```
#include <stdio.h>

            FILE *fopen(const char *path, const char *mode);
            int fclose(FILE *fp);
            int printf(const char *format, ...);
            int fprintf(FILE *stream, const char *format, ...);
            int scanf(const char *format, ...);
            int fscanf(FILE *stream, const char *format, ...);
            int fgetc(FILE *stream);
            int getc(FILE *stream); (macro)
            int getchar(void);
            int ungetc(int c, FILE *stream);
            int fputc(int c, FILE *stream);
            int putc(int c, FILE *stream); (macro)
            int putchar(int c);
            int fseek(FILE *stream, long offset, int whence);
                  whence = SEEK_SET, SEEK_CUR, SEEK_END
            long ftell(FILE *stream);
            void rewind(FILE *stream);
            size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);
            size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);
            FILE *popen( const char *cmd, const char *mode );
            int pclose( FILE *stream );

    #include <sys/types.h>
    #include <sys/stat.h>
    #include <fcntl.h>
    #include <unistd.h>

            int open(const char *pathname, int flags);
            int open(const char *pathname, int flags, mode_t mode);
            int close(int fd);
            ssize_t read(int fd, void *buf, size_t count);
            ssize_t write(int fd, const void *buf, size_t count);
            off_t lseek(int fildes, off_t offset, int whence);
            int fstat(int filedes, struct stat *buf);
            int stat(const char *file_name, struct stat *buf);
            int lstat(const char *file_name, struct stat *buf);
            int pipe( int fd[ ] );
            int dup( int fd );
            int dup2( int fd1, int fd2 );

    #include <sys/types.h>
    #include <sys/stat.h>
    #include <unistd.h>

            int mkdir(const char *pathname, mode_t mode);
            int rmdir(const char *pathname);
            int fchmod(int fildes, mode_t mode);
            int chmod(const char *path, mode_t mode);
            int link(const char *oldpath, const char *newpath);
            int unlink(const char *pathname);
            int symlink(const char *oldpath, const char *newpath);
            char *getcwd(char *buf, size_t size);
            int chdir(const char *path);

    #include <unistd.h>

            int getopt(int argc, char * const argv[], const char *optstring);
            extern char *optarg;
            extern int optind, opterr, optopt;

    #include <stdlib.h>

            char *getenv(const char *name);
            int putenv(char *string);
```

```
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;
   blksize_t st_blksize;
   blkcnt_t st_blocks;
   time_t st_atime;
   time_t st_mtime;
   time_t st_ctime;
};


struct tm {
   int tm_sec;
   int tm_min;
   int tm_hour;
   int tm_mday;
   int tm_mon;
   int tm_year;
   int tm_wday;
   int tm_yday;
   int tm_isdst;
};

struct utsname{
   char sysname[];
   char nodename[];
   char release[];
   char version[];
   char machine[];
   char domainname[];
};


struct passwd{
   char *pw_name;
   char *pw_passwd;
   uid_t pw_uid;
   gid_t pw_gid;
   char *pw_gecos;
   char *pw_dir;
   char *pw_shell;
};

struct sigaction{
   void (*sa_handler)( int );
   void (*sa_sigaction)( int, siginfo_t *, void * );
   sigset_t sa_mask;
   int sa_flags;
   void (*sa_restorer)( void );
};
```

```
#include <time.h>

        time_t time(time_t *t);
        double difftime(time_t time1, time_t time0);
        struct tm *gmtime(const time_t *timep);
        struct tm *localtime(const time_t *timep);
        time_t mktime(struct tm *tm);
        char *asctime(const struct tm *tm);
        char *ctime(const time_t *timep);
        size_t strftime(char *s, size_t max, const char *format, const struct tm *tm);
        char *strptime(const char *s, const char *format, struct tm *tm);

#include <unistd.h>
#include <sys/types.h>

        uid_t getuid(void);
        uid_t geteuid(void);
        pid_t getpid(void);
        pid_t getppid(void);
        struct passwd *getpwnam(const char *name);
        struct passwd *getpwuid(uid_t uid);

#include <stdlib.h>

        int system ( const char *string );

#include <unistd.h>

        char **environ;
        int execl ( const char *path, const char *arg0, ..., (char *)0 );
        int execlp ( const char *file, const char *arg0, ..., (char *)0 );
        int execle ( const char *path, const char *arg0, ..., (char *)0 , char *const envp[] );
        int execv ( const char *path, char *const argv[] );
        int execvp ( const char *file, char *const argv[] );
        int execve ( const char *path, char *const argv[], char *const envp[] );

#include <sys/types.h>
#include <unistd.h>

        pid_t fork ( void );

#include <stdlib.h>

         void qsort(void *base, size_t nmemb, size_t size,  int(*compar)(const void *, const void *))

#include < sys/utsname.h>

        int uname( struct utsname *buff );

#include <pthread.h>

        int   pthread_join(pthread_t, void **);
        int   pthread_create(pthread_t *, const pthread_attr_t *, void *(*)(void *), void *);
        void  pthread_exit(void *);

#include <signal.h>
        void (*signal(int _sig, void (*_func)(int)))(int);
        int kill(pid_t pid, int sig);
        int sigaction(int sig, struct sigaction *act, struct sigaction *oldact);
        int sigemptyset(sigset_t *mask);

#include <sys/wait.h>
        pid_t wait(int *stat_loc);

#include <semaphore.h>
        int sem_init( sem_t *sem, int pshared, unsigned int value );
        int sem_wait( sem_t *sem );
        int sem_post( sem_t *sem );
        int sem_destroy( sem_t * sem );
```

```
constants
O_APPEND
O_ASYNC
O_CLOEXEC
O_CREAT
  S_IRWXU
  S_IRUSR
  S_IWUSR
  S_IXUSR
  S_IRWXG
  S_IRGRP
  S_IWGRP
  S_IXGRP
  S_IRWXO
  S_IROTH
  S_IWOTH
  S_IXOTH
O_DIRECT
O_DIRECTORY
O_EXCL
O_LARGEFILE
O_NOATIME
O_NOCTTY
O_NOFOLLOW
O_NONBLOCK
O_SYNC
O_TRUNC
```