CSE 321 Operating Systems

Lab Assignment 4

**Total Marks: 10**

## **Assignment Specification: Access Control Simulation**

### **Objective**

This assignment focuses on implementing and comparing two access control models:

* **Access Control List (ACL)  
  Capability-Based Access Control (CBAC)**

You will simulate how these models grant or deny permissions to users based on static access control data structures.

### **Functions**

The program defines a set of users and resources and implements two methods to determine access:

1. **Access Control List (ACL)**: Each resource maintains a list of users and their permissions.
2. **Capability-Based Access Control (CBAC)**: Each user holds a list of resources they can access, with associated permissions.

Permissions are defined as:

* **READ (1)**
* **WRITE (2)**
* **EXECUTE (4)**

You will simulate access requests using hardcoded test cases and print whether each request is **GRANTED** or **DENIED**.

### 

### 

### 

### **Provided Code Structure**

#### **Data Types**

* **Permission**: Enum
* **User**, **Resource**:
  + Name
* **ACLEntry**:
  + Username
  + Permissions (rwx bitmask )
* **ACLControlledResource**
  + Resource
  + ACL Entry[MAX\_USERS]
  + ACL Count
* **Capability**
  + Resource Name
  + Permissions (bitmask)
* **CapabilityUser**:
  + User
  + Capability[MAX\_RESOURCES]
  + Capability Count

#### **Core Functions**

* printPermissions(int perm): Prints human-readable permission names (Ex: “Read”, “Execute”, “Write”)
* hasPermission(int userPerm, int requiredPerm): Checks if a user has the required permissions
* checkACLAccess(...): Checks access via ACL
* checkCapabilityAccess(...): Checks access via CBAC

#### **Sample Setup in main()**

* Users: Alice, Bob, Charlie
* Resources: File1, File2, File3
* ACL permissions for File1
* Capabilities defined for each user
* Predefined access requests are checked using both models

### **Task Requirements**

You must complete the following:

#### **1. Understand the Permission Bitmask Logic**

* Learn how permissions are combined using bitwise OR (|) and checked using bitwise AND (&).
* Understand how access is granted when the user’s permissions include the requested permission.

#### **2. Modify and Expand the Program**

* Add at least **two new resources** and **two new users**.
* Assign both ACL and capability-based access for these new entities.
* Add at least **six new test cases** to simulate access requests using both models.

#### **3. Add New Functions (Optional Enhancement)**

* addACLEntry(...): Dynamically add a new ACL entry.
* addCapability(...): Dynamically add a new capability to a user.
* Use these functions to make the system more dynamic and reduce hardcoded values in main().

### **Sample Output**

Your program should produce output like:

ACL Check: User Alice requests READ on File1: Access GRANTED

ACL Check: User Bob requests WRITE on File1: Access DENIED

ACL Check: User Charlie has NO entry for resource File1: Access DENIED

Capability Check: User Alice requests WRITE on File1: Access GRANTED

Capability Check: User Bob requests WRITE on File1: Access DENIED

Capability Check: User Charlie has NO capability for File2: Access DENIED

### 

### **Complete the Following code**

| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #define MAX\_USERS 3  #define MAX\_RESOURCES 3  #define MAX\_NAME\_LEN 20  typedef enum{  //to do  }Permission;  //User and Resource Definitions  typedef struct{  //to do  }User;  typedef struct{  //to do  }Resource;  //ACL Entry  typedef struct{  //to do  }ACLEntry;  typedef struct{  //to do  }ACLControlledResource;  //Capability Entry  typedef struct{  //to do  }Capability;  typedef struct{  //to do  }CapabilityUser;  //Utility Functions  void printPermissions(int perm){  //to do  }  int hasPermission(int userPerm, int requiredPerm){  //to do  }  //ACL System  void checkACLAccess(ACLControlledResource \*res, const char \*userName, int perm){  //to do  }  //Capability System  void checkCapabilityAccess(CapabilityUser \*user, const char \*resourceName, int perm){  //to do  }  int main(){  //Sample users and resources  User users[MAX\_USERS] = {{"Alice"}, {"Bob"}, {"Charlie"}};  Resource resources[MAX\_RESOURCES] = {{"File1"}, {"File2"}, {"File3"}};  //ACL Setup  //to do  //Capability Setup  //to do  //Test ACL  //to do  //Test Capability  //to do  return 0;  } |
| --- |