THE UNIVERSITY OF WESTERN AUSTRALIA

School of Computer Science and Software Engineering

CITS2002 Systems Programming

CITS2002 Systems Programming - Project 1 2017

See also: **Project 1 clarifications**

This project will assess your understanding of introductory C99 programming language features, including its control-structures, simple data-structures, and its standard library support for handling strings and text files. The project can be successfully completed using the information presented in CITS2002's lectures, laboratories, and workshops, to the end of week-6.

Nearly all devices that connect to the Internet use networking hardware that conforms to IEEE-802 standards, commonly termed the *Ethernet standards*. It is common for desktop and laptop computers to have two Ethernet interfaces, one wired (the IEEE-802.3 standard) and one wireless (the IEEE-802.11 standard), while mobile devices and smartphones will typically have a single wireless Ethernet interface and a Bluetooth interface (the IEEE-802.15.1 standard).

To simplify communication between devices, each Ethernet interface has a unique networking address, termed its Media Access Control (MAC) address. Each MAC address is 6-bytes long but, to make them easier for humans to read and compare, they're usually written as a sequence of hexadecimal characters separated by colons or hyphens, as in the example **74:e2:f5:20:f9:8b**. When written this way, MAC addresses are case-insensitive - that is to say that **74-E2-F5-20-F9-8B** is the same address.

When devices communicate using WiFi (the IEEE-802.11 standard) they break their communication into many small *packets* of data. Each packet contains MAC addresses to identify the device which transmitted the packet (the *transmitter's* address), and the device for which the packet is intended (the *receiver's* address). The special MAC address, **ff:ff:ff:ff:ff:ff**, is termed the *broadcast* MAC address, and is used when all devices within range should receive the packet.

Each 6-byte MAC address comprises two 3-byte parts. The first 3-bytes are termed the Organizationally Unique Identifier (OUI), and they identify the vendor of the Ethernet interface. From the address **74:e2:f5:20:f9:8b**, the **74:e2:f5** indicates that the interface was made by Apple (see www.macvendorlookup.com).

The goal of this project is to write a C99 program, named **wifistats**, that reports the number of bytes either transmitted or received by devices using WiFi. The program may be requested to generate reports sorted by either the device MACs or by the vendors' names, from the largest transmitter (most bytes transmitted) to the smallest.

Program requirements

- Your program's single C99 source file **must** be named **wifistats.c**
- Your program **will** be invoked in one of two ways:

1.
```
prompt>  ./wifistats  what  packets
```

where *what* is either the single character 't' or 'r' to request statistics about WiFi transmitters, or receivers, respectively. For example, if the program is invoked as *./wifistats t packets* then the program should produce statistics about the data transmissions of each transmitting device present in the *packets* file.

packets is the name of a text-file providing information about each WiFi packet, one packet per line. Here is a sample-packet file. Each line consists of 4 fields, providing the time that each packet was captured (in seconds and microseconds), the transmitter's and receiver's MAC addresses, and the length (in bytes) of the packet. Each of the fields will be separated by a single TAB ('\t') character.

If we ran:
```
prompt>  ./wifistats t sample-packets
```

this would be the required output.
If we ran:

```
prompt>  ./wifistats r sample-packets
```

this would be the required output.

2.
```
prompt>  ./wifistats  what  packets  OUIfile
```

where *OUIfile* is the name of a text-file providing the OUIs and names of Ethernet hardware vendors. If an *OUIfile* is provided, the program should report its statistics not on individual device MAC addresses but on the vendor that produced each device. Here are two sample OUIfiles: sample-OUIfile-small (40 entries) and sample-OUIfile-large (23,000 entries). The 2 fields will be separated by a single TAB ('\t') character, and the vendors' names may contain spaces.

For example, if we ran:
```
prompt>  ./wifistats t sample-packets sample-OUIfile-small
```

this would be the required output.
If we ran:
```
prompt>  ./wifistats r sample-packets sample-OUIfile-small
```

this would be the required output.

- You may assume that all input data is in the correct format.
  If your program encounters a broadcast MAC address, your program **should** ignore that input line.
  If your program encounters an OUI not found in a provided OUI file, it **should** use/report the vendor's name as **UNKNOWN-VENDOR.**
- Your program **must** support these limits (you may assume that the input data never exceeds these limits):
  - maximum number of distinct MAC addresses - 500.
  - maximum number of distinct OUIs - 25000.
  - maximum length of any vendor name - 90 characters.
- Your program **must** invoke the standard utility program `/usr/bin/sort` to sort the final results. Your program must use the system-calls *fork()* and *execv()* (and possibly others) to invoke `/usr/bin/sort`. Your program **must not** just call the C function *system()*. Note that the command-line options supported by `/usr/bin/sort` on OS-X, Linux, and cygwin(Windows) are often different.

  If the data totals of two or more MAC addresses, or two or more vendors (via their OUIs), are identical, then the MAC addresses or vendors with identical totals should be printed in *ascending alphabetical order*.
- Your program **must** employ sound programming practices, including the use of meaningful comments, well chosen identifier names, appropriate choice of basic data-structures and data-types, and appropriate choice of control-flow constructs.

Assessment

Projects will be marked using an automatic marking program (for correctness) and by visual inspection (for good programming practices). It is thus important that your program produces its output in the correct format. This project is worth **15% of your final mark** for CITS2002. It will be marked out of 40. The project may be completed **individually or in teams of two** (working as a team is **STRONGLY encouraged**).

During the marking, attention will obviously be given to the correctness of your solution. However, a correct and efficient solution should not be considered as the perfect, nor necessarily desirable, form of solution. Preference will be given to well presented, well documented solutions that use the appropriate features of the language to complete tasks in an easy to understand and easy to follow manner. Your program **must** employ sound programming practices, including the use of meaningful comments, well chosen identifier names, appropriate choice of basic data-structures and data-types, and appropriate choice of control-flow constructs. Do not expect to receive full marks for your program simply because it works correctly. Remember, a computer program should not only convey a message to the computer, but also to other human programmers.

Your project will be marked on the computers in CSSE Lab 2.03, using the macOS environment. No allowance will be made for a program that *"works at home"* but not on CSSE Lab 2.03 under macOS, so be sure that your code compiles and executes correctly on these machines before you submit it.

Submission requirements

1. The deadline for the project is **12noon Friday 22nd September (end of week 8)**.

2. Your submission will be compiled, run, and examined using the macOS platform on computers in CSSE Lab 2.03. Your submission must work as expected on this platform. While you may develop your project on other computers, excuses such as *"it worked at home, just not in the lab!"* will not be accepted.

3. Your submission's C99 source file should each begin with the following lines:

   ```
   /*
      CITS2002 Project 1 2017
      Name(s):             student-name1 (, student-name2)
      Student number(s):   student-number1 (, student-number2)
      Date:                date-of-submission
   */
   ```

   If working in a team of two students, only one needs to submit the project.

4. **You must submit your program electronically using *cssubmit*. No other method of submission is accepted.** You should submit a single C source-code file named **wifistats.c** to be assessed. You do not need to submit any input test files that you used while developing your project. The *cssubmit* facility will give you a receipt of your submission. You should print and retain this receipt in case of any dispute. Note also that the *cssubmit* facility does not archive submissions and will simply overwrite any previous submission with your latest submission.

5. You are expected to have read and understood the University Policy on Academic Conduct. In accordance with this policy, you may *discuss* with other students the general principles required to understand this project, but the work you submit must be the result of your own efforts. **All projects will be compared using software that detects significant similarities between source code files. Students suspected of plagiarism will be interviewed and will be required to demonstrate their full understanding of their project submission.**

Good luck!

Chris McDonald.