Process Management
pub unsafe fn fork() -> Result<ForkResult> Create a child process by duplicating the parent process
pub fn waitpid(pid: pid_t, options: Option<WaitPidFlag>) -> Result<WaitStatus> Wait for a child to terminate
pub fn wait() -> Result<WaitStatus> Old version of waitPid
pub fn execve<SA: AsRef<CStr> , SE: AsRef<CStr>> (path: &CStr, args: &[SA], env: &[SE]) -> Result<Infallible> replace a process core image
pub fn exit(code: i32) -> ! rust version of exit (! means never)
pub fn getpid() -> Pid Get caller's Process id
pub fn getppid() -> Pid Get caller's parent's Process id
pub fn getpgrp() -> Pid Get caller's process group id
pub fn setsid() -> Result<Pid> Create new session and set process group id

Signals
pub unsafe fn sigaction(signal: Signal, sigaction: &SigAction) -> Result<SigAction> Define actions to take on signals
pub fn sigprocmask(how: SigmaskHow, set: Option<&SigSet> , oldset: Option<&mut SigSet>) -> Result<()> Examine, or change the signal mask
pub fn kill<T: Into<Option<Signal>> > (pid: Pid, signal: T) -> Result<()> Send a signal to a process
pub fn killpg<T: Into<Option<Signal>> > (pgrp: Pid, signal: T) -> Result<()> Send a signal to a process group
pub fn set(secs: c_uint) -> Option<c_uint> Schedule an alarm signal
pub fn pause() Suspend the caller until the next signal is received

File Management
pub fn mknod<P: ?Sized + NixPath> (path: &P, kind: SFlag, perm: Mode, dev: dev_t) -> Result<()> Create a regular, special or directory i-node
pub fn open<P: ?Sized + NixPath> (path: &P, oflag: OFlag, mode: Mode) -> Result<RawFd> Open a file for reading/writing or both
pub fn close(fd: RawFd) -> Result<()> Close a raw file descriptor (close a file)
pub fn read(fd: RawFd, buf: &mut [u8]) -> Result<usize> read from a raw file descriptor
pub fn write(fd: RawFd, buf: &[u8]) -> Result<usize> Write to a raw file descriptor
pub fn lseek(fd: RawFd, offset: off_t, whence: Whence) -> Result<off_t> Move the read/write file offset
pub fn stat<P: ?Sized + NixPath> (path: &P) -> Result<FileStat> Get a file's status information
pub fn fstat(fd: RawFd) -> Result<FileStat> Get a file's status information
pub fn dup(oldfd: RawFd) -> Result<RawFd> Create a copy of the specified file descriptor
pub fn pipe() -> Result<(RawFd, RawFd)> Create an interprocess channel (i.i., a pipe)
pub fn access<P: ?Sized + NixPath> (path: &P, amode: AccessFlags) -> Result<()> Check a file's accessibility

File System Management
pub fn mkdir<P: ?Sized + NixPath> (path: &P, mode: Mode) -> Result<()> Create a new directory
pub fn unlink<P: ?Sized + NixPath> (path: &P) -> Result<()> Remove a directory entry
pub fn mount<P1: ?Sized + NixPath, P2: ?Sized + NixPath, P3: ?Sized + NixPath, P4: ?Sized + NixPath> (source: Option<&P1> , target: &P2, fstype: Option<&P3> , flags: MsFlags, data: Option<&P4>) -> Result<()> Mount a filesystem
pub fn umount<P: ?Sized + NixPath> (target: &P) -> Result<()> Unmount a file system
pub fn sync() Flush all cached blocks to the disk
pub fn chdir<P: ?Sized + NixPath> (path: &P) -> Result<()> Change the working directory
pub fn chroot<P: ?Sized + NixPath> (path: &P) -> Result<()> Change the root directory

Protection
pub fn fchmod(fd: RawFd, mode: Mode) -> Result<()> Change a file's protection bits
pub fn getuid() -> Uid Get the callers user ID
pub fn getgid() -> Gid Get the caller's group ID
pub fn setuid(uid: Uid) -> Result<()> Set the user ID
pub fn setgid(gid: Gid) -> Result<()> Set the group ID
pub fn chown<P: ?Sized + NixPath> (path: &P, owner: Option<Uid> , group: Option<Gid>) -> Result<()> Change a file's owner and group
pub fn umask(mode: Mode) -> Mode Change the mode mask