情報科学演習第一回

担当教員：小島　英春 教員
提出者：近藤 大翔
学籍番号：09B16031
メールアドレス：u702715k@ecs.osaka-u.ac.jp

提出年月日：2018/4/18

1　課題 1-1

1.1　1

ping コマンドとは指定したホストとの間でネットワークが疎通しているかどうかを調べるコマンドである。引数が通信相手となるので ping exp101 の時は exp101 が通信相手となる。ping コマンドは継続的に疎通を確認してしまうので強制終了させる必要がある。実行結果は以下のようになる

```
64 bytes from exp101.exp.ics.es.osaka-u.ac.jp (192.168.16.65):
 icmp_seq=1 ttl=64 time=0.202 ms
```

64byte は、導通試験をするのに送信したデータのバイト数。from の後は接続確認先の IP アドレスだ。icmp __ req は何回目の導通試験かを表している。ping コマンドはオプションなしで実行すると継続的にパケットを送信する。なので req は送った回数を出力している。ttl はネットワーク用語の TTL で、ネットワーク機器を通過するたびに 1 ずつ消費され、0 になるとその通信データは破棄される。現在の ttl は 6 4 なので後 64 回機器を通過することができる。こちらの ttl がパケットに付属している理由はパケットがネットワークに留まり続けることを防ぐためである。

これで疎通が確認できた。exp102,exp103 と引数を変更してコマンドを試して見た所

```
t-kondoh@exp059:~$ ping exp102
ping: unknown host exp102
t-kondoh@exp059:~$ ping exp103
ping: unknown host exp103
```

のような結果になった。これは対応するアドレスのパソコンが存在しない場合に出るエラーになるので通信を確認することはできなかった。

1.2　2

課題として挙げられている二つの url にアクセスした所同じページにアクセスした。違うアドレスにアクセスしたがなぜ同じページにアクセスできたのかを考察する。これら二つのアドレスはドメイン名と IP アドレスというものに分けられる数字の方が IP アドレスで文字が含まれていた方がドメイン名と呼ばれるネットワークの通信で先に登場したのは IP アドレスの方である。ネットワークで相手を指定するために振り分けられている番号のことでこの番号がそれぞれのパソコンごとに一つずつ割り振られている。インターネット上での住所のような役割を担っている。これらのアドレスを利用することでホスト間の通信が可能になったが、数字の羅列であるため使いづらいものであった。そこで登場したのがドメイン名です。IP アドレスごとに紐づけられた独立したドメイン名を持たせることでそのページごとに特徴を記入する

ことができるようになった。
www-higashi.ist.osaka-u.ac.jp を例にとって考えると、大阪大学であることや東研究所であることがドメイン名から判断できるようになっている。こういう歴史があり特定のページにアクセスする際にドメイン名からでも IP アドレスからでもアクセスできるようにな t。あ

1.3　3

nslookup というコマンドはトラブルシューティングの際に使われる基本的なコマンドである。役割は DNS サーバーに情報を問い合わせる時に使うコマンドである。DNS は（2）で説明したホスト名と IP アドレスを紐付けるシステムである。しかし、nslookup に渡すのはドメイン名だけではいけない、ドメイン名の他にホスト名も指定する必要がある。ホスト名とはホストに割り当てられてい

るドメイン名のことである。ホスト名とドメイン名を足し合わせたものを完全

修飾ドメイン名という。
引数に IP アドレスを渡せば完全修飾ドメイン名が帰ってくるし、引数を逆にすれば結果も逆になる。このコマンドを用いて 1 、 2 それぞれの結果を振り返る。（1）は実行結果を確認すると

```
exp101.exp.ics.es.osaka-u.ac.jp (192.168.16.65)
```

となっていてホスト名と IP アドレスが相互に確認できるようになっている。が、こちらのアドレスにはホスト名がないので結果を nslookup コマンドで結果を確認したところ

```
kondouhirotonoMacBook-Air:desktop kondoutaisyou$ nslookup exp101.exp.
ics.es.osaka-u.ac.jp
Server: 10.0.1.1
Address: 10.0.1.1#53

** server can't find exp101.exp.ics.es.osaka-u.ac.jp: NXDOMAIN
```

のようになり IP アドレスを検知することができなかった。
(2) はホスト名も指定されているのでそれに対応する IP アドレスが帰ってくる。

```
Name: www-higashi.ist.osaka-u.ac.jp
Address: 133.1.17.66
```

nslookup コマンドで以下の内容を検索すると

```
t-kondoh@exp059:~$ nslookup www-ise4.ist.osaka-u.ac.jp
Server:     192.168.25.6
Address:    192.168.25.6#53
Non-authoritative answer:
Name:    www-ise4.ist.osaka-u.ac.jp
Address: 133.1.16.2
```

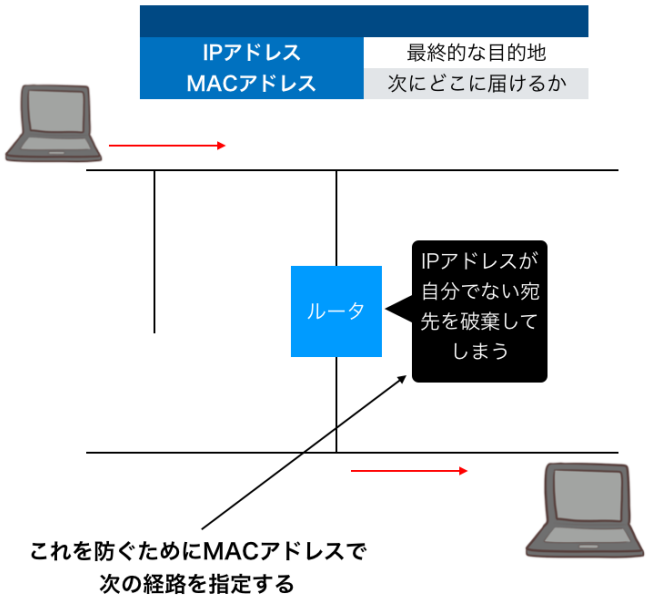
これで数字のアドレスが判別したのでアクセスした所同じページにアクセスすることができた。

2　1-2

2.1　4

コマンド arp は ARP テーブルを確認する時に使うコマンドである。ARP テーブルとは IP アドレスと MAC アドレスとの対応が書かれているものである。

どちらもコンピューターの場所を指す住所のようなものなのだが、これらは使われ方が異なっている。IP アドレスは最終的な目的地であるのに対して、MAC アドレスは次にどこにいくかを指定するために書かれる。これは図を用意したのだが、ネットワークの経路となるルーターは自分宛でない情報を破棄してしまう性質がある。なので情報が破棄されるのを防ぐために MAC アドレスでそのルーターを経由してあげる必要があることを宣言している。



/usr/sbin/arp -a と打つことによってファイル内の ARP テーブルを一括で表示することができる。出力結果は以下のようになった

```
  t-kondoh@exp059:~$ /usr/sbin/arp -a
? (192.168.16.254) at 14:18:77:10:31:aa [ether] on ens192
svm-01.exp.ics.es.osaka-u.ac.jp (192.168.16.241) at 02:a0:98:c4:b2:cf [ether] on ens192
exp101.exp.ics.es.osaka-u.ac.jp (192.168.16.65) at 00:50:56:b7:10:4e [ether] on ens192
cups.exp.ics.es.osaka-u.ac.jp (192.168.16.253) at 00:50:56:b7:5b:4b [ether] on ens192
dhcp-01.exp.ics.es.osaka-u.ac.jp (192.168.16.240) at 00:50:56:b7:21:6e [ether] on ens192
```

この中で MAC アドレスは 00:50:56:b7:10:4e の部分になる

2.2　5

　/bin/ping　ホスト名　でも ping　ホスト名　と同様に ping コマンドを実行できるコマンド実行前の/usr/sbin/arp -a の結果は以下のようになった

```
  t-kondoh@exp059:~$ /usr/sbin/arp -a
? (192.168.16.254) at 14:18:77:10:31:aa [ether] on ens192
svm-01.exp.ics.es.osaka-u.ac.jp (192.168.16.241) at 02:a0:98:c4:b2:cf [ether] on ens192
exp101.exp.ics.es.osaka-u.ac.jp (192.168.16.65) at 00:50:56:b7:10:4e [ether] on ens192
cups.exp.ics.es.osaka-u.ac.jp (192.168.16.253) at 00:50:56:b7:5b:4b [ether] on ens192
dhcp-01.exp.ics.es.osaka-u.ac.jp (192.168.16.240) at 00:50:56:b7:21:6e [ether] on ens192
```

そこで exp100 のに対して ping コマンドを実行すると

```
  t-kondoh@exp059:~$ /usr/sbin/arp -a
? (192.168.16.254) at 14:18:77:10:31:aa [ether] on ens192
svm-01.exp.ics.es.osaka-u.ac.jp (192.168.16.241) at 02:a0:98:c4:b2:cf [ether] on ens192
exp101.exp.ics.es.osaka-u.ac.jp (192.168.16.65) at 00:50:56:b7:10:4e [ether] on ens192
exp068.exp.ics.es.osaka-u.ac.jp (192.168.16.45) at 00:50:56:b7:3c:16 [ether] on ens192
exp100.exp.ics.es.osaka-u.ac.jp (192.168.16.61) at 00:50:56:b7:22:1a [ether] on ens192
cups.exp.ics.es.osaka-u.ac.jp (192.168.16.253) at 00:50:56:b7:5b:4b [ether] on ens192
dhcp-01.exp.ics.es.osaka-u.ac.jp (192.168.16.240) at 00:50:56:b7:21:6e [ether] on ens192
```

イーサネットフレームを用いる時には IP アドレスだけではパケットを送信することができず、MAC アドレスを用いる必要がある。IP アドレスと MAC アドレスの対応表となるのが ARP テーブルである。その MAC アドレスを IP アドレスから割り出してその場所に保存する。なので新規の ARP テーブルが作成された。

2.3　6

　traceroute コマンドは指定したホスト名または ip アドレスまでの経路を調べるためのコマンドになっている。指定された三つのアドレスを指定してみる。

　exp101 に対してコマンドを実行した。

```
  t-kondoh@exp093:~$ /usr/sbin/traceroute exp101
traceroute to exp101 (192.168.16.65), 30 hops max, 60 byte packets
 1  exp101.exp.ics.es.osaka-u.ac.jp (192.168.16.65)  0.258 ms  0.954 ms  0.942 ms
```

　この例ではルーター等の経由がなく直接指定したホストと繋がっていることがわかる。

　次に www.ics.es.osaka-u.ac.jp に対してコマンドを実行する

```
 1  192.168.16.254 (192.168.16.254)  0.942 ms  1.111 ms  1.164 ms
 2  icsintsvgw.ics.es.osaka-u.ac.jp (133.1.240.254)  0.706 ms  0.870 ms  1.067 ms
 3  icsintgw.ics.es.osaka-u.ac.jp (133.1.240.81)  0.638 ms  1.001 ms  1.355 ms

 4  * * *
 5  * * *
 6  * * *
 ~~~~~~~~
```

というような結果が得られた。これは実際のインターネットを経路が通っているので経路が暗号化されていることがわかる。

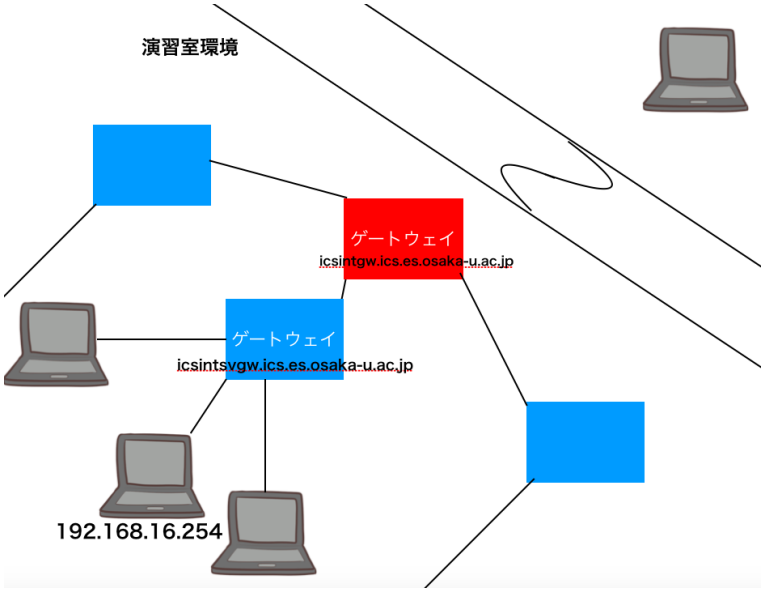
次に icsintgw.ics.es.osaka-u.ac.jp に対してコマンドを実行する

```
 traceroute to icsintgw.ics.es.osaka-u.ac.jp
 (133.1.240.81), 30 hops max, 60 byte packets
 1  192.168.16.254 (192.168.16.254)  0.355 ms  0.407 ms  0.517 ms
 2  icsintsvgw.ics.es.osaka-u.ac.jp (133.1.240.254)  0.698 ms  0.869 ms  1.025 ms
 3  icsintgw.ics.es.osaka-u.ac.jp (133.1.240.81)  1.603 ms  2.039 ms  3.016 ms
```

これは www のネットワークを経由していないので経路が暗号化されていないことがわかる。ゲートウェイの説明をする。ゲートウェイは異なるネットワーク同士を接続するネットワーク機器の事である。通常は異なるネットワークアドレス同士の通信はできない。なので他のインターネットにアクセスする場合はゲートウェイを通る。このようにアクセスするホストが違うと経路も違うことがわかる。

2.4 7

まず、課題 5,6 の結果より、自身のネットワークがゲートウェイを経由して他のインターネットと繋がっていることがわかった。図にするとこのようになる。こういった経路を通ることで他のネットワークとの通信を可能にし



ている。

2.5 8

netstat コマンドは通信中の TCP コネクション（TCP 接続）の状態を表示させるコマンドである。コマンド実行でアクティブになっている TCP 通信の状態を表示することができる。ルーティングが一致していれば対応のインターフェースへ送信する。対応するルーティングがなければデフォルトのルーターへ送信する。コマンドを実行すると結果は以下のようになる。

カーネル IP 経路テーブル
受信先サイト ゲートウェイ ネットマスク フラグ MSS Window irtt インタフェース

```
default      192.168.16.254  0.0.0.0      UG     0 0        0 ens192
link-local   *               255.255.0.0  U      0 0        0 ens192
192.168.16.0 *               255.255.255.0   U      0 0        0 ens192
```

この結果であれば link-local,192.168.16.10 の宛先のパケットはゲートウェイが設定されていないのでネットワーク内のホストと直接通信することができる。なのでそれ以外の宛先のパケットはゲートウェイ 192.168.16.254 を経由して通信が行われる。

2.6 9

前回の情報がこれで

```
? (192.168.16.254) at 14:18:77:10:31:aa [ether] on ens192
svm-01.exp.ics.es.osaka-u.ac.jp (192.168.16.241) at 02:a0:98:c4:b2:cf [ether] on ens192
exp101.exp.ics.es.osaka-u.ac.jp (192.168.16.65) at 00:50:56:b7:10:4e [ether] on ens192
exp068.exp.ics.es.osaka-u.ac.jp (192.168.16.45) at 00:50:56:b7:3c:16 [ether] on ens192
exp100.exp.ics.es.osaka-u.ac.jp (192.168.16.61) at 00:50:56:b7:22:1a [ether] on ens192
cups.exp.ics.es.osaka-u.ac.jp (192.168.16.253) at 00:50:56:b7:5b:4b [ether] on ens192
dhcp-01.exp.ics.es.osaka-u.ac.jp (192.168.16.240) at 00:50:56:b7:21:6e [ether] on ens192
```

現在の状態がこれとなっている。

```
svm-01.exp.ics.es.osaka-u.ac.jp (192.168.16.241) at 02:a0:98:c4:b2:cf [ether] on ens19
dhcp-01.exp.ics.es.osaka-u.ac.jp (192.168.16.240) at 00:50:56:b7:21:6e [ether] on ens1
exp101.exp.ics.es.osaka-u.ac.jp (192.168.16.65) at 00:50:56:b7:10:4e [ether] on ens192
? (192.168.16.254) at 14:18:77:10:31:aa [ether] on ens192
cups.exp.ics.es.osaka-u.ac.jp (192.168.16.253) at 00:50:56:b7:5b:4b [ether] on ens192
```

exp100 などの疎通を確認したテーブルが削除されている。arp テーブルに保存されている情報はある時間経つと消えることになっている。なので exp100 などの ping を送信したホストの情報は削除されている。削除されるまでの時間は OS によって異なっている。

2.7 10

システムコールは OS の機能呼び出しとなる。システムコール write などは基本的に呼び出しが遅くなる。標準ライブラリ fwrite などの方が呼び出し速度は早くなる。標準ライブラリは途中でバッファ処理が入り、メモリ消費量を抑えながらそれなりに早くシステムが早く進むように書かれている。さらに巨大なバッファを用いていなくてもバッファ処理が入るためにバッファがなくてもそれなりに早くシステムが動くようになっている。

システムコールの欠点

- システムコールの利点
  - 理解することでカーネルと OS との関係性が理解できるようになる。
- システムコールの欠点
  - 基本的に呼び出しが遅い
  - 扱いにくいものが多い
- 標準ライブラリの利点
  - バッファ処理が入るためにメモリ消費が抑えられる
  - 呼び出しが早く使いやすい

- 標準ライブラリの欠点
  - 中で動いているシステムが煩雑

のようになっている、なので普段使う分には標準ライブラリの方が使いやすくて早い。しかしシステムコールを理解することでどうやって標準ライブラリが実行されているのかがわかるのでもっと早いプログラムを追求したい場合に理解し使うことが求められるようだ。

2.8 11

プロセスが呼び出すシステムコールをトレースし、その内容を表示することができる。c オプションをつけることで統計情報を表示する。今回では echo hello を呼び出した時にどのシステムコールが何回呼び出されたのかがわかる。表示結果は

```
  hello
% time  seconds  usecs/call  calls  syscall   errors
------ ----------- ----------- --------- --------- ----------------
  0.00 0.000000        0      1        read
  0.00 0.000000        0      1        write
  0.00 0.000000        0      3        open
  0.00 0.000000        0      5        close
  0.00 0.000000        0      4        fstat
  0.00 0.000000        0      8        mmap
  0.00 0.000000        0      4        mprotect
  0.00 0.000000        0      1        munmap
  0.00 0.000000        0      3        brk
  0.00 0.000000        0      3      3 access
  0.00 0.000000        0      1        execve
  0.00 0.000000        0      1        arch_prctl
------ ----------- ----------- --------- --------- ----------------
100.00 0.000000                  35      3 total
```

のようになっている。システムコールごとに何回呼び出されたのか、そしてそれにかかった合計時間はどの程度なのかを出力している。

```
  % time  seconds  usecs/call  calls errors syscall
------ ----------- ----------- --------- --------- ----------------
  0.00 0.000000        0      7        read
  0.00 0.000000        0      3        write
  0.00 0.000000        0     10        open
```

```
  0.00 0.000000         0        12         close
  0.00 0.000000         0        11         fstat
  0.00 0.000000         0        20         mmap
  0.00 0.000000         0        12         mprotect
  0.00 0.000000         0         1         munmap
  0.00 0.000000         0         3         brk
  0.00 0.000000         0         2         rt_sigaction
  0.00 0.000000         0         1         rt_sigprocmask
  0.00 0.000000         0         2         ioctl
  0.00 0.000000         0         7       7 access
  0.00 0.000000         0         1         execve
  0.00 0.000000         0         2         getdents
  0.00 0.000000         0         1         getrlimit
  0.00 0.000000         0         2       2 statfs
  0.00 0.000000         0         1         arch_prctl
  0.00 0.000000         0         1         futex
  0.00 0.000000         0         1         set_tid_address
  0.00 0.000000         0         1         set_robust_list
------ ----------- ----------- --------- --------- ----------------
100.00 0.000000                  101       9 total
```

この結果は ls コマンドを実行した時のものになる。ls コマンドの方がシステムの呼び出しの種類、回数ともに多いことがわかる。

一つのコマンドを実行するだけでシステムコールが呼ばれる回数が爆発的に増えることがわかる。なので一つ一つの標準ライブラリで呼び出されるシステムコールの回数がプログラムの動作に大きな影響を与えることがわかる。