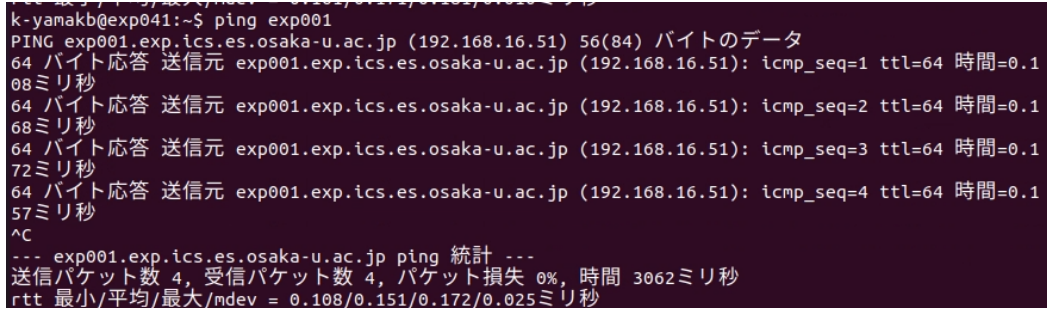


情報科学演習C 課題1レポート

氏名 山久保孝亮
所属 大阪大学基礎工学部情報科学科ソフトウェア科学コース
メールアドレス u327468b@ecs.osaka-u.ac.jp
学籍番号 09B22084
提出日 2024 年 4 月 28 日
担当教員 平井健士 中島悠太

1 課題 1-1

1. ping exp001 を実行すると以下の図 1 のような出力結果が得られる。



```
k-yamakb@exp041:~$ ping exp001
PING exp001.exp.ics.es.osaka-u.ac.jp (192.168.16.51) 56(84) バイトのデータ
64 バイト応答 送信元 exp001.exp.ics.es.osaka-u.ac.jp (192.168.16.51): icmp_seq=1 ttl=64 時間=0.108ミリ秒
64 バイト応答 送信元 exp001.exp.ics.es.osaka-u.ac.jp (192.168.16.51): icmp_seq=2 ttl=64 時間=0.168ミリ秒
64 バイト応答 送信元 exp001.exp.ics.es.osaka-u.ac.jp (192.168.16.51): icmp_seq=3 ttl=64 時間=0.172ミリ秒
64 バイト応答 送信元 exp001.exp.ics.es.osaka-u.ac.jp (192.168.16.51): icmp_seq=4 ttl=64 時間=0.157ミリ秒
^C
--- exp001.exp.ics.es.osaka-u.ac.jp ping 統計 ---
送信パケット数 4, 受信パケット数 4, パケット損失 0%, 時間 3062ミリ秒
rtt 最小/平均/最大/mdev = 0.108/0.151/0.172/0.025ミリ秒
```

図 1: ping exp001 の実行結果

ping コマンドは ECHO_REQUEST を使用してホストから ECHO_RESPONSE を引き出し、通信の状態を確認するコマンドである。ECHO_REQUEST と ECHO_RESPONSE は ICMP のメッセージである。1 行目には左から順に、宛先として指定したサーバとその IP アドレス、送信するパケットデータのサイズを表示している。56(84) は、56byte のデータに ICMP のヘッダー情報の 8byte と TCP データの 20byte が足されて合計 84byte のデータが送信されるということを表している。2 行目以降は同じ結果が繰り返し出力されている。繰り返されているそれぞれの内容は以下の通りである。

- 64 バイト応答：送信された ICMP のデータサイズ (TCP のヘッダー情報の 20byte は含まない)
- 送信元：宛先ホスト名と IP アドレス
- icmp_seq：ping を送信した回数。この回数に抜けがあればその部分でパケットロスが発生している。
- ttl：パケットの生存期間。ネットワーク上でルータなどの危機を通過するたびにその値を 1 ずつ減らしていき、この値が 0 になるとパケットは破棄される
- 時間：パケットを送信して返ってくるまでの時間

そして Ctrl+C を押して強制終了すると統計値が表示される。出力される内容は以下のとおりである。

- 送信パケット数：ping を送信した回数
- 受信パケット数：ping を送信してから宛先ホストから正常にパケットが戻ってきた回数
- パケット損失：送信パケットに対するパケットロスの割合
- 時間：ping を開始してから終了するまでにかかった時間
- rtt 最小/平均/最大/mdev：左から順に応答時間の最小値、平均値、最大値、標準誤差

2. ドメインを入力した時も IP アドレスを入力したときも以下の図 2 のような同じ大阪大学のウェブサイトが開いた. このことから, `www.osaka-u.ac.jp` というドメインは IP アドレスに変換すると `133.1.138.1`



図 2: 検索結果

となったと考えられる.

3. `nslookup osaka-u.ac.jp` を実行すると以下の図 3 のような実行結果が得られる.

```
k-yamakb@exp041:~$ nslookup www.osaka-u.ac.jp
Server:           127.0.0.53
Address:          127.0.0.53#53

Non-authoritative answer:
Name:   www.osaka-u.ac.jp
Address: 133.1.138.1
```

図 3: コマンドの実行結果

`nslookup` コマンドは二つのモードがあり, 対話モードでは様々なホストやドメインに関する情報をネームサーバーに問い合わせたり, ドメイン内のホストの一覧を表示したりすることができる. 非対話モードではホストやドメインの名前と要求された情報だけを表示する. これによりドメイン名から IP アドレスを, IP アドレスからドメイン名を知ることができる.

1. において, 宛先ホスト名を指定したときにそれに対応する IP アドレスが表示されていた. このホスト名を `nslookup` コマンドを使って IP アドレスを調べると以下の図 4 のようになる.

```
k-yamakb@exp028:~$ nslookup exp001.exp.ics.es.osaka-u.ac.jp
Server:           127.0.0.53
Address:          127.0.0.53#53

Non-authoritative answer:
Name:   exp001.exp.ics.es.osaka-u.ac.jp
Address: 192.168.16.4
```

図 4: コマンドの実行結果

この出力結果から, `Non-authoritative answer` の方のアドレスと IP アドレスが一致していることがわかる. `Non-authoritative answer` は, キャッシュ DNS から返ってきた情報であるということを表してい

る。また,2.においてドメインと IP アドレスが同じであると考察したが, 図 3 の出力結果からもそのことを確認することができる。

2 課題 1-2

4. /usr/sbin/arp -a を実行すると以下の図 5 のような出力が得られる。

```
k-yamakb@exp045:~$ /usr/sbin/arp -a
? (192.168.16.252) at cc:48:3a:f3:f8:33 [ether] on ens160
? (192.168.16.240) at 00:50:56:be:1f:d1 [ether] on ens160
dbs.exp.ics.es.osaka-u.ac.jp (192.168.16.247) at 00:50:56:be:96:59 [ether] on ens160
_gateway (192.168.16.254) at 00:00:5e:00:01:10 [ether] on ens160
? (192.168.16.253) at cc:48:3a:f3:f6:33 [ether] on ens160
? (192.168.16.250) at 00:50:56:be:1f:d1 [ether] on ens160
```

図 5: コマンドの実行結果

このコマンドによって ARP キャッシュが出力として表示される。具体的な表示内容としては、まず最初にドメイン (IP アドレス) が表示され、その後 at に続いてそれに対応する MAC アドレスが表示される。[ether] というのは ARP が IP アドレスを MAC アドレスに変換する際にイーサネットフレーム内で動作するため表示されている。ens160 というのは Linux システム上で特定のネットワークインターフェースを識別するための名前である。

5. ping を exp002 と exp003 に対して実行すると以下の図 6 のような出力が得られた。基本的な出力内容

```
k-yamakb@exp045:~$ ping exp002
PING exp002.exp.ics.es.osaka-u.ac.jp (192.168.16.40) 56(84) バイトのデータ
64 バイト応答 送信元 exp002.exp.ics.es.osaka-u.ac.jp (192.168.16.40): icmp_seq=1 ttl=64 時間=0.620ミリ秒
64 バイト応答 送信元 exp002.exp.ics.es.osaka-u.ac.jp (192.168.16.40): icmp_seq=2 ttl=64 時間=0.265ミリ秒
64 バイト応答 送信元 exp002.exp.ics.es.osaka-u.ac.jp (192.168.16.40): icmp_seq=3 ttl=64 時間=0.184ミリ秒
64 バイト応答 送信元 exp002.exp.ics.es.osaka-u.ac.jp (192.168.16.40): icmp_seq=4 ttl=64 時間=0.247ミリ秒
64 バイト応答 送信元 exp002.exp.ics.es.osaka-u.ac.jp (192.168.16.40): icmp_seq=5 ttl=64 時間=0.225ミリ秒
^C
--- exp002.exp.ics.es.osaka-u.ac.jp ping 統計 ---
送信パケット数 5, 受信パケット数 5, パケット損失 0%, 時間 4054ミリ秒
rtt 最小/平均/最大/ndev = 0.184/0.308/0.620/0.158ミリ秒
k-yamakb@exp045:~$ ping exp003
PING exp003.exp.ics.es.osaka-u.ac.jp (192.168.16.18) 56(84) バイトのデータ
64 バイト応答 送信元 exp003.exp.ics.es.osaka-u.ac.jp (192.168.16.18): icmp_seq=1 ttl=64 時間=0.522ミリ秒
64 バイト応答 送信元 exp003.exp.ics.es.osaka-u.ac.jp (192.168.16.18): icmp_seq=2 ttl=64 時間=0.243ミリ秒
64 バイト応答 送信元 exp003.exp.ics.es.osaka-u.ac.jp (192.168.16.18): icmp_seq=3 ttl=64 時間=0.309ミリ秒
64 バイト応答 送信元 exp003.exp.ics.es.osaka-u.ac.jp (192.168.16.18): icmp_seq=4 ttl=64 時間=0.232ミリ秒
64 バイト応答 送信元 exp003.exp.ics.es.osaka-u.ac.jp (192.168.16.18): icmp_seq=5 ttl=64 時間=0.316ミリ秒
^C
--- exp003.exp.ics.es.osaka-u.ac.jp ping 統計 ---
送信パケット数 5, 受信パケット数 5, パケット損失 0%, 時間 4063ミリ秒
rtt 最小/平均/最大/ndev = 0.232/0.324/0.522/0.104ミリ秒
```

図 6: コマンドの実行結果

は上の 1. で述べた内容と同じであったが、IP アドレスは異なっていた。以下の図 7 は図 6 の出力の後に /usr/sbin/arp -a を実行したときの出力である。図 5 の出力結果と比べて、exp002 と exp003 の IP

```
k-yamakb@exp045:~$ /usr/sbin/arp -a
? (192.168.16.252) at cc:48:3a:f3:f8:33 [ether] on ens160
? (192.168.16.240) at 00:50:56:be:1f:d1 [ether] on ens160
exp002.exp.ics.es.osaka-u.ac.jp (192.168.16.40) at 00:50:56:be:94:aa [ether] on ens160
dbs.exp.ics.es.osaka-u.ac.jp (192.168.16.247) at 00:50:56:be:96:59 [ether] on ens160
_gateway (192.168.16.254) at 00:00:5e:00:01:10 [ether] on ens160
dhcp1.exp.ics.es.osaka-u.ac.jp (192.168.16.251) at 00:50:56:be:5b:e9 [ether] on ens160
? (192.168.16.253) at cc:48:3a:f3:f6:33 [ether] on ens160
? (192.168.16.250) at 00:50:56:be:1f:d1 [ether] on ens160
exp003.exp.ics.es.osaka-u.ac.jp (192.168.16.18) at 00:50:56:be:71:b5 [ether] on ens160
```

図 7: コマンドの実行結果

アドレスと MAC アドレスのキャッシュが出力に追加されている。出力が変化した理由としては、ping

コマンドによりメッセージのやり取りが行われたことで ARP キャッシュの内容が変化したためであると考えられる。

6. /usr/sbin/traceroute を exp002exp003,Web サーバ, ゲートウェイに対して実行すると以下の図 8 のような出力が得られる。

```
k-yamakb@exp045:~$ /usr/sbin/traceroute exp002
traceroute to exp002 (192.168.16.40), 30 hops max, 60 byte packets
 1 exp002.exp.ics.es.osaka-u.ac.jp (192.168.16.40) 0.227 ms 0.204 ms 0.204 ms
k-yamakb@exp045:~$ /usr/sbin/traceroute exp003
traceroute to exp003 (192.168.16.18), 30 hops max, 60 byte packets
 1 exp003.exp.ics.es.osaka-u.ac.jp (192.168.16.18) 0.280 ms 0.251 ms 0.242 ms
k-yamakb@exp045:~$ /usr/sbin/traceroute www.ics.es.osaka-u.ac.jp
traceroute to www.ics.es.osaka-u.ac.jp (133.1.240.14), 30 hops max, 60 byte packets
 1 192.168.16.252 (192.168.16.252) 0.319 ms 0.257 ms 0.385 ms
 2 icsintsvgw.ics.es.osaka-u.ac.jp (133.1.240.254) 0.745 ms 0.940 ms 1.064 ms
 3 icsintgw.ics.es.osaka-u.ac.jp (133.1.240.81) 0.731 ms 1.048 ms 1.323 ms
 4 vm04.ics.es.osaka-u.ac.jp (133.1.240.14) 11.786 ms 18.926 ms 19.779 ms
k-yamakb@exp045:~$ /usr/sbin/traceroute icsintgw.ics.es.osaka-u.ac.jp
traceroute to icsintgw.ics.es.osaka-u.ac.jp (133.1.240.81), 30 hops max, 60 byte packets
 1 192.168.16.252 (192.168.16.252) 0.274 ms 0.272 ms 0.360 ms
 2 icsintsvgw.ics.es.osaka-u.ac.jp (133.1.240.254) 0.756 ms 0.909 ms 1.069 ms
 3 icsintgw.ics.es.osaka-u.ac.jp (133.1.240.81) 1.417 ms 2.719 ms 2.131 ms
```

図 8: コマンドの実行結果

図 8 の出力結果を比較すると,exp002 と exp003 に対して traceroute を実行したときは同じとなるがその二つと Web サーバ, ゲートウェイに対して traceroute を実行したときには異なる結果となることわかる.traceroute コマンドは目的の IP アドレスまでの経路を表示するコマンドであり,それぞれ目的地の IP アドレスが異なるために出力の内容が異なると考えられる。

7. ping と traceroute が正しく動作することで宛先までのネットワーク経路と, その経路まで到達することが可能かどうかを知ることができる。つまり,exp001 などのホストや Web サーバに対して先ほどの 2 つのコマンドが実行することによってネットワークの構成を予測することができる。今回推測したネットワークの構成は以下の図 9 のようになった。

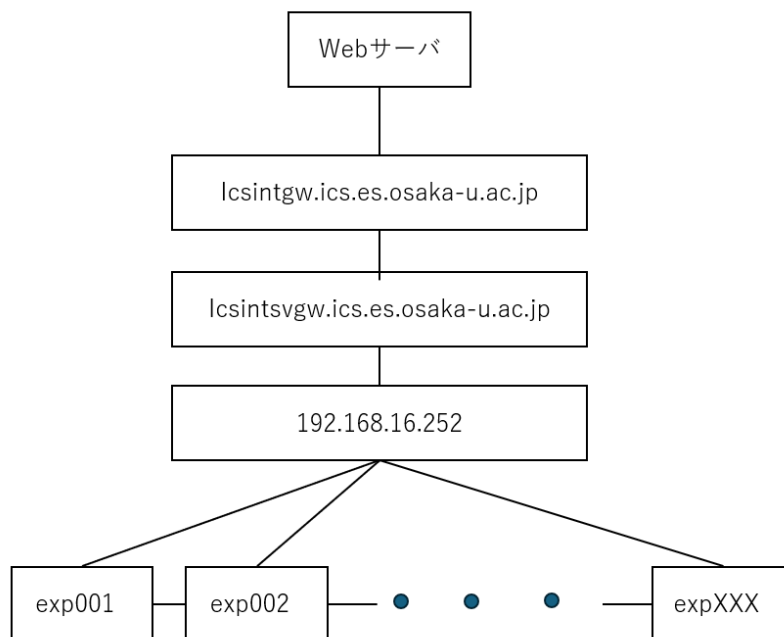


図 9: 推測したネットワークの模式図

これは、図 8 における traceroute コマンドの実行結果が、Web サーバに対して実行した際に、まず実際の IP アドレスである 192.168.16.252 を通りその後仮想 IP アドレスである 192.168.16.254 を通ってから Web のドメインを通っていたことと、それぞれのホストに対して実行したときに一度の移動で到達することができていたことから、ホスト同士がつながっていると考えたためにこのような図となった。

8. /bin/netstat -r を実行すると以下の図 7 のような実行結果が得られる。netstat は r オプションで実行

```
k-yamakb@exp045:~$ /bin/netstat -r
```

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	_gateway	0.0.0.0	UG	0	0	0	ens160
link-local	0.0.0.0	255.255.0.0	U	0	0	0	ens160
192.168.16.0	0.0.0.0	255.255.255.0	U	0	0	0	ens160

図 10: コマンドの実行結果

するとルーティングテーブルを出力する。具体的な出力内容は以下になる。

- 受信先サイト：宛先の IP アドレス。
- ゲートウェイ：ゲートウェイの IP アドレス。
- ネットマスク：サブネットマスク。
- フラグ：U は送信経路がリンクアップしていることを示し、G は送信経路がゲートウェイ宛であることを示し、H は宛先がネットワークではなく完全指定のホストアドレスであることを示している。
- MSS：Maximum Segment Size を表す。
- Window：TCP Window Size を表す。
- irtt：Initial Round Trip Time を表す。

このことから、自分が想像したネットワークは

9. 以下の図 11 は”ping exp001”を実行した直後に”/usr/sbin/arp -a”を実行したときの出力結果で、その下の図 12 は 20 分が経過した後に”/usr/sbin/arp -a”を実行したときの出力結果である。

```
k-yamakb@exp040:~$ /usr/sbin/arp -a
? (192.168.16.252) at cc:48:3a:f3:f8:33 [ether] on ens160
exp001.exp.lcs.es.osaka-u.ac.jp (192.168.16.4) at 00:50:56:be:07:dc [ether] on ens160
dhcp1.exp.lcs.es.osaka-u.ac.jp (192.168.16.251) at 00:50:56:be:5b:e9 [ether] on ens160
? (192.168.16.240) at 00:50:56:be:1f:d1 [ether] on ens160
dbs.exp.lcs.es.osaka-u.ac.jp (192.168.16.247) at 00:50:56:be:96:59 [ether] on ens160
_gateway (192.168.16.254) at 00:00:5e:00:01:10 [ether] on ens160
? (192.168.16.250) at 00:50:56:be:1f:d1 [ether] on ens160
? (192.168.16.253) at cc:48:3a:f3:f6:33 [ether] on ens160
```

図 11: コマンドの実行結果

```
k-yamakb@exp040:~$ /usr/sbin/arp -a
? (192.168.16.252) at cc:48:3a:f3:f8:33 [ether] on ens160
exp001.exp.lcs.es.osaka-u.ac.jp (192.168.16.4) at 00:50:56:be:07:dc [ether] on ens160
dhcp1.exp.lcs.es.osaka-u.ac.jp (192.168.16.251) at 00:50:56:be:5b:e9 [ether] on ens160
? (192.168.16.240) at 00:50:56:be:1f:d1 [ether] on ens160
dbs.exp.lcs.es.osaka-u.ac.jp (192.168.16.247) at 00:50:56:be:96:59 [ether] on ens160
_gateway (192.168.16.254) at 00:00:5e:00:01:10 [ether] on ens160
? (192.168.16.250) at 00:50:56:be:1f:d1 [ether] on ens160
? (192.168.16.253) at cc:48:3a:f3:f6:33 [ether] on ens160
```

図 12: コマンドの実行結果

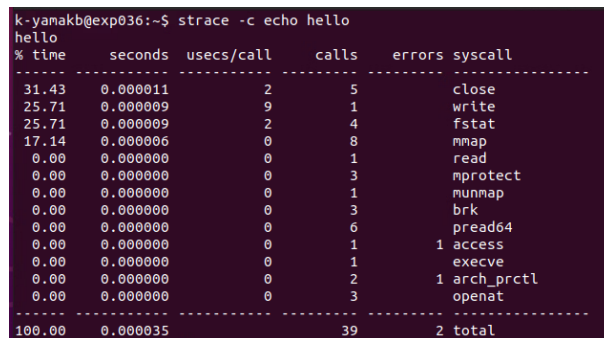
上の図 11 と 12 より、何も変化が起こらなかったことがわかる。Linux の arp キャッシュは数分経てば

キャッシュから追い出されて出力されなくなるはずであるが、何も変化が起こっていない。この理由は ARP キャッシュのタイムアウトの設定が長くなっているか、ARP リクエストがあまり行われないうためにキャッシュが更新されないことなどが考えられる。

3 課題 1-3

10. システムコールとは、カーネルとアプリケーションプログラムのインターフェースである。システムコールによってハードウェアの挙動を隠ぺいすることによってハードウェアによらずにデバイスからの入出力処理を行うことができるようになる。標準ライブラリ関数はあらかじめテストされているためコーディングの効率性と安定性が向上するという利点がある。ただし、また、標準ライブラリ関数とは、プログラミング言語の標準仕様に含まれる関数のことで、システムコールを利用するものも、利用しないものも存在する。ファイル処理に関して、システムコールは標準ライブラリ関数に比べて使いやすさや実行効率が劣ってしまうという欠点がある。しかし、ファイルの属性を変更するなどのシステムコールでしかできないような利点も存在するので、標準ライブラリ関数で `fwrite` とシステムコールの `write` のような同じような機能の仕組みが複数提供されている。ほかにもシステムコールでしかできないような例を挙げると、プログラムの実行中に新しいプロセスを生成したり、シグナルを送ったりするなどである。システムにとって重要で基幹部分に近いプログラムを作成したいときはシステムコールを使うことが必須となる。

11. "strace -c echo hello" を実行すると以下の図 13 のような実行結果が得られる。



% time	seconds	usecs/call	calls	errors	syscall
31.43	0.000011	2	5		close
25.71	0.000009	9	1		write
25.71	0.000009	2	4		fstat
17.14	0.000006	0	8		mmap
0.00	0.000000	0	1		read
0.00	0.000000	0	3		mprotect
0.00	0.000000	0	1		munmap
0.00	0.000000	0	3		brk
0.00	0.000000	0	6		pread64
0.00	0.000000	0	1	1	access
0.00	0.000000	0	1		execve
0.00	0.000000	0	2	1	arch_prctl
0.00	0.000000	0	3		openat
100.00	0.000035		39	2	total

図 13: コマンドの実行結果

`strace -c` コマンドはシステムコールの数を取得することができる。出力は以下になると考えられる。

- % time : 各システムコールがプロセスを実行した時間のパーセント
- seconds : 各システムコールが実行された合計時間
- usecs/call : 各システムコールの平均実行時間
- calls : 各システムコールが実行された回数
- errors : エラーが発生した場合のエラーコード
- syscall : 実行されたシステムコールの名前

以下の図 14 は `strace -c ls` を実行した結果である。

```
k-yamakb@exp031:~$ strace -c ls
```

```
09822084  pro-c2023
Desktop    q.log
Documents  result.html
Downloads  sjis-dos.txt
Music       sjis-dos.txt.bak
Pictures   'tall <deb name>'
Public     test
Templates  test.bak
Videos     test.txt
a.txt      test.txt.bak
b.txt      tex
bar        texput.log
bin        texput.log.bak
enshu      'truct timeval and then an arbitrary number of "pad" bytes'
ex5-1      tsclient
ex5-2      wer:
ex7-1      zikkenA
```

file_utf8.txt	% time	seconds	usecs/call	calls	errors	syscall
	0.00	0.000000	0	7		read
	0.00	0.000000	0	10		write
	0.00	0.000000	0	12		close
	0.00	0.000000	0	11		fstat
	0.00	0.000000	0	28		mmap
	0.00	0.000000	0	8		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	8		pread64
	0.00	0.000000	0	2	2	access
	0.00	0.000000	0	1		execve
	0.00	0.000000	0	2	2	statfs
	0.00	0.000000	0	2	1	arch_prctl
	0.00	0.000000	0	1		futex
	0.00	0.000000	0	2		getdents64
	0.00	0.000000	0	1		set_tid_address
	0.00	0.000000	0	10		openat
	0.00	0.000000	0	1		set_robust_list
	0.00	0.000000	0	1		prlimit64
	100.00	0.000000		124	5	total

図 14: コマンドの実行結果

echo の時と比較して違う点は、システムコールの実行時間がほぼ全てゼロであるという点である。

4 感想

今回の課題を通して、コマンドを使用してネットワーク上での通信の情報を取得したり、IP アドレスとドメインの関係、MAC アドレス等について理解を深めることができた。ネットワーク関連の話題は学習意欲はあったがこれまで触れてきていなかったの、今後の課題においても知識を身につける場であると考えて積極的に学習を進めていきたいと思う。また、システムコールについては、現在オペレーティングシステムの授業でカーネル等について学んでいるので、どこかで登場する機会がある考えられる知識を予め知ることができてよかったと感じた。

5 謝辞

今回の課題を通して質問対応、レポート採点等をしてくださった教授、TA の皆様方ありがとうございました。今後の課題もよろしくお願いいたします。

6 参考文献

- <https://www.server-memo.net/tips/command/ping/ping.html#toc2> 4/28
- <https://win2012r2.com/2022/07/04/dns-non-authoritative-answer/> 4/28
- <https://www.oresamalabo.net/entry/2020/05/11/180603> 4/28
- <https://curtaincall.weblike.jp/portfolio-unix/api.html> 4/28

- <https://engineer-oasis.com/898/#> 4/28
- <https://relax-tech.net/linux-strace-debug/#rtoc-7> 4/28