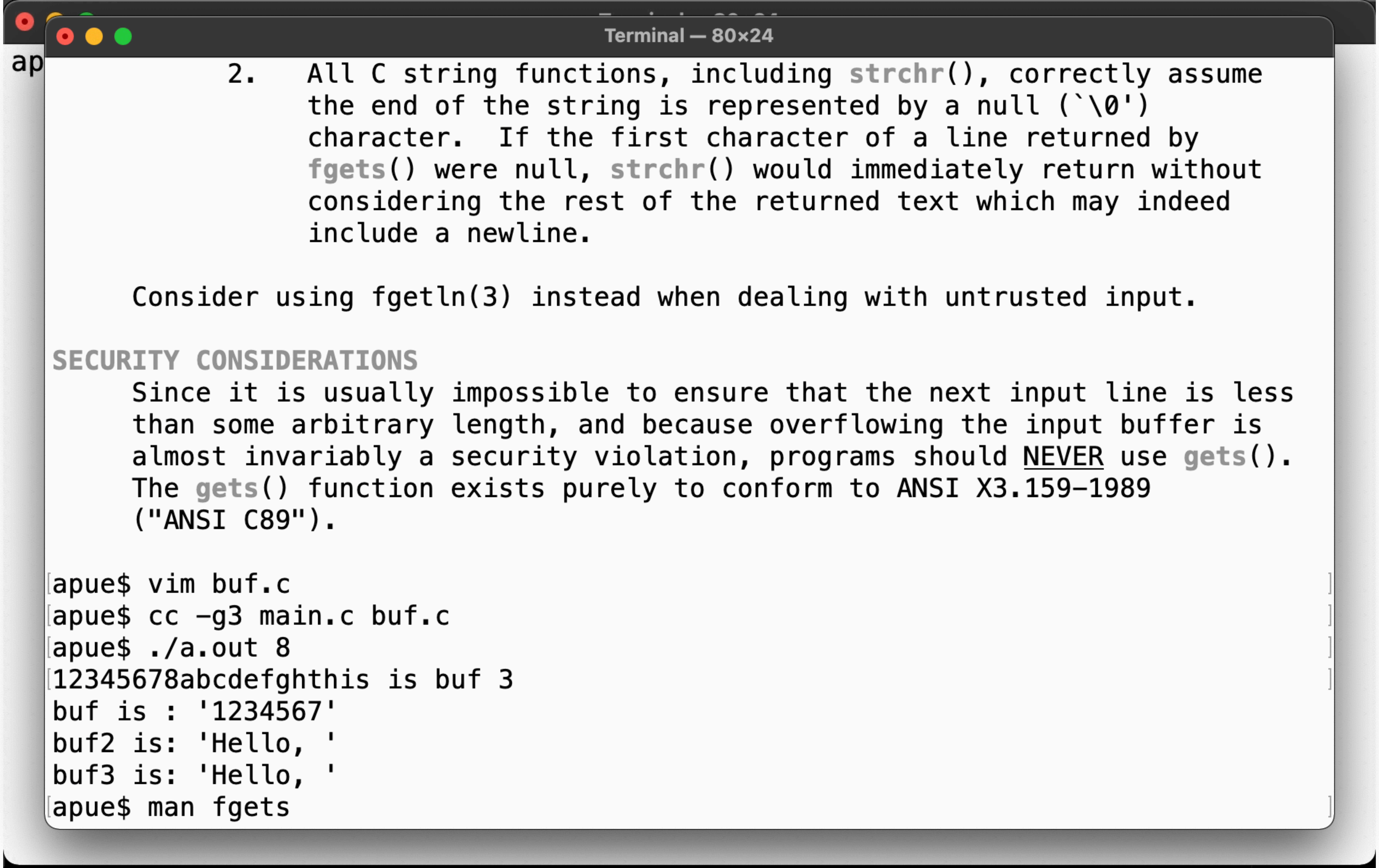
Advanced Programming in the UNIX Environment

Week 05, Segment 12:
Unix Development Tools:
Using `gdb(1)` to understand pointers

Department of Computer Science
Stevens Institute of Technology

Jan Schaumann
jschauma@stevens.edu
https://stevens.netmeister.org/631/



```
            2.   All C string functions, including strchr(), correctly assume
                 the end of the string is represented by a null (`\0')
                 character.  If the first character of a line returned by
                 fgets() were null, strchr() would immediately return without
                 considering the rest of the returned text which may indeed
                 include a newline.

     Consider using fgetln(3) instead when dealing with untrusted input.

SECURITY CONSIDERATIONS
     Since it is usually impossible to ensure that the next input line is less
     than some arbitrary length, and because overflowing the input buffer is
     almost invariably a security violation, programs should NEVER use gets().
     The gets() function exists purely to conform to ANSI X3.159-1989
     ("ANSI C89").

apue$ vim buf.c
apue$ cc -g3 main.c buf.c
apue$ ./a.out 8
12345678abcdefghthis is buf 3
buf is : '1234567'
buf2 is: 'Hello, '
buf3 is: 'Hello, '
apue$ man fgets
```

Using a debugger

The purpose of a debugger such as `gdb(1)` is to allow you to see what is going on "inside" another program while it executes or what it was doing at the moment it crashed.

- we can inspect arbitrary memory locations via the "`x`" command
- "strings" are just arrays of characters; an array is just one way of iterating over sequential memory locations and thus can be accessed equally well via pointer arithmetic: `*(pointer + N) == pointer[N]`
- a buffer overflow does not necessarily lead to a segfault

We only scratched the surface - find a good tutorial and learn more!