ネットワーク
システム管理 #11

たかさきこうや
1限 (09:00-10:45)

1

前回のおさらい

- Tracerouteとラウンドトリップタイム(RTT)
- 通信の要素には容量と遅延がある
- 通信経路、ダイナミックルーティング(動的経路)、スタティックルーティング(静的経路)、ロンゲストマッチ
- ポートとプロトコル
- 平文と暗号化
- HTTP、SMTP

2

第2回課題の提出方法

- 第1回同様メールで提出してください(書き忘れてた)

- ファイル名：c24XXXX-NSA02.pptx
  (半角英数で、前半は自身のアカウント名とします)
- 提出方法：メールに添付しkohya@cuc.ac.jp に送信
- メールの件名：NSA02
  (半角英数とします)
- メールの本文: (不要)
- もう出してしまった人はそれで良いです

3

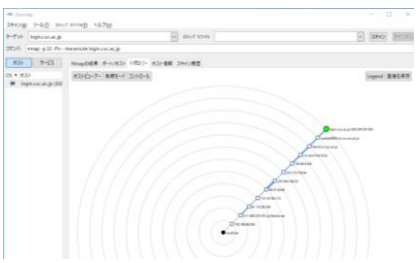
Nmap(Zenmap)によるtracerouteの例(1)

- www.cuc.ac.jpに対して
  443番ポート
  あてに
  traceroute
  - ※右の例は
    いつもとは
    違う経路を
    通っている

4

Nmap(Zenmap)によるtracerouteの例(2)

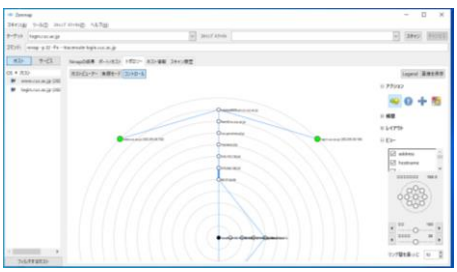
- login.cuc.ac.jp
  に対して
  22番ポート
  あてに
  traceroute



5

両方の結果を比較

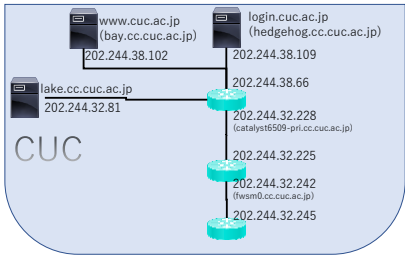
- loginと
  wwwの
  直前まで
  の経路は
  同じ



6

…とすると

- 大学内部側はこんな感じ
- (下の方は割愛)



7

第5回目の講義で

- こんなことをやった
- Wiresharkというアプリをインストールしてもらい通信状況を閲覧
- あれをもう一度やってみよう
- Wiresharkを起動

8

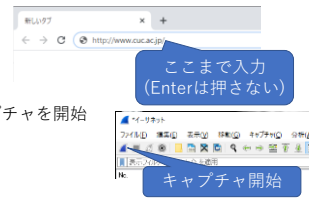
余談：Wiresharkのアップデート

- 割と良く出るので、気が向いたらソフトウェアを更新して最新の状態にしておくことをお勧めします
- ※Wiresharkに限った話ではない

9

ちょっとトリッキーだけど

- ブラウザのアドレスバーに、http://www.cuc.ac.jp/ と入力(httpsではない)し、エンターを押さずに待機
- Wiresharkでパケットキャプチャを開始
- ブラウザのアドレスバーでエンターを入力



10

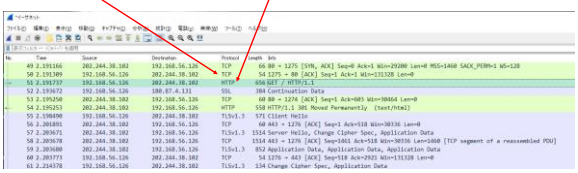
すると

- ブラウザの方は、一瞬だけ http://www.cuc.ac.jp/ につながり、即座に https://www.cuc.ac.jp/ にアドレスが変わる
- 一瞬だけ🛈と出るのは、「暗号化されていない」というアラートメッセージ

11

パケットキャプチャを停止

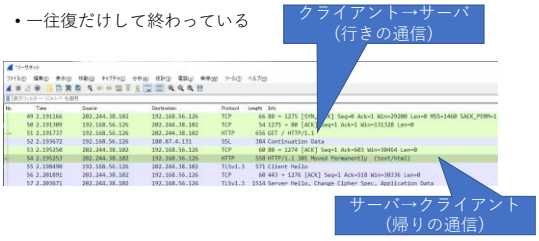
- して、中身を確認すると、確かに192.168.5.126(自PC)から202.244.38.102(www.cuc.ac.jp)に対して、TCP通信が行われ、中身がHTTP通信であることがわかる



12

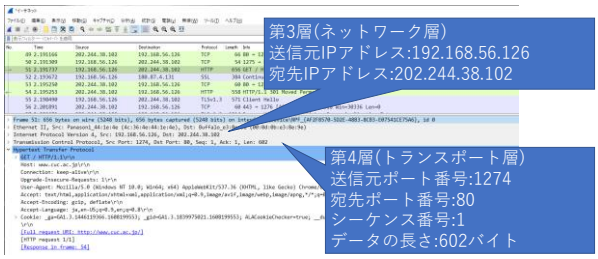
HTTP通信は

- 一往復だけして終わっている



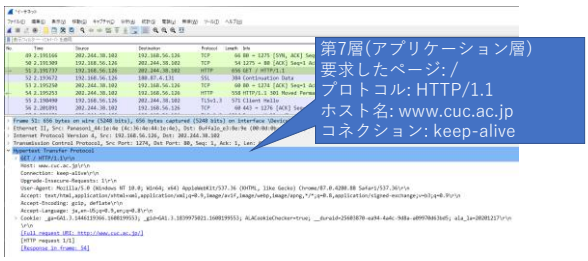
13

「行きの通信」の中身



14

「行きの通信」の中身(続き)



15

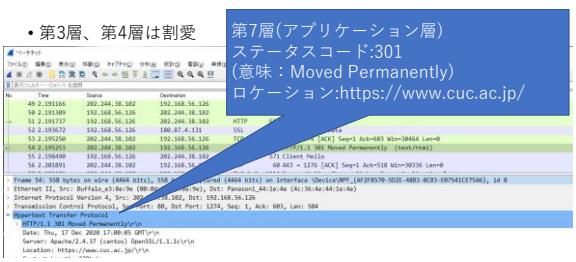
HTTPは

- OSI参照モデルでは7層に相当する「と考えられる」
- が、HTTP自体は5層、6層の内容も含んでいるので「HTTPは5-7層」という説明がされることもある

16

「帰りの通信」の中身

- 第3層、第4層は割愛



17

HTTPのステータスコード

- よく、
「404 Not Found」とか
「500 Internal Server Error」とかがブラウザに表示されているの、見たことない？

- ステータスコード: サーバからの応答で、何が起きたかをクライアントに伝えるための一意な番号

18

HTTPのステータスコード

- 情報 (100–199)
- 成功 (200–299)
- リダイレクト (300–399)
- クライアント側エラー (400–499)
- サーバ側エラー (500–599)

19

ステータスコードの例1

- 200 OK
  - リクエストが成功しデータを返送する
  - Webページは正常に表示される(はず)
  - 炎上した有名人のサイトで「404 Not Found」とか「500 Internal Server Error」と表示されているときに、ステータスコードを見ると200が返されていることがある——そういう時は「ページを消して逃げたふり」をしている
- 301 Moved Permanently … 恒久的移動
  - ページが移動した時に、元の場所にアクセスしたら301が返るようにしておくとよい。※郵便物の転送設定みたいなもの

20

ステータスコードの例2

- 304 not modified
  - アクセスには成功したが、以前アクセスしたときと変わっていない場合、304を返し中身は返さないことで、データの転送量を節約できる
- 401 Unauthorized
  - 認証が必要であり、このステータスコードが返ると大抵のブラウザは認証画面を表示しユーザ名とパスワードの入力を促す

21

ステータスコードの例3

- 403 Forbidden
  - アクセスが禁止されている
- 404 Not Found
  - アクセスした先にページがない
- 500 Internal Server Error
  - サーバが(バグなどで)エラーになっている
- 503 Service Unavailable
  - 過負荷やメンテナンス中などでサービス不可になっている

22

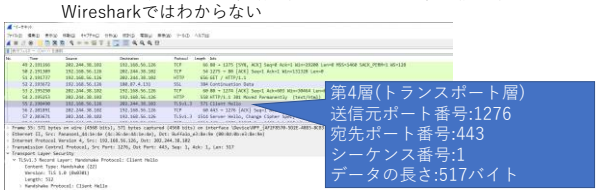
ステータス

- SMTPにしろHTTPにしろ、クライアントとサーバがやり取りをするプロトコルは、異常を検知するとなんらかの手がかりを必ず残す
- なにか異常が起きたときに、「ステータスコード」を見て原因追及をする
  (特に、自分が悪いのか、相手が悪いのかを特定する)ことは非常に重要

23

話を戻すと

- 301 Moved Permanently を受け取ったブラウザは、http://www.cuc.ac.jp/ に代わって https://www.cuc.ac.jp/ にアクセスを試みる…が、暗号化されているので、通信の中身はWiresharkではわからない



24

昨今

- 多くのプロトコルは暗号化されているので、Wiresharkでパケットキャプチャしても通信の実態が確認できることはまずない
- 無論今でも「ダメなサイト」はある

DMMさんとこは割とアカンので間違っても公共のWi-Fiでゲームやっちゃだめよ…

25

サーバを設置する意味

- 例えば、CUCは学内、学外に対する情報発信を行うためにサーバを設置している
  - cuc.ac.jpというドメイン取得もその一環
- ただし、サーバで実施したいサービスや、そのための機能というのは各組織、各部署で**「おなじではない」**
  - 異なった機能が求められることもある

26

SLA

- Service Level Agreement
  - **サービスレベル合意**
- サービスを利用する側と利用させる側がどの程度のサービスをするかについて合意すること

27

どういう場合に合意が必要か

- 例えば、携帯端末の通信速度
  - 速度の上限は定義されていても速度の下限は約束されることはまずない
- 速度は兎も角、99.9%のSLAをうたうサービスも有る
  - 例えば、 1 か月は60x24x30 で、43200分
  - そのうち1/1000→43分までは、サービスが使えなくても責任は問えない、という意味
    - 99.9%だとほんのわずかな気がするけど、1か月のうち40分っていうと割と長いよね…

28

SLAの必要性

- 例えば、朝8時から午後８時までしか使わない、特定少数しか利用しない事務系のシステムに、24x365の稼働は求められない
- 一方、不特定多数が利用するWebサービスは、24x365のサービスが求められる
- が、それが止まっていて困る人はあまりいない
  - そこで金銭の収受が行われないなら

29

逆に

- 株やFX、トレード系は、数秒の停止が致命的なロスに繋がるのでサービス時間内に予告なしに止まることは許されない
- Ex: 東証のダウン(2020年10月01日)

30

混同しがちな例

- 例えば、サーバ用HDDや高価な筐体は
  「故障する確率を下げてくれる」が、所詮確率なので
  「起きるときは起きる」
- だから、
  データセンター(サーバを預かる施設)、
  ネットワーク回線、
  サーバ、CPU、メモリ、HDD、電源
  などを多重化して、全体がストップする可能性を避け
  SLAをあげていく

31

SLAに対する補償

- ただし、99.9%のSLAがあったとして、43分を超えて止まったシステムがあったとして、そのときシステム提供者はサービス利用者に何を「保証を違えた」として「補償」するか
- 割と「え、こんなもん？」という補償レベルだったりする

- ので、サービスの稼働を堅くしようと思った場合、
  「自分で頑張る」しかないのが正直なところ

32

大事なこと

- そのサービスは
  - 「誰のためのもの」なのか
  - 「いつ使うためのもの」か
  - 「何処から使うためのもの」か
  - 「何を扱うサービス」なのか
  - 「何故それでなくてはいけない」のか

- を、考えて、使用するリソースを決定しよう

33

「プログラム」と「ファイル」

- UNIXもWindowsも、ファイルを作ることは割と簡単だ

- では、プログラムとは何か？

- コンピュータ上では、プログラムはファイルの一種にすぎない

34

例えば

- Windows上で、テキストエディタで

```
@echo off
echo あいうえお
timeout /T 3
```

  とだけ書いて、ファイルを test.bat として保存する
- そうすると、アイコンが普段と変わる

- これをダブルクリックしてみると…？

35

Windowsの場合

- 「プログラム」か「ファイル」かは、拡張子で決まる
  - 拡張子が.txtの場合は普通のファイルとして振舞う
    - 普通のファイルとして振舞う場合、Windowsによって.txtに紐づけられたプログラムが起動する
    - 「既定のプログラムの選択」という機能で変更できる(Win10の場合は「既定のアプリの選択」)
      - .txtをメモ帳以外(EmEditorやXyzzy)に変更することも可能

- .batは、Windowsバッチファイルと決められている

36

つまり

- 悪意のある人が、
  - 「普通のWindowsではプログラムとして動くような拡張子」のファイルを貴方のPC上に置き
  - そのファイルのアイコンを適切に設定し
  - その拡張子がバレないようにして
  - 貴方にダブルクリックさせたら
- その時点で貴方は**「負け」**です

37

ちなみに、さっきのファイル

- 「重要書類.xls＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿.bat」
- みたいなファイル名にして保存してみて？
  - 下線部赤のところはスペースをいっぱい入れてみる
- かなり騙されそうな感じになってない？
  - 悪用しないこと！！！！！！！！

- 拡張子を表示しない設定のWindowsの場合、騙されやすさは格段に向上する(ので、拡張子は表示するのが鉄則だ)

38

UNIXの場合

- 若干事情が違う
- UNIX上で

```
echo “あいうえお”
sleep 3
```

とだけ書いて、test.txtでホームディレクトリに保存

- ./test.txt
  として実行しようとしても、実行できない
- 「許可がありません」と言われる
  - 許可とは？？？

39

ファイルの権限の話

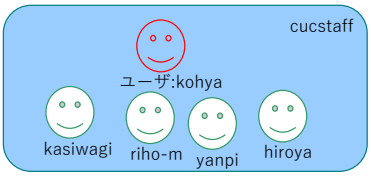
- UNIXでは、「自分自身」が誰なのかをまず明確に規定する

ユーザ:kohya

40

ユーザkohyaは

- 必ず、一つ以上のグループに所属する
- kohyaの場合は、cucstaffというグループ(みんなはstudent?)



41

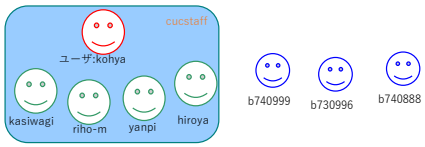
whoamiとgroups

- whoamiは自分が何というユーザであるか
- groupsは自分がどのグループに属しているか
- を表示するコマンド
- groupsの場合、複数グループに属していたら複数表示される
  - その場合先頭グループがデフォルトのグループ

42

更に

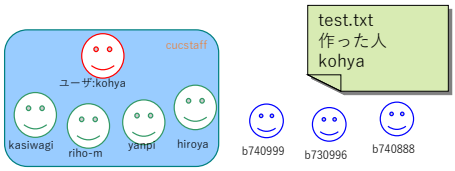
- 少なくともcucstaffというグループに所属していない、多くの人がいる



43

さて、一方

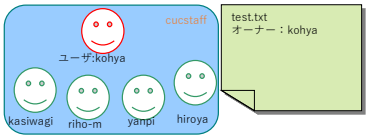
- test.txtというファイルは、kohyaというユーザが作ったもの



44

kohyaというユーザは

- このファイルに対し、
  「読む権利」
  「書き込む権利」
  「プログラムとして実行する権利」
  を指定できる

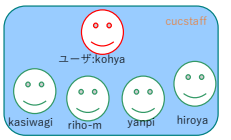


User	読む権利	あり
	書く権利	あり
	実行する権利	あり

45

さらにファイル自身も

- どのグループに属するファイルかという情報を持つことが出来る

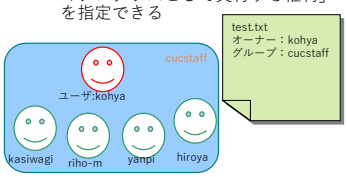


User	読む権利	あり
	書く権利	あり
	実行する権利	あり

46

ファイルが属するグループに対し

- このファイルを
  「読む権利」
  「書き込む権利」
  「プログラムとして実行する権利」
  を指定できる

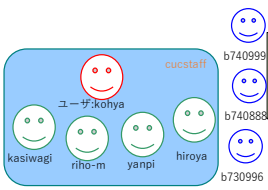


	権限	有無
User	読む権利	あり
	書く権利	あり
	実行する権利	あり
Group	読む権利	あり
	書く権利	なし
	実行する権利	あり

47

更にオーナーでもグループでも

- ない人たちに対し、このファイルを
  「読む権利」「書き込む権利」
  「実行する権利」を指定できる

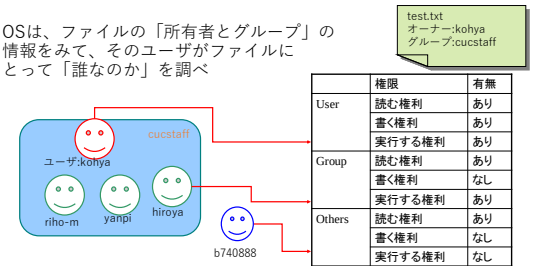


	権限	有無
User	読む権利	あり
	書く権利	あり
	実行する権利	あり
Group	読む権利	あり
	書く権利	なし
	実行する権利	あり
Others	読む権利	あり
	書く権利	なし
	実行する権利	なし

48

すると…

- OSは、ファイルの「所有者とグループ」の情報をみて、そのユーザがファイルにとって「誰なのか」を調べ

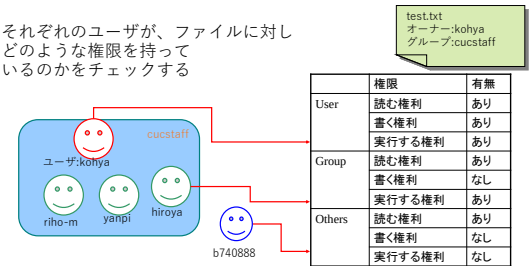


	権限	有無
User	読む権利	あり
	書く権利	あり
	実行する権利	あり
Group	読む権利	あり
	書く権利	なし
	実行する権利	あり
Others	読む権利	あり
	書く権利	なし
	実行する権利	なし

49

更に

- それぞれのユーザが、ファイルに対しどのような権限を持っているのかをチェックする

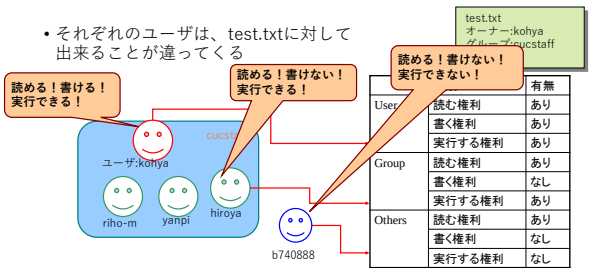


	権限	有無
User	読む権利	あり
	書く権利	あり
	実行する権利	あり
Group	読む権利	あり
	書く権利	なし
	実行する権利	あり
Others	読む権利	あり
	書く権利	なし
	実行する権利	なし

50

その結果

- それぞれのユーザは、test.txtに対して出来ることが違ってくる



	権限	有無
User	読む権利	あり
	書く権利	あり
	実行する権利	あり
Group	読む権利	あり
	書く権利	なし
	実行する権利	あり
Others	読む権利	あり
	書く権利	なし
	実行する権利	なし

51

ここで確認

- UNIXでは、ファイル名とその拡張子が何か、とそのプログラムが実行できるか、は基本的に「関係がない」
- .batだろうが.txtだろうが.shだろうが、実行ユーザに対して実行権限がついていれば実行できるしついてなければ実行できない

52

コマンド： chmod

- ファイルの権限を変更する
- 例

```
chmod u-w ファイル名   ←ユーザから書き込み権を奪う
chmod g+x ファイル名   ←グループに実行権を与える
chmod o-r ファイル名   ←他人から読み込み権を奪う
```

  - こういう、与奪できる権限のことを「パーミッション」と言う

53

先ほど作ったファイル

- chmod u+x test.txt
- とすると、プログラムに「実行権限」がつく

54

じゃあ、実行権限を付けたとして

- UNIXにログインしていれば常にそのプログラムが使えるのか？
- 仮に使えるとしたら、何か問題がないか

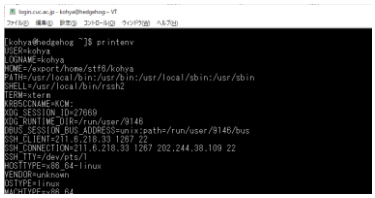
55

もし、UNIXの任意の場所に

- 誰かにlsというプログラムを作られて、それが悪意のあるプログラムだったとしたら、
「何処に置いてあってもそのプログラムを探し出して実行する」ようになっているのはものすごくマズイよね？
- なので、UNIX(やWindows)では
「実行パス」という概念がある
- printenv というコマンドを実行してみよう

56

環境変数



- コンピュータを使う上で、「今どういう設定なのか」が逐一記録される場所がある
  - 環境変数、という
- これの「PATH」の欄に「何処にあるプログラムは実行しても良いか」を記録してあり、ここにあるもの以外は迂闊に実行しない設定になっている
  - PWDとかHOMEなんかも心当たりあるかな？

57

下準備

- cd ← ホームディレクトリに移動
- mkdir bin ← binというディレクトリを作成
- cd bin ← binの下に移動
- echo “cal” > test.sh ← test.shの中にcalとだけ記述
- chmod u+x test.sh ← test.shにオーナーの実行権を付与

58

従って

- もし何らかのプログラムを作り、それを常に実行したいならそのプログラムのある場所に「パスを通す」か、
「パスの通っている場所にプログラムを置く」かする必要がある
- そうでないなら、プログラムをフルパスで実行する
- 例えば、個人のホームディレクトリに、プログラム専用の置き場としてbinというディレクトリを作った場合、これを実行する方法は次の通り

59

実行例

- フルパスで指定して実行
  - ~/bin/test.sh
- そのディレクトリまで行って実行
  - cd bin/
  - ./test.sh
- パスを通してから実行
  - setenv PATH "$HOME/bin":"$PATH"
  - test.sh

60

パスを通すってなんだよ

- 実行プログラムを置いた場所(パス)を「安全なプログラムが置いてある場所とみなして、実行可能なプログラムの置き場所としてあらかじめ登録」すること
- /usr/binとか/bin には、誰もが安心して使えるプログラムがおいてある
- cd /bin
  とかして、どんなプログラムが置かれているか見てみよう

61

毎回パスを通すのは面倒だ

- .tcshrcというファイルをemacsで作る
- 中身は
- setenv PATH "$HOME/bin":"$PATH"
- とだけ書く
- この状態で、今ログインしているターミナルとは別にterateramを立ち上げ、そちらでもlogin.cuc.ac.jpにログイン
- ログインできなかったら、.tcshrcの書き方が間違っているのでもとから開いてたteratermで内容確認のこと

62

.tcshrcって何よ

- 環境設定ファイル
- ログイン時にあらかじめ設定されておいて欲しい内容をいろいろ書くと、ログイン時に自動的に設定してくれる
- WindowsやMacにも同様のものがある
  - ※毎回面倒だ、と思うようなことは自動化しよう
  - ※コンピュータはそういうのが得意だ
- なお、「.」から始まるのでlsでは見えない
  (ls –aをすれば見える)

63

.tcshrcに

- alias rm "rm -i"
- alias cp "cp -i"
- alias mv "mv -i"
- と書いてみよう
- 有効にするにはログインしなおす必要があるが、面倒な人は
- source .tcshrc
- とすると、.tcshrcを再度読み込んでくれる

64

コマンド: alias

- あるコマンドに別の名前を付けること
- alias mv "mv –i"
- とすると、mv –i をmvコマンドとして登録できる
- つまり、mvと打つだけでmv –i を自動的にしてくれる
- では、
  alias koko "hostname; pwd; whoami"
  とすると？

65

コマンド: unalias

- aliasの逆
- unalias koko
- とすると、kokoというaliasを消してくれる

66

便利コマンドの自作

- 簡単なコマンドの組み合わせならaliasコマンドで十分
- alias dou 'ping -c 1 8.8.8.8 | grep icmp_seq | cut -f7 -d" " '

- コマンドを囲むのは
  ダブルクオートでもシングルクオートでもよい

1. pingを1回、8.8.8.8に打つ
2. grep icmp_seqとして1回目のpingの結果行だけ取得する
3. スペースを区切り文字として7番目のフィールドをcutで取得

67

8.8.8.8のところを

- 自宅、ないしは自宅になるべく近いルータへの
  pingにして、大学からのRTTを計測するコマンドを
  「dou」という名前で創ってみよう
- それを.tcshrcに書いてみよう

68

sedコマンド

- 文字列を置換する
- ping -q -c 3 -i 2 www.google.com | grep "packet loss"
  | cut -f 3 -d "," | sed 's/packet loss//'

- sed の後ろに 's/A/B/' と書いてやると、文字Aを文字Bに
  置き換えてくれる
  - 今回の場合は「% packet loss」を「」(からの文字列)に置き換えている…つまり、「% packet loss」という文字を除去している

69

1コマンドの中に

- シングルクオートとダブルクオートが出てくるので、
  （逃げ道はあるけど）test.shの中に書いてしまった方が
  間違いがなくて良い
- パケットロス率の%表示だけを得られる
- ここが0%なら問題はなにもない
- 更にここから「空白を除去」「%」を除去　すると
  「数字だけ」が得られる
- 0より大きければ、異常発生

70

最終課題

- 「何か」を問診をするコマンドを、UNIX上で実装せよ
- それが分かると何かに問題がないことが分かるようなもの
- ファイル名は~/bin/nsa-final.shとする
- より詳細な情報は次回とする

71