BLG312-ASSIGNMENT -3-

Emir Kaan Erdoğan- 150200706,

29.05.2023

1 Introduction

In this assignment, we are requested to implement Banker's Algorithm from scratch.
The Banker's algorithm is a resource allocation and deadlock avoidance algorithm in operating systems. It was developed by Mr Dijkstra and named after the banking industry.
The Banker's algorithm is utilized to ensure that a system avoids deadlock by allocating resources to processes in a safe manner.
While implementing the algorithm, first we create the need matrix, and iterate over the processes by checking whether the process can run with the available resources. If it can, we run the process and make it release its allocated resources ,and update the available resources row matrix. If we manage to somehow figure a path out, then we call the system safe. Otherwise, it is unsafe.

2 Code Design

While designing the program, we initially assume that we take 3 txt files named **”allocations.txt”, ”resources.txt” ,and ”requests.txt”** . By reading this files, we first create the resource row matrix, which is 1xm where m is the number of total resources. Then, we read the allocation matrix, which is nxm , where n is the number of total processes ,and fetch the matrix for the allocated resources. Lastly, we read the request.txt file, which returns an nxm matrix as well and refers to the maximum possible request that a process can place. Having completed fetching the data, we merge all the matrices into a matrix store struct and it also contains an integer array ”is waiting” , that is to control while iterating over the processes to clarify whether the process has been already released its allocated resource. At the end, I send the matrix store to the safety checker function ,and provide pseudocode for this function below:

```
int check-unsafe-state(struct matrix* matrix-store, int num-processes, int num-resource)
int *path = (int*)malloc(num-processes * sizeof(int));
int path-counter = 0;
int checker = 1;
int i = 0;
int check-counter = 0;
int infinite-loop-checker = 0;
int first-index = 1;
int loop-counter = 0;
while (1)
if (((i == infinite-loop-checker) or (check-counter == num-processes)) and (first-
index == 0))
break;

if (matrix-store-¿is-waiting[i] == 0)
i += 1;
i = i continue;

int checker-2 = 1;
for (int j = 0; j ¡ num-resource; j++)
if (checker-2 == 0)
checker = 0;
break;
if (matrix-store-¿need-matrix[i][j] ¡= matrix-store-¿resource-matrix[j])
checker = 1;
checker-2 = 1;
else
checker-2 = 0;
if (checker == 1)
first-index = 0;
path[path-counter] = i;
path-counter++;
update-resource-matrix(matrix-store, num-resource, i);
infinite-loop-checker = i;
check-counter++;
i++;
i = (i % num-processes);
for (int i = 0; i ¡ num-processes; i++)
if (matrix-store-¿is-waiting[i] == 0)
continue;
else
return 0;

print-path(path, num-processes)
return 1;
```