به نام خداوند بخشنده مهربان

دانشگاه صنعتی شریف
مدرس: مهدی خرازی

ارزیابی تمرین برنامه نویسی اول [*]
شبکه‌های رایانه‌ای

دانشکده‌ی مهندسی کامپیوتر
پاییز ۱۳۹۵

این فایل به ضمیمه تمرین شماره ۱ قرار گرفته است. این فایل نحوه کار با judge (که در سایت درس قرار گرفته)،نحوه نمره‌دهی و تمام سناریوهای مورد آزمون را به شما توضیح می‌دهد.

۱. نحوه کار با judge

داوری این تمرین نیز شبیه به داوری تمرین قبل است، با این تفاوت که سعی کرده‌ایم مشکلات قبلی را رفع کنیم و قابلیت‌های بیشتری نیز داشته باشد. برای کار با judge ابتدا فایل را از سایت دانلود کنید و از حالت zip خارج کنید، سپس به داخل آن بروید و مراحل زیر را دنبال کنید.

۱.۱. تنظیمات پیش فرض

ابتدا باید کد خود را در پوشه `script-runners/Tester/codes` بدون هیچ پوشه واسطی قرار دهید، یعنی مثلا `new.sh` شما باید در آدرس `script-runners/Tester/codes/new.sh` قرار گرفته باشد.
نکته: دقت کنید که از قبل کد درست را در این قسمت قرار داده‌ایم تا در صورتی که خواستید جاج را با کد درست اجرا کنید و از صحت اجرای درست جاج روی کامپیوتر خود مطمن شوید. یعنی اگر کد خود را در این قسمت قرار ندهید و ادامه مراحل گفته شده را دنبال کنید، پس از اجرای جاج باید تمام نمرات شما برابر ۱۰۰ شود. پس در صورتی که از صحت جاج مطمن شدید کد خود را در این قسمت قرار دهید و کدی که وجود دارد را پاک کنید تا جاج روی کد شما اجرا شود.

کد خود را با دستور `make` کامپایل کنید تا فایل اجرایی آن ساخته شود، سپس فایل `info.sh` درون کدتان را ویرایش کرده و قسمت map را برابر با `UDP_Hole_Punching` قرار دهید. همین کار را برای `info.sh`

[*] با سپاس از سولماز سلیمی، رضا میرعسگر شاهی، پارسوآ خورسند، پیمان عزتی، مهدی بهروزی‌خواه

کد درست که درون فولدر `script-runners/Tester/judge` قرار دارد انجام دهید و با همان اطلاعات کاربری **خودتان** پر کنید. دقت کنید محتویات این فایل دقیقا باید شبیه به فایلی باشد که در پوشه کد خودتان قرار دارد.

سپس آدرس دقیق محل پوشه `codes` را باید درون فایل `script-runners/Tester/config.json` در قسمت `cf_path` قرار دهید. همچنین قسمت `judge_path` را نیز با آدرس دقیق پوشه judge که در آدرس `script-runners/Tester/judge` برای شما قرار داده شده است پر کنید.

عدد `sleep_time` مشخص می‌کند پس از اجرای هر دستور روی هر گره چند ثانیه صبر کند تا دستور بعدی را به گره دیگری وارد کند. درصورتی که اینترنت خوبی دارید و با پرتو مشکلی ندارید به این عدد دست نزنید. در صورتی که اینترنت شما کند است یا ارتباط شما با پرتو دچار مشکل است، ممکن است در هنگام اجرای judge به مشکل بخورید و بسته‌های شما قبل از رسیدن داوری شوند. در نتیجه در این حالت‌هابهتر است این عدد را تا ۱۰ افزایش دهید (به طبع افزایش این عدد باعث کندتر اجرا شدن جاج خواهد شد و شما مجبورید زمان بیشتری را منتظر بمانید تا نمره خود را ببینید!).

۲.۱. آماده سازی

برای اجرا کد judge شما نیاز به `Python 2.7` دارید. بدین منظور درون فولدری که از زیپ خارج کردید یک ترمینال باز کنید و فایل `init.sh` را اجرا کنید. برای اینکار می‌توانید از دستور: `./init.sh` استفاده کنید. دقت کنید که باید قبل از اجرای این دستور به اینترنت وصل باشید. پس از این دستور باید `Python 2.7` و تمام پیش‌نیازهای اجرایی judge برای شما درون پوشه venv قرار گرفته باشد.

حال دستور `source start.sh` را اجرا کنید تا از این به بعد ترمینالی که باز کرده‌اید از پایتون ساخته شده استفاده کند. پس تا وقتی که ترمینالتان را نبسته‌اید، می‌توانید از آن استفاده کنید اما دقت کنید که بعد هر بار که ترمینالتان را بستید و دوباره بازکردید و خواستید جاج را اجرا کنید، باید حتما قبلش دوباره این دستور را اجرا کنید.

حال باید با دستور `cd script-runners` به پوشه script-runner بروید. تا همه چیز برای اجرا کردن کد محیا شود. برای اجرا کافیست دستور `./run.sh` را بزنید تا کد جاج شروع به اجرا شدن بکند.

۳.۱. خروجی‌های جاج

هنگامی که جاج در حال اجراست، به شما گزارش لحظه‌ای می‌دهد که در حال چه کاری است. هر بار به سراغ یک آزمون می‌رود، اسم آزمون را به رنگ آبی برایتان می‌نویسد و شروع می‌کند سناریو های مختلف آن را اجرا کردن و می‌توانید ببینید که بسته‌هایی رد و بدل می‌شوند. در انتهای هر سناریو بسته‌ها و خروجی‌های شما چک می‌شوند و اگر نمره آن سناریو را گرفتید، نمره شما بروز می شود. در نهایت برای هر آزمون جمع نمره تمام سناریوهای آن‌ها برابر

۱۰۰ می شود. پس اگر کد شما تمام تست‌ها را با موفقیت بگذراند باید در انتها ۴ عدد ۱۰۰ ببینید.
برای اینکه بفهمید چه بسته‌هایی رد و بدل شده و هر گره‌ای چه چیزی چاپ کرده پس از هر بار اجرای جاج، یک فایل با نام `log` برای شما ساخته می‌شود که درون آن به ازای هر سناریو یک پوشه با نام آن سناریو قرار دارد. یک نمونه از log درست نیز برای شما با نام `correct_log` قرار داده شده است که در آخر باید log شما نیز مانند این شود. درون هر سناریو خروجی تمام گره‌های درگیر آن تست را برای شما قرار داده‌ایم. در فایل های `stdout` خروجی‌ای است که گره مورد نظر چاپ کرده است. فایل‌های `recv` بسته‌های دریافتی هر گره روی interface مربوطه‌اش است. و فایل‌های `send` بسته‌های ارسالی هر گره. برای باز کردن فایل‌های `.pcap` نرم افزارهای مختلفی وجود دارد، از جمله این نرم‌افزار ها می‌توانید از نرم افزار معروف `wireshark` استفاده کنید تا هر بسته شما را به تفکیک سرآیند هایش نشان دهد.
فایل `error.log` نیز در صورتی که مشکلی نداشته باشید، خالی است و در صورتی که در آن سناریو مشکلی داشته باشید، مشکل شما را می‌نویسد.

۴.۱ اطلاعات اضافی

در صورتی که راجع به نمره دهی جزئی تر هر سناریو و جاج اطلاعات بیشتری می‌خواهید، می‌توانید به فایل‌های مربوط به هر تست مراجعه کنید. برای اینکار کافیست، به پوشه `script-runners/Tester/tests/` مراجعه کنید. در اینجا فایل هر آزمون را می‌بینید و درون هر آزمون می‌توانید هر سناریو را ببینید. در انتهای هر سناریو، خروجی‌هایی که چک می‌شوند را می‌توانید ببینید. اطلاعات بیشتری در مورد آماده سازی جاج در فایل Readme نیز قرار گرفته‌است که می‌توانید آن بخش را نیز مطالعه کنید.

۲. موارد آزمون

همانطور که در صورت تمرین گفته شده بود، حالت‌های مختلفی برای مسئله وجود دارد. برای صحت عملکرد کد شما، آزمون‌های مختلفی در قالب سناریوهای متفاوتی طرح شده اند. در تمام این سناریوها از نقشه‌ی جدیدی با نام `UDP_Hole_Punching` استفاده شده است که از الآن به بعد می‌توانید برای تست کد خودتان نیز از این نقشه استفاده کنید. تنها نکته‌ای در مورد این نقشه می‌توان گفت این است که گره شماره ۳ در آن کارگزار است و گره‌های ۴، ۷، ۱۳ و ۱۰ نیز گره‌های nat هستند. سایر گره‌های ۰ تا ۱۴ مربوط به کارخواه هستند.

هر آزمون شامل چند سناریو است. در هر آزمون تنها تعداد خاصی از اعضای نقشه فعال هستند. هر گره، دو حالت دارد که یا کد درست روی آن در حال اجراست، یا کد شما. برای تمام سناریوهای یک آزمون، این اعضا و حالتشان ثابت است، اما در اول هر سناریو این اعضاء ریست می‌شوند تا عملکرد گره‌های شما در سناریو قبلی تأثیری در سناریو جدید نگذارد. هر سناریو به این صورت است که دستوراتی به اعضاء شبکه وارد می‌شود تا بسته‌هایی رد و بدل شوند. سپس تمام بسته‌های خروجی،ورودی و تمام پیام‌های چاپ شده هر گره مورد بررسی قرار می‌گیرند و درست بودن هر کدام قسمتی از نمره شما را تشکیل می‌دهد.

اعضای هر سناریو را با لیستی نشان می‌دهیم که حرف c در آن لیست یعنی در این گره کد شما اجرا می‌شود و در صورتی که حرف j باشد یعنی کد درست در آن گره اجرا می‌شود. همچنین در انتهای هر سناریو گفته شده که به ترتیب به هر گره‌ای چه دستوری وارد شده و سناریوی مورد نظر چیست. در اینجا نمرات هر سناریو را آورده‌ایم، اما در صورتی که میخواهید جزئیات بیشتری از هر سناریو ببینید، می‌توانید به توضیحات مربوط به judge مراجعه کنید و کد هر تست را بخوانید.

در این تمرین ۴ آزمون وجود دارد، که هر آزمون `100` نمره دارد اما نمره کل شما ضریبی از هر آزمون است. به این صورت که دو آزمون اول هر کدام `40` نمره و دو آزمون آخر هر کدام `20` نمره دارند، که در مجموع `120` نمره می‌شود و به این دلیل است که آزمون شماره ۳ مربوط به بخش امتیازی است. جزئیات آزمون‌ها با سناریوهایشان را در زیر آورده ایم.

۱.۲ Single Client

- اعضای درگیر آزمون:

  `0: c, 1: j, 2: j , 3:j`

  در این تمام سناریو های این آزمون شما تنها یک عضو شبکه هستید و در نتیجه تنها قابلیت‌های اصلی کارخواه روی کد شما مورد آزمون قرار می‌گیرد.

- سناریو test_connect_to_server: `20` نمره

  در این سناریو تنها بررسی می‌شود که شما بتوانید بسته نوع صفر را بفرستید و دریافت کنید.

0: make a connection to server on port 2000

- سناریو test_get_info: 20 نمره

در این سناریو تنها بررسی می‌شود که شما بتوانید بسته نوع یک را بفرستید و جوابش را دریافت کنید.

0: make a connection to server on port 2000

1: make a connection to server on port 3000

0: get info of 2

- سناریو test_local_session: 20 نمره

در این سناریو تنها بررسی می‌شود که شما بتوانید بسته نوع دو به آدرس محلی را بفرستید و جوابش را دریافت کنید.

0: make a connection to server on port 2000

1: make a connection to server on port 3000

0: get info of 2

0: make a local session to 2

- سناریو test_public_session: 20 نمره

در این سناریو تنها بررسی می‌شود که شما بتوانید بسته نوع دو به آدرس عمومی را بفرستید و جوابش را دریافت کنید.

0: make a connection to server on port 2000

1: make a connection to server on port 3000

0: get info of 2

0: make a public session to 2

- سناریو test_send_message: 20 نمره

در این سناریو تنها بررسی می‌شود که شما بتوانید پیامی بفرستید.

0: make a connection to server on port 2000

1: make a connection to server on port 3000

0: get info of 2

0: make a local session to 2

0: send msg to 2:salam

۲.۲. Server_Clients

- اعضای درگیر آزمون:

`0:c, 1:c, 2:c, 3:c, 4:j, 5:c, 6:c`

در این تمام سناریو های این آزمون شما کل اعضای شبکه بجز nat هستید و باید تمام عملکردهای کارگزار و کارخواه را درست انجام دهید.

- سناریو test_simple_routing: نمره `25`

در این سناریو تنها بررسی می‌شود که شما بتوانید بسته نوع صفر را بفرستید و دریافت کنید.

0: make a connection to server on port 2000

- سناریو test_simple_msg: نمره `25`

در این سناریو تنها بررسی می‌شود که شما بتوانید ارتباطی برقرار کنید و پیامی بفرستید و جواب دهید.

0: make a connection to server on port 2000

2: make a connection to server on port 3000

2: get info of 1

2: make a local session to 1

2: send msg to 1:salam

0: send msg to 2:aleyk

- سناریو test_simple_nat: نمره `25`

در این سناریو بررسی می‌شود شما بتوانید در صورتی که nat شما درگاهی را بسته دوباره روی پورت‌های دیگر تلاش کنید.

4: block port range 3000 3400

5: make a connection to server on port 3310

- سناریو test_local_nat: نمره `25`

مانند سناریو قبل با این تفاوت که باید بتوانید دو عضو NAT را از طریق آدرس محلی بهم وصل کنید.

5: make a connection to server on port 3300

4: block port range 3000 3500

6: make a connection to server on port 3410

6: get info of 1

6: make a local session to 1

۳.۲. Optional Nat

- اعضای درگیر آزمون:

```
3: j, 7: c, 9: j, 10: c, 11: j
```

در این آزمون تمام اعضا شبکه کد درست هستند و تنها شما جای دو NAT حاضر در شبکه هستید و باید بسته‌های دریافتی و ارسالی را درست مدیریت کنید.

- سناریو test_optional_nat_simple: نمره 25

در این سناریو تنها بررسی می‌شود که شما بتوانید یک بسته را از درون به بیرون و از بیرون به درون ترجمه کنید

9: make a connection to server on port 2000

10: bad command

- سناریو test_optional_nat_complicated: نمره 75

در این سناریو شما باید بتوانید بازه‌ای از پورت‌ها را مسدود کنید و کاربران پشت NAT نتوانند با آن پورت‌ها عبور کنند، همچنین باید بتوانید بسته drop را برگردانید.

7: block port range 3000 3500

9: make a connection to server on port 3310

11: make a connection to server on port 4000

9: get info of 2

9: make a public session to 2

۴.۲. Complicated Test

- اعضای درگیر آزمون:

```
3: c, 7: j, 2: c, 8: c, 9: c, 13: j, 14: c
```

در این تمام سناریو های این آزمون شما تمام عضوهای شبکه بجز NAT ها هستید و شبکه ساختار پیچیده تری نسبت به حالت‌های قبل دارد.

- سناریو test_exceptions_1: 25 نمره

این سناریو مخصوص چک کردن حالت‌های خاص گفته شده در انتهای صورت تمرین است.

14: bad command

2: get info of 2

- سناریو test_exceptions_2: 25 نمره

این سناریو برای چک کردن حالت‌های خاص گفته شده در انتهای صورت تمرین است.

14: make a connection to server on port 3000

2: make a connection to server on port 2000

2: make a connection to server on port 2000

14: get info of 2

14: send msg to 2:salam

- سناریو test_complicated: 50 نمره

در این سناریو بررسی می‌شود شما بتوانید ارتباط دو عضو پشت nat که هر بسته‌هایشان به دست هم نمی‌رسد را برقرار کنید. این سناریو معادل هدف تمرین یعنی روش punching hole است.

13: block port range 3000 3500

14: make a connection to server on port 3310

8: make a connection to server on port 3000

9: make a connection to server on port 3000

8: get info of 3

8: make a public session to 3

9: get info of 2

9: make a public session to 2

8: send msg to 3:salam

9: send msg to 2:chetori