Lab Report of Project 6
Banker's Algorithm

Name: Li Dongyue Student Number: 516030910502
Computer System Engineering (Undergraduate), Fall 2018

December 1, 2018

1 Project Introduction

For this project, I will write a program that implements the bankers algorithm discussed in textbook. Customers request and release resources from the bank. The banker will grant a request only if it leaves the system in a safe state. A request that leaves the system in an unsafe state will be denied.

The banker will consider requests from n customers for m resources types. The banker will grant a request if it satisfies the safety algorithm: *we choose a available job to run and iteratively check whether the allocation is satisfiable.*

2 Banker's Algorithm Implementation

The main algorithm, we implement as `CheckAvailable()`. Basically it picks a available job and update the resource need function. If at a certain point, it cannot satisfy the need then we deny the request:

```c
int CheckAvailable(){
  int i, j, count = 0, copy[NUMBER_OF_RESOURCES];
  for(i = 0; i < NUMBER_OF_CUSTOMERS; i++){
    flags[i] = 1;
  }
  for(i = 0; i < NUMBER_OF_RESOURCES; i++){
    copy[i] = available[i];
  }
  while(1){
    for(i = 0; i < NUMBER_OF_CUSTOMERS; i++){
      if(flags[i]){
        for(j = 0; j < NUMBER_OF_RESOURCES; j++){
          if(available[j] < need[i][j])
            break;
        }
        if(j == NUMBER_OF_RESOURCES){
          flags[i] = 0;
          count++;
          for(j = 0; j < NUMBER_OF_RESOURCES; j++)
            available[j] += allocation[i][j];
        }
      }
    }
    if(count == NUMBER_OF_CUSTOMERS){
      for(i = 0; i < NUMBER_OF_RESOURCES; i++)
        available[i] = copy[i];
      return 1;
    }
    if(count == 0)
    {
      for(i = 0; i < NUMBER_OF_RESOURCES; i++)
        available[i] = copy[i];
      return 0;
    }
  }
}
```

banker.c

To test the banker's algorithm, we take users' input to:

- **Resource Request** In this, we first check whether the resources is allocatable. If not, then directly deny the request. Second, we update the resources allocation and run banker's algorithm in `CheckAvailable()`. If check is not satisfiable, then deny the request. Otherwise, accept the request.

```c
void update()
{
  int i, j;
  for(i = 0; i < NUMBER_OF_CUSTOMERS; i++)
  {
    for(j = 0; j < NUMBER_OF_RESOURCES; j++)
    {
      need[i][j] = maximum[i][j] - allocation[i][j];
    }
  }
}

int request_resources(int customer_num, int request[])
{
  int i;
  for(i = 0; i < NUMBER_OF_RESOURCES; i++)
  {
    if(request[i] > maximum[customer_num][i])
    {
      printf("REQUEST EXCEED MAXIMUM\n");
      return 0;
    }
  }
  for(i = 0; i < NUMBER_OF_RESOURCES; i++)
  {
    allocation[customer_num][i] = request[i];
  }
  update();
  return CheckAvailable();
}
```

banker.c

- **Resource Release**: in this, we first check whether the resources is enough to release. If not, then deny the release request. Otherwise, we update the allocation:

```c
void release_resources(int customer_num, int release[])
{
  int i;
  for(i = 0; i < NUMBER_OF_RESOURCES; i++)
  {
    if(release[i] > allocation[customer_num][i])
    {
      printf("RELEASE EXCEED ALLOCATION\n");
      return;
    }
  }
  for(i = 0; i < NUMBER_OF_RESOURCES; i++)
  {
    allocation[customer_num][i] -= release[i];
    available[i] += release[i];
  }
  update();
}
```

banker.c

In `main()`, we iteratively read the user input and run the corresponding functions:

```c
  while (fgets(cmd, 20, stdin))
  {
    if(cmd[0] == '*')
    {
      display();
    }
    else if (cmd[1] == 'Q')
    {
      normalize(cmd, &target_thread, args);
      if(request_resources(target_thread, args))
      {
        printf("REQUEST COMMAND SAFE\n");
      }
      else
      {
        printf("REQUEST COMMAND UNSAFE\n");
      }
    }
    else if (cmd[1] == 'L')
    {
      normalize(cmd, &target_thread, args);
      release_resources(target_thread, args);
    }
    else if(cmd[0] == '!')
    {
      break;
    }
  }
```

banker.c

To compile these files, please input `make`. To run this banker, please input `./banker`.