用量，同時開啟過多 VM 可能導致電腦當機，我們恕不負責。
- 線上考試的 announcement 將會更新在 https://docs.google.com/document/d/1CxGxNtxwPPYwMnenWE4c4zoGobpiGJTlDGUIvy3H21c。
- 題目檔案請至 https://drive.google.com/drive/folders/1fNNF8Qd-OB5bJLkLLr155d1s9pkz1plv?usp=sharing (考試開始後公開) 或 http://linux5.csie.ntu.edu.tw:18080/ 下載。zip 檔的解壓縮密碼將會在考試開始後公佈。
- 完成題目時請至 https://forms.gle/UxT8Y3ec5tXkGmvf9 上傳作答內容，每組每個 subtask **最多上傳 3 次**。
- 計分板連結 https://docs.google.com/spreadsheets/d/1VYhIqfr22s2-CnQL91Elw4SWPbfVWLIPYF8CJxUyD1w。
- 各題後面黑色星號數目代表我們估計的難度。請參考，可用來決定解題順序。
- 題目可能難免有疏誤之處。若發現有解不開題目、題敘不清的狀況，請盡快跟助教或老師反應，或斟酌時間先解別的題目。
- 第 4 題 Shell Script CGI **限制**使用 shell script，第 5 題 Local Judger 與第 6 題 tshark **不限制**所使用的語言。請確保 script 能在助教要求的環境內執行（如 docker 或工作站）。
- 滿分為 220 pts。

1 Wireshark ★★☆☆☆ ~ ★★★★☆ (25 points)

File

- Problem 1.1 ~ 1.2: prob-wireshark-1.pcapng

1.1 HTTP Redirection ★★☆☆☆ (4 points)

請在 prob-wireshark-1.pcapng 中找出經由 HTTP redirect 後的 URL 網址。

1.2 Playing with Traceroute ★★★★☆ (8 points)

請在 prob-wireshark-1.pcapng 內找出 `traceroute` 的 destination （用 FQDN 表示），並回答當時 `traceroute` 的使用者，他發出封包的 TTL 至少要多少才能送達該 destination 。

1.3 Capture the Flag ★★★★☆ (13 points)

有個神秘人將 flag 藏在 nasa-midterm 中，請你試著找到這個 flag (NASA{...} 形式)。

Hint

- 1.2: 可以試著自己側錄看看使用 `traceroute` 時的封包並觀察它
- 1.3: 試著透過 Wireshark 過濾不含圖片的封包，若圖片在動卻沒收到對應的封包，可以試著重整，建議使用 google chrome / firefox 瀏覽器

Submission

- 1.1: 請透過 Google Forms 上傳經由 HTTP redirect 「後」的 URL 網址。
- 1.2: 請透過 Google Forms 上傳 `traceroute` 的 destination （用 FQDN 表示），以及至少需多少 TTL 。需同時繳交兩個答案。兩者全對才給分。
- 1.3: 請透過 Google Forms 上傳 flag。

---

2 Cisco ★★☆☆☆ (20 points)

Resources

- 使用檔案 prob-cisco.pkt
- Packet Tracer account: cisco.packet.tracer@yopmail.com
- Packet Tracer password: Cisco.packet.tracer0
- 沒有 Activity Check，需自行檢查設定結果

Tasks

Note: 粗體（如 **Core**）的是機器的名稱，打字機字型（如：`nasa.csie.org`）的為要設定的 domain name, 參數等等。

1. (4 points) 基本設定
   - 將 **Core** 的 hostname 設為 `Core`。
   - 將 **A-01** 的 hostname 設為 `A-01`。
   - 將 **A-02** 的 hostname 設為 `A-02`。
   - 將 **Core**、**A-01**、**A-02** 的 enable 密碼都設置為 `admin`，並且不能以明文顯示在 `running-config` 中。
   - 將 **Core**、**A-01**、**A-02** 的 domain name 都設置為 `nasa.csie.org` （指不包含 hostname 的部分）。
2. (8 points) VLAN 設定
   - **PC50-1**、**PC50-2** 設置在 VLAN `50`。
   - **PC70-1**、**PC70-2** 設置在 VLAN `70`。
   - **Admin** 設置在 VLAN `99`。
   - **A-01** 與 **Core** 之間設置 trunk link，且上面可通行 VLAN `50`、`70`、`99`。
   - **A-01** 與 **A-02** 之間設置 trunk link，此一 trunk link 需使用 link aggregation 設置，上面可通行 VLAN `50`、`70`、`99`。
   - 將 **Core** 的 interface VLAN 99 的 IP 地址設置為 `192.168.99.1/24`。
   - 將 **A-01** 的 interface VLAN 99 的 IP 地址設置為 `192.168.99.2/24`。
   - 將 **A-02** 的 interface VLAN 99 的 IP 地址設置為 `192.168.99.3/24`。
   - 相同 VLAN 可以互相傳輸訊息，不同 VLAN 之間無法互相傳輸訊息。(例如 **Admin** 可以 ping 到 **Core**、**A-01**、**A-02**，並且 ping 不到其他 PC)
3. (8 points) VTY 設定
   - 以下皆由 "**三台 switch**" 代稱 **Core**、**A-01** 與 **A-02**。
   - 設定**三台 switch** 使用 SSH version 2。
   - 設定**三台 switch** 的 SSH timeout 為 `90` 秒。
   - 在**三台 switch** 上新增帳號密碼 username: `admin`/ password: `admin` 以用做 SSH 登入的帳號密碼，並且密碼不能以明文顯示在 `running-config` 中。
   - 將**三台 switch** 的 vty 0 到 4 設置為能使用 SSH 登入。(telnet 沒有限制)
   - 設定完成後，能在 **Admin** 上使用指令 `ssh -l admin <switch vlan 99 ip>` 連線至**三台 switch**。(例如指令 `ssh -l admin 192.168.99.2` 能連線至 **A-01**)

Submission

- 請上傳完成後的.pkt 檔
- 若一次完成多個小題，只需在最後完成的小題欄位上傳一份檔案就好
- 例如：上傳在第 3 小題的欄位，1-3 小題都會一起檢查

3 pfSense ★★☆☆☆ ~ ★★★★☆ (30 points)

Resources

- pfsense-ubuntu.ova (OS for VM in VLAN 99) (account/password: ubuntu/2022midterm)
- alpine-virt-3.15.4-x86_64.iso
- pfSense-CE-2.5.2-RELEASE-amd64.iso

Note: ova 檔測試過可在 VirtualBox、VMware 上正常運作。
Hint: 若要在 Windows 電腦做此題，請注意本機的防火牆設定。

Description

在這個大題，你需要建立一台 pfSense 虛擬機與三台內網的虛擬機

- 請在你的電腦開一台 pfSence 虛擬機，安裝 pfSense-CE-2.5.2-RELEASE。這台虛擬機應該至少要有一個對外網路介面，三個對內網路介面。三個對內的網路介面分別對應到三個 VLANs - VLAN 5、VLAN 8 和 VLAN 99。
- 在你的電腦上安裝兩台 Alpine 虛擬機，並分別將它們的網路連接到前述 pfSense 虛擬機的 VLAN 5 和 VLAN 8 兩個對內網路介面下。以 pfsense-ubuntu.ova 建立的虛擬機則要將它的網路連接到 pfSense 虛擬機的 VLAN 99 對內網路介面下。
- 上述的兩台 Alpine 虛擬機及 Ubuntu 虛擬機皆僅需要一張網卡，在 pfSense 上需開啟 DHCP Server 服務，使它們可以自動取得相應 VLAN 的 IP 位址。

Note: 由於以下題目要求須透過 VLAN 99 下的機器連上 pfsense 的 Web GUI，而安裝有圖形化介面的 VM 所需的時間較長，因此若選擇使用助教所提供的 ova 檔案匯入，可省下許多安裝的時間，若是想要使用其他的方式架設虛擬機也是被允許。

Tasks

1. (3 points) 請在 VLAN 8 下那台 Alpine 上執行 `ip a; ip r` 指令驗證設定是否正確。請注意，除了 loopback 網路介面之外，Alpine 只有對應到 VLAN 8 的那個網路介面能擁有 IP 位址。
2. (5 points) 請設定 pfSense 的 DNS Resolver 服務，讓 `cloudflare.dns` 這個 domain name 對應到 `1.1.1.1` 和 `1.0.0.1` 這兩個 IP 位址。（可在 VLAN 5 下的機器上執行 `nslookup cloudflare.dns` 驗證設定是否正確）
3. (6 points) VLAN 99 下的機器可以連到 pfSense 的 web GUI 和 ssh，而在 VLAN 5, VLAN 8 下的機器則無法連上。
4. (4 points) VLAN 5, VLAN 8 下的機器到 VLAN 99 下的機器的所有連線是不被允許的，但 VLAN 99 下的機器到 VLAN 5, VLAN 8 下的機器的所有連線是被允許的。
5. (2 points) 你想對 pfSense 做一些你不確定的操作，先把 pfsense 目前的設定備份起來再說。請說明你如何用 pfSense 的 Web GUI 頁面做到此設定。
6. (4 points) VLAN 8 下的機器只能連到 linux[1 - 3].csie.ntu.edu.tw 和 DNS server。
7. (6 points) 請在兩台 Alpine 虛擬機上安裝並設定 OpenSSH server，並對 pfSense 虛擬機做必要的設定，讓兩台 Alpine 的虛擬機可以用 SSH 連線到彼此。

Submission

請向助教分享螢幕以 demo ，可一次 demo 多題。每題 demo 請勿佔用超過一分鐘，以免影響別組權益。請考慮盡早開始本題，以免最後 demo 塞車。

Hint

- 正確地使用 `setup-alpine` 可以避免資料在 Alpine Linux 重開機時遺失。
- Lab 投影片會是你的好朋友。
- pfSense 官方文件 會是你的好朋友。
- pfSense 的 Aliases 會是你的好朋友。

---

4 Shell Script CGI ★★☆☆☆ ~ ★★★☆☆ + 0.5★ (35 points)

Resource

- 你可以用底下任意一種方法得到執行 Script 用的 Docker 環境：
  - `docker pull brianbbsu/cgi-grading-env`。
  - `docker load -i cgi-grading-env.tgz`
  - `docker build -t cgi-grading-env - < cgi-grading-env.Dockerfile`

  其中，若你使用的是 Apple M1（arm64）的設備，建議你直接使用第三種方法 build image。

Description

你知道用 Shell Script 也可以寫網頁後端嗎！下方是一個用 Bash 寫的簡單範例：

```
#!/bin/bash
printf "Content-type: text/plain\n\nMeow!"
```

你可以將其存於工作站家目錄下的 `~/htdocs/cgi-bin/test`，將其設為可執行，並打開瀏覽器連到 `https://www.csie.ntu.edu.tw/~<username>/cgi-bin/test`，就會看到一個可愛的 `Meow!` 了！

透過「通用閘道器介面（Common Gateway Interface, CGI）[1]」，任何能夠輸入、輸出、以及讀取環境變數的程式語言基本上都能用來撰寫網頁後端。在這個大題中，你要使用 Shell Script 來撰寫一些簡單的 CGI 網頁應用。

[1] https://en.wikipedia.org/wiki/Common_Gateway_Interface

Environment

我們會在上述提到的 Docker 環境中透過 CGI 來執行你的 Script。該環境已經安裝好以下的幾種 Shell：`sh`、`bash`、`zsh`、`fish`、`ksh`、`tcsh`。你的 Script 會被放在 `/usr/local/apache2/cgi-bin/` 裡，並且**該位置的上一層（i.e., `/usr/local/apache2/`）會是可寫的**。

如果要在自己電腦上測試，你可以執行下方的指令。所有在目前資料夾裡面的檔案皆會在 `http://localhost:8080/cgi-bin/` 底下。

```
docker run --rm -p 8080:80 -v $(pwd):/usr/local/apache2/cgi-bin -d cgi-grading-env
```

註：你也可以參考上面的 Example，把 Script 放在工作站上測試。但請注意，放在工作站的 Script 實際上是在其他機器上執行的，**該機器上缺少許多 Shell 跟指令**，所以可能會有無法正常執行的狀況。另外，為避免可能造成的資安漏洞，**請記得在結束後將不必留存的 Script 刪除或移出，若未執行本步驟，將審酌情況扣期中考之總分**。

Rules

- **此題限用 Shell Script。原則上 Docker 環境裡有的所有工具跟指令都可以使用**，但請不要嘗試使用網路連線或是安裝其他套件。
- 所有 Request 中，我們能控制的地方（像是 Request Path 跟 `username`）可能會出現的字元集合是 `[a-zA-Z0-9/-_&=?.:]`
- Response Body 請盡量遵守題目說明中的格式（每一行的行尾都要有換行），不過原則上些微的差異不會影響給分與否。

Hint

- Script 開頭記得要加 *shebang*
- Request 的各種資訊會在環境變數裡。另外如果是 POST Request 的話，Request Body 會在標準輸入（stdin）。
- `env` 這個指令可以印出所有環境變數，裡面也包含各種需要的輸入，請仔細檢查看看。
- `jq` 這個工具可以幫助你處理 json，或許會很有用。使用 json 也會是個蠻建議的解法，可以試試看！
- `exec 2>&1` 把這個放在 Script 的前面可以讓你在 Response 裡面看到各種錯誤時的訊息。

Task 4.1 - Hello, CGI! ★★☆☆☆ (7 points)

先來暖身一下吧！請在 Response 的第一行輸出 `Hello, CGI!`，並在第二行輸出該 Request 的 Path。舉例來說，當我們對 `<url to your script>/123/abc` 發送 GET Request 時，你的 Response 要長的像下面這樣：

```
Hello, CGI!
Path: /123/abc
```

請注意，如果是對 `<url to your script>` 發送 Request，你的 Path 應該要是一個空字串（冒號後面還是要有恰好一個空格）。

Task 4.2 - Page View Count ★★☆☆☆ (8 points)

接下來請實作一個瀏覽次數統計吧！Response 應該要是一個數字，代表該 Path 被瀏覽了幾次。另外，如果該次 Request 的 Query String **恰好（exactly）**是 `reset`，則請將該 Path 的瀏覽次數歸零重新計算。底下是一個可能的互動過程：

```
# curl "<url to your script>/path1"
1
# curl "<url to your script>/path1"
2
# curl "<url to your script>/path2"
1
# curl "<url to your script>/path1?reset"
1
# curl "<url to your script>/path1"
2
# curl "<url to your script>/path2?a"
2
# curl "<url to your script>/path2/"
1
```

Task 4.3 - URL Shortener ★★★☆☆ + 0.5★ (10 + 10 points)

最後，我們來寫個短網址工具吧！此題有兩個小題，以下是兩個小題共用的規則：

- 當 Response 的 Status Code 不是 200 時，Response Body 的內容可以隨意。
- 如果 Request 的 Path 不存在，或是 Request Method 不對（例如該 POST 的 endpoint 收到 GET），請一律傳送 Response Status Code 404。
- POST Request 的 JSON 都會是合法的格式，而且不會缺少欄位。JSON 的每個欄位都是對應到一個字串。
- 請使用 `Status` 以及 `Location` 這兩個 Response Header 來做到指定 Response Status Code 以及 Redirect 的行為。

另外你可以在題目說明之後看到範例互動以及使用 `curl` 發送 POST Request 的方式。

Task 4.3a (10 points)

請實作以下兩個 endpoint：

- `POST /signup`：Request Body 會是一個 JSON，其中 `username` 跟 `password` 兩個欄位會是新帳號的帳號密碼。
  1. 如果帳號或密碼為空（長度為 0），或是帳號已經存在，請傳送 Response Status Code 400。
  2. 否則，請傳送 Response Status Code 200，並且 Response Body 是以下**兩行**：

     ```
     200 OK
     Signup successful
     ```

- `POST /verify`：Request Body 會是一個 JSON，其中 `username` 跟 `password` 會是要驗證的帳號密碼。
  1. 如果該組帳號密碼組合並不正確，請傳送 Response Status Code 401。
  2. 否則，請傳送 Response Status Code 200，並且 Response Body 是以下**兩行**：

     ```
     200 OK
     Verify successful
     ```

Task 4.3b (10 points)

請接續 4.3a 另外實作以下三個 endpoint：

- `POST /create`：Request Body 會是一個 JSON，包含有四個欄位 `username`、`password`、`destination`、跟 `short`。`short` 不會包含 `/`
  1. 如果帳號密碼的組合並不正確，請傳送 Response Status Code 401。
  2. 否則，如果 `short` 為空，或是 `short` 跟其中一個 endpoint 的名稱雷同（`signup`、`verify`、`create`），或是已經有相同名稱的其他短網址，請傳送 Response Status Code 400。
  3. 否則，請建立一個從 `<url to your script>/<short>` 轉到 `destination` 的短網址，Response Status Code 200，並且 Response Body 是以下**兩行**：

     ```
     200 OK
     Create successful
     ```

- `GET /[^/]+`：如果這是一個存在的短網址，則將使用者以 Status 302 Redirect 到對應的 `destination`，否則傳送 Response Status Code 404。
- `GET /[^/]+/info`：如果前半部分是一個存在的短網址，則 Response 格式如下**兩行**的內容，否則傳送 Response Status Code 404：

  ```
  200 OK
  meow is created by brian, redirects to http://example.com
  ```

  其中 `meow`、`brian`、與 `http://example.com` 請填入相對應的 `short`、建立者的 `username`、以及 `destination`。

Task 4.3 Example

底下是一個可能的互動過程：

```
# curl "<url to your script>/signup" -X POST \
    -d '{"username": "bb", "password": "pwd123"}'
200 OK
Signup successful
# curl "<url to your script>/signup" -X POST \
    -d '{"username": "bb", "password": "pwd123"}'
(Response HTTP Status 400)
# curl "<url to your script>/verify" -X POST \
```

```
    -d '{"username": "bb", "password": "pwd123"}'
200 OK
Verify successful
# curl "<url to your script>/verify" -X POST \
    -d '{"username": "bb", "password": "pwd1234"}'
(Response HTTP Status 401)
# curl "<url to your script>/create" -X POST \
    -d '{"username": "bb", "password": "pwd123",
         "destination": "http://example.com", "short": "a"}'
200 OK
Create successful
# curl "<url to your script>/a/info"
200 OK
a is created by bb, redirects to http://example.com
# curl "<url to your script>/a"
(Response 302 Redirection to http://example.com)
# curl "<url to your script>/b"
(Response HTTP Status 404)
```

Submission

此題限用 Shell Script。請直接上傳完整的 Script。

---

5 Local Judger ★☆☆☆☆ ~ ★★★☆☆ (25 points)

Resource

- Nasa-Midterm-Judger.zip

Description

在 Lab2 中，我們實作了簡易的 JudgeGirl Judger，但身為 NTU CSIE 的學生，通過計程之後早已不會再接觸到 JudgeGirl，等待著大家的可是 DSA 和 ADA 的考驗。這意味著我們需要支援更多功能的 Local Judger！

請寫 script (不限定 shell script) 來實作一個 local judger，在輸入.c 檔後使用 gcc 編譯，並去特定的資料夾將內部的測試資料爬出來測試。執行時除去 option 外，會有一個.c 檔作為唯一的一個參數。也就是說，如果你的 script 叫做 judge ，那麼它會以如下的方式被執行：

```
./judge [option] code.c
```

你的 script 必須給出每筆測試資料評測後的結果，並依據指定的格式輸出。根據任務要求，你可能還得支援一些特定的 options。

請跟著子任務給予的指示依序將完整的 script 實作出來。

Rules

- 使用一般的 gcc 編譯便可以完成所有任務，並不需要加任何特別的編譯參數。
- 你只需要支援 Accepted、Wrong Answer 和 Time Limit Exceeded 三種評測結果（與 Lab2 相比，少了 Compile Error）。
- 所有 json 檔、option 都會是合乎格式的。
- 所有與「名字」有關的字串皆是僅包含大小寫英文字母（[a-zA-Z]）、數字（[0-9]）以及 dash (-) 的 ASCII 字串，並滿足長度小於 20、不會以 dash 開頭。
- 為了使期中考評測流程順利，執行時間過長的 Submission 會被直接中止並給予 Time Limit Exceeded 的回饋。適當的暴力實作方式一般並不會被視為超時（無法給出嚴謹的定義還請見諒）。
- 當你的 script 執行完畢後，不可以有任何的檔案被移除，也不可以有任何額外的檔案被產生後還留存著。
- 請嚴格依據指定的格式輸出，範例測試資料可以幫助你在上傳前先做好簡易的比對。
- 評分時，你的 script 會被 ta 這個 group 底下的某個使用者放在 linux* 工作站上後執行，只要有辦法讓 ta 這個 group 底下的使用者正常的執行你的 script 即可。**任意惡意的行為將會被視為作弊並給予相對應的懲處。**

Task 1 - Local Testcases ★☆☆☆☆ (6 points)

在這個子任務中，你得先將 Lab2 中我們爬 JudgeGirl 測試資料下來評測的方式改為從資料夾抓測試資料出來評測。具體來說：

- 所有的測試資料都會被存在 testcases 這個資料夾底下。
- 測試資料沒有編號，每筆測試資料都會有他獨有的名字。而若一筆測試資料的名字為 name，則下列兩個檔案會成對地出現在 testcases 底下：

  name.in, name.ans

  其中 name.in 代表著輸入、name.ans 代表著正確答案。
- Wrong Answer 的判斷基準為「存在一個或以上位置的字元不相符」。
- Time Limit Exceeded 的判斷基準為「程式執行超過一秒」。
- 你的 script 會被放在與 testcases 同樣的目錄下執行。
- 輸出結果時，請遵循以下格式輸出在 stdout（與 Lab2 相同）：

```
------ JudgeGuest ------
Data DIR: <directory name>
Test on: <cpp file name>
-----------------------
<testcase 1><verdict>
<testcase 2><verdict>
...
```

  其中：

- `<directory name>` 在這個子任務中固定為 `testcases`。
- `<cpp file name>` 是輸入的 `.c` 檔的檔名。
- `<verdict>` 根據結果，可能是 `Accepted`、`Wrong Answer` 和 `Time Limit Exceeded` 的其中一種。
- 測試資料的輸出順序請以測試資料名字的 dictionary-order 排序。
- **與 Lab2 不同**，輸出測試資料名字時，也就是上述格式提到的 `<testcase i>`，請補空格在後方至長度達到 20 為止。注意到上述格式中並沒有在 `testcase` 和 `verdict` 中間多加空格的原因是因為我們保證測資的名字小於 20，所以在補空格到長度 20 後會自然有間隔出現。

Task 2 - Options ★☆☆☆☆ (4 points)

繼承前一個子任務，你的 script 會需要額外支援以下三種 options：

- `-d <directory name>`
  代表測試資料存放的位置在 `<directory name>` 底下，請將原先預設的 `testcases` 更改為該資料夾，並確保輸出 `Data DIR: <directory name>` 時後方的資料夾名稱是指定的資料夾名稱。
- `-c <executable file name>`
  原先 `Wrong Answer` 的判斷基準為「存在一個或以上位置的字元不相符」。這個 option 代表我們將使用一個額外的可執行檔來檢查正確性。假設該可執行檔的名字為 `checker`，則使用

  ```
  ./checker <input file> <answer file> <output file>
  ```

  來執行後便可根據該程式的回傳值得知結果，若回傳 0 代表 `Accepted`、回傳一個非 0 的結果代表 `Wrong Answer`。舉例來說，如果現在打算評測的測資名字為 `name`，代表輸入為 `name.in`、正確的答案為 `name.ans`，假設你把 c code 的輸出存在 `name.out` 的話，那麼執行

  ```
  ./checker name.in name.ans name.out
  ```

  即可根據回傳值判定是否通過。當執行檔收到的參數不符合格式時為未定義行為，請正確的使用來避免未知的狀況發生。

  為了使同學實作方便，5.2 保證 checker 的路徑只要加 `./` 在前面就可以執行

- `-t <time limit>`
  `<time limit>` 是一個以秒為單位的浮點數，代表該次測試的時限被更改為 `<time limit>` 秒。

注意到 options 可能會以任意順序出現，但全部都會在輸入的.c 檔之前。

Task 3 - Subtasks ★★☆☆☆ (7 points)

繼承前一個子任務，你的 script 會需要如 DSA 或 ADA 上機題目一般支援子任務系統。由於要在一道程式題目拿到滿分的難度可能略高，所以有不少課程會在讓學生可以退一步先解決「子任務」，就如同本題一般，提供一步一步走到滿分的線索或過程，如此一來也可以讓學生先行拿到部份的分數作為保障。對於一個「子任務」，每個子任務應會有它的分數和它對應到的「測試資料們」，由於評測端在學生上傳程式後正常無法預測學生希望會評測哪些子任務，所以評測端通常會直接跑過所有測試資料，並對於每個子任務，去檢查該子任務要求要通過的「測試資料們」有無全數通過，有的話才會給予這個子任務的分數。而這個 Task 就是希望你實作出一個這樣子的系統。

注意到每個子任務要求的「測試資料們」可能重疊，也因此我們會規定一些格式欄讓每個子任務說明自己希望要通過的測試資料。

具體來說，你的 script 必須額外支援以下 option：

```
-s <JSON file name>
```

假設附帶的參數名稱為 `subtasks.json`，`subtasks.json` 將會是一個合乎格式的 JSON 檔案，並由以下簡易的結構呈現：

```
{
  "<subtask name 1>": {
    "score": <number>,
    "testcases": <regex>
  },
  "<subtask name 2>": {
    "score": <number>,
    "testcases": <regex>
  },
  ...
}
```

其中：

- 保證所有的 subtask 名字不重複。
- `<number>` 是一個介於 0 ~ 100 的數字，所有 subtasks 的 number 總和恰為 100。
- `<regex>` 是一個用來匹配測試資料名稱的 regular expression，以 BRE 格式呈現，且為求方便，僅包含字元 `[a-zA-Z0-9-?.+*^$|]`（不含中括號）。換句話說，這個子任務會去確認「名字匹配到該 regular expression 的測試資料必須全數通過」，這樣這個子任務才會被視為通過。
- Subtasks 的數量不超過 5 筆。

接著，在輸出完先前子任務要求輸出的結果後，請額外輸出以下資訊：

```
<empty line>
Subtask <subtask name 1><verdict>
Subtask <subtask name 2><verdict>
...
Total Score: <score>
```

其中：

- subtasks 們輸出的順序請以 subtask 名字的 dictionary-order 排序。
- 輸出 subtasks 名字時，也就是上述格式提到的 `<subtask name i>`，請補空格在後方至長度達到 20 為止。注意到上述格式中並沒有在 `subtask name` 和 `verdict` 中間多加空格的原因是因為我們保證子任務的名字小於 20，所以在補空格到長度 20 後會自然有間隔出現。
- `<verdict>` 根據結果，可能是 `Passed` 和 `Failed` 的其中一種。出現 `Passed` 的條件若且唯若該筆 subtask 提到的 `testcases` 全數獲得 `Accepted`。
- `<score>` 是獲得 `Passed` 的 subtask 的 `score` 總和。

Task 4 - Dependency ★★★☆☆ (8 points)

繼承前一個子任務，在 `subtasks.json` 內，每個子任務可能會多出一項參數 `include`，代表著該子任務通過的標準除了通過 `testcases` 提到的測試資料外，還得確保 `include` 提到的所有子任務皆通過後才算真正通過。其中 `include` 的格式如下：

```
include: ["<subtask name 1>", "<subtask name 2>", ...]
```

其中：

- 保證所有的 subtask name 存在且不重複。
- 不會存在「環」，意即不會存在一個 subtask 的序列 $s_1, s_2, \ldots s_k$，使得 s_1 `include` s_2、s_2 `include` s_3、……、s_{k-1} `include` s_k 且 s_k `include` s_1。

Hint

- Script 開頭記得要加 *shebang*。
- 你可以以 Lab2 寫的 shell script 當作基礎開始著手實作。當然也可以自己用別的語言重新寫一個出來。
- 如果你對格式有些疑問，完全可以參考範例測試資料，它們可以有效的幫助你理解。
- 可以使用 jq[2] 這個工具來處理輸入的 JSON 檔。

Submission

請透過 Google Forms 上傳可執行的 script（請一定要包含 shebang），並勾選你要上傳的 Task。

注意到本題的每一個 Task 都包含前面的所有 Task，所以你必須勾選你覺得「你可以拿到的最高的 Task」，當該次上傳被視為通過之後，較前的 Task 們也會一同被視為通過；反之未通過則會視為什麼都沒通過，所以請不要總是選擇 Task 4 上傳。

---

6 tshark ★☆☆☆☆ (20 points)

Resource

midterm_p6.pcapng

Description

在這個大題，你會拿到一個封包檔，你需要寫幾個簡單 script(不限定 shell script) 以達到題目要求 (不限 output format，確保可讀性即可)

[2] https://stedolan.github.io/jq/

Tasks

1. (4 points) 請統計 pcapng 中 UDP、TCP 分別的封包數 (frames) 以及資料量 (bytes)。
2. (4 points) 請以每五秒為一個區間，統計並列出 pcapng 從開始到結束中各個區間的封包數 (frames) 及資料量 (bytes)。
3. (4 points) 請統計每個 IP 分別出現的次數 (src, dst 一起計算)，例如: 一個 54.250.40.42→10.5.5.207 的封包，會分別計算為 54.250.40.42 的一次，以及 10.5.5.207 的一次，而一個 10.5.5.207→54.250.40.42 的封包，也會計算為 10.5.5.207 的一次，以及 54.250.40.42 的一次。並把出現次數最多的五個 IP 依出現次數降序列出 (出現次數也要列出)。
4. (8 points) 已知這段時間有 server 在向 client 不安全地傳輸一個 mp4 檔，請根據第三題的結果判斷可能的 client、server，並嘗試用 script 重建檔案。

Submission

- Problem 1: 請透過 Google Forms 上傳可執行的 script(請包含 shebang) 及執行結果。
- Problem 2: 請透過 Google Forms 上傳可執行的 script(請包含 shebang) 及執行結果。
- Problem 3: 請透過 Google Forms 上傳可執行的 script(請包含 shebang) 及執行結果。
- Problem 4: 請透過 Google Forms 上傳可執行的 script(請包含 shebang) 及 mp4 播放截圖。

---

7 Partition ★★☆☆☆ (20 points)

Resource

- Nasa-Midterm-Partition.ova
  - 帳號: nasa2022
  - 密碼: nasa2022

Tasks

1. (3 points) Paul Huang 不小心將祕密遺漏在 secret-lv 這個 logical volume 之中，請試著找出遺留在 secret-lv 中的祕密檔案 (檔案名稱為 flag)。
2. (7 points) 找到祕密後（如果沒找到祕密也可以做，但可以 snapshot 一下以方便你等等回去尋找祕密），secret-lv 就已經沒用了，請你依照以下配置重新切割硬碟：

   p.s. 配置中的 XXXXXX 請自行代換為 Team ID，比如說你們的 Team ID 是 5，就將 XXXXXX 代換為 000005

   - 移除 /dev/sdb2 後，將 /dev/sdb1 擴大至（幾乎）2G
   - 用合併完成的 /dev/sdb1 建立 volume group 並命名為 nasa-XXXXXX-vg
   - 使用 nasa-XXXXXX-vg 建立一個 1.5G 的 logical volume，並命名為 nasa-XXXXXX-lv
   - 將 nasa-XXXXXX-lv 掛載在 /mnt/nasa2022 上

完成後截圖 `"lsblk /dev/sdb; df -hT /mnt/nasa2022"` 的執行結果

3. (5 points) 現在希望你能利用 /dev/sd[c-e] 建立 RAID 5
   - 利用 /dev/sdc, /dev/sdd, /dev/sde 建立一個大小為 2G 的 RAID 5 磁碟分區 /dev/md0
   - 掛載在 /mnt/nasa2022-raid5

   完成後截圖 `"lsblk /dev/sd[c-e]; df -hT /mnt/nasa2022-raid5"` 的執行結果

4. (5 points) 在 VM 裡存在一個以 docker 形式運行的測試用 nfs4 server，你可以透過 `"docker exec -it [CONTAINER_ID] bash"` 來存取。請完成下以配置：
   - 在 nfs4 server 的 container 中創建資料夾 /share/nasa2022-nfs4
   - 將 nfs4 server container 中的 /share/nasa2022-nfs4 掛載至本機(原本的 VM) 的 /mnt/nasa2022-nfs4 上

   完成後截圖在本機上 `"df -hT /mnt/nasa2022-nfs4"` 的執行結果

Submission

- Task 1: 請透過 Google Forms 回答檔案 (flag) 內容
- Task 2: 請透過 Google Forms 上傳截圖
- Task 3: 請透過 Google Forms 上傳截圖
- Task 4: 請透過 Google Forms 上傳截圖

---

8 Docker Doko? ★★☆☆☆ ~ ★★★★★ (25 points)

Resource

- linux[1-15] 工作站上的 /tmp2/ototot/Nasa22-Midterm-Docker-Doko.qcow2
- Google Drive 上的 Nasa22-Midterm-Docker-Doko.qcow2

Description

在這個大題中，你需要在提供的 QEMU 虛擬機環境中操作所有的指令，你可以在上面列出的幾個地方獲得我們準備的 qcow2 檔案。在提供的虛擬機內有兩個提供給你的帳號 lanzhu 與 mia，兩隻帳號的密碼皆為 dodoko，且兩者都在 docker 這個 group 裡面，而環境中的 docker 為一般的 docker 配置，並不是 rootless docker。為避免意外的操作致執行環境毀損，可以參考以下指令複製虛擬機器後再於複製後的機器上操作。

1. `qemu-img create -f qcow2 -F qcow2 -b [SRC].qcow2 [DEST].qcow2`
2. `qemu-system-x86_64 -enable-kvm -smp 1 -m 2G -hda [DEST].qcow2 -cpu host -display curses -vga vmware`

Tasks

1. ★★☆☆☆ (7 points) Rina 提供了一個按照 正常流程 裝好 docker 的機器給 LanZhu，但是 Rina 給 LanZhu 的時候不小心把 `/usr/bin/docker` 刪掉了，因此 LanZhu 現在沒辦法直接透過 `docker` 指令操作 docker，而且他也沒有 root 權限能重新安裝 docker，想要找有權限能寫的資料夾讓她能重新把該檔案載回來也找不到，因此她希望你能幫助她在這樣的環境下還是能操作 docker。

   LanZhu 希望你能幫她建立起一個 alpine 的 Image 並在上面執行 `echo hi Rina`，更具體一點來說也就是希望能獲得與執行 `docker run apline echo hi Rina` 等效的結果，並透過 Google Form 上傳能證明成功執行的截圖（例如包含幾個指令與其輸出的截圖）。

   ※ 這題請使用 lanzhu 的帳號登入操作

2. ★★★☆☆ (3 points) LanZhu 從 Rina 那裡打聽到原來在這台機器上有 Rina 以前建立的 docker Image，並且 Rina 當時曾不小心在該 Image 中建立了不應該出現的檔案，因此 Rina 後來又重新建立了一個，但因為 Rina 覺得硬碟很便宜，所以沒有把當時的 Image 完整移除，請問你可以幫 LanZhu 找出那個不該出現的檔案內容嗎？

   ※ 這題請使用 lanzhu 的帳號登入操作

3. ★★★★☆ (7 points) Rina 覺得如果把東西跑在 docker 裡面的話會很安全，因此她建立了一個 container 並修改掉自己的 shell，讓她登入後就會自動進入該 container 裡面，不過她發現這樣可能會沒辦法讓她在她所在的這個 container 裡面再建立新的 container，因此她使用 docker 官方提供的 dind image 建立了一個 dind container 作為登入後進入的 container，讓她還是能保有執行 docker 指令的權限。

   註：該 container 以指令 `docker run --privileged -d --name rina-base docker:dind` 產生

   建立好 container 後 Rina 在該 container 裡面又跑了幾個 container，但是她在把機器交給 LanZhu 的時候忘記把那些裡面的 container 關閉了，因此佔用了許多系統資源。你可以代替 LanZhu 幫忙把 Rina 那些 container 裡面的 container 關閉嗎?

   ※ 這題請使用 lanzhu 的帳號登入操作

4. ★★★★★ (8 points) Mia 參考 Rina 的設定後，也用同樣的方式建立起了一個 dind container 並把自己的 shell 修改掉，讓自己在登入後會直接進到該 container 裡面，但是她發現這樣會導致她不知道如何再重新碰到外面的資料，例如 Mia 就不知道怎麼看到 Yuu 在自己的家目錄底下建立的檔案 `/home/yuu/secret`，因為她被隔離在自己的 container 裡面了。

   你能幫助 Mia 離開她所在的 container，並以 root 的身分執行 `/home/yuu/secret` 嗎?

   ※ 這題請使用 mia 的帳號登入操作

Hint

- 記得在動手前先備份一下。
- 可以參考最一開始提供的指令來備份。
- [Task 1] 若對 docker 不太熟，非常推薦去讀一下 Docker API Example。
- [Task 1] 中有找到很多方式操作的話，建議找最不需要裝額外東西，且最簡潔的方式。
- [Task 2] 有哪些 Image 呢？
- [Task 2] 要怎麼看 Image 是如何被建立的呢？

- [Task 2] Docker API Reference 裡面有完整的 Docker API 資訊。
- [Task 3] container 是誰建立的真的有差嗎？
- [Task 4] dind container 建立的指令有什麼特別的地方嗎？
- [Task 4] 想要逃脫 docker (docker escape) 通常會怎麼做呢？

Submission

- Task 1: 請透過 Google Forms 上傳截圖。
- Task 2: 請透過 Google Forms 回答該檔案內容（格式：NASA...）。
- Task 3: 請找助教 Demo 停止掉 Rina container 裡面的 container 後的狀態。
- Task 4: 請找助教 Demo 以 root 身分執行 /home/yuu/secret 的結果。

---

9 Dockerized Ping Pong! ★★☆☆☆ (20 points)

Description

在這個大題，你需要透過建立 docker network 來讓兩個 docker image 中的程式互相溝通。請先自行從 docker hub pull 以下的 images：

- `jason1024/ping-pong:alice`
- `jason1024/ping-pong:bob`

這兩個 image 中程式會做的事情分別如下：

- alice: 尋找 bob 與透過 TCP 跟 bob 開始 ping pong
- bob: 等待 alice 與透過 TCP 跟 alice 開始 ping pong

Resource

- 請自行 `docker pull jason1024/ping-pong:alice` 與 `jason1024/ping-pong:bob` 兩個 image

Tasks

1. (12 points) 建立新的 docker network，分別用 alice 跟 bob 的 image 建立名叫 alice 跟 bob 的 container，並 attach 到同一個 docker network，讓兩者順利進行 ping pong。如果成功，兩者的輸出都會有 ping pong 等訊息。
2. (8 points) 撰寫 docker-compose.yaml，讓只需要使用 docker-compose up 便可以讓 alice 跟 bob 開始 ping pong。

Submission

- Task 1. 請透過 Google Form 上傳其中一者成功輸出 ping pong 過程的截圖。
- Task 2. 請透過 Google Form 上傳 docker-compose.yaml 的內容與成功 docker-compose up 並包含 ping pong 過程的截圖。
- 請不要用 Task 2 的截圖繳交 Task 1.