Hacking in C
Assignment 1, Wednesday, February 1, 2017

Handing in your answers: Submission via Blackboard (http://blackboard.ru.nl)

Deadline: Wednesday, February 8, 23:59 (midnight)

1. Log into a Linux machine, either locally, or through ssh into `lilo.science.ru.nl` or `stitch.science.ru.nl`.

    (a) Use the command line to create a directory called `sws1-SNUMBER1-SNUMBER2` (replace `SNUMBER1` and `SNUMBER2` by your student numbers). In this directory, create 3 subdirectories called `exercise1`, `exercise2`, and `exercise3`.

    (b) Write a text file called `names.txt` with your names and student S-numbers and place this file into the directory `sws1-SNUMBER1-SNUMBER2`.

    (c) How large (in bytes) is the directory `sws1-SNUMBER1-SNUMBER2`? Write your answer to a file called `1b.txt` and place this file in the `exercise1` directory.

    (d) Write all commands that you used for the other parts of this exercise to a text file called `commands` and place this file in the `exercise1` directory.

2. Download the file http://cryptojedi.org/peter/teaching/genome.txt. You will see that it consists of 500 lines, each consisting of 100 characters, each of those characters being A,C,G, or T. See the content in this file as one long sequence of genome.

    (a) Use Linux shell commands to find out how often the subsequence GATTACA is contained in this genome? Write your answer *and how you found this answer* into a file called `2a.txt`. Place this file in the `exercise2` directory.
    Hint: Be careful, the sequence GATTACA may be spread over two lines in the file!

    (b) Write a shell script called `genome.sh`, which receives as first argument a filename and as second argument a string, and prints, how often the string appears in the file, also counting occurrences of the search string being spread over several lines. Place the script `genome.sh` into the `exercise2` directory.

    (c) Write a shell (bash) script called `gengenome.sh`, which generates output that looks like the content of `genome.txt`, but with random choices of A,C,G, or T. Make sure that

      - the program generates new random output each time it is called;
      - the probability for each of the four letters at each position is 25%; and
      - the program prints exactly 500 lines of 100 characters each.

    Place the script `gengenome.sh` into the `exercise2` directory.
    Note: It may be helpful to take a look at bash programming tutorials online.

    (d) Write a C program called `parsegenome.c`, which checks whether a file given as first command-line argument is of the format of the genome.txt file, i.e., whether it has exactly 500 lines with 100 characters (+ newline) each, where each of the 100 characters of each line is one of either A, C, G, or T. Let the program return -1 if the file does *not* have the correct format and 0 if the file has the correct format. Furthermore, if the file has the correct format, make sure that the program counts how often each of the 4 characters occurs and print these 4 counts to standard output.
    Place the file `parsegenome.c` into the `exercise2` directory.

    (e) Write a Makefile that compiles parsegenome.c and place the Makefile into the `exercise2` directory.

3. Unix and Linux sytems use special files in the `/dev` directory to handle access to devices. Two such special *device files* provide a source of random numbers. These files are `/dev/random` and `/dev/urandom`.

    (a) Find out what the conceptual difference between these two files is. Write your answer to a text file named `exercise3a.txt` and place it in the `exercise3` directory.

    (b) Write a program in a file called `exercise3b.c` that opens the file `/dev/urandom` for reading and then performs a loop which

      - reads one byte from `/dev/urandom`;

- prints one line consisting of the value of this byte as signed decimal integer, as unsigned decimal integer, and in hexadecimal notation (seperated by space);
- exits (from the loop) if the value of the byte is 42.

The last line of output from the program should thus be

```
42 42 2a
```

Place the file `exercise3b.c` into the `exercise3` subdirectory.

(c) Run the program and write the output to a file called `exercise3c`; place this file into the `exercise3` subdirectory.

(d) Write another program called `exercise3d.c`, which does the same as `exercise3b.c`, except for the following:

- Use 16-bit unsigned integers instead of bytes (datatype `uint16_t`, you need to include the file `stdint.h`).
- In the loop, initialize the 16-bit unsigned integer with two random bytes (16 bits) from `/dev/urandom`.
- In the loop, print one line containing the value of hte 16-bit unsigned integer as fixed-width 4-character hexadecimal value (padded at the front with leading zeros).
- Again, terminate the loop if the value is 42, the last line of output is thus
  ```
  002a
  ```
- Run the program 10 times and each time count the number of output lines. Write these counts to a text file called `exercise3d.txt`.

(e) Write a brief description of how you obtained the line counts in part d) in a text file called `exercise3e.txt`.

(f) Place the files `exercise3d.c`, `exercise3d.txt`, and `exercise3e.txt` into the `exercise3` subdirectory.

4. Generate a `tar.gz` archive of the whole `sws1-SNUMBER1-SNUMBER2` directory. Submit this archive in Blackboard.