Information Security
CENG418
week-7

- Operating Systems Security -Unix

UNIX Operating System Security

Roadmap

- Authentication
- Basic Concepts in Access Control & UNIX Access Control Overview
- Files in UNIX
- Processes in UNIX

Authentication of Subjects (users)

- Basis for most type of access control and accountability
- Two steps
  - Identification
  - Verification

Means of Authentication

- Traditionally listed as three factors
- Something you ***know***
  - Password, PIN
- Something you ***have***
  - Card, RFID badge
- Something you ***are***
  - Biometrics

Biometrics expanded

- Recently Biometrics (something you are) has been expanded into:
- Something the individual ***is***
  - **Static Biometrics:** Fingerprint, face
- Something the individual ***does***
  - **Dynamic Biometrics:** handwriting, voice recognition, typing rhythm

Password-Based Authentication

The ID determines:

Whether the user is authorized to gain access to a system.

The privileges accorded to the user.

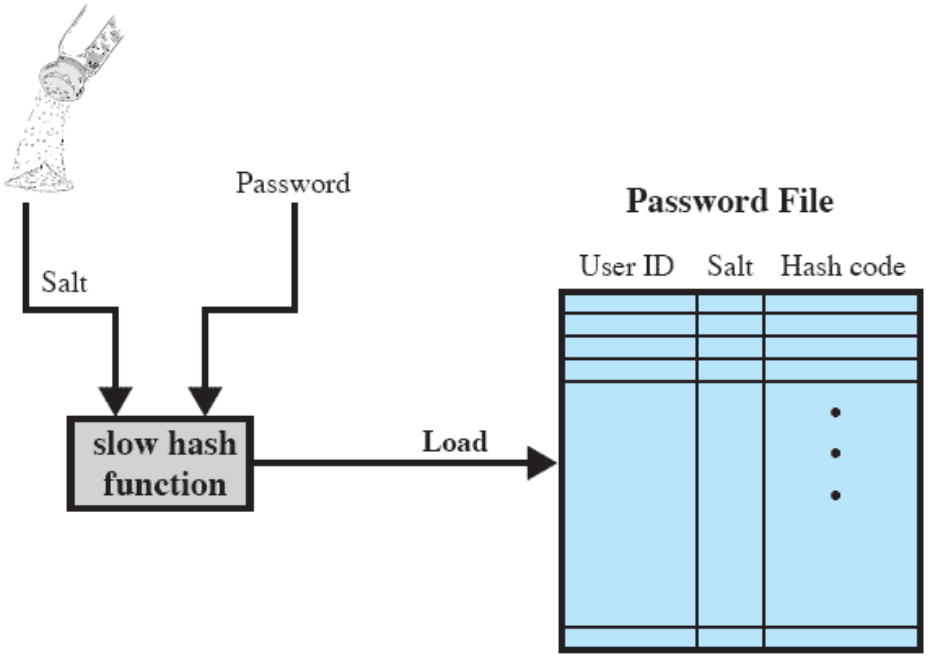
- A few users may have “super user” status that enables them to read files and perform functions that are especially protected by the operating system.
- Some systems have guest or anonymous accounts, have more limited privileges than others.

The discretionary access control.

- For example, by listing the IDs of the other users, a user may grant permission to them to read files owned by that user.

Hashed Passwords

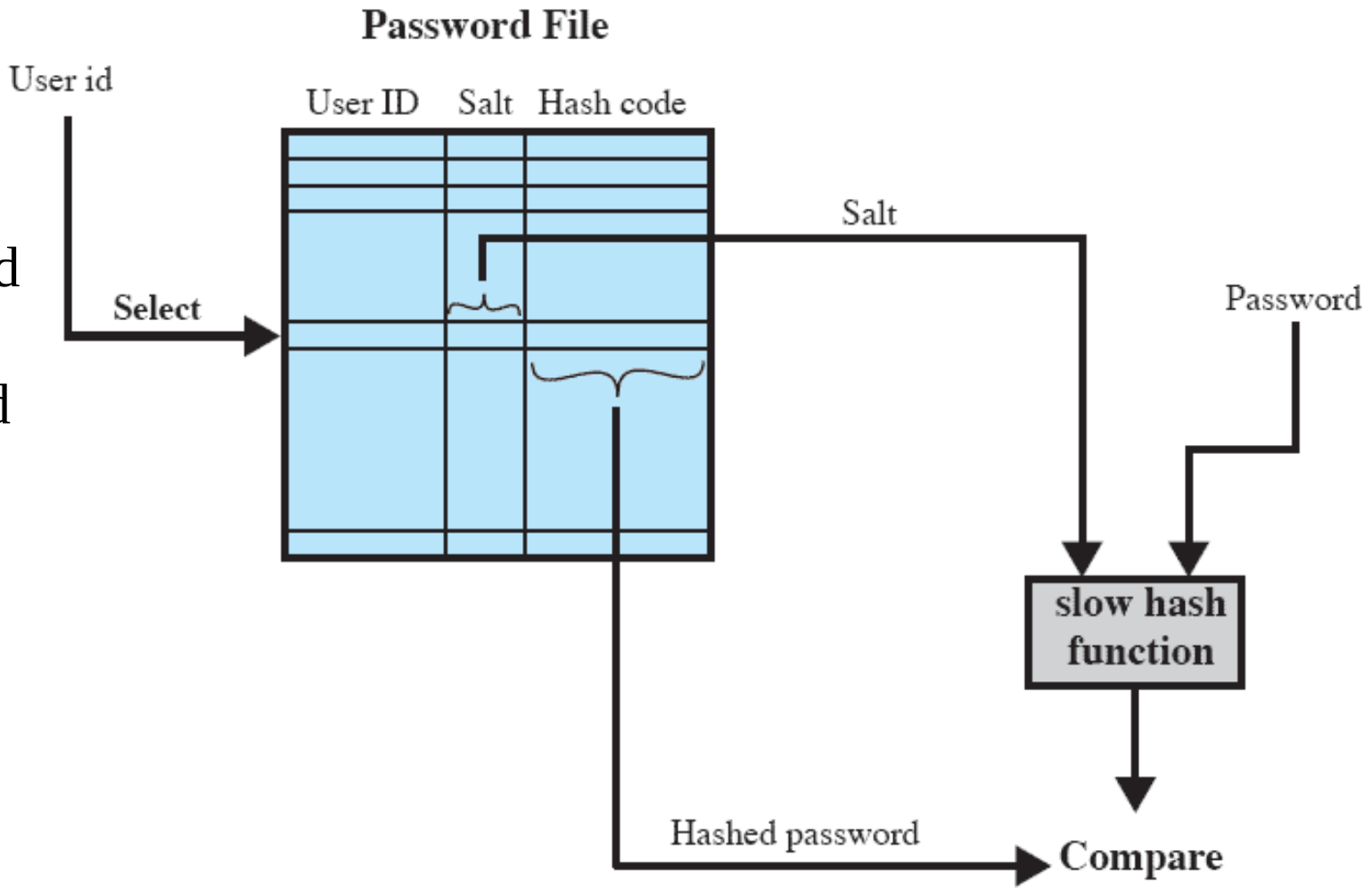
- Widely used technique for storing passwords
- Secure against a variety of cryptanalytic attacks



(a) Loading a new password

UNIX Password Scheme

The user provides an ID and a password



(b) Verifying a password

Salt

- Prevents duplicate passwords from being visible in the password file.
- Greatly increases the difficulty of offline dictionary attacks.
- It becomes nearly impossible to find out whether a person with an account on multiple systems has used the same password for all.

Token-Based Authentication

- Objects that a user possesses for the purpose of user authentication are called tokens.
- Examples include
  - Memory cards
  - Smart cards

Static Biometric Authentication

- Includes
  - Facial characteristics
  - Fingerprints
  - Hand geometry
  - Retinal pattern
- Based on pattern recognition,
  - technically complex and expensive.

Dynamic Biometric Authentication

- Patterns may change
- Includes
  - Iris
  - Signature
  - Voice
  - Typing rhythm

Cost versus Accuracy

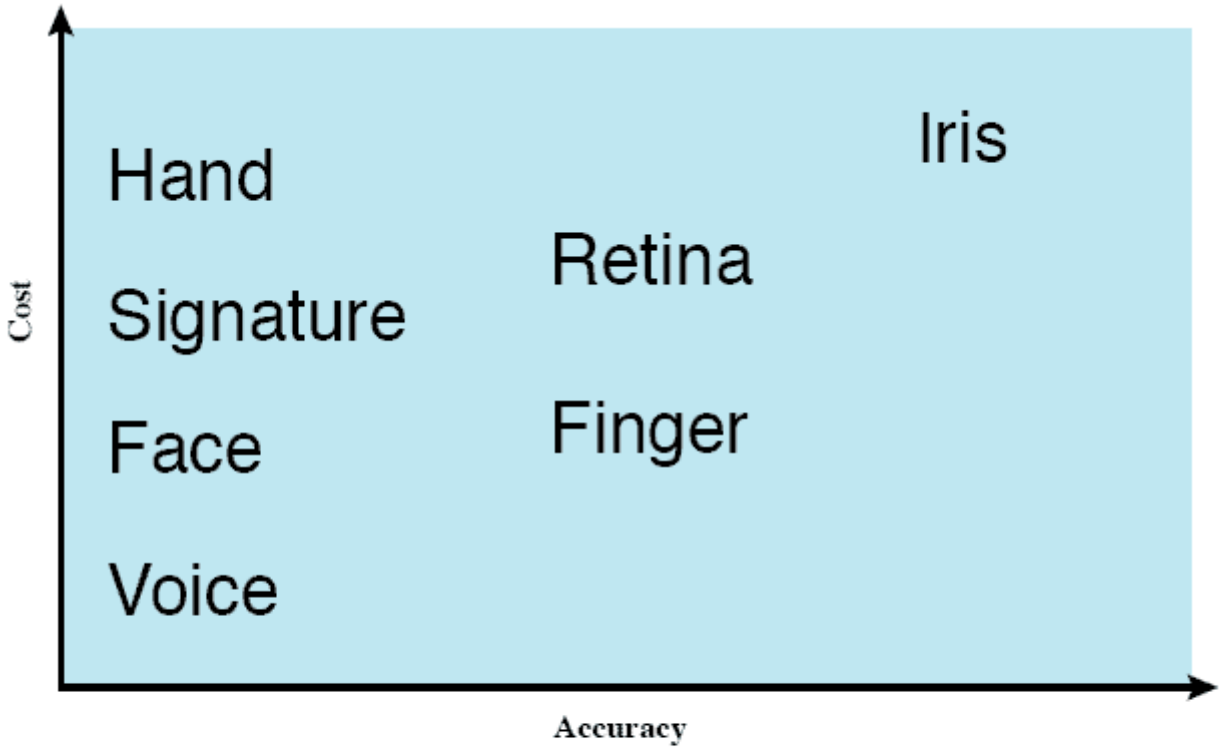


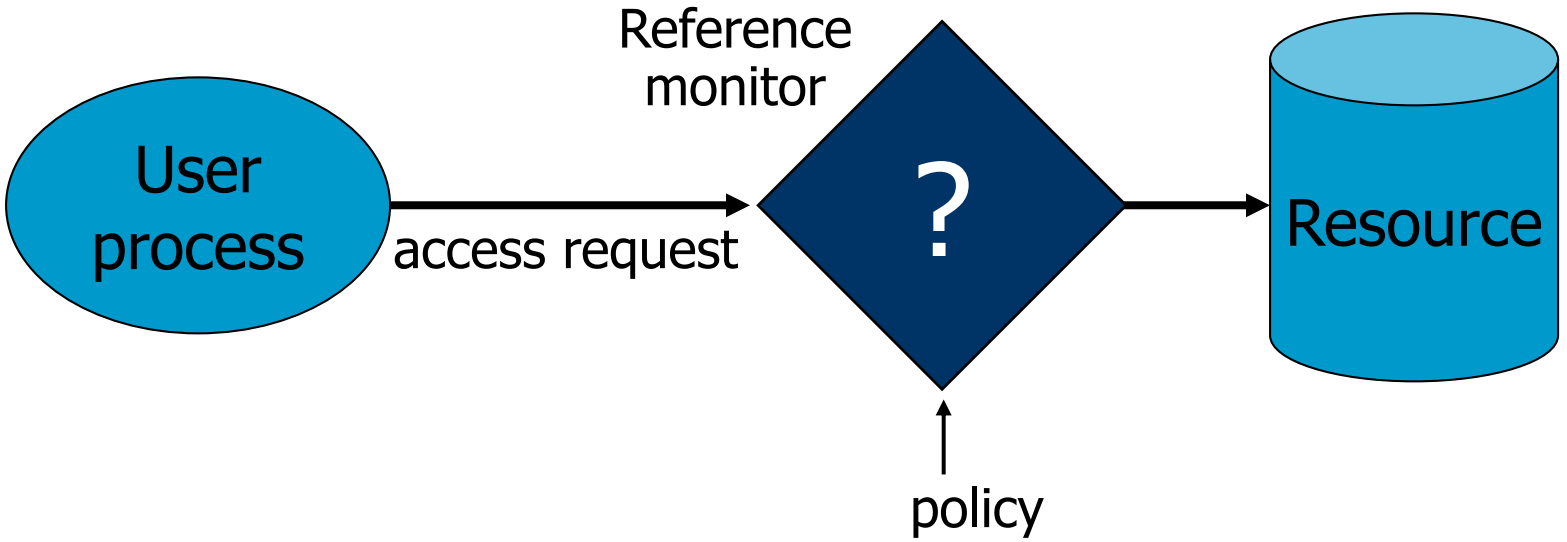
Figure 15.2 Cost Versus Accuracy of Various Biometric Characteristics in User Authentication Schemes.

Roadmap

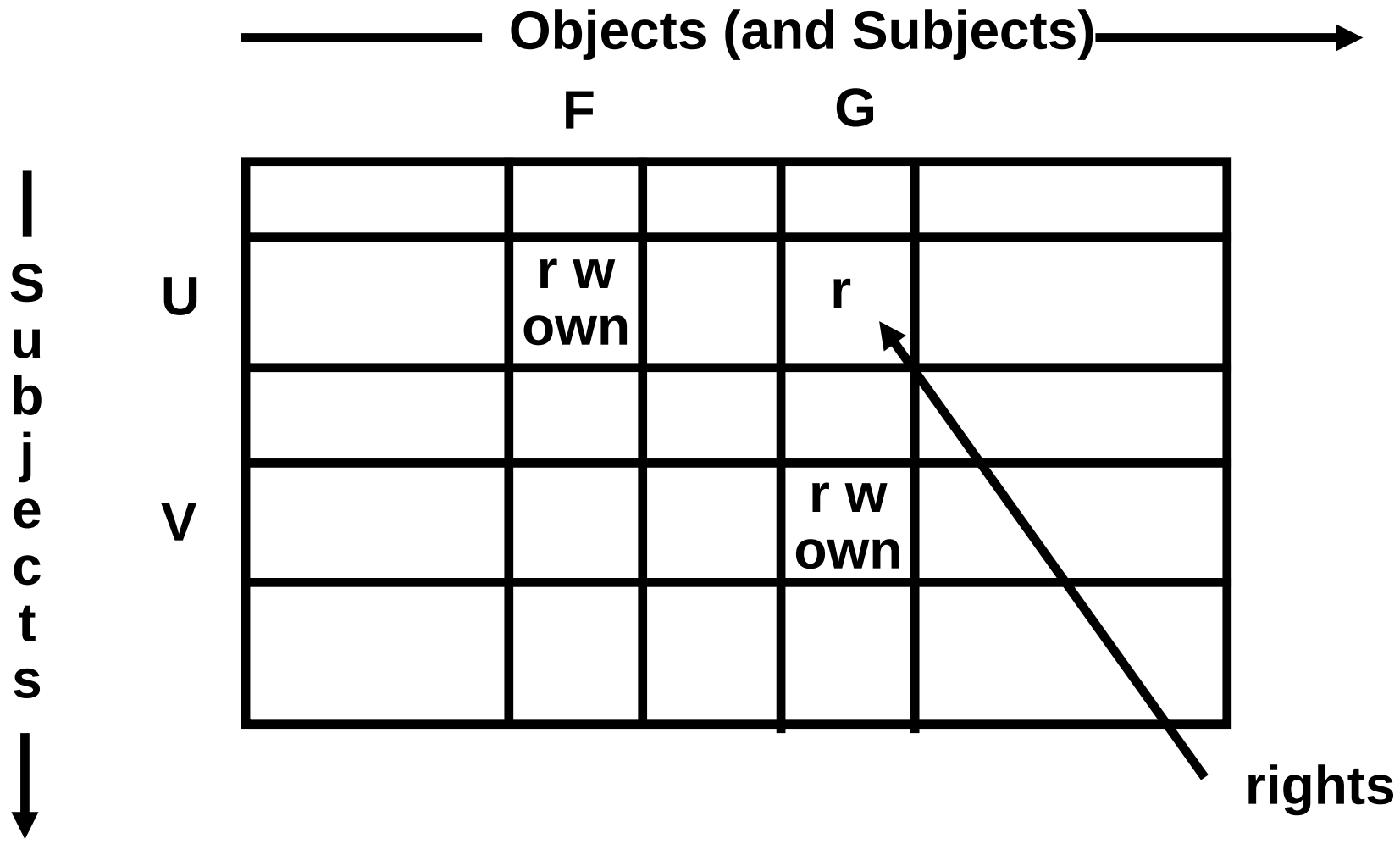
- Authentication
- Basic Concepts in Access Control & UNIX Access Control Overview
- Files in UNIX
- Processes in UNIX

Access control

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



ACCESS MATRIX MODEL



Extended Access Control Matrix

		OBJECTS								
		subjects			files		processes		disk drives	
		S_1	S_2	S_3	F_1	F_2	P_1	P_2	D_1	D_2
SUBJECTS	S_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
	S_2		control		write *	execute			owner	seek *
	S_3			control		write	stop			

* - copy flag set

Figure 15.4 Extended Access Control Matrix

ACCESS MATRIX MODEL

- 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

Principal: The entity in a computer system to which authorizations are granted; thus the unit of accountability in a computer system.

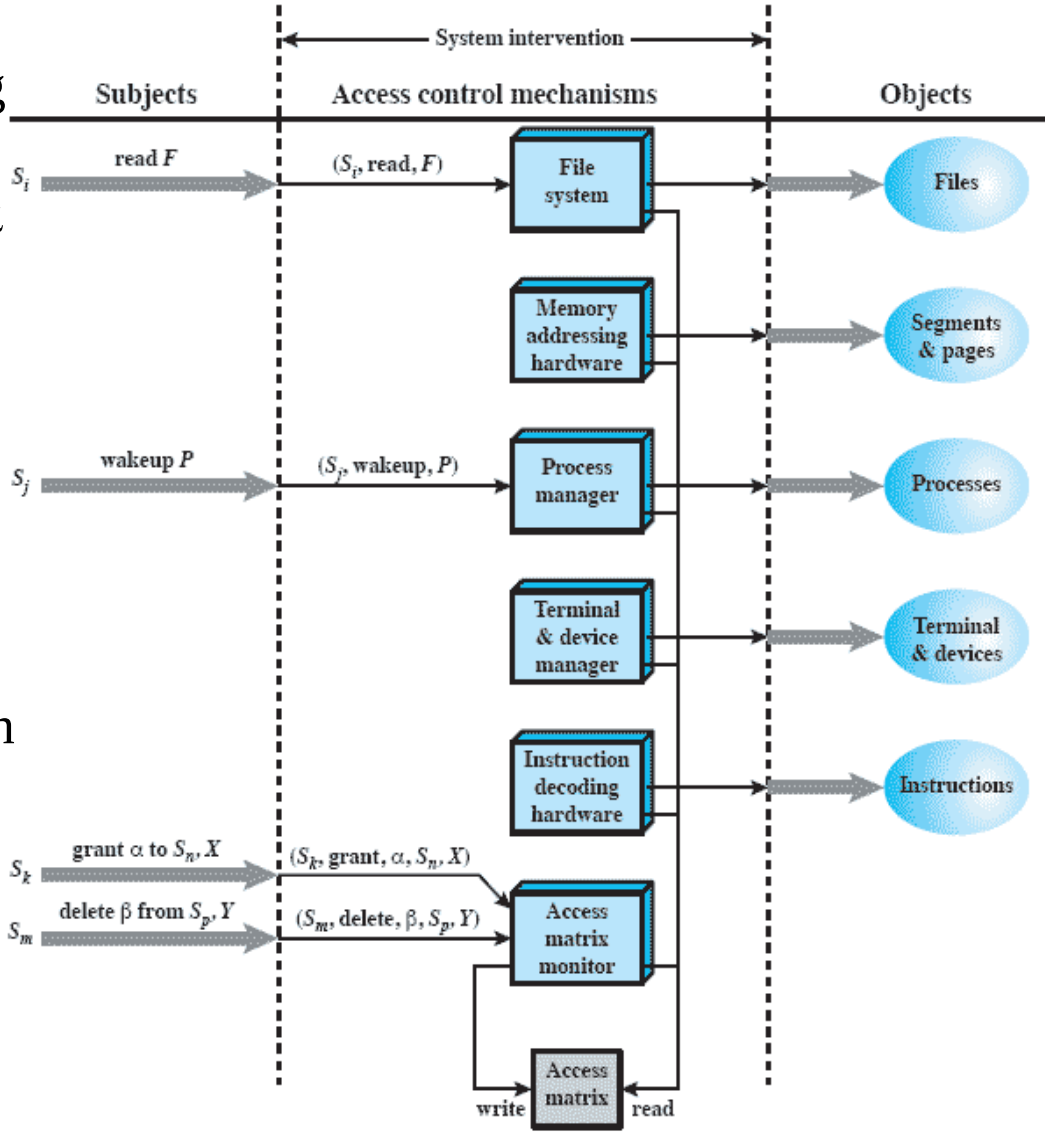
OBJECTS

- 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

Organization of the Access Control Function

An access attempt triggers the following steps:

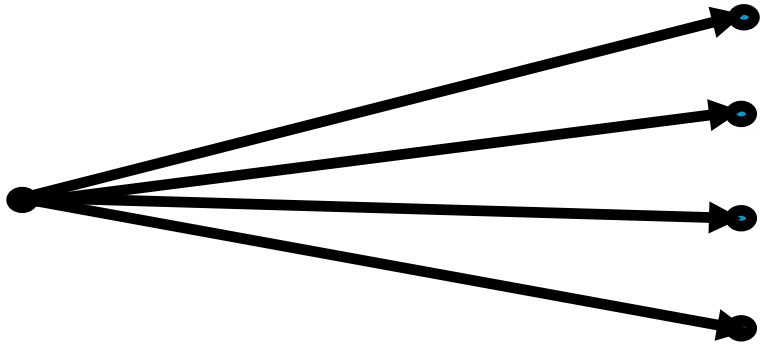
1. A subject S_0 issues a request of type α for object X.
2. The request causes the system to generate a message of the form (S_0,α,X) to the controller for X.
3. The controller interrogates the access matrix A to determine if α is in $A[S_0,X]$.
    - If so, the access is allowed; if not, the access is denied and a protection violation occurs.
    - The violation should trigger a warning and appropriate action.



Basic Concepts of UNIX Access Control: Users, Groups, Files, Processes

- 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

USERS AND PRINCIPALS



USERS

Real World User

PRINCIPALS

Unit of Access Control and Authorization

the system authenticates the human user to a particular principal

USERS AND PRINCIPALS

- 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

Roadmap

- Authentication
- Basic Concepts in Access Control & UNIX Access Control Overview
- Files in UNIX
- Processes in UNIX

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
  - others
  - 12 permission bits
    - rwx for owner, rwx for group, and rwx for others

Unix Permission Triads

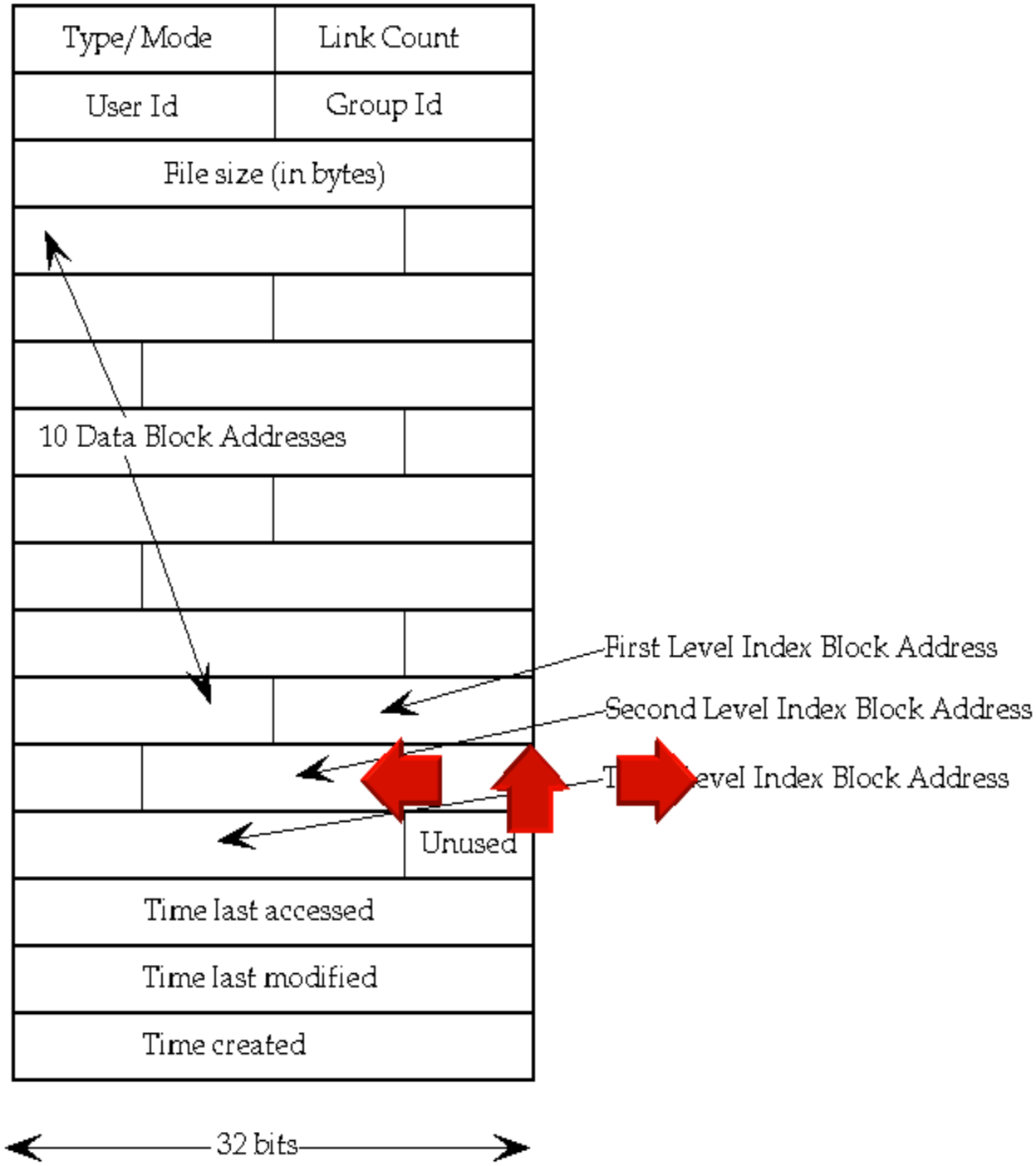
Three permission triads	
first triad	what the owner can do
second triad	what the group members can do
third triad	what other users can do
Each triad	
first character	`r` : readable
second character	`w` : writable
third character	`x` : executable `s` or `t` : setuid/setgid or sticky (also executable) `S` or `T` : setuid/setgid or sticky (not executable)

Example for Symbolic Notation-Unix

Symbolic notation	Numeric notation	English
`----------`	0000	no permissions
`-rwx------`	0700	**read, write, & execute only for owner**
`-rwxrwx---`	0770	read, write, & execute for owner and group
`-rwxrwxrwx`	0777	read, write, & execute for owner, group and others
`---x--x--x`	0111	execute
`--w--w--w-`	0222	write
`--wx-wx-wx`	0333	write & execute
`-r--r--r--`	0444	read
`-r-xr-xr-x`	0555	read & execute
`-rw-rw-rw-`	0666	read & write
`-rwxr-----`	0740	owner can read, write, & execute; group can only read; others have no permissions

UNIX inodes:

Each file corresponds to an inode



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

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

Changing permission behavior

- ***set user ID*:** *setuid*, or SUID mode. When a file with setuid is executed, the resulting process will assume the **effective user ID** given to the owner class. *This enables users to be treated temporarily as root (or another user).*
- ***set group ID,*** *setgid*, or SGID permission. When a file with *setgid* is executed, the resulting process will assume the group ID given to the group class. *When setgid is applied to a directory, new files and directories created under that directory will inherit their group from that directory*.
- The ***sticky*** mode (also known as the *Text* mode). On a directory, the *sticky permission prevents users from renaming, moving or deleting contained files owned by users other than themselves*, even if they have write permission to the directory. Only the directory owner and superuser are exempt from this.

The suid, sgid, sticky 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

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

Roadmap

- Authentication
- Basic Concepts in Access Control & UNIX Access Control Overview
- Files in UNIX
- Processes in UNIX

Subjects vs. Principals

- Access rights are specified for users (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 Systems

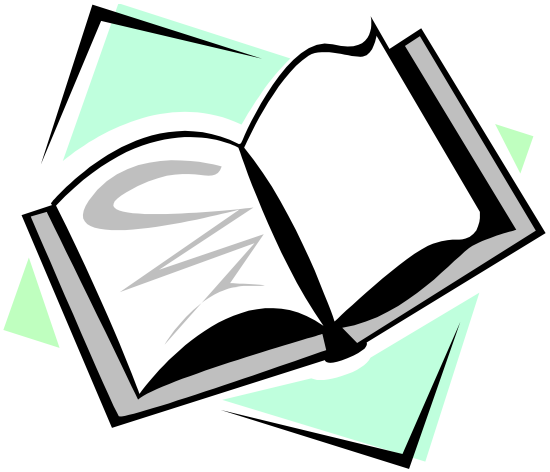
- Each process has three user IDs
  - real user ID (ruid) — owner of the process
  - effective user ID (euid) — used in most access control decisions
  - saved user ID (suid)
- and three group IDs
  - real group ID
  - effective group ID
  - saved group ID

Process User ID Model in Modern UNIX Systems

- 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*
  - it keeps its three user IDs unless the set-user-ID bit of the file is set, *in which case the effective uid and saved uid are assigned the user ID of the owner of the file*
- **A process may change the user ids via system calls**

Readings for This Lecture

- Wiki
  - Filesystem Permissions
- Other readings
  - UNIX File and Directory Permissions and Modes
    - http://www.hccfl.edu/pollock/AUnix1/FilePermissions.htm
  - Unix file permissions
    - http://www.unix.com/tips-tutorials/19060-unix-file-permissions.html



- **Next Week:** ACCESS CONTROL MODELS