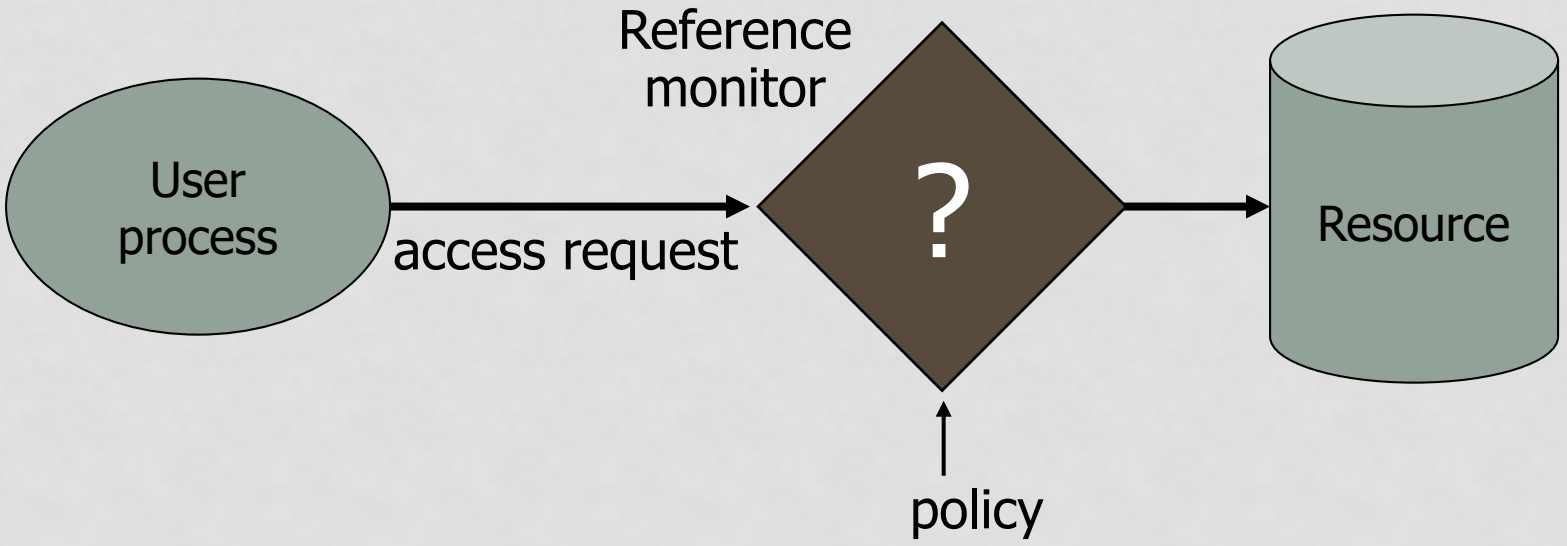
CSC429 – Computer Security

LECTURE 8
ACCESS CONTROL

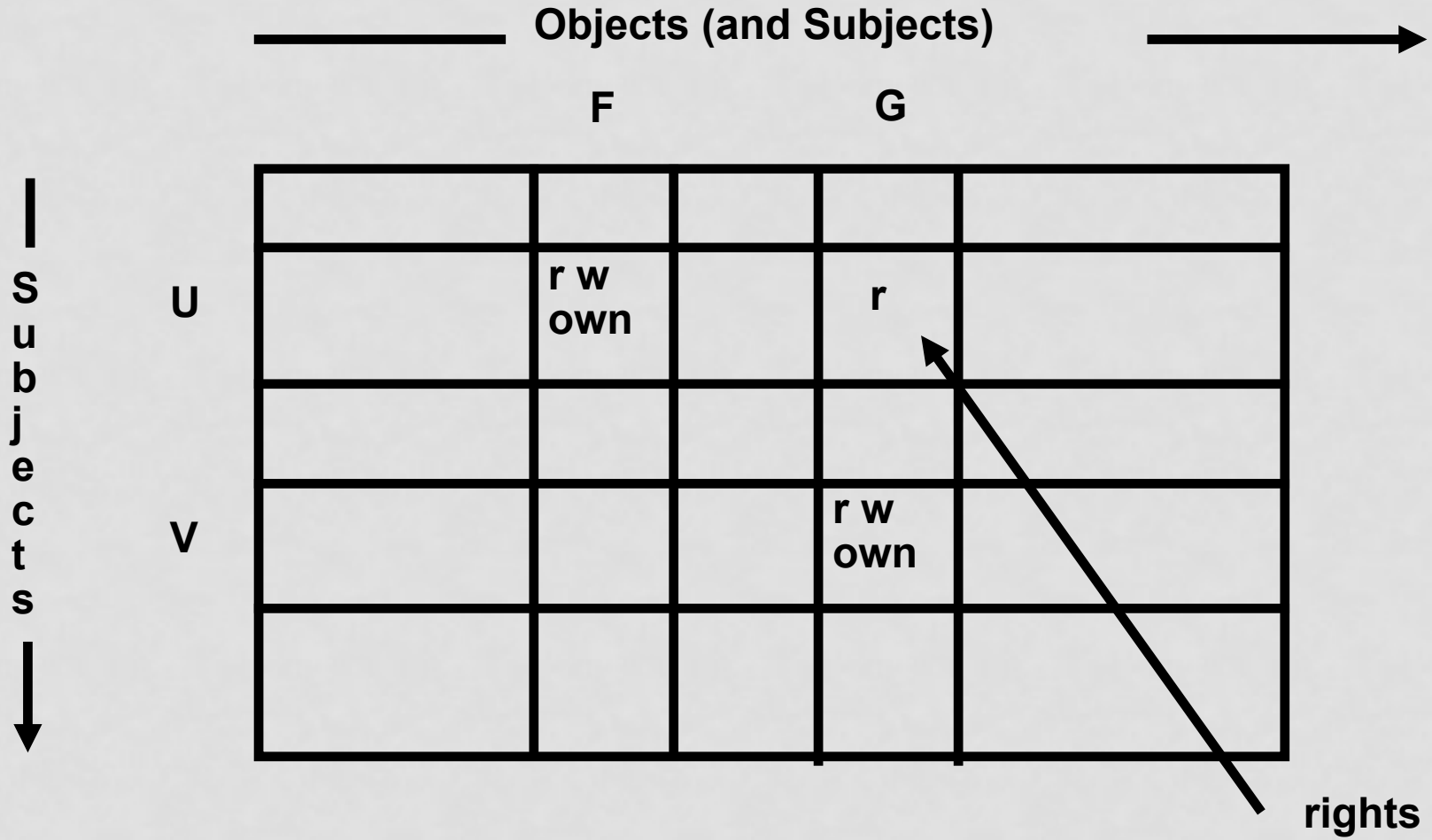
Mohammed H. Almeshekah, PhD
meshekah@ksu.edu.sa

Reference Monitor

- A reference monitor mediates all access to resources
  - Tamper-proof:
  - Complete mediation: control **all** accesses to resources
  - Small enough to be analyzable



Access Control Matrix



Access Control Matrix – Cont'd

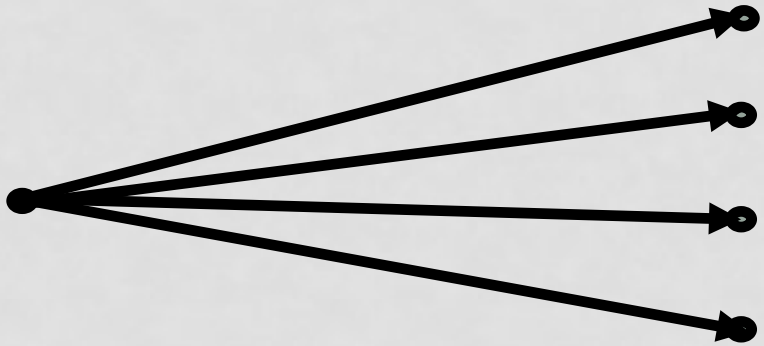
- Basic Abstractions
  - Subjects
  - Objects
  - Rights

- The rights in a cell specify the access of the subject (row) to the object (column)

Principals and Subjects

- A subject is a program (application) executing on behalf of some principal(s)
- A principal may at any time be idle, or have one or more subjects executing on its behalf
- Questions:
  - What are Subjects in Unix?
  - What are Principles in Unix?

User and Principals



USERS

Real World User

PRINCIPALS

Unit of Access Control and Authorization

the system authenticates the human user to a particular principal

User and Principals – Cont'd

- There should be a one-to-many mapping from users to principals
  - a user may have many principals, but
  - each principal is associated with an unique user
- This ensures accountability of a user's actions
- What does the above imply in Unix?

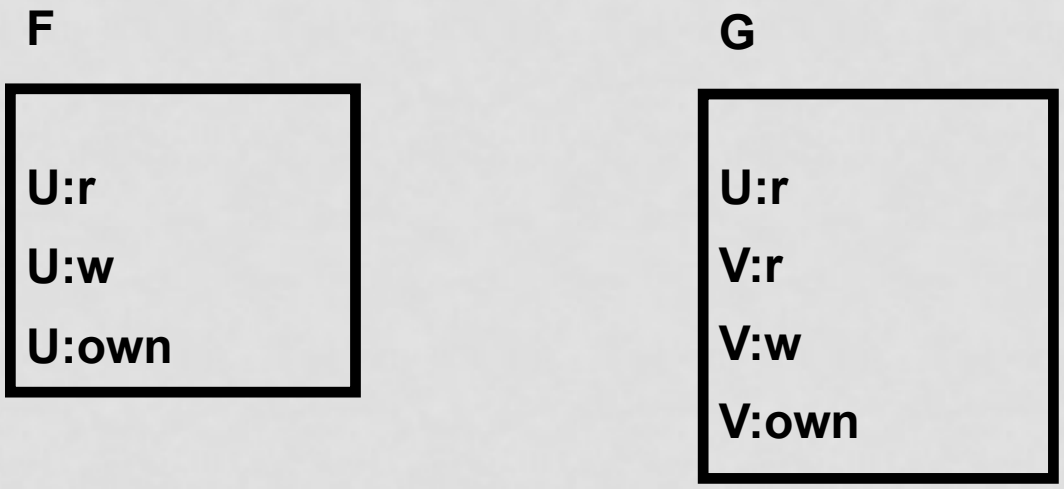
Objects

- An object is anything on which a subject can perform operations (mediated by rights)
- Usually objects are passive, for example:
  - File
  - Directory (or Folder)
  - Memory segment
- But, subjects can also be objects, with operations
  - kill
  - suspend
  - resume

Implementation of the Access Matrix

1. Access Control List (ACL).
2. Capability Lists.
3. Access Control Triplets.

Access Control List (ACL)



each column of the access matrix is stored with the object corresponding to that column

Capability Lists

U	**F/r, F/w, F/own, G/r**
V	**G/r, G/w, G/own**

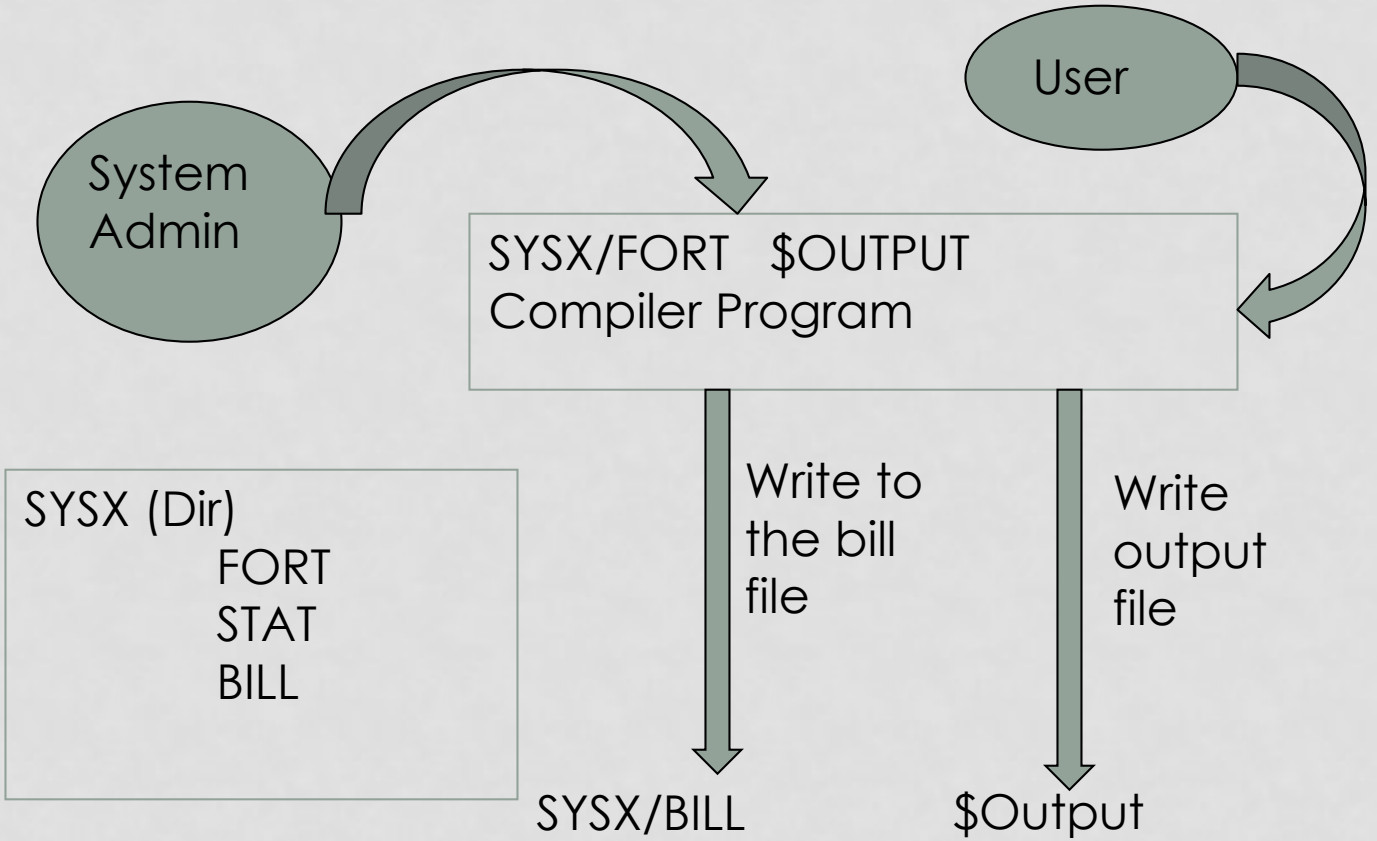
each row of the access matrix is stored with the subject corresponding to that row

Access Control Triplets

Subject	Access	Object
U	r	F
U	w	F
U	own	F
U	r	G
V	r	G
V	w	G
V	own	G

commonly used in relational DBMS

ACL vs. Capabilities – The Confused Deputy Problem



The Confused Deputy Problem

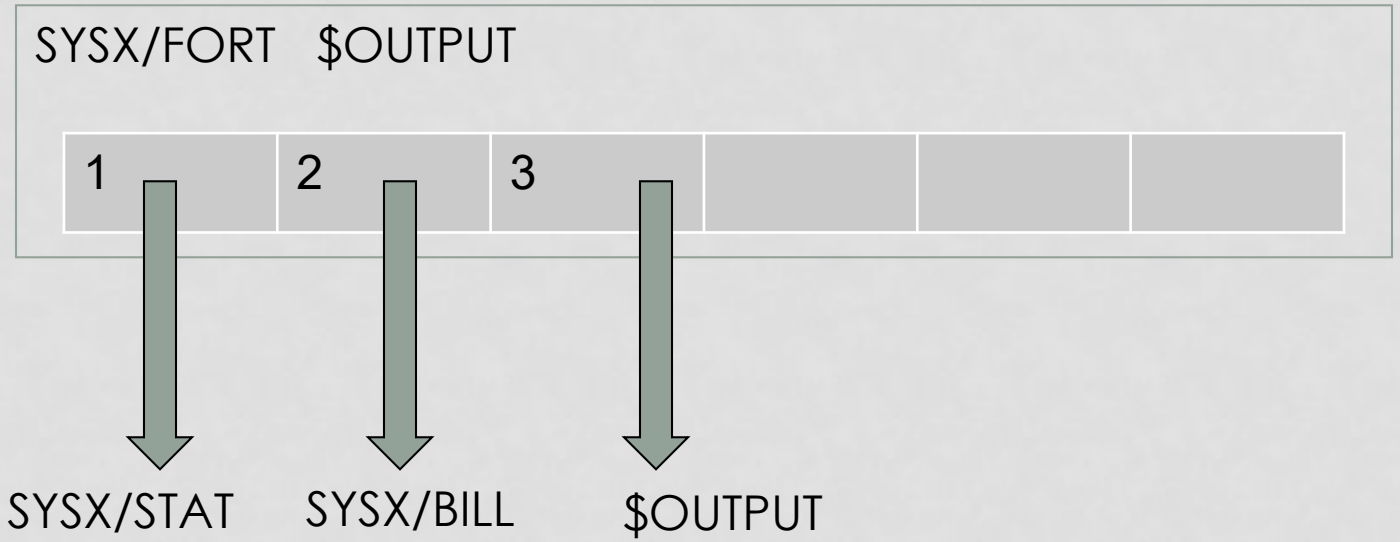
- The compiler runs with authority from two sources
  - the invoker
  - the system admin (who installed the compiler and controls billing and other info)
- It is the deputy of two masters.
- There is no way to tell which master the deputy is serving when performing a write.

Ambient Authority

- Ambient authority means that a user's authority is automatically exercised, without the need of being selected.
  - causes the confused deputy problem
- No Ambient Authority in capability systems

How the Capability Approach Solves the Confused Deputy Problem?

- Invoker must pass in a capability for $OUTPUT, which is stored in slot 3.
- Writing to output uses the capability in slot 3.
- Invoker cannot pass a capability it doesn't have.



Access Control

Unix Access Control – Overview

Quiz Time!

- Mark the following statements as **T**rue or **F**alse:
  1. Processes in Unix-like OSs can only be subjects as they are the only way to operate on objects such as files.
  2. In the Confused Deputy problem, a process is confused because it has have two principles.
  3. Cross-Site scripting (XSS) is one example of the Confused Deputy Problem in web applications.
  4. The Access Control Matrix is one way of implementing access control in OSs.

Basic Concepts of UNIX Access Control

- Each user account has a unique UID
  - The UID 0 means the super user (system admin)
- A user account belongs to multiple groups
- Subjects are processes
  - associated with uid/gid pairs.
- Objects are files

Organization of Objects

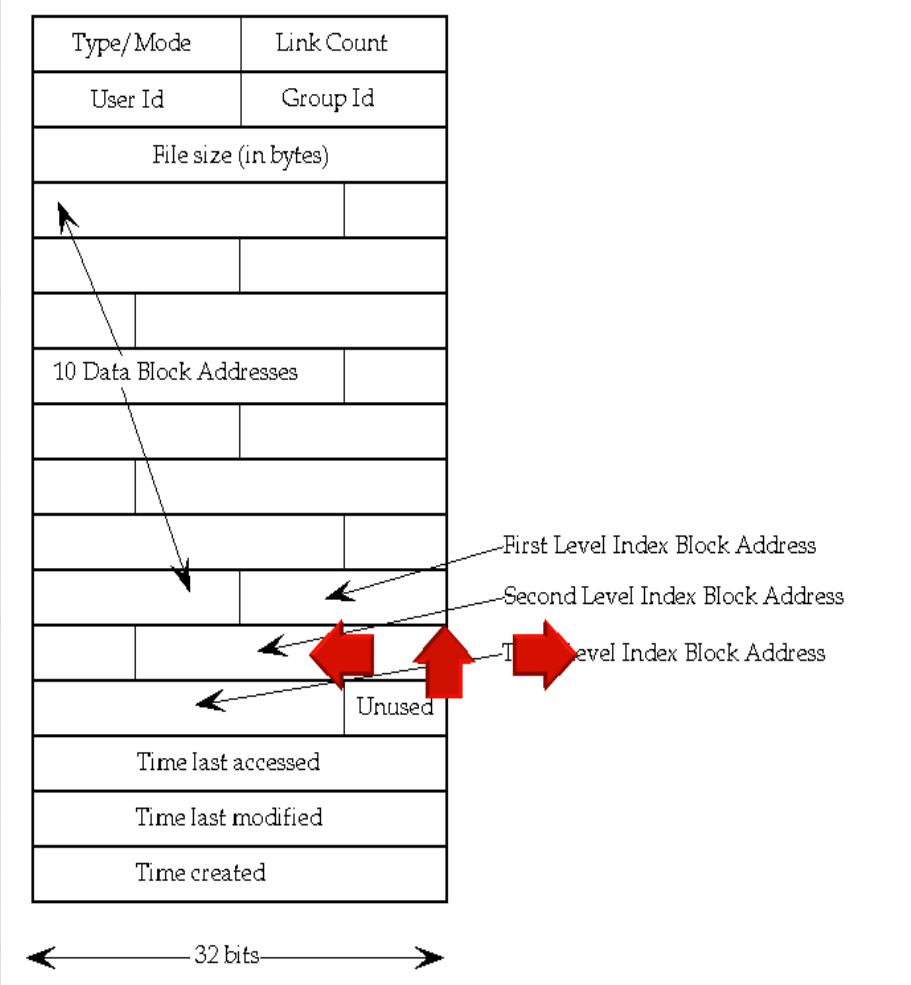
- Almost all objects are modeled as files
  - Files are arranged in a hierarchy
  - Files exist in directories
  - Directories are also one kind of files
- Each object has
  - owner
  - group
  - 12 permission bits
    - rwx for owner, rwx for group, and rwx for others
    - suid, sgid, sticky bits.

Access Control

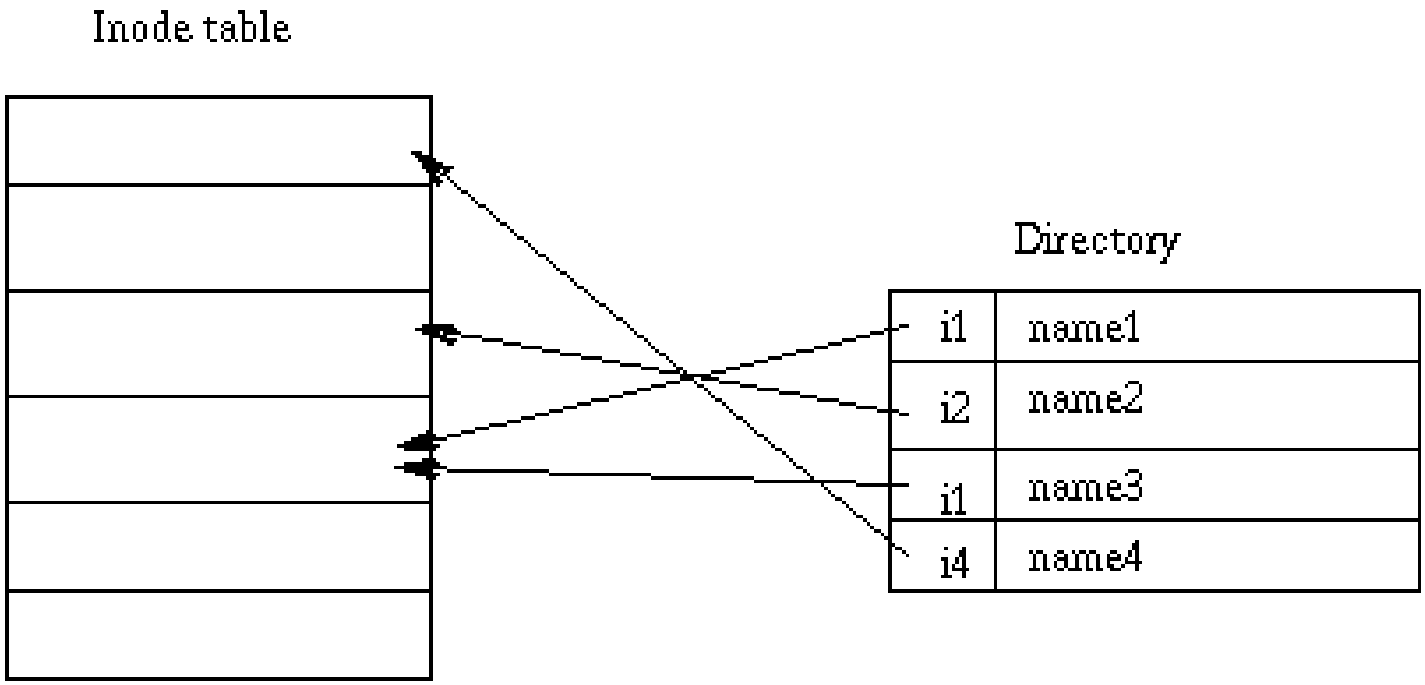
Unix Access Control – Files

Unix i-nodes

- Each files corresponds to an i-node.



Unix Directories



Basic Permissions Bits on Files (Non-directories)

- Read controls reading the content of a file
  - i.e., the read system call
- Write controls changing the content of a file
  - i.e., the write system call
- Execute controls loading the file in memory and execute
  - i.e., the execve system call

Execution of a file

- Binary file vs. script file

- Three questions:
  1. Having execute but not read, can one run a binary file?
  2. Having execute but not read, can one run a script file?
  3. Having read but not execute, can one run a script file?

Permission Bits on Directories

- Read bit allows one to show file names in a directory.
- The execution bit controls traversing a directory
  - does a lookup, allows one to find inode # from file name
  - chdir to a directory requires execution
- Write + execution control creating/deleting files in the directory
  - Deleting a file under a directory requires no permission on the file
- Accessing a file identified by a path name requires execution to all directories along the path.

Some Example

- What permissions are needed to access a file/directory?
  - read a file: /d1/d2/f3
  - write to a file: /d1/d2/f3
  - delete a file: /d1/d2/f3
  - Rename a file: /d1/d2/f3
    to /d1/d2/f4

The Three Sets of Permission bits

- Intuition:
  - if the user is the owner of a file, then the r/w/x bits for owner apply
  - otherwise, if the user belongs to the group the file belongs to, then the r/w/x bits for group apply.
  - otherwise, the r/w/x bits for others apply.
- Can one implement negative authorization, i.e., only members of a particular group are not allowed to access a file?

Other Issues On Objects in UNIX

- Accesses other than read/write/execute
  - Who can change the permission bits?
    - The owner can
  - Who can change the owner?
    - Only the superuser.
- Why do we prevent the owner of a file give the ownership to another user?

Access Control

Unix Access Control – Processes

Unix Subjects and Principles

- Access rights are specified for users/principles (accounts).
- Accesses are performed by processes (subjects).
- The OS needs to know on which users' behalf a process is executing.

Process User ID Model in Modern Unix

- Each process has three user IDs
  - real user ID (ruid) — owner of the process
  - effective user ID (euid) — used in most access control decisions
  - saved user ID (suid)
- and three group IDs
  - real group ID
  - effective group ID
  - saved group ID

Process User ID Model in Modern Unix

- When a process is created by *fork*
  - it inherits all three users IDs from its parent process

- When a process executes a file by *exec*

  if (set-user-ID bit is not set)

  it keeps its three user IDs

  otherwise // set-user-ID bit of the file is set

  euid = ruid

  suid = previous euid

- A process may change the user ids via system calls

Access Control Bits

	suid	sgid	sticky bit
non-executable files	no effect	affect locking (unimportant for us)	not used anymore
executable files	change euid when executing the file	change egid when executing the file	not used anymore
directories	no effect	new files inherit group of the directory	only the owner of a file can delete

The Need for suid/sgid Bits

- Some operations are not modeled as files and require user id = 0
  - halting the system
  - bind/listen on "privileged ports" (TCP/UDP ports below 1024)
  - non-root users need these privileges.

- **setuid**:
  - allows a program running on behalf of one user operate as if it were running as another user (for example root)
  - allows certain processes to have more than ordinary privileges while still being executable by ordinary users

- **How?**
  - effective user identifier takes the value of the owner of the file

Security Problems of Programs with suid/sgid

- These programs are typically setuid root
- Violates the least privilege principle
  - Every program and every user should operate using the least privilege necessary to complete the job
- Why violating least privilege is bad?
- How would an attacker exploit this problem?
- How to solve this problem?

Changing Effective User IDs (EUID)

- A process that executes a set-uid program can drop its privilege; either
  1. drop privilege permanently
     - removes the privileged user id from all three user IDs
  2. drop privilege temporarily
     - removes the privileged user ID from its effective uid but stores it in its saved uid, later the process may restore privilege by restoring privileged user ID in its effective uid