Linux/UNIX System Calls and C Library Functions

```
Program Termination
-----------------------------------------------------------------------
void exit(int status)
void abort(void)
int atexit(void (*function)(void))

Errors
-----------------------------------------------------------------------
extern int errno
void perror(const char *message)
char *strerror(int errnum)

C String Library
-----------------------------------------------------------------------
int bcmp(const void *s1, const void *s2, size_t n)
void bcopy(const void *src, void *dest, size_t n)
void bzero(void *s, size_t n)
char *index(const char *s, int c)
char *rindex(const char *s, int c)
void *memchr(const void *s, int c, size_t n)
void *memrchr(const void *s, int c, size_t n)
int memcmp(const void *s1, const void *s2, size_t n)
void *memcpy(void *dest, const void *src, size_t n)
void *memccpy(void *dest, const void *src, int c, size_t n)
void *mempcpy(void *dest, const void *src, size_t n)
wchar_t *wmempcpy(wchar_t *dest, const wchar_t *src, size_t n)
void *memmove(void *dest, const void *src, size_t n)
void *memset(void *s, int c, size_t n)
char *strcat(char *dest, const char *src)
char *strncat(char *dest, const char *src, size_t n)
char *strchr(const char *s, int c)
char *strrchr(const char *s, int c)
int strcmp(const char *s1, const char *s2)
int strncmp(const char *s1, const char *s2, size_t n)
char *strcpy(char *dest, const char *src)
char *strncpy(char *dest, const char *src, size_t n)
char *strdup(const char *s)
char *strndup(const char *s, size_t n)
size_t strlen(const char *s)
char *strpbrk(const char *s, const char *accept)
char *strsep(char **stringp, const char *delim)
size_t strspn(const char *s, const char *accept)
size_t strcspn(const char *s, const char *reject)
char *strstr(const char *haystack, const char *needle)
char *strtok(char *s, const char *delim)

C Library I/O (Streams, FILE*)
-----------------------------------------------------------------------
FILE *fopen(const char *path, const char *mode)
  mode: r, r+, w, w+, a, a+ [as a string]
FILE *fdopen(int fildes, const char *mode)
FILE *freopen(const char *path, const char *mode, FILE *stream)
int fclose(FILE *stream)
int printf(const char *format, ...)
int fprintf(FILE *stream, const char *format, ...)
int sprintf(char *str, const char *format, ...)
int snprintf(char *str, size_t size, const char *format, ...)
int putc(int c, FILE *stream)
int putchar(int c)
int fputc(int c, FILE *stream)
int puts(const char *s)
int fputs(const char *s, FILE *stream)
int scanf(const char *format, ...)
int fscanf(FILE *stream, const char *format, ...)
int sscanf(const char *str, const char *format, ...)
int getc(FILE *stream)
int getchar(void)
int fgetc(FILE *stream)
char *gets(char *s)
char *fgets(char *s, int size, FILE *stream)
int ungetc(int c, FILE *stream)
int fflush(FILE *stream)
void rewind(FILE *stream)
int fseek(FILE *stream, long offset, int whence)
long ftell(FILE *stream)
int fgetpos(FILE *stream, fpos_t *pos)
int fsetpos(FILE *stream, fpos_t *pos)
int feof(FILE *stream)
int ferror(FILE *stream)
void clearerr(FILE *stream)
int fileno(FILE *stream)

C Memory Management
-----------------------------------------------------------------------
void *calloc(size_t num, size_t size)
void *malloc(size_t num)
void *realloc(void *ptr, size_t num)
void free(void *ptr)

Command-Line Option Decoding
-----------------------------------------------------------------------
int getopt(int argc, char * const argv[], const char *optstring)
extern char *optarg
extern int optind, opterr, optopt
int getopt_long(int argc, char * const argv[], const char *optstring,
                const struct option *longopts, int *longindex)
int getopt_long_only(int argc, char * const argv[], const char *optstring,
                     const struct option *longopts, int *longindex)
  struct option:
    {const char *name;
     int has_arg;
     int *flag;
     int val;}

System Call I/O (File Descriptors)
-----------------------------------------------------------------------
int open(const char *pathname, int flags)
  flags: O_RDONLY, O_WRONLY, O_RDWR, O_CREAT, O_EXCL, O_TRUNC, O_APPEND, etc.
int open(const char *pathname, int flags, mode_t mode)
  mode: octal permissions or use symbolic constants S_I????
int close(int fd)
ssize_t read(int fd, void *buffer, size_t n)
ssize_t write(int fd, const void *buffer, size_t n)
ssize_t pread(int fd, void *buf, size_t count, off_t offset)
ssize_t pwrite(int fd, const void *buf, size_t count, off_t offset)
off_t lseek(int fd, off_t offset, int start_flag)
  start_flag: SEEK_SET, SEEK_CUR, or SEEK_END

File Management and Information
-----------------------------------------------------------------------
int creat(const char *pathname, mode_t mode)
int rename(const char *oldpath, const char *newpath)
int remove(const char *pathname)
int unlink(const char *pathname)
int link(const char *original_pathname, const char *new_pathname)
int symlink(const char *real_pathname, const char *sym_pathname)
int readlink(const char *sym_pathname, char *buffer, size_t buffsize)
int mknod(const char *pathname, mode_t mode, dev_t dev)
mode_t umask(mode_t newmask)
int access(const char *pathname, int amode)
int chmod(const char *pathname, mode_t newmode)
int fchmod(int fildes, mode_t mode)
int chown(const char *pathname, uid_t owner_id, gid_t group_id)
int fchown(int fd, uid_t owner, gid_t group)
int utime(const char *filename, struct utimbuf *buf)
int stat(const char *pathname, struct stat *buffer)
  struct stat:
    {dev_t st_dev;  ino_t st_ino;  mode_t st_mode;
     nlink_t st_nlink;  uid_t st_uid;  gid_t st_gid;
     dev_t st_rdev;  off_t st_size;  unsigned long st_blksize;
     unsigned long st_blocks; time_t st_atime;  time_t st_mtime;
     time_t st_ctime;}
int lstat(const char *pathname, struct stat *buffer)

File Descriptor Control and Information
-----------------------------------------------------------------------
int dup(int oldfd)
int dup2(int oldfd, int newfd)
int fcntl(int fd, int cmd, ...)
  cmd: F_GETFL, F_SETFL, etc.
int ioctl(int fd, int request, ...)
int fstat(int fd, struct stat *buffer)

Directories
-----------------------------------------------------------------------
DIR *opendir(const char *name)
int closedir(DIR *dir)
struct dirent *readdir(DIR *dir)
  struct dirent:
    {long d_ino;                /* inode number */
     off_t d_off;               /* offset to this dirent */
     unsigned short d_reclen;   /* length of this d_name */
     char d_name [NAME_MAX+1];} /* filename (null-terminated) */
void rewinddir(DIR *dir)
void seekdir(DIR *dir, off_t offset)
off_t telldir(DIR *dir)
int mkdir(const char *pathname, mode_t mode)
int rmdir(const char *pathname)
int chdir(const char *path)
char *getcwd(char *buf, size_t size)
int ftw(const char *dir,
        int (*fn)(const char *file, const struct stat *sb, int flag),
        int depth)

Processes
-----------------------------------------------------------------------
pid_t fork(void)
int execl(const char *path, const char *arg, ...)
int execlp(const char *file, const char *arg, ...)
int execle(const char *path, const char *arg, ..., char * const envp[])
int execv(const char *path, char *const argv[])
int execvp(const char *file, char *const argv[])
int execve(const char *filename, char *const argv [], char *const envp[])
pid_t wait(int *status)
  status macros: WIFEXITED, WEXITSTATUS, WIFSIGNALED, WTERMSIG, etc.
pid_t waitpid(pid_t pid, int *status, int options)
  options: WNOHANG, WUNTRACED, etc.
void exit(int status)
void _exit(int status)
int atexit(void (*function)(void))
void abort(void)
pid_t getpid(void)
pid_t getppid(void)
FILE *popen(const char *command, const char *type)
int pclose(FILE *stream)
int system(const char *string)

Environment Manipulation
-----------------------------------------------------------------------
char *getenv(const char *name)
int putenv(char *string)
int setenv(const char *name, const char *value, int overwrite)
int unsetenv(const char *name)
int clearenv(void)
```

```
Signals
----------------------------------------------------------------------
sighandler_t signal(int signum, sighandler_t handler)
  sighandler_t: void (*sighandler)(int)
  sighandler: a function or SIG_IGN or SIG_DFL
int sigaction(int signum, const struct sigaction *act,
              struct sigaction *oldact)
  struct sigaction:
    {void (*sa_handler)(int);
     sigset_t sa_mask;
     int sa_flags;
     void (*sa_sigaction)(int, siginfo_t *, void *);}
  sa_handler: a function or SIG_IGN or SIG_DFL
  sa_flags: SA_RESTART, SA_RESETHAND, SA_NOMASK,
            SA_NODEFER, SA_NOCLDSTOP, etc.
  [use only sa_handler or sa_sigaction, but not both, and
  sa_flags must include SA_SIGINFO in order to use sa_sigaction]
int sigemptyset(sigset_t *set)
int sigfillset(sigset_t *set)
int sigaddset(sigset_t *set, int signum)
int sigdelset(sigset_t *set, int signum)
int sigismember(const sigset_t *set, int signum)
int sigprocmask(int how, const sigset_t *set, sigset_t *oldset)
  how: SIG_SETMASK or SIG_BLOCK or SIG_UNBLOCK.
int sigpending(sigset_t *set)
int sigsuspend(const sigset_t *mask)
int setjmp(jmp_buf env)
int sigsetjmp(sigjmp_buf env, int savesigs)
void longjmp(jmp_buf env, int val)
void siglongjmp(sigjmp_buf env, int val)
int kill(pid_t pid, int sig)
int raise (int sig)
int pause(void)
unsigned int alarm(unsigned int seconds)
unsigned int sleep(unsigned int seconds)
int getitimer(int which, struct itimerval *val)
  struct itimerval:
    {struct timeval it_interval; /* next value */
     struct timeval it_value;}   /* current value */
  struct timeval:
    {long tv_sec;   /* seconds */
     long tv_usec;} /* microseconds */
int setitimer(int which, const struct itimerval *val, struct itimerval *oval)

Pipes and FIFOs
----------------------------------------------------------------------
int pipe(int filedes[2])
int mkfifo(const char *pathname, mode_t mode)

Sockets
----------------------------------------------------------------------
int socket(int domain, int type, int protocol)
  domain: PF_INET, PF_INET6, PF_UNIX, etc. [may also use AF_*]
  type: SOCK_STREAM, SOCK_DGRAM, SOCK_RAW, SOCK_SEQPACKET
  protocol: 0 for default or IPPROTO_TCP, IPPROTO_UDP, IPPROTO_SCTP, etc.
int bind(int sockfd, struct sockaddr *my_addr, int addrlen)
  struct sockaddr_in:
    {sa_family_t sin_family;  /* address family: AF_INET */
     u_int16_t sin_port;      /* port in network byte order */
     struct in_addr sin_addr} /* internet address */
  struct in_addr:
    {u_int32_t s_addr} /* IPv4 address in network byte order */
int listen(int s, int backlog)
int accept(int s, struct sockaddr *addr, int *addrlen)
int connect(int sockfd, struct sockaddr *serv_addr, int addrlen)

int send(int s, const void *msg, size_t len, int flags)
int recv(int s, void *buf, size_t len, int flags)
int sendto(int s, const void *msg, size_t len, int flags,
           const struct sockaddr *to, socklen_t tolen)
int recvfrom(int s, void *buf, size_t len, int flags,
             struct sockaddr *from, socklen_t *fromlen);
int sendmsg(int s, const struct msghdr *msg, int flags)
int recvmsg(int s, struct msghdr *msg, int flags)

unsigned long int htonl(unsigned long int hostlong)
unsigned short int htons(unsigned short int hostshort)
unsigned long int ntohl(unsigned long int netlong)
unsigned short int ntohs(unsigned short int netshort)
in_addr_t inet_addr(const char *cp)
int inet_aton(const char *cp, struct in_addr *inp)
char *inet_ntoa(struct in_addr in)
struct hostent *gethostbyname(const char *name)
  struct hostent:
    {char *h_name;        /* official name of host */
     char **h_aliases;    /* alias list */
     int h_addrtype;      /* host address type */
     int h_length;        /* length of address */
     char **h_addr_list;} /* list of addresses */
struct hostent *gethostbyaddr(const char *addr, int len, int type)
extern int h_errno
void herror(const char *s)
const char *hstrerror(int err)
int getaddrinfo(const char *node, const char *service,
                const struct addrinfo *hints, struct addrinfo **res)
  struct addrinfo:
    {int              ai_flags;
     int              ai_family;
     int              ai_socktype;
     int              ai_protocol;
     size_t           ai_addrlen;
     struct sockaddr *ai_addr;
     char            *ai_canonname;
     struct addrinfo *ai_next;}
const char *gai_strerror(int errcode)
```

```
I/O Multiplexing
----------------------------------------------------------------------
int select(int n, fd_set *readfds, fd_set *writefds, fd_set *exceptfds,
           struct timeval *timeout)
int pselect(int n, fd_set *readfds, fd_set *writefds, fd_set *exceptfds,
            const struct timespec *timeout, const sigset_t *sigmask)
  fd_set macros:  FD_CLR(int fd, fd_set *set)
                  FD_ISSET(int fd, fd_set *set)
                  FD_SET(int fd, fd_set *set)
                  FD_ZERO(fd_set *set)
int poll(struct pollfd *ufds, unsigned int nfds, int timeout)
  struct pollfd:
    {int   fd;       /* file descriptor */
     short events;   /* requested events */
     short revents;} /* returned events */

Threads
----------------------------------------------------------------------
Control:
int pthread_create(pthread_t *restrict thread,
                   const pthread_attr_t *restrict attr,
                   void *(*start_routine)(void*), void *restrict arg)
void pthread_exit(void *value_ptr)
int pthread_join(pthread_t thread, void **value_ptr)
int pthread_detach(pthread_t thread)
int pthread_attr_destroy(pthread_attr_t *attr)
int pthread_attr_init(pthread_attr_t *attr)
int pthread_attr_getdetachstate(const pthread_attr_t *attr, int *detachstate)
int pthread_attr_setdetachstate(pthread_attr_t *attr, int detachstate)
void pthread_cleanup_pop(int execute)
void pthread_cleanup_push(void (*routine)(void*), void *arg)
int pthread_key_create(pthread_key_t *key, void (*destructor)(void*))

Conditions:
int pthread_cond_init(pthread_cond_t *restrict cond,
                      const pthread_condattr_t *restrict attr)
int pthread_cond_destroy(pthread_cond_t *cond)
int pthread_cond_broadcast(pthread_cond_t *cond)
int pthread_cond_signal(pthread_cond_t *cond)
int pthread_cond_wait(pthread_cond_t *restrict cond,
                      pthread_mutex_t *restrict mutex)
int pthread_cond_timedwait(pthread_cond_t *restrict cond,
                           pthread_mutex_t *restrict mutex,
                           const struct timespec *restrict abstime)
int pthread_condattr_init(pthread_condattr_t *attr)
int pthread_condattr_destroy(pthread_condattr_t *attr)

Mutexes:
int pthread_mutex_init(pthread_mutex_t *restrict mutex,
                       const pthread_mutexattr_t *restrict attr)
int pthread_mutex_destroy(pthread_mutex_t *mutex)
int pthread_mutex_lock(pthread_mutex_t *mutex)
int pthread_mutex_trylock(pthread_mutex_t *mutex)
int pthread_mutex_unlock(pthread_mutex_t *mutex)

Signals:
int pthread_kill(pthread_t thread,int sig)
int pthread_sigmask(int how, const sigset_t *set, sigset_t *oldset)
int sigprocmask(int how, const sigset_t *set, sigset_t *oldset)
int sigwait(const sigset_t *restrict set,int *restrict sig)

Semaphores (POSIX)
----------------------------------------------------------------------
sem_t *sem_open(const char *name, int oflag)
sem_t *sem_open(const char *name, int oflag, mode_t mode, unsigned int value)
  oflag: O_CREAT, O_EXCL
int sem_close(sem_t *sem)
int sem_unlink(const char *name)
int sem_post(sem_t *sem)
int sem_wait(sem_t *sem)
int sem_trywait(sem_t *sem)
int sem_timedwait(sem_t *sem, const struct timespec *abs_timeout)
int sem_getvalue(sem_t *sem, int *sval)
int sem_init(sem_t *sem, int pshared, unsigned int value)
int sem_destroy(sem_t *sem)

Memory Mapping
----------------------------------------------------------------------
void *mmap(void *start, size_t length, int prot, int flags,
           int fd, off_t offset)
  prot: PROT_READ, PROT_WRITE, PROT_EXEC, PROT_NONE
  flags: MAP_SHARED, MAP_PRIVATE, MAP_ANONYMOUS, etc.
int munmap(void *start, size_t length)
int msync(void *start, size_t length, int flags)

Shared Memory (POSIX)
----------------------------------------------------------------------
int shm_open(const char *name, int oflag, mode_t mode)
  oflag: O_RDONLY, O_RDWR, O_CREAT, O_EXCL, O_TRUNC
int shm_unlink(const char *name)

File Locking
----------------------------------------------------------------------
int flock(int fd, int operation)
  operation: LOCK_SH or LOCK_EX or LOCK_UN
int lockf(int fd, int cmd, off_t len)
  cmd: F_LOCK, F_TLOCK or F_ULOCK or F_TEST
int fcntl(int fd, int cmd, struct flock *lock)
  cmd: F_GETLK or F_SETLK or F_SETLKW
  struct flock:
    {short l_type;   /* Lock type: F_RDLCK, F_WRLCK, F_UNLCK */
     short l_whence; /* SEEK_SET, SEEK_CUR, or SEEK_END */
     off_t l_start;  /* Starting offset for lock */
     off_t l_len;    /* Number of bytes to lock */
     pid_t l_pid;}   /* PID of process blocking F_GETLK */
```