Version: 1.3
Note: do not use popen(),pclose(),system() for solving the exam problems!

Posix::Files and Directories
```
int open(const char *pathname, int flags);
int open(const char *pathname, int flags, mode_t mode);
int creat(const char *pathname, mode_t mode);
        mode:
        O_CREAT, O_EXCL,  O_TRUNC, O_APPEND,
        O_RDONLY, O_WRONLY, O_RDWR
        S_IRWXU, S_IRUSR, S_IWUSR, S_IXUSR,
        S_IRWXG, S_IRGRP, S_IWGRP, S_IXGRP,
        S_IRWXO, S_IROTH, S_IWOTH, S_IXOTH
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);
        whence: SEEK_SET, SEEK_CUR, SEEK_END
int dup(int oldfd);
int dup2(int oldfd, int newfd);
int select(int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds,
        struct timeval *timeout);
        void FD_CLR(int fd, fd_set *set);
        int FD_ISSET(int fd, fd_set *set);
        void FD_SET(int fd, fd_set *set);
        void FD_ZERO(fd_set *set);
int stat(const char *path, struct stat *buf);
int fstat(int filedes, struct stat *buf);
int lstat(const char *path, struct stat *buf);
 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;
                                };
        macros:
        S_ISREG(m), S_ISDIR(m), S_ISCHR(m), S_ISBLK(m),
        S_ISFIFO(m), S_ISLNK(m), S_ISSOCK(m)
DIR *opendir(const char *name);
struct dirent *readdir(DIR *dir);
int closedir(DIR *dir);
        struct dirent {
                ...
                ino_t     d_ino;
                char      d_name[256]; };
char *getcwd(char *buf, size_t size);
```

Posix::Processes
```
pid_t fork(void);
pid_t vfork(void);
void _exit(int status);
void exit(int status);
int atexit(void (*function)(void));
int on_exit(void (*function)(int , void *), void *arg);
void abort(void);
int execve(const char *filename, char *const argv[], char *const envp[]);
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 system(const char *command);
pid_t wait(int *status);
pid_t waitpid(pid_t pid, int *status, int options);
pid_t getpid(void);
pid_t getppid(void);
```

Posix::Pipes
```
int pipe(int filedes[2]);
int mkfifo(const char *pathname, mode_t mode);
FILE *popen(const char *command, const char *type);
int pclose(FILE *stream);
```

Posix::Threads
```
int pthread_create(pthread_t *thread, const pthread_attr_t *attr,
        void *(*start_routine)(void*), void *arg);
int pthread_join(pthread_t thread, void **value_ptr);
void pthread_exit(void *value_ptr);
```

Posix::Signals
```
typedef void (*sighandler_t)(int);
sighandler_t signal(int signum, sighandler_t handler);
int sigaction(int sig, const struct sigaction act, struct sigaction oact);
        struct sigaction{
          void(*) (int) sa_handler;
          void (*sa_sigaction)(int, siginfo_t *, void *);
          sigset_t sa_mask;
          int sa_flags; };
        special handler values:
        SIG_IGN, SIG_DFL, SIG_ERR
        typical signals:
        SIGUSR1/2, SIGALRM, SIGINT, SIGTERM, SIGKILL
int kill(pid_t pid, int sig);
int raise(int sig);
unsigned int alarm(unsigned int seconds);
int sigprocmask(int how, const sigset_t *set, sigset_t *oldset);
        how: SIG_BLOCK, SIG_UNBLOCK, SIG_SETMASK
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);
```

Standard C Library::I/O
```
int fseek(FILE *stream, long offset, int whence);
long ftell(FILE *stream);
void rewind(FILE *stream);
int close(int fd);
FILE *fopen(const char *path, const char *mode);
FILE *fdopen(int fildes, const char *mode);
FILE *freopen(const char *path, const char *mode, 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);
int fputc(int c, FILE *stream);
int fputs(const char *s, FILE *stream);
int fgetc(FILE *stream);
char *fgets(char *s, int size, FILE *stream);
int fclose(FILE *fp);
int scanf(const char *format, ...);
int fscanf(FILE *stream, const char *format, ...);
int sscanf(const char *str, const char *format, ...);
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, ...);
```

Standard C Library::Strings
```
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);
size_t strlen(const char *s);
size_t strspn(const char *s, const char *accept);
char *strstr(const char *haystack, const char *needle);
char *strtok(char *s, const char *delim);

void *memcpy(void *dest, const void *src, size_t n);
void *memccpy(void *dest, const void *src, int c, size_t n);
void *memset(void *s, int c, size_t n);
```

Standard C Library:: Dynamic Memory
```
void *calloc(size_t nmemb, size_t size);
void *malloc(size_t size);
void free(void *ptr);
void *realloc(void *ptr, size_t size);
```

Miscellaneous
```
int rand(void); /* returns random number between 0 and RAND_MAX */
int rand_r(unsigned int *seedp);
void srand(unsigned int seed);
void perror(const char *s);
unsigned int sleep(unsigned int seconds);
```