# CSCI-4220 Network Programming Project #1

Shigeru Imai (imais@rpi.edu)

February 28, 2014

## 1 Compilation (20 points)

Base compilation command:
```
$ gcc -Wall proj1test.c layer2.c layer3.c layer4.c layer5.c
```

`proj1test.c` is taken from the project description web page. If some compile instructions are provided in README, I follow that. Otherwise, I use the above command.

Points are computed as follows:

- No errors (10 points), one type of error (-5 points), two or more types of errors (-10 points)
- No warnings (10 points), one type of warning (-5 points), two or more types of warnings (-10 points)

Even if there are multiple errors/warnings, if they are essentially the same, I count them as one type of error/warning. For example, there are two cast-related warnings below, but I regard them as one type of warning. So the penalty is 5 points in this case.

```
layer2.c:  In function layer2_read:
layer2.c:15:13:  warning:  initialization makes integer from pointer without a cast
layer2.c:  In function layer2_write:
layer2.c:39:13:  warning:  initialization makes integer from pointer without a cast
```

## 2 Testing Layers (60 points)

Submitted `layer2.c`, `layer3.c`, `layer4.c`, and `layer5.c` are tested layer by layer. Each layer is worth 15 points. The files used for testing, `layer2test.c`, `layer3test.c`, `layer4test.c`, `layer5test.c`, `debug.c`, and `layer1.c`, are available on the course website. When testing layer X, 5 base points are given if compilable `layerX_read` and `layerX_write` are implemented.

(a) **Layer 2** (15 points)

Compilation command:
```
$ gcc -Wall layer2test.c debug.c layer2.c layer1.c
```

Test case:
```
$ ./a.out "0123456789ABCDEF" | ./a.out
```

Expected output:
```
layer2_test:  0123456789ABCDEF (16 bytes)
```

Points breakdown:
Base points (5 points), sender working correctly (5 points), receiver working correctly (5 points)

(b) **Layer 3** (15 points)
Compilation command:
```
$ gcc -Wall layer3test.c debug.c layer3.c layer2.c layer1.c
```

Test case:
```
$ ./a.out "0123456789ABCDEF0123456789abcdefghijklmnop" | ./a.out
```

Expected output:
```
layer3_test:  0123456789ABCDEF0123456789abcdefghijklmnop (42 bytes)
```

Points breakdown:
Base points (5 points), sender working correctly (5 points), receiver working correctly (5 points)

(c) **Layer 4** (15 points)
Compilation command:
```
$ gcc -Wall layer4test.c debug.c layer4.c layer3.c layer2.c layer1.c
```

Test case:
```
$ ./a.out "0123456789ABCDEF012" | ./a.out
```

Expected output:
```
layer4_test:  0123456789ABCDEF012 (19 bytes)
```

Points breakdown:
Base points (5 points), sender working correctly (5 points), receiver working correctly (5 points)

(d) **Layer 5** (15 points)
Compilation command:
```
$ gcc -Wall layer5test.c debug.c layer5.c layer4.c layer3.c layer2.c layer1.c
```

Test case:
```
$ ./a.out "Stephen van" Rensselaer 660000002 3.88888 | ./a.out
```

Expected output:
```
Name:  Stephen van Rensselaer
RIN: 660000002
GPA: 3.889
```

Points breakdown:
Base points (5 points), sender working correctly (5 points), receiver working correctly (5 points)

## 3 Testing Error Cases (20 points)

(a) **Error Propagation** (10 points)
`layer1_read()` is configured to always return an error. The error generated by `layer1_read()` should be propagated to the upper layers.

Compilation command:
```
$ gcc -Wall -DERROR_TEST1 layer5test.c debug.c layer5.c layer4.c layer3.c layer2.c
layer1.c
```

Test case:
```
$ ./a.out "Stephen van" Rensselaer 660000002 3.88888 | ./a.out
```

Expected output:
```
Reading error
```

(b) **Checksum Error Detection** (10 points)
`layer1_read()` is configured to change `Rensselaer` to `Rennselaer` (the first `'s'` becomes `'n'`). A checksum error should be detected at layer 4.

Compilation command:
```
$ gcc -Wall -DERROR_TEST2 layer5test.c debug.c layer5.c layer4.c layer3.c layer2.c
layer1.c
```

Test case:
```
$ ./a.out "Stephen van" Rensselaer 660000002 3.88888 | ./a.out
```

Expected output:
```
Reading error
```

## Notes

- If you are not satisfied with your grade, please run your program with the test files or try equivalent test cases. If you think that I made a mistake in grading, come to my office hour or send me an email.
- There are some students whose submitted files are not easily compilable with the test files. In that case, I slightly modify their code to make it work. If you are not sure how to test your code, please let me know.