pid\_t fork(void);

creates a new process by duplicating the calling process. The calling process is referred to as the parent process.

int stat(const char \**path*, struct stat \**buf*);

returns information about a file “pathname”, in the buffer pointed to by buf.

int kill(pid\_t *pid*, int *sig*);

can be used to send any signal to any process group or process. If the pid is >0, then signal sig is sent to the process with the ID (pid). If the pid =0, sig is sent to every process in the process group of the calling process. If pid <0, then sig is sent to every process for which the calling process has permission to send signals, except for process 1 (init). If pid is <0, then sig is sent to every process in the process group whose ID is -pid. If sig is 0, no signal is sent.

void \*mmap(void \*addr, size\_t lengthint " prot ", int " flags ,

int fd, off\_t offset);

creates a new mapping in the virtual address space of the calling process. The starting address for the new mapping is specified in *addr*. The *length* argument specifies the length of the mapping. If *addr* is NULL, then the kernel chooses the address at which to create the mapping; this is the most portable method of creating a new mapping. If *addr* is not NULL, then the kernel takes it as a hint about where to place the mapping; on Linux, the mapping will be created at a nearby page boundary. The address of the new mapping is returned as the result of the call. Returns a pointer to the mapped area

int chmod(const char \**pathname*, mode\_t *mode*);

change a file’s mode bits, with *mode*. (The file mode consists of the file permission bits plus the set-user-ID, set-group-ID, and sticky bits.)

pid\_t waitpid(pid\_t *pid*, int \**status*, int *options*);

The **waitpid**() system call suspends execution of the calling process until a child specified by *pid* argument has changed state. By default, **waitpid**() waits only for terminated children, but this behavior is modifiable via the *options* argument, as described below.

The value of *pid* can be:

< -1

meaning wait for any child process whose process group ID is equal to the absolute value of *pid*.

-1

meaning wait for any child process.

0

meaning wait for any child process whose process group ID is equal to that of the calling process.

> 0

meaning wait for the child whose process ID is equal to the value of *pid*.

Fork(), Failes:

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

Exec(), failes:

* [E2BIG] - The number of bytes in the new process's argument list is larger than the system-imposed limit. This limit is specified by the sysctl(3) MIB variable KERN\_ARGMAX.
* [EACCES] - Search permission is denied for a component of the path prefix.
* [EACCES] - The new process file is not an ordinary file.
* [EACCES] - The new process file mode denies execute permission.
* [EACCES] - The new process file is on a filesystem mounted with execution disabled (MNT\_NOEXEC in <sys/mount.h>).
* [EFAULT] - The new process file is not as long as indicated by the size values in its header.
* [EFAULT] - Path, argv, or envp point to an illegal address.
* [EIO] - An I/O error occurred while reading from the file system.
* [ELOOP] - Too many symbolic links were encountered in translating the pathname. This is taken to be indicative of a looping symbolic link.
* [ENAMETOOLONG] - A component of a pathname exceeded {NAME\_MAX} characters, or an entire path name exceeded {PATH\_MAX} characters.
* [ENOENT] - The new process file does not exist.
* [ENOEXEC] - The new process file has the appropriate access permission, but has an unrecognized format (e.g., an invalid magic number in its header).
* [ENOMEM] - The new process requires more virtual memory than is allowed by the imposed maximum (getrlimit(2)).
* [ENOTDIR] - A component of the path prefix is not a directory.
* [ETXTBSY] - The new process file is a pure procedure (shared text) file that is currently open for writing or reading by some process.