THE POSIX.1 FIPS STANDARD

FIPS stands for Federal Information Processing Standard. This standard was developed by National Institute of Standards and Technology. The latest version of this standard, FIPS 151-1, is based on the POSIX.1- 1998 standard. The FIPS standard is a restriction of the POSIX.1-1998 standard, Thus a FIPS 151-1 conforming system is also POSIX.1-1998 conforming, but not vice versa.

FIPS 151-1 conforming system requires following features to be implemented in all FIPS conforming systems.

_POSIX_JOB_CONTROL	_POSIX_JOB_CONTROL must be defined.
_POSIX_SAVED_IDS	_POSIX_SAVED_IDS must be defined.
_POSIX_CHOWN_RESTRICTED	_POSIX_CHOWN_RESTRICTED must be defined and its value is not -1, it means users with special previlege may change ownership of any files on a system.
_POSIX_NO_TRUNC	If the defined value is -1, any long path name passed to an API is silently truncated to NAME_MAX bytes, otherwise error is generated.
_POSIX_VDISABLE	POSIX_VDISABLE must be defined and its value is not -1.
_POSIX_NO_TRUNC	Must be defined and its value is not -1, Long path name is not support.
NGROUP_MAX	Symbol's value must be at least 8.
The read and write API should return the number of bytes that have been transferred after the APIs have been	
The group ID of a newly created file must inherit the group ID of its containing directory.	

Context Switching

A *user mode* is the normal execution context of any user process, and it allows the process to access its specific data only.

A *kernel mode* is the protective execution environment that allows a user process to access kernels data in a restricted manner.

When the APIs execution completes, the user process is switched back to the user mode. This context switching for each API call ensures that process access kernels data in a controlled manner and minimizes any chance of a runway user application may damage an entire system. So in general calling an APIs is more time consuming than calling a user function due to the context switching. Thus for those time critical applications, user should call their system APIs only if it is necessary.

An APIs common Characteristics

Most system calls return a special value to indicate that they have failed. The special value is typically -1, a null pointer, or a constant such as EOF that is defined for that purpose.

To find out what kind of error it was, you need to look at the error code stored in the variable errno. This variable is declared in the header file errno.h as shown below.

volatile int **errno**

- The variable errno contains the system error number.
  void **perror** (*const char *message*)
- The function perror is declared in stdio.h.

Following table shows Some Error Codes and their meaning:

Errors	Meaning
EPERM	API was aborted because the calling process does not have the super user privilege.
EINTR	An APIs execution was aborted due to signal interruption.
EIO	An Input/Output error occurred in an APIs execution.
ENOEXEC	A process could not execute program via one of the Exec API.
EBADF	An API was called with an invalid file descriptor.
ECHILD	A process does not have any child process which it can wait on.
EAGAIN	An API was aborted because some system resource it is requested was temporarily unavailable. The API should call again later.
ENOMEM	An API was aborted because it could not allocate dynamic memory.
EACCESS	The process does not have enough privilege to perform the operation.
EFAULT	A pointer points to an invalid address.
EPIPE	An API attempted to write data to a pipe which has no reader.
ENOENT	An invalid file name was specified to an API.