



Santa Claus Problem

NVS Projekt 1

Alexander Grill 5CH1F
12. Januar 2021

Informatik
HTBLUvA Wr.Neustadt
Österreich

Inhaltsverzeichnis

1	Einführung	2
1.1	Vorwort	2
1.2	Motivation	2
2	Aufgabenstellung	4
2.1	Erleuterung der Grundproblematik	4
2.2	Idee	5
2.3	Kommandozeilenparameter	5
2.4	Konsolenausgabe	5
3	Themenbereiche	6
4	Implementierung	7
4.1	Aufbau	7
4.2	Source Codes Dokumentation	8
4.2.1	Elves [Klasse]	8
4.2.2	Reindeer [Klasse]	9
4.2.3	Santa Claus [Klasse]	10
4.2.4	Zusätzliche Funktionen	11
4.2.5	Sonstiges	11
4.3	Verwendete Bibliotheken	11
4.3.1	CLI11	11
4.3.2	spdlog	11
4.3.3	tabulat	11
4.3.4	json	11
5	Anwendungsfälle	11

1 Einführung

In diesem Kaptiel wird erklärt was der Grund für die Umsetzung war, welche Punkte in die Benotung miteinfließen und um welche Problemstellung es sich handelt. Die genauere Erläuterung der Grundproblematik wird aber im 2 Kaptiel beschrieben.

1.1 Vorwort

Auf Grund der aktuellen Lage(COVID-19) in Österreich, wurde der Unterricht an den Schulen in Form von Distance-Learning abgehalten. Dehsalb war die Durchführung der Praktischen Arbeit in Netzwerktechnik nicht möglich. Herr Professor Kolouseck gab uns daraufhin eine Projektarbeit die bis zum 20.01.2021 zu erledigen ist. Die Gesamtnote des Projekt ersetzt ausschließliche die Note, die man bei der Praktischen Arbeit erworben hätte. In diesem Projekt geht es darum mit Prozessen, Thread und Synchronisation für die jeweiligen Anwendungsszenarien entwickelt werden, um damit zu zeigene, dass die Praktischen Fähigkeiten zum Implementieren verteiler Systeme erworben wurden. Die Programmiersprache die für die Implementierung verwendet wird, ist C++.

Das Projekt besteht aus 3 wichtigen Punkten:

- praktischen Ausarbeitung
 - dabei wird die Grundproblematik simuliert, klargestellt und verschiedene Szenarios angewendet
- theoretisch Ausarbeitung
 - in diesem Teil wird die Aufgabenstellung, Probleme, Dokumentation der Source Codes usw. festgehalten
- Git Hub Repository
 - das Projekt muss auf GitHub gehostet werden, um die Verwaltung der Projekts zu erleichtern und, um den Workflow zu dokumentiern
 - Commites müssen gemacht werden, das man etwas als Patch verwenden kann (Fehler, Probleme können zurückgenommenr werden)
 - Commit-Meldungen sollen kurz und prägnant sein und sollen ausdrücken für was dieser Commit steht.

1.2 Motivation

In diesem Projekt wird die Santa Claus Problematik erläutert und mit Hilfe eines C++ Projekt simuliert. Santa Claus wird wegen zwei Faktoren geweckt, sonst benötigt er umbedingt seinen Schlaf. Er wird geweckt, wenn Elfen ihm brauchen, weil sie mit der Arbeit nicht weiter kommen, und dadurch die Produktion der Geschenke für die Kinder verlangsamt oder gar gestoppt wird. Deshalb ist es wichtig, dass in diesem Moment Santa geweckt wird, um Santa den Elfen helfen zu können. Zugleich kommen auch die Rentiere in unterschiedlichen Zeitpunkten zurück von ihrere Reise und sammeln sich im Stall. Dort warten sie gemeinsam bis sie vollzählig sind und Santa mit ihnen, folgedessen die Geschenke zu den Kindern liefern. Auch in diesem Moment ist es äußerst wichtig, dass Santa geweckt wird, weil er schon einen Teil der Geschenke mit den Rentieren ausliefern

kann. Zu beachten ist jedoch, dass es zu keinem Zusammenstoß zwischen den Gruppen kommt und Santa Claus nicht weiß, welche Tätigkeiten zuerst vollendet werden sollen bzw. welche eine höhere Priorität haben.

2 Aufgabenstellung

Dieses Kapitel umfasst die genaue Beschreibung der Grundproblematik und deren Aufgabenstellung. Weiters wird auch über die Idee der Umsetzung geschrieben. Zusätzliche Erweiterungen wie Kommandozeilenparameter, Konsolenausgabe etc.

2.1 Erleuterung der Grundproblematik

Dieses Problem stammt aus William Stallings Operating Systems. Dabei wird folgende Problemstellung beschrieben:

Santa Claus sitzt in seinem Spielwarenladen am Nordpol und schläft, während seine zurückgekehrten Rentiere im Stall fressen um Kräfte für die jährliche Auslieferung der Geschenke an die Kinder zu Weihnachten zu sammeln. Seine fleißigen Elfen arbeiten sorgfältig an den Geschenken der Kinder in der Spielzeugfabrik. Hin und wieder kann es vorkommen, dass die Elfen beim Basteln vor einem Problem stehen und ohne Hilfe vom Santa nicht mehr weiter machen können. Es wäre eine Katastrophe, wenn die Geschenke nicht rechtzeitig am Heiligen Abend an die Kinder ausgeliefert werden können, weil einige Elfen die Produktion der Geschenke blockiert haben.

Aus diesem Grund muss Santa Claus unbedingt geweckt werden, obwohl er ein gewisses Maß an dringendem Schlaf benötigt. Erst wenn 3 oder mehr Elfen ohne Hilfe nicht mehr weiter arbeiten können wecken sie ihn auf. Wenn drei Elfen ihr Problem gelöst haben, müssen alle anderen Elfen, die den Weihnachtsmann besuchen wollen auf die Rückkehr dieser Elfen warten. Jedoch ist zu beachten, dass wenn Elfen vor der Tür seines Ladens warten, während das letzte Rentier für die Auslieferung aus dem Tropen zurückkehrt, beschließt der Weihnachtsmann, die Elfen bis nach Weihnachten warten zu lassen. Denn die es ist wichtiger den Schlitten fertigzustellen und die Pakete auszuliefern. Nachdem Santa Claus ihnen beim Problem geholfen hat, können die Elfen erleichternd weiterbasteln und Santa Claus seinen Schlaf forsetzen.

Allerdings möchte Santa Claus die Kinder nicht zu lange auf ihre Geschenke warten lassen, und deshalb soll er auch aufgeweckt werden, wenn genug Rentiere bereit sind bzw. zurück aus ihrem Urlaub im Südpazifik gekommen sind, um den Schlitten mit den Geschenken zu ziehen und eine Ladung Geschenke zu verteilen. Das letzte Rentier, das ankommt, muss den Weihnachtsmann holen, während die anderen Rentiere gemütlich in der warmen Hütte warten, bevor sie vor dem Schlitten gespannt werden.

Die restliche Zeit kann Santa schlafen, um für die nächste anstrengenden Tätigkeiten Kräfte zu schöpfen. Man kann davon ausgehen, dass stets genug Arbeit für die Elfen und genug Geschenke für eine Ladung Geschenke vorhanden sind. Das heißt Elfen und Rentiere sind weitgehend unabhängig voneinander. Santa Claus, seine Rentiere und die Elfen müssen jeweils durch einen eigenen Thread umgesetzt werden, sodass das unterschiedlichen Eintreten von Szenarien festgehalten werden kann.

2.2 Idee

Das Programm besteht aus drei Threads nämlich: SantaClaus, Rentiere, Elven. Der Thread SantaClaus schläft so lange bis, entweder alle seine benötigten Rentier zurückgekommen sind, oder 3 oder mehrere Elven ihm dringend brauchen. Im Thread Rentier wird modifiziert, dass noch einer zufälligen Dauer, Rentiere zurückkommen und Santa danach aufwecken, sodass sie schon einen Teil der Geschenke an die Kinder liefern können. Im Thread Elven wird ebenso modifiziert, dass nach einer zufälligen Zeit Elven Santa um seine Hilfe bitten. Nach einer bestimmten Anzahl von Elfen wird auch hier Santa geweckt, sodass sie weiter arbeiten können. Das wichtige ist vorallem, dass die Geschenke bis zum Heiligen Abend ausgeliefert werden können und, dass wenn alle Rentier da sind, Santa auf die Hilferufe der Elfen verzichtet und die Geschenke ausliefert.

2.3 Kommandozeilenparameter

Mit Hilfe von Kommandozeilenparameter soll dem Benutzer ermöglicht, werden die Anzahl der Rentier und der Elfen zu definieren. Diese zwei Zahlen legen die maximal benötigte Anzahl fest, um den Thread Santa Claus aus seinem Schlaf zu holen. Ebenso kann der Benutzer auch die Zeit in Stunden bis zu Weihnachten angeben, dadurch kann geprüft werden ob es sich zu geringer Zeit ausgeht die Pakete bis zu Weihnachten auszuliefern. Es werden keine negativen Zahlen, Buchstaben, Sonderzeichen akzeptiert und wenn der Benutzer sonstige Hilfe braucht kann er sich die Information mit dem Kommandozeilenparameter holen.

Santa Claus Problem

Usage: ./santa_claus_problem [OPTIONS]

Options:

-h,--help	Print this help message and exit
-r,--r INT	number of reindeer, which will be needed to fly
-e,--e INT	number of elves, that work in the factory
-t,--t INT	number of hours until christmas
-j,--j TEXT:FILE	write santa, reindeer, elves details in json File
-d,--d	show you a table about the Objects Santa, Elves, Reindeer

2.4 Konsolenausgabe

Die Abarbeitung bzw. Resultate der Threads SantaClaus, Elven und Rentiere werden in der Kommandozeile ausgegeben. Somit kann der Benutzer gut nachvollziehen, welche Tätigkeiten das Programm abgewickelt hat, welche noch bevor stehen und ob es zu Fehler-situation oder Ausnahmen gekommen ist. Angenommen es kommen Rentier zurück oder Elven benötigen Santa's Hilfe so wird auch in der Kommandozeile dies wird mit spdlog festgehalten. Um am Ende einen genauen und übersichtlichen Überblick den Benutzer zu verschaffen, wird mit dem Kommandozeilenparameter -tab eine Tabelle ausgegeben die Zeigt, wie viele Rentiere im Stall sind, wie viele Elfen Hilfe brauchten und wie viele Stunden Santa Claus geschlafen hat.

3 Themenbereiche

Das Projekt umfasst folgende Themengebiete, die unter anderem in folgenden Folien gut dokumentiert sind und von denen ich mir einige Tipps geholt habe:

- 10_processes
- 11_threads
- 12_threads2
- 13_synchronization
- 14_condition_synchronization
- 15_sync_mechanisms
- 16_threadsafe_interfaces
- 17_dist_sync
- 18_task_based_programming
- 19_parallel_programming
- 20_threads_perfmem
- 21_encoding
- 22_data_interoperability
- 23_character_encoding

4 Implementierung

In diesem Kapitel geht es grundsätzlich um die technische Realisierung der Aufgabenstellung. Im folgendem Abschnitt wird auch der Aufbau des Projekts beschrieben. Außerdem enthält dieses Kapitel auch die Dokumentation des Source Codes und wichtige Informationen bezüglich Bibliotheken, die im Projekt verwendet wurde.

4.1 Aufbau

Das Projekt wurde in 2 wesentlich Verzeichnisse eingeteilt, nämlich in source und documentation. In dem Verzeichniss documentation sind alle Unterlagen der theoretisch Arbeit abgelegt worden. Im Verzeichniss source befinden sich alle relevanten Dateien der technischen Umsetzung der Aufgabenstellung. Die drei erwähnten Objekte wurden zu je 3 Module aufgeteilt. Das heißt die Klassendefinition, Konstruktoren und Methodenprototypen von Santa Claus, Elven und Rentiere stehen in der jeweiligen h-Datei. Die Methodendefinitionen, der dazugehörigen Klasse, wurden in der dazugehörigen cpp-Datei kodiert. Zusätzlich Funktionen die für die Umsetzung des Projekts dringend notwendig waren, wurden in dem Modul utils declariert und definiert. Da einige Source Code Abschnitte nicht verständlich und nachvollziehbar sind, wurden alle Variablen, Funktionen, Klassen, Methoden und sonstiges gut dokumentiert.

```
documentation.....enthält alle Dateien der theoretisch Ausarbeitung
├── Logo.png
├── SantaClausProblem_Dokumentation.pdf
└── SantaClausProblem_Dokumentation.tex
source.....enthält alle Dateien der praktischen Ausarbeitung
├── build.....darin befinden sich alle automatisch generiert werden Dateien
│   ├── build.ninja
│   ├── compile_commands.json
│   ├── mesoninfo
│   ├── mesonlogs
│   ├── meson-private
│   ├── santa_claus_problem
│   ├── santa_claus_problem@exe
│   └── santa_problem.json
├── include.....darin befinden sich alle h-Dateien
│   ├── Elves.h
│   ├── Reindeer.h
│   ├── SantaClaus.h
│   └── utils.h
├── src.....darin befinden sich alle cpp-Dateien
│   ├── Elves.cpp
│   ├── Reindeer.cpp
│   ├── SantaClaus.cpp
│   ├── utils.cpp
│   ├── main.cpp
└── meson.build
```


|_meson_options.txt

4.2 Source Codes Dokumentation

In diesem Abschnitt werden einzelne Programmteile beschreiben und genau erläutern, welche Aufgabe sie haben und welche Resultate davon entzogen werden. Es wird auch auf einzelne Grundgedanken, Umsetzungen und Lösungsvarianten daraufeingegangen, um

4.2.1 Elves [Klasse]

```
//Klasse Elven
class Elves{
private:
//Variablen
    SantaClaus *sc;      //Verweis auf das dazugehörige SantaClaus Objekt
    std::mutex &mxe;      //Mutex Objekt
    int elves{0};        //Elfen
    int maxelves{0};      //Elven die benötigt werden um Santa zu wecken
    int elvessum{0};      //alle Elfen die Hilfe benötigten
public:
//Condition Variable
    std::condition_variable elfTex;
//Konstruktor
    Elves(int me, std::mutex& xe):mxe{xe}{
        maxelves = me;
    }
//Methoden
    //Arbeitsablauf der Elfen, Santa wird geweckt wenn Elfen in benötigen
    void tinker();
    //Santa Claus hilft jeden einzelnen Elfen
    void get_Help();
    //gibt die Anzahl der Elfen zurück, die Hilfe brauchen
    int get_Elves();
    //gibt die Maximale Anzahl der Elfen zurück,
    //die benötigt werden um Santa zu wecken*/
    int get_MaxElves();
    int get_SumElves();
    void set_Santa(SantaClaus *s);
};
```

4.2.2 Reindeer [Klasse]

```
//Klassen Rentiere
class Reindeer{
private:
//Variablen
    SantaClaus *sc;           //Verweis auf das dazugehörige SantaClaus Objekt
    std::mutex &mxr;          //Mutex Objekt
    int reindeer{0};          //Rentiere
    int maxreindeer;          //Rentiere die benötigt werden um Santa zu wecken
public:
//Condition Variable
    std::condition_variable reindeerSem;
//Konstruktor
    Reindeer(int mr, std::mutex& xr):mxr{xr}{
        maxreindeer = mr;
    }
//Methoden
    //Rückkunft aller Rentiere aus dem Osten
    void comeback();
    //wenn alle Rentiere da sind, werden sie vom Santa an dem Schlitten angehängt
    void get_Hitched();
    //gibt die Anzahl der Rentier zurück, die zurückgekommen sind
    int get_Reindeer();
    //gibt die Maximale Anzahl der Rentier zurück,
    //die benötigt werden um Santa zu wecken*/
    int get_MaxReindeer();
    //setzt den Verweis, auf das jeweilige SantaClaus Objekt
    void set_Santa(SantaClaus *s);
    //setzt die Anzahl der Renntier auf 0 ->für Debuggen notwendig gewesen
    void reset_Reindeer();
};
```

4.2.3 Santa Claus [Klasse]

```
//Klasse Santa Claus
class SantaClaus{
private:
//Variablen
    Elves &elv;           //Verweis auf das dazugehörige Elven Objekt
    Reindeer &ren;        //Verweis auf das dazugehörige Rentier Objekt
    std::mutex &mxs;      //Mutex Objekt
    double blithelytime{0}; //gesamte Schlafzeit
    bool doaction{false};  //bool Variable, für santaSem.wait
    bool readytofly{false}; //bool Variable, für ren.reindeerSem.notify_one
    bool readytohelp{false}; //bool Variable, für elv.elfTex.notify_one
public:
//Condition Variable
    std::condition_variable santaSem;
//Konstruktor
    SantaClaus(Elves& e, Reindeer& r, std::mutex& xs): elv{e}, ren{r}, mxs{xs}{
    }
//Methoden
    /*Santa schläft und wird geweckt, wenn alle Rentiere zurück
    sind oder Elfen ihm 4brauchen*/
    void sleep();
    //gibt die Zeit aus, die Santa munter war
    int get_Blithelytime();
    //gibt true, false zurück je nachdem wie viele Rentiere zurück gekommen sind
    bool get_Readytofly();
    //gibt true, false zurück je nachdem wie viele Elven Santa benötigen
    bool get_Readytohelp();
    //addiert die Zeit, in der Santa munter war
    void set_Blithelytime(double t);
    //setzt die Variable readytohelp auf false
    void set_Readytohelp();
    //setzt die Variable doaction auf true
    void set_Doaction();
};
```

4.2.4 Zusätzliche Funktionen

4.2.5 Sonstiges

4.3 Verwendete Bibliotheken

4.3.1 CLI11

4.3.2 spdlog

4.3.3 tabulat

4.3.4 json

5 Anwendungsfälle

Im folgendem Abschnitt werden alle Anwendungsfälle, die ein Benutzer durchführen kann, verkörpert. Dabei wird besonders auf die Kommandozeilenparameter eingegangen.

```
./santa_claus_problem
```

Dabei wird das Programm `santa_claus_problem` gestartet. Wobei 9 Rentiere und 3 Elfen gebraucht werden um Santa Claus aus seinem Schlaf zu wecken. Es werden 24 Stunde bis zur Auslieferung der Geschenke gewährleistet

```
./santa_claus_problem -h
```

```
./santa_claus_problem -r 3 -e 5 -t 10
```

```
./santa_claus_problem -r 3 -e 5 -t 10 -d
```

```
./santa_claus_problem -r 3 -e 5 -t 10 -d -e santa_problem.json
```