**main function:**

An operating system must provide the users with an extended (i.e., virtual) machine, and it must manage the I/O devices and other system resources.

**fork**

fork() causes creation of a new process.

**Kill**

 The kill utility sends a signal to the processes specified by the pid operands.

Only the super-user may send signals to other users' processes.

**Mmap**

 The mmap() system call causes the pages starting at addr and continuing for at most len bytes to be mapped from the object described by fd, starting at byte offset offset.  If offset or len is not a multiple of the pagesize, the mapped region may extend past the specified range.  Any extension beyond the end of the mapped object will be zero-filled.

**Chmod**

The chmod utility modifies the file mode bits of the listed files as specified by the mode operand. It may also be used to modify the Access Control Lists (ACLs) associated with the listed files.

**Waitpid**

 The wait() function suspends execution of its calling process until stat\_loc information is available for a terminated child process, or a signal is received.  On return from a successful wait() call, the stat\_loc area contains termination information about the process that exited as defined below.

**System Calls Fail:**

**fork():** fork() will fail and no child process will be created if:

* [EAGAIN]           The system-imposed limit on the total number of processes under execution would be exceeded.  This limit is configuration-dependent.
* [EAGAIN]           The system-imposed limit MAXUPRC (<sys/param.h>) on the total number of processes under execution by a single user would be exceeded.
* [ENOMEM]           There is insufficient swap space for the new process.

**read():**The read() call may also return the following errors:

* [ECONNRESET]       The connection is closed by the peer during a read attempt on a socket.
* [ENOTCONN]         A read is attempted on an unconnected socket.
* [ETIMEDOUT]        A transmission timeout occurs during a read attempt on a socket.

**exec():** Exec can fail if the file name given does not exist or is not a valid

executable file.

**unlink():** The unlink() system call will fail if:

* [EACCES]           Search permission is denied for a component of the

                        path prefix.

* [EACCES]           Write permission is denied on the directory containing

                        the link to be removed.

* [EBUSY]            The entry to be unlinked is the mount point for a

                        mounted file system.

* [EBUSY]            The file named by the path argument cannot be unlinked

                        because it is being used by the system or by another

                        process.

* [EFAULT]           Path points outside the process's allocated address

                        space.

* [EIO]              An I/O error occurs while deleting the directory entry

                        or deallocating the inode.

* [ELOOP]            Too many symbolic links are encountered in translating

                        the pathname.  This is taken to be indicative of a

                        looping symbolic link.

* [ENAMETOOLONG]     A component of a pathname exceeds {NAME\_MAX} charac-

                        ters, or an entire path name exceeds {PATH\_MAX} char-

                        acters (possibly as a result of expanding a symlink).

**mount():** mount() will fail when one of the following occurs:

* [EPERM]         The caller is not the super-user, and the device-node and the mountpoint do not have adequate ownership and permissions.
* [ENAMETOOLONG]  A component of a pathname exceeded {NAME\_MAX} characters, or an entire path name exceeded {PATH\_MAX} characters.
* [ELOOP]         Too many symbolic links were encountered in translating a pathname.
* [ENOENT]        A component of dir does not exist.
* [ENOTDIR]       A component of name is not a directory, or a path prefix of special is not a directory.
* [EINVAL]        A pathname contains a character with the high-order bit set.
* [EBUSY]         Another process currently holds a reference to dir.
* [EFAULT]        Dir points outside the process's allocated address space.

**Trap instruction:**

A trap instruction switches from user mode to kernel mode. An interrupt is used to save the state of a process during a context switch.