

# System programming: file management



---

CS 2204

Class meeting 10



# UNIX system programming

---

- Programming that uses special features of the UNIX system
- Programs make *system calls*
- Types of system calls
  - File I/O
  - Process management
  - Inter-process communication (IPC)
  - Signal handling



# Basic file I/O

---

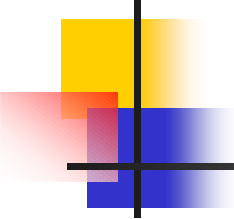
- Processes keep a list of open files
- Files can be opened for reading, writing
- Each file is referenced by a *file descriptor* (integer)
- Three files are opened automatically
  - FD 0: standard input
  - FD 1: standard output
  - FD 2: standard error



# File I/O system calls - `open()`

---

- `fd = open(path, flags, mode)`
- `path`: string, absolute or relative path
- `flags`:
  - `O_RDONLY` - open for reading
  - `O_WRONLY` - open for writing
  - `O_RDWR` - open for reading and writing
  - `O_CREAT` - create the file if it doesn't exist
  - `O_TRUNC` - truncate the file if it exists
  - `O_APPEND` - only write at the end of the file
- `mode`: specify permissions if using `O_CREAT`



# File I/O system calls - `close()`

---

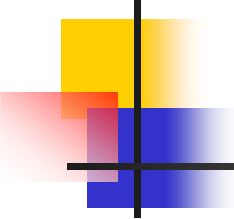
- `retval = close(fd)`
- Close an open file descriptor
- Returns 0 on success, -1 on error



# File I/O system calls - `read()`

---

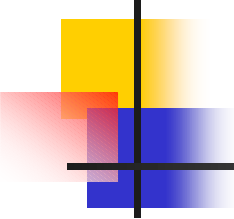
- `bytes_read = read(fd, buffer, count)`
- Read up to count bytes from file and place into buffer
- `fd`: file descriptor
- `buffer`: pointer to array
- `count`: number of bytes to read
- Returns number of bytes read or -1 if error



# File I/O system calls - `write()`

---

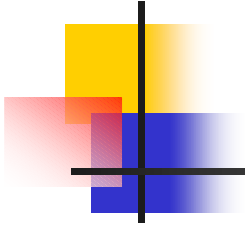
- `bytes_written = write(fd, buffer, count)`
- Write count bytes from buffer to a file
- `fd`: file descriptor
- `buffer`: pointer to array
- `count`: number of bytes to write
- Returns number of bytes written or -1 if error



# File I/O system calls - lseek ()

---

- `retval = lseek(fd, offset, whence)`
- Move file pointer to new location
- `fd`: file descriptor
- `offset`: number of bytes
- `whence`:
  - `SEEK_SET` - offset from beginning of file
  - `SEEK_CUR` - offset from current offset location
  - `SEEK_END` - offset from end of file
- Returns offset from beginning of file or -1



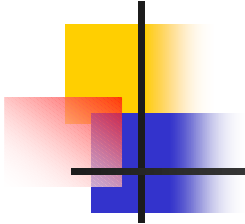
# Simple file I/O example



# File I/O using FILEs

---

- Most UNIX programs use higher-level I/O functions
  - `fopen()`
  - `fclose()`
  - `fread()`
  - `fwrite()`
  - `fseek()`
- These use the `FILE` datatype instead of file descriptors
- Need to include `stdio.h`



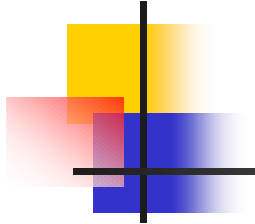
# Simple file I/O example using FILES



# Using datatypes with file I/O

---

- All the functions we've seen so far use raw bytes for file I/O, but program data is usually stored in meaningful datatypes (int, char, float, etc.)
- `fprintf()`, `fputs()`, `fputc()` - used to write data to a file
- `fscanf()`, `fgets()`, `fgetc()` - used to read data from a file



# Simple file I/O example using FILES and meaningful datatypes