Josh Kunz
April 1, 2014

Musl `snprintf` Code Coverage

1 `gcov` Coverage

```
File '../musl-printf-standalone/vfprintf.c'
Lines executed:96.99% of 365
Branches executed:99.51% of 408
Taken at least once:92.65% of 408
Calls executed:96.77% of 62
../musl-printf-standalone/vfprintf.c:creating 'vfprintf.c.gcov'
```

2 Un-coverable Sections

This section details the parts of the code under test that could not be tested. As detailed below, we failed to reach these parts of the code due to the limitations of the test system, toolchain, and test interface.

2.1 Character Count Overflow

```
470  if (l > INT_MAX - cnt) {
471      errno = EOVERFLOW;
472      cnt = -1;
473  } else cnt += l;
```

In the code snippet above, the variable `l` contains the number of characters that were written in the last conversion, and `cnt` contains the total number of characters written so far (not including the characters in `l`). The check above is basically ensuring that a single call never generates more than `INT_MAX` characters.

This section is un-coverable because we are limited by the amount of available memory on the system and the build toolchain. To fall into this branch, we would have to supply `snprintf` with a buffer that is larger than `INT_MAX`. The system used for testing does not have that much available memory. In addition to this, when attempting to make a global storage buffer of `INT_MAX` or near `INT_MAX` size, the linker failed to correctly link the test script on the test system.

If we were not limited to testing via the `snprintf` function, and had an actual I/O backed `printf` function to test, this functionality could be trivially invoked. To invoke this behaviour we would supply a conversion string to `printf` the first conversion specification has a field width of `INT_MAX` and the second conversion specification has a field width that is non-zero.

2.2 Automatic Internal Buffer Creation

```
677 if (!f->buf_size) {
678     saved_buf = f->buf;
679     f->wpos = f->wbase = f->buf = internal_buf;
680     f->buf_size = sizeof internal_buf;
681     f->wend = internal_buf + sizeof internal_buf;
682 }
683 ret = printf_core(f, fmt, &ap2, nl_arg, nl_type);
684 if (saved_buf) {
685     f->write(f, 0, 0);
686     if (!f->wpos) ret = -1;
687     f->buf = saved_buf;
688     f->buf_size = 0;
689     f->wpos = f->wbase = f->wend = 0;
690 }
```

We were unable to test lines 677-682, and lines 684-690 of the lines above due to the test interface used. The first `if` clause in the snippet above tests for the existence of a temporary buffer and creates one if it does not exist. The second `if` clause cleans up this buffer if it is created in the first `if` clause. Since we're using `snprintf`, and the buffer used to hold the outputted string is provided in advance (due to the semantics of the `snprintf` function), this code is never invoked.