C Programming

File handling using open(), read(), write() and close()
The previous examples of file handling deal with File Control Blocks (FCB). Under MSDOS v3.x (or greater) and UNIX systems, file handling is often done using handles, rather than file control blocks.

Writing programs using handles ensures portability of source code between different operating systems. Using handles allows the programmer to treat the file as a stream of characters.

---

open()

```
#include <fcntl.h>
int  open(  char  *filename,  int  access,  int  permission  );
```

The available access modes are

```
O_RDONLY            O_WRONLY            O_RDWR
O_APPEND            O_BINARY            O_TEXT
```

The permissions are

```
S_IWRITE        S_IREAD S_IWRITE | S_IREAD
```

The *open()* function returns an integer value, which is used to refer to the file. If un- successful, it returns -1, and sets the global variable *errno* to indicate the error type.

---

read()

```
#include  <fcntl.h>
int  read(  int  handle,  void  *buffer,  int  nbyte );
```

The *read()* function attempts to read nbytes from the file associated with handle, and places the characters read into *buffer*. If the file is opened using O_TEXT, it removes carriage returns and detects the end of the file.

The function returns the number of bytes read. On end-of-file, 0 is returned, on error it returns -1, setting errno to indicate the type of error that occurred.

---

write()

```
#include  <fcntl.h>
int  write(  int  handle,  void  *buffer,  int  nbyte  );
```

The *write()* function attempts to write nbytes from *buffer* to the file associated with handle. On text files, it expands each LF to a CR/LF.

The function returns the number of bytes written to the file. A return value of -1 indicates an error, with errno set appropriately.

---

close()

```
#include  <fcntl.h>
int  close(  int  handle  );
```

The *close()* function closes the file associated with handle. The function returns 0 if successful, -1 to indicate an error, with errno set appropriately.
