System File API

Reading

- Operating systems store the logical blocks that comprise each file in a file system.
- Operating systems provide simple access to their file systems through a primitive Application Programming Interface (API).
- File access begins with a command to "open" the file.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int open(char *filename, int flags, mode_t
 mode); // when creating a file
int open(char *filename, int flags); // when
                    // opening an existing file
int creat( char *filename, mode_t mode); // also
                                  // creates a file
    return: file descriptor (fd) on success,
     -1 on error
```

- The parameters include the name of the file, constraints on how the file will accessed, and user permissions.
- Some values for flags:

```
O_RDONLY: Reading only
O_WRONLY: Writing only
O_RDWR: Reading and writing
```

- The open function returns an integer known as a file descriptor. The file descriptor may then be used to read and write to the file.

- The contents of a file are read using the read function:

```
#include <unistd.h>
ssize_t read(int fd, void *buf, size_t n);
    return: number of bytes read if OK,
     0 on EOF, -1 on error
```

- The read function parameters include a location where the data read from the file is to be stored, known as `buf`. The final parameter is the size, in bytes, of that location.
- Once reading operations have been completed, the file is closed:

```
int close( int fd );
```

- Example: Opening and reading a small text file
  - `read-file.c`

System File API

Standard Files and Writing

- At startup, each program (process) is supplied with file descriptors that refer to the keyboard and the monitor. The numbers for these descriptors and their aliases are included within unistd.h:

```
STDIN_FILENO   0
STDOUT_FILENO  1
STDERR_FILENO  2
```

- Hence, you can read from the keyboard by using the STDIN_FILENO file descriptor. In addition, you can write to the monitor using either the STDOUT_FILENO or STDERR_FILENO file descriptors.
- Example: Reading from standard input
  - `read-stdin.c`

- When opening a file for writing, additional flags are used, such as:

```
O_CREAT: If the file doesn't exist, then create
 a truncated (empty) version of it.
O_TRUNC: If the file already exists, then
 truncate it.
O_APPEND: Before each write operation, set the
 file position to the end of the file.
```

- The creat() function opens a file using the flags O_CREAT|O_WRONLY|O_TRUNC.
- Mode bits include permissions, such as S_IRWXU (user has read, write, and execute permission), S_IRGRP|S_IWGRP (group members have read and write permission), S_IROTH (others have read permission)
- Content is written to a file using the write function:

```
#include <unistd.h>
ssize_t write(int fd, const void *buf,
 size_t n);
```

```
    return: number of bytes written if OK,
     -1 on error
```

- The write function parameters include a location where the data resides in memory that is to be written to the file, known as `buf`. The final parameter is the size, in bytes, of that location.
- Once writing operations have been completed, the file is closed using the close function.
- Example: Writing to a file
  - `write-file.c`