System Calls Understanding

**POSIX calls**

* fork
  + Description

fork() creates a new process by duplicating the calling process.

* pid\_t fork (void);
* Return value

On success, the PID of the child process is returned in the parent, and 0 is returned in the child. On failure, -1 is returned in the parent, no child process is created, and erno is set appropriately.

* stat
* Description

Display file or file system status.

-L, --derefence follow links

-f, --file-system display file system status instead of file status

* kill
* Description

The default signal for kill is TERM. Use -l or -L to list available signals. Particularly useful signals include HUP, INT, KILL, STOP, CONT and 0. Alternate signals may be specified in three ways: -9-SIGKILL-KILL. Negative PID values may be used to choose whole process groups. A PID of -1 is special; it indicates all processes except the kill process itself and init.

* kill [ -signal | -s signal ]
* pid ...kill [ -L | -V, --version ]
* kill -l [ signal ]
* mmap
* Description

mmap() 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.

* void \* mmap (void \* addr, size\_t length,
* int prot, int flags, int fd, off\_t offset);
* int munmap (void \* addr, size\_t Länge);
* chmod
  + Description

This manual page documents the GNU version of chmod, chmod changes the file mode bits of each given file according to mode, which can be either a symbolic representation of changes to make, or an octal number representing the bit pattern for the new mode bits.

* chmod [OPTION]... MODE[,MODE]... FILE...
* chmod [OPTION]... OCTAL-MODE FILE...
* chmod [OPTION]... --reference=RFILE FILE...
* waitpid
  + Description

All of the system calls are used to wait for state changes in a child of the calling process and obtain information about the child whose state has changed.

* pid\_t wait(int \*status);
* pid\_t waitpid(pid\_t pid, int \*status, int options);
* int waitid(idtype\_t idtype, id\_t id, siginfo\_t \*infop, int options);

**System Calls Fail**

Give a condition that causes it to fail

* Fork
  + EAGAIN fork() cannot allocate sufficient memory to copy the parent’s page tables and allocate a task structure for the child.
* Exec
  + All of these functions may fail and set errno for any of the errors specified for execve(2).
* Unlink
  + EIO An I/O error occurred.
* Read
  + EAGAIN The file descriptor fd refers to a file other than a socket and

has been marked nonblocking (O\_NONBLOCK), and the read would block

* Mount
* Chmod
  + EIO An I/O error occurred.
* Kill
  + EPERM The process does not have permission to send the signal to any of the target processes.

**Traps**

A trap, also known as an exception or a fault, is typically a type of [synchronous](https://en.wikipedia.org/wiki/Synchronization_(computer_science)) [interrupt](https://en.wikipedia.org/wiki/Interrupt) typically caused by an [exceptional](https://en.wikipedia.org/wiki/Exception_handling) condition. A trap usually results in a switch to [kernel mode](https://en.wikipedia.org/wiki/Kernel_mode), wherein the operating system performs some action before returning control to the originating process. A trap in a system [process](https://en.wikipedia.org/wiki/Process_(computing)) is more serious than a trap in a user [process](https://en.wikipedia.org/wiki/Process_(computing)), and in some systems is fatal. In some usages, the term trap refers specifically to an interrupt intended to initiate a [context switch](https://en.wikipedia.org/wiki/Context_switch) to a [monitor program](https://en.wikipedia.org/wiki/Profiler_(computer_science)) or [debugger](https://en.wikipedia.org/wiki/Debugger).

Deriving from this original usage, trap is sometimes used for the mechanism of intercepting normal control flow in some domains.

In [SNMP](https://en.wikipedia.org/wiki/Simple_Network_Management_Protocol), a trap is a type of [PDU](https://en.wikipedia.org/wiki/Protocol_data_unit) used to report an alert or other asynchronous event about a managed subsystem.