網路程式設計概論11組期末專題

Online Judge

組員
林家甫
尤茂為

1.簡介

1.1專題題目簡介

改良版Online Judge，client端選擇欲測試的題目並輸入相對應的檔案名稱，經過加密後傳送給server，server解密後再經過編譯、執行，並比較輸出與正確答案，根據執行時間給client相對應的分數。

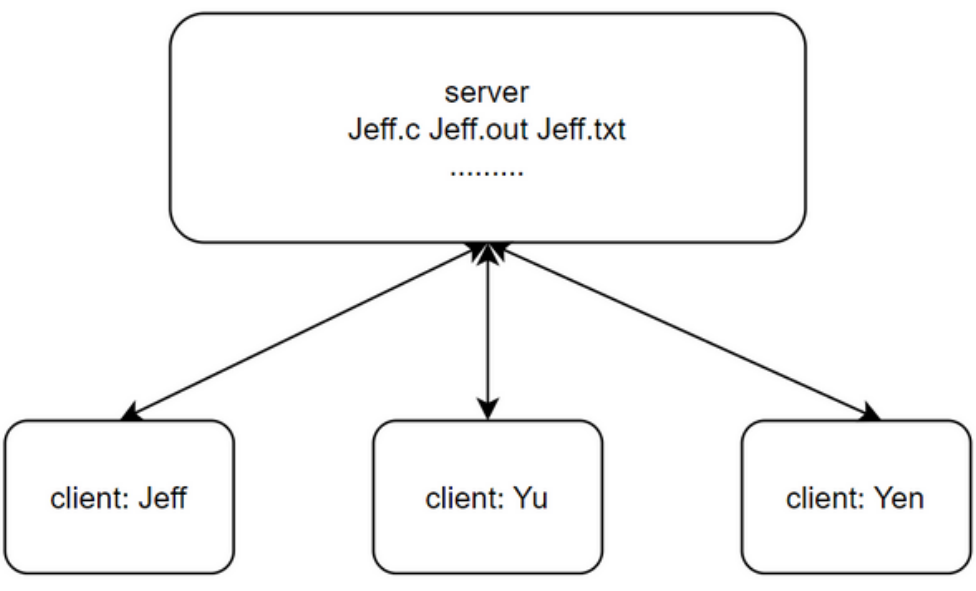
1.2成員分工

林家甫: 專題基本架構、server編譯執行功能、檔案內容加密解密、評分系統構想、報告製作

尤茂為: multi-client同時測試功能、題目及測資設計、報告製作、評分系統構想

1.3開發與執行環境 Ubuntu 22.04.3 LTS

1.4 示意圖



2.研究方法與設計

2.1 Server端功能

1)解密client傳來的source code

2)將source code編譯成執行檔

3)執行該執行檔並與正確輸出答案比對

4)根據答對的測資比例及執行時間，傳送給client分數

5)根據執行結果，傳送AC/WA/TLE/SEG Fault給client

6)保留Scoreboard，查看所有client的使用紀錄

7)自動刪除執行完的執行檔，保留乾淨的空間

8)透過fork()，同時應付多個client端

9)若某client端有預期外的錯誤，不影響server及其他client端的運作

2.2 Client端功能

1)輸入使用者ID與其他user區別

2)選擇欲測試題目、輸出題目內容

3)可針對同題重複測試或繼續測試不同題

4)將欲傳送給server的檔案隨機透過caesar或atbash加密法加密，增加安全性

5)可傳送 C\C++ 類型的檔案

6)支援非本機端測試

2.3 特殊需求

使用者的ID不可重複 (server編譯後檔案會以該ID命名，若使用者名稱相同，可能共享同一個執行檔)

2.4 Server與Client程式互動規則與資料傳輸格式

1)建立連線後，client輸入ID

2)server fork() child來處理此client

3)client傳送欲測試的題目與加密後檔案給server

4)server待收到完整檔案後，解密並編譯

5)server執行編譯後的執行檔，與正確測資比較

6)server傳送測試結果給client

7)client根據測試結果決定要不要繼續測試

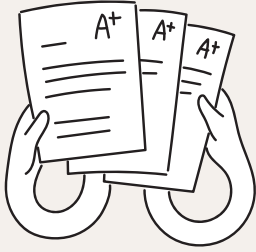
8)server根據client的回覆決定是否關閉server child

server與client間的資料是由字串形式傳輸，client的c/c++檔內容也是由一筆加密後的字串形式傳送

2.5 例外狀況之分析與處理

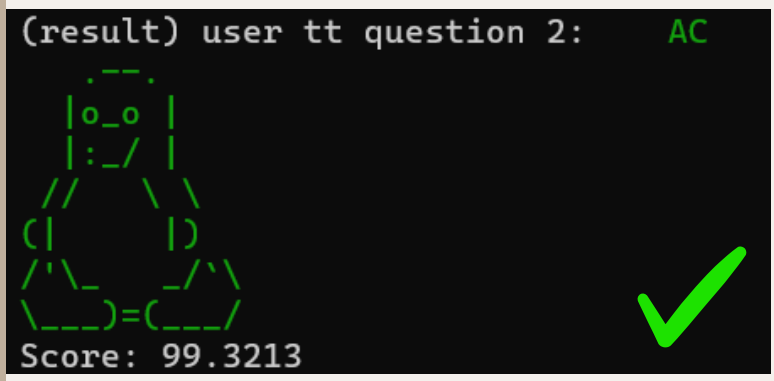
client若傳送非預期形式的回覆，server將會忽略該資料，並結束server child，並不會干擾到其他server-client之間的資料傳輸及連線

3.成果

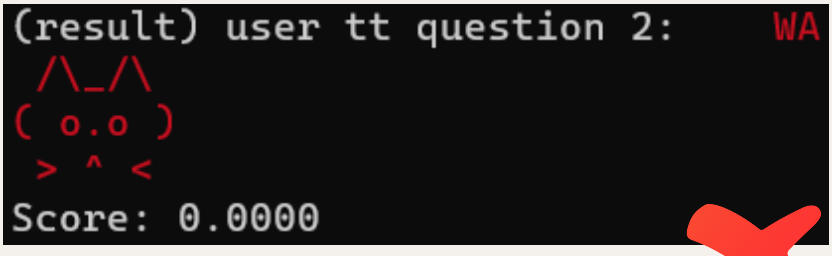


3.1 主要功能與特色

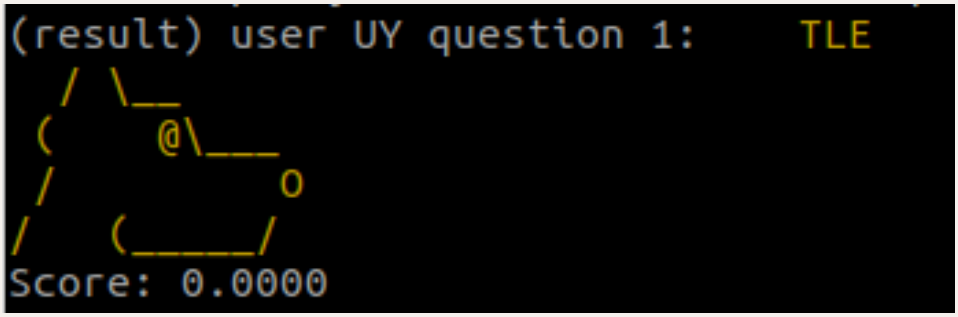
程式名稱、完整的題目敘述、輸出輸入限制



AC結果與酷酷的圖



WA結果



TLE結果

```
ScoreBoard:
ID: nms Problem number: 2       AC      Score: 99.1096
ID: nms Problem number: 3       AC      Score: 99.1370
ID: qoq Problem number: 5       AC      Score: 98.6840
ID: nms Problem number: 4       AC      Score: 99.1049
ID: qoq Problem number: 2       AC      Score: 99.5472
ID: qoq Problem number: 5       WA      Score: 0.0000
```

server端的Scoreboard

```
code:
 #include<stdio.h>

int main(){
        int times;
        scanf("%d", &times);
        while(times--){
                int a, b;
                scanf("%d %d", &a, &b);
                printf("sum: %d\n", a+b);
        }

        return 0;
}


encrypt code:
 ◆◆◆◆◆◆◆ȟ◆◆◆◆◆r' ᴍᴍ◆◆: ◆◆◆ы◆◆ml◆◆: ◆◆◆◆m̃l◆◆◆◆q◆◆Dĭ◆◆◆◆◆◆◆◆ml◆◆◆◆ĭ◆◆◆◆◆◆◆◆ml◆◆◆ll◆◆: ď◆$ml◆◆◆◆q◆◆!◆Dĭ◆◆◆ď◆◆
Ō◆ml◆◆◆◆◆q◆◆◆H◆◆◆ó◆◆◆ĎŌ◆ml◆ᴍml◆◆◆◆◆y◆◆ᴍ◆ᴍᴍ
```

Caesar cipher

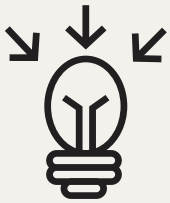
```
encrypt code:
 #rmxofwv<hgwrl.s>

rmg nzrm(){
        rmg grnvh;
        hxzmu("%w", &grnvh);
        dsrov(grnvh--){
                rmg z, y;
                hxzmu("%w %w", &z, &y);
                kirmgu("hfn: %w\m", z+y);
        }

        ivgfim 0;
}
```

Atbash cipher

4.結論



4.1 遭遇困難及解決經過

1)執行完後的執行檔刪除後導致server崩潰

SOL:

server child再fork一個child，交由此child來刪除執行檔

2)client傳送執行檔案(一大筆的字串)後，使後續順序錯亂

SOL:

在傳送的字串後方加上終止符號，使server得知檔案字串已傳送完畢

3) 大量且快速的傳輸資料導致server 接收不及

SOL:

在一系列的write & read，其實在本機端中其實都非常快速的執行，會導致對方的buffer 還沒有盡到user mode，對方就收到新的一個message。常常miss掉message 或者和前一個message 合併在一起。

example

3.1 原本需要被檢測的程式碼是一行一行傳送，可是write 太頻繁，導致server的buffer無法負荷。因此，改成把全部的程式碼存到code[MAXLINE*10]的陣列，再一次送出所有程式碼。

3.2 利用sleep()，讓程式不要快速的傳大量的封包。

再傳送encode type 和傳送code間有sleep(0.5)讓client緩緩。

4)在處理segmentation fault 的同時，scoreboard會印出不正常的字串。推測可能是因為fork 導致process 間的shared memory page 問題。
SOL:
fflush(stdout) ，將stdout buffer 清空

5)多次使用到execlp，所以都必須fork child 來處理。因此，必須妥善處理child process 的問題。
SOL:
利用wait(NULL)，讓parent等待child process執行完再繼續執行

6)如何偵測segmentation fault?
SOL:
偵測有無User_name.out

7)過多的user使用，導致存放執行檔的資料夾空間被浪費
SOL:
server傳送score及測試結果給client後，利用 execlp 來刪除對應的執行檔。

8)client傳送給server的過程過於簡單，傳輸過程中可能會有安全上的風險
SOL:
增加兩種加密法並隨機使用其中一種，將加密後的編碼傳送給server

4.2 專題製作心得

從這次的專題題目發想，到動手實作，到最後的階段，我們都覺得很不可思議。當一開始有這個想法時，我們便著手開始思考流程圖，server 和 client 應該需要有什麼互動、需要有什麼功能。

我也了解到之前在測online judge時，為什麼即使網路好，也需要pending 很久。實作後才發現原來是因為問題討論3的3.2。

寫程式的過程中，每一次多加新的功能都是一個挑戰，從最簡單的一個client、檢查AC or WA，一路到可以服務multiple clients、segmentation fault、encoding等等，每一次的update又多學到了一些新的知識，雖然有部分是小地方的錯誤，但處理完後也是對此次的專題有了貢獻。

對於這次的專題，讓我們更深入了解online judge的實作原理(雖然沒有參考目前存有online judge的code)，此外，除了online judge效能與安全性的考量，我們也注意到了使用者介面的重要，設計完善、直觀且有完整回饋的使用者介面能吸引更多user使用。

總結，這次的專題我們認為除了程式設計，團隊合作及溝通也是重要的一環，對於兩人有不同意見該如何整合、執行，有效的團隊合作使得專題的呈現及效果超出預期。

5. 成果未來改進或延伸方向

5.1 未來發展方向

1)server可以多增加偵測Run Time Error的功能
2)各個client應該有能查看其他人成績的功能。比如說: server 端將結果寫進log file
3) 增加GUI的介面，做成一個user-friendly的程式
4) 比較每個同學的程式碼，檢測是否有抄襲!!
5) 利用AES 等進階加密功能

5.2 參考文獻與附錄

1) execlp
2) ASCII art 的繪製
3) Atbash cipher 演算法
4) Redirect in & Redirect out (這邊原本要利用 pipeline 來實現，結果發現有更好的選擇)
5) fflush
6) 計算時間的gettimeofday