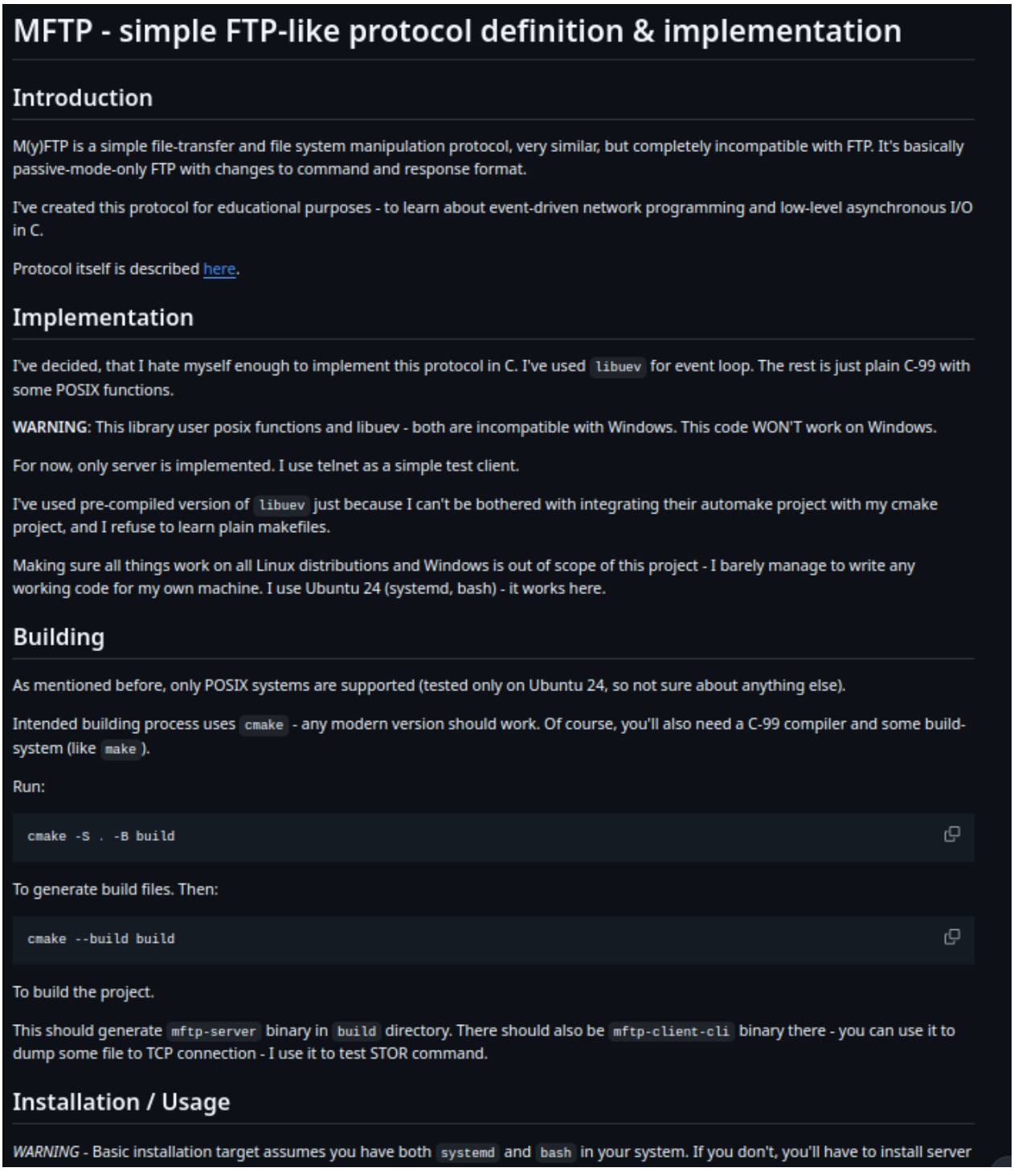
Mikołaj Trafisz
Nr. Albumu:
Data zajęć: 29.10.24r.

Sprawozdanie

Zadanie: Tworzenie dokumentacji elektronicznej, edycja wybranych plików konfiguracyjnych. Elementy zdalnej pracy.

Z Markdowna, jako narzędzia do formatowania tekstu, używam często do podstawowej dokumentacji moich prywatnych projektów. Każdy z moich projektów zawiera plik README.md, np. https://github.com/mtrafisz/mftp/blob/master/readme.md:



MFTP - simple FTP-like protocol definition & implementation

Introduction

M(y)FTP is a simple file-transfer and file system manipulation protocol, very similar, but completely incompatible with FTP. It's basically passive-mode-only FTP with changes to command and response format.

I've created this protocol for educational purposes - to learn about event-driven network programming and low-level asynchronous I/O in C.

Protocol itself is described here.

Implementation

I've decided, that I hate myself enough to implement this protocol in C. I've used `libuev` for event loop. The rest is just plain C-99 with some POSIX functions.

WARNING: This library user posix functions and libuev - both are incompatible with Windows. This code WON'T work on Windows.

For now, only server is implemented. I use telnet as a simple test client.

I've used pre-compiled version of `libuev` just because I can't be bothered with integrating their automake project with my cmake project, and I refuse to learn plain makefiles.

Making sure all things work on all Linux distributions and Windows is out of scope of this project - I barely manage to write any working code for my own machine. I use Ubuntu 24 (systemd, bash) - it works here.

Building

As mentioned before, only POSIX systems are supported (tested only on Ubuntu 24, so not sure about anything else).

Intended building process uses `cmake` - any modern version should work. Of course, you'll also need a C-99 compiler and some build-system (like `make`).

Run:

```
cmake -S . -B build
```

To generate build files. Then:

```
cmake --build build
```

To build the project.

This should generate `mftp-server` binary in `build` directory. There should also be `mftp-client-cli` binary there - you can use it to dump some file to TCP connection - I use it to test STOR command.

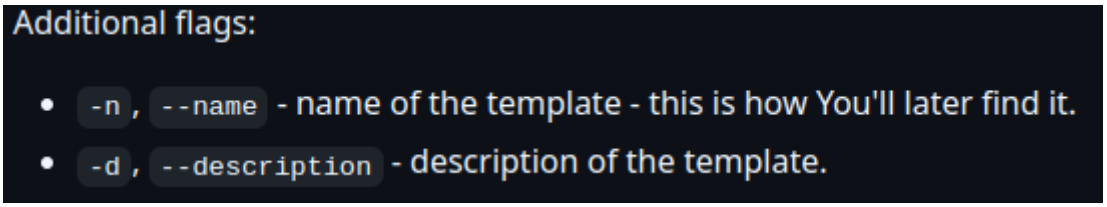
Installation / Usage

WARNING - Basic installation target assumes you have both `systemd` and `bash` in your system. If you don't, you'll have to install server

Do podstawowej dokumentacji technicznej najprzydatniejsze formatowania to:

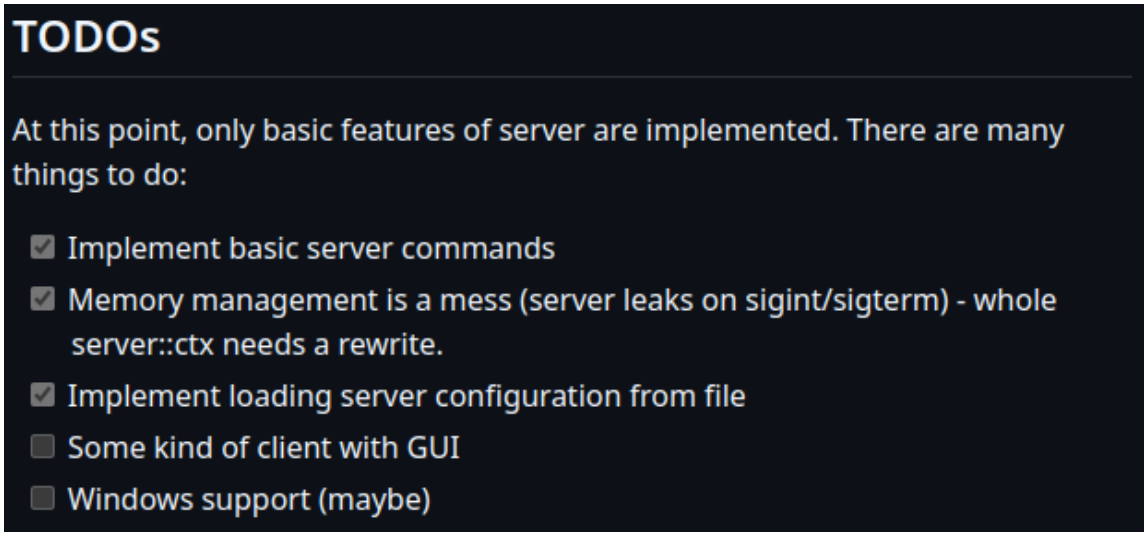
- “```<kod>```” - do zawierania przykładów kodu, lub wypisywania poleceń, które należy wykonać, by osiągnąć jakiś cel (np. Instalacja dokumentowanego projektu). Formatowanie to pozwala zwykle osiągnąć lepszy kontrast w porównaniu do zwykłego tekstu. Jako przykład: wykaz poleceń, które należy wykonać, by zainstalować jeden z moich programów (źródło: https://github.com/mtrafisz/templater/blob/master/readme.md)

- “`<tekst>`” - do wyróżniania tekstu o specjalnym znaczeniu. Np:



W tym przypadku, flagi programu są wyróżnione, by nie mieszały się z ich opisem.

- “- [ ]” i “- [x]” – do tworzenia “checklist”. Np:



Wnioski do tego zastosowania:
Markdown jest niezastąpiony do tworzenia prostej, “roboczej” dokumentacji technicznej. Gdybym musiał otwierać oddzielny program, np edytor PDF, do tworzenia dokumentacji technicznej, większość moich projektów nie miała by ani linijki komentarza czy dokumentacji.
Niektóre edytory tekstu potrafią także wyświetlać markdownową dokumentacje w kodzie (w postaci komentarzy), jako wyrenderowany markdown. Pozwala to tworzyć czytelniejszą dokumentacje już w czasie programowania.

Markdown jest dla mnie również przydatny do tworzenia elektronicznych notatek i ściąg. Jako przykład, moje notatki do Matury z Polskiego (źródło: https://github.com/mtrafisz/notatki-matura/), lub notatki do sprawdzianu z języka Niemieckiego:

Lektury do powtórzenia

- Pieśń o Rolandzie
- Kronika Polska
- Boska Komedia
- Kazania Sejmowe
- Skąpiec
- Konrad Wallenrod
- Pan Tadeusz
- Potop
- Dziady
- Balladyna
- Kordian
- Lalka
- zbrodnia i kara - można film

Epoki Literackie

Antyk

- Epoka Rozumu
- Ład, harmonia i proporcja
- Ścisłe gatunki, zasady i motywy literackie - trzy jedności, decorum, mimesis, fatum
- Motyw podróży - Odyseusz
- Konflikt między prawem sakrum i profanum - Antygona
- Idealny rycerz - Achilles
- Człowiek w obec cierpienia - Hiob
- Filozofia: epikureizm, stoicyzm, sceptycyzm, hedonizm (skrajny epikureizm)

Średniowiecze

- Epoka Wiary
- Teocentryzm - skupienie na bogu
- Autorzy anonimowi
- Dance Macabre - Taniec Śmierci - Śmierć dosięgnien każdego, niezależnie od statusu społecznego, ani tego, kim był za życia. *Rozmowa Mistrza Polikarpa ze Śmiercią*
- Ars Moriendi - Sztuka umierania, Idealny rycerz - Pieśń o Rollandzie
- Vanitas Vanitatum - przemijanie, marność nad marnościami

1: Luki abc

1. c `weißt`
2. a `wann`
3. b `weiß`
4. a `den`
5. a `an der`
6. c `wann`
7. c `in die`
8. b `halben`
9. a `am`
10. a `Sie`

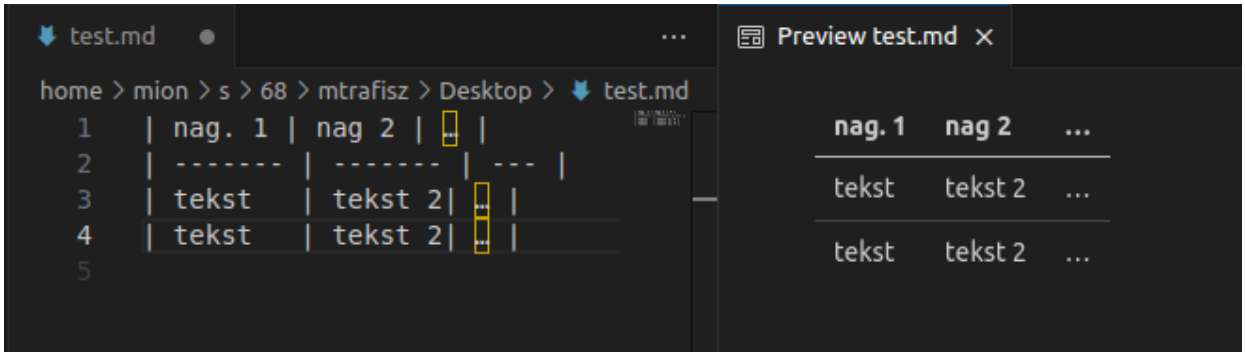
2. Przyimki `in` `zu` `an` `aus`

Nie umiem :)

Polski	Bezok.	Prateritum
pożyczać	liehen	lieh
czytać	lesen	lies
leżeć	liegen	lag
brać	nehmen	nahm
nazywać	nennen	nannte
jeździć konno	reiten	ritt
radzić	raten	riet

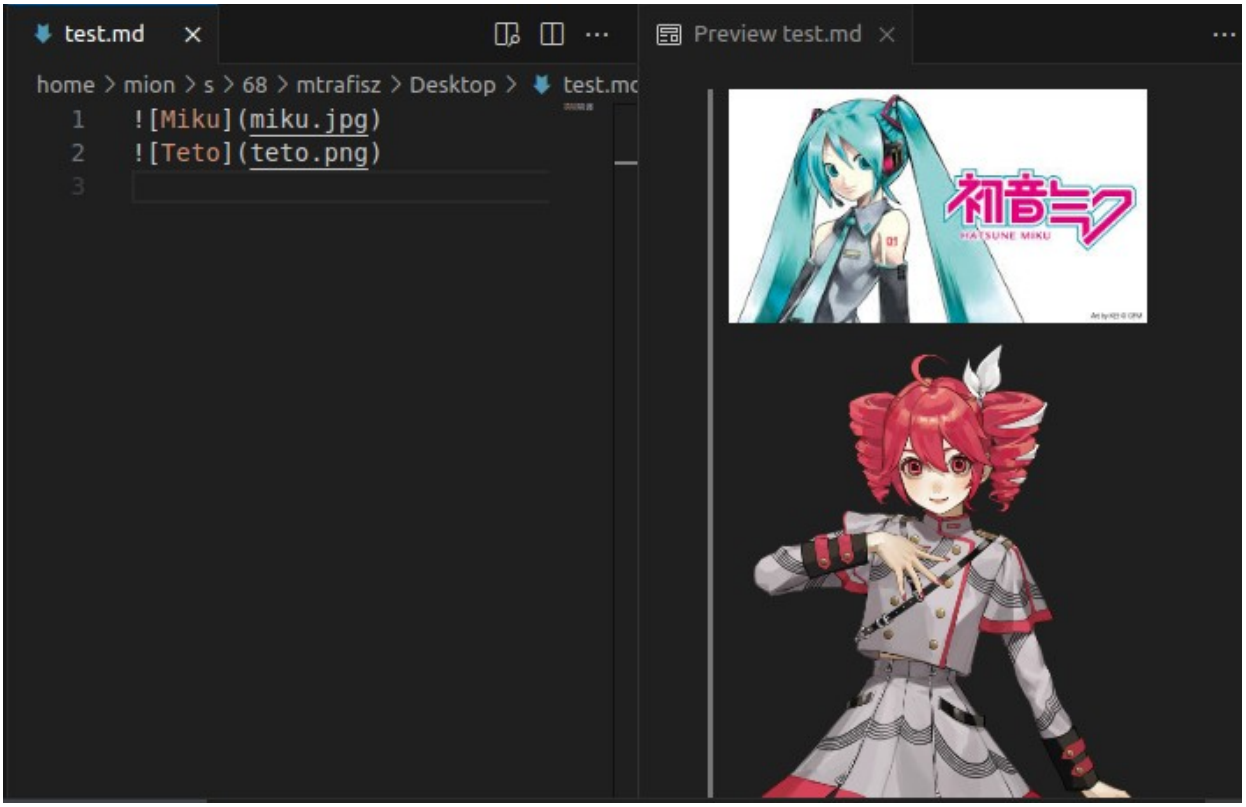
Do notatek mniej technicznych, najprzydatniejszymi formatowaniami są:

- Tabele:



- Listy punktowane i numerowane:
  “
  1. tekst
  2. inny tekst
  …
  “

  lub
  “
  - tekst
  - inny tekst
  - …
  “
- Zdjęcia:



Wnioski do tego zastosowania: Markdown jest idealnym programem do tworzenia prywatnych notatek. Wiele programów do notowania (np. Obsidian) korzysta z Markdowna (lekko zmodyfikowanego), jako format do zapisywania plików użytkownika.

Wnioski do laboratorium:
Do zalet markdowna należy:

- Prostota tworzenia – wystarczy edytor tekstu.
- Uniwersalność – Markdown jest czytelny nawet bez dedykowanego oprogramowania do renderowania.

Do wad:

- Dość mało możliwości formatowania.
- Czytniki i programy do tworzenia .md często rozszerzają podstawowe formatowanie, przez co dokument stworzony na jednej maszynie, może wyświetlać się źle na innej.