

# Protokoll von 4. Praktikum

## Betriebssystem - Prof. Dr. Ronald Moore

Praktikantin: Huong Ly Nguyen

MatrikelNr: 772829

Praktikant: Saad Bourbuh

MatrikelNr: 1115826

---

### 1. Vorbereitung

Zuerst haben wir uns an die Erstellung eines eigenen Docker-Images gewagt. Es sollte ein von uns geschriebenes C++ Programm enthalten, welches eine umfangreiche Speichernutzung aufweist. Der folgende Code zeigt das genutzte Programm:

```
cpp
#include <iostream>
#include <new>

struct Node {
    char data[1024];
    Node* next;
};

int main() {
    Node* head = new Node;
    Node* current = head;

    for (int i = 0; i < 1000000; ++i) {
        try {
            current->next = new Node;
            current = current->next;
            // Speicher tatsächlich verwenden
            for (int j = 0; j < 1024; ++j) {
                current->data[j] = j % 256;
            }
        } catch (std::bad_alloc& ba) {
            std::cerr << "bad_alloc caught: " << ba.what() << '\n';
        }
    }
}
```

```

        break;
    }
}
std::cout << "Finished allocating memory\n";
// Gebrauchten Speicher freigeben
while (head != nullptr) {
    Node* next = head->next;
    delete head;
    head = next;
}
return 0;
}

```

Dieses Programm versucht, eine Millionen Knoten zu erstellen.

Das Dockerfile zur Erstellung meines Docker-Images ist folgendermaßen aufgebaut:

```

Dockerfile
FROM debian:latest
MAINTAINER Ly-Saad
WORKDIR /home/saad/Schreibtisch/Prak 4 BS/Example

# Install g++
RUN apt-get update && apt-get install -y g++

COPY main.cpp .

# Compile the program
RUN g++ -o mydocker main.cpp

CMD ["/mydocker"]

```

```
saad@saad-Pc:~/Schreibtisch/Prak 4 BS/Example$ sudo docker build -t node4 .
[sudo] Passwort für saad:
[+] Building 2.4s (10/10) FINISHED
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 277B 0.0s
=> [internal] load metadata for docker.io/library/debian:latest 1.1s
=> [1/5] FROM docker.io/library/debian:latest@sha256:d568e251e460295a874 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 947B 0.0s
=> CACHED [2/5] WORKDIR /home/saad/Schreibtisch/Prak 4 BS/Example 0.0s
=> CACHED [3/5] RUN apt-get update && apt-get install -y g++ 0.0s
=> [4/5] COPY main.cpp . 0.1s
=> [5/5] RUN g++ -o mydocker main.cpp 1.1s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:0028d8ec05a1242c5b98db4c0eea85b14e4be574ca68a 0.0s
=> => naming to docker.io/library/node4 0.0s
```

Build eine Image

## 2. Experimente mit Speicherverwaltung

Nach gründlicher Recherche habe ich herausgefunden, dass man die Ressourcen für einen Docker-Container begrenzen kann, indem man beim Start des Containers die Option `-m` oder `--memory` hinzufügt.

```
=> => naming to docker.io/library/node4 0.0s
saad@saad-Pc:~/Schreibtisch/Prak 4 BS/Example$ sudo docker run --memory=1GB node4
Finished allocating memory
saad@saad-Pc:~/Schreibtisch/Prak 4 BS/Example$ sudo docker run --memory=999MB node4
Finished allocating memory
saad@saad-Pc:~/Schreibtisch/Prak 4 BS/Example$ sudo docker run --memory=50MB node4
saad@saad-Pc:~/Schreibtisch/Prak 4 BS/Example$ sudo docker run --memory=100MB node4
saad@saad-Pc:~/Schreibtisch/Prak 4 BS/Example$ sudo docker run --memory=300MB node4
saad@saad-Pc:~/Schreibtisch/Prak 4 BS/Example$ sudo docker run --memory=800MB node4
Finished allocating memory
saad@saad-Pc:~/Schreibtisch/Prak 4 BS/Example$ sudo docker run --memory=600MB node4
Finished allocating memory
saad@saad-Pc:~/Schreibtisch/Prak 4 BS/Example$ sudo docker run --memory=400MB node4
saad@saad-Pc:~/Schreibtisch/Prak 4 BS/Example$ sudo docker run --memory=500MB node4
saad@saad-Pc:~/Schreibtisch/Prak 4 BS/Example$ sudo docker run --memory=550MB node4
Finished allocating memory
saad@saad-Pc:~/Schreibtisch/Prak 4 BS/Example$ sudo docker run --memory=530MB node4
Finished allocating memory
saad@saad-Pc:~/Schreibtisch/Prak 4 BS/Example$ sudo docker run --memory=510MB node4
saad@saad-Pc:~/Schreibtisch/Prak 4 BS/Example$ sudo docker run --memory=520MB node4
Finished allocating memory
saad@saad-Pc:~/Schreibtisch/Prak 4 BS/Example$ sudo docker run --memory=530MB node4
Finished allocating memory
saad@saad-Pc:~/Schreibtisch/Prak 4 BS/Example$ sudo docker run --memory=515MB node4
Finished allocating memory
saad@saad-Pc:~/Schreibtisch/Prak 4 BS/Example$ sudo docker run --memory=511MB node4
Finished allocating memory
saad@saad-Pc:~/Schreibtisch/Prak 4 BS/Example$ sudo docker run --memory=510MB node4
saad@saad-Pc:~/Schreibtisch/Prak 4 BS/Example$
```

Ich habe meinen Docker-Container mit unterschiedlichen Speicherlimits gestartet und das Verhalten des Programms analysiert. Meine Beobachtungen dabei waren:

Bei einem Limit von 511MB war das Programm in der Lage, alle Knoten zu erstellen, aber bei einem Limit kleiner als 510MB wird es nicht funktioniert.