Assignment 1

CPSC 525 Fall 2025
Due date is posted on D2L. Weight: 20% of your final grade.

As we discussed in our first lecture, limited collaboration is allowed on this assignment. You may discuss this assignment with one other classmate, and you may help each other with high-level design. However, you must write your answers by yourself (both written answers and coding questions). Some amount of similarity will be tolerated, but identical answers or answers with only cosmetic differences will not be accepted (we may ask you to resubmit if we detect too much similarity).

1 Written answers (50%)

Please complete this part of the assignment on Gradescope. You can find a link to it on D2L.

2 Coding question (50%)

For this part of the assignment, you will be writing a C program that runs on Linux. The program will either increment an integer stored in an existing file, or it will create a new file and store an integer equal to 0 in it. The integer stored in the file will be in binary form (little endian order). Your program will check for all possible error conditions and report an appropriate status if it cannot complete the task successfully. While this is a simple coding assignment, the main focus of it is on detecting error conditions and reporting appropriate error codes.

You will be submitting your code to Gradescope, via a link posted on D2L. Gradescope will automatically grade your code, and you can re-submit as many times as you like before the deadline.

The autograder on Gradescope will be configured to test your code using Fedora 38. Unfortunately, in our department we have already upgraded all machines to Fedora 42, and we no longer have access to the older Fedora 38. However, you should still be able to do all your development and testing on our Linux computers (e.g. `cslinux.ucalgary.ca` or the Linux workstations on the main floor of MS), as the two versions of Fedora are sufficiently similar.

Starter code

Start by downloading and compiling the starter code:

```
$ git clone https://gitlab.com/cpsc525f25/binadd.git
$ cd binadd
$ make
$ ./binadd
Need 1 argument!
```

The starter code contains an incomplete implementation of the program that you need to write. The only file you need to modify and subsequently submit for grading is `binadd.c`. It contains the function `binadd()`, which you need to finish writing.

The rest of the starter code contains the following files:

- `main.c` contains a driver, which parses command line arguments (name of the file), calls the `binadd()` function from `binadd.c` and then displays the returned error code.
- `binadd_errors.h` contains status code definitions that `binadd()` must return.
- `binadd_errors.c` contains convenience functions for formatting status codes as English text.
- `Makefile` will help you with code compilation. Use `make(1)` to compile the code into an executable called `binadd`.

The `binadd()` function, takes a single parameter, the name of the file in which to increment the integer.

```
int binadd(const char * fname);
```

If the file can be read and modified, and its contents represent either an 8-bit, 16-bit or 32-bit unsigned integer stored using little endian byte order, the function will update the file so that it contains that integer incremented by 1. If the file has incompatible length, or the integer cannot be incremented without overflow, or there are any errors while updating the file contents, the function will return an appropriate status code.

If the file does not exist, the function will create the file and store in it a 32-bit integer value zero (0). If the function cannot successfully accomplish this, it will return an appropriate status code.

Required Pseudocode

Your `binadd()` must implement the following pseudocode.

- use `open(2)` on `fname` to open the file for reading and writing
- if `open(2)` fails, check `errno(3)` to find out why
    - if `open(2)` failed because the file does not exist:
        - use `open(2)` again to create `fname` file, with permissions 0600 (-rw-------)
        - use `write(2)` to write a 32-bit value of `0` (zero) into the file
        - use `close(2)` to close the file
        - if there were no errors, return `BA_SUCCESS`
        - if you detect any errors, close the file if it is open, and return appropriate status code
    - if `open(2)` failed because of some other reason, return `BA_EOPEN`
- use `lseek(2)` to determine file size and remember it (let's say the file is n-bit long)
- if n is not one of {8, 16, 32} return `BA_ESIZE`
- use `lseek(2)` again to rewind the file to the beginning
- use `read(2)` to load the contents of the entire n-bit integer from file into memory
- increment the n-bit integer (8-bit, or 16-bit or 32-bit)
- if you detect overflow (i.e. the integer was already at its maximum size), return `BA_EMAX`
- use `lseek(2)` for the 3rd time to rewind the file to the beginning
- use `write(2)` to save incremented integer back to the file, overwriting its contents
- use `close(2)` to close the file
- if there were no errors, return `BA_SUCCESS`
- if you detect any errors, close the file if it is open, and return appropriate status code

You may only use the system calls mentioned above. The correct sequence of system calls for the case where the file exists is: open, lseek, lseek, read, lseek, write, close. The correct sequence of system calls in case where the file does not exist is: open, open, write, close. Do not repeat system calls upon failure.

At each step, you need to check for error conditions, e.g. by checking the results of all system calls, checking for integer overflow. If you detect an error, you must return the appropriate status code from the table below. Please note that, if you detect an error condition while the file is open, you must `close(2)` the file before returning the error status, and in that case you can ignore the return value of such `close(2)` call.

Status code	Meaning of the status code.
`BA_SUCCESS`	Indicates success.
`BA_EOPEN`	Indicates that `open(2)` failed for a reason other than the file does not exist.
`BA_ECREATE`	Return this if `open(2)` failed when trying to create the file.
`BA_ECLOSE`	Return this if `close(2)` failed.
`BA_EREAD`	Return this if `read(2)` failed.
`BA_EWRITE`	Indicates that `write(2)` failed.
`BA_ESEEK`	Indicates that `lseek(2)` failed.
`BA_ESIZE`	Return this if the file length is not 1, 2 or 4.
`BA_EMAX`	Return if the file already contains a maximum integer (i.e. incrementing it would result in overflow).

Sample execution

Here is a sample execution of the `binadd` program, using a correctly implemented `binadd()` function:

```
$ rm -f file1.dat    # delete file if it exists
$ ./binadd file1.dat # this will create the file with value 0
Success (BA_SUCCESS)
$ xxd -p < file1.dat # verify contents
00000000
$ ./binadd file1.dat # this will increment the number in file
Success (BA_SUCCESS)
$ xxd -p < file1.dat # verify contents
01000000
$ chmod -w file1.dat # remove write access to file
$ ./binadd file1.dat
Failed to open existing file. (BA_EOPEN)
$ ./binadd /var/x    # try to create file where we don't have permission
Failed to create new file (BA_ECREATE)
$ printf '\xfe\xff\xff\xff' > file2.dat # make file with MAX-1 number
$ xxd -p file2.dat
fefffffff
$ ./binadd file2.dat
Success (BA_SUCCESS)
```

```
$ xxd -p file2.dat
ffffffff
$ ./binadd file2.dat
Value is at maximum (BA_EMAX)
```

Marking

- The autograder will grade your code using many different test cases.
- The test cases will verify that your program runs correctly and that it returns correct status codes under error conditions.
- You should design your own test cases to verify your code runs correctly.
- The output of the autograder on Gradescope may contain hints as to why your solution isn't working.
- Your code should not produce any stdout/stderr output. If you use printf or similar to assist you with debugging, make sure to disable it before submitting your code. If you don't, your program will be marked as incorrect.
- Document and format your code. Ugly formatting and/or insufficient documentation will result in up to 20% penalty.

Hints

Opening existing file for reading and writing:

```
int fd = open(fname, O_RDWR);
```

Creating a file with "-rwx------" permissions:

```
int fd = open(fname, O_CREAT | O_RDWR | O_EXCL, S_IRWXU);
```

Seeking to the beginning of the file:

```
off_t res = lseek(fd, SEEK_SET, 0);
off_t res = lseek(fd, 0, SEEK_SET);
```

Please note that this is a very simple coding question, and the main purpose of it is to warm up your coding skills. Future assignments will very likely require more coding effort.

Submission

If you worked on the assignment with another student, put the name of the other student at the top of the `binadd.c` in a comment.

Submit `binadd.c` on Gradescope. The file you submit must compile and run with unmodified starter code, so make sure you only modify the `binadd.c` file. Do not submit any other files.

General information about all assignments

- All assignments are due on the date listed on D2L. Please refer to the outline for late submission policy and relevant penalties.

- You can submit as many times as you like.
- Assignments will be marked by your TAs. If you have questions about the assignment, post them on Ed Discussion. If possible, make your questions public, so that the answers will be visible to everyone. If you need to include personal/sensitive information in your post, you can mark your post private, and only the TAs and the instructor will be able to see the question.
- If you have questions about assignment marking, contact your TAs first. If you still have questions after you have talked to your TA, then you can contact your instructor.
- Other than the limited collaboration allowed (as discussed during lectures), assignments must reflect your own work. Here are some examples of what you are not allowed to do for assignments: you are not allowed to copy code or written answers (in part, or in whole) from anyone else; you are not allowed to share your solutions (including code or pseudocode) with anyone; you are not allowed to sell or purchase a solution; you are not allowed to make your solutions available publicly, you are not allowed to use AI tools to generate answers. This list is not exclusive. For further information on plagiarism, cheating and other academic misconduct, check the information at this link: http://www.ucalgary.ca/pubs/calendar/current/k.html.
- We may use automated similarity detection software to check for plagiarism. Your submission will be compared to other students (current and previous), as well as to any known online sources. Any cases of detected plagiarism or any other academic misconduct will be investigated and reported.