附录A 函数原型

本附录包含了正文中说明过的标准UNIX、POSIX和ANSI C的函数原型。通常我们想了解的是函数的参数(fgets的哪一个参数是文件指针?)和返回值(sprintf返回的是指针还是计数值?)。

这些函数原型还说明了要包含哪些头文件，以获得特定常数的定义，或获得ANSI C函数原型，以帮助在编译时进行错误检测。

```
void     _exit(int status);
                    <unistd.h>
                    此函数不返回

void     abort(void);
                    <stdlib.h>
                    此函数不返回

int      access(const char *pathname, int mode);
                    <unistd.h>
                    mode: R_OK, W_OK, X_OK, F_OK
                    返回：若成功则为0，若出错则为－1

unsigned
int      alarm(unsigned int seconds);
                    <unistd.h>
                    返回：0或上次设置的alarm的剩余秒数

char    *asctime(const struct tm *tmptr);
                    <time.h>
                    返回：指向以null终止的字符串的指针

int      atexit(void (*func)(void));
                    <stdlib.h>
                    返回：若成功则为0，若出错则为非0

void    *calloc(size_t nobj, size_t size);
                    <stdlib.h>
                    返回：若成功则为非空指针，若出错则为NULL

speed_t  cfgetispeed(const struct termios *termptr);
                    <termios.h>
                    返回：波特率值

speed_t  cfgetospeed(const struct termios *termptr);
                    <termios.h>
                    返回：波特率值

int      cfsetispeed(struct termios *termptr, speed_t speed);
                    <termios.h>
                    返回：若成功则为0，若出错则为－1

int      cfsetospeed(struct termios *termptr, speed_t speed);
                    <termios.h>
                    返回：若成功则为0，若出错则为－1

int      chdir(const char *pathname);
                    <unistd.h>
                    返回：若成功则为0，若出错则为－1
```

```
int     chmod(const char *pathname, mode_t mode);
                <sys/types.h>
                <sys/stat.h>
                mode: S_IS[UG]ID, S_ISVTX, S_I[RWX](USR|GRP|OTH)
                返回：若成功则为0，若出错则为－1

int     chown(const char *pathname, uid_t owner, gid_t group);
                <sys/types.h>
                <unistd.h>
                返回：若成功则为0，若出错则为－1

void    clearerr(FILE *fp);
                <stdio.h>

int     close(int filedes);
                <unistd.h>
                返回：若成功则为0，若出错则为－1

int     closedir(DIR *dp);
                <sys/types.h>
                <dirent.h>
                返回：若成功则为0，若出错则为－1

void    closelog(void);
                <syslog.h>

int     creat(const char *pathname, mode_t mode);
                <sys/types.h>
                <sys/stat.h>
                <fcntl.h>
                mode: S_IS[UG]ID, S_ISVTX, S_I[RWX](USR|GRP|OTH)
                返回：若成功则为只写打开的文件描述符，若出错则为－1

char    *ctermid(char *ptr);
                <stdio.h>
                返回：控制终端的路径名

char    *ctime(const time_t *calptr);
                <time.h>
                返回：指向以null终止的字符串的指针

int     dup(int filedes);
                <unistd.h>
                返回：若成功则为新文件描述符，若出错则为－1

int     dup2(int filedes, int filedes2);
                <unistd.h>
                返回：若成功则为新文件描述符，若出错则为－1

void    endgrent(void);
                <sys/types.h>
                <grp.h>

void    endpwent(void);
                <sys/types.h>
                <pwd.h>

int     execl(const char *pathname, const char *arg0, ... /* (char *) 0 */ );
                <unistd.h>
                返回：若出错则为－1，若成功则无返回

int     execle(const char *pathname, const char *arg0, ... /* (char *) 0,
              char *const envp[] */ );
                <unistd.h>
                返回：若出错则为－1，若成功则无返回

int     execlp(const char *filename, const char *arg0, ... /* (char *) 0 */ );
                <unistd.h>
                返回：若出错则为－1，若成功则无返回
```

```
int     execv(const char *pathname, char *const argv[]);
                <unistd.h>
                返回：若出错则为－1，若成功则无返回

int     execve(const char *pathname, char *const argv[], char *const envp[]);
                <unistd.h>
                返回：若出错则为－1，若成功则无返回

int     execvp(const char *filename, char *const argv[]);
                <unistd.h>
                返回：若出错则为－1，若成功则无返回


void    exit(int status);
                <stdlib.h>
                无返回

int     fchdir(int filedes);
                <unistd.h>
                返回：若成功则为0，若出错则为－1

int     fchmod(int filedes, mode_t mode);
                <sys/types.h>
                <sys/stat.h>
                mode: S_IS[UG]ID, S_ISVTX, S_I[RWX](USR|GRP|OTH)
                返回：若成功则为0，若出错则为－1

int     fchown(int filedes, uid_t owner, gid_t group);
                <sys/types.h>
                <unistd.h>
                返回：若成功则为0，若出错则为－1

int     fclose(FILE *fp);
                <stdio.h>
                返回：若成功则为0，若出错则为－1

int     fcntl(int filedes, int cmd, ... /* int arg */ );
                <sys/types.h>
                <unistd.h>
                <fcntl.h>
                cmd: F_DUPFD, F_GETFD, F_SETFD, F_GETFL, F_SETFL
                返回：若成功则取决于cmd，若出错则为－1

FILE   *fdopen(int filedes, const char *type);
                <stdio.h>
                type: "r", "w", "a", "r+", "w+", "a+",
                返回：若成功则为文件指针，若出错则为NULL

int     feof(FILE *fp);
                <stdio.h>
                返回：若已至流的文件尾端则为非0值(真)，否则为0 (假)

int     ferror(FILE *fp);
                <stdio.h>
                返回：若流出错则为非0值(真)，否则为0 (假)

int     fflush(FILE *fp);
                <stdio.h>
                返回：若成功则为0，若出错则为EOF

int     fgetc(FILE *fp);
                <stdio.h>
                返回：若成功则为下一个字符，若已至文件尾端或出错则为 EOF

int     fgetpos(FILE *fp, fpos_t *pos);
                <stdio.h>
                返回：若成功则为0，若出错则为非0
```

```
char    *fgets(char *buf, int n, FILE *fp);
                    <stdio.h>
                    返回：若成功则为buf，若已至文件尾端或出错则为NULL

int     fileno(FILE *fp);
                    <stdio.h>
                    返回：与该流相结合的文件描述符

FILE    *fopen(const char *pathname, const char *type);
                    <stdio.h>
                    type: "r", "w", "a", "r+", "w+", "a+",
                    返回：若成功则为文件指针，若出错则为NULL

pid_t   fork(void);
                    <sys/types.h>
                    <unistd.h>
                    返回：子进程中为0，父进程中为子进程ID，若出错则为-1

long    fpathconf(int filedes, int name);
                    <unistd.h>
                    name: _PC_CHOWN_RESTRICTED, _PC_LINK_MAX, _PC_MAX_CANON,
                          _PC_MAX_INPUT, _PC_NAME_MAX, _PC_NO_TRUNC,
                          _PC_PATH_MAX, _PC_PIPE_BUF, _PC_VDISABLE
                    返回：若成功则为相应值，若出错则为－1

int     fprintf(FILE *fp, const char *format, ...);
                    <stdio.h>
                    返回：若成功则为已输出的字符数，若出错则为负值

int     fputc(int c, FILE *fp);
                    <stdio.h>
                    返回：若成功则为c，若出错则为EOF

int     fputs(const char *str, FILE *fp);
                    <stdio.h>
                    返回：若成功则为非负值，若出错则为EOF

size_t  fread(void *ptr, size_t size, size_t nobj, FILE *fp);
                    <stdio.h>
                    返回：读到的对象数

void    free(void *ptr);
                    <stdlib.h>

FILE    *freopen(const char *pathname, const char *type, FILE *fp);
                    <stdio.h>
                    type: "r", "w", "a", "r+", "w+", "a+",
                    返回：若成功则为文件指针，若出错则为NULL

int     fscanf(FILE *fp, const char *format, ...);
                    <stdio.h>
                    返回：赋值的输入项数，若在任一变换前输入出错或EOF，则返回EOF

int     fseek(FILE *fp, long offset, int whence);
                    <stdio.h>
                    whence: SEEK_SET, SEEK_CUR, SEEK_END
                    返回：若成功则为0，若出错则为非0

int     fsetpos(FILE *fp, const fpos_t *pos);
                    <stdio.h>
                    返回：若成功则为0，若出错则为非0

int     fstat(int filedes, struct stat *buf);
                    <sys/types.h>
                    <sys/stat.h>
                    返回：若成功则为0，若出错则为－1

int     fsync(int filedes);
                    <unistd.h>
                    返回：若成功则为0，若出错则为－1
```

```
long      ftell(FILE *fp);
                    <stdio.h>
                    返回：若成功则为当前文件位置指示器，若出错则为－1 L

int       ftruncate(int filedes, off_t length);
                    <sys/types.h>
                    <unistd.h>
                    返回：若成功则为0，若出错则为－1

size_t    fwrite(const void *ptr, size_t size, size_t nobj, FILE *fp);
                    <stdio.h>
                    返回：写的对象数

int       getc(FILE *fp);
                    <stdio.h>
                    返回：若成功则为下一个字符，若已至文件尾端或出错则为 EOF

int       getchar(void);
                    <stdio.h>
                    返回：若成功则为下一个字符，若已至文件尾端或出错则为 EOF

char     *getcwd(char *buf, size_t size);
                    <unistd.h>
                    返回：若成功则为buf，若出错则为NULL

gid_t     getegid(void);
                    <sys/types.h>
                    <unistd.h>
                    返回：调用进程的有效组I D

char     *getenv(const char *name);
                    <stdlib.h>
                    返回：与name相关连的value的指针，若没有找到，则为NULL

uid_t     geteuid(void);
                    <sys/types.h>
                    <unistd.h>
                    返回：调用进程的有效用户I D

gid_t     getgid(void);
                    <sys/types.h>
                    <unistd.h>
                    返回：调用进程的实际组I D

struct
group    *getgrent(void);
                    <sys/types.h>
                    <grp.h>
                    返回：若成功则为指针，若已至文件尾端或出错则为 NULL

struct
group    *getgrgid(gid_t gid);
                    <sys/types.h>
                    <grp.h>
                    返回：若成功则为指针，若出错则为NULL

struct
group    *getgrnam(const char *name);
                    <sys/types.h>
                    <grp.h>
                    返回：若成功则为指针，若出错则为NULL

int       getgroups(int gidsetsize, gid_t grouplist[]);
                    <sys/types.h>
                    <unistd.h>
                    返回：若成功则为添加组I 数，若出错则为 - 1

int       gethostname(char *name, int namelen);
                    <unistd.h>
                    返回：若成功则为0，若出错则为－1
```

```
char    *getlogin(void);
                    <unistd.h>
                    返回：若成功则为login名字符串的指针，若出错则为NULL

int     getmsg(int filedes, struct strbuf *ctlptr, struct strbuf *dataptr, int *flagptr);
                    <stropts.h>
                    *flagptr: 0, RS_HIPRI
                    返回：若成功则为非负值，若出错则为－1

pid_t   getpgrp(void);
                    <sys/types.h>
                    <unistd.h>
                    返回：调用进程的进程组ID

pid_t   getpid(void);
                    <sys/types.h>
                    <unistd.h>
                    返回：调用进程的进程ID

int     getpmsg(int filedes, struct strbuf *ctlptr, struct strbuf *dataptr, int *bandptr,
            int *flagptr);
                    <stropts.h>
                    *flagptr: 0, MSG_HIPRI, MSG_BAND, MSG_ANY
                    返回：若成功则为非负值，若出错则为－1

pid_t   getppid(void);
                    <sys/types.h>
                    <unistd.h>
                    返回：调用进程的父进程ID

struct
passwd  *getpwent(void);
                    <sys/types.h>
                    <pwd.h>
                    返回：若成功则为指针，若已至文件尾端或出错则为NULL

struct
passwd  *getpwnam(const char *name);
                    <sys/types.h>
                    <pwd.h>
                    返回：若成功则为指针，若出错则为NULL

struct
passwd  *getpwuid(uid_t uid);
                    <sys/types.h>
                    <pwd.h>
                    返回：若成功则为指针，若出错则为NULL

int     getrlimit(int resource, struct rlimit *rlptr);
                    <sys/time.h>
                    <sys/resource.h>
                    返回：若成功则为0，若出错则为非0

char    *gets(char *buf);
                    <stdio.h>
                    返回：若成功则为buf，若已至文件尾或出错则为NULL

uid_t   getuid(void);
                    <sys/types.h>
                    <unistd.h>
                    返回：调用进程的实际用户ID

struct
tm      *gmtime(const time_t *calptr);
                    <time.h>
                    返回：指向一时间结构的指针

int     initgroups(const char *username, gid_t basegid);
                    <sys/types.h>
                    <unistd.h>
                    返回：若成功则为0，若出错则为－1
```

```
int       ioctl(int filedes, int request, ...);
                    <unistd.h>      /* SVR4 */
                    <sys/ioctl.h>   /* 4.3+BSD */
                    返回：若出错则为－1，若成功则为其他

int       isastream(int filedes);
                    返回：若为流设备则为1（真），否则为0（假）

int       isatty(int filedes);
                    <unistd.h>
                    返回：若为终端设备则为1（真），否则为0（假）

int       kill(pid_t pid, int signo);
                    <sys/types.h>
                    <signal.h>
                    返回：若成功则为0，若出错则为－1

int       lchown(const char *pathname, uid_t owner, gid_t group);
                    <sys/types.h>
                    <unistd.h>
                    返回：若成功则为0，若出错则为－1

int       link(const char *existingpath, const char *newpath);
                    <unistd.h>
                    返回：若成功则为0，若出错则为－1

struct
tm        *localtime(const time_t *calptr);
                    <time.h>
                    返回：指向一时间结构的指针

void      longjmp(jmp_buf env, int val);
                    <setjmp.h>
                    不返回

off_t     lseek(int filedes, off_t offset, int whence);
                    <sys/types.h>
                    <unistd.h>
                    whence: SEEK_SET, SEEK_CUR, SEEK_END
                    返回：若成功则为新的文件位移，若出错则为－1

int       lstat(const char *pathname, struct stat *buf);
                    <sys/types.h>
                    <sys/stat.h>
                    返回：若成功则为0，若出错则为－1

void      *malloc(size_t size);
                    <stdlib.h>
                    返回：若成功则为非空指针，若出错则为NULL

int       mkdir(const char *pathname, mode_t mode);
                    <sys/types.h>
                    <sys/stat.h>
                    mode: S_IS[UG]ID, S_ISVTX, S_I[RWX](USR|GRP|OTH)
                    返回：若成功则为0，若出错则为－1

int       mkfifo(const char *pathname, mode_t mode);
                    <sys/types.h>
                    <sys/stat.h>
                    mode: S_IS[UG]ID, S_ISVTX, S_I[RWX](USR|GRP|OTH)
                    返回：若成功则为0，若出错则为－1

time_t    mktime(struct tm *tmptr);
                    <time.h>
                    返回：若成功则为日历时间，若出错则为－1

caddr_t   mmap(caddr_t addr, size_t len, int prot, int flag, int filedes, off_t off);
                    <sys/types.h>
                    <sys/mman.h>
                    prot: PROT_READ, PROT_WRITE, PROT_EXEC, PROT_NONE
                    flag: MAP_FIXED, MAP_SHARED, MAP_PRIVATE
                    返回：若成功则为映射区的起始地址，若出错则为－1
```

```
int     msgctl(int msqid, int cmd, struct msqid_ds *buf);
                <sys/types.h>
                <sys/ipc.h>
                <sys/msg.h>
                cmd: IPC_STAT, IPC_SET, IPC_RMID
                返回：若成功则为0，若出错则为－1

int     msgget(key_t key, int flag);
                <sys/types.h>
                <sys/ipc.h>
                <sys/msg.h>
                flag: 0, IPC_CREAT, IPC EXCL
                返回：若成功则为消息队列I D 若出错则为－1

int     msgrcv(int msqid, void *ptr, size_t nbytes, long type, int flag);
                <sys/types.h>
                <sys/ipc.h>
                <sys/msg.h>
                flag: 0, IPC_NOWAIT, MSG_NOERROR
                返回：若成功则为消息数据部分的长度，若出错则为－1

int     msgsnd(int msqid, const void *ptr, size_t nbytes, int flag);
                <sys/types.h>
                <sys/ipc.h>
                <sys/msg.h>
                flag: 0, IPC_NOWAIT
                返回：若成功则为0，若出错则为－1

int     munmap(caddr_t addr, size_t len);
                <sys/types.h>
                <sys/mman.h>
                返回：若成功则为0，若出错则为－1

int     open(const char *pathname, int oflag, ... /* , mode_t mode */ );
                <sys/types.h>
                <sys/stat.h>
                <fcntl.h>
                oflag: O_RDONLY, O_WRONLY, O_RDWR;
                       O_APPEND, O_CREAT, O_EXCL, O_TRUNC,
                       O_NOCTTY, O_NONBLOCK, O_SYNC
                mode: S_IS[UG]ID, S_ISVTX, S_I[RWX](USR|GRP|OTH)
                返回：若成功则为文件描述符，若出错则为－1

DIR    *opendir(const char *pathname);
                <sys/types.h>
                <dirent.h>
                返回：若成功则为指针，若出错则为NULL

void    openlog(char *ident, int option, int facility);
                <syslog.h>
                option: LOG_CONS, LOG_NDELAY, LOG_PERROR, LOG_PID
                facility: LOG_AUTH, LOG_CRON, LOG_DAEMON, LOG_KERN,
                          LOG_LOCAL[0-7], LOG_LPR, LOG_MAIL, LOG_NEWS,
                          LOG_SYSLOG, LOG_USER, LOG_UUCP

long    pathconf(const char *pathname, int name);
                <unistd.h>
                name: _PC_CHOWN_RESTRICTED, _PC_LINK_MAX, _PC_MAX_CANON,
                      _PC_MAX_INPUT, _PC_NAME_MAX, _PC_NO_TRUNC,
                      _PC_PATH_MAX, _PC_PIPE_BUF, _PC_VDISABLE
                返回：若成功则为相应值，若出错则为－1

int     pause(void);
                <unistd.h>
                返回：－1，errno设置为EINTR

int     pclose(FILE *fp);
                <stdio.h>
                返回：cmdstring的终止状态，若出错则为－1
```

```
void      perror(const char *msg);
                  <stdio.h>

int       pipe(int filedes[2]);
                  <unistd.h>
                  返回：若成功则为0，若出错则为－1

int       poll(struct pollfd fdarray[], unsigned long nfds, int timeout);
                  <stropts.h>
                  <poll.h>
                  返回：准备就绪的描述符数，超时返回0，若出错则返回－1

FILE     *popen(const char *cmdstring, const char *type);
                  <stdio.h>
                  type: "r", "w"
                  返回：若成功则为文件指针，若出错则为NULL

int       printf(const char *format, ...);
                  <stdio.h>
                  返回：若成功则为输出的字符数，若输出错则返回负值

void      psignal(int signo, const char *msg);
                  <signal.h>

int       putc(int c, FILE *fp);
                  <stdio.h>
                  返回：若成功则为c，若出错则为EOF

int       putchar(int c);
                  <stdio.h>
                  返回：若成功则为c，若出错则为EOF

int       putenv(const char *str);
                  <stdlib.h>
                  返回：若成功则为0，若出错则为非0

int       putmsg(int filedes, const struct strbuf *ctlptr, const struct strbuf *dataptr,
                  int flag);
                  <stropts.h>
                  flag: 0, RS_HIPRI
                  返回：若成功则为0，若出错则为－1

int       putpmsg(int filedes, const struct strbuf *ctlptr, const struct strbuf *dataptr,
                  int band, int flag);
                  <stropts.h>
                  flag: 0, MSG_HIPRI, MSG_BAND
                  返回：若成功则为0，若出错则为－1

int       puts(const char *str);
                  <stdio.h>
                  返回：若成功则为非负值，若出错则为EOF

int       raise(int signo);
                  <sys/types.h>
                  <signal.h>
                  返回：若成功则为0，若出错则为－1

ssize_t   read(int filedes, void *buff, size_t nbytes);
                  <unistd.h>
                  返回：若成功则为读到的字节数，若已至文件尾端则为0，若出错则为－

struct
dirent   *readdir(DIR *dp);
                  <sys/types.h>
                  <diret.h>
                  返回：若成功则为指针，若出错则为NULL

int       readlink(const char *pathname, char *buf, int bufsize);
                  <unistd.h>
                  返回：若成功则为读到的字节数，若出错则为－1
```

```
ssize_t  readv(int filedes, const struct iovec iov[], int iovcnt);
                    <sys/types.h>
                    <sys/uio.h>
                    返回：若成功则为读到的字节数，若出错则为 - 1

void    *realloc(void *ptr, size_t newsize);
                    <stdlib.h>
                    返回：若成功则为非空指针，若出错则为 NULL

int      remove(const char *pathname);
                    <stdio.h>
                    返回：若成功则为0，若出错则为－1

int      rename(const char *oldname, const char *newname);
                    <stdio.h>
                    返回：若成功则为0，若出错则为－1

void     rewind(FILE *fp);
                    <stdio.h>

void     rewinddir(DIR *dp);
                    <sys/types.h>
                    <dirent.h>

int      rmdir(const char *pathname);
                    <unistd.h>
                    返回：若成功则为0，若出错则为－1

int      scanf(const char *format, ...);
                    <stdio.h>
                    返回：赋值的输入项数，若在任一变换前输入出错或 EOF则返回EOF

int      select(int maxfdp1, fd_set *readfds, fd_set *writefds, fd_set *exceptfds,
              struct timeval *tvptr);
                    <sys/types.h>
                    <sys/time.h>
                    <unistd.h>
                    返回：准备就绪的描述符数，超时返回 0，若出错则为－1
                    FD_ZERO(fd_set *fdset);
                    FD_SET(int filedes, fd_set *fdset);
                    FD_CLR(int filedes, fd_set *fdset);
                    FD_ISSET(int filedes, fd_set *fdset);

int      semctl(int semid, int semnum, int cmd, union semun arg);
                    <sys/types.h>
                    <sys/ipc.h>
                    <sys/sem.h>
                    cmd: IPC_STAT, IPC_SET, IPC_RMID, GETPID, GETNCNT, GETZCNT,
                         GETVAL, SETVAL, GETALL, SETALL
                    返回：(与命令有关)

int      semget(key_t key, int nsems, int flag);
                    <sys/types.h>
                    <sys/ipc.h>
                    <sys/sem.h>
                    flag: 0, IPC_CREAT, IPC_EXCL
                    返回：若成功则为信号量I D 若出错则为－1

int      semop(int semid, struct sembuf semoparray[], size_t nops);
                    <sys/types.h>
                    <sys/ipc.h>
                    <sys/sem.h>
                    返回：若成功则为0，若出错则为－1

void     setbuf(FILE *fp, char *buf);
                    <stdio.h>

int      setegid(gid_t gid);
                    <sys/types.h>
                    <unistd.h>
                    返回：若成功则为0，若出错则为－1
```

```
int     setenv(const char *name, const char *value, int rewrite);
                <stdlib.h>
               返回：若成功则为0，若出错则为非0值

int     seteuid(uid_t uid);
                <sys/types.h>
                <unistd.h>
               返回：若成功则为0，若出错则为－1

int     setgid(gid_t gid);
                <sys/types.h>
                <unistd.h>
               返回：若成功则为0，若出错则为－1

void    setgrent(void);
                <sys/types.h>
                <grp.h>

int     setgroups(int ngroups, const gid_t grouplist[]);
                <sys/types.h>
                <unistd.h>
               返回：若成功则为0，若出错则为－1

int     setjmp(jmp_buf env);
                <setjmp.h>
               返回：若被直接调用则为0，若从longjmp返回则为非0值

int     setpgid(pid_t pid, pid_t pgid);
                <sys/types.h>
                <unistd.h>
               返回：若成功则为0，若出错则为－1

void    setpwent(void);
                <sys/types.h>
                <pwd.h>

int     setregid(gid_t rgid, gid_t egid);
                <sys/types.h>
                <unistd.h>
               返回：若成功则为0，若出错则为－1

int     setreuid(uid_t ruid, uid_t euid);
                <sys/types.h>
                <unistd.h>
               返回：若成功则为0，若出错则为－1

int     setrlimit(int resource, const struct rlimit *rlptr);
                <sys/time.h>
                <sys/resource.h>
               返回：若成功则为0，若出错则为非0值

pid_t   setsid(void);
                <sys/types.h>
                <unistd.h>
               返回：若成功则为进程组I D 若出错则为－1

int     setuid(uid_t uid);
                <sys/types.h>
                <unistd.h>
               返回：若成功则为0，若出错则为－1

int     setvbuf(FILE *fp, char *buf, int mode, size_t size);
                <stdio.h>
                mode: _IOFBF, _IOLBF, _IONBF
               返回：若成功则为0，若出错则为非0值

void   *shmat(int shmid, void *addr, int flag);
                <sys/types.h>
                <sys/ipc.h>
                <sys/shm.h>
                flag: 0, SHM_RND, SHM_RDONLY
               返回：若成功则为共享存储段指针，若出错则为－1
```

```
int     shmctl(int shmid, int cmd, struct shmid_ds *buf);
                    <sys/types.h>
                    <sys/ipc.h>
                    <sys/shm.h>
                    cmd: IPC_STAT, IPC_SET, IPC_RMID,
                         SHM_LOCK, SHM_UNLOCK
                    返回：若成功则为0，若出错则为－1

int     shmdt(void *addr);
                    <sys/types.h>
                    <sys/ipc.h>
                    <sys/shm.h>
                    返回：若成功则为0，若出错则为－1

int     shmget(key_t key, int size, int flag);
                    <sys/types.h>
                    <sys/ipc.h>
                    <sys/shm.h>
                    flag: 0, IPC_CREAT, IPC_EXCL
                    返回：若成功则为共享存储段ＩＤ若出错则为－1

int     sigaction(int signo, const struct sigaction *act, struct sigaction *oact);
                    <signal.h>
                    返回：若成功则为0，若出错则为－1

int     sigaddset(sigset_t *set, int signo);
                    <signal.h>
                    返回：若成功则为0，若出错则为－1

int     sigdelset(sigset_t *set, int signo);
                    <signal.h>
                    返回：若成功则为0，若出错则为－1

int     sigemptyset(sigset_t *set);
                    <signal.h>
                    返回：若成功则为0，若出错则为－1

int     sigfillset(sigset_t *set);
                    <signal.h>
                    返回：若成功则为0，若出错则为－1

int     sigismember(const sigset_t *set, int signo);
                    <signal.h>
                    返回：若真则为1，若假则为0

void    siglongjmp(sigjmp_buf env, int val);
                    <setjmp.h>
                    不返回

void  (*signal(int signo, void (*func)(int)))(int);
                    <signal.h>
                    返回：信号的以前配置，若出错则为SIG_ERR

int     sigpending(sigset_t *set);
                    <signal.h>
                    返回：若成功则为0，若出错则为－1

int     sigprocmask(int how, const sigset_t *set, sigset_t *oset);
                    <signal.h>
                    how: SIG_BLOCK, SIG_UNBLOCK, SIG_SETMASK
                    返回：若成功则为0，若出错则为－1

int     sigsetjmp(sigjmp_buf env, int savemask);
                    <setjmp.h>
                    返回：若直接调用则为0，若从siglongjmp调用返回则为非0

int     sigsuspend(const sigset_t *sigmask);
                    <signal.h>
                    返回：－1，errno设置为EINTR
```

```
unsigned
int       sleep(unsigned int seconds);
                    <unistd.h>
                    返回：0或未睡足的秒数

int       sprintf(char *buf, const char *format, ...);
                    <stdio.h>
                    返回：存入数组中的字符数

int       sscanf(const char *buf, const char *format, ...);
                    <stdio.h>
                    返回：赋值的输入项数，若在任一变换前输入出错或EOF则为EOF

int       stat(const char *pathname, struct stat *buf);
                    <sys/types.h>
                    <sys/stat.h>
                    返回：若成功则为0，若出错则为－1

char     *strerror(int errnum);
                    <string.h>
                    返回：消息字符串指针

size_t    strftime(char *buf, size_t maxsize, const char *format, const struct tm *tmptr);
                    <time.h>
                    返回：如有空间为存入数组的字符数，否则为0

int       symlink(const char *actualpath, const char *sympath);
                    <unistd.h>
                    返回：若成功则为0，若出错则为－1

void      sync(void);
                    <unistd.h>

long      sysconf(int name);
                    <unistd.h>
                    name: _SC_ARG_MAX, _SC_CHILD_MAX, _SC_CLK_TCK,
                          _SC_NGROUPS_MAX, _SC_OPEN_MAX, _SC_PASS_MAX,
                          _SC_STREAM_MAX, _SC_TZNAME_MAX, _SC_JOB_CONTROL,
                          _SC_SAVED_IDS, _SC_VERSION, _SC_XOPEN_VERSION
                    返回：若成功则为对应的值，若出错则为－1

void      syslog(int priority, char *format, ...);
                    <syslog.h>

int       system(const char *cmdstring);
                    <stdlib.h>
                    返回：shell的终止状态

int       tcdrain(int filedes);
                    <termios.h>
                    返回：若成功则为0，若出错则为－1

int       tcflow(int filedes, int action);
                    <termios.h>
                    action: TCOOFF, TCOON, TCIOFF, TCION
                    返回：若成功则为0，若出错则为－1

int       tcflush(int filedes, int queue);
                    <termios.h>
                    queue: TCIFLUSH, TCOFLUSH, TCIOFLUSH
                    返回：若成功则为0，若出错则为－1

int       tcgetattr(int filedes, struct termios *termptr);
                    <termios.h>
                    返回：若成功则为0，若出错则为－1

pid_t     tcgetpgrp(int filedes);
                    <sys/types.h>
                    <unistd.h>
                    返回：若成功则为前台进程组的进程组ID，若出错则为－1
```

```
int       tcsendbreak(int filedes, int duration);
                      <termios.h>
                      返回：若成功则为0，若出错则为－1

int       tcsetattr(int filedes, int opt, const struct termios *termptr);
                      <termios.h>
                      opt: TCSANOW, TCSADRAIN, TCSAFLUSH
                      返回：若成功则为0，若出错则为－1

int       tcsetpgrp(int filedes, pid_t pgrpid);
                      <sys/types.h>
                      <unistd.h>
                      返回：若成功则为0，若出错则为－1

char     *tempnam(const char *directory, const char *prefix);
                      <stdio.h>
                      返回：指向一唯一路径名的指针

time_t    time(time_t *calptr);
                      <time.h>
                      返回：若成功则为时间值，若出错则为 - 1

clock_t   times(struct tms *buf);
                      <sys/times.h>
                      返回：若成功则为过去的墙上时钟时间（单位：滴答），若出错则为－1

FILE     *tmpfile(void);
                      <stdio.h>
                      返回：若成功则为文件指针，若出错则为NULL

char     *tmpnam(char *ptr);
                      <stdio.h>
                      返回：指向唯一路径名的指针

int       truncate(const char *pathname, off_t length);
                      <sys/types.h>
                      <unistd.h>
                      返回：若成功则为0，若出错则为－1

char     *ttyname(int filedes);
                      <unistd.h>
                      返回：指向终端路径名的指针，若出错则为NULL

mode_t    umask(mode_t cmask);
                      <sys/types.h>
                      <sys/stat.h>
                      返回：以前的文件方式创建屏蔽

int       uname(struct utsname *name);
                      <sys/utsname.h>
                      返回：若成功则为非负值，若出错则为－1

int       ungetc(int c, FILE *fp);
                      <stdio.h>
                      返回：若成功则为c，若出错则为EOF

int       unlink(const char *pathname);
                      <unistd.h>
                      返回：若成功则为0，若出错则为－1

void      unsetenv(const char *name);
                      <stdlib.h>

int       utime(const char *pathname, const struct utimbuf *times);
                      <sys/types.h>
                      <utime.h>
                      返回：若成功则为0，若出错则为－1

int       vfprintf(FILE *fp, const char *format, va_list arg);
                      <stdarg.h>
                      <stdio.h>
                      返回：若成功则为输出字符数，若输出错则为负值
```

```
int      vprintf(const char *format, va_list arg);
                 <stdarg.h>
                 <stdio.h>
                 返回：若成功则为输出字符数，若输出错则为负值

int      vsprintf(char *buf, const char *format, va_list arg);
                 <stdarg.h>
                 <stdio.h>
                 返回：存入数组的字符数

pid_t    wait(int *statloc);
                 <sys/types.h>
                 <sys/wait.h>
                 返回：若成功则为进程ID 若出错则为－1

pid_t    wait3(int *statloc, int options, struct rusage *rusage);
                 <sys/types.h>
                 <sys/wait.h>
                 <sys/time.h>
                 <sys/resource.h>
                 options: 0, WNOHANG, WUNTRACED
                 返回：若成功则为进程ID 若出错则为－1

pid_t    wait4(pid_t pid, int *statloc, int options, struct rusage *rusage);
                 <sys/types.h>
                 <sys/wait.h>
                 <sys/time.h>
                 <sys/resource.h>
                 options: 0, WNOHANG, WUNTRACED
                 返回：若成功则为进程ID 若出错则为－1

pid_t    waitpid(pid_t pid, int *statloc, int options);
                 <sys/types.h>
                 <sys/wait.h>
                 options: 0, WNOHANG, WUNTRACED
                 返回：若成功则为进程ID 若出错则为－1

ssize_t  write(int filedes, const void *buff, size_t nbytes);
                 <unistd.h>
                 返回：若成功则为写的字节数，若出错则为－1

ssize_t  writev(int filedes, const struct iovec iov[], int iovcnt);
                 <sys/types.h>
                 <sys/uio.h>
                 返回：若成功则为写的字节数，若出错则为－1
```