

Lab 0: 基础知识及技能

3250101697冯梓航

由于最近任务略有点多，这个报告很简陋，请见谅。期末之后如果可以重新提交的话我会尽可能补齐！

1. Prerequisite

1.1 Challenge 1: Linux环境的搭建与简单使用

1. 在实验报告中给出任意 4 个 shell 命令的用法介绍以及在 Linux 环境下的实操截图

(a) ls 命令

1		ls	# 简单列出文件
2		ls -l	# 详细列表（权限、大小、修改时间）
3		ls -a	# 显示隐藏文件
4		ls -lh	# 显示文件大小（KB/MB）

(b) cd 命令

1		cd /home	# 进入 /home 目录
2		cd ..	# 返回上一级目录
3		cd ~	# 回到自己的家目录
4		cd -	# 回到上一次所在的目录

(c) cp 命令

1		cp a.txt b.txt	# 复制 a.txt 为 b.txt
2		cp -r dir1 dir2	# 复制整个目录 dir1 到 dir2
3		cp -i a.txt /tmp	# 覆盖前提醒是否确认

(d) grep 命令

1		grep "hello" a.txt	# 在 a.txt 里找包含 hello 的行
2		grep -i "error" log	# 不区分大小写查找 error
3		grep -n "test" a.txt	# 显示匹配内容所在行号

```
BillFeng — bill@node01: ~ — ssh bill@10.211.55.11 — 80x24

permitted by applicable law.
Last login: Wed Jun  3 17:23:13 2026 from 10.211.55.11
[bill@node01:~$ ls
Angband-4.2.5          CBLAS.tgz          setup_compute.sh
Angband-4.2.5.tar.gz  hpl-2.3            ssh_backup_1780479033
BLAS-3.12.0           hpl-2.3.tar.gz     ssh_backup_final_1780479298
blas-3.12.0.tgz       openmpi-5.0.10
CBLAS                 openmpi-5.0.10.tar.gz
[bill@node01:~$ ls -lh
total 67M
drwxr-xr-x 11 bill bill 4.0K May 31 21:25 Angband-4.2.5
-rw-rw-r-- 1 bill bill 25M Aug 19 2023 Angband-4.2.5.tar.gz
drwxr-xr-x 4 bill bill 16K Jun  3 10:37 BLAS-3.12.0
-rw-r--r-- 1 bill bill 324K Jun  3 10:30 blas-3.12.0.tgz
drwxr-xr-x 7 bill bill 4.0K Jun  3 10:41 CBLAS
-rw-r--r-- 1 bill bill 193K Jun  3 10:29 CBLAS.tgz
drwxr-xr-x 11 bill bill 4.0K Jun  3 10:51 hpl-2.3
-rw-rw-r-- 1 bill bill 646K Dec  3 2018 hpl-2.3.tar.gz
drwxrwxr-x 12 bill bill 4.0K Jun  3 10:25 openmpi-5.0.10
-rw-r--r-- 1 bill bill 41M Jun  3 10:17 openmpi-5.0.10.tar.gz
-rw-rw-r-- 1 bill bill 1.5K Jun  4 14:29 setup_compute.sh
drwxrwxr-x 2 bill bill 4.0K Jun  3 17:30 ssh_backup_1780479033
drwxrwxr-x 2 bill bill 4.0K Jun  3 17:35 ssh_backup_final_1780479298
bill@node01:~$
```

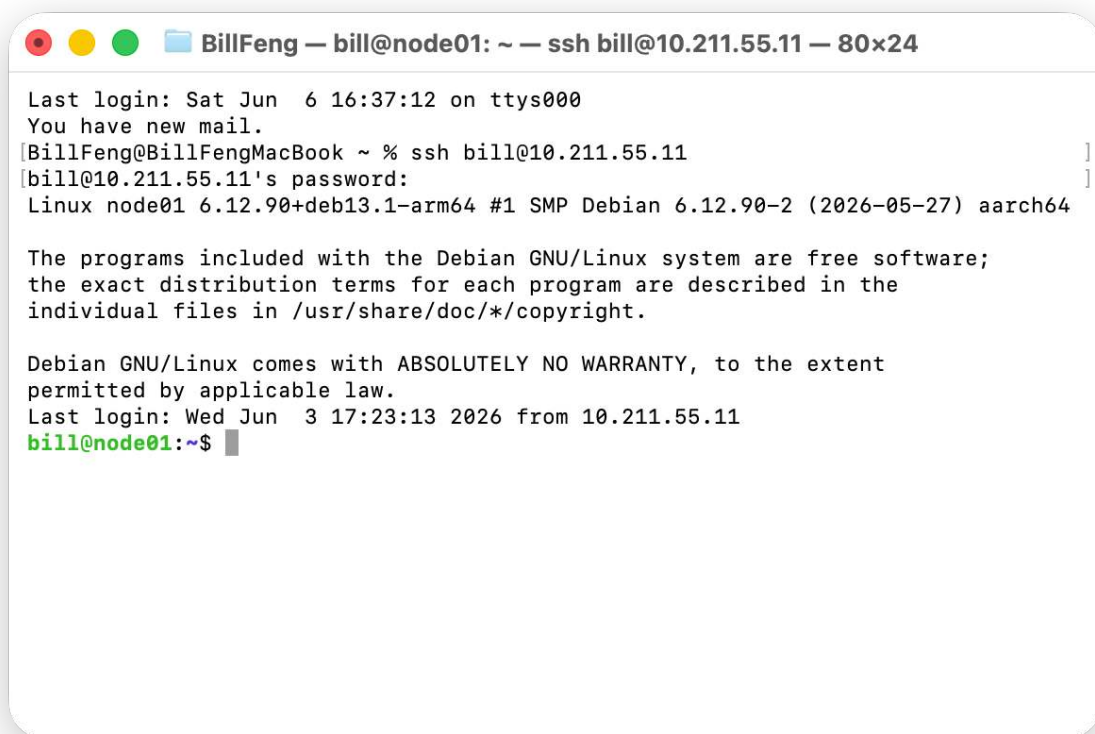
Image 1

```
BillFeng — bill@node01: /cluster/shared — ssh bill@10.211.55.11 — 117x37

[bill@node01:~$ ls -lh
total 67M
drwxr-xr-x 11 bill bill 4.0K May 31 21:25 Angband-4.2.5
-rw-rw-r-- 1 bill bill 25M Aug 19 2023 Angband-4.2.5.tar.gz
drwxr-xr-x 4 bill bill 16K Jun  3 10:37 BLAS-3.12.0
-rw-r--r-- 1 bill bill 324K Jun  3 10:30 blas-3.12.0.tgz
drwxr-xr-x 7 bill bill 4.0K Jun  3 10:41 CBLAS
-rw-r--r-- 1 bill bill 193K Jun  3 10:29 CBLAS.tgz
drwxr-xr-x 11 bill bill 4.0K Jun  3 10:51 hpl-2.3
-rw-rw-r-- 1 bill bill 646K Dec  3 2018 hpl-2.3.tar.gz
drwxrwxr-x 12 bill bill 4.0K Jun  3 10:25 openmpi-5.0.10
-rw-r--r-- 1 bill bill 41M Jun  3 10:17 openmpi-5.0.10.tar.gz
-rw-rw-r-- 1 bill bill 1.5K Jun  4 14:29 setup_compute.sh
drwxrwxr-x 2 bill bill 4.0K Jun  3 17:30 ssh_backup_1780479033
drwxrwxr-x 2 bill bill 4.0K Jun  3 17:35 ssh_backup_final_1780479298
[bill@node01:~$ cd
[bill@node01:~$ cd /
[bill@node01:/]$ ls
bin  cluster  etc  lib  media  opt  root  sbin  sys  usr
boot dev  home  lost+found  mnt  proc  run  srv  var
[bill@node01:/]$ cd cluster
[bill@node01:/cluster$ cd shared
[bill@node01:/cluster/shared$ ls
hpl      hpl_34.err  hpl_35.out  hpl_37.err  hpl_38.out
hpl_33.err  hpl_34.out  hpl_36.err  hpl_37.out  run_hpl.sbatch
hpl_33.out  hpl_35.err  hpl_36.out  hpl_38.err
[bill@node01:/cluster/shared$ grep "Gflops" hpl_38.out
Gflops : Rate of execution for solving the linear system.
T/V      N      NB      P      Q      Time      Gflops
[bill@node01:/cluster/shared$ grep -n "Gflops" hpl_38.out
15:Gflops : Rate of execution for solving the linear system.
45:T/V      N      NB      P      Q      Time      Gflops
[bill@node01:/cluster/shared$ cat hpl_38.out
=====
HPLinpack 2.3 -- High-Performance Linpack benchmark -- December 2, 2018
Written by A. Petit et R. Clint Whaley, Innovative Computing Laboratory, UTK
Modified by Piotr Luszczek, Innovative Computing Laboratory, UTK
```

Image 2

2. 在本机中通过 ssh 远程连接到你的 Linux 环境



A terminal window titled "BillFeng — bill@node01: ~ — ssh bill@10.211.55.11 — 80x24". The window shows the output of an SSH login. It starts with "Last login: Sat Jun 6 16:37:12 on ttys000" and "You have new mail.". Then, the user "BillFeng" runs "ssh bill@10.211.55.11". The prompt changes to "bill@10.211.55.11's password:". After the password is entered, the system displays the Linux version: "Linux node01 6.12.90+deb13.1-arm64 #1 SMP Debian 6.12.90-2 (2026-05-27) aarch64". It then shows the Debian GNU/Linux disclaimer: "The programs included with the Debian GNU/Linux system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/*/copyright. Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law." Finally, it shows the last login time: "Last login: Wed Jun 3 17:23:13 2026 from 10.211.55.11". The prompt is now "bill@node01:~\$".

```
BillFeng@BillFengMacBook ~ % ssh bill@10.211.55.11
bill@10.211.55.11's password:
Linux node01 6.12.90+deb13.1-arm64 #1 SMP Debian 6.12.90-2 (2026-05-27) aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Jun 3 17:23:13 2026 from 10.211.55.11
bill@node01:~$
```

3. 完成 SadServers 上的题目 "Saint John": what is writing to this log file? 给出解答以及通过截图。

```
proxy-us.sadservers.com
2026-06-06 09:05:04.023144 token: 1729006273
2026-06-06 09:05:04.323534 token: 1389474509
2026-06-06 09:05:04.623911 token: 636788592
2026-06-06 09:05:04.924275 token: 527422109
2026-06-06 09:05:05.224640 token: 1071358883
2026-06-06 09:05:05.525025 token: 1015259153
2026-06-06 09:05:05.825406 token: 239088863
2026-06-06 09:05:06.125773 token: 784320790
2026-06-06 09:05:06.426137 token: 1516884576
2026-06-06 09:05:06.726509 token: 727868320
2026-06-06 09:05:07.026883 token: 122327359
2026-06-06 09:05:07.327266 token: 1972755201
2026-06-06 09:05:07.627636 token: 1882765570
2026-06-06 09:05:07.928013 token: 220085213
2026-06-06 09:05:08.228379 token: 538310100
2026-06-06 09:05:08.528723 token: 1276573668
2026-06-06 09:05:08.829110 token: 1814333517
2026-06-06 09:05:09.129514 token: 603120906
2026-06-06 09:05:09.429887 token: 2072634735
2026-06-06 09:05:09.730267 token: 737093605
2026-06-06 09:05:10.030631 token: 812358856
2026-06-06 09:05:10.331026 token: 74804446
2026-06-06 09:05:10.631422 token: 1133831817
2026-06-06 09:05:10.931838 token: 657045526
2026-06-06 09:05:11.232248 token: 433331402
2026-06-06 09:05:11.532609 token: 296106915
2026-06-06 09:05:11.832965 token: 1398618640
2026-06-06 09:05:12.133191 token: 368176567
2026-06-06 09:05:12.433615 token: 392034500
2026-06-06 09:05:12.734009 token: 388599437
2026-06-06 09:05:13.034380 token: 1954063154
2026-06-06 09:05:13.334756 token: 1036410283
2026-06-06 09:05:13.635135 token: 975766819
2026-06-06 09:05:13.935534 token: 31705808
admin@ip-10-1-13-47:/var/log$ fuser /var/log/bad.log
/var/log/bad.log: 591
admin@ip-10-1-13-47:/var/log$ kill 591
admin@ip-10-1-13-47:/var/log$ fuser /var/log/bad.log
admin@ip-10-1-13-47:/var/log$
```

Image 3

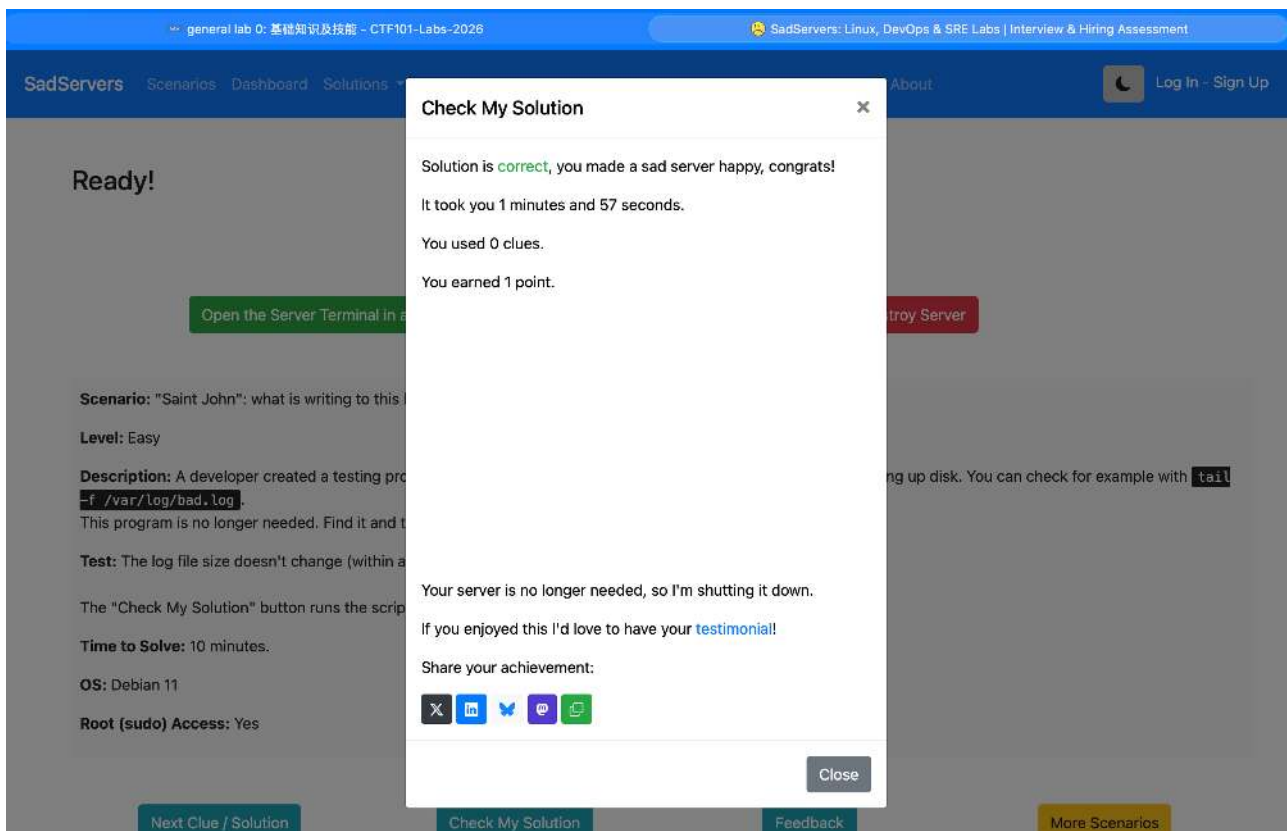
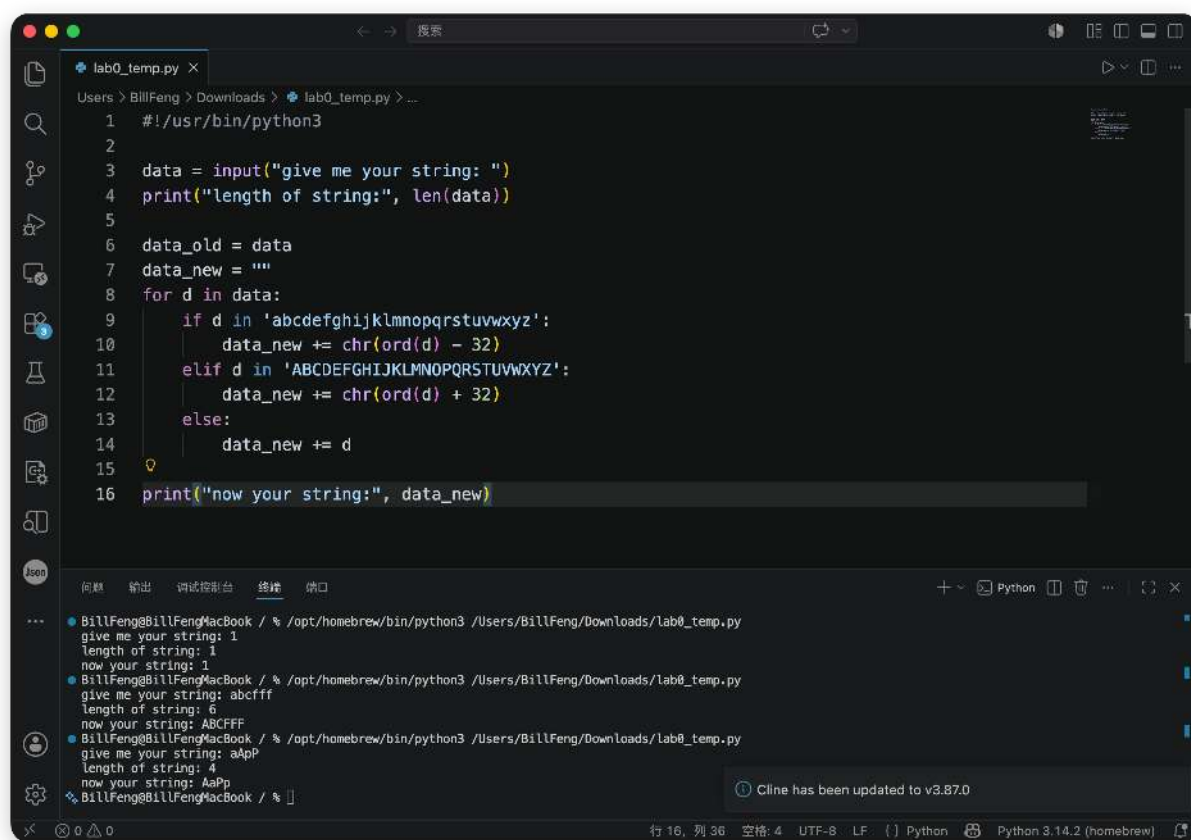


Image 4

1.2 Challenge 2: 基础的Python编程



The image shows a Cline IDE window with a dark theme. The editor displays a Python script named `lab0_temp.py` with the following code:

```
1 #!/usr/bin/python3
2
3 data = input("give me your string: ")
4 print("length of string:", len(data))
5
6 data_old = data
7 data_new = ""
8 for d in data:
9     if d in 'abcdefghijklmnopqrstuvwxyz':
10         data_new += chr(ord(d) - 32)
11     elif d in 'ABCDEFGHIJKLMNOPQRSTUVWXYZ':
12         data_new += chr(ord(d) + 32)
13     else:
14         data_new += d
15
16 print("now your string:", data_new)
```

Below the editor, the terminal output shows three test cases:

```
BillFeng@BillFengMacBook / % /opt/homebrew/bin/python3 /Users/BillFeng/Downloads/lab0_temp.py
give me your string: 1
length of string: 1
now your string: 1

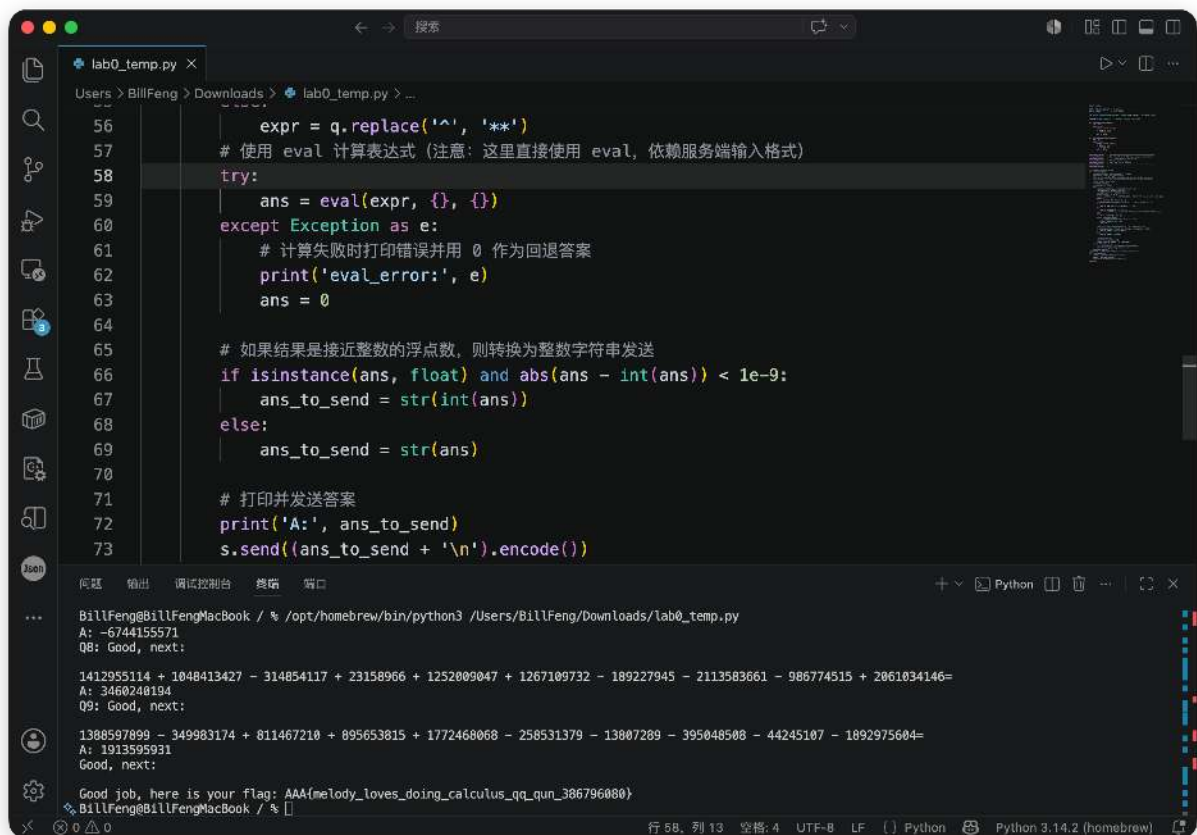
BillFeng@BillFengMacBook / % /opt/homebrew/bin/python3 /Users/BillFeng/Downloads/lab0_temp.py
give me your string: abcfff
length of string: 6
now your string: ABCFFF

BillFeng@BillFengMacBook / % /opt/homebrew/bin/python3 /Users/BillFeng/Downloads/lab0_temp.py
give me your string: aApP
length of string: 4
now your string: AaPp
```

The status bar at the bottom indicates the file is at line 16, column 36, with 4 spaces, UTF-8 encoding, LF line endings, and Python 3.14.2 (homebrew) interpreter.

Image 5

- 实现了字符串长度测量和大小写替换。



```
56     expr = q.replace('^', '**')
57     # 使用 eval 计算表达式 (注意: 这里直接使用 eval, 依赖服务端输入格式)
58     try:
59         ans = eval(expr, {}, {})
60     except Exception as e:
61         # 计算失败时打印错误并用 0 作为回退答案
62         print('eval_error:', e)
63         ans = 0
64
65     # 如果结果是接近整数的浮点数, 则转换为整数字符串发送
66     if isinstance(ans, float) and abs(ans - int(ans)) < 1e-9:
67         ans_to_send = str(int(ans))
68     else:
69         ans_to_send = str(ans)
70
71     # 打印并发送答案
72     print('A:', ans_to_send)
73     s.send((ans_to_send + '\n').encode())
```

BillFeng@BillFengMacBook / % /opt/homebrew/bin/python3 /Users/BillFeng/Downloads/lab0_temp.py
A: -6744155571
Q8: Good, next:
1412955114 + 1048413427 - 314854117 + 23158966 + 1252809047 + 1267109732 - 189227945 - 2113583661 - 986774515 + 2061834146=
A: 3460248194
Q9: Good, next:
1388597899 - 349983174 + 811467210 + 895653815 + 1772468068 - 258531379 - 13807289 - 395048508 - 44245107 - 1892975604=
A: 1913595931
Good, next:
Good job, here is your flag: AAA{melody_loves_doing_calculus_qq_qun_386796080}
BillFeng@BillFengMacBook / % []

Image 6

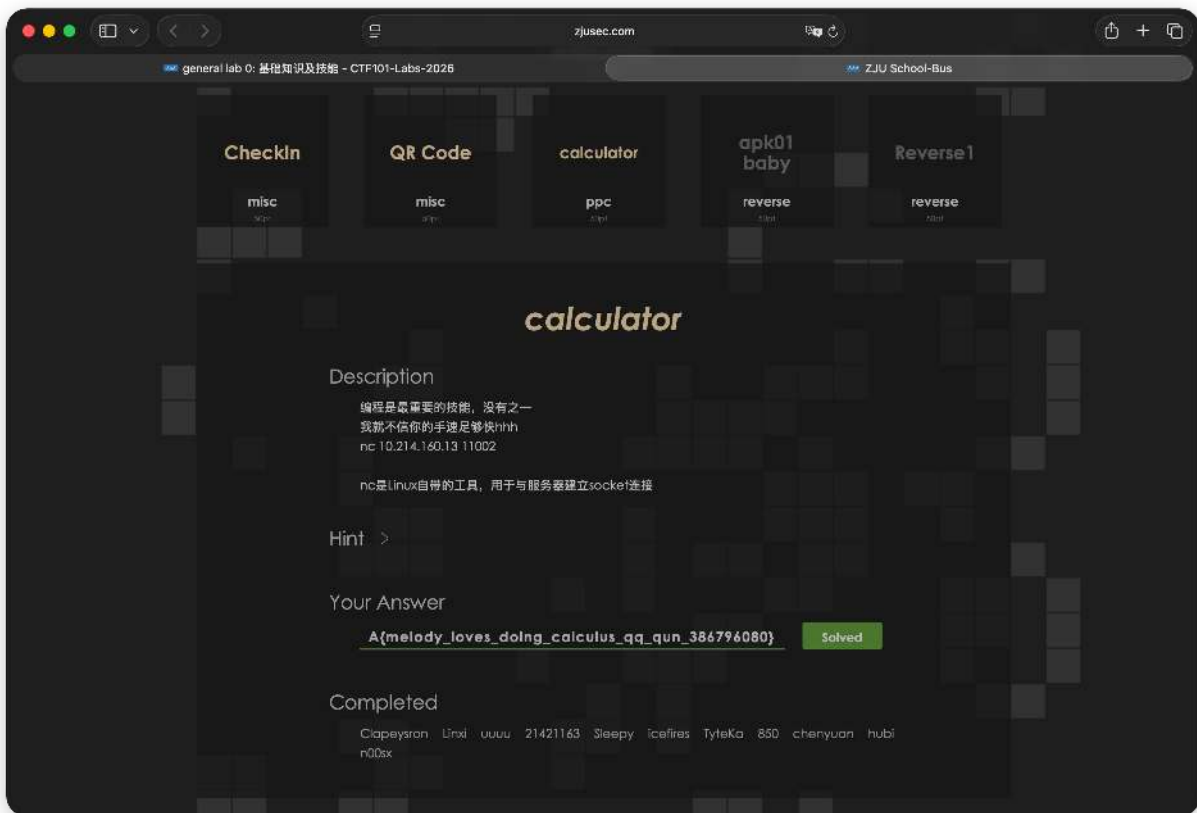


Image 7

1.3 Challenge 3

- asm_tour: `ACTF{We1com3_7o_R3_00_00_0000_0e74_0e4f_030f}`

2. Web

大概只做了第一题

2.1 Challenge 1

首先用浏览器打开网页，我发现按钮会在鼠标移动到对应位置时消失。通过控制台可以看出有一个 `getflag()` 函数，直接调用会向 `/flag.php?token=xxx` 发请求并弹窗显示 "One more time! 1 / 1337", 说明需要累计调用 1337 次才能拿到 flag。

此外每次请求需要从页面重新提取 token (token 一次性使用)，且必须保持同一 Session 才能使计数器累加。每次先请求 `lab0.php` 提取 token 再请求 `flag.php`，最终在第 1337 次成功获取 flag:

`flag{56297ad00e70449a16700a77bf24b071}`。

- ai搓出来的脚本...

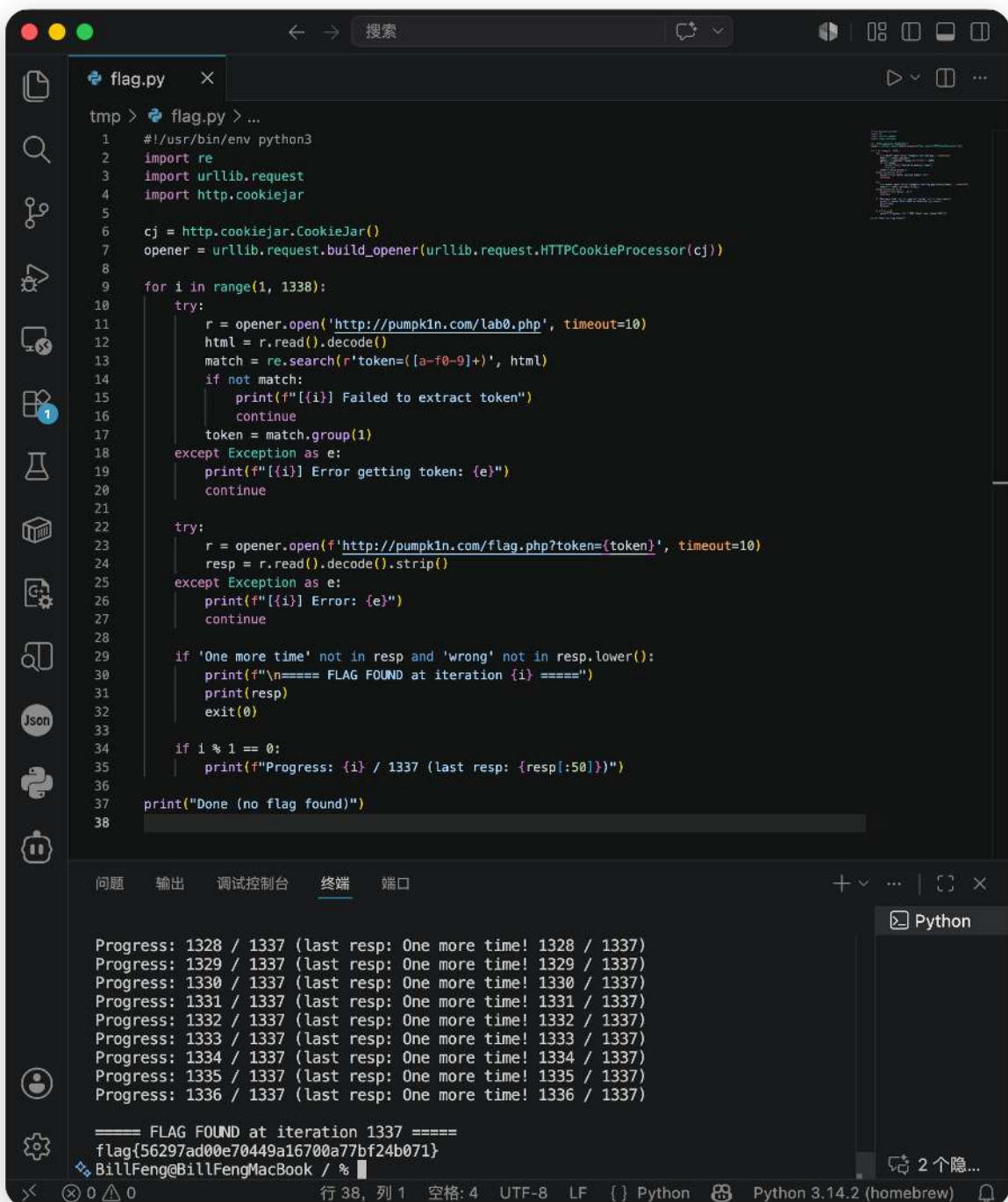


Image 8

3. Reverse

本学期学习了80x86汇编，然而打开ida的时候还是眼花缭乱，学习了一些快捷键后在 `string` 窗口找到了 `Access Granted`。随后定位到此处并生成反汇编代码。通过规则判断出flag，随后在Linux环境中测试flag的正确性。

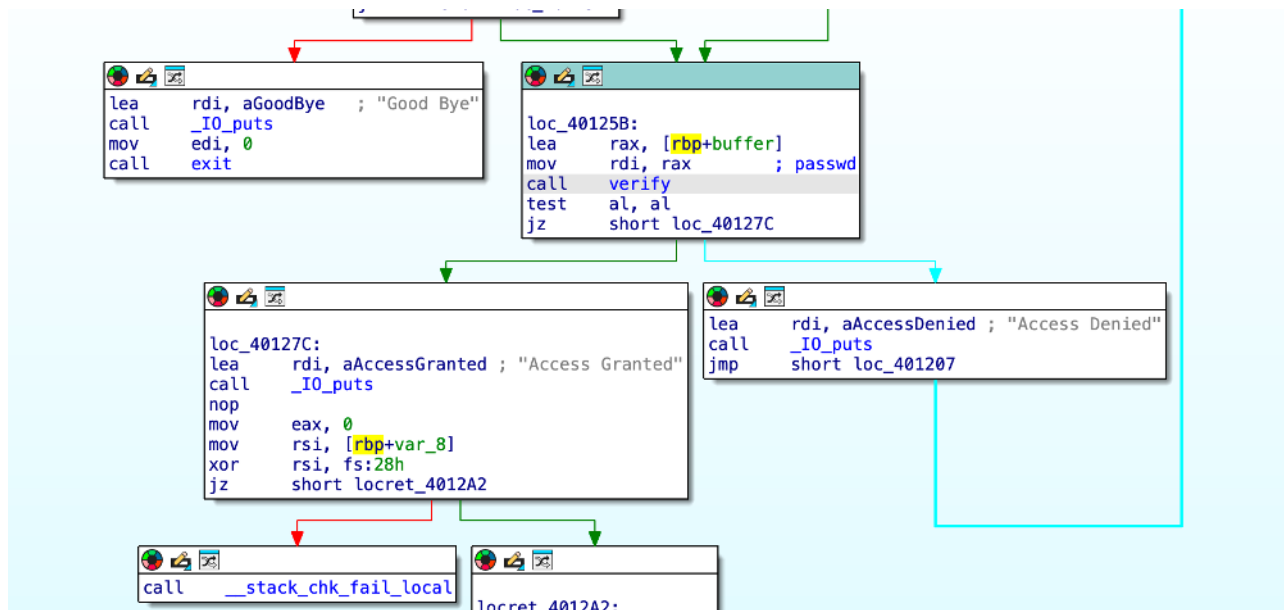


Image 9

```

1 int __fastcall main(int argc, const char **argv, const char **envp)
2 {
3     char buffer[264]; // [rsp+10h] [rbp-110h] BYREF
4     unsigned __int64 v5; // [rsp+118h] [rbp-8h]
5
6     v5 = __readfsqword(0x28u);
7     memset(buffer, 0, 0x100u);
8     while ( 1 )
9     {
10         banner();
11         IO_gets(buffer);
12         if ( j_strlen(buffer) == 1 && buffer[0] == 113 )
13         {
14             IO_puts("Good Bye");
15             exit(0);
16         }
17         if ( !verify(buffer) )
18             break;
19         IO_puts("Access Denied");
20     }
21     IO_puts("Access Granted");
22     return 0;
23 }

```

Image 10

```

1 pool __cdecl verify(char *passwd)
2 {
3     int v2; // ecx
4     int v3; // r8d
5     int v4; // r9d
6     char v5; // [rsp+0h] [rbp-F0h]
7     int i; // [rsp+18h] [rbp-D8h]
8     char *table[14]; // [rsp+20h] [rbp-D0h]
9     char tmp[64]; // [rsp+90h] [rbp-60h] BYREF
10    unsigned __int64 v9; // [rsp+D8h] [rbp-18h]
11
12    v9 = __readfsqword(0x28u);
13    table[0] = "1040";
14    table[1] = "1040";
15    table[2] = "1040";
16    table[3] = "1968";
17    table[4] = "1152";
18    table[5] = "1680";
19    table[6] = "1312";
20    table[7] = "1616";
21    table[8] = "1888";
22    table[9] = "1616";
23    table[10] = "1824";
24    table[11] = "1840";
25    table[12] = "1616";
26    table[13] = "2000";
27    if ( j_strlen(passwd) != 14 )
28        return 1;
29    memset(tmp, 0, sizeof(tmp));
30    for ( i = 0; i < (unsigned __int64)j_strlen(passwd); ++i )
31    {
32        sprintf((unsigned int)tmp, (unsigned int)"%d", 16 * passwd[i], v2, v3, v4, v5);
33        if ( (unsigned int)j_strcmp(tmp, table[i]) )
34            return 1;
35    }
36    return 0;
37 }

```

Image 11

```

parallels@ubuntu-linux-2404:~$ '/media/psf/homework/CTF/lab0/crackme'
Enter Password (or q to quit): AAA{HiReverse}
Access Granted
parallels@ubuntu-linux-2404:~$

```

Image 12

4. Misc

4.1 Challenge 1

- Magic!自动解决了。

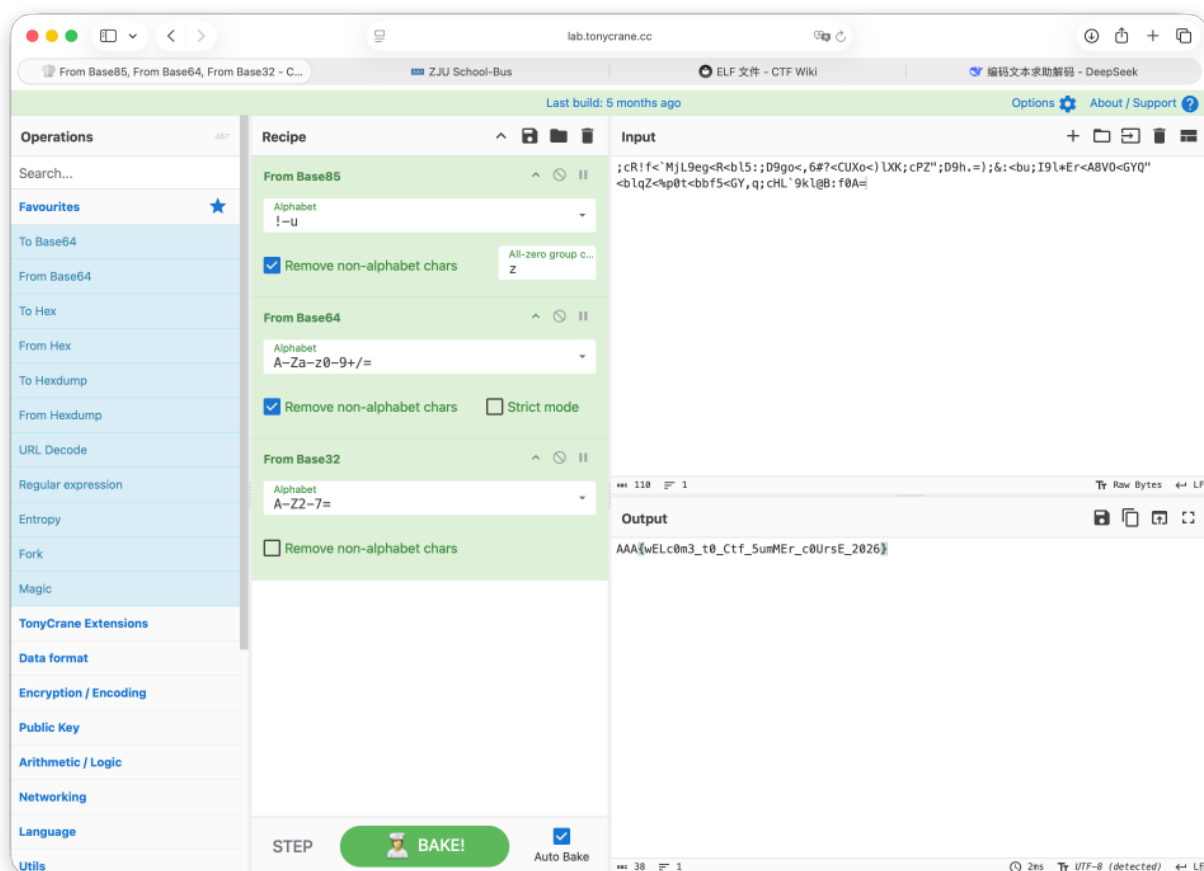


Image 13

4.2 Challenge 2

看来是肉眼难以发现的色差。将整体二值化，背景变黑其余变白即可找到flag的前半段。

2. 关键是：这不是传统的 LSB 隐写！

那些 LSB 变化是因为白色文字经过抗锯齿渲染后，边缘像素的 RGB 值发生了微小变化，导致 LSB 被动改变——而不是故意在 LSB 中编码数据。文字本身就是信息载体。

3. 提取方法：阈值化 + OCR

```
# 把背景色替换为黑，其他像素替换为白，做成高对比度图
new_pixels = []
for p in pixels:
    if (p[0], p[1], p[2]) == (50, 119, 194): # 背景色
        new_pixels.append((0, 0, 0, 255)) # 变黑色
    else:
        new_pixels.append((255, 255, 255, 255)) # 变白色 (文字)
out = Image.new('RGBA', (w, h))
out.putdata(new_pixels)
out.save('text_revealed.png')
```

4. OCR 识别

```
tesseract text_clean.png stdout --psm 3
# 输出: AAA{gr3@t_JQ8!_let'S_
```

核心思路就是：这不是 LSB 比特编码，而是视觉上的文字隐写——文字颜色和背景非常接近（肉眼难辨），但通过阈值化分离后就清晰可见了。

Image 14



Image 15

AAA{gr3@t_J08!_1et'5_

后半段比较简单，直接查看十六进制(hex)即可。

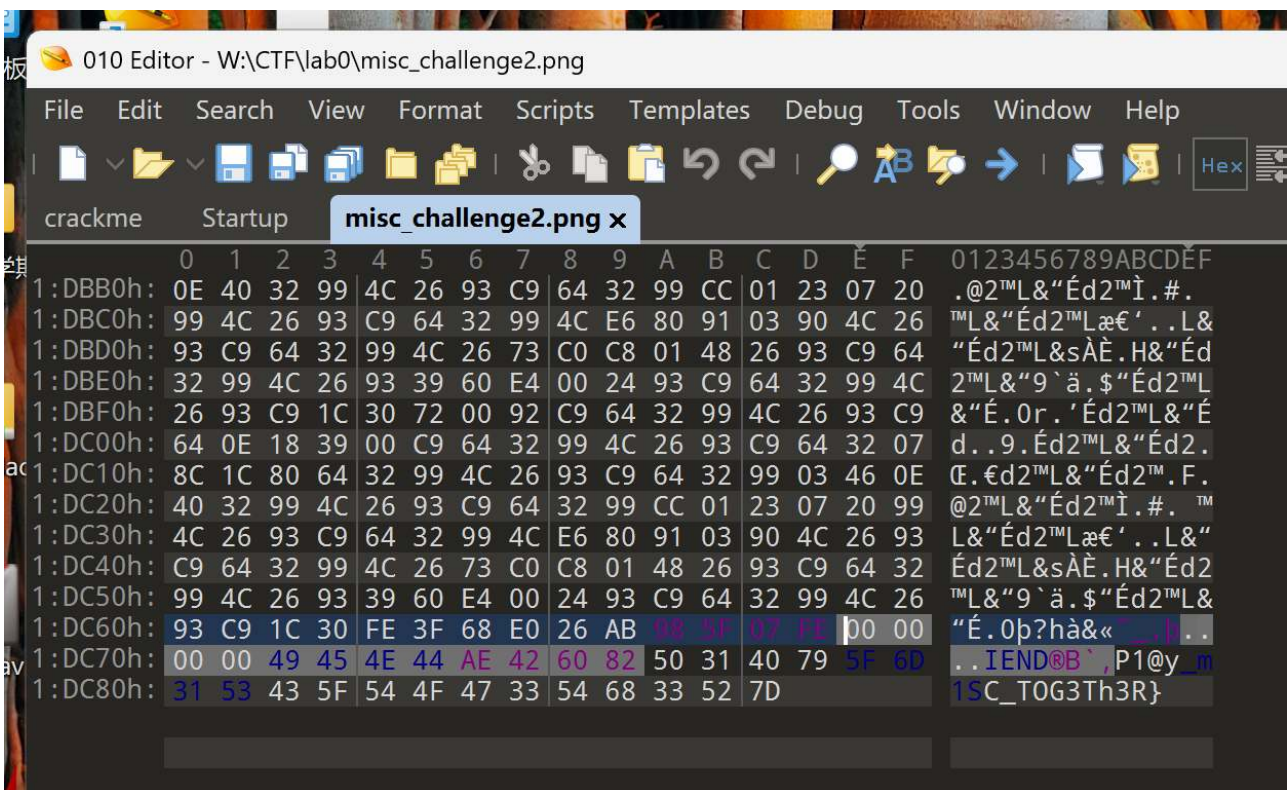


Image 16

P1@y_m1SC_TOG3Th3R}

连起来就是 AAA{gr3@t_J08!_1et'5_P1@y_m1SC_TOG3Th3R}