Operating Systems

Lecture 8b

Dr Ronald Grau School of Engineering and Informatics Spring term 2018

Previously

Memory management

- Virtual memory
- Addressing and address spaces
- Partitioning and segmentation
- Paging & Page replacement

Recap: Questions

Recap questions

1. How can we avoid page faults when a process starts?
2. How can thrashing be controlled?
3. What is a working set?
4. What is the significance of the reference bit in the page table?
5. What is the main problem of the FIFO page replacement policy?
6. What information is required to operate the LRU policy?

Today

Files

- Persistent storage
- Formats, access, operations
- Attributes & permissions
- Implementation

File abstraction

Files

- Persistent storage of data
- Named
- Byte-wise access
- Access protection
- Consistency guarantees

Operating system duties:

- Manage efficient access (e.g. caching)
- Protect storage device from being damaged/corrupted
- Provide uniform interface for heterogeneous devices

File abstraction

Basic terminology

- Bit: 0, 1
- Byte: typically 8 bits
- Word: the number of bits a processor can operate on at once
- Character: maps one or more bytes to letters, digits, etc
- Character set: defines the mapping system (e.g. ASCII, Unicode)
- Field: a group of characters
- Record: a group of fields
- File: a group of related records
- Volume: a unit of data storage holding multiple files
- File system: the way the OS organises files and manages access

File names

- Human-readable name to identify a file
- Maps to an internal identifier in the file system
- Length of the file name
  - MS-DOS: 8 characters + extension
  - Modern OS: 255 characters
- Relevance of upper/lower case, allowed characters
  - Unix distinguishes between upper/lower case, unlike Windows
- File name extensions
  - MS-DOS: separate from the le name
  - Modern OS: convention, multiple extensions possible

File types

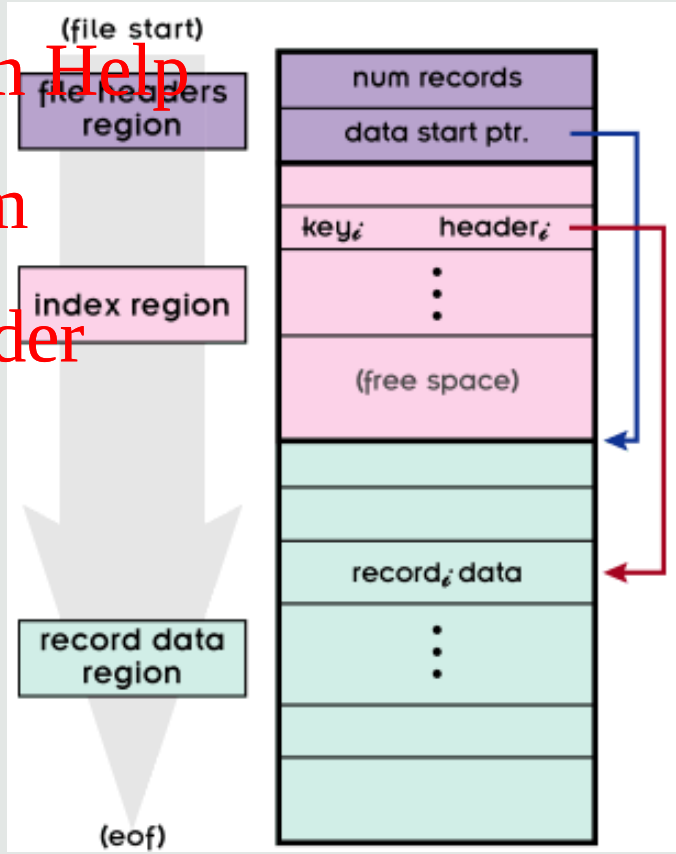
- Regular files
  - Text files: interpretation as characters
  - Binary files: interpretation application-specific
    e.g. executables, pictures
- Directories: hierarchy of files
- Special files for I/O devices

File formats

Define how the data is organised within a file

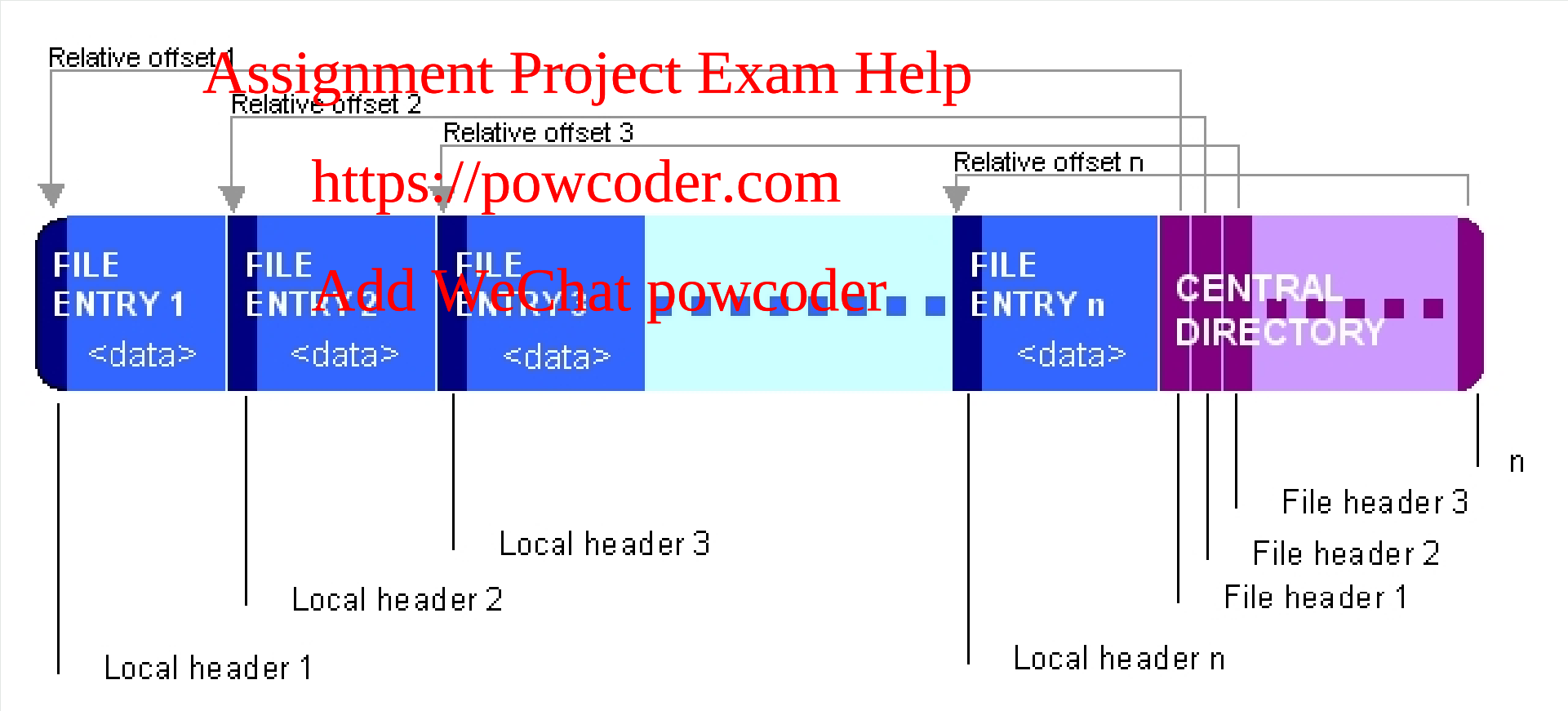
- Fixed-length records
- Variable-length records
- Indexed records
- Trees: XML, JSON

record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4	32343	El Said	History	60000



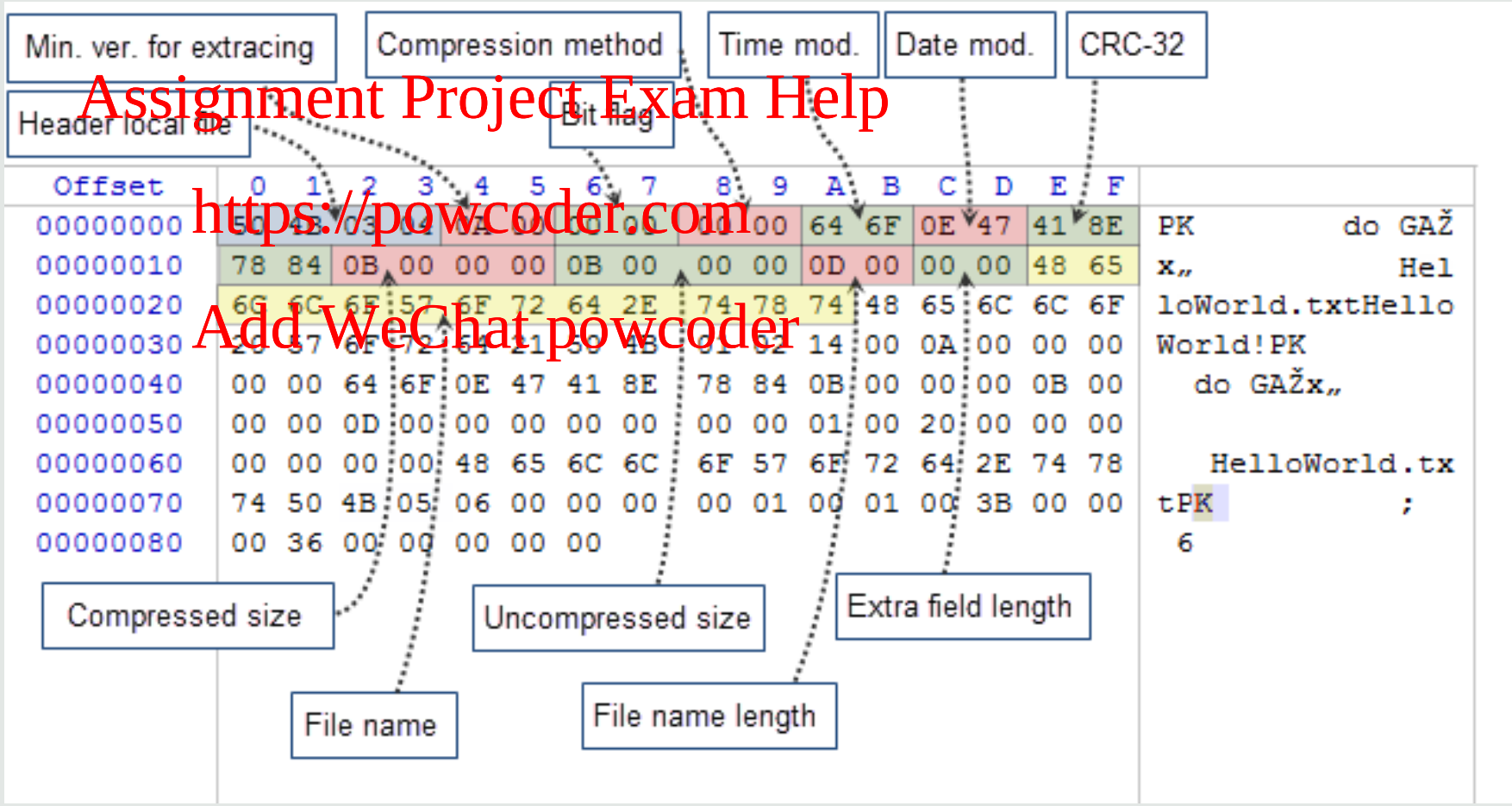
File formats

Example: ZIP format



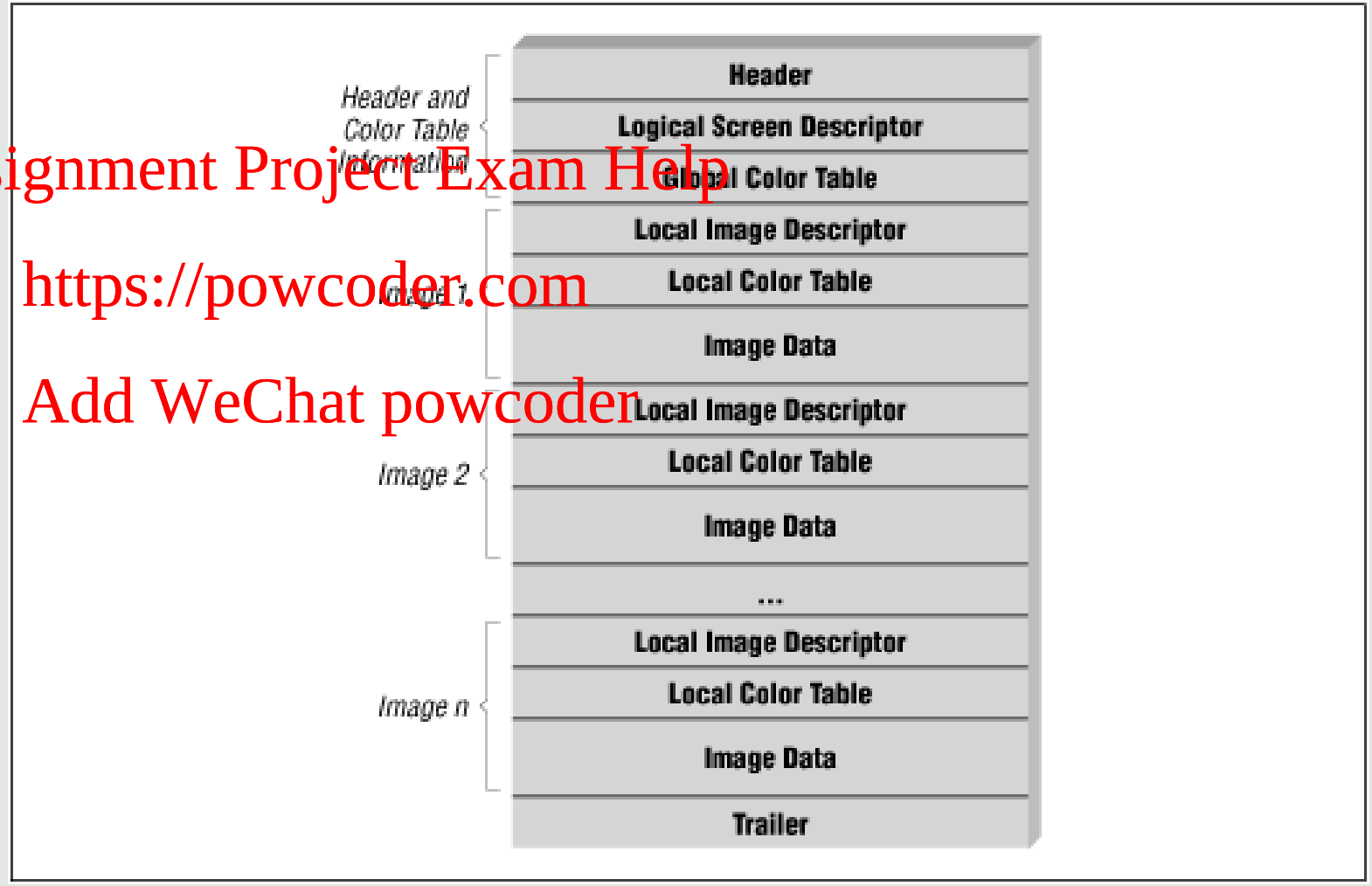
File formats

Example: ZIP format



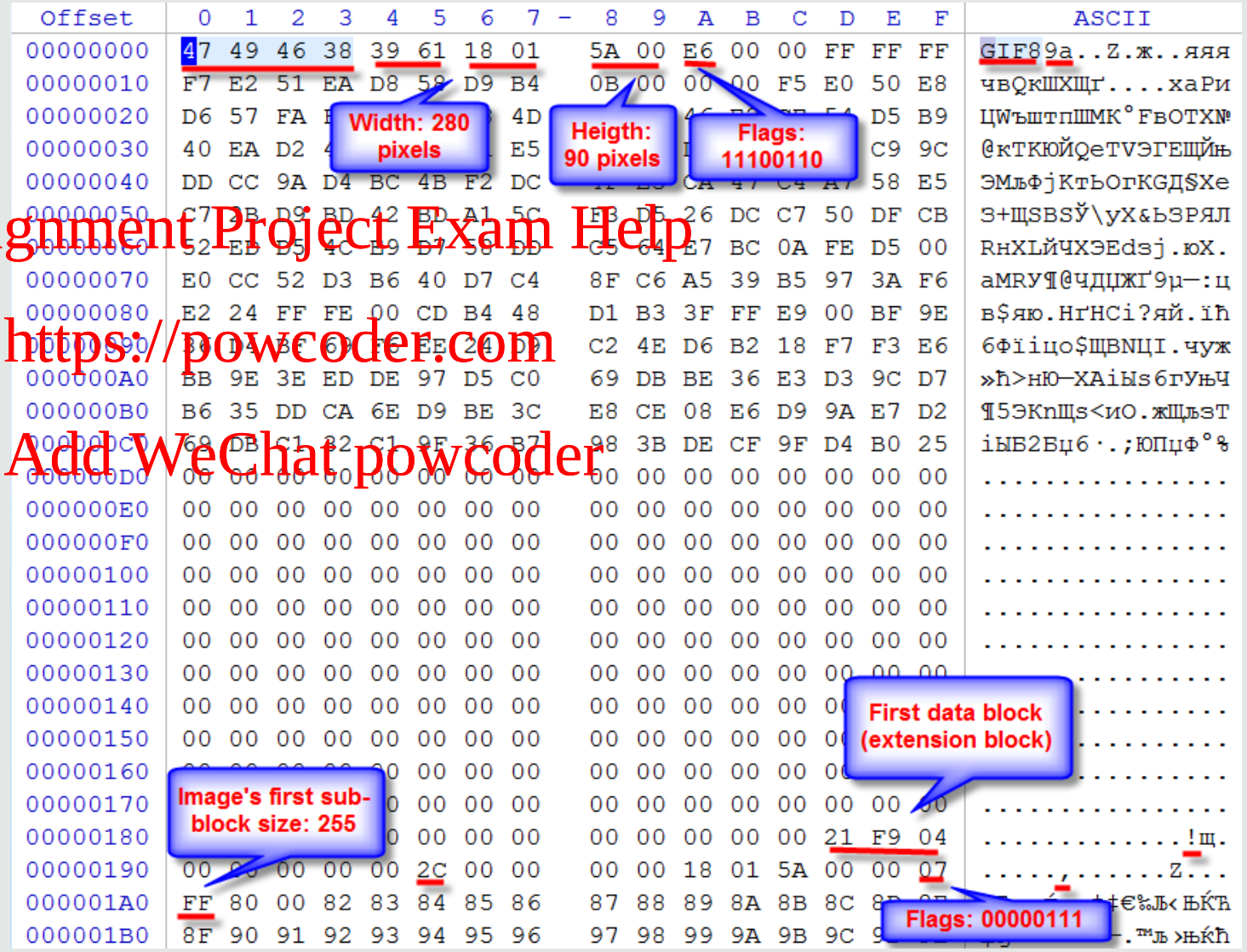
File formats

Example: GIF format

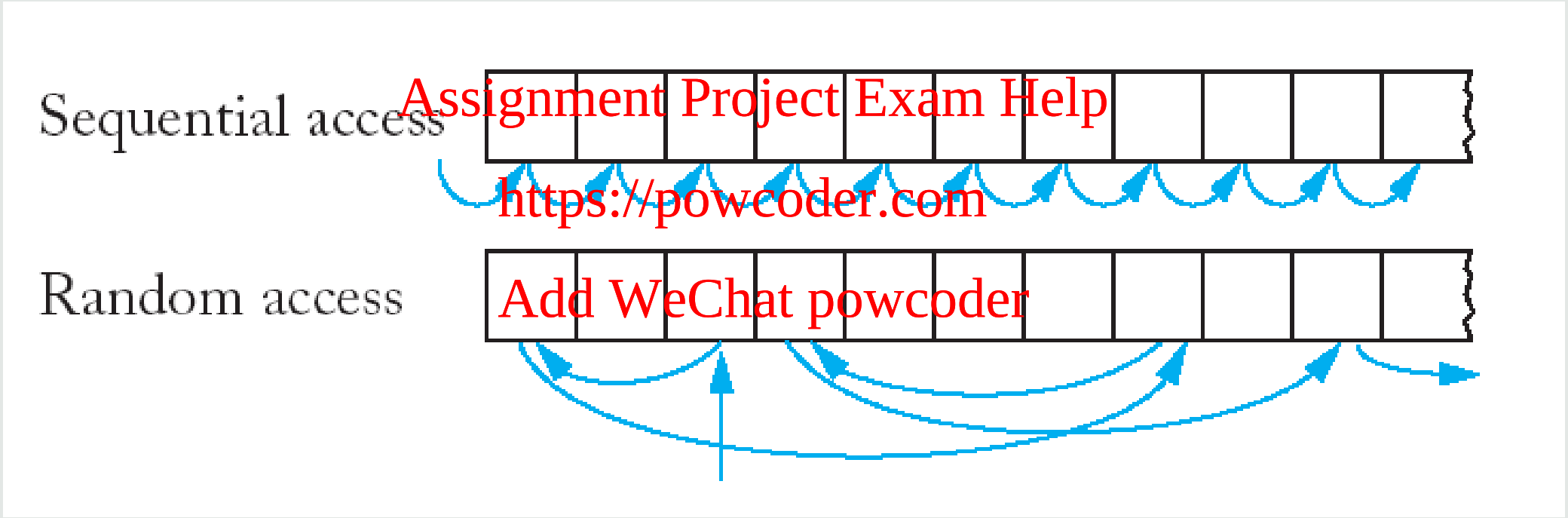


File formats

Example: GIF format

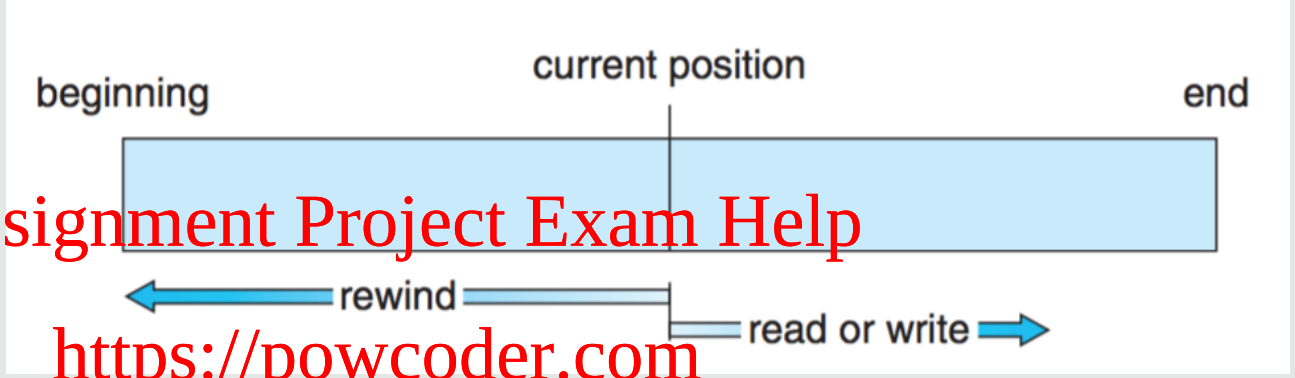


File access

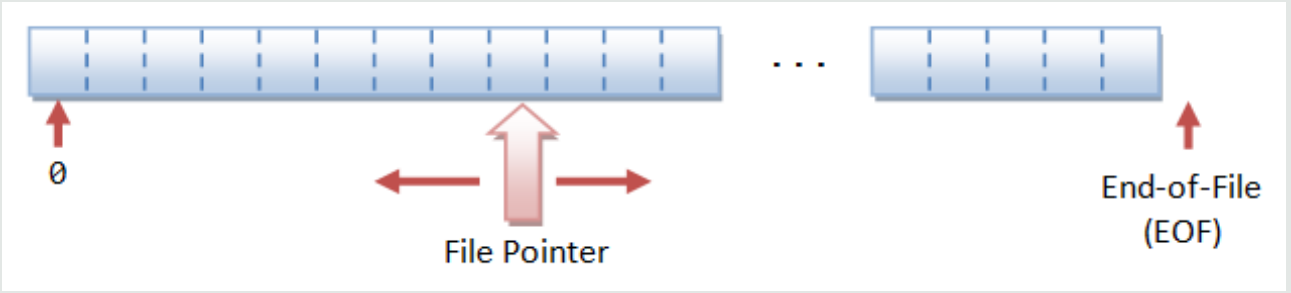


File access

Sequential access:



Random access:



File operations

Typical operations:

- Create, Delete, Rename
- Open, Close
- Read, Write, Append, Truncate
- Seek
- Get Attributes, Set Attributes
- Lock, Unlock

File operations

Example: POSIX System calls


```
int open(const char *filename, int flags, mode_t mode);
```


O_RDONLY	open for reading only
O_WRONLY	open for writing only
O_RDWR	open for reading and writing

O_APPEND	append on each write
O_CREAT	create file if it does not exist: REQUIRES mode
O_TRUNC	truncate size to 0

```
// example:
int fd = open("myfile.txt", O_WRONLY | O_CREAT, 0644);
if (fd < 0) switch(errno) { ... }
```

File operations

Example: POSIX System calls

```
ssize_t read(int fd, void *buf, size_t count);
```



```
// example:
char buffer[BUF SIZE];
ssize_t bytes read = read(fd, buffer, BUF_SIZE);
```

File operations

Example: POSIX System calls

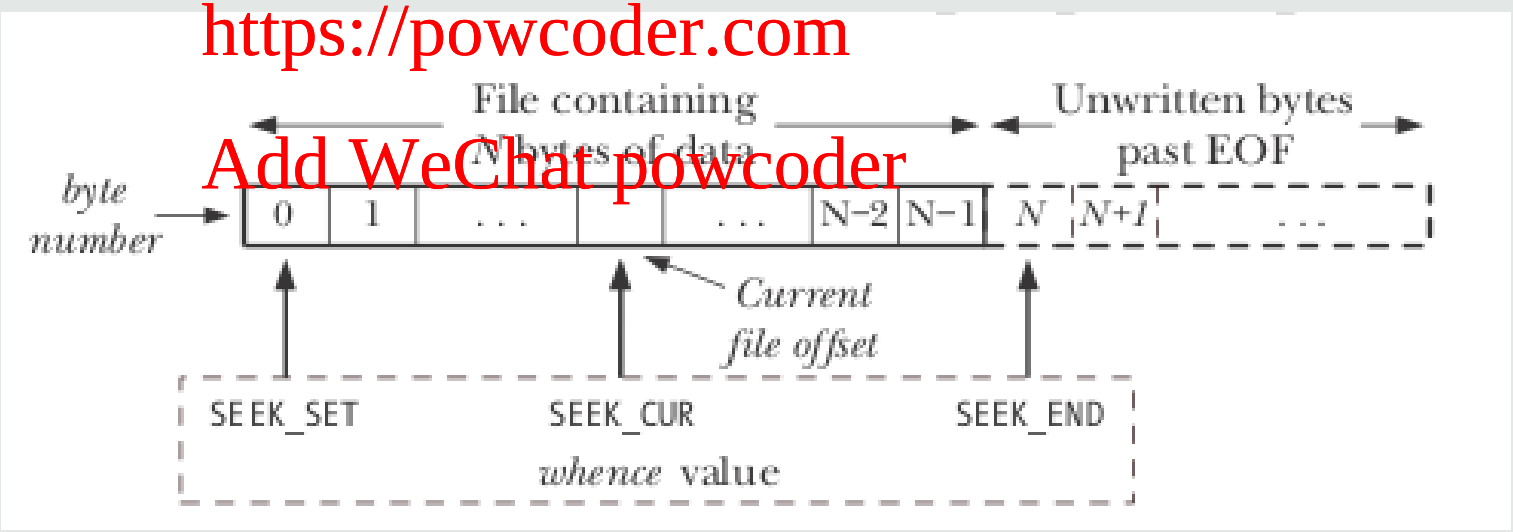
```
ssize_t write(int fd, void *buf, size_t count);

// example:
char buffer[BUF_SIZE];
ssize_t bytes_written = write(fd, buffer, BUF_SIZE);
```

File operations

Example: POSIX System calls

```
ssize_t lseek(int fd, off_t offset, int whence);
```


```
// example:
off_t pos = lseek(fd, 0, SEEK_CUR);
```

File operations

Example: POSIX System calls

```
int close(int fd);

int unlink(const char *pathname);

int fcntl(int fd, int cmd, ... /* args */ );
```

File operations

Example: POSIX System calls

```
int stat(const char *filename, struct stat *buf);
```



```
struct stat {
    dev_t     st_dev;     /* ID of device containing file */
    ino_t     st_ino;     /* inode number */
    mode_t    st_mode;    /* protection */
    nlink_t   st_nlink;   /* number of hard links */
    uid_t     st_uid;     /* user ID of owner */
    gid_t     st_gid;     /* group ID of owner */
    dev_t     st_rdev;    /* device ID (if special file) */
    off_t     st_size;    /* total size, in bytes */
    blksize_t st_blksize; /* blocksize for file system I/O */
    blkcnt_t  st_blocks;  /* number of 512B blocks allocated */
    time_t    st_atime;   /* time of last access */
    time_t    st_mtime;   /* time of last modification */
    time_t    st_ctime;   /* time of last status change */
};
```

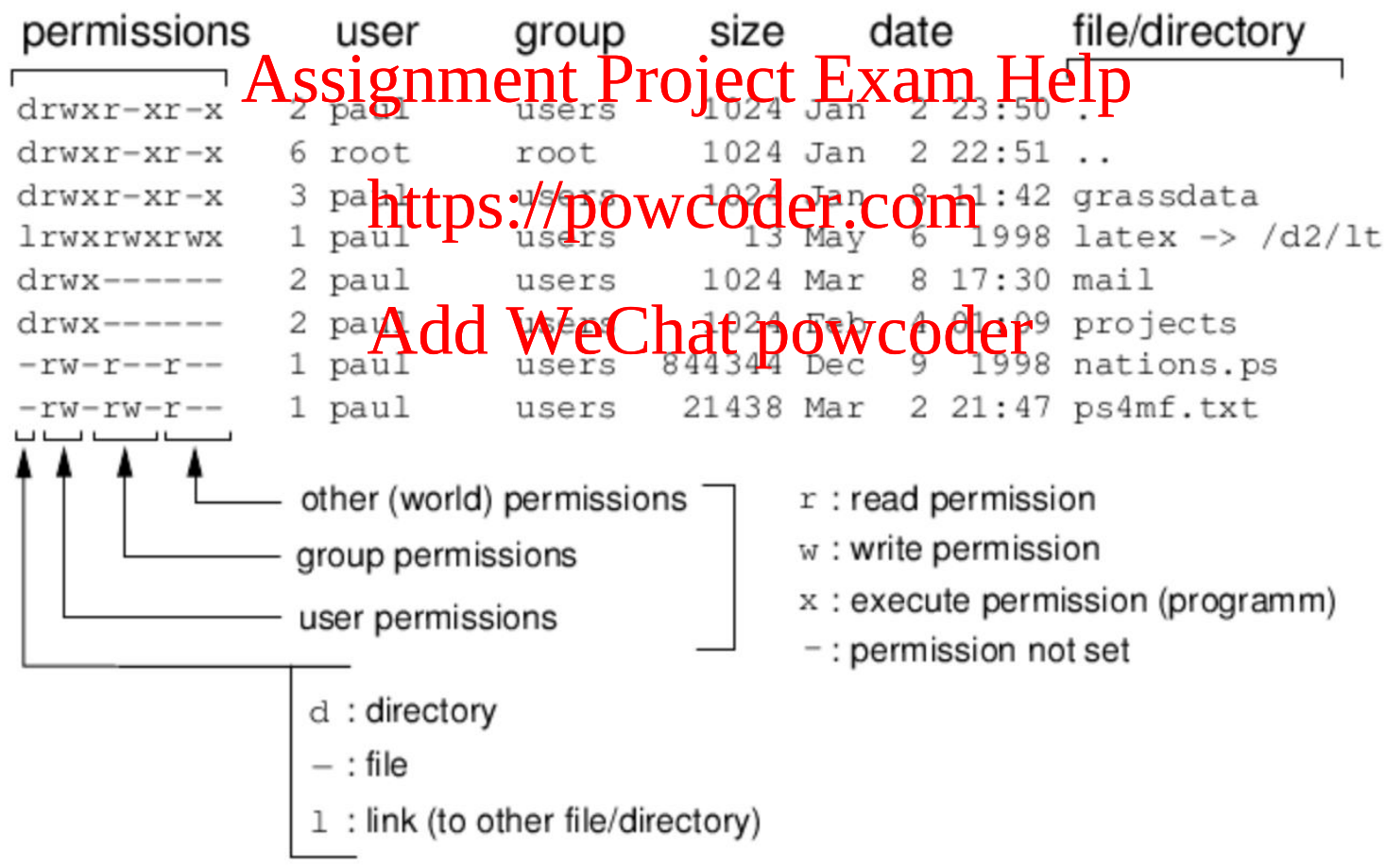
File attributes

Typical metadata:

- Creator, Owner, Owner's group, Protection, Password
- Read-only flag, Write flag, Executable flag, Hidden flag, System flag, Archive flag, ASCII/binary flag, Random access flag, Temporary flag, Lock flags
- Creation time, Time of last access, Time of last change
- Record length, Key position, Key length
- Current size, Maximum size

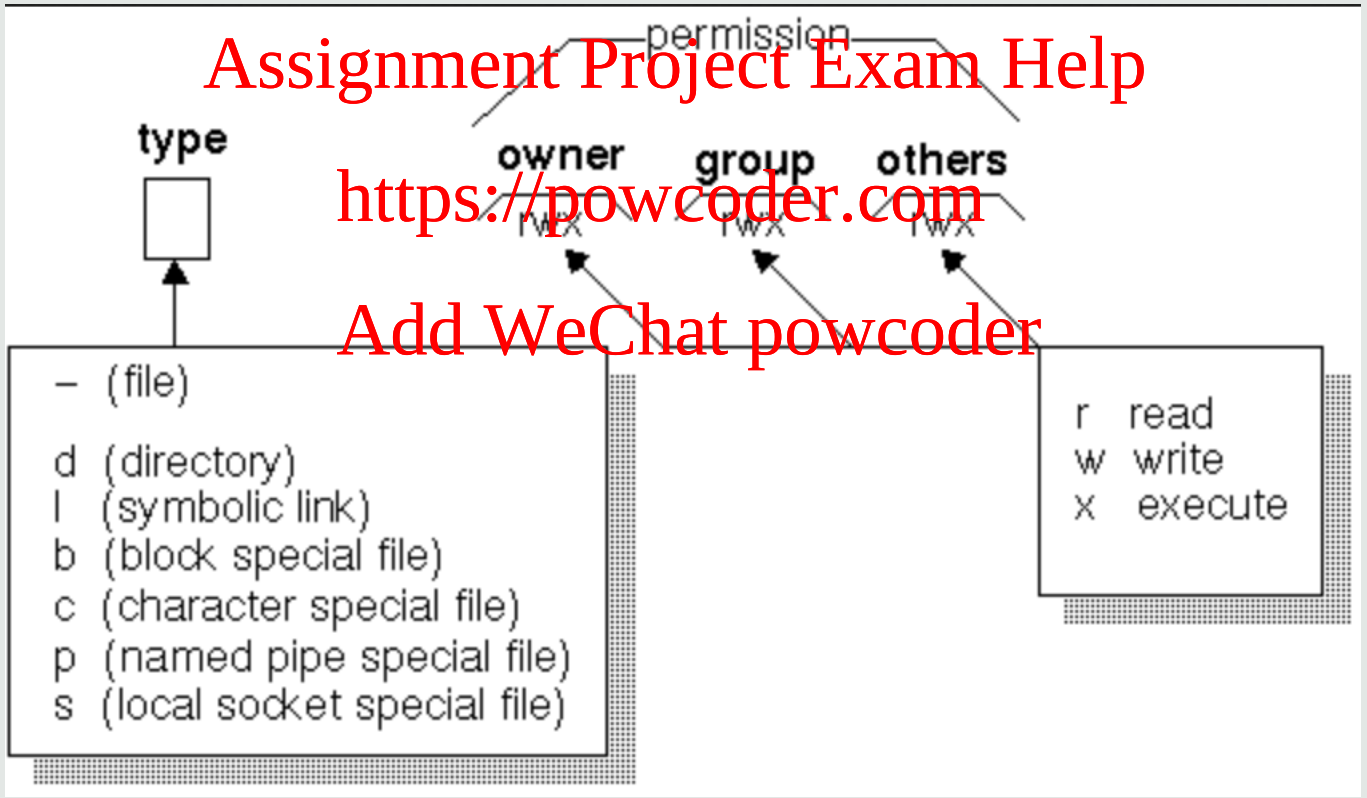
File attributes

Example: UNIX file attributes



File attributes

Example: UNIX file permissions



File attributes

Example: UNIX file permissions

	File	Directory
Read	Read	Ignored
Write	Write	Allowed to create files in the directory
Execute	Execute	Allowed to view files in the directory
Set UID	Allows all users to execute the file with the permissions of the owner	Ignored
Set GID	Allows all users to execute the file with the permissions of the group	All new files created in the directory are set to the same group as the directory
Sticky Bit	Ignored	Prohibits non-owners from deleting or renaming a file

File implementation

Example:

`java.io.File`

`File(String pathname)`


```
boolean exists()
boolean createNewFile()
boolean renameTo(File dest)
boolean delete()
long length()
long lastModified()
boolean canRead() )
boolean canWrite()
boolean canExecute()
boolean setReadable(boolean readable,
boolean ownerOnly)
boolean setWritable(boolean writable,
boolean ownerOnly)
boolean setExecutable(boolean executable,
boolean ownerOnly)
. . .
```

File implementation

Example:

`java.io.RandomAccessFile`

`RandomAccessFile(String name, String mode)`
`throws FileNotFoundException`


```
void close()
int read(byte[] b)
int write(byte[] b)
void seek(long pos)
long getFilePointer()
void setLength(long newLength)
String readLine()
double readDouble()
. . .

throw
       IOException,
       EOFException,
       SecurityException,
       . . .
```

File implementation

Example:

java.io Streams and Readers/Writers

FileInputStream(File file)	int read(byte[] buf) void close()
FileOutputStream(File file)	int write(byte[] buf) void close()
FileReader(File file)	int read(char[] cbuf) void close()
FileWriter(File file)	int write(char[] cbuf) void close()

Other aspects

- How files are stored
- How storage space is managed
- Performance
- Reliability

Summary

Files

- Naming: human-readable identifiers
- Attributes: permissions, ownership, dates, . . .
- Access: open - read / write - close
- Unified interface for access to a variety of devices
- Examples: POSIX, Java

Read

- Tanenbaum & Bos., Modern Operating Systems
  - Chapter 4
- Silberschatz et al., Operating System Concepts
  - Chapters 10 & 11

Next Lecture

- Introduction
- Operating System Architectures
- Processes
- Threads - Programming
- Process Scheduling - Evaluation
- Process Synchronisation
- Deadlocks
- Memory Management
- **File Systems (continued)**
- Input / Output
- Security and Virtualisation