情報科学演習C

課題1

【担当教員】　内山　彰　教員

【提出者】　内藤　圭吾　(09B14049)
ソフトウェア科学コース・3年
u434666c@ics.osaka-u.ac.jp
【提出日】　2016年4月28日

1　課題 1-1

1.1

ping コマンドは引数として渡されたホスト名または IP アドレスとのパケットの送受信ができるかどうかを調べることができるコマンドである。演習室の PC で実行したところ以下の結果が得られた。

```
$ ping exp101
PING exp101.exp.ics.es.osaka-u.ac.jp (192.168.16.101): 56 data bytes
ping: sendto: Host is down
ping: sendto: Host is down
--- exp101.exp.ics.es.osaka-u.ac.jp ping statistics ---
7 packets transmitted, 0 packets received, 100.0% packet loss


$ ping exp101
PING exp201.exp.ics.es.osaka-u.ac.jp (192.168.16.201): 56 data bytes
64 bytes from 192.168.16.201: icmp_seq=0 ttl=64 time=1.012 ms
64 bytes from 192.168.16.201: icmp_seq=1 ttl=64 time=0.402 ms
64 bytes from 192.168.16.201: icmp_seq=2 ttl=64 time=0.647 ms
64 bytes from 192.168.16.201: icmp_seq=3 ttl=64 time=0.431 ms
64 bytes from 192.168.16.201: icmp_seq=4 ttl=64 time=0.415 ms
^C
--- exp201.exp.ics.es.osaka-u.ac.jp ping statistics ---
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.402/0.581/1.012/0.233 ms
```

ping の引数として exp101 を指定したとき、”Host is down ”と帰ってきているので、指定したホストが現在パケットの送受信ができないことを示している。また、引数として exp201 を指定したときはパケットが送信されていることが分かる。戻り値の各値は次の意味を持つ。

- icmp_seq　何秒に ICMP echo を送信したか。
- ttl　Time To Live の略称で、パケットの生存時間。
- time　ICMP echo を送信してから、応答があるまでにかかった時間。

止めるまで永遠に送信し続けるので、途中で Ctrl-C で中断する必要がある。その後の文章を見るとパケットを 5 つ送信し、その全てに応答があり、ロスしたパケットは存在しないことが分かる。

1.2

与えられた 2 つの URL を開くと以下のように同じ web サイトが開く。このことから、web サイトにはドメイン名と IP アドレスいずれでもアクセスすることができることが分かる。

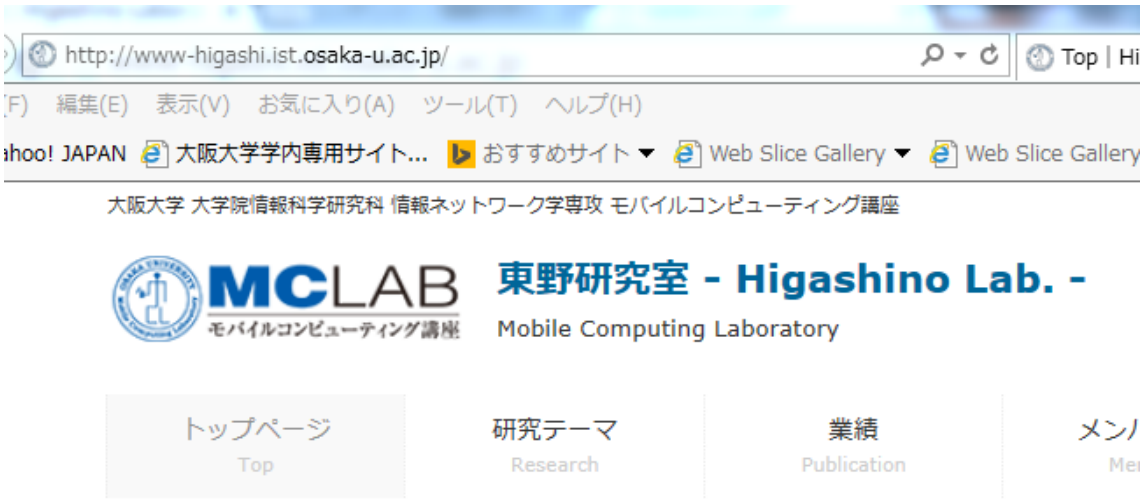

図 1: ドメイン名によるアクセス

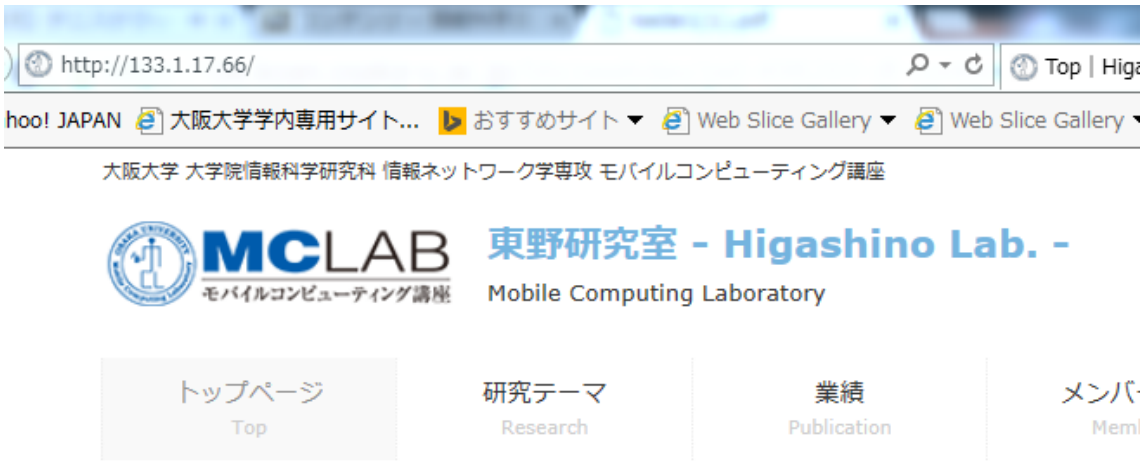

図 2: IP アドレスによるアクセス

1.3

nslookup はドメインの情報を DNS サーバ (Domain Name System　サーバー) に問い合わせて表示するコマンドである。引数としてドメイン名を指定すると、そこから IP アドレスを調べることができ、逆に IP アドレスからドメイン名を調べることもできる。

```
exp183[2]% nslookup www-ise4.sst.osaka-u.ac.jp
Server:         133.1.240.242
Address:        133.1.240.242#53

Non-authoritative answer:
Name:   www-ise4.ist.osaka-u.ac.jp
Address: 133.1.16.2
```

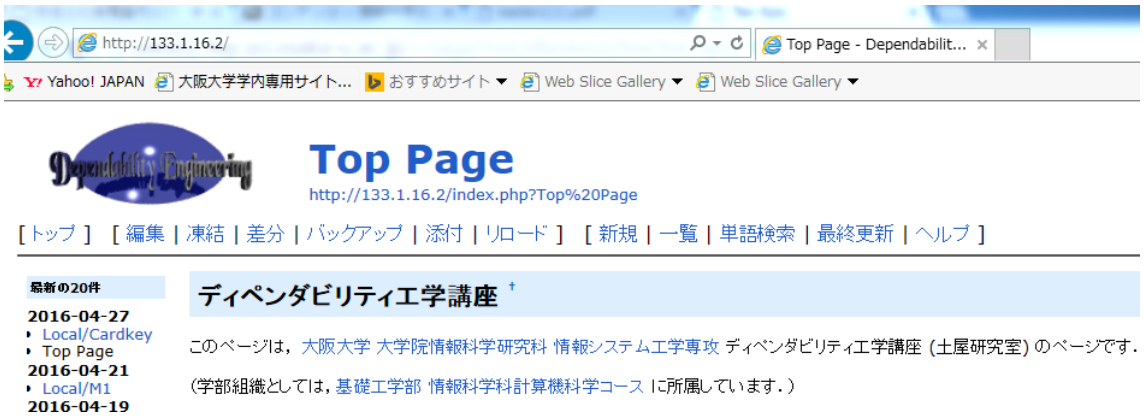

図 3: IP アドレスによるアクセス

ここで得られた IP アドレス ”133.1.16.2 ” にアクセスすると確かに土屋研究室の web サイトに飛ぶことが確認できる。逆に IP アドレスを引数として実行すると、以下の実行結果が得られる。

```
exp183[1]% nslookup 133.1.16.2
Server:         133.1.240.242
Address:        133.1.240.242#53

Non-authoritative answer:
2.16.1.133.in-addr.arpa name = cezanne.ics.es.osaka-u.ac.jp.

Authoritative answers can be found from:
16.1.133.in-addr.arpa   nameserver = ns2.ist.osaka-u.ac.jp.
16.1.133.in-addr.arpa   nameserver = ns1.ist.osaka-u.ac.jp.
16.1.133.in-addr.arpa   nameserver = ns3.ist.osaka-u.ac.jp.
ns1.ist.osaka-u.ac.jp   internet address = 133.1.173.34
ns2.ist.osaka-u.ac.jp   internet address = 133.1.173.39
ns3.ist.osaka-u.ac.jp   internet address = 133.1.138.166
```

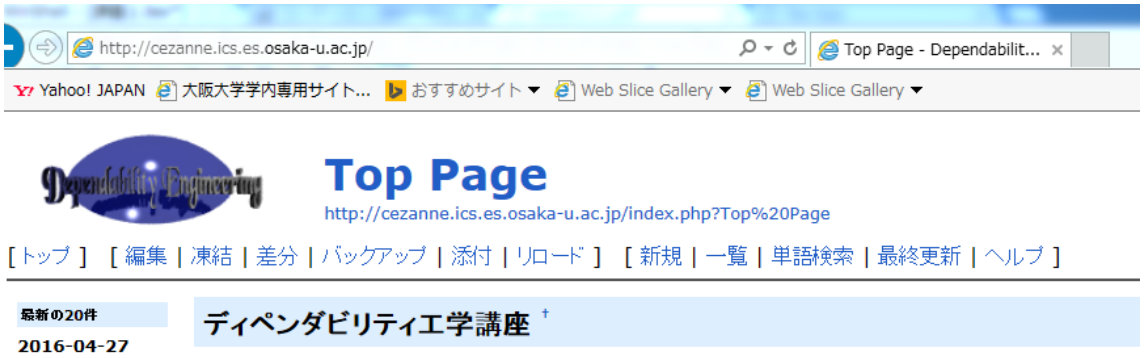

図 4: ドメイン名によるアクセス

ここで得られたドメイン名 ”cezanne.ics.es.osaka-u.ac.jp ” は課題のドメイン名をは異なるが、実際にアクセスすると、前述のドメイン名と同様に土屋研究室の web サイトにアクセスできることが確認できた。一つの IP アドレスに複数のドメイン名が設定されているためこの様な異なるドメイン名が表示された。

以上のことより、1 では出力に現れた数字の列は IP アドレスであり、2 のように、ホスト名、IP アドレスいずれからでもそのホストにアクセスできることが分かる。

2 課題 1-2

2.1

arp コマンドは ARP テーブルの表示及び設定を行うコマンドである。今回行ったオプション-a は ARP テーブルの表示を行うコマンドである。ARP テーブルとは ARP キャッシュの保存されているテーブルである。ARP は、IP アドレスから Ethernet の MAC アドレスの情報を得られるプロトコルである。実行すると以下の結果が得られた。

```
exp088[1]%/usr/sbin/arp -a
exp088.exp.ics.es.osaka-u.ac.jp (192.168.16.88) at 08:00:27:c4:a5:f7 on em0 permanent [
    ethernet]
dbs.exp.ics.es.osaka-u.ac.jp (192.168.16.252) at 00:21:28:ca:95:0a on em0 expires in 1200
    seconds [ethernet]
cups.exp.ics.es.osaka-u.ac.jp (192.168.16.253) at 00:21:28:ca:95:0a on em0 expires in 953
    seconds [ethernet]
expgw.exp.ics.es.osaka-u.ac.jp (192.168.16.254) at 5c:dd:70:0c:22:7d on em0 expires in 659
    seconds [ethernet]
exp201.exp.ics.es.osaka-u.ac.jp (192.168.16.201) at 08:00:27:e5:94:ae on em0 expires in 861
    seconds [ethernet]
ldap1.exp.ics.es.osaka-u.ac.jp (192.168.16.234) at 00:21:28:ca:95:0a on em0 expires in 952
    seconds [ethernet]
ldap2.exp.ics.es.osaka-u.ac.jp (192.168.16.235) at 00:21:28:ca:8c:da on em0 expires in 1019
    seconds [ethernet]
\
```

2.2

ping を実行した後の出力は以下の用になった。

```
Script started on Thu Apr 21 11:36:22 2016
exp183[1]% /sbin/ping exp201
PING exp201.exp.ics.es.osaka-u.ac.jp (192.168.16.201): 56 data bytes
64 bytes from 192.168.16.201: icmp_seq=0 ttl=64 time=0.882 ms
64 bytes from 192.168.16.201: icmp_seq=1 ttl=64 time=0.693 ms
64 bytes from 192.168.16.201: icmp_seq=2 ttl=64 time=0.499 ms
64 bytes from 192.168.16.201: icmp_seq=3 ttl=64 time=0.293 ms
^C
--- exp201.exp.ics.es.osaka-u.ac.jp ping statistics ---
4 packets transmitted, 4 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.293/0.592/0.882/0.219 ms
exp183[2]% /sbin/ping exp201
PING exp203.exp.ics.es.osaka-u.ac.jp (192.168.16.203): 56 data bytes
64 bytes from 192.168.16.203: icmp_seq=0 ttl=64 time=0.866 ms
64 bytes from 192.168.16.203: icmp_seq=1 ttl=64 time=0.669 ms
64 bytes from 192.168.16.203: icmp_seq=2 ttl=64 time=1.451 ms
^C
--- exp203.exp.ics.es.osaka-u.ac.jp ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.669/0.995/1.451/0.332 ms
exp183[3]% /sbin/ping exp203
PING exp209.exp.ics.es.osaka-u.ac.jp (192.168.16.209): 56 data bytes
64 bytes from 192.168.16.209: icmp_seq=0 ttl=64 time=1.591 ms
64 bytes from 192.168.16.209: icmp_seq=1 ttl=64 time=0.455 ms
64 bytes from 192.168.16.209: icmp_seq=2 ttl=64 time=0.313 ms
^C
--- exp209.exp.ics.es.osaka-u.ac.jp ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.313/0.786/1.591/0.572 ms
exp183[4]% /usr/sbin/arp -a
dbs.exp.ics.es.osaka-u.ac.jp (192.168.16.252) at 00:21:28:ca:95:0a on em0 expires in 1200
    seconds [ethernet]
smb.exp.ics.es.osaka-u.ac.jp (192.168.16.253) at 00:21:28:ca:95:0a on em0 expires in 1010
    seconds [ethernet]
```

```
expgw.exp.ics.es.osaka-u.ac.jp (192.168.16.254) at 5c:dd:70:0c:22:7d on em0 expires in 696
    seconds [ethernet]
exp209.exp.ics.es.osaka-u.ac.jp (192.168.16.209) at 08:00:27:59:e5:1b on em0 expires in 1186
    seconds [ethernet]
exp183.exp.ics.es.osaka-u.ac.jp (192.168.16.183) at 08:00:27:1c:34:b1 on em0 permanent [
    ethernet]
exp201.exp.ics.es.osaka-u.ac.jp (192.168.16.201) at 08:00:27:2f:04:c0 on em0 expires in 1175
    seconds [ethernet]
ldap1.exp.ics.es.osaka-u.ac.jp (192.168.16.234) at 00:21:28:ca:95:0a on em0 expires in 1009
    seconds [ethernet]
exp203.exp.ics.es.osaka-u.ac.jp (192.168.16.203) at 08:00:27:e4:28:4f on em0 expires in 1181
    seconds [ethernet]
ldap2.exp.ics.es.osaka-u.ac.jp (192.168.16.235) at 00:21:28:ca:8c:da on em0 expires in 1075
    seconds [ethernet]

\
```

出力の意味は前述の通りである。
ping を実行すると、ARP テーブルに ping 実行の際に指定したホストが追加されていることが分かる。これは、ping は指定したホストに対して、ICMP　echo を送信しているので、ホストとの間で通信が行われ、ARP テーブルにそのホストが追加されたためである。

2.3

traceroute は指定したホストまでの経路を表示するコマンドである。実行結果は以下の様になった。

```
exp183[1]% /usr/sbin/traceroute exp201
traceroute to exp201.exp.ics.es.osaka-u.ac.jp (192.168.16.201), 64 hops max, 40 byte packets
 1  exp201 (192.168.16.201)  0.354 ms  0.399 ms  0.224 ms
exp183[2]% /usr/sbin/traceroute exp201 www.ics.es.osaka-u.ac.jp
traceroute to vm04.ics.es.osaka-u.ac.jp (133.1.240.14), 64 hops max, 40 byte packets
 1  * * *
 2  icsintsvgw.ics.es.osaka-u.ac.jp (133.1.240.254)  0.845 ms  0.735 ms  0.604 ms
 3  icsintgw.ics.es.osaka-u.ac.jp (133.1.240.81)  0.548 ms  0.387 ms  0.2
```

traceroute は ttl の値を 1 から始め、1 つづつ増やしながら送信することで、経路全てのホストから, パケットが消滅した際に発生する、ICMP　TIME　EXCEEDED というパケットを受け取ることで全経路を表示する。演習室内のホストに対して実行した場合は、直接アクセスできるため、途中のホストは表示されないが、www.ics.es.osaka-u.ac.jp は web サイトであるので、アクセス経路に複数の存在する。出力は左から、ドメイン名、IP アドレス、3 回送信した際の往復に掛かった時間、となっている。

2.4

演習室の学生が使う PC はすべて同じサーバー内に存在している仮想計算機である。

2.5

netstat は、ホストのネットワーク接続状態や、ネットワーク統計などを確認するためのコマンドである。演習室で実行すると、以下の実行結果が得られた。

```
exp183[1]% /ust/bin/netstat -r
Routing tables

Internet:
Destination        Gateway            Flags      Netif Expire
default            expgw              UGS         em0
```

```
localhost          link#2             UH          lo0
192.168.16.0       link#1             U           em0
exp183             link#1             UHS         lo0

Internet6:
Destination        Gateway            Flags      Netif Expire
::                 localhost          UGRS        lo0
localhost          link#2             UH          lo0
::ffff:0.0.0.0     localhost          UGRS        lo0
fe80::             localhost          UGRS        lo0
fe80::%lo0         link#2             U           lo0
fe80::1%lo0        link#2             UHS         lo0
ff01::%lo0         localhost          U           lo0
ff02::             localhost          UGRS        lo0
ff02::%lo0         localhost          U           lo0
```

Destination は宛先の IP アドレス、Gateway はゲートウェイとなっているホスト名、Netif はインターフェース、 Expire はその経路の有効期限、Flags はその経路の特性で、各記号は以下の意味を持つ。

- U : 有効
- H : ホスト
- G : ゲートウェイ
- R : 回復される動的な経路
- S : 手動で追加された経路

また、::はその間のアドレスは全て 0 であることを示す。インターフェースの em0 は FreeBSD で用いられるインターフェース、lo0 はループバックアドレスは自分自身を指す。また localhost も自分自身を指している。このことから、出力を確認すると、自分自身以外へのネットワークは default と、192.168.16.0 だけであることが分かる。default はローカルシステムからリモートシステムへアクセスする際、既知のパス全てが使えない場合に、外部へ直接接続されているものである。よって、外部へのネットワークは 192.168.16.0 だけである。自分の想像したネットワーク構成との関係性として、唯一の外部へのネットワークである、192.168.16.0 が演習室の仮想計算機を構成しているサーバであることが考えられる。

2.6

時間をおいて再度 arp を実行したところ以下の実行結果が得られた。

```
exp183[1]% /usr/sbin/arp -a
dbs.exp.ics.es.osaka-u.ac.jp (192.168.16.252) at 00:21:28:ca:95:0a on em0 expires in 1152
    seconds [ethernet]
expserv.exp.ics.es.osaka-u.ac.jp (192.168.16.253) at 00:21:28:ca:95:0a on em0 expires in 1200
    seconds [ethernet]
expgw.exp.ics.es.osaka-u.ac.jp (192.168.16.254) at 5c:dd:70:0c:22:7d on em0 expires in 1020
    seconds [ethernet]
exp183.exp.ics.es.osaka-u.ac.jp (192.168.16.183) at 08:00:27:1c:34:b1 on em0 permanent [
    ethernet]
ldap1.exp.ics.es.osaka-u.ac.jp (192.168.16.234) at 00:21:28:ca:95:0a on em0 expires in 1030
    seconds [ethernet]
ldap2.exp.ics.es.osaka-u.ac.jp (192.168.16.235) at 00:21:28:ca:8c:da on em0 expires in 1096
    seconds [ethernet]
```

以前実行したときと比べると、ping を実行することで追加された、exp201 などが消えていることが分かる。これは、動的に得られた ARP 情報は一定時間の後に消えるからである。もし、時間経過を待たずに動的に得られた ARP 情報を削除したい場合は、オプション-d で arp を実行することで、即座に削除することができる。

3 課題 1-3

3.1

システムコールとは、非特権モードで動作するカーネル以外のプログラムが、特権モードで動作するカーネルに対して、仕事を依頼する方法である。システムコールの利点としては、特権モードで動作しているカーネルに仕事を依頼するので、他のプログラムに問題を与えない限り、非特権モードでは実行できない処理を実行できる点にある。デメリットとしては、プログラムからシステムコールを使うと、カーネルに処理を依頼し、その結果が帰って来るまでプログラムが止まるので、プログラムの処理が遅くなる点にある。ライブラリでは、システムコールを呼ぶことで処理時間があまり遅延しないように、データをキャッシュしたり、処理順をいれかえるなのでの工夫がしてあるが、自作のプログラムではそのような処理を行うことは煩雑であるため、システムコールを直接呼び出すことなくライブラリをそのまま使うことになる。

3.2

truss コマンドは、引数として渡したプログラムが実行するシステムコールや、受け取ったシグナルなどを表示するコマンドである。echo、pwd、ls を引数に truss -c を実行すると、以下の出力結果得る。

```
exp183[1]% truss -c echo hello
hello
syscall                     seconds   calls  errors
readlink                0.000023187       1       1
lseek                   0.000011734       1       0
mmap                    0.000235784      11       0
close                   0.000060901       3       0
fstat                   0.000058108       2       0
lstat                   0.000165663       6       1
access                  0.000066768       1       0
sigprocmask             0.000289422      12       0
munmap                  0.000113423       6       0
read                    0.000358146       2       0
                      ------------ ------- -------
                        0.001383136      45       2
exp183[2]% truss -c echo pwd
/.amd_mnt/home/exp/exp4/k-naitou
syscall                     seconds   calls  errors
readlink                0.000021511       1       1
lseek                   0.000011454       1       0
mmap                    0.000190529      11       0
close                   0.000053079       3       0
fstat                   0.000059784       3       0
lstat                   0.000148344       6       1
write                   0.000015365       1       0
ioctl                   0.000012572       1       0
access                  0.000018438       1       0
sigprocmask             0.000449778      12       0
__getcwd                0.000032965       1       0
munmap                  0.000093029       5       0
read                    0.000391950       2       0
                      ------------ ------- -------
                        0.001498798      48       2
exp183[3]% truss -c ls
#EXreport.txt#                 hatten2.c
#half_adder.vhd#               hatten2.txt
#hatten2.c#                    hatten2.txt~
#kadai2.txt#                   kadai1.txt
#proD#                         kadai2.txt
Adder.cr.mti                   kadai3-2.txt
Adder.mpf                      kadai3.txt
DFF.vhd                        kadai4.txt
DFF.vhd.bak                    modelsim
Desktop                        multi.vhd
```

```
Downloads                       multi.vhd.bak
Dratch.vhd                      pro-c2014
Dratch.vhd.bak                  pro-c2015
EXreport.txt                    proD
EXreport.txt~                   reportpage.pdf
FF.cr.mti                       result.html
FF.mpf                          sml-mode-4.0
JKFF.vhd                        sml-mode.tar.gz
JKFF.vhd.bak                    system2
ProC.pdf                        test.sml
bin                             test_full_adder.vhd
expB                            test_full_adder.vhd.bak
first.sml                       transcript
first.sml~                      typescript
hatten2                         updowncount.cr.mti
hatten2-2                       updowncount.mpf
hatten2-2.c                     updowncount.vhd
hatten2-3                       updowncount.vhd.bak
hatten2-3.c                     vsim.wlf
hatten2-3.c~                    work
hatten2-4                       workspace
hatten2-4.c 若絵                   ?????????df
hatten2-4.c~
syscall                 seconds   calls  errors
getuid                0.000009498       1       0
readlink              0.000019835       1       1
lseek                 0.000059783       3       0
mmap                  0.000498105      21       0
open                  0.000643376       9       0
close                 0.000367086      13       0
fstat                 0.000324343      12       0
lstat                 0.000408990       6       1
write                 0.015981639      33       0
ioctl                 0.000043023       3       0
access                0.000079898       3       0
sigprocmask           0.000595043      20       0
munmap                0.000178513       9       0
read                  0.002286883       8       0
                    ------------- ------- -------
                      0.021496015     142       2
```

左列の文字列はシステムコール、右列 second は実行時間、calls は実行回数、errors はエラー回数である。

4 発展課題

4.1

4.1.1 telnet

telnet はネットワークを介して、他のホストコンピュータへアクセスするためのコマンドである。使い方は、引数にホスト名を指定して実行するか、telnet コマンドを実行した後に、open コマンドを引数にホスト名を指定して実行するかの二通りの使い方がある。現在のセッションを終了するには close コマンド、telnet ごと終了するには quit コマンドを使用する。

4.1.2 srh

srh コマンドはネットワークで接続されたリモートホスト上でコマンドを実行するためのコマンドである。rsh　ホスト名　コマンド　の順に引数を指定することで実行することができる。このコマンドを実行するには、リモートホスト上にローカルホストと同じユーザーが存在しているか、リモートホストの側でほかのユーザーのコマンドの実行を許可している必要がある。

4.1.3 ftp

ftp コマンドは FTP(File Transfer Protocol) を利用してファイルの転送を行うコマンドである。ホスト名または IP アドレスを引数に指定して実行することでそのホストに接続し、引数なしで実行することで、ftp の内部コマンドモードに移行する。そこで、get コマンドや put コマンドを実行することでファイルの送受信ができる。ホストからの切断には close コマンド、ftp 内部コマンドモードの終了には bye、quit コマンドを使用する。

4.1.4 finger

finger コマンドはユーザー情報を表示するコマンドである。引数にユーザー名を指定することで、ユーザー情報の表示ができる。

4.1.5 talk

talk コマンドは他のユーザーと会話するためのコマンドである。引数にユーザー名を指定することによって、端末上で、他のユーザーとの会話ができる。

4.2

それぞれの実行結果は以下のようになった。

Listing 1: 改行なし、fwrite

```
Script started on Thu Apr 14 14:11:19 2016
exp088[1]\%truss -c ./hatten2

hellohellohellohellohellohellohellohellohellohellohellohellohellohellohellohellohellohellohellohellohellohello

seconds   calls  errors
readlink               0.000020952       1       1
lseek                  0.000011733       1       0
mmap                   0.000343898      11       0
close                  0.000059784       3       0
fstat                  0.000049168       3       0
lstat                  0.000143593       6       1
write                  0.000363454       1       0
ioctl                  0.000012851       1       0
access                 0.000018717       1       0
sigprocmask            0.000212036      12       0
munmap                 0.000135493       6       0
read                   0.000364012       2       0
                     ------------- ------- -------
                       0.001735691      48       2
```

Listing 2: 改行あり、fwrite

```
exp088[2]\%truss -c ./hatten2-2

hello
hello
hello
hello
hello
hello
hello
```

```
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
```

```
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
syscall                     seconds   calls  errors
readlink                0.000022070       1       1
lseek                   0.000011174       1       0
mmap                    0.000277409      11       0
close                   0.000054196       3       0
fstat                   0.000067327       3       0
lstat                   0.000150298       6       1
write                   0.043763662     100       0
ioctl                   0.000013409       1       0
access                  0.000018438       1       0
sigprocmask             0.000270428      12       0
munmap                  0.000088280       6       0
read                    0.000383010       2       0
                      ------------- ------- -------
                        0.045119701     147       2
```

Listing 3: 改行なし、write

```
exp088[3]\% truss -c ./hatten2-3

hellohellohellohellohellohellohellohellohellohellohellohellohellohellohellohellohellohellohellohellohellohello

                        seconds   calls  errors
readlink                0.000026819       1       1
lseek                   0.000014807       1       0
mmap                    0.000176000      10       0
close                   0.000053637       3       0
fstat                   0.000038551       2       0
lstat                   0.000146109       6       1
write                   0.013855122     100       0
access                  0.000018438       1       0
sigprocmask             0.000367364      12       0
munmap                  0.000086604       6       0
read                    0.000412622       2       0
                      ------------- ------- -------
                        0.015196073     144       2
```

Listing 4: 改行あり、write

```
exp088[4]\% truss -c ./hatten2-4

hello
hello
hello
hello
hello
hello
hello
hello
```

```
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
```

```
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
syscall                  seconds   calls  errors
readlink             0.000022350       1       1
lseek                0.000012013       1       0
mmap                 0.000177959      10       0
close                0.000056153       3       0
fstat                0.000051123       2       0
lstat                0.000171531       6       1
write                0.006791093     100       0
access               0.000019276       1       0
sigprocmask          0.000296407      12       0
munmap               0.000129905       6       0
read                 0.000381613       2       0
                     ------------- ------- -------
                       0.008109423     144       2
```

ここで、それぞれシステムコール呼び出しの部分だけ抜き出し、write、fwrite の違いを確認する。

Listing 5: 改行なし、fwrite

```
readlink             0.000020952       1       1
lseek                0.000011733       1       0
mmap                 0.000343898      11       0
close                0.000059784       3       0
fstat                0.000049168       3       0
lstat                0.000143593       6       1
write                0.000363454       1       0
ioctl                0.000012851       1       0
access               0.000018717       1       0
sigprocmask          0.000212036      12       0
munmap               0.000135493       6       0
read                 0.000364012       2       0
                   ------------- ------- -------
                     0.001735691      48       2
```

Listing 6: 改行なし、write

```
readlink             0.000026819       1       1
lseek                0.000014807       1       0
mmap                 0.000176000      10       0
close                0.000053637       3       0
fstat                0.000038551       2       0
lstat                0.000146109       6       1
write                0.013855122     100       0
access               0.000018438       1       0
sigprocmask          0.000367364      12       0
munmap               0.000086604       6       0
read                 0.000412622       2       0
                   ------------- ------- -------
                     0.015196073     144       2
```

Listing 7: 改行あり、fwrite

```
readlink             0.000022070       1       1
lseek                0.000011174       1       0
mmap                 0.000277409      11       0
close                0.000054196       3       0
fstat                0.000067327       3       0
lstat                0.000150298       6       1
write                0.043763662     100       0
ioctl                0.000013409       1       0
access               0.000018438       1       0
sigprocmask          0.000270428      12       0
munmap               0.000088280       6       0
read                 0.000383010       2       0
                   ------------- ------- -------
                     0.045119701     147       2
```

Listing 8: 改行あり、write

```
readlink             0.000022350       1       1
lseek                0.000012013       1       0
mmap                 0.000177959      10       0
close                0.000056153       3       0
fstat                0.000051123       2       0
lstat                0.000171531       6       1
write                0.006791093     100       0
access               0.000019276       1       0
sigprocmask          0.000296407      12       0
munmap               0.000129905       6       0
read                 0.000381613       2       0
                   ------------- ------- -------
                     0.008109423     144       2
```

比較すると、改行なしの fwrite 以外は write を 100 回呼び出しているが、改行なしの fwrite は 1 回となっ

ている。これは、fwrite がシステムコールを呼び出す回数を少なくするための工夫として、改行を待って、改行する際に write を呼び出して、一度に書き出していると考えられる。これは、前述の標準ライブラリ関数の、動作を軽くするための工夫である。

4.3 感想

linux には自分の知らないネットワークに関するコマンドが多数存在し、今回の課題を通して、コマンドに対する理解はもちろんのこと、コマンドの動作を調べる過程で、ネットワークそのものに対する理解も深まり、大変ためになる課題でした。