

**Langtonov mravec s ukladaním vzorov na serveri**

SEMESTRÁLNA PRÁCA

*Dokumentácia*

Vypracovali: **Maroš Gorný a Roman Koki**

Študijná skupina: **5ZYI31 a** **5ZYI35**

Predmet: **Princípy operačných systémov**

Cvičiaci: *Ing. Patrik Rusnák, PhD.*

Obsah

[1. Úvod 4](#_Toc124112421)

[2. Štruktúra projektu 5](#_Toc124112422)

[2.1. Client 5](#_Toc124112423)

[2.2. Server 5](#_Toc124112424)

[2.3. structuresEnums 6](#_Toc124112425)

[2.4. ant 6](#_Toc124112426)

[2.5. client\_definition 7](#_Toc124112427)

[2.6. server\_definition 7](#_Toc124112428)

[2.7. client\_server\_definition 7](#_Toc124112429)

[2.8. settings 8](#_Toc124112430)

[3. Využitie socketov 9](#_Toc124112431)

[4. Využitie vlákien 10](#_Toc124112432)

[5. Synchronizačný problém 11](#_Toc124112433)

[5.1. Hlavné synchronizačné problémy 11](#_Toc124112434)

[5.2. Prostriedky využité pri riešení synchronizačného problému 11](#_Toc124112438)

# Úvod

Cieľom našej semestrálne práce bolo vytvoriť simuláciu Langtonového mravca ktorá funguje na princípe klient-server architektúre a je možné pripojiť viac klientov a vytvoriť viac mravcov. V danej práci bola využitá asynchrónna komunikácia implementovaná v jazyku C. Na vývoj projektu sme použili verzionovací systém GitHub.

# Štruktúra projektu

Obrázok, na ktorom je text

Automaticky generovaný popis

## Client

Pri potvrdení spojenia so serverom sa každému klientovy vytvorí jedinečný socket ktorý daného klienta identifikuje. Následne sa inicializujú dáta ktoré bude klient zdieľať so serverom a to tak, že hneď po pripojení server pošle informácie s aktuálnymi dátami klientovi a ten si informácie zapíše a uloží. Keď je klient pripravený na interakciu s používateľom, vytvorí si vlákno pre posielanie správ serveru a hlavne vlákno bude slúžiť ako čítanie správ zo servera.  
  
Cez klienta môže používateľ nastavovať nastavenia simulácie, ktoré budú odoslané na server.

## Server

Server si na začiatku vytvorí takzvaný pasívny socket na ktorý sa klienti budú môcť pripojiť. Následne sa inicializujú dáta ktoré bude server zdieľať s klientom. Vytvorí sa vedľajšie vlákno pre samotnú simuláciu mravcov, ktorá však bude čakať na to pokiaľ ju používateľ spustí. Hneď po pripojení klienta na server sa automaticky vytvoria dve ďalšie vlákna ktoré budú slúžiť na komunikáciu s klientom a teda jedno vlákno na čítanie správ od klienta a jedno na odosielane správ klientovi.

Keď klient nastaví všetky potrebné informácie, vlákno ktoré čaká na spustenie simulácie sa spustí a tým pádom na serveri začne bežať simulácia Langtonového mravca.

## structuresEnums

V hlavičke structuresEnum sú definované všetky enums a structs potrebné pre chod aplikácie.

* Enums
  + LOADING\_TYPE
  + LOGIC\_TYPE
  + ANT\_DIRECTION
  + BACKGROUND\_COLOR
  + COLLISION
  + ACTION\_CODE
* Structs
  + DATA
  + BOX
  + DISPLAY
  + ANT

## ant

Obrázok, na ktorom je text

Automaticky generovaný popis

V súboroch ant.c a ant.h sa nachádza funkcia pre vlákno mravca ktoré sa spúšťa v hlavnej simulácii. V danej funkcii je naprogramovaná logika mravca, tak aby sa vedel pohybovať po displeji a aby vedel čo ma robiť v prípade stretu s ďalším mravcom.

Taktiež sú tu aj pomocné funkcie ktoré však slúžia len na zjednodušenie funkcii pre vlákno.

## client\_definition

Obrázok, na ktorom je text

Automaticky generovaný popis

V súboroch client\_definition.c a client\_definition.h sa nachádzajú dve funkcie pre vlákna. Jedno na čítanie dát od servera a jedno na posielanie dát serveru. Taktiež sú tu pomocné funkcie ktoré vytlačia text na terminál alebo nainicializujú/uvoľnia zdieľané dáta.

## server\_definition

Obrázok, na ktorom je text

Automaticky generovaný popis

V súboroch server\_definition.c a server\_definition.h sa nachádzajú dve funkcie pre vlákna. Jedno na čítanie dát od klienta a jedno na posielanie dát klientovi. Taktiež sú tu pomocné funkcie ktoré vytlačia text na terminál alebo nainicializujú/uvoľnia zdieľané dáta.

## client\_server\_definition

Obrázok, na ktorom je text

Automaticky generovaný popis

V súboroch client\_server\_definition.c a client\_server\_definition.h sa nachádzajú pomocné funkcie pre výpis informácií a na zastavenie komunikácie.

## settings

Obrázok, na ktorom je text

Automaticky generovaný popis

V súboroch settings.c a settings.h nájdeme hlavne funkcie ktoré slúžia na nastavenie simulácie poprípade na vytlačenie displeja na terminál.

# Využitie socketov

Sockety sme využili na komunikáciu medzi klientom a serverom. Hlavne na prenos nastavení ktoré klient potreboval odovzdať serveru a následne server potreboval danú informáciu poslať všetkým ostatným klientom. Taktiež boli sockety použité na posielanie súborov medzi klientom a serverom. Všetky dáta ktoré sa prenášali cez sockety boli vo forme stringov.

# Využitie vlákien

Vlákna sme v našom projekte využili hlavne pre každého mravca ktorý sa vygeneruje a taktiež pre server a každého klienta.

Na serveri bude bežať 1 hlavné vlákno, 1 vlákno pre simuláciu, N vlákien podľa počtu mravcov a 2\*M vlákien podľa počtu klientov.   
V hlavnom vlákne sa bude spracovávať prijímanie nových socketov. Vo vlákne pre simuláciu sa nastavia všetky nastavenia podľa dát ktoré bude mať simulácia k dispozícii a vytvorí sa N vlákien podľa počtu mravcov ktorí majú byť umiestnený na displej. V hlavnom vlákne sa tiež vytvorí 2\*M vlákien pre každého klienta, pričom jedno bude vždy na čítanie dát od klienta a druhé na zapisovanie dát klientovi.

Klient bude mať jedno hlavné vlákno v ktorom sa pripojí na server, nainicializuje dáta a následne v ňom bude čítať dáta ktoré prišli zo serveru. Taktiež bude mať jedno vedľajšie vlákno s ktorým bude posielať dáta na server.

# Synchronizačný problém

Hlavný synchronizačný problém ktorý sme riešili bol ten, že čo sa stane v prípade ak sa stretnú viacerí mravci(vlákna) na jednom políčku. V simulácií sa stret mravcov vyriešil tromi spôsobmi

## Hlavné synchronizačné problémy

1. Prežije len jeden mravec zo všetkých mravcov ktorí sa stretli
2. Polovica mravcov sa začne správať podľa doplnkovej logiky
3. Zomrú všetky mravce ktoré sa stretli

Pri prvom spôsobe sme situáciu vyriešili pomerne jednoducho a to tak, že prvý mravec ktorý sa dostavil (mal to šťastie a bol o pár makro sekúnd rýchlejší) na políčko si ho uzamkol a až do ďalšieho posunu si ho nechal uzamknuté. Tým pádom všetky ostatné mravce ktoré sa pokúsili uzamknúť dané políčko tak spraviť nemohli a umreli.

Pri druhom spôsobe sme situáciu vyriešili pomocou počítadla ktoré sa nachádza na políčku a začína vynulované. V prípade, že sa mravec na políčko dostaví, zväčší počítadlo o jedno a skontroluje koľko mravcov sa na aktuálnom políčku práve nachádza. Ak bolo počítadlo bezo zvyšku deliteľné dvomi, tak mravcovi ktorý akurát prišiel sa v tom momente zmenila na trvalo jeho logika. Následne sa počkalo na všetkých mravcov aby sa dostavili na svoje pozície a keď už sa žiadny mravec nepohyboval, počítadlo sa resetovalo na 0.

Tretí spôsob sme riešili taktiež s počítadlom, ktoré sa nachádzalo na políčku, ale tu si mravce kontrolovali počítadlo až keď sa všetky mravce zastavili. Ak počítadlo na ktorom bol mravec bolo väčšie alebo rovné ako dva, tak bolo jasné, že sa stretli aspoň dva mravce a všetky mravce ktoré sa stretli zomreli.



## Prostriedky využité pri riešení synchronizačného problému

* pthread\_mutex
* pthread\_mutex\_lock / pthread\_mutex\_unlock
* pthread\_mutex\_trylock
* pthread\_barrier\_t

Pri všetkých spôsoboch sme samozrejme používali mutex, ale taktiež aj barieru a to práve na to, aby sme zaručili to, že mravce (vlákna) budú na seba čakať a nestane sa to, že jeden mravec by bol o dva kroky dopredu.

Pri prvom spôsobe sme pomocou metódy pthread\_mutex\_trylock buď mutex uzamkli, alebo v opačnom prípade, uzamknutie nebolo úspešné a mravec zomrel.

Pri druhom a tretom spôsobe sme políčko vždy uzamkli, inkrementovali počítadlo a následne odomkli mutex na danom políčku. Avšak pri druhom spôsobe už mravec hneď po inkrementovaní vedel, či zmení logiku alebo či je v druhej polovici ktorá logiku nezmení, poprípade je prvý a nič sa mu nestane a následne len všetky mravce počkali na bariéru aby mohli pokračovať ďalej. Pri tretom spôsobe mravec uzamkol mutex, inkrementoval počítadlo, odomkol mutex, ale následne počkal pokiaľ všetky mravce dokončia svoj pohyb a skontrolovalo sa, či je počítadlo väčšie ako 1 a teda, že na políčku je viac mravcov.