Operating Systems Concepts

File Systems (cont.)

CS 4375, Fall 2025

Instructor: MD Armanuzzaman (***Arman***)
marmanuzzaman@utep.edu

November 19, 2025

Summery

- File Allocation Methods
    - Contiguous
    - Linked
    - Indexed
- Free-Space Management
    - Linked List
    - Bit Vector
    - Grouping
    - Counting

Agenda

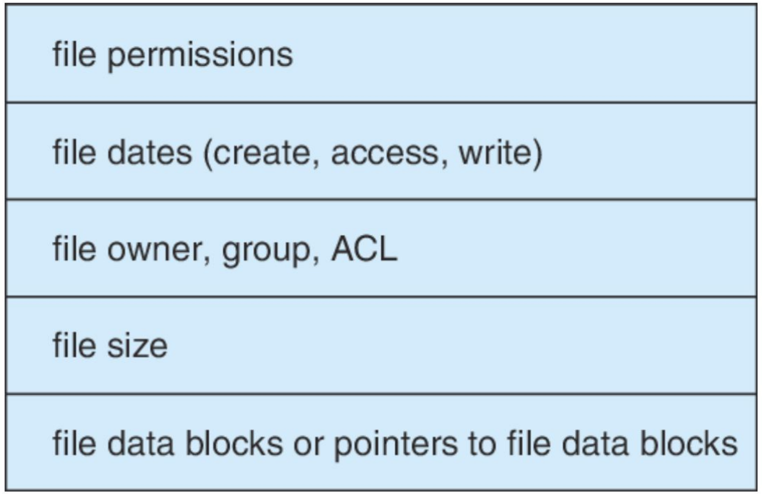
- File Control Block
- Directories
  - Structure
  - Pathname Translation
  - Implementation
- Inode
- Soft Links and Hard Links

File System

- An interface between users and files
- Provides organized and efficient access to data on secondary storage
  - Organizes data into files and directories and supports primitives to manipulate them (*create*, *delete*, *read*, *write*, etc.)
  - Improves I/O efficiency between disk and memory (perform I/O in units of blocks rather than bytes)
  - Ensures confidentiality and integrity of data

File Control Block

- **Inode** in UNIX
- The file system contains file structure via a File Control Block (FCB)
- A typical File Control Block:



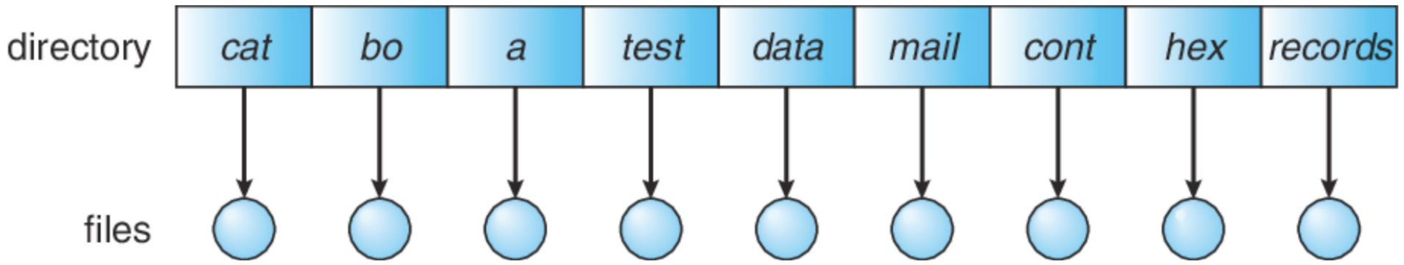
file permissions
file dates (create, access, write)
file owner, group, ACL
file size
file data blocks or pointers to file data blocks

Directories

- Directories provide:
  - A way for users to organize their files
  - A convenient file name space for both users and file systems
- Most file systems support multi-level directories
  - Naming hierarchies (`/, /usr, /usr/local, /usr/local/bin, ...`)
- A directory is typically just a file that happens to contain special metadata
  - Directory = list of (name of file, file attributes)
  - Attributes include such things as:
    - size, protection, location on disk, creation time, access time, …
  - The directory list is usually unordered (effectively random)
    - When you type “ls”, the “ls” command sorts the results for you

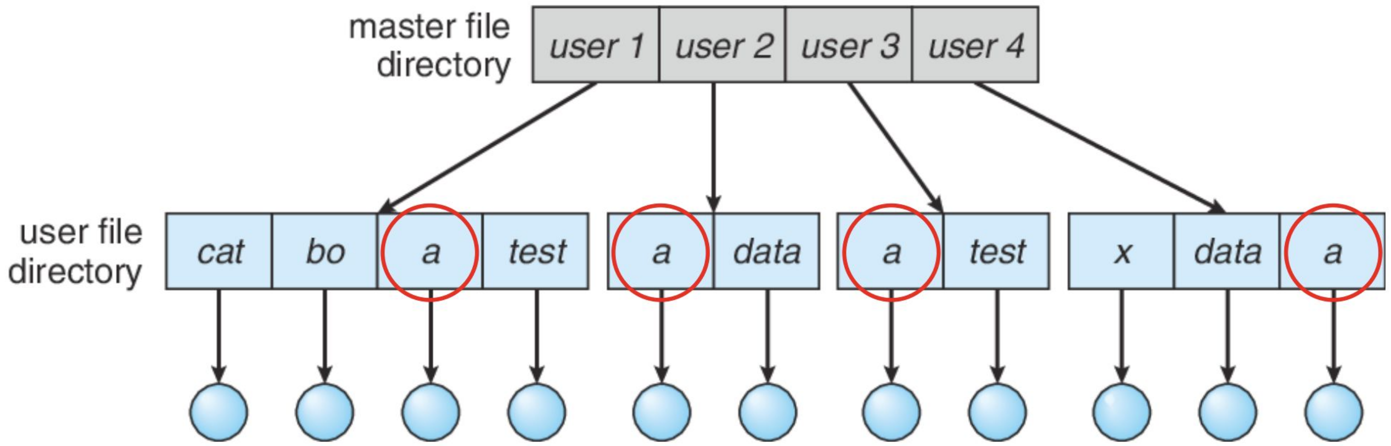
Directories - Structure

- Single-level directory structure
  - Simplest form of logical organization: one global or root directory containing all the files
  - Global namespace: unpractical in multiuser systems
  - No systematic organization, no groups or logical categories of files that belong together.



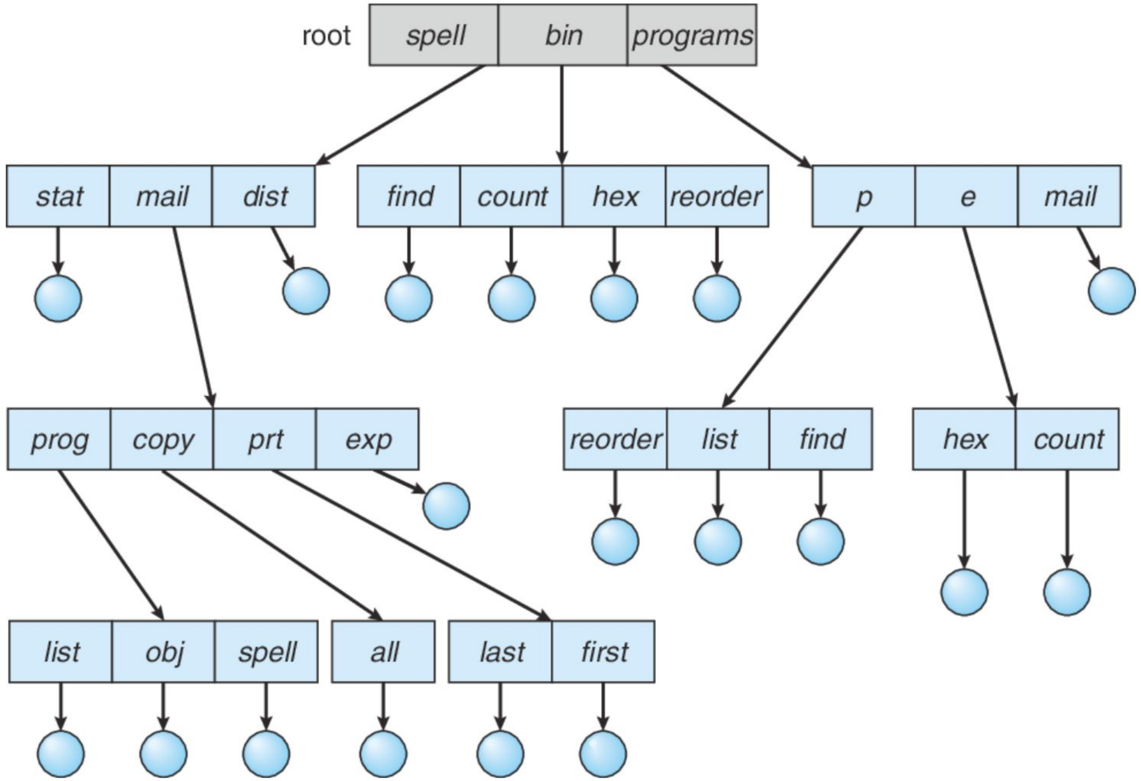
Directories - Structure

- Two-level directory structure
  - In multiuser systems, the next step is to give each user their own private directory
  - Avoids filename confusion
  - Still no grouping. Not satisfactory for users with many files



Directories - Structure

- Tree-level directory structure

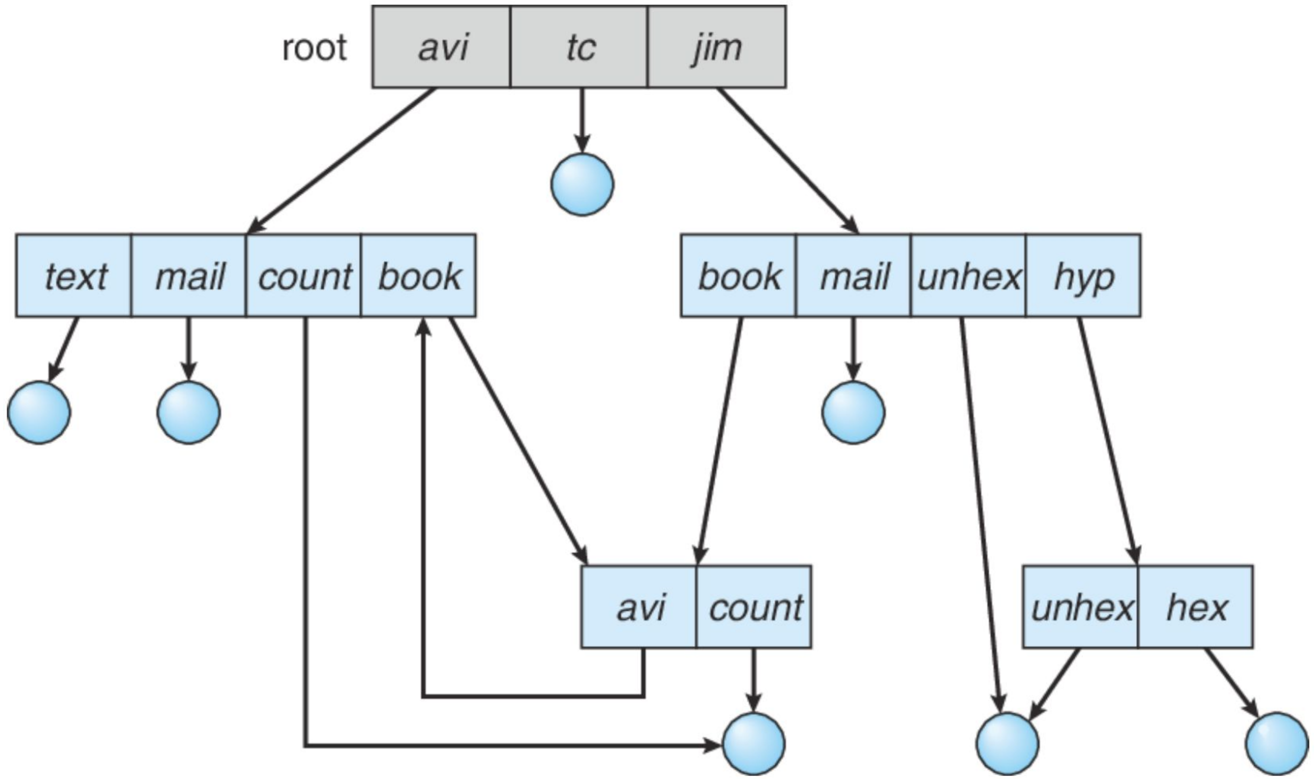


Directories - Structure

- Tree-level directory structure
  - Natural extension of the two-level scheme
  - Provides a general hierarchy, in which files can be grouped in natural ways
  - Good match with human cognitive organization: tendency to categorize objects in embedded sets and subsets
  - Navigation through the tree relies on **pathnames**
    - Absolute pathnames start from the root, example: `/home/Desktop/os-cs4375/xv6/user`
    - Relative pathnames start from current **working directory**, example: `src/lib/user`
    - **Current** and **parent** directories are referred to as `.` and `..` respectively

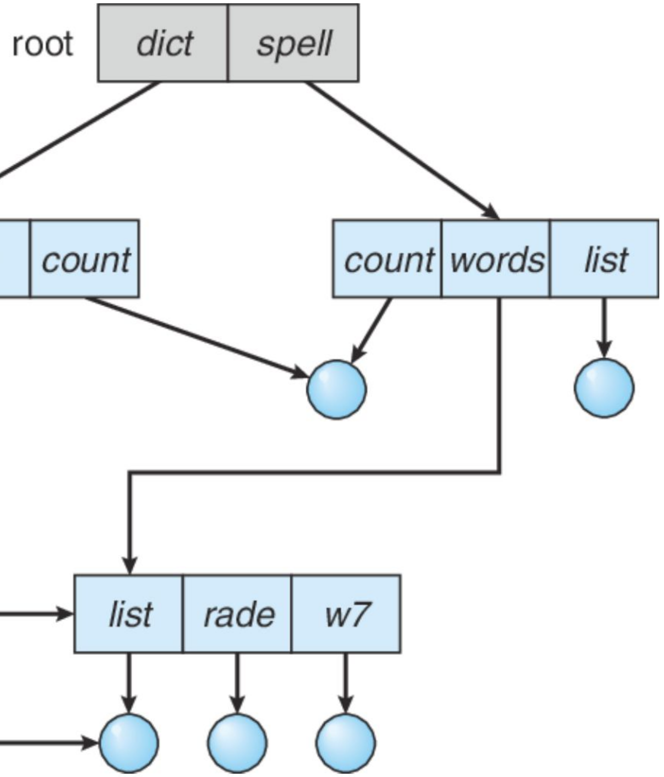
Directories - Structure

- General Graph directory structure



Directories - Structure

- A-cyclic Graph directory structure



Directories - Pathname Translation

- What goes on inside the file system when you want to open “/one/two/three”?

```
fd = open(“/one/two/three”, O_RDWR);
```

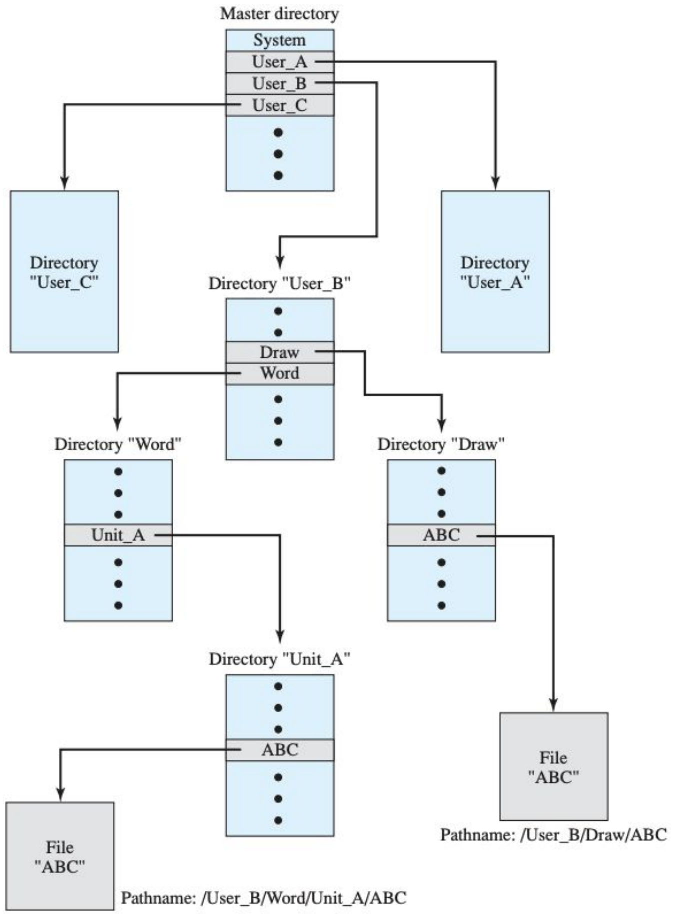
Directories - Pathname Translation

- What goes on inside the file system when you want to open "/one/two/three"?

  `fd = open("/one/two/three", O_RDWR);`

  - Open directory "`/`" (well known, can always find)
  - Search the directory for "`one`", get location of "`one`"
  - Open directory "`one`", search the directory for "`two`", get location of "`two`"
  - Open directory "`two`", search the directory for "`three`", get location of "`three`"
  - Open the file "`tree`"
  - Permissions are checked at each step!
- File system spends lots of time walking down directory paths. this is why **open** is separate from **read/write** (session state)
- OS will cache prefix lookups to enhance performance
  - `/a/b, /a/bb, /a/bbb` all share the "`/a`" prefix

Directories - Implementation



Stallings, W. (2004) Operating Systems: Internals and Design Principles (5th Edition).

Directories - Implementation - UNIX Directories

- Directory is a special file that contains list of names of files and their inode numbers
- To see contents of a directory:

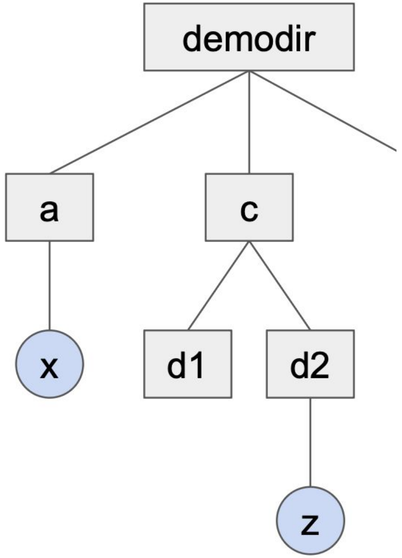
```
$ls -1ia .
9535554 .
9535489 ..
9535574 .bash_history
9535555 bin
9535584 .emacs.d
9535560 grading
9535803 hw1
9535571 test
9535801 .viminfo
```

Directories - System View

- User view vs system view of directory tree
  - Representation with “dirlists (directory files)”
- The real meaning of “A file is in a directory”
  - Directory has a link to the inode of the file
- The real meaning of “A directory contains a subdirectory”
  - Directory has a link to the inode of the subdirectory
- The real meaning of “A directory has a parent directory”
  - “..” entry of the directory has a link to the inode of the parent directory

Example

User view:



Please show the system representation

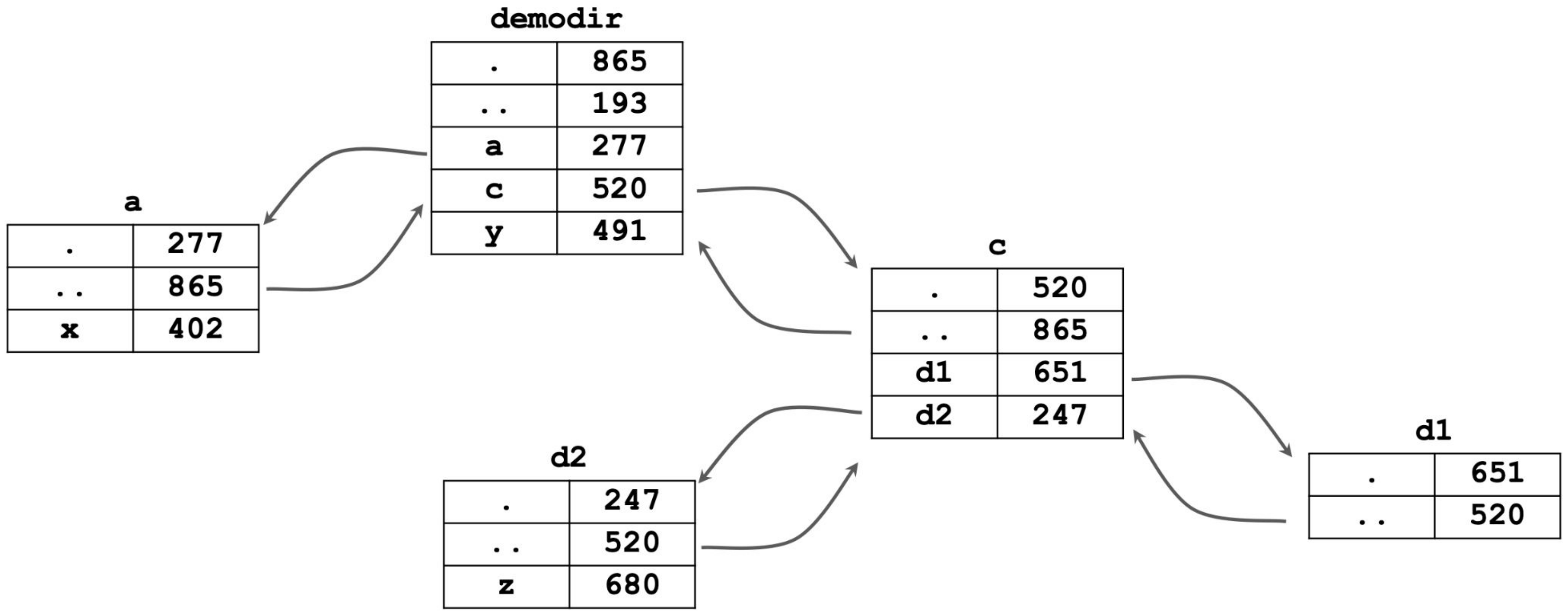
(system view) of this directory tree

Inode listing of directory tree:

```
$ls -iaFR  demodir:
865 .  193 ..  277 a/  520 c/  491 y
demodir/a:
277 .  865 ..  402 x
demodir/c:
520 .  865 ..  651 d1/  247 d2/
demodir/c/d1:
651 .  520 ..
demodir/c/d2:
247 .  520 ..  680 z
```

Example

System view of this directory tree



demodir

.	865
..	193
a	277
c	520
y	491

a

.	277
..	865
x	402

c

.	520
..	865
d1	651
d2	247

d2

.	247
..	520
z	680

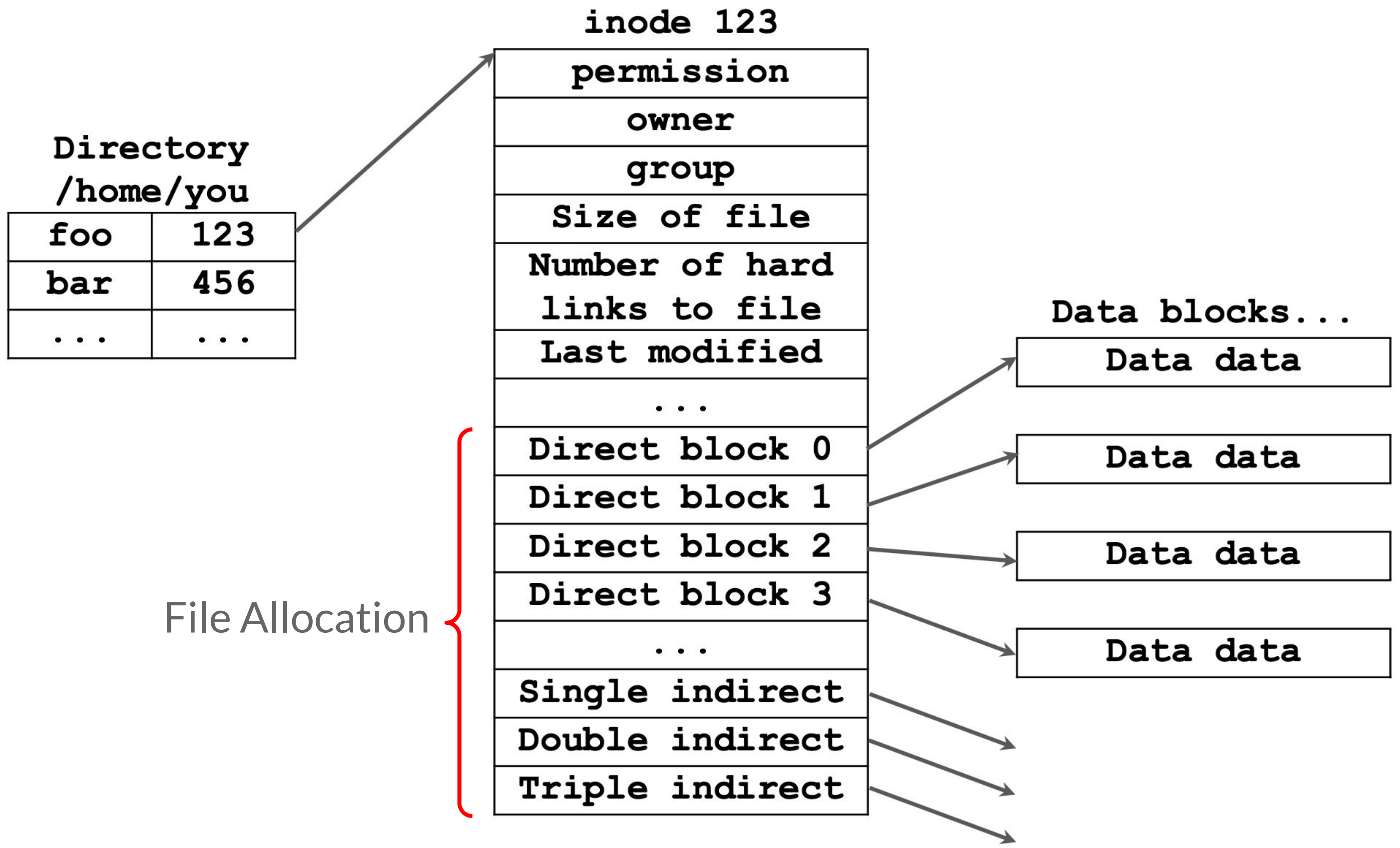
d1

.	651
..	520

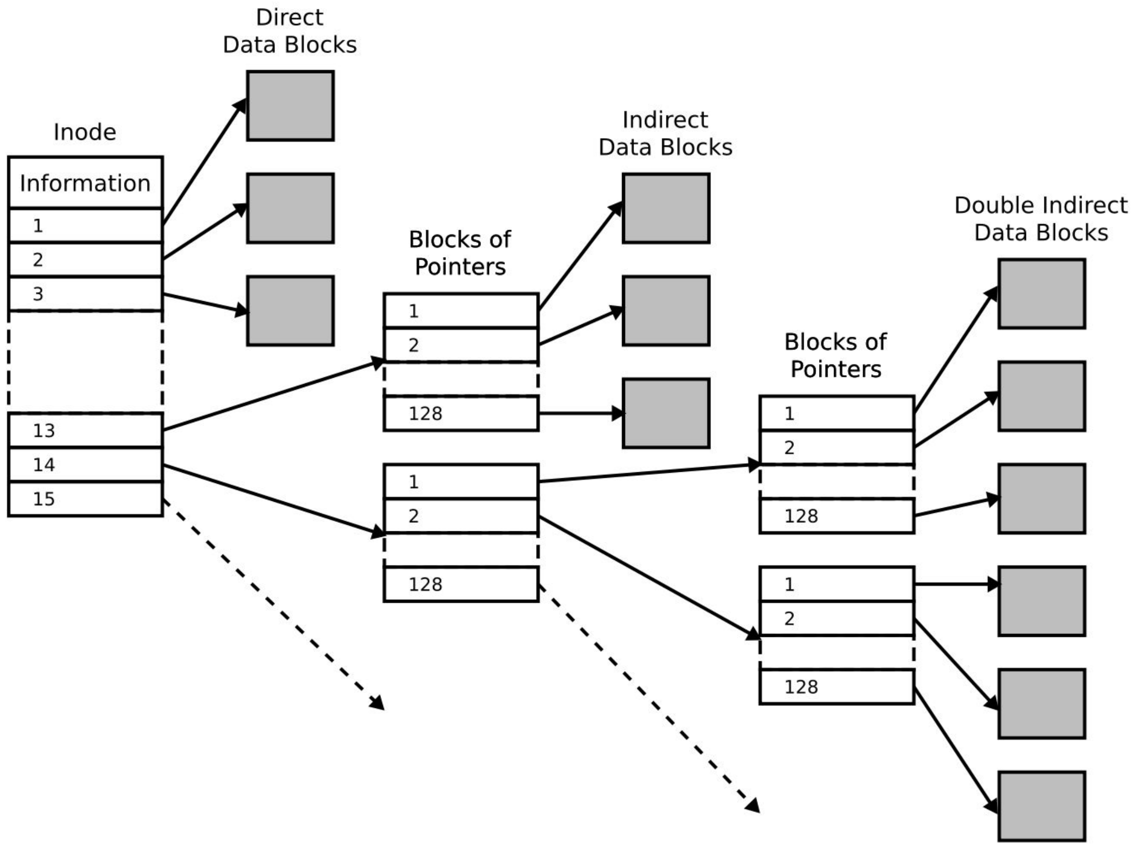
Inode

- **Inode** = structure maintaining all metadata about a file (or directory), except for name
- **Inode number** = unique ID of inode
- One or more file names can point (link) to the same inode
- Why do we need inode numbers? Can't we use absolute paths as IDs?
- Where do we store file names?
  - Inode numbers provide location independence
  - File names are stored in directory entries, in the form of
    - pair (name, inode number)
  - A directory is just a file, whose contents is a list of directory/file entries

Inode


Directory
/home/you
foo
123
bar
456
...
...
inode 123
permission
owner
group
Size of file
Number of hard links to file
Last modified
...
Direct block 0
Direct block 1
Direct block 2
Direct block 3
...
Single indirect
Double indirect
Triple indirect
File Allocation
Data blocks...
Data data
Data data
Data data
Data data

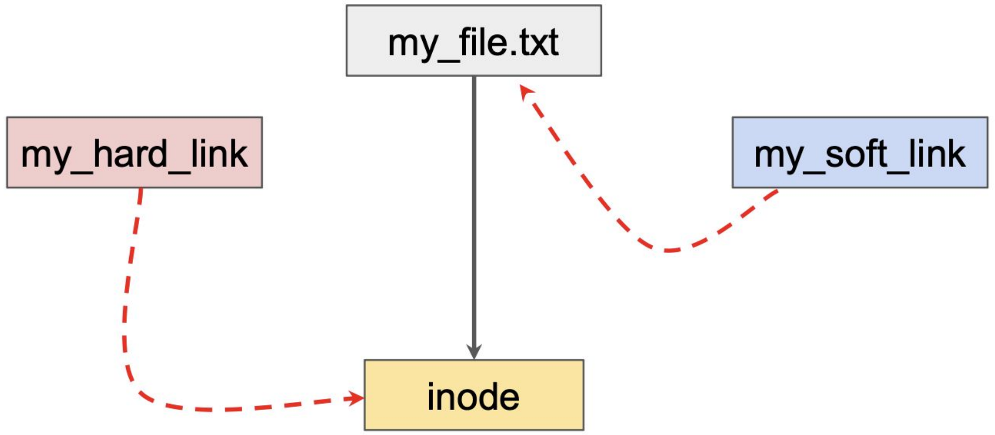
Inode



From:
https://en.wikipedia.org/wiki/Inode_pointer_structure

Soft Link vs Hard Link

- A hard link creates another file with a link to the same underlying inode.
- A soft link (symbolic link) is a link to another name in the file system.



Soft Link vs Hard Link - Link Counts

- The kernel records the number of links to any file/directory.
- The **link count** is stored in the inode.
- The **link count** is a member of **struct stat** returned by the stat system call.

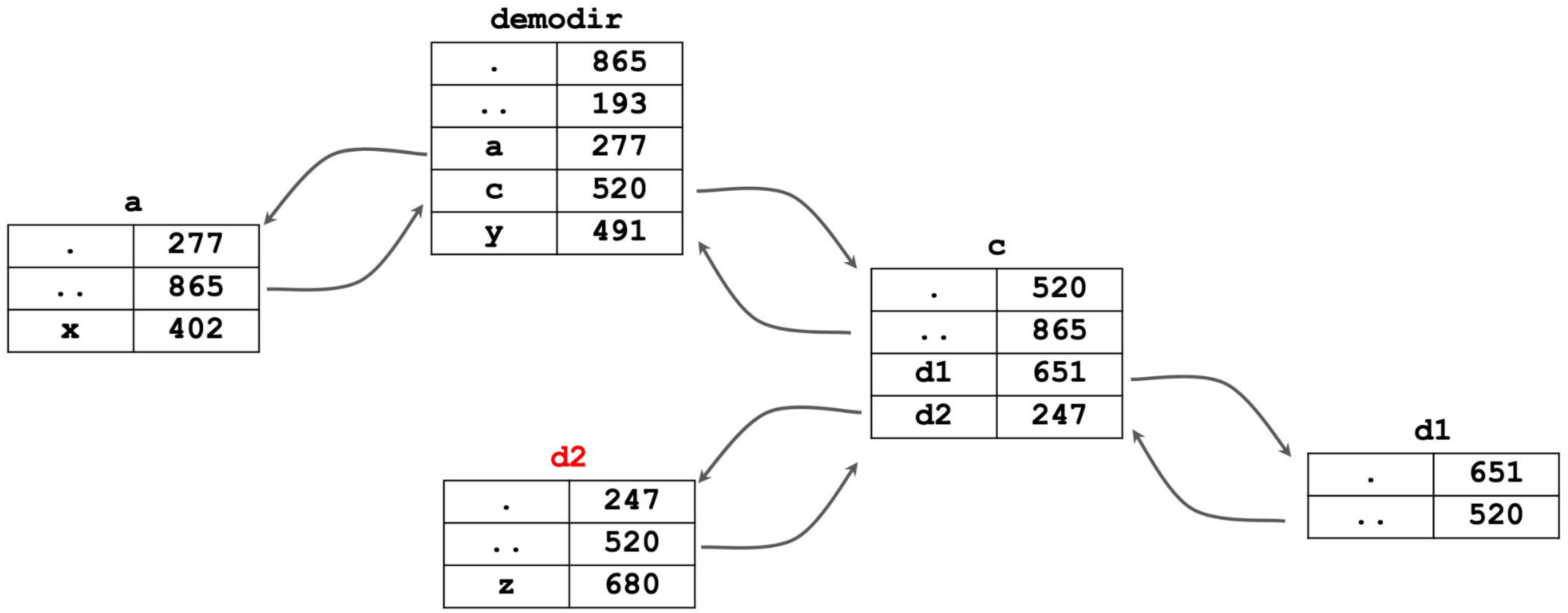
Soft Link vs Hard Link

- What will happen to the directory tree?

```
$ cp demodir/a/x demodir/c/xcopy

$ ln demodir/a/x demodir/c/d1/xlink

$ ln -s demodir/a/x demodir/c/d1/slink

$ mv demodir/y demodir/a/y
```

Implementing “pwd”

How do we get the full path of current directory? (Assume working directory is “d2”)



demodir

.	865
..	193
a	277
c	520
y	491

a

.	277
..	865
x	402

c

.	520
..	865
d1	651
d2	247

d2

.	247
..	520
z	680

d1

.	651
..	520

Implementing “pwd”

```
1.  “.” is 247
    cd ..
2.  247 is called “d2”
    “.” is 520
    cd ..
3.  520 is called “c”
    “.” is 865
    cd ..
4.  865 is called “demodir”
    “.” is 193
    cd ..
```

Acknowledgements

- “Operating Systems Concepts” book and supplementary material by A. Silberschatz, P. Galvin and G. Gagne
- “Operating Systems: Internals and Design Principles” book and supplementary material by W. Stallings
- “Modern Operating Systems” book and supplementary material by A. Tanenbaum
- R. Doursat and M. Yuksel from University of Nevada, Reno
- Farshad Ghanei from Illinois Tech
- T. Kosar and K. Dantu from University at Buffalo

Announcement

- MITRE eCTF team competition 2026
    - *T3S-UTEP*

What is a “CTF”?

- CTF stands for “capture the flag”
- Flags are strings of text, like `98dt2n29ncq23f`
- During CTF competitions, competitors find flags by solving puzzles or hacking systems

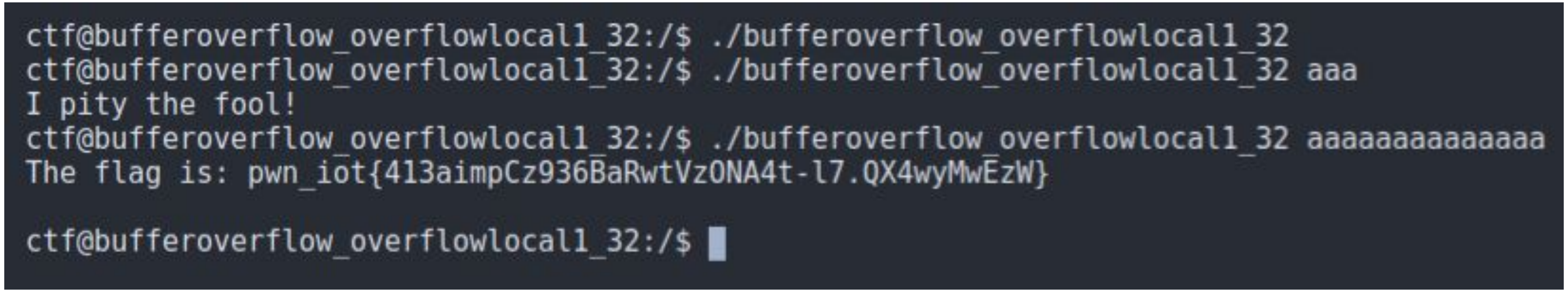
What is a “CTF”?

- CTF stands for “capture the flag”
- Flags are strings of text, like `98dt2n29ncq23f`
- During CTF competitions, competitors find flags by solving puzzles or hacking systems

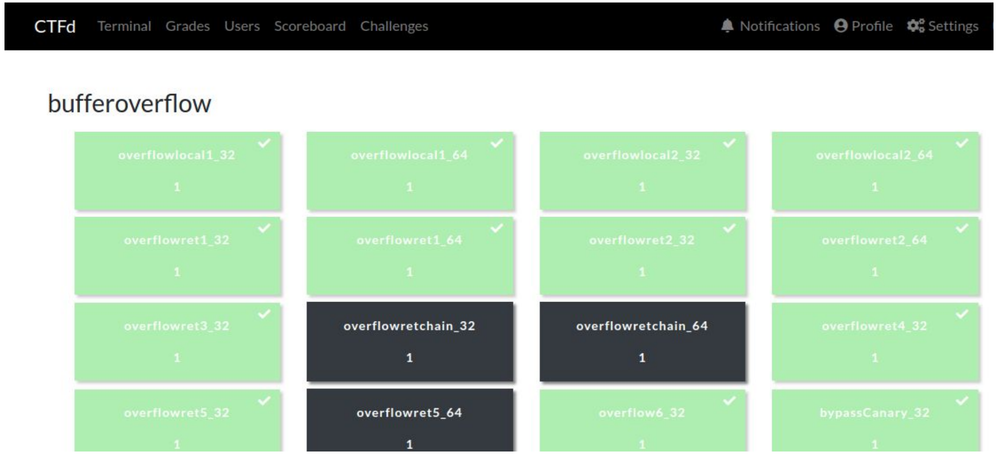
What is an “eCTF”?

- eCTF stands for “embedded capture the flag”
- These are like normal CTF competitions, but with a focus on embedded systems

CTF Examples



```
ctf@bufferoverflow_overflowlocal1_32:/$ ./bufferoverflow_overflowlocal1_32
ctf@bufferoverflow_overflowlocal1_32:/$ ./bufferoverflow_overflowlocal1_32 aaa
I pity the fool!
ctf@bufferoverflow_overflowlocal1_32:/$ ./bufferoverflow_overflowlocal1_32 aaaaaaaaaaaaaa
The flag is: pwn_iot{413aimpCz936BaRwtVzONA4t-l7.QX4wyMwEzW}

ctf@bufferoverflow_overflowlocal1_32:/$ 
```



CTFd Terminal Grades Users Scoreboard Challenges Notifications Profile Settings

bufferoverflow

overflowlocal1_32 1 ✓	overflowlocal1_64 1 ✓	overflowlocal2_32 1 ✓	overflowlocal2_64 1 ✓
overflowret1_32 1 ✓	overflowret1_64 1 ✓	overflowret2_32 1 ✓	overflowret2_64 1 ✓
overflowret3_32 1 ✓	overflowretchain_32 1	overflowretchain_64 1	overflowret4_32 1 ✓
overflowret5_32 1 ✓	overflowret5_64 1	overflow6_32 1 ✓	bypassCanary_32 1 ✓

Embedded Systems in Medical Devices

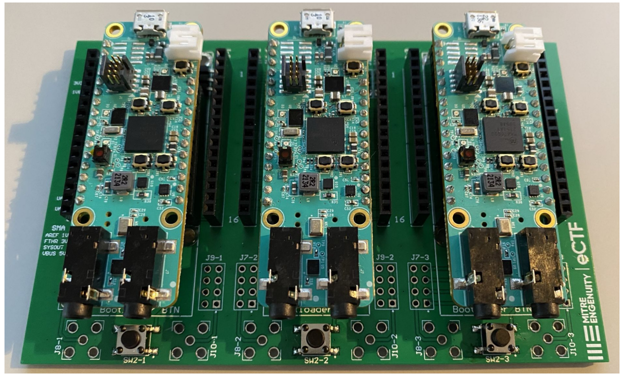
Insulin Pump

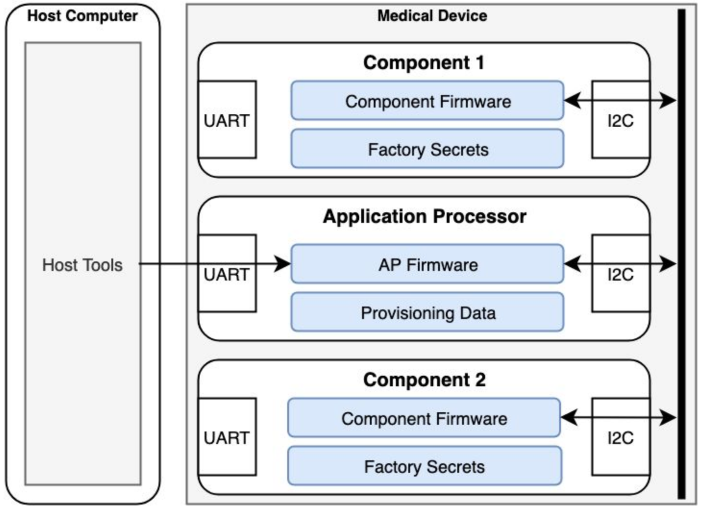
CT

Endoscope

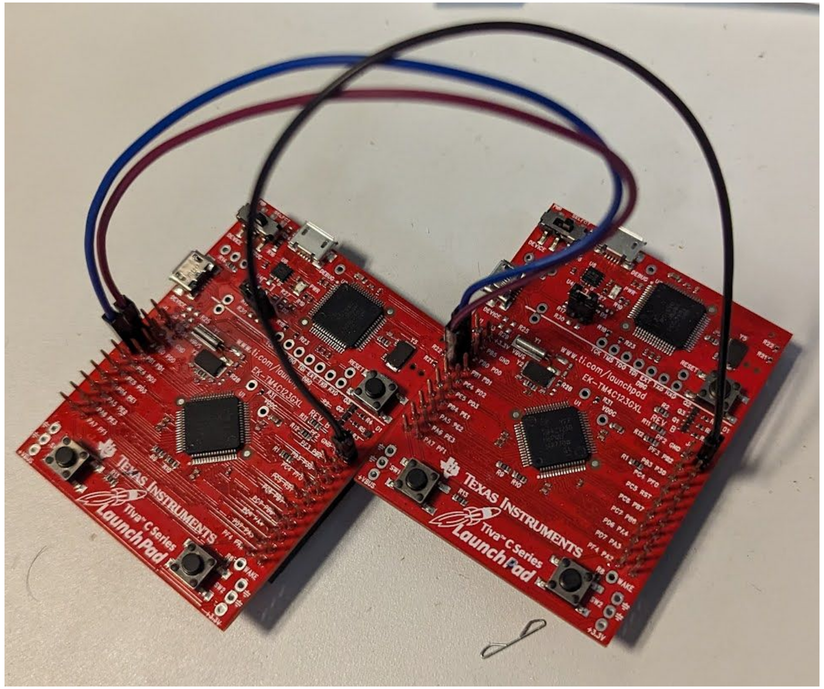
Medical Devices

- A medical device has three parts
  - Two components
  - One application processor
- Components have an ID and hold sensitive data
- The application processor (AP) manages the components
  - Checks if all the required components are presented
  - Attests components
  - Boots each component and the whole medical device

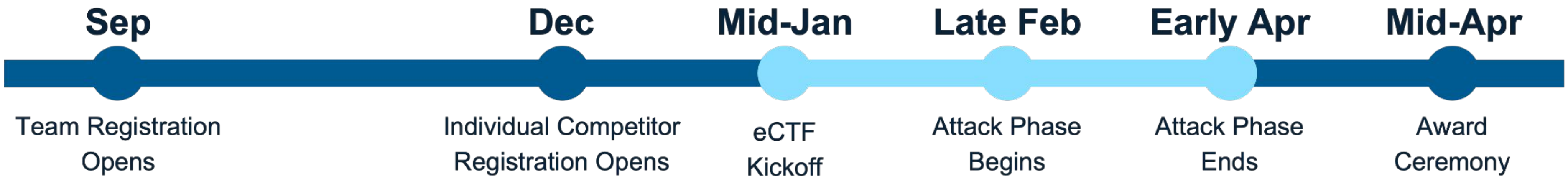




Protected Automotive Remote Entry Device (PARED)



Timeline and Phases for eCTF 2026



Design Phase

Teams design and implement systems that meets security and functionality requirements

Handoff

Organizers test each design for functionality

Attack Phase

Teams analyze and attack each other’s designs for points

Announcement

- MITRE eCTF team competition 2026
  - *T3S-UTEP*
- Shoot me an email if you are interested
  - I will add you to the teams channel
  - YOU WILL BE ADDED TO THE TEAM, **BUT DOES NOT GUARANTEE CREDIT**
- Available for independent study
  - YOU MUST BE COMMITTED
- STUDENTS…, Jaime Acosta, Mohammad Saidur Rahman, **MD Armanuzzaman**